ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АХБОРОТ ТЕХНОЛОГИЯЛАРИ ВА КОММУНИКАЦИЯЛАРИНИ РИВОЖЛАНТИРИШ ВАЗИРЛИГИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

Компьютер инжиниринги факультети

«Мультимедиа технологиялари» кафедраси

«Веб дастурлаш» фанидан маърузалар матни

Ташкент 2015

1-маъруза. Веб-дастурлаш фанига кириш

Режа:
1. Web-саҳифа, Web-сайт, Web-сервер.
2. Разметкали тиллар: HTML, XML, XHTML, WML.
3. Сценарийли тиллар. "клиент-сервер" технологияси

Калит сўзлар: Web-саҳифа, Web-сайт, Web-сервер, HTML, XML, XHTML, WML, клиент-сервер

web-саҳифа, web-сайт, web-сервер

Web-технологияни (Интернет-технология) ўрганишни Web-дизайннинг қуйидаги учта тушунчасини ўрганишдан бошлаймиз: Web-саҳифа, Web-сайт ва Web-сервер.

Web-саҳифа – ўзининг уникал адресига эга бўлган ва махсус кўриш дастури ёрдамида (браузер) кўрилувчи хужжатдир. Унга матн, графика, овоз, видео ёки анимация маълумотлар бирлашмаси - мультимедияли хужжатлар, бошқа хужжатларга гипермурожаатлар кириши мумкин.

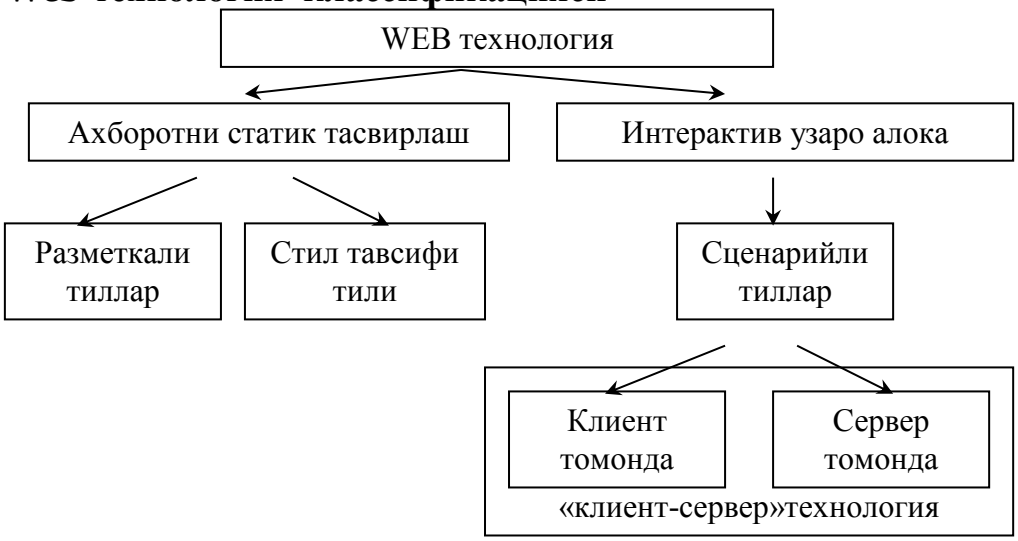
Web-сайт – бир қанча web-саҳифаларнинг мантиқий бирлашмаси.

Web-сервер – тармоққа уланган компьютер ёки ундаги дастур ҳисобланиб, умумий ресурсларни клиентга тақдим этиш ёки уларни бошқариш вазифаларини бажаради. Web-серверлар маълумотлар базалари ва мультимедияли маълумотларни бир бирига мослаштиради; Web-серверда Web-саҳифа ва Web-сайтлар сақланади.

Биз Интернет тармоғидаги Web-саҳифаларни кўришимиз учун WWW (World Wide Web) деб аталувчи сервисдан фойдаланамиз.

World Wide Web (WWW, Бутун дунё ўргемчак тўри) – бу клиент-сервер технологияси асосида ташкил этилган, кенг тарқалган Internet хизматидир.

Web-технология классификацияси



разметкали тиллар: HTML, XML, XHTML, WML.

Web-технологиянинг (Интернет-технология) Web-дизайн қисмини ўрганишни разметкали тил таснифи билан бошлаймиз.

Махсус тил мавжуд бўлиб, бу тил ёрдамида матнлар, график маълумотлар Web-саҳифа хужжатга жойлаштирилади ва бу хужжатни барча компьютерда кўриш имконияти мавжеддир. Бундай махсус тиллар разметкали тиллар деб аталади. Уларнинг асосий вазифаси – Web-саҳифага “маълумотларни жойлаштириш” ва улар орасидаги алоқани (гиперссилокалар) таъминлашдан иборат.

HTML (HyperText Markup Language)

Дастлаб World Wide Web тизими матнли маълумотларни ва HTML хужжатларни кўришга мўлжалланган, матнни тахрирловчи тилга ўхшаш тизим бўлган. Айни дамда HTML тили WWW дага энг оммабоп тиллардан бири ҳисобланади. HTML тилида ёзилган маълумотлар ўз ичига матн файллар, график маълумотлар ва бошқалврни олади.

Хужжатлар орасидаги алоқани таъминлаш ва маълумотларни форматлаш воситалари тэг (tag) деб аталувчи восита орқали амалга оширилади.

Web-саҳифанинг матн ва тэглари аралаш равишда HTML-хужжат деб аталувчи файлининг ичига жойлаштирилади. Қандай тэгни қўллаганингизга қараб брауэер ойнасида маълумотлар турлича кўринади. HTML хужжатга маълумотларни жойлаштириш ва таҳрирлаш учун юзлаб тэглар мавжуд. Масалан, ва тэглари абзацни ташкил этади, ва жуфт тэглари эса, матнни ёзма (курсив) ҳолда кўрсатиш учун қўлланилади. Шу билан бирга гиперматнли ссилкалар тэглари ҳам мавжуд. Ушбу элементлар фойдаланувчига гиперматн устига сичқонча курсори босилганда бошқа хужжатга боғланиш имконини беради.

XML (eXtensible Markup Language).

XML тили ҳам HTML тилига ўхшаш тил хисобланади. HTML дан фарқли томони шундаки, XML да дастурчи ўзининг шахсий тэгларини яратади ва улар орасига маълумотлар жойлаштиради. XML-тэглар ҳарфлар катта кичиклигини фарқлайди.

XHTML.

XHTML тили HTML ва XML тилларининг бирлашмасини ташкил этади. XHTML тилида ёзилган хужжатнинг ташқи кўриниши платформага боғлиқ (Windows, Mac ёки Unix) равишда ўзгариб кетмайди. Шунга қарамай XHTML таркибида HTML дискрипторлардан фойдаланилади.

Бугунги кунда мобил алоқа воситаларидан фойдаланувчилар учун янги тил ишлаб чиқилган бўлиб, у WML (Wireless Markup Language) деб аталади; CDF (Channel Definition Format) - Microsoft ишлаб чиққан браузерларда push-канал ҳосил қилишда қўлланилади;

сценарийли тиллар. "клиент-сервер" технологияси

Ҳозирда Web-саҳифанинг ривожланиши янада интерактив поғонасига чиққан. Web-сайтлар аста секинлик билан иловалар интерфейсига ўхшаб бормоқда. Буларнинг барчаси замонавий Web-дастурлаш технологияси ёрдамида амалга ошмоқда.

Web-дастурлаш технологияларини, дастурларини асосан иккита қисмга ажратиш мумкин: клиент томонидаги дастурларлаш (client-side) ва сервер томонидаги (server-side). Ушбу технологияларни тушуниш учун аввало бевосита "клиент-сервер" технологиясини тушуниш керак.

Web-саҳифанинг интерактив дастури сценарий деб аталади.

Бундай атама дастурнинг натижасига боғлиқ ҳолда вужудга келган. Унинг асосий вазифаси Web-саҳифасида фойдаланувчи ҳолатига, ҳаракатига «реакция» беришдир.

Шу тариқа сценарийлар клиент томонида бажарилувчи ва сервер томонида бажарилувчи сценарийларга бўлинади. Клиент томонида бажарилувчи сценарийлар броузер ёрдамида бажарилади. Сервер томонида бажарилувчи сценарийлар эса Web-сервер ёрдамида бажарилади.

Клиент томонидаги сценарийлар

Клиент томонидаги сценарийлар фойдаланувчи томонидан киритилаётган маълумотларни тўғрилигини серверга мурожаат қилмасдан текширади. Кўп ҳолларда бу сценарийлар JavaScript ва VBScript тилларида ёзилади.

JavaScript

JavaScript – бу тил Netscape ва Sun Microsystems томонидан яратилган бўлиб, Web-саҳифанинг фўнкционал имкониятларини орттириш мақсадида қўлланилади.

JavaScript ёрдамида одатда маълумотли ва мулоқот ойналарини чиқариш, анимацияларни кўрсатиш каби вазифаларни бажариш мумкин. Бундан ташқари, JavaScript-сценарий баъзан ўзи ишлаб турган браузер ва платформа типини аниқлаш мумкин. JavaScript-сценарийлар фойдаланувчи томонидан киритилаётган маълумотларни тўғрилигини текширишда ҳам қулай ҳисобланади.

VBScript

VBScript тили Microsoft корпорацияси томонидан яратилган бўлиб, Visual Basic тилининг бир қисми ҳисобланади. VBScript тили Internet Explorer ва Microsoft Internet Information Server (IIS) лар билан ишлашга мўлжалланган тилдир.

VBScript тилининг JavaScript тили билан умумий қисимлари бир нечта, жумладан у айнан Microsoft Internet Explorer билан ишлаш ва унинг қўлланиш соҳасини чеклай олиш имкониятига эга. VBScript интерпретаторли тил ҳисобланиб, Microsoft нинг Web-технологиялари билан ҳамкорликда ишлай олади, масалан ASP (Active Server Page) билан. Шунга қарамай VBScript клиент томонида ишловчи сценарий ҳисобланади, ASP эса сервер томонида ишлайди.

Сервер томонидаги сценарийлар

Сервер томонида бажарилиши керак бўлган сценарийлар одатда сайт папкасининг ичидаги махсус папкага жойлаштирилади. Фойдаланувчи сўровига асосан сервер бу сценарийни бажаради. Бажарилган сценарий натижаси web-серверга узатилади ва ундан сўнг клиентга узатилади. Сервер томонидаги сценарийларни ташкил этиш учун одатда Perl, ASP, PHP, JSP и SSI каби тил ва технологиялардан фойдаланилади.

Perl

Perl тили Web-иловалар яратишда энг оммабоп тиллардан бири ҳисобланади. Матнларни қидириш ва тахрирлаш, файллар билан қулай ишлай олиш қоидалари билан Perl тили Internet нинг асосий тилларидан бири бўлиб қолди. Perl – интерпретаторли тил ҳисобланади, шу боис унда яратилган сценарийлар ишлаши учун сервер компьютерда Perl-интерпретатор ўрнатилган бўлиши керак.

Бевосита Perl-коднинг интерпретация қилиниш жараёни унинг самарадорлигини пасайтиради. Бугунги кунда Perl нинг асосий ютуқларидан, унинг барча платформалар учун ишлай олиши ва унинг барча ресурслари бепул тарқатилаётганлигидир. Кўпгина Web-серверлар UNIX да ишлайди, Perl интерпретатор эса бу операцион тизимнинг бир қисми ҳисобланади.

ASP (Active Server Pages)

ASP-маълумотлар базалари ташкил этиш ва улар билан ишлаш вазифаларини бажаришда жуда мослашувчан, қулай воситадир. ASP воситалари сервер томонида ишлайди ва HTML-код ва сценарийлар каби фойлларни қайта ишлайди. ASP технологияси VBScript, Java ва JavaScript тилларини қўллаб қувватлайди. ASP-код ихтиёрий HTML-хужжатдан, шу билан бирга бошқа ASP-хужжатдан чавирилиши мумкин. ASP-код жойлаштирилган Web-саҳифалар файллари кенгайтмаси .asp бўлади.

ASP технология Windows NT ва Microsoft IIS Web-серверига мўлжалланган ҳисобланиб, имкониятлари ва самарадорлиги юқори бўлганлиги боис кўпгина компаниялар ўз воситаларига ASP ни қўллаб

қувватлаш имкониятларини киритмоқдалар. ASP-воситаларини ишлаб чиқиш бўйича йирик компания Chillsoft Лидер среди независимых производителей ASP-средств – компания Chillsoft UNIX нинг бир қанча тури ва турли Web-серверларди ASP ни қўллаш имкониятини киритган. Кўпгина HTML-мухаррирлар, масалан Adobe GoLive ҳам ASP ни қўллаб қувватлайди.

ASP технологияси бир нечта қулайликларни ўзида жамлаган: HTML-хужжатни динамик генерацилайди, формаларни қўллаб қувватлайди, маълумотлар базасига рухсатни ташкил этади ва у билан ишлай олади. ASP – дастурлаш тили ҳам, илова ҳам эмас, у интерактив Web-саҳифа ҳосил қилиш технологияси.

PHP

PHP – бу серверда қайта ишланувчи сценарийлар тилидир. ASP каби PHP кодлар ҳам бевосита HTML-хужжатни таркибига қўшилади. Ушбу тилнинг номи Personal Home Page Tools сўзларининг қисқартмасидан олинган. PHP да C ва Perl тилларида учраган бир қатор муаммолар ҳал этилган, бундан ташқари, PHP маълумотлар базаси билан ишлаш учун жуда қулай воситадир. Умуман олганда Perl, PHP – очиқ тизимли тиллар ҳисобланади ва уларни дастурчилар модернизациялаштираолади.

JSP

JSP (JavaServerPage) технологияси ўзининг функционал имкониятларига кўра ASP га ўхшашдир. Асосий фарқи шундаки, бунда VBScript ва JavaScript билан бирга Java тили ҳам қўлланила олади. Шунга қарамай JSP Java дан олдинроқ қўлланилган ва ушбу технология мукаммал Web-иловалар яратиш учун етарли имкониятга эга.

SSI

SSI (Server Side Include) воситаси дастлаб HTML-файлни дастлаб серверда қайта ишлайди ва ундан сўнг уни клиентга узатади. Дастлабки қайта ишлаш вақтида хужжатга динамик генерация қилинган маълумотлар қўшилади, масалан жорий вақт ҳақидаги маълумот. Умуман олганда SSI технологияси HTML-файлнинг таркибига қўшимча қўлланмалар қўшишга мўлжалланган, HTMLнинг қисми ҳисобланади.

Такрорлаш учун саволлар

1. Интернет технология деганда нимани тушунасиз?
2. Қандай разметкали тилларни биласиз?
3. Клиент-сервер технологияни қандай тушунасиз?

2-маъруза. WWW ривожланиш боскичлари.

Режа:

1. Word Wide Web
2. CERN вужудга келиши
3. ривожланиш боскичлари

Калит сўзлар: WWW, CERN, HTML, Xanadu, гипералока, UNIX, W3C Windows MS DOS.

Бутун дунё чўлғами (паутина) Word Wide Web (WWW) ёки (W3) 1989 йили пайдо бўлди. Унинг мохиятини Швейцариядаги CERN (The European Laboratory for partile physios – элементар заррачаларнинг Европа лабораторияси) деб номланган лабораториянинг бир гурух олимлари ишлаб чиқдилар. Уларнинг фикрича, хар хил электрон хужжатлар ўзаро алмашув пайтида истаган компьютерда бир хил кўринишга эга булиши керак. Табиийки, бундай хужжатлар билан ишлаш мухити этиб Интернет танланган. CERN глобал тармоқдаги энг гавжум жойлардан бири хисобланган. Бу муаммо билан Лаборатория хизматчиси физик Тим Бернерс-Ли шугулланди ва 1991 йил тугатди. CERN олимлари навбатдаги авлод HTML (Hyper text Markup Language) ва www ларнинг ривожланишини билиб берган www (w3 consortium) деб номланган Консорциум нинг юзага келишига сабабчи бўлдилар.

1960 йили америкалик олим Теодор Хольм Хельсон нинг шунга ўхшаш муаммо билан машғул булганини айтиб ўтиш зарур. У ўз олдига шундай мақсад қўйган эди: инсоният яратган хар хил қийматдаги матнли хужжатларни махсус компьютер тармоғига бирлаштириш ва уларни ўзаро мантиқан боғлаш. Бунда фойдаланувчи асосий ёки қўшимча ахборотли ихтиёрий хужжатнинг бир жойидан бошқасига ўтиш мумкин. 1965 йили Нельсон Т. Х. бундай матнли ахборотларни ташкил этиш услубини **гиперматн,** узининг амалга ошмаган лойихасини эса **Xanadu** деб номлади.

Ана уша Т. Нельсоннинг **Xanadu** даги гояси WWW нинг ривожига туртки бўлди.

Физик Тим Бернерс–Ли ўзининг яратган ўзаро боғланган платформали мустақил матнли хужжатларни ёзиш тилини HTML деб номлади. Бу хужжатлар ўзаро гиперссылкалар (гипералока) ёрдамида боғланади. Гиперссылка - бу интернет сахифасидаги бошқа объект билан богловчи ажратилган суз туркуми. Ахборотнинг турли таркибий кисмлари орасидаги алока. У WWW доирасидаги объектдан объектга утишни тахмиллайди. Гиперматнли хужжатлар билан танишиб чикиш учун Тим Бернерс – Ли Web – (шархловчи) деб ном олган программа ёзди.

1993 йили америкалик талаба Марк Андрессен Mosaic Web – шархловчи дастурни ёзди. Бу дастур биринчилfp катори график интерфейсга эга бўлади ва сичконча билан ишлай бошлайди. Mosaic ишлатиш учун кулай, UNIX, PC, ва Macintosh платформаларида ишлайди ва бепул таркатилади.

Бироқ вақт ўтгач тадқиқотчи Mosaic асосчи Silicon Graphics билан бирлашди. Улар хозирги кунда бошловчи броузер – Netscape ни яратдилар. Тахминан Web даги барча трафикларнинг 80% Netscape га тўғри келади. Хонадонлардаги компьютерларни Netscape билан текин юклаш мумкин.

Кейинроқ бозорда Microsoft компанияси махсулоти Internet Explorer номли янги броузер пайдо бўлди. У щам тезда интернет тармоғига киритила бошлади. Кайси бир жихатдан WWW нинг машхур бўлиб кетиши Microsoft Windows га ўхшаб кетади. Windows MS DOS матн барча вазифаларни қулай график интерфейс орқали бажаради. Худди шундай WWW нинг график мохияти Интернет ва электрон алоқа воситаларининг эътиборини жалб этди.

Келажакдаги WWW броузер ва компьютерларда ахборотларнинг ташқи кўриниши билан бошқариладиган, ишлатишда энг қулай тил HTML билан чамбарчас боғланади. Охирги йиллар мобайнида HTML да бир канча ўзгаришлар содир бўлди. 24-декабрь 1999 йил махсус нотижорат ташкилот WWW Consortium (W3C) томонидан қабул қилинган HTML файллари,

шахсан, аудио – видеоклиплар билан ишлашда, айниқса сахифаларни ўзаро боғлашда катта қулайлик туғдиради.

Такрорлаш учун саволлар

1. Word Wide Web деганда нимани тушунасиз?
2. объектдан объектга утиш нима?
3. WWW Consortiumни қандай тушунасиз?

3-маъруза. HTMLни белгилаш тили тўғрисида умумий тушунча HTML ҳужжатнинг структураси. Сарлавҳа элементлари.

Режа:

1. HTMLнинг асосий теглари.
2. Web – сахифанинг номатн элементлари.
1. Номерланган ва маркерланган рўйхатлар

Калит сўзлар: TXT, HTML, ADRESS**,** BODY BGCOLOR, Graphies Interchauqe FOP, Bit MaP бит матрицаси.

1. HTML нинг асосий теглари.

HTML хужжатлари – бу матнли файллар бўлиб, уларга белгилаш теглари деб номланган махсус кодлар киритилган. Бу теглар Web-браузерларга матн ва графикларни қандай қилиб шарщлаш ва акс эттириш лозимлигини кўрсатиб туради. HTML-файл – бу оддий матнли файл. Шунинг учун уни истаган матн редакторида, масалан MS Word ёки оддий «Блокнот» да яратиш мумкин. Щужжат яратилгач, уни матн форматида сақлаш керак. Лекин бу ишни бажаришда олдин унинг кенгайишини ўзгартириш, яъни .TXT ўрнига. HTML ёки .HTM ни қўйишни эсдан чиқармаслик керак. .HTML ва HTM кенгайиши HTML-файл учун стандарт щисобланади. Бундан ташқари, бу кенгайишлар компьютерга файлда матнлардан ташқари HTML кодлари щам мавжудлигини кўрсатиб туради

HTML тили щарфлар размерига бефарқдир, яъни бош ва кичик щарфлар бир хил қабул қилинади. Лекин тегларни ёзишда кўпинча бош щарфлардан фойдаланилади.

Web-сащифа кўриниши ва акс эттирилаётган ахборотнинг қанақалигидан қатъий назар, HTML ва WWW спецификациясига асосан щар бир Web-сащифада иштирок этиши зарур бўлган қуйидаги тўртта теглар мавжуд:

1. <HTML> Браузерча хужжат HTML тилида ёзилганлиги тўғрисида хабар беради.

2. <HEAD> HTML – щужжатнинг кириш ва бош қисмини белгилайди.

3. <BODY> Асосий матн ва ахборотни белгилайди.

4. <ADRESS> Бу Web-сащифа тўғрисида кўпроқ тўла-тўкис ахборот олиш учун керак бўладиган электрон почта адрессига эга.

Бу теглар Web-браузерга HTML – щужжатнинг щар хил қисмларини аниқлаш учун жуда зарурдир, лекин улар Web-сащифанинг ташқи

кўринишига тўғридан-тўғри таъсир этмайдилар. Улар HTML га киритилган навбатдаги янги маълумотлар уй сащифаларида тўғри шарщлаш, шу билан бирга барча Web-браузерларда бир хил кўринишга эга бўлиши учун жуда зарурдир. Масалан, сизнинг Web-серверингизда барча HTML-щужжатларни кўрадиган ва уларнинг рўйхатини тузадиган дастур ишга туширилган. У <HEAD> теглари ичида жойлашган матнларни кўради, холос (бу ерда щужжатлар номи щам жойлаштирилган бўлади). Шундай қилиб, агар уй сащифаларида <HEAD> ва </HEAD> теглари бўлмаса, у щолда у рўйхатга киритилмайди. Анчагина номи чиққан Web-серверлар - қидирув воситаларининг кўпчилиги мана шундай ишлайди. Улар ахборотларни <HEAD> тегларидан оладилар.

<HTML> ва </HTML> теглари.

Бу теглар броузерларга улар орасидаги матнни худди HTML матни каби шарщлаш (изощлаш) зарурлиги тўғрисида хабар беради, чунки HTML-щужжатлари фақат матнлидир. <HTML> тег эса файлнинг гипермати боғланиш тилида ёзилганлигини гапириб туради.

<HEAD> ва </ HEAD> теглари.

Улар Web-сащифалар номларини белгилайдилар. Бунинг учун <HEAD> ва </HEAD> теглар орасида Web-сащифа номи киритилади. ***.** Щар бир HTML – щужжат фақатгина битта номга эга бўлади. Сўнгра унинг олди ва орқа томонларини <TITLE> ва </TITLE> теглари билан белгиланг.

У, одатда браузер дарчаси сарлавщасида кўрсатилади. Контейнер <TITLE> тегини щужжат файлининг номи билан адаштирмаслик керак. Аксинча у файл номи ва манзилига бутунлай боғлиқ бўлмаган матн **сатридир.** Файл номи компьютернинг операцион тизими (ОТ) орқали қатъий равишда аниқланади. Шу билан бирга щужжатлар номи (тег <TITLE> билан бирга) ни щужжат ичидаги кўпинча <H> теглари билан жойлашадиган сарлавщалардан фарқлаш керак бўлади.

<BODY> ва </BODY> теглари.

<BODY> ва </BODY> теглари <HEAD> каби HTML – щужжатнинг махсус қисмларини белгилашда ишлатилади. <BODY> теглари эгаллаб олган матн щужжатнинг асосий қисми щисобланади. Матннинг катта қисми ва бошқа ахборотлар щам унинг таркибига киритилади.

<ADDRESS> ва </ADDRESS> теглари.

Бу теглар мазкур сащифага нисбатан кимдадир савол ёки фикр туғилиб қолган тақдирда кимга мурожаат қилиш кераклиги тўғрисидаги ахборотларни ўз ичига олади.

<ADDRESS> теглари бу ахборотларни асосий блокдан ажратиб олиш учун ишлатилади. Уй сащифасига бу тегларни киритиш учун қуйидаги қадамларни бажаринг:

1. <BODY> ва </BODY> теглари орасида исмингизни ва электрон почта адресини теринг.

2. Сўнгра исмингиз ва адресингизга <ADDRESS> тегини киритинг.

3. Исм(ном) ва адресдан сўнг ёпувчи </ADRESS> тегни киритинг.

Энди Web-сащифа (мисол тариқасида) ни кўриб чиқамиз:

```
<HTML>
<HEAD>
<TITLE> Web-сащифа мисоли </TITLE>
</HEAD>
<BODY>
<H1> бизнинг Web-сащифамиз </H1>
<P> бу Web-сащифа Web-дизайнер бўлиш мумкинлигини намойиш қилиш мақсадида яратилгандир. Бунинг учун Web-серверга созланиши қийин дастур талаб қилинмайди. Бунда сизнинг операцион тизимингиз муваффақият билан унинг ўрнини босаолади. <P>
</BODY>
</HTML>
```

Бу ерда терминология тўғрисида бироз олдиндан келишиб олишимиз керак. HTML – щужжатда хато бўлса, лекин барибир броузер томонидан чидаб бўларли даражада кўрсатилса, бундай щужжатни **яхши расмийлаштирилган** щужжат дейилади. Аксинча, расмийлаштиришда хатоси бўлмаган HTML – щужжат **стандарт** щужжат дейилади.

Сащифамиз кодига яна бир бор нигощ ташлаймиз. Барча HTML – щужжат жуфт теглар - <HTML> ва </HTML> ичида жойлашганлиги маълум бўлади. Бу стандарт HTML – щужжатларни расмийлаштиришнинг **биринчи қоидасидир. Иккинчи қоида** бўйича HTML – щужжат иккита бир-бирига тенг бўлмаган секцияга бўлинган бўлади.

Биринчи (кичик) секция – бу HTML-сарлавща. HTML-сарлавща жуфт теглар - <HELD> ва </HEAD> билан ажралиб туради. У браузер дарчасида акс этмайди, лекин браузер ўз эщтиёжлари учун фойдаланадиган хизмат ахборотларини ўз ичига олади.

Иккинчи (катта) секция – бу щужжат **жисми** деб аталадиган шахсий щужжат. Худди мана шу щужжат жисми браузер дарчасида акс эттирилади. Жисм жуфт теглар - <BODY> ва </BODY> билан ажралиб туради. Бу ердан стандарт HTML – щужжатларни расмийлаштиришнинг **иккинчи қоидаси** келиб чиқади: щар бир щужжатда HTML – сарлавща ва тест ссўлкалари бўлиши ва бу иккала секциялар тўғри расмийлаштирилган бўлишлари шарт.

<BODY> тегида матн ва фан ранги тўғрисидаги ахборот мавжуд бўлиши мумкин. Бунинг учун чап тег форматини озгина ўзгартириш керак бўлади. Масалан: <BODY BGCOLOR қ «FFFFOO» TEXT қ «OOOOOO»>.

Web – сахифанинг номатн элементлари

Матн рамкасига сиғмайдиган барча элементлар (график тасвирлар, овозлар, видеофилмлар ва қўшимча элементлар) Web – сахифанинг номатн

элементлари щисобланади. Улар матн щисобламайдилар, шунинг учун алощида файл кўринишида Web – сервер дискларида, сақланадилар ва браузер томонидан шахсий Web – сахифалардан алощида талаб қилинади.

Web – сахифа номатн элементлари билан танишишни график тасвирлардан бошлаймиз. Графикадан фойдаланиш имкониятларини ихтиёрий публикация хилида ва шу билан бирга Web – хужжатлар иловаларида бащолаш жуда қийин. Ақл билан танланган ва хужжатларга тўғри жойланган графика уни ташқаридан қараганда жозибали ва энг асосийси, хужжатнинг асосий ғояларидан бирини беради.

Расм ва графикалар WWW учун умрбод зарурдир. Бу экранда бир вақтда щам тасвирни, щам матнни кўришга имконият беридаган интернетнинг ягона воситаси щисобланади.

Бошқа томондан уй сахифангизни хакдан ташқарии катта микдордаги тасвирлар билан безаб ташламанг. Агар сизнинг уй сахифангиз тасвирлар ва пиктограммалар билан тўлиб кетган бўлса, матн бундай нохуш щолатда йўқолиб қолади ва ташриф буюрувчилар щам талвасага тушиб қоладилар. Бундан ташқарии сащифа жуда узоқ вақт ичида юкланади ва сизнинг барча урунишингиз бещуда кетади.

Графикадан фойдаланишда оралиқ (середина) га эътибор бериш зарур.

Аввало браузерлар томонидан қувватланадиган график файллар форматларига баъзи бир тушунчаларни берамиз.

Фойдаланиш мумкин бўлган барча браузерлар GIF (Graphies Interchauqe FOP мат – графика алмашиш формати) ва JPEG (Joint Picture Encodinq Group харакатсиз тасвирларнинг кодлаштириш гурущи) график форматларни қувватлаб турадилар. Булар Интернетдаги графиканинг оммалаштириш стандарт форматидир.

Форматларни кодлаштиришда тасвир сиқилади ва натижада жуда кичик размерга эга бўлади (сифати қониқарли даражада бўлишигп қарамай). Барча замонавий график дастурлар бу иккала форматни қувватлаб турадилар. GIF штрихли тасвирлар (штрихли расмлар, схемалар,) Web – сахифанинг расмийлаштириш график элементлари) учун идеал щисобланса, JPEG эса, одатда ярим товушли (фотография, картина) графикани кодлашлаштиради. Бундан ўлароқ GIF – файл экранларда анимацион фильмлар сингари кўрсатиладиган бир нечта график тасвирларни щам ўз ичига олади.

Internet Explorer щам PNG (portable Network Graphics – ўзгарувчан тармоқ графикаси) ва BMP (Bit MaP бит матрицаси) форматларни қувватлаб туради. PNG яқинда ишлаб чиқилган бўлиб, бундан мақсад, GIF ва JPEG лар ўрнини эгаллаш ва иккала форматнинг ижобий томонларини бирлаштириш. Лекин щали у оммалашган эмас.

BMP – бу Windows системасида тасвирларни сақловчи стандарт форматдир. Бу формат истеъмол учун тавсия қилиниши мумкин эмас, чунки у берилган маълумотни сиқиб қўйишга қарши.

Щар хил график форматлар файлининг кенгайишини эсга олинг. GIF – файлнинг кенгайиши qif, JPEF – файларники jpq, jpe ва jpeq, PNG – файлларнинг кенгайиши- pnq, BMP – файиллар – вжр.

График тасвирлар Web – сахифага якка <IMG> теги ёрдамида қўйилади. қуйида уни формати келтирилган:

<IMG SRG қ “{тасвир файли адреси}” [WIDTHқ “{эни}”]

[HEIGHTқ “{Баландлик}”] [ACTқ “{Альтернатив матн}”]

[BORDER қ “{Чегара қалинлиги}”]

[ALIGN қ “left / riqht /top /texttop/ middle /alsmiddle/ /baseline/ botton/ absbotton”]

[VSPACE қ “{Вертикал бўйича сахифа матнигача масофа}”]

[HSPACE қ “ {Горизонтал бўйича сахифа матнигача масофа}”]>

Кўриниб турибдики, бу тегда жуда кўп атрибутлар бор экан. Бу IMG (Image) сўзи тасвирни изощлайди. Атрибутлар ичида SRG (sonrcc - манбаъ) мажбурий атрибут щисобланди. Бу адрес ўзига Web – сервер адресини қабул қилаоладиган (агар файл бошқа серверда бўлса) тўлиқ ва қисқартирилган (фақат файл номларидан иборат) кўринишида бўлиши мумкин.

<IMG SRC қ www.tsue.uz /sample /firstl.qif >

<IMG SRC қ “/folder 1/folder 2/first2.qif “>

WIDTH ва HEIGHT атрибутлари расмлар эни ва баландлигини пикселларда беришга имкон беради. Бу щолда браузер расм файлини олгунча кадар унинг щақиқий размерини ўрнатади ва натижада сизнинг сахифангиз дизайнига салбий таъсир бўлмайди.

ALT – атрибути «альтернатив матн» деб аталадиган график образ пайдо бўлиши лозим бўлган матн сатрини беради. Бундан мақсад шуки, сащифанинг керакли жойини тезроқ аниқлаш учун, фойдаланувчи браузердаги график тасвир кўрсатувчини ўчириб қўйиши мумкин. Натижада сахифадаги графика ўрнида бўш жой акс этади. Шунинг учун бу бўш жойдан унумли фойдаланиш йўлларини ўйлаб қўйиш керак бўлади.

Border атрибути тасвир атрофидаги (калинлигини берадиган пикселларда).

ALIGN атрибути тасвирнинг нисбий эгаллаб турган жойини бошқариш имкониятини яратиб беради. У қуйидаги ащамиятга эга:

- left – тасвир чапга сурилади, матн эса унинг ўнг томонидан айланиб ўтади.
- riqht - тасвир ўнгга сурилади, матн эса унинг чап томонидан ўтади.
- top - тасвир жорий сатрнинг юқорисига тўғриланади.
- texttop - тасвир жорий сатрнинг энг баланд символ чўққисига тўғриланади.
- middle - тасвир маркази жорий сатрнинг базавий чизиғига тўғриланади.
- abcmiddle - тасвир маркази жорий сатр марказига аниқ тўғриланади.
- baseline - тасвирнинг пастки қирраси жорий сатр базарий чизиғига тўғриланади.
- botton – тасвирнинг пастки қирраси жорий сатр қуйи қисмига тўғрилади.

- absbotton – тасвирнинг қуйи қирраси жорий сатр энг қуйи қисмидаги символнинг қуйи қисмига тўғрилади.

қуйида биз ALIGN атрибутининг хархил қийматларидан фойдаланишга мисол келтирамиз:

Щозир биз охирги иккита атрибутлар HSPACE ва VSPACE ни кўриб чиқамиз. Бу атрибутлар тасвир ва унинг сатри матни ўртасидаги оралиқни беради. Оралиқ қиммати пикселларда берилади.

3. Номерланган ва маркерланган рўйхатлар

Бугун рўйхатларни щар бир Web – сахифада кўриш мумкин. HTML тилида ахборотларни рўйхат кўринишида ифодалаш учун махсус теглар мўлжалланган. Рўйхатлар маълумотларни электрон ва нашр хужжатларда ифодалашнинг энг кўп қўлланиладиган формалардан бири щисобланади.

Рўйхатлардан фойдаланиш қуйидаги холларда жуда қулайдир:

- Ўқиш пайтида қулайлик туғдириш учун ахборот фрагметнларини ягона структурага бирлаштириш;
- Мураккаб қидирув жараёнларни ёзиб чиқиш;
- Пунктлари (жойлари) мос хужжат бўлишларини кўрсатиб турувчи ахборот мундарижасининг жойлашуви;

HTML тилида рўйхатларнинг қуйидаги асосий хиллари мавжуд маркерланган, номерланган ва аниқлаш рўйхати. Яна бошқа хиллар щам бор, лекин юқоридаги учта рўйхат турлари энг кўп ишлатилади. Уларнинг ўзаро ўхшашлиги шундаки, уларнинг барчаси янги сатрнинг щар бир пунктига жойлашган бўлади ва сиз бемалол, щар бир матнни ажратишингиз мумкин. Юқорида келтирилган рўйхат типларининг асосий фарқи номерлаштириш усулида ва унинг структурасидадир.

Маркерланган рўйхат

Маркерланган (акс щолда бу каби рўйхатлар номерланган ёки тартибга келитирлмаган рўйхатлар дейилади) рўйхатни яратиш учун ичида рўйхатнинг щамма элементлари мавжуд тег – контейнер <UL> </UL> (UL – Unordered List – тартибга солинмаган рўйхат) дан фойдаланиш зарурдир. Рўйхатнинг очувчи ва ёнувчи теглари сатрни рўйхат бошига ва охирига ўтказишни таъминлаб туради. Шу билан бирга рўйхатни хужжатдаги асосий маълумотлардан ажратиб туради. Шунинг учун бу ерда <p > абзац тегини қўллаш ёки
 сатрини мажбурий ўтказиш зарурияти йўқ.

Рўйхатнинг щар бир элементи тег <LI> (/LI – List Item - рўйхати элементи) билан бошланиши керак. Тег <LI> мос ёпувчи тега мухтож эмас, лекин унинг мавжудлиги хато щисобланмайди. Браузерлар хужжатни акс эттиришда щарбир янги рўйхат элементини янги сатрдан бошлайдилар.

<LI> теги билан белгиланган рўйхат элементларидан ташқарии бошқа HTML – элементлари щам иштирок этиш мумкинлигини айтиб ўтиш зарурдир. Юқоридаги мисолда худи шундай элементлардан бири рўйхат пункти (жойи) щисобланмайдиган, лекин унинг сарлавхасида асосий ролни ўйнайдиган оддий матн кўрсатилган.

HTML тили бўйича баъзи дарсликларда рўйхат сарлавщаси учун тег – контейнер <LN>(LN – List Ytader рўйхат сарлавхасини ишлаб чиқиш кераклиги тўғрисидаги кўрсатмалар учраб туради. Щозирги пайтда номи чиққан браузерлардан бирортаси щам бу тегни тан олмайдилар ва у HTML спецификасига кирмайди. Шундай қилиб бу тег бита яримта хатоликларни келтириб чиқармаса щам, уни қўллаш щеч қанақа маъно касб этмайдилар.

<UL> тегида иккита параметр кўрсатилиши мумкин COMPACT ва TYPE.

COMPACT параметри маъносиз ёзилади ва мазкур рўйхатни компакт кўринишида чиқариш зарурлигини браузерга бириктириб қўйиш учун ишлатилади. Масалан, шрифтни ёки рўйхат сатрлари орасидаги масофани кичрайтириш мумкин ва б.к.

TYPE параметри рўйхат сатри белгиланадиган значокни бериш (юклаш) имконини яратади; учтасидан биттасига рухсат берилган: disc, circle ва square. Бу параметр рўйхат маркерлар хилини мажбурий топшириш учун ишлатилади. Конкрет меркер хили фойдаланиладиган браузерга боғлиқ бўлади. Акс эттиришнинг типик вариантлари қуйидагича:

TYPE қ disc – маркерлар бўялган айланалар билан акс эттирилади;

TYPE қ circle – маркерлар бўялмаган айланалар билан акс эттирилади.

TYPE қ square – маркерлар бўялган квадратчалар билан акс эттирилади.

Мисол: <UL TYPE қ circle > .

TYPE қ disc индамаслик (умолчание) бўйича фойдаланиладиган значение (маъно, мохият, қиймат) щисобланади. Киритилган маркерланган рўйхатлар учун индамаслик бўйича берилган даражада disc значенияси ишлатилса, иккинчида circle, учинчи ва ундан кейинги даражаларда square ишлатилади.

Масалан, HTML 4.0 спецификациясида TYPE қ square значениясида акс этадиган маркер хили учун бўялмаган квадратча кўрсатилади (square outtine).

Параметр TYPE шу значениялари билан алощида рўйхат элементлари маркерлари хилларининг кўрсатмаси учун қўлланиши мумкин. Бунинг учун параметр хилларининг кўрсатмаси учун қўлланиши мумкин. Бунинг учун параметр TYPE ўзига мос келган значениялари билан <LI> элементлари рўйхати тег ида кўрсатма беришига рухсат этилган.

Мисол: <LI TYPE қ circle >

График рўйхат маркерлари

Рўйхат маркерлари сифатида график тасвирлардан фойдаланиш мумкин, чунки улар чиройли, ўзига жалб қилувчи даражасида расмийлаштирилган HTML хужжатларни яратишда кўплаб ишлатилади.

Щақиқатан, бундай имконият тўғри – тўғри HTML томонидан тақдим этилмайди, лекин бироз суний равишда амалга оширилади. Мақсад шуки, <UL> рўйхат теги (қуйида кўриладиган барча теглар сингари) ягона топшириқни бажаради, яъни берилган тегдан кейин жойлашган барча ахборотлар қандайдир катталикка ўнг томонга силжиш билан акс этиши зарурлиги тўғрисида браузерга кўрсатма беради. Алощида рўйхат элементларига кўрсатма берувчи <LI> теглари стандарт рўйхат элементлари маркерларини олиб кетишни таъминлайди.

Агар график маркерлар билан рўйхат тузиш талаб қилинса, у щолда, умуман билан <LI> тегларисиз ишни щал қилиш мумкин. Фақатгина щар бир рўйхат элементи олдига қўйилган график тасвирни жойлаштириш керак бўлади. Бунда щал қилиниши лозим бўлган ягона масала бу рўйхат элементларини бир-биридан ажратишдир. Бунинг учун <p > абзац тегларидан ёки
 сатрини мажбурий ўтказишдан фойдаланса бўлади.

Мазкур мисолда рўйхат элементи маркери сифатида Tips.qit график файли ишлатилди. HTML – сащифаларда графикларни қўллаш узатилган ахборотлар хажмини анча ошиб кетишига олиб келишини айтиб ўтиш мақсадга мувофиқдир. Лекин бу щолатда хажм кенгайишини назарга олмаса щам бўлади, чунки у жуда кам ўзгарган. Бу ерда барча маркерлар учун фақат бир марта бериладиган файл ишлатилади. Кичик тасвирли файл размерга щам жуда кичик бўлади.

Аниқлаш рўйхати

Аниқлаш рўйхатлари қайдайдир терминлар рўйхатини ва уларнинг таърифини яратиш учун жуда мос тушади. Идеал щолатда аниқлаш рўйхатидан фойдаланиш – бу словарь. HTML ёрдами билан сиз щарбир термин ва унинг аниқланишини осонгина топаоламиз.

Аниқлаш рўйхатлари тег кантейнер <DT> (Definition List) ёрдами билан берилади. Контейнер ички қисмида тег <DT> (Definition Term) билан аниқланадиган термин белгиланса тег <DD> (Definition Descrintion) билан эса абзац ўз аниқланиши билан белгиланади. <DT> ва <DD> теглари учун мос ёпувчи тегларни ёзмаса щам бўлади.

Умуман, аниқлаш рўйхатлари қуйидагича ёзилади:

<DL>
<DT> термин
<DD> термини аниқлаш
</DL>

<DL> тегида ишлатилиши бошқа рўйхатларга ўхшаш COMPACT параметри кўрсатилиши мумкин.

Такрорлаш учун саволлар

1. ADRESS теги хақида нимани тушунасиз?

2. График рўйхат маркерларига нималар киради?
3. Аниқлаш рўйхати мисоллар келтиринг?

4-маъруза. HTML НИНГ АСОСИЙ ТЕГЛАРИ

Режа:

1. Ҳужжатнинг HEAD бўлими
2. Мантиқий ва физик форматлаш.
3. HTML-ҳужжатни форматлаш
4. HTML-ҳужжатнинг ичидаги сарлавҳа тэглари.

Калит сўзлар: HTML-ҳужжат, сарлавҳа, форматлаш теглари

HTML-ҳужжатини ёзишни бошлашда ишлатиладиган биринчи тэг бу <HTML> тэгидир. У ҳар доим ҳужжат ёзувининг бошида бўлиши лозим. Якунловчи тэг эса </HTML> шаклига эга бўлиши керак. Бу тэглар, улар орасида жойлашган ёзувнинг ҳаммаси бутун бир HTML-ҳужжатини англатиши билдиради. Аслида эса ҳужжат оддий матнли ASCII-файлидир. Бу тэгларсиз браузер ҳужжати форматини аниқлаб, таржима қила олмайди. Кўпинча бу тэг параметрга эга эмас. HTML 4.0 версиясига қадар VERSION параметри мавжуд эди. HTML 4.0да эса VERSION ўрнига <!DOSTYPE> параметри пайдо бўлди.

```
<HTML>
   <HEAD>
    *
    сарлавҳа қисми
    *
   </HEAD>
 <BODY>
  *
  тана қисми
  *
</BODY>
</HTML>
```

<HTML> ва </HTML> орасида 2 бўлимдан ташкил топиши мумкин бўлган ҳужжатнинг ўзи жойлашади. Мазкур ҳужжатнинг биринчи бўлими сарлавҳалар бўлими (<HEARD> ва </HEARD>), иккинчи бўлим эса ҳужжат тана қисмидир (<BODY> ва </BODY>), уни ҳужжат танаси ҳам деб юритамиз. Фрейм тузилиши ҳужжатлар учун <BODY> бўлимининг ўрнига <FRAMESET> бўлимидан фойдаланилади.

Ҳужжатнинг HEAD бўлими.

BODY бўлинмасида пайдо бўлиши мумкин бўлган баъзи HTML-тэглар блок даражасидаги (block level) тэглар деб аталса, бошқалари матн даражасидаги (text level) тэглари ёки кетма-кет тэг (inline) деб аталади. Блок даражасидаги тэглар ўзида матн даражасидаги тэглар ёки блок даражасидаги бошқа тэгларни мужассамлаштириши мумкин. Блок тэглари ҳужжат тизимини таърифлайди.

Мантиқий ва физик форматлаш

HTML-ҳужжатларда матнни форматлаш учун шартли равишда мантиқий ва физик форматлаш тэгларига тақсимласа бўлувчи тэглар яратилган. Мантиқий форматлаш тэглари фрагментнинг браузер ёрдамида экранда намойиш этилишига таъсир кўрсатмайдиган структуравий белгилашни амалга оширади. Шу сабабли бундай белгилаш мантиқий деб аталади. <CITE> тэги цитаталар ёки китоблар, мақолалар ва бошқа манбаларга ссылкаларнинг

номларини белгилашда фойдаланилади. Браузерлар бундай матнни курсив (қия) шаклда чиқариб беради.

Мисол:
<CITE> Даракчи </CITE> энг оммабоп газеталардан бири.
<EM> тэги – (Emphasis – ажратиб кўрсатиш, таъкидлаш) матннинг мухим фрагментларини ажратиб кўрсатишда фойдаланилади. Браузерлар одатда бундай матнни курсив шаклда акс эттиради.

Мисол:
Матннинг <EM> мухим сўзларини </EM> ажратиб кўрсатиш. Бу ерда мухим сўзларини деган ифода курсивда ажратиб кўрсатилади.

<DEL> тэги матнни учириб ташланган сифатида белгилайди. Бу тэг орқали белгиланган матннинг устига чизилган бўлади.

Мисол:
Бу <DEL> ўчирилган матндир </DEL> Бу ерда ушбу тэглар орасидаги «ўчирилган матндир» жумласи қуйидаги кўринишга эга бўлади: ўчирилган матндир (устига чиза олмадик)
Тэг CITE ва DATETIME параметрларига эга бўлиши мумкин. CITE - фрагемнтнинг ўчирилиб ташлаш сабабларини аниқлаштирувчи ҳужжатнинг URL-манзилини кўрсатади.
DATETIME – ўчириб ташланиш вақтини: YYYY-MM-DDThh:mm:ssTZD форматида кўрсатади. Бу формат ўчириб ташланиш йили, ойи, куни, соати, дақиқаси ва сонияларини, шунингдек соат тизими(TimeZone)ни аниқлайди.

<KBD> тэги матнни фойдаланувчи томонидан клавиатурада киритилганидек, яъни бир хил кенгликдаги (моноширинный) шрифтда акс эттиради. Мисол учун, матн мухарририни ишга тушириш учун <KBD> NOTEPAD </KBD> деб киритинг.

HTML-ҳужжатни форматлаш

Хар қандай матн маълум бир тузулишга эга бўлади. Одатда бундай тузулишнинг элементлари – сарлавҳалар, рўйхатлар, кичик сарлавҳалар, жадваллар, хатбоши ва бошқалар ташкил этади.

Хатбошиларга бўлиш. Хатбоши даражаси тэги.Одатда хатбошилар матнда фикр тугалланганлигини ифодалайди. Оддий матнли мухарирларда хатбоши бошқа қаторга ўтиш белгисини киритиш (<ENTER> тугмасини босиш) орқали тузилади. Аммо HTML-ҳужжатини тузуш жараёнида бошқа қаторга ўтиш белгилари хатбошининг ҳосил бўлишига олиб келмайди. Асл ҳужжатнинг бошқа қаторга суриш белгилари эътиборга олинмаганлиги сабабли, муаллиф ҳужжати ойнасида аъло даражада кўринган матн, браузер ойнасида умуман ўқиб бўлмайдиган даражада бўлиши мумкин. Шунинг учун

HTML-тилида матнларни хатбошиларга бўлиш учун <P> тэги киритилган. Бу тэгни хар бир хатбоши олдидан қўйиш керак. Ёпувчи </P> тэги бу ҳолда мажбурий эмас. Браузерлар бир неча кетма-кет жойлашган <P> тэгини бир тэгдек изоҳлайди. Одатда барузерлар хат бошиларни бир-биридан битта бўш қатор билан ажратади. <P> тэги атрибутлари:ALIGNГоризонтал текислаш атрибутининг қийматлари:LEFT, RIGHT, CENTER, JUSTIFY.

Мисол:

```
<HTML>
<HEAD>
<TITLE> Хатбоши тэгининг қўлланилиши </TITLE>
</HEAD>
<BODY>
<P> Бу энг оддий хатбоши бўлиб, кўп белгилардан иборатлигидан бир қанча қаторни эгаллайди </P>
<P ALIGN=CENTER> Бу матнли хатбоши ойна маркази бўйича текисланади, чунки  горизонтал текисланиш параметрига эга </P>
</BODY>
</HTML>
```


 тэги мажбурий бошқа қаторга суриш вазифасини бажаришга ҳизмат қилади (ёпувчи тэгга эга эмас).
 тэгининг киритилиши кейинги матн Янги қатордан бошланишини таъминлайди. Масалан, бундай усулдан маҳсус рўйхатни белгилаш тэгларини ишлатмаган ҳолда рўйхат шаклидаги тузилмаларни тузуш ёки шеърларни ёзиш каби ҳолларда фойдаланиш мумкин.

<P> тэги киритилучи хатбошини инкор этиши туфайли,
 тэги ёрдамида киритилувчи хатбошини моделлаштириш мумкин.

Масалан,

```
<HTML>
<HEAD>
<TITLE> бошқа қаторга сурушнинг қўлланилиши </TITLE>
</HEAD>
<BODY>
<P>Кераксиз сўз демагин
<BR> Кўпроқ сўзламоқ учун  <BR> Охир дакки емагин,  <BR> <BR> деб жавоб берди.
</P>
<CITE> Комил Парпиев </CITE>
</BODY>
</HTML>
```


 тэгининг параметрлари:CLEAR

Параметр маънолари: LEFT, RIGHT, ALL, NONE.
<NOBR> тэги бошқа қаторга ўтишни инкор қилиш учун қўлланилади. Бу тэг орқали белгиланган матн узунлигидан қатъий назар бир қаторда жойлаштирилади.

Жуда узун қаторлар ҳосил бўлишининг олдини олиш учун қатор суриш мумкин бўлган жой кўрсатилса, бу зарурият туғилганда амалга оширилади (“юмшоқ” тарзда бошқа қаторга суруш). Бу максадда <WBR> (Word Break) тэгидан фойдаланилади.

HTML-ҳужжатнинг ичидаги сарлавҳа тэглари

<H1 H2 …H6>.Ҳужжатнинг алоҳида бўлимларига сарлавҳа бериш учун 6 босқичли <H1> </H1>, <H2> </H2>, <H3> </H3>, <H4> </H4>, <H5> </H5>, <H6> </H6> теглардан фойдаланилади. 1 рақамли сарлавҳа энг йириги ҳисобланади. Энг кичиги эса 6 рақамли сарлавҳадир. Сарлавҳа тэги блок даражасидаги тэгдир, шунинг учун улар ёрдамида алоҳида матн сўзларининг ўлчамларини катталаштириш учун уларни белгилай олмайди. Сарлавҳа тэгларидан фойдаланилаётганда сарлавҳадан олдин ва кейин бўш қатор колдирилади, шунинг учун матн боши тэги ва бошқа қаторга суруш керак булмайди. Сарлавҳа тэглари атрибутлари: ALIGNТэглар параметри қийматлари:LEFT, RIGHT, CENTER, JUSTIFY (ўзгартирилмаган бўлса LEFT).

Мисол:
<HTML><HEAD><TITLE> Сарлавҳа мисоллари </TITLE></HEAD>
<BODY>
<H1> Сарлавҳа ўлчами 1 </H1>
<H2> Сарлавҳа ўлчами 2 </H2>
<H3 ALIGN=CENTER> Сарлавҳа ўлчами 3 </H3>
<H4 ALIGN=RIGHT> Сарлавҳа ўлчами 4 </H4>
<H5> Сарлавҳа ўлчами 5 </H5>
<H6> Сарлавҳа ўлчами 6 </H6>
Бу ерда ҳужжатнинг асосий матни
</BODY> </HTML>

<HR> тэги - горизонтал чизиқ. Саҳифанинг у ёки бу қисми тугалланганида визуал ажратиб кўрсатиш учун ишлатилади. Бу тэг ёпишни талаб қилмайди. Қатордан олдин ва кейин автоматик равишда бўш қатор қолдирилади.

<HR> тэги атрибутлари вазифаси:
ALIGN - LEFT, RIGHT, CENTER (ўзгартирилмаган х.)
WIDTH - чизиқ узунлиги фоиз кўрсаткичида ёки пикселда белгиланади.
SIZE - -чизиқ қалинлиги пикселларда ўрнатилади.
NOSHADE - -чизиқнинг рельефлилигини олиб ташлаш.

COLOR -- чизиқ рангги ўрнатилади.

Масалан:

```
<HTML><HEAD><TITLE> горизонтал чизиқ </TITLE></HEAD>
<BODY>
<P> Горизонтал чизиқ устига жойлаштирилган матн </P>
<HR ALIGN=CENTER WIDTH=70% NOSHADE>
<P> Кейин матннинг ўзи киритилади
</BODY>
</HTML>
```

<PRE> тэги олдиндан форматлаштирилган матннинг қўлланилишидир. Анъанавий усулда, бошқа қаторга суриш белгилари ёрдамида бажарилган форматланган матнни киритиш учун, керакли бўш жой миқдори, табуляция рамзлари ва бошқалар учун махсус <PRE> тэг контейнери мўлжалланган. <PRE> тэги билан белгиланган матн оддий матн мухарририда қандай кўринишга эга бўлса, худди шундай шаклда акс этади. Акс эттириш учун хар доим бир хил кенгликдаги шрифт қўлланилади. Бу тэг жадвални белгиловчи махсус тэгларни ишлатмасдан қурулган жадвалларда қўлланилади. Яна бир қўлланиш ҳолати катта блокларни чиқаришдир.

Мисол:

```
<HTML><HEAD><TITLE> форматланган матн </TITLE></HEAD>
<BODY>
<H3> Сонлар жадвали </H3>
<PRE>
0.5138707
0.1757256
0.3086337
0.858954
0.3896298
0.2777221
0.8229621
0.1519211
0.6254769
</PRE>
</BODY>
</HTML>
```

<PRE> тэгининг атрибути:

WIDTH - атрибут қиймати пиксел, ёки фоизларда берилиши мумкин ва форматланган матннинг максимал қатор узунлигини кўрсатади.

<BLOCKQUOTE> тэги асосий матндан цитаталарни ажратиб кўрсатиш учун қўлланилади. Бу тэг контейнер бўлиб, турли хил тэгларнинг форматларини аниқлайди.

<BLOCKQUOTE> тэги ёрдамида белгиланган матн акс этиш жараёнида асосий матндан бўш қаторлар билан ажратилади ва ўнг томонга кичик силжиш билан киритилади.

Мисол:

```
<HTML><HEAD>
<TITLE>Бошқа қаторга суришдан фойдаланишни цитаталаш блоки </TITLE></HEAD>
<BODY>
<P>
<I>
<B> Мисли чавандознинг </I> </B>қўлёзма вариантида айтилганки,
<BLOCKQUOTE>
Керакли иш билан шуғулланмоқ, яхши хулқлилик ва одоблилик - инсон зийнатидир  -  деди устоз
<P ALIGN=RIGHT> Ривоят </P>
</BLOCKQUOTE>
</BODY>
</HTML>
```

<ADDRESS> тэги ҳужжат муаллифини идентификация килиш ва муаллиф манзилини кўрсатиш учун қўлланилади. Одатда бу тэг ҳужжатнинг бошида ёки охирида жойлаштирилади. Бу тэгда тез-тез ҳужжат яратилган ва сўнгги маротаба янгиланган санаси кўрсатилади.
<ADDRESS> тэги контейнер ҳисобланади.

Мисол:

```
<HTML><HEAD> <TITLE> манзил блоки </TITLE></HEAD>
<BODY>
<P> мулоқот учун маълумот </P>
<ADDRESS> <P>Jukka Korpela, m.s. (Math) <BR> Helsinki University of Technology Computing Centre <BR>FIN – 02150 Espoo <BR> FINLAND</P>
<P> Telephone Internetional + 35894514319 </P>
<P> Electronic Mail: HYPERLINK "mailto:J.Korpela@hut.fi" ,J.Korpela@hut.fi
</P>
</ADDRESS>
</BODY>
</HTML>.
```

<! - - … - - > ҳужжатга шархлаш (коментарий) киритиш тэглари Шархлар қаторларнинг ихтиёрий сонидан ташкил топиши мумкин. Ва <!-- тэги билан бошланиб, --> тэги билан тугаши лозим. Тэг ичига киритилганларнинг барчаси саҳифани текшириш чоғида ишламайди. Одатда шархлар шахсий фойдаланиш учун мўлжалланган кузатишлар учун муаллиф томонидан ишлатилади.Шархларни ёзиш учун яна бир тэг-контейнер мавжуд бўлиб, у ҳам бўлса <COMMENT> дир.

Такрорлаш учун саволлар

1. HTML ҳужжатнинг умумий тузиши нимадан иборат?
2. Қандай форматлаш тегларини биласиз?
3. Комментарийлардан фойдаланишни изоҳланг.

5-маъруза. Гиперматнли ҳужжатларни, гипербоғланиш (ссылка) ва графикларни яратиш.

1. TARGET атрибутининг захирага олинган қийматлари
2. Ўзгарувчан» фреймлар
3. Формалар

Калит сўзлар: TARGET, FRAME, parent, IFRAME, E– mail, CHECKBOX, SUBMIT, RESET.

Бу ерда биз фреймларни қувватлаш учун мўлжалланган махсус <A> тегининг атрибутини кўриб чиқамиз. Бу атрибут – TARGET

TARGET қ «{фрейм номи}» / - self / - parent /top/ /-blank”

Маълум бўлди-ки, бу атрибут қиймати <FRAME> тегининг NAME атрибутида кўрсатилган фрейм номи бўлиши мумкин, худди шундай захирадаги қайси бир қиймат щам бўлиши эхтимолдан холи эмас. Биринчиси билан щаммаси оддий: Гипералоқанинг NREF атрибутида кўрсатилган адресдаги Web – сащифа кўрсатилган фреймда акс этади. Барча захирага олинган қийматлар жадвал Зда кўриб чиқилган.

TARGET атрибутининг захирага олинган қийматлари

қиймат	Таъриф
- self	Сахифа гипербоғланиш жойлашган фреймди акс этади.
- parent	Сахифа жорий фреймга нисбатан щисобланган дарчада акс этади. Бизнинг щолатда – браузер дарчасида – (фреймлар тўплами йўқ бўлган).
- top	Сахифа дарча ва фреймлар иерархасини юқори даражасидаги дарчада акс этади. Бизнинг холатда – браузер дарчасида (олдинги холат каби).
- blank	Сахифа браузернинг янги дарчасида акс этади.

«Ўзгарувчан» фреймлар

«Ўзгарувчан» фреймларни фақат Internet Explorer қувватлайди. «Ўзгарувчан» фрейм нима дегани? Оддий матнли Web – сащифага киритиш мумкин бўлган фреймни «Ўзгарувчан» фрейм дейилади. Бунда «Ўзгарувчан» фрейм теги жойлашган жойда, браузер кичкина дарчасида Интернет – адреси SRC атрибутида кўрсатилган Web – сахифани акс эттиради. «Ўзгарувчан» фреймлар жуфт теглар <IFRAME>… </ IFRAME> ёрдамида ўрнатилади.

<IFRAME> тегида одатий фреймлар <FRAME> таъриф тегидаги каби атрибутлар ишлатилади. Бундан ташқарии «ўзгарувчан» фреймнинг хужжатдаги размери ва жойлашувини амалга оширида қуйидаги атрибутлардан фойдаланиш мумкин: WIDTN, HEIGHT, HSPACE, VSPACE, ALIGN. Уларнинг вазифаси ва фойдаланиш тартиби ўрнатилган тавсифлар учун мос келган атрибутларга тўғри келади. Улар тег <IMG> билан берилади.

Намойиш учун қуйидаги мисолни кўриб чиқамиз:

1.<HTML>

2.<HEAD>

3<TITLE> Использование “плавающих” фреймов </TITLE> </HEAD>

4.<BODY>

5. CENTER><H1> Пример использования концепции “плавающих фреймов” </H1> </CTNTER>

<IFRAME SRCқfourth.html NAMEқ “F” HEIGHTқ250 WIDTHқ45% HSPACTқ1- SCROLLINGқYES ALIGNқRIGHT>

Сизнинг браузер «ўзгарувчан» фреймларнинг акс этишига имкон бермайди.

</ IFRAME>

«Ўзгарувчан» фреймларни биринчи ва ягона бўлиб қувватлаб турган браузер – бу Microsoft Internet Explorer браузеридир. Бундай фреймлар экраннинг истаган жойда, худди график тасвирлар ва жадваллар сингари, жойлаши мумкин.

</BODY>

</HTML>

«Ўзгарувчан» фреймлар концепциясини қувватламайдиган браузерлар мазкур мисолда fourth. html хужжат қийматларини акс эттириш ўрнига «Сизнинг браузер «ўзгарувчан» фреймни акс эттиришга имкон бермаяпти» деган матн чиқаради.

Таъкидлаб ўтиш зарур, яъни «ўзгарувчан»фреймлар кинцепцияси иделогия бўйича ўрнатилган тасвир ва жадвалларга жуда яқин. Бу ерда HTML – хужжатнинг керакли жойига бошқа HTML – хужжат бемалол жойлашади.

Формалар

HTML – формалар маълумотларни узоқдаги фойдаланувчидан Web – серверга щавола этиш учун мўлжалланган. Улар ёрдамида фойдаланувчи билан сервер ўртасида оддий мулоқат ташкил қилиш мумкин. Масалан, фойдаланувчини серверда қайд қилиш, тақдим қилинган рўйхатдан керакли хужжатни танлаб олиш ёки Web – сащифангиздан тўппа – тўғри электрон почтани жўнатиш. Бунинг учун браузерлар ўрнатилган функцияга эгадирлар. Сизга ягона битта иш қолади – бу HTML – файлнинг керакли жойига навбатдаги сатрни қўйишдир.

<AHREF қ mailto : Сизнинг – l – mail - адрес> браузер учун матн </A>

Сизнинг E– mail – адрес сўзи ўрнига хақиқий электрон почта адресини ёзинг. Браузер учун матн эса экранда ёруғ ранг билан ажралиб туради. Сичқонча курсори уни эгаллагач, у қўл кўринишида кўриниб, сичқончани чиқиллатишга таклиф қилгандек бўлади. Чиққиллатилгач, электрон почтанинг стандарт формаси юкланади.

Лекин, қаоро қабул учун мижоздан конкрет ахборот олиш зарур. Унга савол бериб ва жавоб учун жой ажратиб, бу ишни бажариш мумкин. Жавоблар эркин матн кўринишида щам, белгиналган квадрат кўринишида щам тақдим этилиши мумкин. Формалар жавоблар хилини щар хил кўринишда ифодалашга имкон беради. Энг асосийси - қўйилган саволга конкрет жавоб олишдир. Бундай системалаш келажакда олинган маълумотларни компьютер ёрдамида қайта ишлашни таъминлаш мумкин.

Демак, щар бир анкета <FORM> ва </FORM> теги билан рамкага солинади. Бу тег ичида иккита атрибут (параметр) мавжуд – METHOD ва ACTION. Биринчиси, берилган форма навбатдаги иш юритиш ишларига узатиш услубларини аниқлайди ва иккита қийматга эга бўлади: POST ва GET. ACTION атрибути эса анкета билан бундан кейин нима иш қилиш кераклигини аниқлайд ива кўп щолларда қуйидаги кўринишга эга бўлади:

<mailto : Сизнинг – E– mail - адрес>

Бунда сиз электрон почта орқали жўнатилган анкетани оласиз. Анкета алоқида файл сифатида келади ва сиздан қайси директорий ва қандай ном билан юкланишини сўрайди.

Форма <INPUT> ва <SELECT> теглари йиғиндисидан иборат. Улар мижоз томонидан киритилган ахборотнинг услубини аниқлайди. +уйидаги мисолда тег <INPUT> структурасини (мос атрибутлар билан) кўриб чиқамиз:

<INPUT TYPE “TEXT” NAME қ «Мижоз номи» SIZE қ “30”>

TYPE инструкциясининг TEXT қиймати киритиладиган ахборот эркин матн бўлишини, NAME инструкцияси эса киритилажак услуб номига тегишли эканлигини кўрсатади.

SIZE (размер) мижоз ўз номини кирита оладиган майдоннинг максимал узунлигини аниқлайди.

Навбатдаги жадвал жавобларнинг баъзи бир услубларини намойиши қилади:

INPUT қиймати	Мос услуб (стиль)
TEXT	Матнли майдон
RADIO	Селектор тугмаси
CHECKBOX	Назорат индикатори
SUBMIT	Анкетани жўнатиш тугмаси
RESET	Анкетани тозалаш тугмаси

Тег <SELECT> мижоз таклиф қилинаётган рўйхатдан сўзни (ёки сўзлар йиғиндиси) танлаш пайтидаги жавоб услубини аниқлайди.

Тег <TEXTAREA> форма ичида кўп сатрли матнни киритиш учун майдон яратади. Бу майдон браузер дарчасида горизонтал ва вертикал айлантириш тасмаси билан тўғрибурчак кўринишда акс этади.

Охирида форма мисолини кўриб чиқамиз:

Файлни fifth.html номи билан сақланг ва браузер ёрдами билан очинг.

Мавзу бўйича хулоса

HTML гиперматн белгилаш тили щисобланиб (дастурлаш тили эмас) унинг асосий қоидалари қуйидагича: щар бир харакат унинг боши ва охиридаги теглар билан аниқланади; теглар ва йўрикномалар ички буйруқ щисобланиб браузер дарчасида кўринмайди.

HTML теглари параметр ва атрибутларга эга. улар жуфт, баъзида ёлғиз кўрнишида бўладилар. Асосий теглар: <HTML>, <HEAD>, <BODY>, <ADRESS>.

Гиперматн хужжат, гиперщаволалар WWW тизимининг асосий тушунчаларидир. Гиперщаволанинг кўрсаткичлари сўз, сўз гурущи, тасвир бўлиши мумкин. Щаволалар абсолют ва нисбий бўладилар.

HTML тилида нумерланган ва маркерланган рўйхатларни ифодаловчи махсус теглар тўплами мавжуд. Бундан ташқарии аниқлаш рўйхат щам бор.

Браузерлар билан қувватланиб турувчи жадваллар кучли щисобланиб кенг кўламда ишлатилади. Фрейм ва формалар щам қўллашда анча кулайликка эгадирлар.

Такрорлаш учун саволлар

1. HTML (Huper Text Markup Langyage) тили?
2. HTML нинг асосий қоидалари?
3. HTML – файл?
4. HTML– файлнинг кенгайиши?

6-маъруза. HTML тилида матнлар билан ишлаш. Номерланган ва маркерланган рўйхатлар.

Режа:

1. Номерланган рўйхатлар
2. маркерланган рўйхатлар
3. Рангли матн жихозлаш имкониятлари

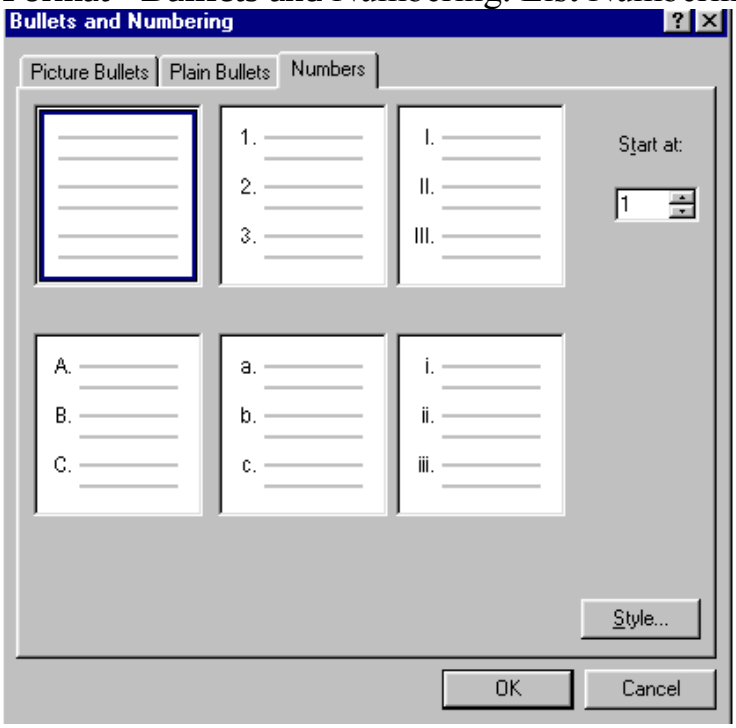
Калит сўзлар: List properties, Specufy pucture, Microsoft Frontpage, Align Roght, Formetting.

Номерланган рўйхатлар учун 1 услуб маркерлангани учун эса 5 услуб белгиланган. Сатр бошини номерлашнинг стандарт операцияси:

Formating ёки Format-Numbered List Up даги Numbering тугмаси.

Frontpage даги номерлаш хилларининг ўзгариши:

Format - Bullfets and Numbering. List Numbering дарчаси фаоллашади (расм 1)



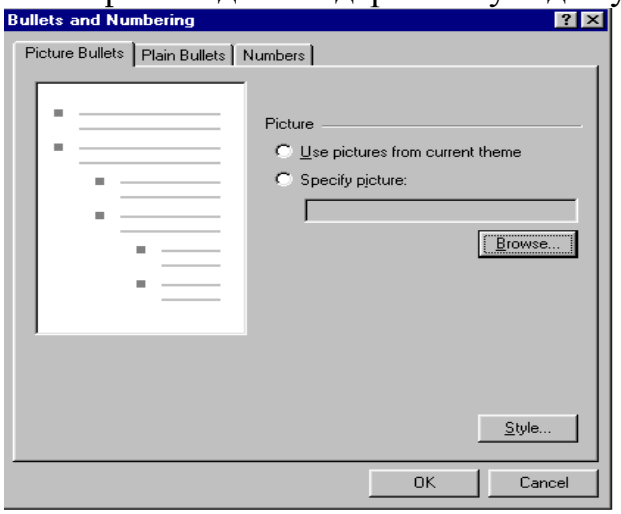
Расм 1 List Propertios савол-жавоб дарчасидаги Numbers қушимча варағи.

Индамаслик бўйича араб рақамли вариант танлаб олинади. Рим рақамли катта ва кичик харфли лотин алфавитини, кичик ва номерсиз рим рақамли вариантларни танлаш мумкин. Frantpage рўйхат номери **Starlat** (List properties дарчаси) майдонига ўрнатилади.

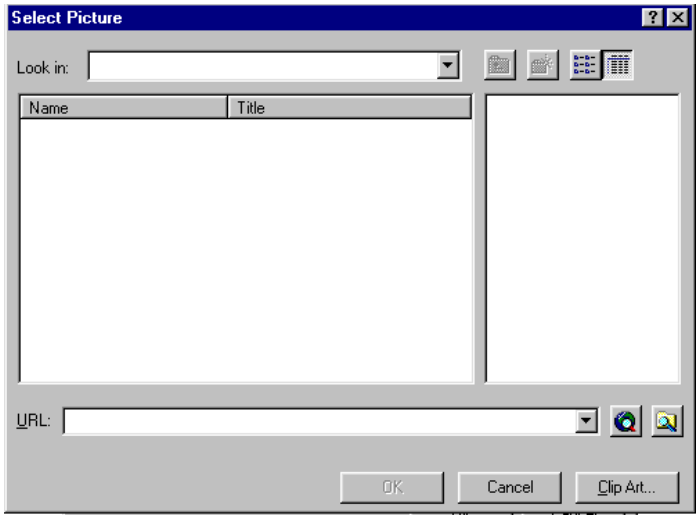
Маркерланган рўйхатларни яратиш учун **Bulleted List** услуби ёки **Formattins** даги Bullets тугмаси кўплаб ишлатилади. Маркер вариантлари хили(нуқта, айлана, квадрат кўринишларида) қўшимча варақа (**plain Bullets)**даги **List properties** савол-жавоб дарчасида танлаб олинади.

Маркерларни тасвирлаш учун что резервга олинган символлардан ташқари ихтиёрий график тасвирлардан фойдаланиш мумкин (List properties дарчаси - қўшимча варақдаги picture Bullets – ташқи кўринишли маркерни ўрнатиш учун). Индамаслик бўйича икки алтернативдаги **use piefures from curret Home** пункти харакатда бўлади. У кўпинча маркерларни тасвирлашда **сайт мавзуси**дан фойдаланиш кераклигини кўрсатади, яъни сайт таркибига кирувчи барча сахифаларни биринчи кўринишда жихозлашда график тасвирлар тўплами ва матнни жихозлаш қоидалардан фойдаланиш керак. Агар сахифаларни яратишда махсус маркер формалар ишлатилса, у холда график файлга олиб борувчи йўлни кўрсатиш билан **Specufy pucture** пункти танлаб олинади.

List properties диалог дарчасининг Picture Bullets кўшимча вараги ва Select picteru диалог дарчаси куйида кўрсатилган.



Расм 2 List Properties диалог дарчасининг Picture Bullets кўшимча вараги.



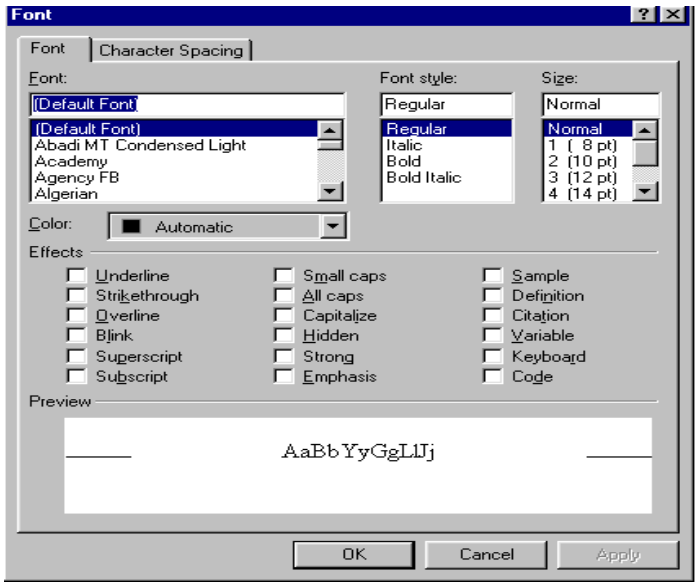
Расм 3 Select picteru диалог дарчаси.

Тавсия қилинаётган тасвирларни яратилажак сайт файлларини сақлаш учун мўлжалланган папка структурасидан танлаб олиш керак. Лекин, умуман олганда зарурий тасвирларни локал файл системасидан ёки сайт папкаси стурктурасида файлни кўрсатиш билан, ёки папка чегарадан ташқарида жойлашган тақдирда, локал компьютердаги хохлаган папка билан танишиб чиқиш имконини яратидиган папка ва ой тасвири туширилган тугмани босиш билан мақсадга эришиш мумкин. Иккинчи вариант – узоқдаги компьютердан тасвир танлаш (ихтиёрий тасвирни **Hofer hefa** даги танлаш учун системага ўрнатилган, ер шари ва ой тасвирли тугмани босиббраузер ишга туширилади). Учинчи вариант – тасвирни Clip Art дан танлаб олиш (Clip Art тугмаси). Бунда файл танланган тасвир билан бирга Microsoft Frontpage вақтинчалик папкасига ўтказилади.

Frontpage одатдаги маркерланган рўйхатдан ташқари жихозлашда оддий рўйхатдан фарқ қилмайдиган бир нечта рўйхатларни тақдим қила олади.

Direktory List ва **Menu List** услублари каскадли услуб жадваллари (CSS) ёрдами билан номерланмаган рўйхат элементларини тасвирлашнинг хархил вариантларини яратишга имкон беради. **Definition List** услуби аниқлаш рўйхати (ўқитиладиган материаллар рўйхатида бир нечта аниқловчи кўрсаткичлар бўлади).

Шрифт, унинг ташқи кўриниши, ўлчами ва матн жихозлаш услуби Font (Format-Font) диалог дарчаси ёрдами билан яратилиши мумкин. (расм 4)

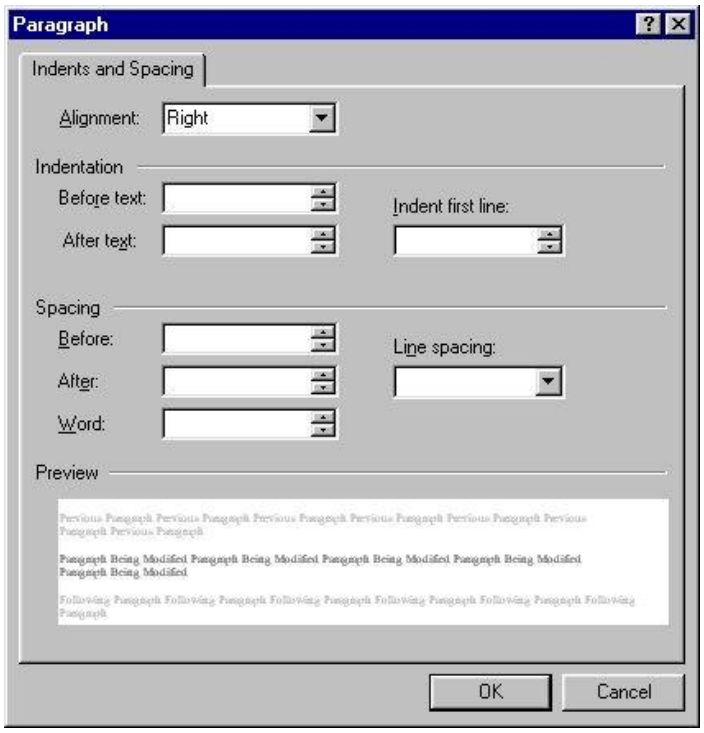


Расм 4 Font диалог дарчаси

Матнни жихозлаш учун хархил **тўғрилаш** (вўключка) ва **чекиниш** (отступ)лар хам ишлатилади. Тўғрилаш-матннинг горизонтал базовий сахифа линиясига нисбатан жойлашган, тўғрилаш учун **Formetting** инструментлар панелида (ИП) машхур тугмалар мавжуд:

Align List – чап қиррага сиқилган;
Align Center – марказга сиқилган;
Align Roght – ўнг қиррага сиқилган.

Матнни тўғрилаш одатда сўз бошида қўллаш мақсадида берилади. Тўғрилашнинг яна бир хили мавжуд бўлиб, бунда матн сахифа эни бўйича кенгайтирилган бўлади. Унинг учун махсус тугма йўқ. Format – Paragraph ни бажариш зарур хисобланади. (расм 5)



Расм 5 Paragraph диалог дарчаси.

Paragraph нинг фаоллашган дарчасида сўз боши параметрлари берилади(чекиниш, интервал). Матнни тўғрилаш **Aligument** рўйхатида кўрсатилади. Матнни сатр узунлиги бўйича чўзиш учун-**Sustify** пункти, Defanlt- тўғрилашни тўхтатиш учун ишлатилади. **Indentation** бошқариш органлар блоки матнни сахифа қирраларидан чекиниш кераклигини кўрсатиб туриш учун хизмат қилади.

Before text майдони – чап қиррадан чекиниш;
Affer text майдони – ўнг қиррадан чекиниш;
Indent fircf line - қизил сатрдан чекиниш.

Decrease Indent ва **Inerease Indent** тугмалари ёрдами билан матн чекинишини ўзгартириш мумкин.

Матнни позициялаш барча ностандарт символларни акс эттириш учун узлуксиз пробел (бўшлиқ) табуляция тугмаси ёрдами билан амалга ошириш мумкин, яъни сноскалар, амперсантлар ва қуштирноқлар.

Ва нихоят, хархил интерваллрни ўрнатиш учун бошқарув органлар блоки Spacing хизмат қилади:

After – сўз бошидан кейин; Word – алохида сўзлараро пробел (бўшлиқ); **Line Spacing** рўйхати – сатрлараро интервал (Single – одатдаги); Befure – сўз боши олдидаги интервал (1,5 lines – ярим қуюқ, Djuble - қўшалоқ).

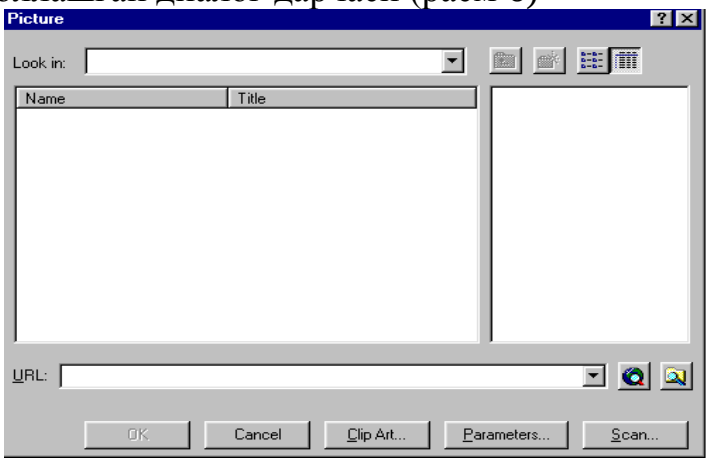
Рангли матн жихозлаш имкониятлари

Ранг матн фони учун ва шрифтнинг ўзи учун берилиши мумкин (тугмалар: **Highligt Color, Fant Color**). Хужжат бўлимларини ажратиш учун тез-тез горизонтал чизиқлар (Insert – horisontal Line) ишлатилади. Чизилажак чизиқ объект хисобланади ва унинг хусусиятини ўрнатиш ва тахрирлаш мумкин (Format – Propertios) ёки ажратилган чизиқда ўнг Click ни босиб **Horisontal Line Iroperfies** буйруғи чиқарилади (зарурий алохида чизиқ хусусиятлари (widch) ва ўлчов бирлигини аниқлаш учун). Height – пикселидаги қалинлик (индамаслик бўйича) Aligument - туғрилаш, **Solix Line (no spading)** переключатели билан чизиқ ясси кўринишда тасвирланади.

Web – сахифани жихозлашда графикадан фойдаланиш.

Frontpage қуйидаги графика хилларидан фойдаланиши мумкин:
- стандарт график файллар;
- Caplrt коллекциясидаги картинкалар;
- видеофрагментлар.

График файлларини ўрнатиш:
Insert – Picture – Frantpage ёки Up даги тугмалар
Picture нинг фаоллашган диалог дарчаси (расм 6)



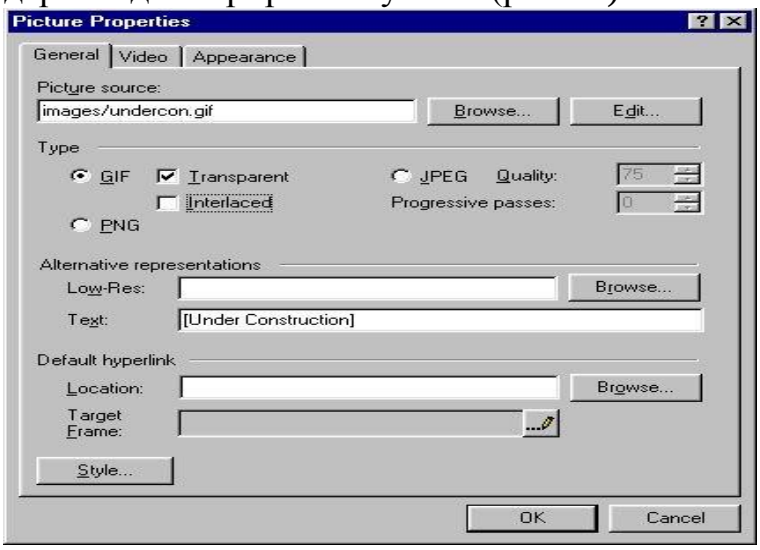
Расм 6 Picture нинг диалог дарчаси

Look майдонида тасвир манбаининг URLи кўрсатилади. URL (Universal Resunice Luckfor) - унинг жойлашган жойини кўрсатувчи ва уни қидириб топишга имкон яратувчи Internet даги конкрет ресурс рўйхат тузувчисидир . Унда протокол номи кўрсатилган бўлиб, унга асосан интернетга кириш (масалан HTTP), ресурс домен номлари (www. Amazor.com) ва ресурснинг ўзига йўл топган (|cutend| undex.htm) ишлари бажарилади. Принцип бўйича тасвир жойлашган жойига нисбатан чегаралаш мавжуд эмас, лекин Web-сайти ўрнатилган жойда тасвирли файлнинг бўлиши мақсадга мувофиқ хисобланади. Файл ўрнашган жой номи URL майдонида кўрсатилган бўлади. Ўнг тугма Internet ёки локал машинадаги сўралган тасвирни қидириб топишга имкон беради.

Clip Art тугмаси MS Office таркибидаги расмли диалог дарчани ишга туширади, **Scan** тугмаси билан диалог ойна чақирилади, ундаги **Source** тугмаси ёрдамида рўйхатдан мос перифирия қурилмаси танланади. Ундан эса тасвир олинади, Acquire тугмаси лоихаланаётган сахифага тасвир жўнатади.

Графикани ўрнатиш учун ўз компьютерида қандайдир график файл танлаб олинади ва яратилажак сахифага жойлаштирилади. Тасвир **Normal** варағида пайдо бўлади. Cliek расмда уни ажратиб кўрсатади, Frantpage эса ўз навбатида тасвирлар билан ишлашга мосланган яна битта инструментал панелни кўз билан кўриб ишлашга тайёрлайди.

График тасвирлар билан ишлашдаги кўплаб имкониятлар Up даги Puctures (view-Taolbars- Pittimes, ёки сахифада расм ажратиш) тугмаси ёрдамида амалга оширилади. Расмнинг оддий хусусиятларини **Picture Properties** (тасвирнинг ўзидаги ўнг Click – Picture Properties) диалог дарчасида тахрирлаш мумкин (расм 7)



Расм 7 Picture Properties нинг диалог дарчаси.

Picture source сатри график файлнинг ўрнашган жойини кўрсатади. **Bravse** тугмаси файл системасини кўриб чиқиш учун: Edit- Frantpage га ўрнатилган график мухарирни ишга тушириш учун. Номустақил радиотугма

ўчирувчи-ёқўвчилар(переключатель) график файл типини кўрсатади: GIF, IPEG ва PVG, GIF файлининг **Interlaud** хусусияти сатр орқали расм тасвирини кўрсатади (юклашга қараб аниқлик ошиб боради). **Transparent** хусусияти тиниқ, GIF-файллар учун мўлжалланган. Бу файлда Web-сахифа дизайнида самарали фойдаланадиган ранглардан бири тиниқ қилиб ўрнатилган. **IPEG** файллари учун сиқиқ параметрларни мўлжаллаш мумкин, чунки бу файлда озгина ахборот йўкотиш билан тасвирлар сиқиқ холда сақланади. PVG формат файллари сахифалари қандай бўлса шундайлигича ўрнатилади. Агар қандайдир сабаб билан расм узоқдаги фойдаланувчи браузердаги **Alternative renpesientftion** майдонларида тўғри акс эттирилмаган тақдирда алтернатив расм тақдимоти услуби қўлланади.

Такрорлаш учун саволлар:

1. Яратилган Web-сахифани сақлаш ва кўриб чиқиш. Назорат сўзлари (н/с):- My Webs папкаси, cmaqes, private.
2. Frantpage ни матнли жихозлаш. Н/с: шрифт, шрифт ташқи кўриниши, шрифт ўлчами.
3. Матнни услубий жихозлаш. Н/с: Normal, Formatted, Address, Heading.
4. Матнни Frontpage даги номерланган рўйхат кўринишда жихозлаш. Н/с: рўйхат вариантлари, вариантларни ўзгартириш, Starfat.

7-маъруза. Жадваллар билан ишлаш.

Режа:

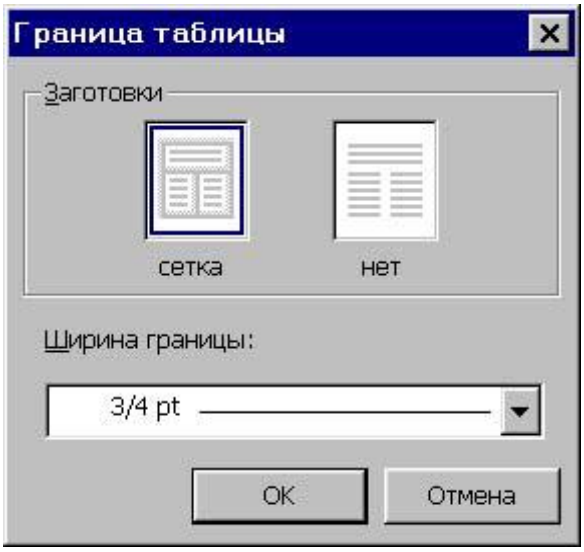
1. Web- сахифа билан иш режимида жадваллар яратиш ва тахрирлаш
2. Мультимедиа элементларини ўрнатиш.
3. Интернет – сахифани тахрирлаш
жараёнида кўриб чиқиш.

Калит сўзлар: TABLE, хавола, domain, URL, BGSOUND, CIF файл , Internet Assistant , Corel Draw Adobe photushop, Paintshop Pro, CIF Construction Set, Web page Wizord, HTML Hot Dog Pro тахрирлагичлар , Coffer Cup:

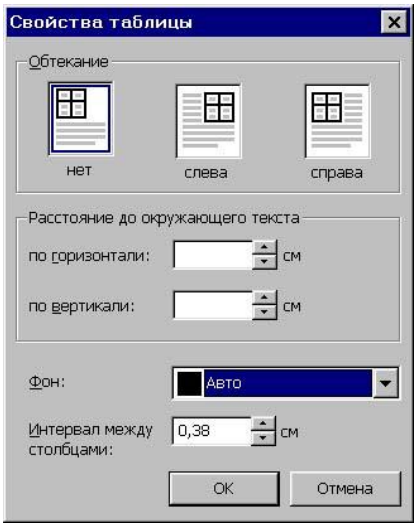
Web- сахифа билан иш режимида жадваллар яратиш ва тахрирлаш одатдаги Word хужжатлари билан ишлаш операцияларига жуда ўхшаб кетади. Бунда ячейкаларни устун ва сатр бўйича бирлаштириб анча мураккаб кўринишидаги жадвалларни тузиш мумкин, масалан:

<table>
<tr><td colspan="4">Жадвал мисоли</td></tr>
<tr><td rowspan="4">Бирлашган Ячейкалар билан</td><td colspan="2">Сатр бўйича</td><td></td></tr>
<tr><td></td><td>ва</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td colspan="2">Шу жумладан</td><td>Устун бўйича</td></tr>
<tr><td colspan="3">Тўла бўлмаган сатр билан</td><td></td></tr>
</table>

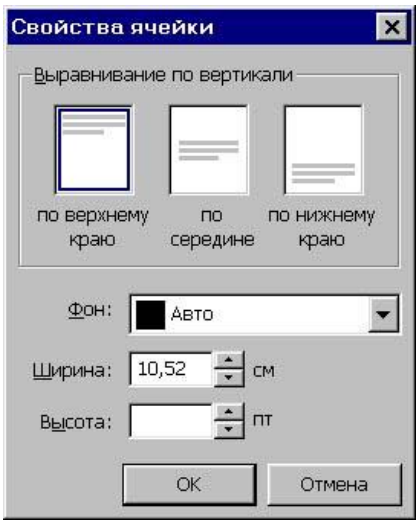
Жадваллар билан ишлаш – бу HTML- тахрирлагич сингари Word 97нинг энг сезиларли афзалликларидан бири щисобланади. Границў (расм.1.), Свойства таблицў (расм.2.) ва Свойства ячейки (расм.3) буйруқларидан фойдаланиб хохлаганча жадваллар ва ажратиб олинган ячейкалар, характеристикалар (разлиновка фон)ини ўзгартириши мумкин. Эътибор бериш керак-ки, кўринмас чизиқ чизилган (разлиновка) жадваллар-бу Интернет сахифаларга матн ва расмларни жойлаштиришда энг қулай усулдир. Масалан, матн-график сарлавхаларни яратиш, расмларга, фотоларга имзоларни чиройли қилиб жойлаш ва б.қ. Шу каби барча усуллар Word 97 туфайли қўл билан бесўнақай <TABLE>теги лойихаларини ёзиш зарурияти қолмайди.



Расм.1



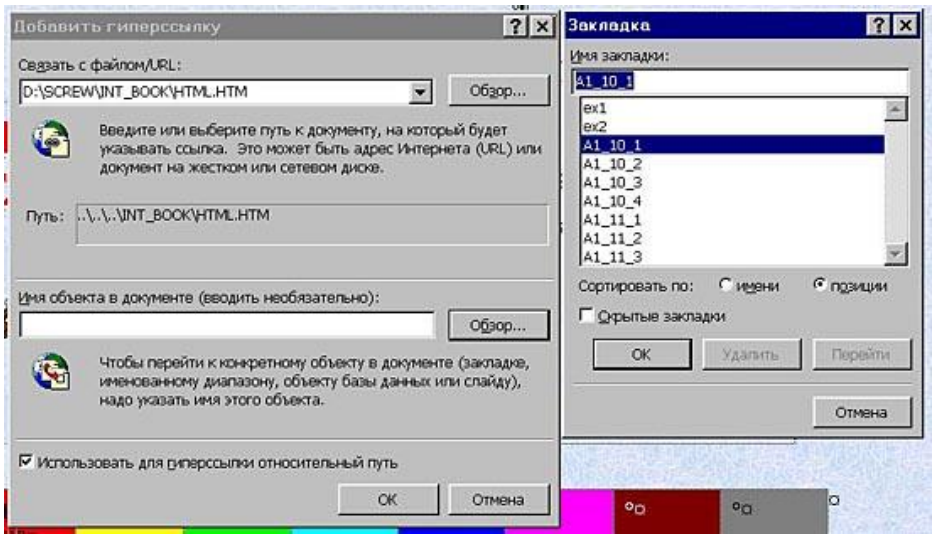
Расм. 2



Расм. 3

Word 97 да аввал матндаги домен номлари (www ёки http:// дан бошланадиган) ва электрон почта манзили англанади. Сервиз менюдаги Автозамена буйруқини танлаб олингач, диалог дарчада кўринган Автоформат закладкасида сичқончани чиқиллатиб Интернет адресида байроқча белгиланади. Мақсад, Word киритилгач тармоқ манзили матнини шу каби хаволаларга талаб қилинган рангда ўзгартириб берсин.

Иккинчи усул - ихтиёрий матн фрагменти ажратиьб олинади ва Гиперссўлка буйруқи (Вставка менюси) билан ёки асбоблар панелидаги Добавить гиперссўлку тугмаси босилади. Мулоқот дарчаси (расм 4) нинг энг юқори майдонига талаб қилинаётган манзил киритилади. Масалан, WWW.domain.Ru ёки http: user. Domain.Ru – Интернет сервер манзили ёки mailto: user @ domain.com –n электрон почта манзили.



Расм 4

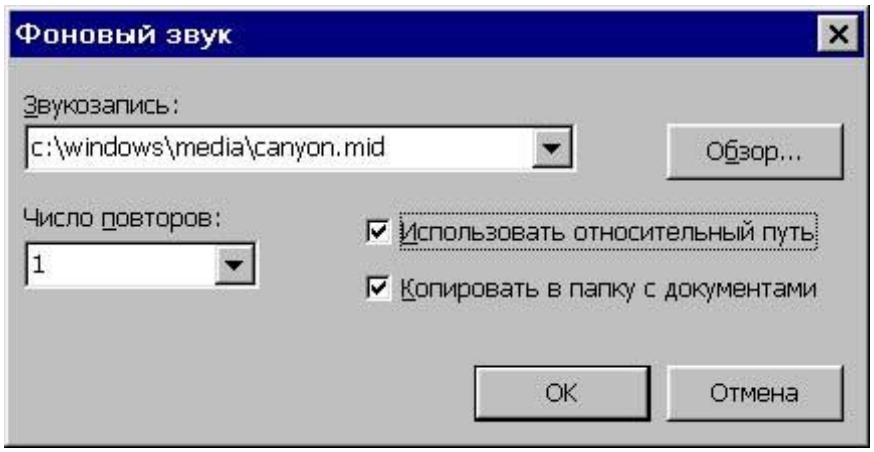
қўшиб қўйилган гиперхаволани ўзгартириш учун унда сичқоннинг ўнг тугмаси билан чиқиллатиб менюдаги Гиперссўлка буруқи танланади. Гиперхаволани ўзгартиришда экранда худди шундай диалог дарчаси пайдо бўлади (янги хавола яратиш каби). Гиперхаволани ўчириш учун, уни тахрирлашга чиқарилади ва Связать с файлами (URL майдони) қиймати йўқолади. Бунда хавола матни ўчиб кетмайди, лекин гиперхавола сингари расмийлашган хисобланмайди.

Мультимедиа элементларини ўрнатиш.

Рисунок буйруқи (меню-Вставка) ёрдамида матнларга ўрнатилажак расмлардан ташқарии яратилажак Интернет сахифага видео фрагмент, фонли товуш, югуриб ўтувчи сатр ва шу каби элементларни қўшиб қўйиш мумкин.

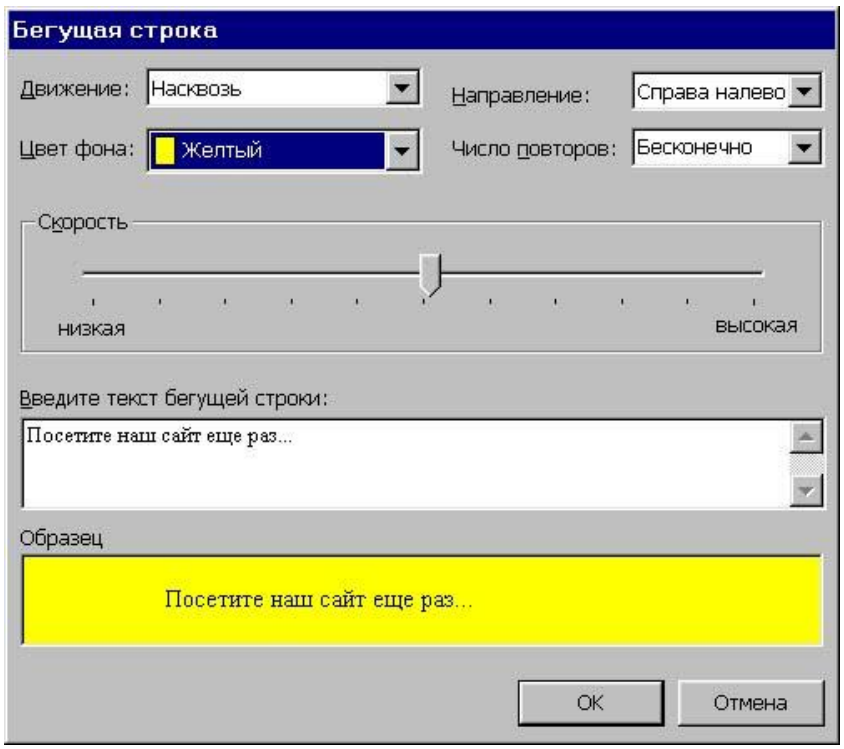
- Фонли товуш Вставка менюсида Фоновўй звук, Свойства буруқини танлаш ва кўринган диалог дарчасидан (расм 13) ихтиёрий файл

номини (Wav ёки midi) ва такрорлаш сонини кўрсатинг. Кўрсатилган файл ўша папкага нусхаланади ва номи тег (BGSOUND) таркибига киритилади.



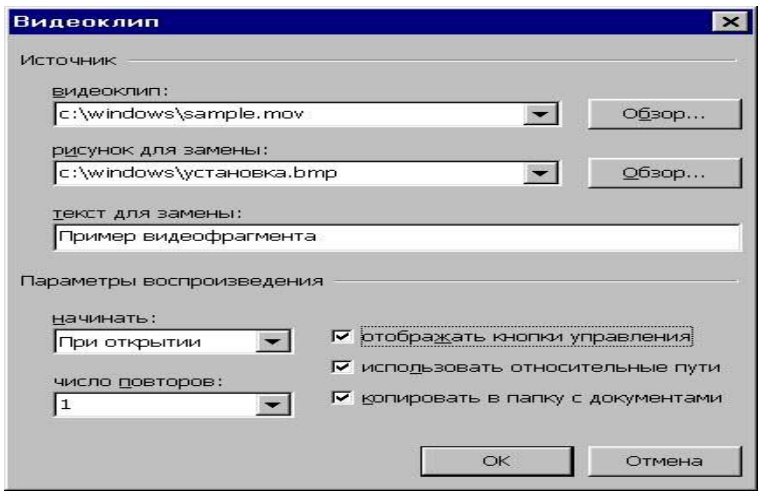
Расм 5

- Югуриб ўтувчи сатр . Вставка менюсида Бегущая строка ни танланг ва хосил бўлган диалог дарчасида маъқул бўлган параметрни кўрсатинг (расм 6): Движение: Насквозь- рулонли харакат, яъни сатр экраннинг бир қиррасида йўқолиб қолиб, яна қарама-қарши томонида пайдо бўлади -SCROLL. До края – сатрнинг бир марта қарама-қарши қиррагача харакати ва тўхташи - SLIDE. LOOP қ1 бўлганида қирралараро – бир қиррадан иккинчи қиррагача ва тескари бўлган харакат – ALTIRNATE; сатр фони ранги; харакат йўналиши (чапдан ўнгга ёки ўнгдан чапга); такрорланиш сони (тўхтовсиз харакат); харакат тезлиги ва сатр матни) . Параметр ўзгариши шу захотиёқ қуйида жойлашган. Образец майдонида акс этади.



Расм 6

- Видеофрагмент – Вставка менюсида Видеоклип буйруқини танланг ёки расм чизиш асбоблар панелидаги Видеоклип тугмасидан фойдаланинг. Экранда диалог дарчаси пайдо бўлади (расм.15.) унда расм ва матнларни алмаштириб берувчи видеоёзув файли кўрсатилади. Юклаш шароити (сичқонни чиқиллатиб автоматик усулда); такрорлаш сони (узлуксиз), акс этиш ёки бошқариш панелини очиш (эшитиб кўриш) ва бошқалар. Тўқри, бу параметрлар барча браузерларда аниқ ва равшан қабул қилинмайди, айниқса бу холат сичқонни чиқиллатиб юклашда ва бошқариш панелини эшитиб кўриш вақтида содир бўлади.



Расм. 7.

Интернет – сахифани тахрирлаш жараёнида кўриб чиқиш.

Word 97 HTML хужжатларни акс этиш учун мўлжалланган бўлса-да, унда Просмотр Web – страницў тугмасида сичқонни бир марта чиқилатиш билан жорий хужжатни Internet Explorer браузерига ўтказиш имконияти мавжуд. Шу билан бирга яратилажак сахифага мос келган HTML – код билан щам бевосита ишлаш мумкин (Вид менюсидаги Источник HTML буйруқи билан). Одатдаги акс эттириш услубига қайтиш учун асбоблар панелининг юқори қисмидаги Закрўть источник HTML тугмасида чиқиллатиш зарур.

Word 97 фойдаланувчиларга Интернет сащифани тахрирлаш учун қулай мухит – WYSIMYG тақдим этсада, у камчиликлардан холи эмас. Word 97 (аниқроқи, унга ўрнатилган HTML-форматидаги конвертор). Интернет – Excplorer браузернинг охирги версияларига мўлжалланган, шу сабабли олинадиган Интернет-сащифани бошлангич хужжатга максимал даражада мос келиши учун юзага келадиган (генерация) HTML- матнига мажбур бўлмаган қатор тег ва параметрлар қуйиладики, улар кўпинча бошқа браузердаги сахифаларни кўриб чиқишда халақит беради. Масалан, Word 97 HTML-кодга битта шрифт номидаги кўплаб <FONT> тегларини мажбурий равишда FACE параметри қийматига хар сафар русча харфлардан лотинчага ва аксинча ўтиш вақтида қўшиб беради. Аслида уларда фақатгина Unicode шрифт номлари ёзилади. Бу эса Windows (3. xx ва 95) нинг нисбатан одинги версияларида нокоррект ишлашга олиб келади.

Шундай қилиб, Word 97 ни мураккаб жадвалли Интернет-сахифаларни яратишда ёки кўринмас разлиновкали жадваллар ёрдами билан сахифалашда, шу билан бирга матни файлга жойлаштирилган кўплаб бир турли сахифаларни автоматлаштириш жараёнини тайёрлашда қўшимча восита сифатида қўллаш мақсадида мувофиқ бўлади. Лекин Интернет-сахифани хақиқий профессионал даражада яратиш истаги бўлганда Word 97 нинг мавжудлиги фойдаланувчини HTML тили билан танишиб чиқиш зарурлигидан озод қилмайди.

Такрорлаш учун саволлар:

1. Web- сахифа мастери ёрдамида «Создание новой страницў» спецификасини очиб кўрсатинг?

2. Word даги сахифа фонини танлашни амалга ошириш?

3.Word тахрирлагичида Web -сахифадаги барча матнлар рангини ўзгартириш?

4. Word тахрирлагичида Web-сахифа жадваллари билан ишлаш?

5. Word да гипералоқалар билан ишлаш?
6. Web-сахифаларга хархил видеофрагментлар фонли овозлар, югурувчи сатрлар ва шахсий компьютерни қўшиш усуллари?

8-маъруза. Фреймлар ва улар асосида саҳифаларни яратиш.

1.Фреймлар

2. Фреймлар тизимини тасвирлаш

3.Фреймлар орасидаги ўзаро таъсир

Калит сўзлар: FRAMESE, FRAME, ROWS, COLS, MARGINWIDTH MARGINHEIGHT, SCROLLING, NORESIZE, FRAMEBORDER

Фреймлар барузерни кузатув ойнасини ёнма-ён жойлашган бир нечта тўғри бурчакли соҳаларга бўлиш имконини беради. Мазкур бўлаклардан ҳар бирига алоҳида HTML-файл, яъни бошқалардан мустақил равишда кўздан кечирилувчи файлларни юклаш мумкин. Зарурият туғилганда фреймлар орасида ўзаро боғлиқликни ташкил этиш мумкин. Ўзаро боғлиқлик ташкил этилганда фреймлардан бирида ссылка танланса, бошқа фрейм ойнасида керакли ҳужжатнинг юкланишига олиб келади.

Гарчи HTML-ҳужжатларда фойдаланувчига ахборот акс эттирилишининг турли усуллари ҳавола этилсада, ахборотни ифодалашнинг фрейм тизими ҳам ўзининг афзалликларига эга. Қуйидаги холларда айнан фрейм тизими қўл келади:

- Бир сохада ишлаётганда бошқа бир сохага ҳужжатларни юклаш орқали бошқаришни ташкил этиш зарурати туғулганда;
- Экраннинг бошқа ҳудудларида нима бўлишидан қатъий назар экранда доимо кўриниб туриши керак бўлган ахборотни кўздан кечириш дарчасининг маълум қисмига жойлаштириш лозим бўлганда;
- Дарчанинг хар бири мустақил равишда кўриб чикилиши мумкин бўлган ёнма-ён бир неча сохаларида жойлаштириш қўлай бўлган ахборотни такдим этиш зарурати туғулганда.

Фреймлар тизимини тасвирлаш учун <FRAMESET>, <FRAME> ёки <NOFRAME> тэгларидан фойдаланилади.

<FRAMESET> тэги фреймларни белгилайди.

Фреймлардан ташкил топган Web-саҳифалар <BODY> бўлинмасига эга бўлиши мумкин эмас.

<FRAMESET> ва </FRAMESET> контейнерлари хар бир фреймни белгилаш блокини ўраб туради. Бундай контейнернинг ичида фақат <FRAME> тэглари ёки киритилган <FRAMESET> тэглари мавжуд бўлади.

<FRAMESET> тэгининг атрибутлари:
ROWS,
COLS.

Ушбу параметрлар қийматлари пикселларда, фоизларда ёки нисбий бирликларда берилиши мумкин. Қатор ёки устунлар сони мос рўйхатдаги қийматлар сони билан аниқланади. Масалан:
<FRAMESET ROWS = “100, 240, 140”> - учта фреймдан иборат тўпламни белгилайди. Қийматлар пикселларда берилган. Биринчи фрейм 100 пиксел, иккинчиси 240 пиксел ва ниҳоят сўнгги фрейм 140 пиксел баландликка эга.
<FRAMESET ROWS = “25%, 50%, 25%”> -
экраннинг мақбул баландлигидан юқори қаторнинг қиймати 25 фоиз, ўрта қаторники 50 фоиз, қуйи қаторники 25 фоиз эканлигини билдиради.
<FRAMESET COLS = “*, 2*, 3*”> - қийматлар нисбий бирликларда. “Юлдузча” – “*” фазони пропорционал тақсимлаш учун ишлатилади. Хар бир юлдузча бутуннинг бир қисмини билдиради. Ҳисоблаб топиш учун юлдузчалар олдидаги сонларни қўшиш ва хосил бўлган сондан касрнинг махражи сифатида фойдаланилади. Юқоридаги мисолда биринчи устун дарча умумий кенглигининг 1/6, иккинчи устун 2/6, учинчи устун 3/6 қисмини эгаллайди.
<FRAMESET COLS = “100, 25%, *, 2*”>.
<FRAME> тэги алохида файлларни белгилайди, бу тэг <FRAMESET> ва </FRAMESET> тэглари жуфтлигининг ичида жойлашиши лозим. Масалан:
<FRAMESET ROWS = “*, 2*”>
<FRAME>
</FRAME>
</FRAMESET>

<FRAMESET> тэги берилганида қанча алоҳида фреймлар белгиланган бўлса, шунча фрейм тэгларини ёзиш лозим.
<FRAME> тэги атрибутлари:

SRC
NAME
MARGINWIDTH
MARGINHEIGHT
SCROLLING
NORESIZE
FRAMEBORDER=YES/NO (Фақат IE лар учун)

SRC атрибути бошидан бошлаб мазкур фреймга юкланувчи ҳужжатнинг URL-манзилини белгилайди. Одатда бундай манзил сифатида асосий ҳужжат қайси каталогда бўлса, уша ерда жойлашган HTML-файлнинг номидан фойдаланилади. Масалан:
<FRAMESET SRC=“sample.html”>

Зеро, фреймни тасвирлашда берилган HTML-файл тўлиқ HTML-ҳужжат бўлиши керак, яъни у HTML, HEAD, BODY ва бошқаларга эга бўлиши лозим. Агар фреймдан тасвирни акс эттиришда фойдаланилса, унда:
<FRAME SRC=“http://www.bhv.ru/exampl.gif”>

NAME параметри берилган фреймга ссылка сифатида ишлатиш мумкин бўлган фреймнинг номини белгилайди. Масалан:
<FRAME SRC=“sample.html” NAME=“frame1”>
frame1 деб номланган ушбу фреймга ссылка қилиниши мумкин. Масалан:
<A HREF=“other.html” TARGET=“frame1”>
frame1 </A> фреймига other.html файлини юклаш учун шу ерга сичқонча курсори босилади.

MARGINWIDTH ва MARGINHEIGHT атрибути фрейм хошия(чегара) кенглигини белгилайди.
Атрибутлар қийматлари пикселларда берилади.
Масалан:
<FRAME MARGINWIDTH = “5” MARGINHEIGHT = “7”>
Бу ерда фрейм юқори ва пастда 5 пиксел, ўнг ва чап томонларидан эса 7 пиксел чегарага эга. Ишлатилиши мумкин бўлган энг кичик қиймат 1 пикселдир.

SCROLLING атрибутидан **прокрутка** йўлакларини акс эттиришни бошқаришда фойдаланилади. Унинг синтаксиси
<FRAME SCROLLING= “YES/ NO/ AVTO”> кўринишга эга.

NORESIZE атрибути фойдаланувчи томонидан фрейм ўлчами ўзгартирилишининг олдини олишда ишлатилади. Масалан:
<FRAME NORESIZE>.

Табиийки NORESIZE атрибутининг битта фреймга нисбатан қўлланилиши бошқа фреймлар ўлчами ўзгартирилишининг ҳам олди олинишига сабаб бўлади.

Гарчи фреймлар тизими HTML 4.0да стандарт билан мустахкамланган бўлсада, <NOFRAMES> тэги фреймларни қўллаб-кувватламайдиган браузерлар ёрдамида кўздан кечиришда асқотади. Демак, фреймларга боғланмаган браузерлар учун <NOFRAMES>ва </NOFRAMES> тэглари жуфтлигидан фойдаланилади. Масалан:

<NOFRAMES> бутун HTML-ҳужжат</NOFRAMES>

Мазкур тэглар орасига жойлаштирилган барча маълумотлар фреймларни қўллаб-қувватлаш имкониятига эга бўлмаган браузерлар ёрдамида акс эттирилади. Фреймларга боғланган браузерлар эса <NOFRAMES> ва </NOFRAMES> орасидаги барча ахборотга боғлиқ емас. Юқорида келтирилган параметрлар ишлатилган мисолларни кўриб чикамиз.

Фреймлар орасидаги ўзаро таъсир

Фреймлар билан ишлаётганда фойдаланувчи учун қўлай бўлган ҳужжат юклаш схемасини яратиш мумкин. Фреймлар орасидаги ўзаро алоқа ҳужжатларни бошқа фреймдаги буйруқлар ёрдамида айнан танланган фреймга юклаш имконини беришидадир. Бу мақсадда <A> тэгининг TARGET атрибутидан фойдаланилади. TARGET атрибути ушбу ссылка кўрсатаётган ҳужжат юкланувчи фрейм ёки браузер ойнаси номини

белгилайди. Ўзгартирилмаган ҳолда ушбу параметр йўқ бўлганда ҳужжат жорий фрейм ёки ойнада юкланади. Фрейм номи сифатида мавжуд дарча ёки фрейм номи берилиши ёки бўлмаса янги ойна очиш учун янги ном берилиши мумкин. 4 та захирадаги номлар бор. Улар:
_blank, _self, _top, _parent. Булардан ташқари "_" белгиси билан бошланувчи хар қандай номдан фойдаланиш мақсадга мувофиқ эмас.
TARGET="_blank" – ҳужжатнинг янги ойнага юкланишини таъминлайди. Бу ойна номга эга бўлмаслиги туфайли унга бошқа ҳужжатни юклашнинг иложи бўлмайди.
TARGET="_self" дан фойдаланилганда ҳужжат жорий фреймга юкланади.
TARGET="_top" ҳужжатнинг бутун дарчага юкланишига сабаб бўлади.
TARGET="_parent" ҳужжатнинг жорий фреймнинг фрейм-ота-онаси томонидан эгалланган сохасига юкланишига олиб келади.

Таркиби "А" фреймига юкланган frame_a.html файлининг биттагина test.html файлига TARGET параметрининг турли қийматларига эга 6 та ссылкаси мавжуд.

Такрорлаш учун саволлар:

1. Фреймлар қандай ишлайди?
2. Ҳужжатда линкни фреймга боғлашни тушунтиринг?
3. Объект нима ва қанақа объектларни биласиз?

8-маъруза. Формалар. Формалар билан ишлаш.

Режа:

1. FORM элементи
2. <INPUT> теги
3. Атрибутлар

Калит сўзлар: FORM, FORM METHOD, URL, TYPE, CHECKBOX, ALIGN, VALUE, *OPTION*. SELECTED.

HTML-формалар

Формалар Веб дастурлашда фойдаланувчи томонидан киритилаётган маълумотларни тартибга солиш учун қўлланилади. Форма элементлари тўлдирилгандан кейин ундаги маълумотлар сервердаги маълумотларни қайта ишловчи дастурга юборилади. Кўп сонли жўнатилаётган маълумотлар жўнатиш тугмаси босилгандан сўнг серверда жойлашган Common Gateway Interface (CGI) ёрдамида ёки махсус файл орқали қайта ишланади. Шу тариқа фойдаланувчи Internet орқали Web-сервер билан биргаликда ишлайди.

Форманинг берилиши —FORM элементи

FORM элементи ҳужжатни маълум бир *формага* солади ва *форма* элементлари тэгларини бошқа тэглардан ажратиб туради. <FORM> бир нечта <*INPUT*> ёки шу каби бошқа тэглар кетма кетлигидан ташкил топади. Улар <FORM> ва </FORM> тэглари орасига жойлаштирилади. Формада усулдан (method), *формага* киритилган маълумотларни қайта ишлаш учун ҳолатлар (action) мавжуд. Усул (GET ёки POST) формага киритилган маълумотлар қай тарзда серверга жўнатилиш усулини белгиласа, ҳолат эса сервердаги қайси дастурга юборилиш URL (Uniform Resource Location) манзилини ифодалайди.

<FORM METHOD=“post” ACTION= “<URL>”>

Форманинг бошқарув элементлари —<INPUT> теги

Ушбу тэг *форманинг* қайси нуқтасига маълумот киритилишини белгилайди. У фойдаланувчи томонидан киритилаётган маълумотларни формага келтиради. Булар матн киритиш майдони, рўйхатлар, расмлар ёки тугмалар бўлиши мумкин. Майдон типи TYPE атрибути ёрдамида аниқланади.

TYPE=text атрибути

Агар фойдаланувчи унча катта бўлмаган матн киритса (бир ёки бир нечта сатр), <*INPUT*> тэгидан фойдаланади ва TYPE атрибутига text қиймати ўзлаштирилади. Стандарт ҳолат учун бу қийматни бериш муҳим эмас. Бундан ташқари *майдонни* номлаш ва унга мурожаат қилиш учун NAME атрибути ҳам берилади.

Сизнинг исмингиз <INPUT NAME=Name SIZE=35>

Фойдаланиш мумкин бўлган яна учта қўшимча атрибутлар мавжуд. Биринчиси MAXLENGTH деб аталади, у фойдаланувчи киритаётган матн майдони максимум узунлигини белгилайди. Стандарт бўйича бу қиймат чегараланмаган. Иккинчи атрибут SIZE ҳисобланади, у эса матн майдонини кўриниб турувчи қисмини белгилайди. Стандарт бўйича унинг қиймати браузерга боғлиқ бўлади. Агар MAXLENGTH қиймати SIZE қийматидан катта бўлса, браузер маълумотни ойнага мослаштиради. Сўнга қўшимча атрибут матн майдонини бошланғич қийматини белгиловчи VALUE дир.

TYPE=checkbox атрибути

HTML *формада* мустақил белгилагич (байроқча) дан фойдаланиш учун <*INPUT*>тэгининг атрибутига TYPE=checkbox ни ўзлаштириш керак. Формага боғлиқ равишда фойдаланувчи бир ёки бир нечта белгилагичларни белгилаши мумкин. Агар <*INPUT*> тэги атрибути билан CHECKBOX қиймати қўлланилса, у билан бирга NAME ва VALUE атрибутлари ҳам қўлланилиши керак. NAME атрибути ушбу маълумот киритиш объектининг номини ифодалайди. VALUE атрибутида ушбу *майдон*нинг қиймати кўрсатилади.

Россия<INPUT NAME="Давлат" TYPE=checkbox
VALUE="Россия">
Страны СНГ<INPUT NAME="Давлат" TYPE=checkbox
VALUE="СНГ">

Баъзи ҳолларда ушбу майдон белгиланган ҳолда қўлланилиши ҳам мумкин. Бундай ҳолларда <*INPUT*> тэгида CHECKED атрибути қўлланилиши керак.

TYPE=radio атрибути

Баъзан бир нечта қийматлар орасидан бирини танлашга тўғри келади. Бундай ҳолларда формада <*INPUT*> тэги билан бирга TYPE=radio атрибути қўлланилади. Агар <*INPUT*> тэги атрибути билан ушбу қиймати қўлланилса, у билан бирга NAME ва VALUE атрибутлари ҳам қўлланилиши керак. NAME атрибути ушбу маълумот киритиш объектининг (*тугма*) номини ифодалайди. VALUE атрибутида ушбу *майдон* нинг қиймати кўрсатилади..

Эркак жинси <INPUT NAME="Жинс" TYPE=radio
VALUE="Эркак">
Аёл жинси <INPUT NAME="Жинс" TYPE=radio
VALUE="Аёл">

TYPE=image атрибути

Форманинг таркибига қараб баъзан унда жойлашган расмнинг устига сичқончани босиш билан ундаги маълумотларни жўнатишга тўғри келиб қолади. Бунинг учун <*INPUT*> тэги TYPE=image атрибути билан қўлланилади. Фойдаланувчи расм устига сичқонча курсорини босса, айнан шу ердаги экран координаталарини браузер сақлаб қолади. Сўнг формага киритилган маълумотларни “қайта ишлайди”. Агар <*INPUT*> тэги image

атрибути билан қўлланилса, у билан бирга NAME ва SRC атрибутлари ҳам қўлланилиши керак. NAME майдоннинг номини белгилайди. SRC атрибути эса расм жойлашган манбанинг URI манзилини беради. ALIGN атрибути қўшимча ҳисобланади ва у ҳам баъзан <IMG> теги билан қўлланилади.

Нуқтани танланг <INPUT TYPE=image NAME=point
SRC=image.gif>

TYPE=password атрибути
Агар *формада* пароллардан фойдаланиш керак бўлиб қолса, TYPE атрибути қийматига password (TYPE=password) ни ўзлаштирилади. Ушбу типдан фойдаланиш киритилаётган маълумотни ошкор бўлмаган ҳолда кўрсатишни таъмин этади. Шу сабаб, киритилган маълумот очиқ канал орқали жўнатилади ва ушбу маълумот тутиб олиниши мумкин.

Номингиз<INPUT NAME=login>Парол
<INPUT TYPE=password NAME="Сўз">

TYPE=reset атрибути
Базан фойдаланувчи *формани* тўлдириш вақтида, уларни бошдан тўлдиришга тўғри келади. Ушбу ҳолда Reset тугмаси мавжуд бўлиб, бу тугманинг босилиши формани дастлабки, кириш ҳолатиги олиб келади (формани “тозалайди”). Reset иугмасини ташкил қилиш учун <*INPUT*> тэги атрибутига TYPE=reset ўзлаштирилади. Агар *формада* reset атрибути қўлланилса, <*INPUT*> тэгига VALUE атрибутини қўшимча қилиш мумкин. Ушбу атрибут тугмадаги ёзувни ифодалайди.
<INPUT TYPE=reset VALUE="Формани тозалаш ">

TYPE=submit атрибути
HTML *форма* да фойдаланувчи маълумот киритиш жараёнини якунлаш жараёни мавжуд. Бунинг учун <*INPUT*> тэгининг атрибутига TYPE=submit қиймат ўзлаштирилади. Агар *формада* <*INPUT*> тэги submit атрибути билан қўлланилса, унга қўшимча равишда иккита атрибутдан фойдаланиш мумкин: NAME ва VALUE. NAME атрибути *майдоннинг* номини ифодалайди. VALUE атрибути — Submit тугмаси матнини кўрсатади.
<INPUT TYPE=submit VALUE="Хабарни жўнатиш "/>

TYPE=hidden атрибути
Яширин *майдон*. *INPUT* тэгини TYPE=hidden атрибути билан қўлланилиши фойдаланувчига маълум бўлмаган NAME ва VALUE атрибутларидаги қийматларни жўнатишга имкон беради.

<TEXTAREA> – кўп сатрли матн киритишни ташкил этиш тэги
Баъзан формада кўп сатрли матнларни киритиш талаб этилади. Бунинг учун **<*TEXTAREA*>** тэги ёрдамида бир неча сатрдан иборат бўлган матн майдони

ташкил этиш мумкин. Ушбу тэг учта атрибут билан ишлатилади: COLS, NAME ва ROWS.

Атрибут COLS
Майдоннинг устунлари (белгилар сони) сонини белгилайди.

Атрибут NAME
Майдоннинг номини белгилайди.

Атрибут ROWS
Майдоннинг кўринувчи сатрлари сонини белгилайди.
<TEXTAREA NAME=мавзу COLS=38 ROWS=3></TEXTAREA>

<SELECT>- формада рўйхатдан фойдаланиш тэги
Агарда *форма* мукаммал бўлса, гоҳида унда ҳаракатланувчи рўйхат ҳам қўлланилади. Бунинг учун ***SELECT*** тэгидан фойдаланилади. Рўйхат бўлимларини аниқлаш учун <*OPTION*>тэгидан фойдаланилади. <*SELECT*> тэги муҳим бўлмаган учта атрибутни қўллаб қувватлайди: MULTIPLE, NAME ва SIZE.

MULTIPLE атрибути
Бир вақтнинг ўзида бир нечта вариантни танлаш имконини беради.

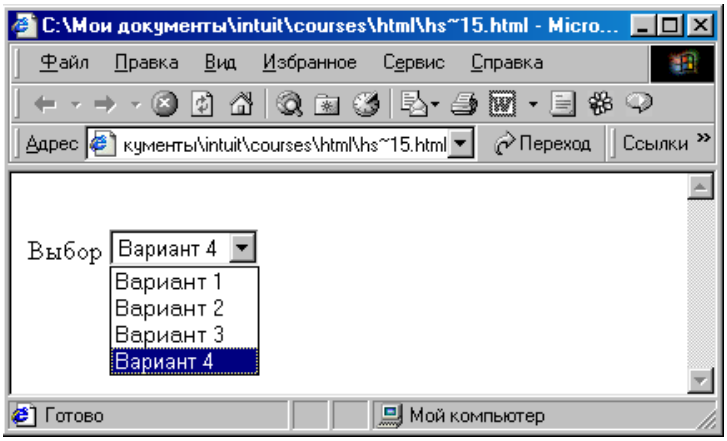
NAME атрибути
Объект номини ифодалайди.

SIZE атрибути
Рўйхатни кўринувчи сатрлари сонини ифодалайди. SIZE > 1 бўлган ҳолда браузер оддий рўйхатни кўрсатади. *Формада* <*OPTION*> теги фақат <*SELECT*> тэглари орасида қўлланилади. Параметрлари: SELECTED ва VALUE.

SELECTED атрибути
Дастлаки ҳолатда ушбу элемент танланган эканлигини билдиради.

VALUE атрибути
Рўйхатга ўзлаштирилиши мумкин бўлган қийматни ифодалайди.

Танлаш
<SELECT NAME="Танлаш">
<OPTION>Вариант 1
<OPTION>Вариант 2
<OPTION VALUE="Вариант 3">Вариант 3
<OPTION SELECTED>Вариант 4
</SELECT>



Қалқиб чиқувчи меню

Такрорлаш учун саволлар:

1. Форманинг берилиши —FORM элементи хақида нималарни биласиз?
2. TYPE атрибути елементлари?
3. формада рўйхатдан фойдаланиш тэги?

9-маъруза. Клиент томони (соҳаси) да дастурлаш. JavaScript га кириш. JavaScript ни HTML-хужжатга жойлаштириш.

Режа:
1. Клиент томони (соҳаси) да дастурлаш. JavaScript га кириш
2. JavaScript нинг объектли модели тушунчаси
3. JavaScript нинг URL-схемаси
4. Синфлар иерархияси

Калит сўзлар: Клиент томонда дастурлаш, JavaScript тили

Клиент томони (соҳаси) да дастурлаш. JavaScript га кириш.

Web саҳифани генерация қилиш жараёнида "клиент-сервер " архитектураси билан боғлиқ равишда амалга оширилади. Саҳифалар клиент томонида ҳам сервер томонидаги каби генерация қилинади. 1995 йилда Netscape компанияси мутахассислари клиент томонидаги саҳифаларни генерация қилиш учун махсус дастурлаш тили яратишган ва уни JavaScript деб номлашган.

JavaScript – клиент томонидаги гиперматнли Web саҳифанинг сценарийларини бошқарувчи тилдир. Аниқроқ айтадиган бўлсак, JavaScript – бу фақатгина клиент томонидаги дастурлаш тили эмас. JavaScript нинг аждоди Liveware - Netscape сервери томонидаги восита ҳисобланади. Шундай қилиб, JavaScript кўпроқ клиент томонидаги сценарийларни ташкил этувчи тил сифатиди оммавийлашган.

JavaScript нинг асосий ғояси HTML саҳифаларни кўриш вақтида HTML тэг ва контейнерларнинг атрибутлари қийматларини ва хусусиятларини ўзгартиришдан иборат. Шу сабаб саҳифани қайта юклаш амалга ошмайди.

Амалиётда буни биз, саҳифа фонининг рангини ёки хужжатдаги расм хусусиятларини ўзгартиришда, янги ойна очиш ёки огоҳлантириш бериш жараёнларида яққол кузатишимиз мумкин.

JavaScript тили ECMA (European Computer Manufacturers Association – Европа компьютер ишлаб чиқариш ассоциацияси) томонидан стандартлаштирилган. Ушбу стандартлар ECMA-262 ва ISO-16262 номларини келтириб чиқарди. Бу стандартлар JavaScript 1.1 га мош тушувчи ECMAScript тилини тақдим этади. Бугунги кунда JavaScript нинг барча турлари ҳам ECMA стандартига мос тушавермайди.

JavaScript нинг объектли модели тушунчаси

Клиент томонидаги саҳифани яратишни бошқаришда хужжатнинг объектли механизмидан фойдаланилган. Бунда ҳар бир HTML-контейнер-бу объект ҳисобланади ва қуйидаги учликни ташкил этади:

- хусусиятлар

- усуллар
- холатлар

Объектли модел саҳифалар ва браузерлар ўртасидаги боғланишсифатида кўриниши мумкин. Объектли модел – бу HTML код орқали берилган элементларни объект, усул, хусусият ва холатлар кўринишида таниш ва улар билан ишлаш демакдир. У ёрдамида биз браузерга ва фойдаланувчига мурожаат қилишимиз, хабарлар юборишимиз мумкин. Браузер бизнинг буйруқралимизни бажаради ва экранда саҳифанинг керакли қисмларини ўзгартиради.

Объектлар бир хил типли хусусиятлар, усуллар ва холатлар тўпламини бир хил типли объектлар синфларида бирлаштиради. Объектларнинг ўзлари фақат хужжатни браузер ёрдамида юклашда ёки дастурнинг натижаси сифатида намоён бўлади. Ушбу ҳолатни доимо ёдда тутиш керак.

Хусусият

Кўпгина HTML-контейнерларда атрибутлар мавжуд. Масалан, якор контейнерида <A ...>...</A> HREF атрибути мавжед. Ушбу атрибут уни гипер мурожаатга айлантиради:

```
<A HREF=intuit.htm>intuit</A>
```

Агар <A ...>...</A> якор контейнерини объект сифатида кўрадиган бўлсак, HREF атрибути "якор" объектини хусусияти ҳисобланади:

```
document.links[0].href="intuit.htm";
```

Барча атрибутлар қийматларини ҳам ўзгартириб бўлавермайди. Масалан график расимларнинг ўлчамлари дастлабки берилган қиймати асосида аниқланади, яъни уларни ўзгартириб бўлмайди. Кетма кет келган барча расимлар қийматлари ўзининг дастлабки қийматигача масштабланиши мумкин. Microsoft Internet Explorer да расим ўлчамлари ўзгартирилиши мумкин.

Умумийлик учун расм хусусиятлари JavaScript да HTML-разметкада мавжуд бўлмаган объектларга бўлинади. Масалан, восита сифатида Navigator деб номланувчи объектни, ёки JavaScript даги энг асосий объектлардан – браузер ойнаси объектини олишимиз мумкин.

Усуллар

JavaScript атамаларида объект усуллари унинг хусусиятларини ўзгартирувчи функцияларни англатади. Масалан, "документ" объектида open(), write(),

close() усуллар мавжуд. Ушбу усуллар мавжуд хужжатнинг қайта ишлаш ёки таркибини ўзгартириш учун хизмат қилади. Оддий мисол келтирамиз:

```
function hello()
{ id=window.open("","example","width=400, height=150");
id.focus(); id.document.open();
id.document.write("<H1>Салом!</H1>");
id.document.write("<HR><FORM>");
id.document.write("<INPUT TYPE=button VALUE='Ойнани ёпиш ");
id.document.write("onClick='window.opener.focus();window.close();'>");
id.document.close();
}
```

Ушбу мисолда open() усули хужжатга ёзиш имкониятини яратади, write() усули ушбу ёзишни амалга оширади, close() усули хужжатга ёзишни ёпади. Буларнинг барчаси оддий файлга ёзган каби амалга ошади. Агар ойнада холат сатри мавжуд бўлса (одатда хужжатнинг юкланиш даражаси берилади), хужжатга ёзиш жараёни тугалланмаган бўлса, хужжат юкланиш вақтида унда тўғри тўртбурчак шаклидаги ёзув давом этаётганлигини ифодаловчи белги “кўринади”.

Ҳолат

Усуллар ва хусусиятлардан ташқари объектларни ҳолатлар билан ҳам характерлаш мумкин. Шахсан, JavaScript да дастурлашда ушбу ҳолатларни қайта ишловчи воситалар мавжуд. Масалан, button типидаги объект билан onClick ҳолати амалга ошиши мумкин, яъни фойдаланувчи тугмани босиши мумкин. Ушбу атрибут қиймати сифатида дастурчи томонидан JavaScript да тузилган ҳолатни қайта ишловчи дастур кўрсатилади:

```
<INPUT TYPE=button VALUE="Нажать" onClick="window.alert('Салом!');" />
```

Ҳолатларни қайта ишлаш жараёнлари уларнинг ҳолатлари билан боғлиқ контейнерларда кўрсатилади. Масалан, BODY контейнери бутун хужжатнинг хусусиятини аниқлайди, шунинг учун бутун хужжатни ёпишни қайта ишловчи ҳолат onLoad атрибутининг қиймати сифатида BODY контейнери ичида берилади.

Изоҳ. *Қатъий айтиш мумкинки, ҳар бир браузур, Internet Explorer, Netscape Navigator ёки Opera да бўлганидек, ўзининг объектли моделига эга. Турли браузерлар объектли моделлари (ҳатто турли версиялари) бир биридан фарқланади, лекин мантиқий таркиби бир ҳилда бўлади*.

Кодни HTML-саҳифага жойлаштириш

JavaScript кодини браузерда бажарилаетганда браузердаги JavaScript интерпретатори ёрдамида амалга оширилади. JavaScript ни қўллашда тўртта функционал усулдан фойдаланиш мумкин:

- гиперматнли мурожаат (URL схема);
- ҳолатни қайта ишловчи (handler);
- подстановка (entity) (Microsoft Internet Explorer нинг 5.X ва юқори версияларида мавжуд);
- қўйиш ёки ўрнатиш (вставка).

HTML хужжат яратаётганда JavaScript нинг бир нечта усулларидан фойдаланиш мумкин.

JavaScript нинг URL-схемаси

URL (Uniform Resource Locator) схемаси – бу Web-технологиянинг асосий элементларидан бири ҳисобланади. Web да ҳар бир ахборот ресурси ўзининг уникал URL ига эга. URL A контейнернинг HREF атрибутида, IMG контейнернинг SRC атрибутида, FORM контейнерининг ACTION атрибутида ва бошқаларда берилади. Барча URL мулоқот протоколи турига қараб турли қисмларга бўлинади, масалан, FTP-архивга боғланиш учун ftp схема қўлланилади, Gopher-архивга боғланиш учун - gopher схемадан фойдаланилади, электрон почтани жўнатиш учун - smtp схемадан фойдаланилади. Схема тури URL нинг биринчи компонентаси орқали аниқланади: http://intuit.ru/directory/page.html

Гиперматнли тизимли дастурлаш тилининг асосий вазифаси гиперматнли ўтишларни дастурлашдир. Бу шуни англатадики, у ёки бу гиперматнли ссылканинг босилиши гиперматнли ўтишни амалга оширувчи дастурни ишга туширади. Web-технологиада шунга ўхшаш стандарт дастурлар саҳифани юклаш дастурлари ҳисобланади. JavaScript шу стандарт дастурларни фойдаланувчи дастурига айлантиради. HTTP протокол бўйича стандарт ўтишлардан фарқланиш мақсадида JavaScript да алоҳида URL схема жорий этилган:

<A HREF="JavaScript:JavaScript_код">...</A>
<IMG SRC="JavaScript:JavaScript_код">

Масалан, “Диққат!!!” номли гиперматнли ссылка босилганда огоҳлантириш ойнасининг очилиши қуйидагича амалга оширилади:

<A HREF="JavaScript:alert('Внимание!!!');"> Диққат!!!</A>

submit типидаги тугмани босиш орқали формадаги матн объекти тўлдирилиши қуйидагича амалга оширилади:

```
<FORM NAME=myform METHOD=post ACTION="JavaScript:window.document.myform. myname.VALUE='Сиз Click тугмасини босдингиз';void(0);">
<TABLE BORDER=0 width="100%">
<TR>
<TD><INPUT NAME= "myname" value= "" /></TD>
<TD><INPUT TYPE=submit VALUE=Click></TD>
<TD><INPUT TYPE=reset VALUE=Reset></TD>
</TR>
</TABLE>
</FORM>
```

URL да мураккаб дастурларни жойлаштириш ва функцияларни чақириш мумкин. JavaScript нинг бу схемаси барча браузерларда ҳам ишлайвермайди, Netscape Navigator типидаги ва Internet Explorer нинг тўртинчи версиясидан бошлаб ишлайди.

Ҳолатларни қайта ишловчилар

Ҳолатни қайта ишловчи типидпги (handler) дастурлар, шу ҳолатга алоқадор контейнер атрибутида берилади. Масалан, тугма босилган вақтда click ҳолати амалга ошади:

```
<FORM><INPUT TYPE=button VALUE="Тугма" onClick="window.alert('tuit');"> </FORM>
```

Подстановкалар

Подстановкалар (entity) Web-саҳифада жуда кам учрайди. Шунга қарамай у HTML-саҳифани браузер томонида генерация қилиш қулай восита ҳисобланади. Подстановкалар HTML-контейнер атрибутининг қиймати сифатида фойдаланилади. Масалан, стандарт ҳолат бўйича форма объектлари маълумотларини жўнатиш учун адрес сифатида жорий саҳифа URL адреси кўрсатилади:

```
<script>
function myfunction()
{
str = window.location.href;
return(str.length);
}
</script>
<form><input value="&{window.location.href};" size="&{myfunction()};" /></form>
<script>
<!—бу изоҳ ...javascript-код...// -->
```

```
</script>
<body>
Ҳужжат танаси
</body>
```

Биламизки, ҳужжатнинг <head> қисмидаги матнлар браузер ойнасида кўринмайди. Шунинг учун бу қисмга ҳужжат танасида чақирилувчи ва ишлатилувчи ўзгарувчилар ва функциялар жойлаштирилади. Бу соҳада Netscape Navigator браузери Internet Explorer га қараганда бироз қатъийроқ. Агар ҳужжат танасидаги функция сарлавҳа қисмида эълон қилинмаган бўлса, ушбу функция аниқланмаганлиги ҳақида хабар беради.

Мисол: Функцияларни жойлаштириш ва фойдаланиш:

```
<html>
<head>
<script>
function time_scroll()
{
d = new Date();
window.status = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
setTimeout('time_scroll();',500);
}
</script>
</head>
<body onLoad= “time_scroll();”>
<h1>ҳолат сатридаги соат </h1>
```

Internet Explorer 4.0 да подстановкалар ишламайди, шу боис улардан фойдаланишда эҳтиёт бўлиш керак. Бунда аввало браузер турини билиш талаб этилади.

Ўрнатиш (SCRIPT контейнери-интерпретаторни мажбурий чақириш)

SCRIPT контейнери – бу подстановка усулининг ривожланган варианти ҳисобланади. Жумладан, SCRIPT одатда Server Side Includes, яъни сервер томонидаги ҳужжатларни генерация қилувчи ҳам деб аталади. Интерпретатор SCRIPT теглари орасидаги барча қисимни генерация қилади ва шундан сўнг яна HTML қисимга қайтади.

SCRIPT контейнери иккита асосий функцияни бажаради:

- HTML-ҳужжатга кодни жойлаштириш;
- HTML-разметкаларни браузер томонида шартли генерациялаш.

Биринчи функцияси ўзгарувчилар ва функцияларни жойлаштириш учун қўлланилади. Иккинчиси - бу хужжатни юклаш ёки қайта юклаш вақтида JavaScript код натижасини жойлаштиришдир.

HTML-хужжатга кодни жойлаштириш

Шахсан, бу ерда асосий хилма хиллик йўқ. Код сарлавҳа контейнери HEAD орасига ҳам, BODY контейнери орасига ҳам жойлаштирилиши мумкин. Сарлавҳа қисмида қўлланилишини кўриб ўтамиз.

Сарлавҳа қисмида код SCRIPT контейнери орасига жойлаштирилади:

```
<HTML>
<HEAD>
<SCRIPT>
function time_scroll()
{
d = new Date();
window.status = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
setTimeout('time_scroll();',500);
}
</SCRIPT>
</HEAD>
<BODY onLoad=time_scroll()>
<CENTER>
<H1>Ҳолат сатридаги соат </H1>
<FORM>
<INPUT TYPE=button VALUE="Ойнани ёпиш " onClick=window.close()>
</FORM>
</CENTER>
</BODY>
</HTML>
```

Ушбу мисолда биз ҳужжат сарлавҳасида time_scroll() функциясини яратдик ва унга BODY (onLoad=time_scroll()) контейнерининг load ҳолатида мурожаат қилдик.

Қуйидаги функцияни яратиш ва чақириш орқали алоҳида ойна яратиш мумкин:

```
function sel()
{
id = window.open("","example","width=500,height=200,status,menu");
id.focus();
id.document.open();
id.document.write("<HTML><HEAD>");
id.document.write("<BODY>");
id.document.write("<CENTER>");
id.document.write("<H1>Change text into child window.</H1>");
id.document.write("<FORM NAME=f>");
id.document.write("<INPUT TYPE=text NAME=t SIZE=20 MAXLENGTH=20
VALUE='This is the test'>");
```

```
id.document.write("<INPUT TYPE=button VALUE='Close the window'
onClick=window.close()></FORM>");
id.document.write("</CENTER>");
id.document.write("</BODY></HTML>");
id.document.close();
}
<INPUT TYPE=button VALUE="Ҳолат сатрини ўзгартириш"
onClick="id.defaultStatus='Салом'; id.focus();">
```

Синфлар иерархияси

Объектга-мўлжалланган дастурлаш тили объектлар дарахтидан ташкил топади. JavaScript да бу иерархик дарахт Window объектидан бошланади, яъни ҳар бир объект у ёки бу ойнада ёзилади. Ихтиёрий объектга ёки объект хусусиятига мурожаат қилиш учун ундан юқорида турганг объект орқали мурожаат қилиш керак бўлади:

Умуман айтганда, JavaScript классик объектли тил ҳисобланмайди (уни соддалаштирилган объектли тил ҳам дейиш мумкин). Унда меросийлик ва наслдорлик мавжуд эмас. Дастурчи function оператори ёрдамида ўзининг класини, синфини объектини яратиши мумкин, аммо уларни яратишда одатда стандарт объектлардан ҳам фойдаланади. Бу шуни англатадики, JavaScript-дастурнинг амал қилиш соҳаси жорий саҳифа чегарасидан чиқиб кетмайди.

Баъзан JavaScript нинг турли объектларида бир ҳил номли ҳусусиятлар бўлади. Бу ҳолда дастурчи қайси объект хусусиятига мурожаат қилаётганини аниқ кўрсатиши керак. Масалан, Window ва Document ларда location ҳусусияти мавжуд. Фақат, Window учун бу Location синфи объекти, Document – URL да кўрсатилиб юкланаётган ҳужжатни адресини ифодалайди.

Таъкидлаш керакки, кўпгина объектларда объект хусусиятларини оддий қийматга ўзгартирувчи стандарт усуллар мавжуд бўлади. Масалан, стандарт ҳолда барча объектлар учун белгиларни сатрга айлантирувчи усул мавжуд: toString().

Такрорлаш учун саволлар

1. Клиент томонда дастурлаш деганда нимани тушунасиз?
2. JavaScript тили объекти нима ва қанақа объектларни биласиз?
3. JavaScript тили функциясини HTML ҳужжатда эълон қилинг.
4. JavaScript тили синфлари иерархиясини нима?

10-маъруза. JavaScript да маълумотлар типлари, ўзгарувчилар, ифодалар ва арифметик амаллар.

Режа:
1. JavaScript да ўзгарувчилар
2. JavaScript да маълумотлар типи
3. JavaScript тили операторлари
4. JavaScript тилида функция

Калит сўзлар: ўзгарувчилар, маълумотлар типи, операторлар, функция

JavaScript да ўзгарувчилар

JavaScript тилида ўзгарувчиларни ишлатиш мумкин ва уларни номлари билан адреслаш мумкин. Ўзгарувчилар глобалли ва локалли бўлиши мумккин. Глобалли ўзгарувчилар сценарийнинг хохлаган жойида рухсати бўлиши мумкин. Локалли ўзгарувчиларнинг харакати эса эълон қилинган ўзгарувчилар ичидаги функциялар билан чегараланган. Basic дастурлаш тили сингари JavaScript сценарийсини яратаётган вақтда аввалдан эълон қилинмаган ўзгарувчиларни ишлатиш мумкин.

Ўзгарувчилар эълони
Java Script да хамма ўзгарувчилар var калит сўзи орқали эълон қилинади ва қуйидагича кўрсатилган:

var MyHelloMsg;

Ўзгарувчи типи ўзлаштириладики қачонки, унга бирор бир қиймат ўзлаштирилса, қуйида аввалдан эълон қилинмаган матнли қатор ўзгарувчига ёзилмоқда:

MyMsg = "Салом!";

MyMsg ўзгарувчи номи ўзлаштирилгандан сўнг рухсат берилади.

Ўзгарувчи номини танлаганда, ыуйидаги оддий ыоидаларни ушлаб ыщйиш керак:
Ўзгарувчи номи харфлардан ёки "_", "$" белгилардан бошланиш керак ва фақат харфлардан, сонлардан ва "_", "$" белгилардан иборат бўлиши керак;
Ўзгарувчилар номи JavaScript нинг захираланган калит сўзлари билар мос келмаслиги керак.

Қуйида JavaScript нинг захираланган калит сўзлар келтирилган:

break	case	catch	class	continue	const

debugger	default	delete	do	else	enum
export	extends	false	finally	for	function
if	import	in	new	null	return
super	switch	this	throw	true	try
typeof	var	void	while	with	

Бу сўзлар орасида JavaScript тилида ва унинг ривожланишида ўзлаштириш режалаштирилмоқда.

Ўзгарувчининг қийматини ўзлаштириш
"=" ўзлаштириш оператори ёрдамида ўзгарувчилар қиймати ўзлаштирилади. Мисол қилиб ўуйидаги ўзгарувчи келтирилган ва унда матнли қатор ёзилган:

```
var MyHelloMsg;

MyHelloMsg = "Hello, world!";
```

MyHelloMsg сонли ўзгарувчини дастурнинг хохлаган жойида ўзлаштириш мумкин, мисол учун:

```
MyHelloMsg = 4;
```

Бу оператор бажарилгандан сўнг ўзгарувчи типи ўзгаради, шунингдек интерпретация жараёнида браузер хеч қандай огохлантирувчи хабарларни юбормайди.

Ўзгарувчини махсус null қиймати орқали ўзлаштириш мумкин:

```
MyHelloMsg = null;
```

Бундай ўзлаштириш хеч қандай типда ўзгарувчини белгиламайди.

JavaScript да маълумотлар типи
JavaScript тилида бир нечта маэлумотлар типи мавжуж. Булар сонлар, матнли қаторлар, мантиқий маэлумотлар, объектлар, аниқланмаган типли маэлумотлар, хамда махсус тип null.

Сонлар
JavaScript тили хар хил форматдаги сонларни ишлатишга рухсат беради, булар бутун сонлар, сузувчи нуқтали ўнли форматдаги сонлар ва илмий нотация сонлар. Бутун сонлар 8, 10, 16 асосида берилиши мумкин. мисол учун:

```
25        10 асосидаги бутун сон
```

0137 8 асосидаги бутун сон

0xFF 16 асосидаги бутун сон

386.7 Сузувчи ўнли нуқтали сон

25e5

или 25E5 Илмий нотациядаги сон, 2500000 га тенг.

Айрим холларда "сон бўлмаган" арифметик функциялар келиб чиқиши мумки. JavaScript да айтилганидек NaN (Not a Number). "Сон бўлмаган" – бу хеч қандай сонга лойиқ бўлмаган махсус қиймат. Бу сонлар устида операция бажарилаётган вақтда, ва натижа сон кўринишида тақдим этилмаган холларда пайдо бўлади. "Сон бўлмаган" қийматга тўғри келишини isNaN функцияси ёрдамида текшириш мумкин.

Матнли қатор
Матнли қатор – бу бир ёки қўштирноқ кетма кетлик белгиси, мисол учун:

"Hello, world!"

""

"12345"

'Бу матнли қатор'

"" қатори –бўшдир. Қуйидаги 2 та ўзлаштириш эквивалент эмаслигини аниқлаймиз:

MyStr=""

MyStr1=null

Биринчи холда MyStr ўзгарувчисида матнли қатор сақланмоқда (бўш бўлса хам), иккинчисида эса хеч нарса.

Мантиқий маълумотлар
Мантиқий маълумотлар фақат 2 та қийматни, яъни True ва False ни ўз ичига олади. Бу қийматлар 0 ва 1 сонлар билан боғлиқ эмас. Бу қийматларнинг асосий образи солиштириш операцияси бажарилаётган вақтга қаратилган, хамда шартли операциялар ишлатилганда хам.

Аниқланмаган типли маълумотлар

Агар ўзгарувчи эълон қилинган бўлса, аммо унга хали қиймат ўзлаштирилмаган бўлса, у холда у аниқланмаган типга бўлади. Мисол учун қуйидаги қаторда аниқланмаган типга эга бўлган MyVariable ўзгарувчиси эълон қилинган: ***var MyVariable;***

Агарда бу ўзгарувчини null қиймати билан ўзгартирсак, у холда ўзгарувчи типи ўзгаради ва null қийматга эга бўлган ўзгарувчига айланади: ***MyVariable = null;***

Маълумотлар типини ўзгартириш

Агарда ифодаларда хар хил типли ўзгарувчилар учраб қолса, JavaScript интерпретатори автоматик холда сонли маэлумотларни матнли қаторларга ўзгартириши мумкин. Тескари айлантиришни (қаторни-сонга) махсус функциялар ёрдамида, яъни parseInt ва parseFloat функциялари ёрдамида ўзгартириш мумкин. Буни қуйидаги мисол орқали кўриш мумкин:

Мисол

```
<HTML>
<HEAD><TITLE>Type conversion sample</TITLE></HEAD>
<BODY>
<H1>Type conversion sample</H1>
<TABLE>
<SCRIPT LANGUAGE="JavaScript">
var MyTextBuf = "";
MyTextBuf = 4 + " – тўрт сони" + "<BR>";
MyBuf2 = (parseInt("2") + 2) + "  - тўрт сони" + "<BR>";
document.write(MyTextBuf + MyBuf2);
</SCRIPT>
</TABLE></BODY></HTML>
```

Бу ерда биз ***MyTextBuf*** ўзгарувчисини эълон қилдик ва уни бўш қатор билан инициализация қилдик. Қуйида биз бу қаторда 4 сонни суммасини ва 2 та матнли қаторни ўзлаштирдик.

MyTextBuf = 4 + " – тўрт сони" + "
";

Бу ифода хисобланаётган вақтда 4 қиймат автоматик холда матнли қаторга ўзгаради. Кейинги йиғинди HTML хужжатларида ишлатиладиган   ажралмайдиган пробел белгисига ахамият бериш лозим. Агарда уни оддий пробелга алмаштирсак, қатор конкатенациясидан кейин бу пробел йўқолади. Кейинги қаторда parseInt функцияси ёрдамида матнли қатор “2” сонли қаторга ўзгаради, натижада 2 сони қўшилади, ундан кейин хар иккала матнли қаторларда конкатенация бажарилади:

MyBuf2 = (parseInt("2")+2)+" – тўрт сони"+ "
";

Натижада MyBuf2 ўзгарувчиси MyTextBuf.ўзгарувчиси каби қаторга эга бўлади.

JavaScript тили операторлари

Унар оператори

Унар оператори белгининг ўзгариши учун тўлдириш операциясини бажаришда, инкраментда хамда декрементда ишлатилади:

- тескари холатда белгининг ўзгариши
! Қушимча. Мантиқий ўзгарувчиларнинг қийматини реверсирования қилиш учун ишлатилади.
++ Ўзгарувчи қийматини ошириш. Ўзгарувчи префикси ёки унинг суффикси бўлиб қўлланиши мумкин.
-- Ўзгарувчи қийматини камайтириш. Ўзгарувчи префикси ёки унинг суффикси бўлиб қўлланиши мумкин.

Унар операторини ишлатишга доир мисоллар:
```
i=0;      // i тенг 0 даги ўзгарувчининг бошланғич қиймати
i++;      // i тенг 1 даги қиймат
--i;      // i тенг 0 даги қиймати
var j=3;  // j тенг 3 даги ўзгарувчининг қиймати
i = -j;   // i тенг -3 даги ўзгарувчининг қиймати
var fYes = true; // fYes тенг true даги ўзгарувчининг қиймати
testFlag(!fYes); // testFlag функциясига false қиймати узатилмокда
```

Бинар оператори

Бинар оператори 2 та операндни бирлаштиради. JavaScript тилида бинар операторлари айириш, бўлиш, қўшиш, кўпайтириш хамда бўлинмани қолдиғини хисоблаш учун ишлатилади (кўрилади):
- Айириш
+ Қўшиш
* Кўпайтириш
/ Бўлиш
% Бўлинмани қолдиғини хисоблаш

Бу операторлар С тилида ишлатилганидек JavaScript да хам худди шундай ишлатилади, мисол учун:
```
i=0;        // i тенг 0 даги ўзгарувчининг қиймати
i = i + 1;  // i тенг 1 даги қиймат
```

```
var j=9;    // j тенг 9 даги ўзгарувчининг қиймати
i = j / 2;  // i тенг 4 даги ўзгарувчининг қиймати
k = j % 2;  // i тенг 1 даги ўзгарувчининг қиймати
```

Алохида битлар билан ишлаш операторлари

Сценарияларда шундай операторлар ишлатиладики, улар алохида битлар билан ишлаш операторлари хисобланади, улар қуйидагилар: И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ, НЕ:

```
&   И
|   ИЛИ
^   ИСКЛЮЧАЮЩЕЕ ИЛИ
~   НЕ
```

Силжувчи операторлари

JavaScript да силжиш операцисини бажариш учун 3 та оператор курилган:

```
>>   Силжиш ўнг томонга
<<   Силжиш чап томонга
>>>  Бўшатиладиган разрядларни ноллар билан тўлдириб ўнгга силжиш
```

Мунособат операторлари

Мунособат операторлари ўзгарувчиларнинг қийматини солиштириш учун ишлатилади. Бу операторлар солиштириш натижаларига боғлиқлик true ёки false мантиқий қийматларни қайтаради ва шартли операторларда асосий бўлиб ишлатилади. True қийматини қайтарадиган JavaScript тилининг мунособат операторлари кўрсатилган:

```
>    Чап операнд ўнг операнддан катта
>=   Чап операнд ўнг операнддан катта ёки тенг
<    Чап операнд ўнг операнддан кичик
<=   Чап операнд ўнг операнддан кичик ёки тенг
==   Чап операнд ўнг операндга тенг
!=   Чап операнд ўнг операндга тенг эмас
```

Мантиқий операторлар

```
||   ИЛИ оператори. True қиймат қайтаради, қачонки операндлардан бири true бўлса.
&&   И оператори. True қиймат қайтаради, қачонки икки операнд true бўлса
```

Ўзлаштириш оператори

Ўзлаштириш оператори ўзгарувчиларнинг қийматини ўзлаштириш учун ишлатилади. JavaScript тилида ва C дастурлаш тилидаги каби бу оператор бошқа операторлар билан комбинациясига рухсат этилади. Қуйида ўзлаштириш операторини бошқа операторлар билан комбинацияси берилган:

= Оддий ўзлаштириш
+= Сонли қийматни катталаштириш ёки қаторларни қўшилиши
-= Сонли қийматни кичиклаштириш
*= Кўпайтириш
/= Бўлиш
%= Бўлишдан қолган қолдиқни хисоблаш
>>= Ўнгга силжиш
>>>= Бўшатиладиган разрядларни ноллар билан тўлдириб ўнгга силжиш
<<= Чапга силжиш
|= ИЛИ
&= И
^= ИСКЛЮЧАЮЩЕЕ ИЛИ

С тили билан таниш бўлмаганлар учун ўзлаштириш операторини бошқа операторлар билан биргаликда ишлатилиши қийинроқ ва ғайриоддий туйилиш мумкин, лекин аслида сценарийни осонлаштирадибошланғич текстни соддалаштиради.

Масалан сонли ўзгарувчилар қийматини ошириш учун += оператори ишлатилади. Аввал бу вазифани ечимини += операторини ишлатмаган холатда кўриб чиқамиз. Қуйида nCounter ўзгарувчиси эълон қилинди ва унга бошланғич 1 қиймати ўзлаштирилди, сўнг бу қиймат 5 га оширилди:

```
var nCounter = 1;
nCounter = nCounter + 5;
```

Энди буни += оператори ёрдамида бажарамиз:

```
var nCounter = 1;
nCounter += 5;
```

Кўриниб турибдики 2-усул 1-усулга нисбатан қисқа.
Ўзгарувчи қийматини 3 разрядга ўнгга силжитиш учун >>= рператоридан фойдаланиш мумкин ва у қуйидаги матнда кўрсатилган:

```
nCounter >>= 3;
```

Натижа эса қуйидаги матнда кўрсатилганидек бўлади:

```
nCounter = nCounter >> 3;
```

JavaScript тилида функция

Бошланғич матн бўлагини функция кўринишида ёзиш мумкин ва уларни JavaScript сценарийсининг турли жойларидан мурожаат қилиш мумкин. Одатда функциялар HTML документини сарлавха бўлимида аниқланади. Функциялар чақирилишидан аввал эълон қилиниши керак ва барча функция эълони HTML документ сарлавхасида жойлаштирилган бўлиши керак.

Функциянинг умумий эълони қуйида келтирилган:

```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{ . . .
```

Функция матни қаторлари
. . .
[return қиймат]
}

Барча параметрлар функцияга қийматига берилади. Шунинг учун функция унга параметр сифатида бериладиган ўзгарувчилар қийматини ўзгартира олмайди.

Return калит сўзи ёрдамида функция қиймати қайтарилади.

Такрорлаш учун саволлар:

1. Ўзгарувчилар нима ва уларни вазифаси?
2. Ўзгарувчи типини тушунтириб беринг?
3. Қандай JavaScript тили операторларини биласиз?
4. JavaScript тили функцияси нима ва унинг вазифаси?

11-маъруза. JavaScript дастурларида жараёнларни бошқариш элементлари

Режа:
1. Шарт операторлари.
2. Бошқарув ўтказувчи операторлар
3. Цикллар.
4. switch оператори.

Калит сўзлар: шартли ва бошқарув ўтказувчи операторлар, цикл операторлари, switch оператори

Шарт операторлари

if оператори.

Бу оператор **JavaScript** дастурлаш тилидаги муҳим операторлардан биридир. У шартга боғлиқ равишда код фрагментини бажаришга мўлжалланган. *if* операторининг структурасини қуйидагича ифодалаш мумкин:

1- вариант. if оператори синтаксиси:
if(шарт)
{
Амаллар кетма кетлиги
}

2- вариант. if оператори синтаксиси:
if(шарт)
{
Амаллар кетма кетлиги 1

```
}
else
{
Амаллар кетма кетлиги 2
}
```

Бу ерда **шарт** JavaScript тилидаги мантикий еки ихтиёрий шартдир. Агарда шарт мантикий типдаги узгарувчи булса, қиймати рост (True) бўлса, у ҳолда *Амаллар кетма кетлиги1* бажарилади. Акс ҳолда *Амаллар кетма кетлиги2* бажарилади.

Мисол. if шарт оператори

```
function checkData()
{
if (document.form1.threeChar.value.length==3)
{return true;
}
else
{  alert('3 ракамини киритинг');
return false;
}
}
```

else оператори

Биз юқорида фақат *if* операторининг асосий қисминигина кўрдик. Бу операторнинг бир нечта кенгайган шакли мавжуд. *else* оператори *if* операторида текширилаётган ифода нотўғри бўлган ҳолатдагина кенгайтиради ҳамда бу ҳолатда янги шартда бирор амал бажаради.

else оператори ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (шартли ифода) *бажариладиган_блок1*
else *бажариладиган_блок2*

Бу if...else конструкцияси қуйидагича интерпретация қилиниши мумкин: агар шарт бажарилса (яъни ифода=true), у ҳолда *бажариладиган_блок1* даги амаллар бажарилади, акс ҳолда *бажариладиган_блок2*даги амаллар бажарилади. *else* операторидан фойдаланиш мажбурий эмас.

elseif оператори.

if шарт операторининг яна бир кенгайган шакли – бу *elseif* операторининг қўлланилишидир. *elseif* – бу *else* ҳамда *if* операторларининг комбинациясидир. У худди *else* оператори каби *if* операторида шарт бажарилмаган ҳолда кенгайтиради. Бироқ *else* операторидан фарқи бир-бирига зид амалларни фақат агарда ***elseif*** шарт рост бўлгандагина бажаради. *else* ҳамда *elseif* операторлари ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (ифода1) *бажариладиган_блок1*

elseif (ифода2) *бажариладиган_блок2*
else бажариладиган_блокN

elseif операторлари битта *if*-блокида бир неча марта учраши мумкин. *elseif* тасдиғи фақат олдинда турган *if*-шартлари ҳамда *elseif*-шартлари False қийматни, берилган *elseif*-шарти эса True қийматни қайтаргандагина бажарилади.

Мисол. elseif оператори

```
fanction makeMinutes() {
   var minstring="";
   var now = new Date();
   var min = Date.getMinutes();
   if (min<10) {
     minstring+=":0"+min;}
   elseif(min>10){
     minstring+=":"+min;}
   return minstring;
}
```

switch операторы.

Яна бир шартни текшириб турли амалларга боғлиқ равишда иш кўрсатадиган конструкция бу – *switch* операторидир. Бу операторни узбек тилига таржима қилинганда “йўналишни ўзгартиргич” маъносини беради ҳамда бу операторнинг вазифаси ҳам шунга ўхшашдир. Ўзгарувчини қандай қийматни қабул қилишига боғлиқ равишда у йўналишни ўзгартириб турли блоклардаги амалларни бажаради. *switch* оператори *if...elseif...else* ёки *if* оператори мажмуига жуда ўхшаш бўлади. *switch* операторининг структурасини қуйидагича ифодалаш мумкин:

```
switch (ифода ёки ўзгарувчи){
case қиймат1:
   амаллар_блоки1
break;
case қиймат2:
   амаллар_блоки2
break;
...
default:
   амаллар_блоки_автоматик_тарзда
}
```

if операторидан фарқли томони бу ерда ифодалар мантиқий тип қабул қилмай, балки фақат *case* калит сўзидан кейинги қийматларни (*қиймат1*, *қиймат2* ва ҳ.к.) таққослайди холос. Агар ифода қиймати қандайдир вариант билан устма-уст тушса, икки нуқтадан кейинги *break* операторигача бўлган *амаллар_блоки*даги амалларни бажаради. Агарда ифода қиймати берилган

вариантлардан ҳеч бирига устма-уст тушмаса, *default* калит сўзидан кейинги автоматик тарзда бажариладиган блок (*амаллар_блоки_автоматик_тарзда*) бажарилади. *switch* операторидаги ифода фақат бир марта ҳисобланади, *elseif* операторида эса ҳар бир текширишда ҳисобланади, шунинг учун агарда ифода етарли даражада мураккаб бўлса, у ҳолда *switch* оператори тезроқ ишлайди.

Мисол 1: *switch* оператори:

```
function getName(){
var names = array("Name1"," Name2"," Name3");
var selected=“”;
switch (names[0]){
case "Name1":  selected="Name1 is selected";
break;
case "Name2":  selected="Name2 is selected";
break;
case "Name3":  selected="Name3 is selected";
break;
default: selected="Default select"+$names[0];
}
return selected;
}
```

Юкоридаги мисолдан куриниб турибдики, getName() функцияси ишга туширилганда, names массивидаги биринчи элемент текширилади. Бу холда массив 1-элементи Name1 булгани учун бизга selected="Name1 is selected"; кийматни кайтаради.

Мисол 2: *switch* оператори:

```
var change = prompt("Харакатни танланг:\n1 – Мошина сотиб олиш\n2 – Мошина сотиш \n3 – Мошина алмаштириш ");
switch (change) {
  case "1": {
    document.write("Сиз мошина сотиб олишни хохлайсиз ");
    break;
  }
  case "2": {
    document.write("Сиз мошина сотишни хохлайсиз ");
    break;
  }
  case "3": {
    document.write("Сиз мошина алмаштиришни хохлайсиз ");
    break;
  }
  default: {
    document.write("Сиз нотугри команда киритдингиз ");
```

```
    break;
  }
}
```

switch операторининг констркуцияси учун худди *if* оператори каби альтернатив синтаксиси мавжуд. Бу ерда *switch* операторидаги очиладиган фигурали қавс икки нуқтага ўзгартирилади, ёпиладигани эса мос равишда *endswitch*; калит сўзига ўзгартирилади.

Цикллар

JavaScript тилида шартга боғлиқ равишда қайтариладиган амаллардан иборат бир нечта конструкциялар мавжуд. Бу *while*, *do..while*, *foreach* ҳамда *for* цикллардир. Уларни батафсил кўриб чиқамиз.

while

Структураси:

while (ифода) { *бажариладиган_блок* }

ёки

while (ифода): *бажариладиган_блок* endwhile;

while – бу оддий цикл. У ифода қиймати True (бу ерда худди *if* оператори каби ифода мантиқий типга ўзлаштирилади) бўлгунича *бажариладиган_блок*даги буйруқларни бажаришга буюради. Ифода қиймати ҳар цикл бошланганда текшириб борилади, агарда унинг қиймати *бажариладиган_блок* бажарилиш жараёнида ўзгарган тақдирда ҳам итерация тугамагунча (яъни *бажариладиган_блок*даги барча буйруқлар бажарилмагунча) цикл тўтатилмайди.

Мисол: while оператори

```
i = 0
n = 0
while (i<5) {
 i ++;
if (i == 3)
   continue n + = i
  }
```

Мисол: while оператори

```
n = 0;
x = 0;
while( n < 3 ) {
  n ++;
  x += n;
}
```

do... while

do..while цикли *while* циклга ўхшайди, аммо фарқли томони шундаки, ифоданинг ростлигига цикл бошида эмас, балки охирида текширилади.

Қулай томони шундаки, *бажариладиган_блок* *do..while* цикли ичида ҳеч бўлмаганда бир марта бажарилади.

Структураси:

do { *do..while* цикли } while (ифода);

Мисол. do..while оператори

```
do {
  i+=1;
  document.write(i);
} while (i<5);
```

for

Бу JavaScript тили цикл операторларидан бири. Улар С дастурлаш тилидаги цикллар кабидир.

Структураси:

for (ифода1; ифода2; ифода3) { *бажариладиган_блок* }

ёки

for (ифода1; ифода2; ифода3): *бажариладиган_блок* endfor;

Бу ерда кўриниб турибдики шар учта ифодадан ташкил топади. Биринчи *ифода*1 ифода цикл бошида шартсиз бажарилади. Ҳар бир итерациянинг бошланишида *ифода*2 бажарилади. Агар у True қийматни қабул қилса, у ҳолда цикл ўз ишини давом эттиради ва *бажариладиган_блок*даги барча буйруқларни бажаради. Агар *ифода*2 False қийматни қабул қилса, у ҳолда цикл тўхтатилади. Ҳар бир итерация (яъни *бажариладиган_блок*даги барча буйруқларни бажарилишидан кейин) охирида *ифода*3 бажарилади.

Ҳар бир 1-,2- ва 3-ифодалар бўш бўлиши мумкин. Агар *ифода*2 бўш бўлса, бу циклни чексиз бажарилишини билдиради. Бу унчалик бефойда эмас, чунки циклни *break* оператори ёрдамида тўхтатса бўлади.

Мисол: Формада танланган элементлар сонини экранга чикариш

```
<SCRIPT>
function howMany(selectObject) {
  var numberSelected=0;
  for (var i=0; i < selectObject.options.length; i++) {
    if (selectObject.options[i].selected==true)
      numberSelected++;
  }
  return numberSelected;
}

</SCRIPT>
<FORM NAME="selectForm">
<P><B>Choose some music types, then click the button below:</B>
<BR><SELECT NAME="musicTypes" MULTIPLE>
```

```
<OPTION SELECTED> R&B
<OPTION> Jazz
<OPTION> Blues
<OPTION> New Age
<OPTION> Classical
<OPTION> Opera
</SELECT>
<P><INPUT TYPE="button" VALUE="How many are selected?"
onClick="alert('Number of options selected:'+howMany(document.selectForm. musicTypes))">
</FORM>
```

Агарда ушбу оператор ичидаги барча учала ифода ҳам тушириб қолдирилса, у ҳолда счётчик **var i** ўзгарувчини бошланғич қиймати берилмайди ва ҳар бир цикл охирида у ўзгармайди. Бу барча буйруқларни алоҳида буйруқлар кўринишида ёки циклдан аввал *бажариладиган_блок* ичида ёзса ҳам бўлади:

```
var i=0; // счётчикни бошланғич қийматини берамиз
for ( ; ; ){
   if (i>=10) break;
    // агар i катта ёки тенг 10 бўлса, у ҳолда цикл ишини тўхтатамиз.
   if (i % 2 == 0) print $i;
    // агар сон жуфт бўлса, уни экранга чиқарамиз.
i++; // счётчик қийматини биттага оширамиз.
}
```

for цикли конструкциясидаги учинчи ифодада вергулдан кейин яна бир нечта оддий буйруқларни ҳам ёзса бўлади. Масалан, агар биз оддийгина барча сонларни экранга чиқармоқчи бўлсак, дастурни қуйидагича ёзса бўлади:

```
for (i=0; i<10; document.write(i), i++)
/* Агарда бажариладиган_блок буйруқлардан ташкил топмаган
ёки битта буйруқдан ташкил топган бўлса,
фигурали қавсга олинган қисмни
ташлаб кетса бўлади.*/
```

Бошқарув ўтказувчи операторлар

Баъзида цикл ёки унинг алоҳида итерация ишини тезда тўхтатишга тўғри келади. Бунинг учун *break* ҳамда *continue* операторлари керак бўлади.

Break

Break оператори мавжуд циклни амалга оширишни тугаллайди, *for, while, do while* ёки *switch break* структурани бошқарувчи, цикл ёки шартни текширишни тугаллаш кераклигини билдирувчи, унинг таркибига кирувчи рақамли аргумент билан қўлланилади.

Мисол. Break оператори

```
for (i = 0; i < a.length; i++) {
  if (a[i] = theValue)
    break;
}
```

Ушбу скриптда a массив элементи theValue узгарувчи кийматига тенг булганда цикл тухтатилади.

continue

Баъзан цикл ишини бутунлай тўхтатиш лозим бўлмайди, фақатгина унинг янги итерациясини бошлаш керак. Continue *оператори исталган циклни амалга ошириш блокидан кейинги инструкцияларни ўтказиб юбориш ва янги доира билан амалга оширишни давом эттириш имконини беради.* continue *ни унинг таркибида бошқарилувчи конструкциялар ишини якунлаш кераклигини кўрсатувчи рақамли аргумент тарзида ишлатиш мумкин.*

Олдинги параграфда берилган мисолдаги break *операторини* continue *га алмаштирамиз. Бундан ташқари тўрт цикли миқдорини камайтирамиз.*

Мисол:

```
i = 0;
n = 0;
while (i < 5) {
  i++;
  if (i == 3)
    continue;
  n += i;
}
```

Бу скриптда агарда i узгарувчи i=3 кийматга эга булса, у холда n=(1,3,7,12) кийматларга эга булади.

Мисол

```
checkiandj :
  while (i<4) {
    document.write(i + "<BR>");
    i+=1;
    checkj :
      while (j>4) {
        document.write(j + "<BR>");
        j-=1;
```

```
        if ((j%2)==0)
          continue checkj;
        document.write(j + " is odd.<BR>");
      }
    document.write("i = " + i + "<br>");
    document.write("j = " + j + "<br>");
  }
```

Ушбу циклда агарда continue оператори топилса, checkj цикли тухтатилади ва checkj нинг кейинги циклига утилади. Хар сафар continue топилганда checkj итерацияни бошидан бошлайди, токи унинг шарти false булмагунича. checkj шарти false булганда, checkiandj нинг колган операторлари бажарилади ва бу цикл хам checkiandj шарти false булгунига кадар давом этади.

Такрорлаш учун саволлар

1. Шартли операторлар ишлаши ва вазифасини тушунтириб беринг.
2. Бошқарув ўтказувчи операторлар нима?
3. Қанақа цикл операторларини биласиз ва улар қандай ишлайди?
4. switch оператори ишлаш принципи.

12-маъруза. JavaScript дастурларида жараёнларни бошқариш элементлари

Режа:
1. Шарт операторлари.
2. Бошқарув ўтказувчи операторлар
3. Цикллар.
4. switch оператори.

Калит сўзлар: шартли ва бошқарув ўтказувчи операторлар, цикл операторлари, switch оператори

Шарт операторлари

if оператори.

Бу оператор **JavaScript** дастурлаш тилидаги муҳим операторлардан биридир. У шартга боғлиқ равишда код фрагментини бажаришга мўлжалланган. *if* операторининг структурасини қуйидагича ифодалаш мумкин:

1- вариант. if оператори синтаксиси:

```
if(шарт)
{
Амаллар кетма кетлиги
}
```

2- вариант. if оператори синтаксиси:

```
if(шарт)
{
Амаллар кетма кетлиги 1
}
else
{
Амаллар кетма кетлиги 2
}
```

Бу ерда **шарт** JavaScript тилидаги мантикий еки ихтиёрий шартдир. Агарда шарт мантикий типдаги узгарувчи булса, қиймати рост (True) бўлса, у ҳолда *Амаллар кетма кетлиги1* бажарилади. Акс ҳолда *Амаллар кетма кетлиги2* бажарилади.

Мисол. if шарт оператори

```
function checkData()
{
if (document.form1.threeChar.value.length==3)
{return true;
}
```

```
else
{  alert('3 ракамини киритинг');
return false;
}
}
```

else оператори

Биз юқорида фақат *if* операторининг асосий қисминигина кўрдик. Бу операторнинг бир нечта кенгайган шакли мавжуд. *else* оператори *if* операторида текширилаётган ифода нотўғри бўлган ҳолатдагина кенгайтиради ҳамда бу ҳолатда янги шартда бирор амал бажаради.

else оператори ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (шартли ифода) *бажариладиган_блок1*
else *бажариладиган_блок2*

Бу if...else конструкцияси қуйидагича интерпретация қилиниши мумкин: агар шарт бажарилса (яъни ифода=true), у ҳолда *бажариладиган_блок1* даги амаллар бажарилади, акс ҳолда *бажариладиган_блок2*даги амаллар бажарилади. *else* операторидан фойдаланиш мажбурий эмас.

elseif оператори.

if шарт операторининг яна бир кенгайган шакли – бу *elseif* операторининг қўлланилишидир. *elseif* – бу *else* ҳамда *if* операторларининг комбинациясидир. У худди *else* оператори каби *if* операторида шарт бажарилмаган ҳолда кенгайтиради. Бироқ *else* операторидан фарқи бир-бирига зид амалларни фақат агарда ***elseif*** шарт рост бўлгандагина бажаради. *else* ҳамда *elseif* операторлари ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (ифода1) *бажариладиган_блок1*
elseif (ифода2) *бажариладиган_блок2*
else бажариладиган_блокN

elseif операторлари битта *if*-блокида бир неча марта учраши мумкин. *elseif* тасдиғи фақат олдинда турган *if*-шартлари ҳамда *elseif*-шартлари False қийматни, берилган *elseif*-шарти эса True қийматни қайтаргандагина бажарилади.

Мисол. elseif оператори

```
fanction makeMinutes() {
   var minstring="";
   var now = new Date();
   var min = Date.getMinutes();
   if (min<10) {
     minstring+=":0"+min;}
   elseif(min>10){
     minstring+=":"+min;}
   return minstring;
}
```

switch оператори.

Яна бир шартни текшириб турли амалларга боғлиқ равишда иш кўрсатадиган конструкция бу – *switch* операторидир. Бу операторни узбек тилига таржима қилинганда “йўналишни ўзгартиргич” маъносини беради ҳамда бу операторнинг вазифаси ҳам шунга ўхшашдир. Ўзгарувчини қандай қийматни қабул қилишига боғлиқ равишда у йўналишни ўзгартириб турли блоклардаги амалларни бажаради. *switch* оператори *if...elseif...else* ёки *if* оператори мажмуига жуда ўхшаш бўлади. *switch* операторининг структурасини қуйидагича ифодалаш мумкин:

```
switch (ифода ёки ўзгарувчи){
case қиймат1:
   амаллар_блоки1
break;
case қиймат2:
   амаллар_блоки2
break;
...
default:
   амаллар_блоки_автоматик_тарзда
}
```

if операторидан фарқли томони бу ерда ифодалар мантиқий тип қабул қилмай, балки фақат *case* калит сўзидан кейинги қийматларни (*қиймат1*, *қиймат2* ва ҳ.к.) таққослайди холос. Агар ифода қиймати қандайдир вариант билан устма-уст тушса, икки нуқтадан кейинги *break* операторигача бўлган *амаллар_блоки*даги амалларни бажаради. Агарда ифода қиймати берилган вариантлардан ҳеч бирига устма-уст тушмаса, *default* калит сўзидан кейинги автоматик тарзда бажариладиган блок (*амаллар_блоки_автоматик_тарзда*) бажарилади. *switch* операторидаги ифода фақат бир марта ҳисобланади, *elseif* операторида эса ҳар бир текширишда ҳисобланади, шунинг учун агарда ифода етарли даражада мураккаб бўлса, у ҳолда *switch* оператори тезроқ ишлайди.

Мисол 1: *switch* оператори:

```
function getName(){
```

```
var names = array("Name1"," Name2"," Name3");
var selected=“”;
switch (names[0]){
case "Name1":  selected="Name1 is selected";
break;
case "Name2":  selected="Name2 is selected";
break;
case "Name3":  selected="Name3 is selected";
break;
default: selected="Default select"+$names[0];
}
return selected;
}
```

Юкоридаги мисолдан куриниб турибдики, getName() функцияси ишга туширилганда, names массивидаги биринчи элемент текширилади. Бу холда массив 1-элементи Name1 булгани учун бизга selected="Name1 is selected"; кийматни кайтаради.

Мисол 2: *switch* оператори:

```
var change = prompt("Харакатни танланг:\n1 – Мошина сотиб олиш\n2 – Мошина сотиш \n3 – Мошина алмаштириш ");
switch (change) {
  case "1": {
    document.write("Сиз мошина сотиб олишни хохлайсиз ");
    break;
  }
  case "2": {
    document.write("Сиз мошина сотишни хохлайсиз ");
    break;
  }
  case "3": {
    document.write("Сиз мошина алмаштиришни хохлайсиз ");
    break;
  }
  default: {
    document.write("Сиз нотугри команда киритдингиз ");
    break;
  }
}
```

switch операторининг констркуцияси учун худди *if* оператори каби альтернатив синтаксиси мавжуд. Бу ерда *switch* операторидаги очиладиган фигурали қавс икки нуқтага ўзгартирилади, ёпиладигани эса мос равишда *endswitch*; калит сўзига ўзгартирилади.

Цикллар

JavaScript тилида шартга боғлиқ равишда қайтариладиган амаллардан иборат бир нечта конструкциялар мавжуд. Бу *while*, *do..while*, *foreach* ҳамда *for* цикллардир. Уларни батафсил кўриб чиқамиз.

while

Структураси:

while (ифода) { *бажариладиган_блок* }

ёки

while (ифода): *бажариладиган_блок* endwhile;

while – бу оддий цикл. У ифода қиймати True (бу ерда худди *if* оператори каби ифода мантиқий типга ўзлаштирилади) бўлгунича *бажариладиган_блок*даги буйруқларни бажаришга буюради. Ифода қиймати ҳар цикл бошланганда текшириб борилади, агарда унинг қиймати *бажариладиган_блок* бажарилиш жараёнида ўзгарган тақдирда ҳам итерация тугамагунча (яъни *бажариладиган_блок*даги барча буйруқлар бажарилмагунча) цикл тўтатилмайди.

Мисол: while оператори

```
i = 0
n = 0
while (i<5) {
 i ++;
if (i == 3)
   continue n + = i
  }
```

Мисол: while оператори

```
n = 0;
x = 0;
while( n < 3 ) {
  n ++;
  x += n;
}
```

do... while

do..while цикли *while* циклга ўхшайди, аммо фарқли томони шундаки, ифоданинг ростлигига цикл бошида эмас, балки охирида текширилади. Қулай томони шундаки, *бажариладиган_блок* *do..while* цикли ичида ҳеч бўлмаганда бир марта бажарилади.

Структураси:

do { *do..while* цикли } while (ифода);

Мисол. do..while оператори

```
do {
  i+=1;
  document.write(i);
} while (i<5);
```

for

Бу JavaScript тили цикл операторларидан бири. Улар С дастурлаш тилидаги цикллар кабидир.

Структураси:

for (ифода1; ифода2; ифода3) { *бажариладиган_блок* }

ёки

for (ифода1; ифода2; ифода3): *бажариладиган_блок* endfor;

Бу ерда кўриниб турибдики шар учта ифодадан ташкил топади. Биринчи *ифода*1 ифода цикл бошида шартсиз бажарилади. Ҳар бир итерациянинг бошланишида *ифода*2 бажарилади. Агар у True қийматни қабул қилса, у ҳолда цикл ўз ишини давом эттиради ва *бажариладиган_блок*даги барча буйруқларни бажаради. Агар *ифода*2 False қийматни қабул қилса, у ҳолда цикл тўхтатилади. Ҳар бир итерация (яъни *бажариладиган_блок*даги барча буйруқларни бажарилишидан кейин) охирида *ифода*3 бажарилади.

Ҳар бир 1-,2- ва 3-ифодалар бўш бўлиши мумкин. Агар *ифода*2 бўш бўлса, бу циклни чексиз бажарилишини билдиради. Бу унчалик бефойда эмас, чунки циклни *break* оператори ёрдамида тўхтатса бўлади.

Мисол: Формада танланган элементлар сонини экранга чикариш

```
<SCRIPT>
function howMany(selectObject) {
  var numberSelected=0;
  for (var i=0; i < selectObject.options.length; i++) {
    if (selectObject.options[i].selected==true)
      numberSelected++;
  }
  return numberSelected;
}

</SCRIPT>
<FORM NAME="selectForm">
<P><B>Choose some music types, then click the button below:</B>
<BR><SELECT NAME="musicTypes" MULTIPLE>
<OPTION SELECTED> R&B
<OPTION> Jazz
<OPTION> Blues
<OPTION> New Age
<OPTION> Classical
<OPTION> Opera
```

```
</SELECT>
<P><INPUT TYPE="button" VALUE="How many are selected?"
onClick="alert('Number of options selected:'+howMany(document.selectForm. musicTypes))">
</FORM>
```

Агарда ушбу оператор ичидаги барча учала ифода ҳам тушириб қолдирилса, у ҳолда счётчик **var i** ўзгарувчини бошланғич қиймати берилмайди ва ҳар бир цикл охирида у ўзгармайди. Бу барча буйруқларни алоҳида буйруқлар кўринишида ёки циклдан аввал *бажариладиган_блок* ичида ёзса ҳам бўлади:

```
var i=0; // счётчикни бошланғич қийматини берамиз
for ( ; ; ){
   if (i>=10) break;
    // агар i катта ёки тенг 10 бўлса, у ҳолда цикл ишини тўхтатамиз.
   if (i % 2 == 0) print $i;
    // агар сон жуфт бўлса, уни экранга чиқарамиз.
i++; // счётчик қийматини биттага оширамиз.
}
```

for цикли конструкциясидаги учинчи ифодада вергулдан кейин яна бир нечта оддий буйруқларни ҳам ёзса бўлади. Масалан, агар биз оддийгина барча сонларни экранга чиқармоқчи бўлсак, дастурни қуйидагича ёзса бўлади:

```
for (i=0; i<10; document.write(i), i++)
/* Агарда бажариладиган_блок буйруқлардан ташкил топмаган
ёки битта буйруқдан ташкил топган бўлса,
фигурали қавсга олинган қисмни
ташлаб кетса бўлади.*/
```

Бошқарув ўтказувчи операторлар

Баъзида цикл ёки унинг алоҳида итерация ишини тезда тўхтатишга тўғри келади. Бунинг учун *break* ҳамда *continue* операторлари керак бўлади.

Break

Break оператори мавжуд циклни амалга оширишни тугаллайди, *for, while, do while* ёки *switch break* структурани бошқарувчи, цикл ёки шартни текширишни тугаллаш кераклигини билдирувчи, унинг таркибига кирувчи рақамли аргумент билан қўлланилади.

Мисол. Break оператори

```
for (i = 0; i < a.length; i++) {
```

```
  if (a[i] = theValue)
    break;
}
```
Ушбу скриптда a массив элементи theValue узгарувчи кийматига тенг булганда цикл тухтатилади.

continue

Баъзан цикл ишини бутунлай тўхтатиш лозим бўлмайди, фақатгина унинг янги итерациясини бошлаш керак. Continue *оператори исталган циклни амалга ошириш блокидан кейинги инструкцияларни ўтказиб юбориш ва янги доира билан амалга оширишни давом эттириш имконини беради.* continue *ни унинг таркибида бошқарилувчи конструкциялар ишини якунлаш кераклигини кўрсатувчи рақамли аргумент тарзида ишлатиш мумкин.*

Олдинги параграфда берилган мисолдаги break *операторини* continue *га алмаштирамиз. Бундан ташқари тўрт цикли миқдорини камайтирамиз.*

Мисол:
```
i = 0;
n = 0;
while (i < 5) {
  i++;
  if (i == 3)
    continue;
  n += i;
}
```
Бу скриптда агарда i узгарувчи i=3 кийматга эга булса, у холда n=(1,3,7,12) кийматларга эга булади.

Мисол
```
checkiandj :
  while (i<4) {
    document.write(i + "<BR>");
    i+=1;
    checkj :
      while (j>4) {
        document.write(j + "<BR>");
        j-=1;
        if ((j%2)==0)
          continue checkj;
        document.write(j + " is odd.<BR>");
      }
    document.write("i = " + i + "<br>");
    document.write("j = " + j + "<br>");
  }
```

Ушбу циклда агарда continue оператори топилса, checkj цикли тухтатилади ва checkj нинг кейинги циклига утилади. Хар сафар continue топилганда checkj итерацияни бошидан бошлайди, токи унинг шарти false булмагунича. checkj шарти false булганда, checkiandj нинг колган операторлари бажарилади ва бу цикл хам checkiandj шарти false булгунига кадар давом этади.

Такрорлаш учун саволлар

1. Шартли операторлар ишлаши ва вазифасини тушунтириб беринг.
2. Бошқарув ўтказувчи операторлар нима?
3. Қанақа цикл операторларини биласиз ва улар қандай ишлайди?
4. switch оператори ишлаш принципи.

13-маъруза. IMAGE обьекти. DRAG ва DROP

Режа:

1. Ҳужжатда расмларни тасвирлаш
2. алтернатив матн ишлатиш
3. Add search from (қидирув формасини қўшиш) ўрнатиш

Калит сўзлар: SRC, HEIGHT, BORDER, Drop-Down.

Web-саҳифаларни безашда тасвирлар етакчи ўринни эгаллайди. Расмлар html-ҳужжатидан ташқарида алоҳида файлларда сақланади, лекин браузер ёрдамида web-саҳифа ичида жойлаштирилади.

1. Ҳужжатда расмларни тасвирлаш учун <IMG> тоқ теги ишлатилади.
2. Бу тег ҳар доим SRC мажбурий атрибутни ўзида мужассам этиши шарт.

Унинг қиймати бўлиб тасвир файлининг абсолют ёки нисбий шаклда ёзилган URL адреси ҳисобланади. Ҳужжат юкланганда расм ҳам юкланади ва ҳужжатнинг <IMG> теги келган жойида тасвирланади. Масалан,

<img src= “clock.jpg”>

3. Тасвир web-ҳужжатга ўлчамларини сақлаган ҳолда кўчирилади. Агар унинг ўлчамларини ўзгартириш зарурати туғилса, расмнинг керакли ўлчамларини WIDTH (кенглиги) va HEIGHT (баландлиги) атрибутлари ёрдамида бериш мумкин. Бу атрибутларнинг қиймати расмнинг web-саҳифадаги кенглиги ва баландлигини пикселларда ёки фоизларда аниқлайди. Масалан,

<img src=“clock.jpg” width=50 height=150>

Тасвирларни ҳужжатларда келтираётганимизда биз эҳтиёт бўлишимиз керак. Биринчидан, web-саҳифа тасвирларни кўрсата олмайдиган браузер ёрдамида юкланиши мумкин. Иккинчидан, фойдаланувчилар кўпинча расмларни кўрсатмаслик режимига қўйишлари мумкин (ҳужжатнинг тез юкланишини амалга ошириш учун). У ва бу ҳолда ҳам, агар расмни кўриш имконияти ё'қ бўлса, у ерда қандай расм бор эканлигини билдириш имкониятини бериш маҳсадга мувофиқдир. Шу мақсадда алтернатив матн ишлатилади.

4. Алтернатив матн – бу моҳияти жиҳатидан, тасвирни у ёки бу даражада изоҳлаб

берувчи жумладир. Агар браузер қандайдир сабабга кўра расмни кўрсатиб бермаса, у 31 ҳолда унинг ўрнига шу алтернатив матн кўрсатилади. Алтернатив матнни экранга

чиқариш учун <IMG> тегининг ичида махсус ALT атрибутининг қиймати келтирилади.

Масалан,

<img src=“clock.jpg” alt=“қўнғироқ чаладиган соатнинг расми”>

Тасвирни ҳам, матнга ўхшаб, кўрсатгич, ишора сифатида ишлатиш мумкин. Бунинг учун <IMG> tegi <A> жуфт теглари орасида келтирилиши лозим. Тасвир-ишора кўк рамкада тасвирланади. Бундай расмга сичқонча

кўрсатгичи олиб борилса, унинг шакли кўрсатувчи қўл шаклига айланади. Бу усул билан web-саҳифаларда ўтишнинг график тугмачалари ҳосил қилинади (1-расм):

1-расм. Ўтишнинг график тугмачаси ва алтернатив матн. Тасвир-ишоранинг рамкасиз бўлиши учун BORDER атрибутига нол қиймат берилади. Агар BORDER атрибутига сон берилса, рамканинг қалинлигини аниқлайди.
Масалан,
 <img src=“clock.jpg” border=0 alt=“қўнғироқ чаладиган соатнинг расми”>
Drop-Down List – one record per item (очилувчи рўйхат) – рўйхат бўлиб, унда барча ёзувлардан битта майдон қиймати тасвирланади. Бундай очилувчи
рўйхатни мавжуд Мбдан майдон формаси қийматини танлаш учун формага жойлаштиришингиз мумкин. Ундан ташқари, танлашда, масалан, ҳамкасбнинг исмини, форманинг берилган элементи шу исмни ўзини эмас, балки адреси ёки телефонини кўрсатиши мумкин. Тасвирланувчи майдон Дисплай валуес фром тҳис фиелд очилувчи рўйхатида, қайтарилувчи рўйхат эса Субмит валуес фром тҳис фиелд очилувчи рўйхатида кўрсатилади.
Масалан, ахборот тасвирланишининг жадвал вариантини танланг ва Нехт тугмасини босинг. Экранда мастернинг охирги диалоги ҳосил бўлади. Бу ерда ахборотлар узлюксиз блок (Дисплай алл ресордс тогетҳер) тасвирланадими ёки гуруҳлар бўйича (Сплит ресорд Дата гроупс) берилган ёзувлар миқдори бўйича бўлиб чиқиладими, танланади.
Агар Add search from (қидирув формасини қўшиш) ўрнатилса, саҳифага сайтга ташриф буюрувчининг ахборотни мустақил равишда қидириш ва танлаш имконини берувчи қидирув формаси жойлашади. Бу байроқча фақат 3-қадамда қидирув майдонлари танланган бўлса, ишлайди. Финиш тугмасини босинг. Саҳифага ахборотларни тасвирлаш учун жадвал жойлашади. Унда

устун сарлавҳаларни ўзгартириш мумкин. Кўпинча саҳифа файли кенгайтмасини .ASP (Active Server Page) форматига ўгириш керак бўлади (Save as...).

Сайтингиз Web-серверга жойлаштирилмагунча МБ ишини текшириш мумкин эмас. Агар сиз саҳифани кўриш учун браузерни юкласангиз, фақат устунлар сарлавҳаларини кўриш мумкин.

Юқоридагилардан ташқари Front PAGE яна параметрли гиперишораларни ҳосил қилиши мумкин. У ишлатилса, алоҳида МБ га мурожаат ташкил этилади. Масалан, МБ дан ҳамкасблар рўйхати саҳифага жойлаштирилган бўлса, ҳамкасбнинг исмини гиперишорага айлантириш мумкин. Бу гиперишорага мурожаат бўлганда, танланган инсон ҳақида қўшимча маълумотларни кўриш мумкин бўлади. Ундан ташқари яна сайтга ташриф буюрувчиларнинг ўзларига Мбга ахборотни киритиш имкониятини ташкил этиш мумкин.

Такрорлаш учун саволлар:

1. Ҳужжатда расмларни тасвирлаш қандай амалга оширилади?
2. алтернатив матн ишлатишга мисоллар келтиринг?
3. Add search from (қидирув формасини қўшиш) ўрнатиш?

14-маъруза. MOUSEDOWN, MOUSEMOVE ва MOUSEUP

Режа:

1. Drag ва Drop механизмини хосил килиш.
2. MouseMove ходисаси содир булиши.
3. Event MouseMove ни window capture Events да аниклаш.

Калит сўзлар: MouseDrag, Capture Events, MouseMove, StartDrag.

Айтиб утганимиздек Java S да MouseDrag қоидаси йук. Шунинг учун биз Drag ва Drop механизмини хосил килишда MouseDown, MouseMove ва MouseUp ходисаларидан фойдаланишимиз керак. Навбатдаги мисолимизда MouseMove ёрдамида сичкончанинг жорий координаталари холатлар сатрида намоён булади. Куришимиз мумкинки бу скрипт коди олдинги мисол билан бир хил:

```
< html >
< script lauguage = “ JavaScript " >
< ! - - hide
Window.Capture Events ( Event. MouseMove ) ;
Wihdow.On MouseMove = displayCoords;
function displayCoords (l) {
status = “x : “ + l.pagex + “ y : “ + l.page y;
}
//--
</script>
Сичконча координаталари холатлар сатрида намоён булади
</html>
```

Event.MouseMove ни ёзишда сиз MouseMove сузини бош харфлар билан ёзишингиз кераклигига эътиборни каратинг. MouseMove ходисаси содир булганда кайси функция чакирилишини курсатишга келганда эса сиз уни кичкина харфлар билан ёзишингиз керак булади:

```
Window.on mousemove = …
```

Энди биз охирги 2 та мисолни бирлаштиришимиз мумкин. Биз фойдаланувчи сичкончани тугмачасини босганда унинг координатаси пайдо булишини хохлаймиз. Мисолнинг коди куйидаги куринишда булади:

```
< html >
< script lauguage = “ JavaScript " >
< ! - - hide
Window.CaptureEvents ( Event.MouseDown/Event.MouseUp ) ;
Wihdow.On MouseDown = StartDrag;
Wihdow.On MouseUp = EndDrag;
Wihdow.On MouseMove = MoveIt;
function StartDrag (l) {
Window.CaptureEvents ( Event.MouseMove);
}
function MoveIt (l) {
// Координаталарни курсатиш
status = “x: “ + l.pagex + “ y : “ + l.page y;
}
function EndDrag (l) {
window.releaseEvents (Event.MouseMove);
}
//--
</script>
```

Сичконча тугмаисини босинг ва уни юбормаган холда сичкончани узини харакатлантиринг. Сичконча координаталари холатлар сатрида номаён булади.

```
</html>
```

1 чи дан биз window объектини MouseDown ва MouseUp ходисаси хакида сичкончани кабул килишга мажбур киламиз.

```
Window.CaptureEvents ( Event.MouseDown/Event.MouseUp ) ;
```

Куриб турганингиздек биз / (ёки) белгисидан window объекти курсатилган

ходисалардан бир нечтасини кабул килиш керак деган максадда фойдаланамиз. Курсатилган ходисалар жойга эга булса унда нима содир булишини куйидаги 2 та сатр оркали тавсифланади.

```
Wihdow.On MouseDown = StartDrag;
Wihdow.On MouseUp = EndDrag;
```

Куйидаги сатрда MouseMove ходисаси руй бериб window объекти сигнал кабул килганда нима содир булиши аник булган.

```
Wihdow.On MouseMove = MoveIt;
```

Лекин тухтанг ахир биз Event.MouseMove ни window.captureEvents ( ) да аникламадикку! Бу шуни англатадики жорий ходиса window объекти томонидан камраб олинмайди. Унда нимага агар window объекти бу ходиса хакида хеч нимага эга булмаса, биз window объектига MoveIt ( ) функциясини курсатамиз! Бу саволга жавобни MouseDown ходисаси содир булгандан сунг чакириладиган StartDrag ( ) функциясида топишингиз мумкин:

```
function StartDrag (l) {
Window.CaptureEvents ( Event.MouseMove);
}
```

Бу шуни билдирадики сичконча тугмачаси босилгандан дарров window объекти MouseMove ходисасини камираб олади. Ва агар MouseUp ходисаси содир булса биз MouseMove ходисасини тухтатишимиз керак. Бу EndDrag ( ) функциясида ReleaseEvents ( ) усули ёрдамида бажарилади.

```
function EndDrag (l) {
window.releaseEvents (Event.MouseMove);
}
```

MoveIt ( ) функцияси сичкончани координаталарини холатлар сатрига ёзади. Энди бизда Drag ва Drop механизмини амалга оширувчи скриптнинг барча элементлари бор. Ва биз энди объектларимизни экранда чизишни бошласак хам булади.

Такрорлаш учун саволлар:

1. Drag ва Drop механизмини хосил килишда нималарга етибор бериш керак?
2. MouseMove ходисаси содир булиши?
3. Event MouseMove ни window capture Events да аниклаш?

15-маъруза. PHP. Сервер томондан дастурлаш. PHP га кириш. PHP ни ўрнатиш ва тестлаш

Режа:
1. PHP га кириш.
2. PHP имкониятлари.
3. Дастурий воситани созлаш ва ўрнатиш.

Калит сўзлар: PHP дастурлаш тили, сервер томонда дастурлаш, денвер пакети

PHP га кириш

Ҳозирги кунда интернет кенг оммалашгани сабабли, замон тараққиётини веб-технологиясиз тассаввур этиш мумкин эмас. Веб технологияларига талаб ошган сари Web-дастурлаш тилларини билиш ҳар бир дастурчи учун муҳим вазифа саналмоқда. Шуларни инобатга олган ҳолда замонавий веб-дастурлаш тилларидан бири ҳисобланган, содда, ўрганишга қулай, барча маълумотлар базаси билан ишлай оладиган PHP ҳақида батавсилроқ тўхталишга ният қилдик. Келгусида бу тил ўзбек тилида ёритилиб борилади ҳамда мутахассис ва ўрганувчилар учун форум ташкил қилинади.

PHP тарихи. Кўпгина бошқа дастурлаш тилларидан фарқли равишда, PHP қандайдир ташкилот ёки кучли дастурчи томонидан яратилган эмас. Уни оддий фойдаланувчи Расмус Лердорф 1994 йили ўзининг бош саҳифасини интерактив услубда кўрсатиш учун яратган. Унга Personal Home Page (PHP – шахсий бош саҳифа) деб ном берган.

1995 йили Расмус PHPни ўзининг HTML формалари билан ишлайдиган бошқа дастур билан умумлаштириб PHP/FI Version 2 (”Form Interpretator“) ҳосил қилди. 1997 йилга бориб PHP дан фойдаланувчи сайтлар 50 мингдан ошди. Шундан сўнг веб технология усталари PHP ғояси асосида мукаммал тил яратишга Зива Сураски ва Энди Гутманс асосчилигида киришилди. PHPни самарали деб ҳисобланмагани учун деярли нолдан бошлаб, мавжуд C ва Перл тилларидан ибрат олиб PHP3 талқинини яратилди. 1999 йилга келиб PHP асосида қурилган сайтлар миллиондан ошиб кетди. 2000 йилда эса Zend Technologies ширкати янги кўпгина функцияларни қўшган ҳолда PHP4 шарҳловчисини яратди.

PHP – веб техноогия тили. PHPни ўрганиш учун аввал HTML ва дастурлаш тилидан ҳабардор бўлиш талаб қилинади. HTML/CSS ва JavaScript ларни мукаммал билгувчилар учун PHPни ўрганиш мураккаблик туғдирмайди. PHPнинг вазифаси HTML файлини яратиб бериш. JavaScript ёрдамида бажариладиган кўпгина операцияларни PHP орқали ҳам амалга ошириш мумкин, аммо эътибор қилиш лозимки, PHP – серверда; JavaScript – клиент томонда бажарилади. PHPда ёзилган код сервернинг ўзида бажарилиб, клиентга HTML шаклида етиб боради. Бу ҳавфсизлик жаҳатдан

анча мақсадга мувофиқ. JavaScript ёрдамида код ёзиш, маълумот узатиш ва қабул қилишни бироз тезлаштирса-да, кодни клиент кўриш имкониятига эга бўлади. Барибир ҳар иккисини бошқаси боса олмайдиган ўз ўрни бор, равшанки бу ўрин PHPда муҳимроқ ва каттароқ.

PHP имкониятлари

«PHP да ҳар қандай дастур бажарса бўлади», – деган эди унинг яратувчиси. Биринчи навбатда PHP тили сервер томонидан бажариладиган скриптлар яратиш учун фойдаланилади ва айнан шунинг учун у яратилган. PHP тили ихтиёрий CGI-скриптлари масалаларини ечишга ва бундан ташқари html формали маълумотларни қайта ишлашга ҳамда динамик равишда html саҳифаларни ишлаб чиқишга қодир. Бироқ PHP тили фойдаланиладиган бошқа соҳалар ҳам мавжуд. Бу соҳаларни биз учта асосий қисмга бўламиз:

Биринчи соҳа – биз юқорида айтиб ўтганимиздек, сервер томонидан бажариладиган иловалар (скриптлар) яратиш. PHP тили бундай турдаги скриптларни яратиш учун жуда кенг қўлланилади. Бундай иш кўрсатиш учун PHP-парсер (яъни php-скриптларни қайта ишловчи) ва скриптларни қайта ишловчи web-сервер, скриптларни натижасини кўриш учун браузер ва албатта php-кодини ёзиш учун қандай бўлса ҳам матн муҳаррири керак бўлади. PHP-парсер CGI-дастурлар кўринишида ёки сервер модуллари кўринишида тарқалган. Уни ва web-серверни компьютеримизга қандай ўрнатамиз, биз бу ҳақида кейинроқ кўриб ўтамиз.

Иккинчи соҳа – буйруқлар сатрида бажариладиган скриптларни яратиш. Яъни PHP тили ёрдамида бирор-бир компьютерда браузер ва web-серверлардан мустақил равишда ўзи бажариладиган скриптларни ҳам яратиш мумкин. Бу ишларни бажариш учун ҳеч бўлмаганда PHP-парсер (бу ҳолатда биз уни буйруқлар сатри интерпретатори (CLI, command line interpreter) деб атаймиз) талаб этилади. Бундай ишлаш услуби турли масалаларни режалаштириш ёрдамида бажарилиши учун керак бўлган скриптлар ёки оддий матнни қайта ишлаш учун керак бўлган масалага ўхшаш ишлайди.

Ва ниҳоят охирги учинчи соҳа – бу мижоз томонидан бажариладиган GUI-иловаларни (график интерфейс) яратиш. Бу соҳа PHP тилини эндигина ўрганаётган фойдаланувчилар учун унча муҳим бўлмаган соҳадир. Бироқ агарда сиз PHP тилини чуқур ўрганган бўлсангиз, бу соҳа сиз учун анча муҳимдир. PHP тилини бу соҳага қўллаш учун php кенгайтмали махсус ёрдамчи – PHP-GTK талаб этилади.

Шундай қилиб, PHP тилини қўлланилиш соҳалари кенг ва турличадир. Юқоридаги масалаларни еча оладиган бошқа турлича дастурлаш тиллари ҳам мавжуд, унда нима учун PHP тилини ўрганишимиз керак? У тил бизга нима беради? Биринчидан, PHP тили ўрганиш учун жуда қулай. PHP тилини синтаксиси асосий қоидалари ва ишлаш принципи билан етарлича танишиб чиқиб ўзингизни шахсий дастурингизни тузиб кўриб, сўнгра уни

бошқа дастурлаш тилларида тузилган вариантлари билан солиштирсангиз бунга гувоҳи бўласиз.

Иккинчидан, PHP тили барча бизга маълум платформаларда, барча операцион тизимларда ҳамда турлича серверларда эркин ишлай олади. Бу хусусият жуда муҳим. Масалан, кимдир Windows операцион тизимдан Linux операцион тизимга ёки IIS сервердан Apache серверга ўтмоқчи бўлса PHP тилини ўрганиши шарт.

PHP дастурлаш тилида дастурлашнинг иккита ҳаммабоп парадигмалари ишлатилади, булар процедурали ва объектли дастурлаш. PHP4 дастурлаш тили процедурали дастурлашни бутунлай қўллаб қувватлайди, бироқ объектли стилдаги дастурларни ҳам қўлласа бўлади. PHP5 дастурлаш тилининг биринчи тестлаш версиясида PHP4 дастурлаш тилида учрайдиган объектга йўналтирилган дастурлаш моделларининг камчиликлари тўлдирилган. Шундай қилиб, ҳозирда таниш бўлиб улгурган ишлаш принципини танлаш керак.

Агарда PHP тилини ҳозирги имкониятлари тўғрисида гаплашадиган бўлсак, у ҳолда биз PHP тилини биринчи версиясидан анча йироқлашиб кетган бўламиз. PHP дастурлаш тили ёрдамида тасвирлар, PDF-файллар, флэш-роликлар яратиш мумкин; ҳозирги вақтдаги замонавий маълумотлар базасини қўллаб қувватлайди; ихтиёрий матнли файл форматлари билан, ҳамда XML ва файллар тизими билан ишлайдиган функциялар ҳам қўшилган. PHP тили турли сервислар ўртасидаги протоколларнинг ўзаро алоқасини қўллаб қувватлайди. Буларга мисол тариқасида папкаларга киришни бошқариш протоколи LDAP, тармоқ қурилмалари билан ишлайдиган протокол SNMP, маълумотларни узатиш протоколлари IMAP, NNTP ҳамда POP3, гиперматнларни узатиш протоколи HTTP ва бошқаларни олиш мумкин.

PHP дастурлаш тилини турли дастурлаш тиллари ўртасидаги ўзаро алоқасига диққатни қаратсак, бунга Java дастурлаш тилини айтиб ўтиш керакки, Java дастурлаш тили объектларини PHP тили ўз объектлари сифатида қарайди. Объектларга мурожаат сифатида CORBA кенгайтмасидан фойдаланилади.

Матнли ахборотлар билан ишлаш учун PHP тили ўзига Perl дастурлаш тилидаги тартибланган ифодалар билан ишлай оладиган механизмларни (катта бўлмаган ўзгаришларсиз) ва UNIX-тизимини мерос қилиб олади. XML-ҳужжатларини қайта ишлаш учун стандарт сифатида DOM ва SAX, XSLT-трансформацияси учун API дан фойдаланиши мумкин.

Электрон тижорат иловаларини яратиш учун бир қатор тўловни амалга оширадиган Cybercash, CyberMUT, VeriSign Payflow Pro ҳамда CCVS каби фойдали функциялар мавжуд.

Дастурий воситани созлаш ва ўрнатиш

Юқорида PHP тили имкониятларини, қўлланилиш соҳаларини муҳокама қилдик ва тарихини ўргандик. Энди дастурий воситани ўрнатишга керак бўлган ускуналар мажмуини кўриб ўтсак. Модомики, асосий курснинг амалиёти сифатида биз қуйидаги масалаларни кўриб чиқамиз: клиент-сервер технологияси сифатида ишланадиган масалалар, мос равишда скриптлар яратилишида қўлланилиши, серверларни қайта ишлаш. Булар учун бизга web-сервер ҳамда PHP тили интерпретатори керак бўлади. Web-сервер сифатида, масалан, web-мутахассислар ўртасида машҳур бўлган Apache серверни оламиз. Дастур натижасини кўриш учун web-браузер керак бўлади, бунга мисол Internet Explorer.

Денвер Дистрибутиви

Биз юқорида Linux ва Windows платформалари учун PHP дастурий воситасини созлаш ва ўрнатиш билан етарлича танишмиз. PHP дастурий воситаси ва уни ишлаши учун керак бўладиган компоненталарни ўрганишни хоҳламайдиганлар учун PHP дастурининг тайёр PHP тилини тўлдирадиган дистрибутивлари мавжуд. Бундай дистрибутивлар ичида кенг тарқалгани - Денвер (http://dklab.ru/chicken/web/). Уни ўрнатишни ўрганиш учун web-мутахассислар сайтларига мурожаат қилиш керак. Денверни ўрнатиш жуда оддий ҳамда унга ҳеч қандай билим талаб этилмаслигини айтиб ўтиш керак. Бу дистрибутивни PHP тилини эндигина ўрганаётган ёш дастурчилар учун тавсия этамиз. Жиддий масалаларни ҳал этиш учун эса PHP дастурлаш тилини тўлиқ ўрнатиш ва созлаш керак бўлади.

PHP дастурлаш тили, сервер томонда дастурлаш, денвер пакети

Такрорлаш учун саволлар:

1. Қанақа дастурлаш тилларини биласиз?
2. PHP дастурлаш тили имкониятлари ҳақида гапиринг.
3. Сервер томонда дастурлаш деганда нимани тушунасиз?
4. PHP ни ишга тушириш, Денвер пакети ва ундан фойдаланиш.

16-маъруза. PHP асосий тузилиши. Маълумотлар типлари. Ифодалар.

Режа:
1. PHP асосий тузулиши
2. Асосий синтаксислар.
3. Маълумотлар типлари.
4. Альтернатив синтаксислар.

Калит сўзлар: PHP код синтаксиси, маълумот типлари, ифодалар, жараенларни бошқариш

PHP асосий тузулиши

Кўп ҳолларда *PHP* тилини интерпретатори ишлаётганлигини текшириб кўриш учун тузиладиган дастур энг содда дастур деб аталади. Ҳозир биз *PHP* тилидаги ушбу дастурни чуқур ўрганамиз ҳамда уни бошқа дастурлаш тиллари Си, Perl ва JavaScript лардан фарқли томонини текширамиз. Ушбу **мисолни** кўрамиз:

```
<html>
  <head>
  <title> Мисол </title>
  </head>
  <body>
  <?php
  echo "<p> Салом, бу мен – PHP скрипт! </p>";
  ?>
</body>
</html>
```

Бу *PHP* дастурлаш тилининг махсус кодли теглари ёрдамида тузилган содда html-файлдир.

Юқорида айтиб ўтганимиздек, *PHP* дастурлаш тили Си ва Perl дастурлаш тилига ўхшаш. Бироқ келтирилган дастур Си ва Perl дастурлаш тилидаги дастурдан анча катта фарқ қилади. Бу ерда HTML саҳифага чиқариш учун бир қатор махсус буйруқларни ёзиш шарт эмас. Бевосита *PHP*-код асосида қурилган бирор вазифани бажарадиган HTML-*скрипт* ёзилади (бизни мисолда экранда чиқарилган матн). *PHP* дастурлаш тилининг Си ва Perl дастурлаш тилларидан камчилиги шуки, мураккаб скриптларни *PHP* дастурлаш тили анча секин бажаради.

PHP-скриптлар – бу серверда бажариладиган ва қайта ишланадиган дастурлардир. Бу скриптларни JavaScript типидаги скриптлар билан таққослаш мумкин эмас, чунки JavaScript тилидаги скриптларда ёзилган буйруқлар фақат клиент компьютеридагина бажарилади. Клиент компьютерида ва сервер компьютерида бажариладиган скриптларнинг фарқи нимада? Агарда скрипт серверда қайта ишланса, мижоз компьютерига фақатгина натижа юборилади. Масалан, агарда серверда скрипт

бажарилаётган бўлса, юқорида келтирилганга ўхшаб мижоз HTML-саҳифа кўринишдаги натижани олади:

```
<html>
   <head>
   <title> Мисол </title>
   </head>
   <body>
   <p> Салом, бу мен – PHP скрипт! </p>
   </body>
</html>
```

Бу ҳолатда мижоз қандай код бажарилаётганини билмайди. Ўз серверингизни HTML-файлларни *PHP* процессори қайта ишлайдиган қилиб созлаб олишингиз ҳам мумкин. Яъни клиентлар оддий HTML-файлни қабул қилдими ёки скрипт натижасини кўрдими буни била олмайди. Агарда скрипт клиент компьютерида қайта ишланса (масалан, JavaScript тилидаги дастур), у ҳолда клиент скрипт кодидан иборат HTML-саҳифани кўради.

Биз юқорида айтиб ўтгандикки, *PHP-скриптлар HTML*-код ичида ёзилади. Қандай қилиб деган савол туғилади. Бунинг бир нечта усуллари мавжуд. Булардан бири биринчи мисолда келтирилганидек, *<?php* теги билан бошланиб *?>* теги билан тугаган синтаксис. Бундай кўринишдаги махсус теглар HTML ва *PHP* режимидагина ишлатилади. Бу синтаксис *PHP* тилини *XML* ҳужжатлари билан биргаликда ишлайдиган дастурларида жуда маъқул кўрилади (масалан, XHTML тилида ёзилган дастурларда). Бироқ базан қуйидаги альтернатив вариантдан фойдаланса ҳам бўлади(echo "Some text" буйруғи «Some text» матнини экранга чиқаради.):

```
1.  <? echo "Бу PHP тилида
2.          оддий қайта ишлашнинг
3.          инструкцияси"; ?>
4.
5.  <script language="php">
6.   echo "Бир нечта редакторлар
7.         (FrontPage) қуйидагича
8.         қабул қилишади";
9.  </script>
10.
11.<% echo " ASP технологиясидаги тегдан
12.         ҳам фойдаланса бўлади"; %>
13.
```

Бу келтирилган усуллардан биринчиси ҳар доим ҳам бажарилавермайди. Ундан фойдаланиш учун қисқа тегларни ишлатиш керак, ёки *PHP*3 учун *short_tags*() функцияни ишлатиш керак, ёки *PHP* тилининг конфигурацион файлига short_open_tag буйруқни ўрнатиш керак, ёки *PHP* дастурлаш тилида *enable-short-tags* параметр билан компиляция қилиш керак. Агарда php.ini-dist буйруққа юқоридагилар автоматик қўшилган бўлса, у

ҳолда қисқа теглардан фойдаланиш тавсия этилмайди. Иккинчи усул худди ўрнига қўйишга ўхшайди, масалан, JavaScript кодлари ва унинг учун мос html теглар. Шунинг учун ундан ҳар доим фойдаланиш мумкин, лекин бу ноқулайлиги учун камдан-кам ишлатилади. Учинчи усулдан фақат ASP технологиясидаги теглар *asp_tags* конфигурациясида ишлатилгандагина фойдаланилади.

PHP дастурлаш тили файлни қайта ишлаётганда у оддий матнни *PHP* код интерпретация қилиши керак бўлган махсус тегларни учратмагунча қайтариб беради. Интерпретатор ҳақида гапирганда у топилган барча кодни ёпиладиган теггача бажаради, сўнг яна оддий матн қайтарилади. Бу механизм *PHP*-кодни HTML саҳифага айлантиради, яъни барча *PHP* теглардан ташқари барча матнларни ўзгаришсиз сақлайди ва ичкаридагиларни эса интерпретациялайди. Яна шуни айтиш керакки, php-файл *CGI*-скриптга ўхшамайди. php-файл бажарилиши шарт эмас, ёки яна қандайдир белгиланади.

php-файлни серверда қайта ишлаш учун жўнатишда сервер томонидан браузер сатрида бу файлни йўлини кўрсатиш шарт. *PHP* скриптлар www орқали киришга рухсат этилган жойда жойлашиши шарт. Агарда php-файл локал компьютерда мавжуд бўлса, у ҳолда уни буйруқлар сатри интерпретатори ёрдамида қайта ишлаш мумкин.

Хулоса.

Шундай қилиб, биз *PHP* дастурлаш тили ҳақида маълумотга эга бўлдик, у қандай дунёга келган ва тарқалган, уни қандай ва қаерда фойдаланилишини ўргандик, дастурий воситани ўрнатдик ҳамда уни ишлаши учун барча созлашларни бажардик ва *php*-дастур нималардан ташкил топишини англадик. Кейинги бўлимларда биз *PHP* дастурлаш тилининг асосий синтаксисларини кўриб чиқамиз ҳамда бир қанча амалий масалаларни ҳал этамиз.

Асосий синтаксислар

Инструкцияни бир нечта қисмга бўлиб кўриб чиқамиз, яъни комментарийлар яратиш, ўзгарувчилар, ўзгармаслар ва маълумот типлари, операторларга.

Биз энди *PHP* дастурлаш тилининг асосий синтаксис элементларини ўрганишга ўтамиз. Мисол сифатида электрон мактуб тайёрлаш масаласини кўриб ўтайлик. Унинг маъноси қуйидагидан иборат.

Фараз қиламизки, сизда қандайдир эълон ва эълонни жўнатишингиз керак бўлган бир нечта одамлар мавжуд бўлсин. Бунинг учун сиз эълонни ичида ўзгарадиган (қабул қилувчи билан боғлиқ бўлмаган) бир нечта параметрлари мундарижаси билан тайёрлайсиз.

Биринчи навбатда PHP дастурлаш тили синтаксисига нисбатан нималарни билиш керак. Бу HTML-код ичига ўрнатилган ва PHP дастурлаш тилидаги коддир, уни интерпретатор фарқлай билади. Аввалги бўлимларда

булар ҳақида айтиб ўтгандик. Ҳаммасини қайтариб ўтмаймиз, фақат биз кўп ҳолларда мисолларда <?php ?> вариант ўрнига қисқартирилган <? ?> теглардан фойдаланишни айтиб ўтамиз.

Инструкцияларни ажратилиши.

PHP дастурлаш тилидаги дастур(ихтиёрий дастурлаш тилидаги) – бу буйруқлар (инструкциялар) тўпламидир. Дастурни қайта ишлаш учун бир буйруқни бошқа буйруқдан фарқини билиш керак. Бунинг учун махсус символлар – ажратгичлардан фойдаланилади. PHP дастурлаш тилида инструкцияларни худди Си ёки Perl дастурлаш тиллари каби ажратилади, яъни ҳар бир ифода нуқтали вергул (“;”) билан тугайди.

«?>» ёпиладиган тег ҳам инструкцияни тугашини англатади, шунинг учун ундан олдин нуқтали вергул қўйилмайди. Масалан, қуйидаги икки фрагментлар эквивалентдир:

```
<?php
echo "Hello, world!"; // буйруқлар охирида нуқтали вергул қўйиш шарт
?>
<?php
echo "Hello, world!" ?>
<!-- "?>" борлиги учун
   нуқтали вергул ташлаб кетилди -->
```

Комментарийлар.

Кўп ҳолларда дастур тузганда кодни тушунарли бўлиши учун унга қандайдир изоҳ-***комментарийлар*** қўйиш керак бўлиб қолади. Бу ҳолат катта ҳажмдаги дастурлар яратганда ҳамда агарда битта дастур устида бир нечта дастурчи ишлаётганда жуда муҳим. Комментарийлар дастурнинг коди тушунарли бўлиши учун ёзилади. Бундан ташқари масалани қисмларга ажратиб ҳал қилинганда ишнинг камчилиги бор жойида кейинчалик эсдан чиқмаслиги учун комментария ёзиб қўйилади. Барча дастурлаш тилларида дастур ичига комментария қўшиш имконияти мавжуд. *PHP* дастурлаш тили бир қанча кўринишдаги комментарийларни қўллаб қувватлайди: Си, C++ дастурлаш тиллари стилидаги ҳамда Unix қобиғидаги комментарийлар. // ва # белгилар бир сатрли комментарийларни англатса, /* ва */ белгилар эса мос равишда кўп сатрли комментарийларнинг бошланиш ва тугашини англатади.

Мисол:PHP дастурлаш тилида комментарийнинг қўлланилиши

```
<?php
echo "Мени исмим Алишер";
  // Бу бир сатрли комментарий
  // C++ дастурлаш тили стилидаги
echo "Мени фамилиям Болиев";
```

```
/* Бу кўп сатрли комментарий. Бу ерга бир қанча сатр ёзиш мумкин. Дастур бажарилиш жараёнида бу ердаги барча ёзувлар (комментарийланган), ўқилмайди. */
echo "Мен PHP дастурлаш тилини INTUIT.ru дан ўрганяпман";
  # Бу комментарий
  # Unix қобиғидаги комментарий.
?>
```

ўзгарувчилар, ўзгармаслар ва операторлар

Ҳар бир дастурлаш тилида муҳим элементлардан бири бу *ўзгарувчилар*, *ўзгармаслар* ва улар қўлланиладиган *операторлар*дир. PHP дастурлаш тили бу элементларни қандай белгилаши ва қайта ишлашини кўриб чиқамиз.

Ўзгарувчилар

PHP дастурлаш тилида *ўзгарувчилар* олдига доллар белгиси (“$”) қўйиб эълон қилинади, масалан, $my_var.

Ўзгарувчилар номлари регистрларни фарқлайди, яъни $my_var ҳамда бош ҳарфли $My_var ўзгарувчилари турли хил ўзгарувчилардир.

PHP дастурлаш тилида ўзгарувчилар номи қолган дастурлаш тиллари қоидалари каби эълон қилинади: ўзгарувчи номи лотин алфавити билан бошланиши ва ундан кейин ҳарфлар ёки тагига чизилган белги ёки рақамлар бўлиши мумкин.

PHP4 дастурлаш тилида булардан ташқари ўзгарувчига қиймат ўзлаштиришнинг яна бир усули мавжуд: ссылка бўйича *ўзлаштириш*. Ссылка бўйича ўзгарувчига қиймат ўзлаштириш учун уни номи бўлиши шарт, яъни у қандайдир ўзгарувчини тақдим этиши керак. Бир ўзгарувчи қийматини бошқа ўзгарувчига Ссылка бўйича *ўзлаштириш* учун биринчи ўзгарувчи олдига амперсанд & белгиси қўйиш шарт.Бунга юқоридаги мисолни кўриб чиқамиз, фақат first ўзгарувчи second ўзгарувчига ссылка бўйича ўзлаштирилади:

Мисол. Ссылкалар бўйича ўзлаштириш.

```
<?php
$first = ' Text '; // $first ўзгарувчига
                   // ' Text ' қиймат ўзлаштирилди
$second = &$first;
/* $second.орқали $first ўзгарувчига ссылка қиламиз
   Энди бу ўзгарувчилар қийматлари
    ҳар доим тенгдир */
// $first ўзгарувчи қийматини
// ' New text ' қийматга ўзгартирамиз
$first = ' New text ';
echo "first номли ўзгарувчи қиймати $first га тенг <br>";
// $second ўзгарувчи қийматини экранга чиқарамиз
echo "second номли ўзгарувчи қиймати " . "$second га тенг";
```

?>

Бу скриптни натижаси эса қуйидагича бўлади:

- first номли ўзгарувчи қиймати New text га тенг.
- second номли ўзгарувчи қиймати New text га тенг.

Яъни $first ўзгарувчи қиймати ўрнига $second ўзгарувчи қиймати ўзлаштирилди.

Ўзгармаслар

Скрипт бажарилиш жараёнида ўзгармайдиган қийматли катталикларни сақлаш учун ***ўзгармаслар***дан фойдаланилади. Бундай катталиклар математик ўзгармаслар, пароллар, файлларнинг йўллари ва бошқалар бўлиши мумкин. Ўзгармасларнинг ўзгарувчилардан асосий фарқи шуки, уларни фақат бир мартагина ўзлаштирилади ва уни қийматини эълон қилингандан кейин бекор қилиб бўлмайди. Бундан ташқари ўзгармаслар олдида доллар белгиси қйилмайди ҳамда уни оддий қиймат ўзлаштириш каби қараш мумкин эмас. Ўзгармаслар қандай аниқланади? Бунинг учун махсус define() функцияси мавжуд, унинг синтаксиси қуйидагичадир:

define("Ўзгармас номи", "Ўзгармас қиймати", [регистрга_сезгирлиги_кичик])

Ўзгармаслар номи регистрга сегирлиги катта. Ҳар бир ўзгармасларда уни ўзгартириш мумкин, яъни *регистрга_сезгирлиги_кичик* аргументни қиймати сифатида True ыиймати кўрсатилади. Ўзгармаслар номи ҳар доим катта регистр билан ёзишга келишиб олинган.

Ўзгармасни қийматини билиш учун уни номини кўрсатиш керак. Ўзгарувчидан фарқи ўзгармас номи олдига $ белги қўйилмайди. Бундан ташқари ўзгармасни қийматини билиш учун константа номи билан параметр сифатида constant() функциясидан фойдаланиш мумкин.

Мисол. PHP дастурлаш тилида ўзгармаслар.

```
<?php
// ўзгармасни аниқлаймиз PASSWORD
define("PASSWORD","qwerty");
// регистрланмаган PI ўзгармасни қийматини аниқлаймиз 3.14
define("PI","3.14", True);
// PASSWORD ўзгармас қийматини оламиз,  яъни qwerty
echo (PASSWORD);
// бу ҳам qwerty ни чиқаради
echo constant("PASSWORD");
echo (password);
/* password ни чиқаради ва биз регистрланган ўзгармас PASSWORD ни кутгандик.*/
echo pi;
// 3.14 ни чиқаради, чунки ўзгармас PI регистрланмаган ва аниқланган.
?>
```

Дастурчи томонидан ўзгарувчилардан ташқари юқорида айтиб ўтганимиздек *PHP* дастурлаш тилида мавжуд ўзгармаслар ҳам интерпретатор томонидан аниқланади. Масалан, __FILE__ ўзгармас дастур бажарилиш жараёнида файл номини (ва файл йўлини), __FUNCTION__ функция номидан ташкил топади, __CLASS__ - синф номи, PHP_VERSION – *PHP* дастурлаш тили интерпретатори версиясини ўзида сақлайди. Бундай ўзгармасларнинг барча рўйхатини *PHP* дастурлаш тили учун мўлжалланган қўлланмалардан топиш мумкин.

Амаллар.

Ўзгарувчилар, ўзгармаслар ва ифодалар устида турли ҳисоблашларни бажарадиган бу ***амаллар***дир. Биз ҳали бу ифодалар ҳақида тўхтаб ўтганимиз йўқ. Ифодалар қийматини ушбу амаллар ёрдамида аниқланади. Ўзгарувчилар ва ўзгармаслар – бу ифодаларнинг асосий ва жуда содда шаклидир. Шундай ифодаларни кўпайтириши мумкин бўлган амаллар тўплами мавжуд. Уларни қуйида тўлиқроқ муҳокама қиламиз:

9.1-жадвал. Арифметик амаллар.

Белгиланиши	**Номланиши**	**Мисол**
+	Қўшиш	$a + $b
-	Айириш	$a - $b
*	Кўпайтириш	$a * $b
/	Бўлиш	$a / $b
%	Бўлишдаги қолдиқ	$a % $b

9.2-жадвал. Сатрли амаллар.

Белгиланиши	**Номланиши**	**Мисол**
.	*Конкатенация* (*сатрларни қўшиш*)	$c = $a . $b (бу $c сатр $a ва $b сатрлардан иборат)

9.3-жадвал. Ўзлаштириш амаллари.

Белгиланиши	**Номланиши**	**Изох**	**Мисол**
=	Ўзлаштириш	*Операторлан* ўнг томонда турган ўзгарувчилар устида бажарилган амаллардан ҳосил бўлган натижа қиймати ўзлаштирилади.	$a = ($b = 4) +5; ($a 9 га тенг, $b 4 га тенг)
+=		Қисқартириш. Ўзгарувчига сон қўшилади ва кейин натижа ўзлаштирилади.	$a += 5; ($a = $a + 5 ифодага эквивалент;)
.=		Ўзлаштириш ва конкатенация амаллари комбинациясини	$b = "Ҳаммага "; $b .= "салом";

		қисқартирилган шакли(даставвал сатрлар қўшилади, сўнгра ҳосил бўлган сатр ўзгарувчига ўзлашади).	($b = $b . "салом" ифодага эквивалент;) Натижаси: $b="Ҳаммага салом"

9.4-жадвал. Матиқий амаллар.			
Белгиланиши	**Номланиши**	**Изох**	**Мисол**
and	ВА	$a ва $b рост (True)	$a and $b $a && $b $a or $b $a || $b $a xor $b ! $a
&&	ВА		
or	ЁКИ	$a ёки $b ўзгарувчилардан ҳеч бўлмаганда биттаси рост бўлса (иккаласи ҳам рост бўлишиш мумкин).	
||	ЁКИ		
xor	Инверсия ЁКИ	Ўзгарувчилардан биттаси рост бўлса. Агарда иккаласи ҳам рост бўлса инверсияланади.	
!	Инверсия (NOT)	Агарда $a=True, у ҳолда !$a=False ва акс ҳолда тескариси бўлади.	

9.5-жадвал. Таққослаш амаллари.			
Белгиланиши	**Номланиши**	**Изох**	**Мисол**
==	Тенглик	Ўзгарувчилар қийматлари тенг	$a == $b
===	Эквивалентлик	Ўзгарувчилар қийматлари ва типлари тенг	$a === $b
!=	Тенгсизлик	Ўзгарувчилар қийматлари тенг эмас	$a != $b
<>	Тенгсизлик		$a <> $b
!==	Ноэквивалентлик	Ўзгарувчилар эквивалент эмас	$a !== $b
<	Кичик		$a < $b
>	Катта		$a > $b
<=	Кичик ёки тенг		$a <= $b
>=	Катта ёки тенг		$a >= $b

9.6-жадвал. Инкремент ва декремент амаллари.			
Белгиланиши	**Номланиши**	**Изох**	**Мисол**
++$a	Пре-	$a қиймати бирга оширилади ва	<?

	инкремент	$a қиймати қайтарилади	$a=4; echo "4 бўлиши шарт:" .$a++; echo "6 бўлиши шарт:" .++$a; ?>
$a++	Пост-*инкремент*	$a қиймати қайтарилади ва сўнгра $a қиймати бирга оширилади	
--$a	Пре-*декремент*	$a қиймати бирга камайтирилади ва $a қиймати қайтарилади	
$a--	Пост-*декремент*	$a қиймати қайтарилади ва сўнгра $a қиймати бирга камайтирилади	

маълумотлар типлари

PHP дастурлаш тили саккизта содда *маълумот типлари*ни қўллаб қувватлайди:

Тўрттаси скаляр *типлар*:

- *boolean (мантиқий)*;
- *integer (бутун)*;
- *float (нуқтаси силжийдиган)*;
- *string (сатрли)*.

Иккитаси аралиш *типлар*:

- *array (массив)*;
- *object (объект)*.

Иккитаси махсус *типлар*:

- *resource (ресурс)*;
- *NULL*.

PHP дастурлаш тилида ўзгарувчилар типлари ошкора эълон қилинмайди. Кўпинча ўзгарувчи қўлланилган контексдан, яъни ўзгарувчига ўзлаштирилган қиймат итпидан мустақил равишдаги дастур бажарилиш жараёнидан интерпретатор ўзи бу ишни бажаради. Қуйида юқорида санаб ўтилган *маълумотлар типлари*ни бирма-бир кўриб чиқамиз.

Boolean типи(Буль ёки мантиқий тип).

Бу содда тип қийматни рост эканлигини ифодалайди, яъни ўзгарувчи фақат иккита қиймат қабул қилади – рост TRUE ёки ёлғон FALSE.

Мантиқий типларни аниқлаш учун TRUE ёки FALSE калит сўзларидан фойдаланамиз. Бу иккала типлар регистрланмаган.

Мисол. Мантиқий тип.

```
<?php
$test = True;
?>
```

Мантиқий типлар турли *бошқариладиган конструкциялар*да (цикллар, шартлар ва шунга ўхшаш, булар ҳақида кейинроқ айтиб ўтамиз) қўлланилади. Бир қанча амаллар (масалан, тенглик амали) ҳам мантиқий тип қабул қилиши мумкин, яъни фақат икки қиймат рост ёки ёлғон қийматни қабул қилади. Улар *бошқариладиган конструкциялар*да шартларни

текшириш учун қўлланилади. Масалан, шартли конструкторда амаллар ёки ўзгарувчилар қиймати ҳақиқийлигини текширади ва натижадан қатъий назар шу ёки бошқа амалларни бажарилишини текширади. Бу ерда шарт рост ёки ёлғон бўлиши мумкин, чунки *мантиқий тип амаллари* ва *ўзгарувчилар* кўрсатилган.

Мисол. Мантиқий типларнинг қўлланилиши.

```
<?php
// '==' амал тенгликка текширади мантиқий қийматни қайтаради
if ($know == False) { // агар $know қиймат false бўлса
echo "PHP дастурлаш тилини ўрган!";
}
if (!$know) { // худди юқоридагидек $know қиймати false бўлади
echo " PHP дастурлаш тилини ўрган!";
}
/*  == амал $action ўзгарувчи қиймати билан "PHP дастурлаш тилини ўрганиш!" сатрни устма-уст тушишини текширади. Агар устма-уст тушса true қийматни қайтаради, бошқа ҳолда false ни қайтаради. Агар true ни қайтарса фигурали қавс ичидаги амаллар бажарилади. */
if ($action == " PHP дастурлаш тилини ўрганиш ")
{ echo "Ўрганишни бошладим";}
?>
```

Integer (бутун) типи.

Бу тип бутун сонлар тўпламидан Z = {..., -2, -1, 0, 1, 2, ...} бирини қайтаради. Бутун сонлар хоҳишга қараб олдига «-» ёки «+» белгиларни қўйиб саноқ системасини ўнлик, ўн олтилик ёки саккизлик тизимларида кўрсатилган бўлиши мумкин.

Агар сиз саккилик саноқ системасидан фойдаланаётган бўлсангиз, олдиндан 0 (ноль) рақамини кўрсатишингиз керак. Ўн олтилик саноқ системасида эса рақамлар олдига 0x белгини қўйиш шарт.

```
<?php
# ўнлик рақам
$a = 1234;
# манфий сон
$a = -123;
# саккизлик сон (ўнлик системасидаги
# 83 сонга эквивалент)
$a = 0123;
# ўн олтилик сон (ўнлик системасидаги
# 26 сонга эквивалент)
$a = 0x1A;
?>
```

*Бутун сон*ни ўлчами платформага боғлиқ, лекин қоидага кўра максимал қиймати икки миллиард (бу ишорали 32 битли қиймат) атрофида бўлади. Ишорасиз *бутун сон*ни *PHP* дастурлаш тили қўллаб қувватламайди.

Агар сиз *бутун сон* чегарасидан ташқари бирор қиймат берсангиз интерпретатор бу сонни *қўзғалувчан вергулли сон*га ўзгартиради. Худди шундай *бутун сон* чегарасидан ташқари чиқиб кетадиган бирор амал бажарсангиз ҳам бу сонни *қўзғалувчан вергулли сон*га ўзгартирилади.

PHP дастурлаш тилида бутун сонларни бўлиш амали мавжуд эмас. 1/2 ифода қиймати *қўзғалувчан вергулли сон* 0.5 га тенг. Сиз натижангизни бутун типга стандарт қоида асосида ёки round() функциясидан фойдаланган тақдирда ўзгартиришингиз мумкин. Ўзгарувчини аниқ бир типга ўзгартириш учун унинг олдига қавс ичида керакли типни ёзиш керак бўлади. Масалан, $a=0.5 ўзгарувчини бутун типга ўзгартириш учун (integer)(0.5) ёки (integer) $a кўринишда ёки қисқартирилган (int)(0.5) кўринишда ёзиш керак бўлади. Бундай ошкора янги типга ўтиш имконияти барча *маълумотлар типлари* учун ўринли бўлади (албатта, ҳар доим ҳам қийматни бир типдан бошқасига олиб ўтиш шарт эмас). Биз келтирилган барча типларни чуқур ўрганишимиз шарт эмас, чунки *PHP* дастурлаш тили контексдан мустақил равишда ўзи бу ишларни бажаради.

Float (қўзғалувчан вергулли сон) типи.

*Қўзғалувчан вергулли сон*лар (улар икки карра аниқлик ёки ҳақиқий сонлардир) қуйидаги синтаксислар ёрдамида аниқланиши мумкин:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

*Қўзғалувчан вергулли сон*ни ўлчами ҳам платформага боғлиқ, лекин қоидага кўра максимал қиймати ~1.8e308 аниқлик билан 14 хонали рақам атрофида бўлади.

Resource (ресурслар) типи.

Ресурс – бу ташқи ресурсга (масалан, маълумотлар базаси билан боғланиш) ссылка орқали боғланган махсус ўзгарувчидир. Ресурслар махсус функциялар (масалан, mysql_connect(), pdf_new() ва шунга ўхшашлар) ёрдамида яратилади ва фойдаланилади.

Null типи.

Махсус *NULL* қиймати *ўзгарувчи*ни қийматга эга эмаслиги ҳақида огоҳлантиради.

Ўзгарувчи NULL қиймат қабул қилади, агарда:

- унга *ўзгармас NULL* ($var = NULL) ўзлаштирилган бўлса;
- унга ҳеч қандай қиймат берилмаган бўлса;
- у *unset*()функция ёрдамида тозаланган бўлса.

*NULL тип*ли фақат битта қиймати мавжуд – регистрга сезгирлиги кичик NULL калит сўзидир.

Масаланинг ечилиши.

Энди бўлимнинг бошида қўйилган масалага қайтсак. У турли сабаблар бўйича ҳар хил одамларга тузилган мактубни жўнатишдан иборат эди. Бу масалани ҳал этиш учун ўрганилган воситалардан – *ўзгарувчилар*, *амаллар*, *ўзгармаслар*, *сатрлар* ва *массивлар*дан фойдаланишга ҳаракат қиламиз. Кўрсатилган мактуб қабул қилувчига боғлиқ равишда мурожаат ва ҳолати ўзгаради, шунинг учун табиий равишда бу катталикни *ўзгарувчи* деб белгилаймиз. Бундан ташқари ҳодисалар ва одамлар кўп, шунинг учун *массив ўзгарувчи типи*дан фойдаланиш қулай. Мактуб матни ҳар доим ўзгармас, шунинг учун уни *ўзгармас* деб бериш мақсадга мувофиқдир. Жуда узун ва кўпол сатрларни ёзмаслик учун сатрлар *конкатенация*(қўшиш) амалидан фойдаланамиз. Шундай қилиб, қуйидагига эга бўламиз:

```
<?
// бизнинг ёзувимиз
// ўзгармас бўлсин.
define("SIGN","Ҳурмат билан, Азамат");
// одамлар ва ҳодисалар массивини берамиз
$names = array("Иван Иванович",
               "Петр Петрович",
               "Семен Семенович");
$events = array(
   "f" => "очиқ эшиклар куни",
   "o" => "кўргазманинг очилиши",
   "p" => "битирувчилар бали");

// таклифнома матнини тузамиз.
$str = "Ҳурматли, $names[0]";
$str .= "<br> Сизни таклиф этамиз ".
       $events["f"];
$str .= "<br>" . SIGN;
echo $str; // матнни экранга чиқарамиз.
?>
```

Хулоса.

Шундай қилиб, бу бўлимда биз *PHP* дастурлаш тилининг асосий синтаксиси билан танишиб чиқдик, турли типдаги ўзгарувчилар, ўгармаслар ва амаллар билан ишлашни, *PHP* дастурлаш тилидаги мавжуд типларини ўргандик. *Массивлар* ва *сатрлар* маълумот типлари ҳақида гап кетганда уларни чуқур ва қисмларга ажратиб ўргандик. Бу конструкциялар фойдаланишга қулай ва соддадир. Булар ҳақида кенг маълумотлар кейинги бўлимларда келтирилган. Масаланинг ечилиши бор билимларга асосланган ҳолда содда ечилган, шунинг учун ечим амалиётда қўллашга жуда яқин келмайди. Кейинги бўлимларда бу камчиликларни тўғрилаймиз ва электрон мактубни умумий шаблонини яратамиз.

Альтернатив синтаксислар

PHP дастурлаш тили ўзининг бир нечта *if, while, for, foreach* ҳамда *switch* бошқариладиган структуралари учун альтернатив синтаксисни тақдим этади. Ҳар бир ҳолатда очиладиган қавс икки нуқтага (:), ёпиладигани эса мос равишда endif;, endwhile; ва ҳоказоларга ўзгартирилади.

Масалан, *if* шарт оператори синтаксисини қуйидагича тфодалаш мумкин:

if (ифода) : *бажариладиган_блок* endif;

Маъноси ўзгармасдан қолади: агар *if* шарт оператори думалоқ қавси ичидаги шарт рост бўлса, икки нуқтадан «:» то endif; буйруғигача барча код бажарилади. Бундай синтаксисдан фойдаланиш html-код ичида қурилган php-код учун қулайдир.

Мисол. Альтернатив синтаксисдан фойдаланиш.

```
<?php
$names = array("Карим","Салим","Содиқ");
if ($names[0]=="Карим"):
?>
Салом, Карим!
<?php endif ?>
```

Агарда *else* ҳамда *elseif* конструкцияларидан фойдаланилса, у ҳолда ҳам альтернатив синтаксисдан фойдаланса бўлади:

```
<?php
if ($a == 5):
   print "a ўзгарувчи 5 га тенг";
   print "...";
elseif ($a == 6):
   print "a ўзгарувчи 6 га тенг ";
   print "!!!";
else:
   print "a ўзгарувчи на 5 га ва на 6 га тенг ";
endif;
?>
```

Такрорлаш учун саволлар

1. PHP кодни тузилишини тушунтириб беринг.
2. PHP да қандай маълумот типларидан фойдаланилади?
3. PHP да ифодалар қандай эълон қилинади?
4. Альтернатив синтаксислар деганда нимани тушунасиз?

17-маъруза. PHP бошқариш элементлари

Режа:
1. Шарт операторлари.
2. Бошқарув ўтказувчи операторлар
3. Цикллар.
4. switch оператори.

Калит сўзлар: шартли ва бошқарув ўтказувчи операторлар, цикл операторлари, switch оператори

PHP дастурлаш тили ўзининг бир нечта *if*, *while*, *for*, *foreach* ҳамда *switch* бошқариладиган структуралари учун альтернатив синтаксисни тақдим этади. Ҳар бир ҳолатда очиладиган қавс икки нуқтага (:), ёпиладигани эса мос равишда endif;, endwhile; ва ҳоказоларга ўзгартирилади.

Масалан, *if* шарт оператори синтаксисини қуйидагича тфодалаш мумкин:

if (ифода) : *бажариладиган_блок* endif;

Маъноси ўзгармасдан қолади: агар *if* шарт оператори думалоқ қавси ичидаги шарт рост бўлса, икки нуқтадан «:» то endif; буйруғигача барча код бажарилади. Бундай синтаксисдан фойдаланиш html-код ичида қурилган php-код учун қулайдир.

Мисол. Альтернатив синтаксисдан фойдаланиш.

```
<?php
$names = array("Карим","Салим","Содиқ");
if ($names[0]=="Карим"):
?>
Салом, Карим!
<?php endif ?>
```

Агарда *else* ҳамда *elseif* конструкцияларидан фойдаланилса, у ҳолда ҳам альтернатив синтаксисдан фойдаланса бўлади:

```
<?php
if ($a == 5):
   print "a ўзгарувчи 5 га тенг";
   print "...";
elseif ($a == 6):
   print "a ўзгарувчи 6 га тенг ";
   print "!!!";
else:
   print "a ўзгарувчи на 5 га ва на 6 га тенг ";
endif;
?>
```

Шарт операторлари

if оператори.

Бу оператор **JavaScript** дастурлаш тилидаги муҳим операторлардан биридир. У шартга боғлиқ равишда код фрагментини бажаришга мўлжалланган. *if* операторининг структурасини қуйидагича ифодалаш мумкин:

1- вариант. if оператори синтаксиси:

```
if(шарт)
{
Амаллар кетма кетлиги
}
```

2- вариант. if оператори синтаксиси:

```
if(шарт)
{
Амаллар кетма кетлиги 1
}
else
{
Амаллар кетма кетлиги 2
}
```

Бу ерда **шарт** JavaScript тилидаги мантикий еки ихтиёрий шартдир. Агарда шарт мантикий типдаги узгарувчи булса, қиймати рост (True) бўлса, у ҳолда *Амаллар кетма кетлиги1* бажарилади. Акс ҳолда *Амаллар кетма кетлиги2* бажарилади.

Мисол. if шарт оператори

```
function checkData()
{
if (document.form1.threeChar.value.length==3)
{return true;
}
else
{  alert('3 ракамини киритинг');
return false;
}
}
```

else оператори

Биз юқорида фақат *if* операторининг асосий қисминигина кўрдик. Бу операторнинг бир нечта кенгайган шакли мавжуд. *else* оператори *if* операторида текширилаётган ифода нотўғри бўлган ҳолатдагина кенгайтиради ҳамда бу ҳолатда янги шартда бирор амал бажаради.

else оператори ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (шартли ифода) *бажариладиган_блок1*

else *бажариладиган_блок2*

Бу if...else конструкцияси қуйидагича интерпретация қилиниши мумкин: агар шарт бажарилса (яъни ифода=true), у ҳолда *бажариладиган_блок1* даги амаллар бажарилади, акс ҳолда *бажариладиган_блок2*даги амаллар бажарилади. *else* операторидан фойдаланиш мажбурий эмас.

elseif оператори.

if шарт операторининг яна бир кенгайган шакли – бу *elseif* операторининг қўлланилишидир. *elseif* – бу *else* ҳамда *if* операторларининг комбинациясидир. У худди *else* оператори каби *if* операторида шарт бажарилмаган ҳолда кенгайтиради. Бироқ *else* операторидан фарқи бир-бирига зид амалларни фақат агарда ***elseif*** шарт рост бўлгандагина бажаради. *else* ҳамда *elseif* операторлари ёрдамида кенгайтирилган *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (ифода1) *бажариладиган_блок1*
elseif (ифода2) *бажариладиган_блок2*
else бажариладиган_блокN

elseif операторлари битта *if*-блокида бир неча марта учраши мумкин. *elseif* тасдиғи фақат олдинда турган *if*-шартлари ҳамда *elseif*-шартлари False қийматни, берилган *elseif*-шарти эса True қийматни қайтаргандагина бажарилади.

Мисол. elseif оператори

```
fanction makeMinutes() {
   var minstring="";
   var now = new Date();
   var min = Date.getMinutes();
   if (min<10) {
     minstring+=":0"+min;}
   elseif(min>10){
     minstring+=":"+min;}
   return minstring;
}
```

switch оператори.

Яна бир шартни текшириб турли амалларга боғлиқ равишда иш кўрсатадиган конструкция бу – *switch* операторидир. Бу операторни узбек тилига таржима қилинганда “йўналишни ўзгартиргич” маъносини беради ҳамда бу операторнинг вазифаси ҳам шунга ўхшашдир. Ўзгарувчини қандай қийматни қабул қилишига боғлиқ равишда у йўналишни ўзгартириб турли блоклардаги амалларни бажаради. *switch* оператори *if...elseif...else* ёки *if* оператори мажмуига жуда ўхшаш бўлади. *switch* операторининг структурасини қуйидагича ифодалаш мумкин:

switch (ифода ёки ўзгарувчи){

```
case қиймат1:
   амаллар_блоки1
break;
case қиймат2:
   амаллар_блоки2
break;
...
default:
   амаллар_блоки_автоматик_тарзда
}
```

if операторидан фарқли томони бу ерда ифодалар мантиқий тип қабул қилмай, балки фақат *case* калит сўзидан кейинги қийматларни (*қиймат1*, *қиймат2* ва ҳ.к.) таққослайди холос. Агар ифода қиймати қандайдир вариант билан устма-уст тушса, икки нуқтадан кейинги *break* операторигача бўлган *амаллар_блоки*даги амалларни бажаради. Агарда ифода қиймати берилган вариантлардан ҳеч бирига устма-уст тушмаса, *default* калит сўзидан кейинги автоматик тарзда бажариладиган блок (*амаллар_блоки_автоматик_тарзда*) бажарилади. *switch* операторидаги ифода фақат бир марта ҳисобланади, *elseif* операторида эса ҳар бир текширишда ҳисобланади, шунинг учун агарда ифода етарли даражада мураккаб бўлса, у ҳолда *switch* оператори тезроқ ишлайди.

Мисол 1: *switch* оператори:

```
function getName(){
var names = array("Name1"," Name2"," Name3");
var selected=“”;
switch (names[0]){
case "Name1":  selected="Name1 is selected";
break;
case "Name2":  selected="Name2 is selected";
break;
case "Name3":  selected="Name3 is selected";
break;
default: selected="Default select"+$names[0];
}
return selected;
}
```

Юкоридаги мисолдан куриниб турибдики, getName() функцияси ишга туширилганда, names массивидаги биринчи элемент текширилади. Бу холда массив 1-элементи Name1 булгани учун бизга selected="Name1 is selected"; кийматни кайтаради.

Мисол 2: *switch* оператори:

```
var change = prompt("Харакатни танланг:\n1 – Мошина сотиб олиш\n2 – Мошина сотиш \n3 – Мошина алмаштириш ");
```

```
switch (change) {
  case "1": {
    document.write("Сиз мошина сотиб олишни хохлайсиз ");
    break;
  }
  case "2": {
    document.write("Сиз мошина сотишни хохлайсиз ");
    break;
  }
  case "3": {
    document.write("Сиз мошина алмаштиришни хохлайсиз ");
    break;
  }
  default: {
    document.write("Сиз нотугри команда киритдингиз ");
    break;
  }
}
```

switch операторининг констркуцияси учун худди *if* оператори каби альтернатив синтаксиси мавжуд. Бу ерда *switch* операторидаги очиладиган фигурали қавс икки нуқтага ўзгартирилади, ёпиладигани эса мос равишда *endswitch*; калит сўзига ўзгартирилади.

Такрорлаш учун саволлар:

1. Шарт операторларига мисоллар келтиринг?
2. Бошқарув ўтказувчи операторлар нима?
3. switch оператори?

18-маъруза. PHP да сатр ва массивлар билан ишлаш

Режа:
1. PHP да сатр типи
2. PHP да массивлар типи

***Калит сўзлар**: PHP да сатр типи, массив типи*

String (сатр) типи

Сатр – бу белгилар тўпламидир. *PHP* дастурлаш тилида белги бу бир байт ва 256 та турли белгилар мавжуд. *PHP* дастурлаш тили Unicode типидаги белгиларни қабул қилмайди. *PHP* дастурлаш тилида амалда сатрларга чегирма мавжуд эмас, шунинг учун сатрларни ишлатганда унинг аниқ узунлиги ҳақида ўйлаш шарт эмас.

PHP дастурлаш тилида сатрлар учта турли хил усулларда аниқланади:

- *битталик қўштирноқлар* ёрдамида (‘’);
- *қўштирноқлар* ёрдамида (“”);
- *heredoc-синтаксиси* ёрдамида.

Биттали тирноқлар

Сатрларнинг аниқлашнинг оддий усули – у «'» *биттали қўштирноқлар* ичида ёзилади. Агарда сатр ичида ҳам биттали тирноқ ишлатишга тўғри келиб қолса, биттали тирноқдан олдин «\» белгини қўйиш, яъни уни экранлаш шарт. Агарда «\» белги биттали тирноқдан олдин ёки сатрнинг охирида бўлса, у ҳолда белгини иккилантириш керак, яъни «\\'».

Агарда биттали тирноқ ичидаги сатр ичида ихтиёрий белгидан олдин («\» ва «'» лардан фарқли равишда) тескари слэш «\» белгиси учраса, у ҳолда уни оддий белги деб қараб барча белгиларни ўз ҳолича экранга чиқаради. Шунинг учун тескари слэш «\» белгисини сатр охирида ёпиладиган қўштирноқдан аввал турганини экранлаш шарт.

PHP дастурлаш тилида тескари слэш «\» белгиси билан ифодаланадиган бир қатор белгилар мажмуи мавжуд. Уларни ***кетма-кетликни бошқарувчилар*** деб аталади ҳамда улар махсус вазифаларни бажаради. Улар ҳақида кейинроқ тўхталиб ўтамиз. Ўзгарувчилар ва кетма-кетликни бошқарувчилар битталик қўштирноқлар сатри ичида учрашса, улар ўртасидаги фарқ *кетма-кетликни бошқарувчилар*ни *қайта ишланмайди*.

```
<?php
echo 'Сатрлар мажмуи';
// Экранга чиқаради: ' белгини чиқариш учун ундан олдин \ белги қўйилади.
echo ' Белгини \' чиқариш учун ундан олдин'
    ' \\ белгини қўйиш керак';
// Экранга чиқаради: Сиз шуни ўчирмоқчимисиз C:\*.*?
echo ' Сиз шуни ўчирмоқчимисиз C:\\*.*?';
// Экранга чиқаради: Буни қўйманг: \n
// янги қаторга
```

```
echo ' Буни қўйманг: \n янги қаторга ';
// Экранга чиқаради: ўзгарувчи $expand ҳам
// $either қўйилмайди
echo 'ўзгарувчи $expand ҳам $either' .
    'қўйилмайди';
?>
```

Array (массив) типи.

PHP дастурлаш тилида ***массив*** типи тартибланган карталарга ўхшайди ва қийматини калитга ўзлаштирадиган типдир. Бу тип бир неча йўналишларда оптималлаштирилади, шунинг учун сиз уни хусусий *массив*, рўйхат (вектор), хеш-жадвали (картани амалга ошириш учун ишлатилади), стэк, навбат ва бошқалар сифатида фойдаланишингиз мумкин. Модомики, *PHP* дастурлаш тилида бир массивни қийматини бошқасига ўзлаштириш учун дарахтлардан фойдаланасиз.

*Массивлар*ни *array*() конструкцияси ёрдамида аниқланади ёки элементларига қиймат бериш билан аниқланади.

array() конструкцияси ёрдамида аниқлаш.

```
array ([key] => value,
      [key1] => value1, ... )
```

PHP дастурлаш тилининг ***array***() конструкцияси вергул билан ажратилган жуфт параметрлар *калит* => ***қиймат*** билан ажратилган. => белги мос равишда қиймат ва унинг калити ўртасида алоқа ўрнатади. Калит бутун сон бўлиши мумкин, унинг қиймати эса *PHP* дастурлаш тилидаги ихтиёрий типни қабул қилиши мумкин. Калит рақамини биз кўпинча индекс деб атаймиз. *PHP* дастурлаш тилида индекслаш нолдан бошланади. Массив элементининг қийматини олиш учун массив номи ва квадрат қавс ичида унинг калити кўрсатилиши керак. Агар массив калити стандарт бутун сон бўлса, у ҳолда унинг қийматини бутун сон деб қараса бўлади, акс ҳолда у сатр деб қаралади. Шунинг учун $a["1"] ёзув $a[1] ёзувга тенг кучли, $a["-1"] ёзув эса $a[-1] ёзувга тенг кучли.

Мисол. PHP дастурлаш тилида массивлар.

```
<?php
$books = array ("php" =>
              "PHP users guide",
              12 => true);
echo $books["php"];
//экранга чиқаради: "PHP users guide"
echo $books[12];     //экранга чиқаради: 1
?>
```

Агарда элемент учун калит берилмаган бўлса, у ҳолда калит сифатида калитнинг максимал қийматига бир қўшиб ҳисобланади. Агарда қиймати мавжуд калит кўрсатилган бўлса, у ҳолда шу калит қийматини экранга чиқаради. PHP 4.3.0 дастурлаш тили версиясидан бошлаб калитнинг

максимал қиймати манфий сон деб қаралса, у ҳолда массивнинг кейинги калити ноль (0) бўлади.

Мисол. PHP дастурлаш тилида массивлар.

```
<?php
// $arr ҳамда $arr1 массивлар эквивалентдир.
$arr = array(5 => 43, 32, 56, "b" => 12);
$arr1 = array(5 => 43, 6 => 32,
          7 => 56, "b" => 12);
?>
```

Агарда TRUE ёки FALSE калит сифатида қўлланилса, у ҳолда унинг қиймати мос равишда *integer* типининг бир ва нолига ўзлаштирилади. Агар *NULL* дан фойдаланилса, у ҳолда калит ўрнига бўш сатр ҳосил бўлади. Бу бўш сатрни калит сифатида фойдаланса бўлади, аммо уни қўштирноққа олиш керак бўлади. Бу усул бўш квадрат қавсни ишлатиш каби эмас. Массивлар ёки объектлар калити сифатида фойдаланиш мумкин ҳам эмас.

Квадрат қавс синтаксиси ёрдамида аниқлаш.

Массивга қиймат бериш орқали массив яратиш мумкин. Биз юқорида айтиб ўтганимиздек, массив элементи қийматига эга бўлиш учун квадрат қавс ичига унинг калити кўрсатилиши керак, масалан, $book["php"]. Агарда янги калит ва янги қиймат кўрсатсангиз қуйидагича бўлади: $book["new_key"]="new_value" ҳамда массивга янги элемент қўшилади. Агарда калитни кўрсатмай фақат қийматни ўзлаштирсак, яъни $book[]="new_value", у ҳолда массивга янги элемент қўшилади ва уни калити мавжуд максимал қийматга бир қўшилади. Агарда биз қиймат берган *массив* яратилмаган бўлса, у ҳолда биз қиймат бергандан кейин у *яратилади*.

```
<?
$books["key"]= value; // key калити билан value қиймат $books массивига қўшилади
$books[] = value1; /* 13-калит билан value1 қиймати массивга қўшилади, чунки
бизда калитнинг максимал қиймати 12 эди. */
?>
```

Массивнинг аниқ бир элементини ўзгартириш учун унинг шу калити билан янги қийматга ўзлаштириш керак. Массив элементи калитини ўзгартириш мумкин эмас, фақат ўчириш (калит ва элементи жуфтлигини) ва янги қўшиш мумкин холос. Массив ***элементини ўчириш*** учун ***unset***() функциясидан фойдаланиш керак.

```
<?php
$books = array ("php" =>"PHP users guide",12 => true);
$books[] = "Book about Perl"; /* 13-калит(индекс) билан янги элемент қўшилди,
 бу қуйидагига эквивалент $books[13] = "Book about Perl";
$books["lisp"] = 123456; /* Бу массивга янги "lisp" калитли 123456 қиймали янги
```

```
 элемент қўшиш*/
unset($books[12]); // Бу 12-калитли элементни массивдан ўчириш
unset ($books); // массивни бутунлай ўчириш
?>
```

Бўш квадрат қавсдан фойдаланганда калитнинг максимал қиймати массивда мавжуд охирги қайта индексланган калитлар орасидан қидирилади. Массивни ***array_values***() функцияси ёрдамида ***қайта индекслаш*** мумкин.

Мисол. Массивни қайта индекслаймиз.

```
<?php
$arr =
 array ("a","b","c");
/* "a", "b" ва "c" қийматли массивни яратамиз. Бу ерда калит кўрсатилмаган бироқ мос равишда улар 0,1,2 бўлади. */
print_r($arr); // массивни экранга чиқарамиз (калити ва қийматини)
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
// массивдан ҳамма элементини ўчирамиз
print_r($arr);  // массивни экранга чиқарамиз (калити ва қийматини)
$arr[] = "aa";   // массивга янги элемент қўшамиз. Уни индекси(калити) 3 бўлади, 0 эмас.
print_r($arr);
$arr =
array_values($arr); // массивни қайта индекслаймиз.
$arr[] = "bb"; // бу элементни калити 1 бўлади.
print_r($arr);
?>
```

Бу скриптнинг натижаси қуйидагича бўлади:

```
Array ( [0] => a [1] => b [2] => c )
Array ( )
Array ( [3] => aa )
Array ( [0] => aa [1] => bb )
```

Такрорлаш учун саволлар

1. PHP да қандай типларни биласиз?
2. PHP да сатр типини тушунтириб беринг?
3. PHP да массивлар тушунтириб беринг?

19-маъруза. PHP да функциялар. Ўзгарувчига боғлиқ функциялар билан ишлаш.

Режа:
1. Функциялар
2. Функцияларнинг аргументлари
3. Ўзгарувчан узунлик аргументлари рўйхатлари
4. Функциялар ичида ўзгарувчилардан фойдаланиш
5. Функциянинг ўзгарувчилари
6. Ички жойлашган (ичма-ич) функциялар
7. Синфлар ва объектлар
8. Ўзгарувчиларни инициаллаштириш

Калит сўзлар: функциялар, класс ва объектлар, хатоликлар.

Функциялар

Функциялар нима учун керак? Бу саволга жавоб бериш учун, функция ўзи нима эканлигини тушуниб олиш лозим бўлади. Дастурлашда, худди математикадаги каби, унга боғлиқ кўпгина аргументларнинг унинг кўпгина маъноларида акс этишидир. Демак, функция аргументнинг ҳар бир маънолари жамланмаси учун унинг бажарган иши натижаси сифатида қандайдир маъно қайтаради. Функциялар нима учун керак, буни мисоллар билан ойдинлаштиришга ҳаракат қиламиз. Дастурлашдаги функцияга классик мисол – бу соннинг факториал аҳамиятини ҳисоблаб берувчи функция. Демак, биз унга сон берамиз, у эса бизга унинг факториалини қайтаради. Бунда биз факториалини олишни хоҳлаган ҳар бир сон учун айнан бир хил кодни қайтаравермаймиз – бу сонга тенг бўлган аргументли функцияни чақиришнинг ўзи кифоя қилади.

Шу йўл билан биз бирон-бир маълумотга боғлиқлик зарурияти туғилган амални бажарганимизда, бу ҳолда ҳам биз айнан шундай амалларни бажаришимиз оширишимиз лозим бўлади, фақат бошқа бошланғич маълумотлардан фойдаланамиз, функциялар механизмидан фойдаланиш– функция танаси кўринишидаги амаллар блокини тахт қилиш, ўзгарувчан маълумотларни эса – унинг параметрлари сифатида фойдаланиш қулайроқ бўлади.

Функция топшириғи (эълони) умумий тарзда қандай бўлишини кўрамиз. Функция қуйидаги синтаксис ёрдамида аниқланади:

```
function  Функция_номи (1-параметр, 2-параметр,  ... N-параметр) {
  Амаллар блоки
  return "функцияга айланувчи маъно";
}
```

Агар php-дастурда тўғридан-тўғри ёзилса, ҳеч нарсани ишлаб бўлмайди. Биринчидан, функция номи функция параметрлари номлари (1-параметр, 2-параметр ва б.) PHP да номланиш қоидаларига мувофиқ келиши керак (унда яхшиси кириллча символларни ҳам ишлатмаган маъқул).

Функция номлари регистрга нисбатан сезувчан бўлади. Иккинчидан, функция параметрлари – тилнинг ўзгарувчан қисмлари, шунинг учун уларнинг ҳар бирининг номлари олдидан $ белгиси туриши лозим бўлади. Параметрлар рўйхатида ҳеч қандай кўп нуқталарни қўйиш мумкин эмас. Учинчидан, амаллар блоки сўзи билан бирга функция танасида исталган тўғри PHP-код мавжуд бўлиши керак (параметрларга мувофиқ бўлиши мажбурий эмас). Ва ниҳоят, return калит сўзидан сўнг тартибли php-ифода келиши лозим (маънога эга бўлган қандайдир символлар). Бундан ташқари, функцияда қайтарилувчи маъно каби параметрлар бўлмаслиги ҳам мумкин. Функцияни тўғри эълон қилишга мисол – юқорида келтирилган факториални ҳисоблаш функцияси.

Функцияни қачон чақириш мумкин? Бу ғалати савол бўлиб туюлиши мумкин. Функцияни уни аниқлангандан кейин чақириш мумкин, яъни function f_name(){...} блокидан пастда исталган дастур қаторида. PHP3 да бу айнан шундай. Лекин PHP4 да бундай талаб йўқ. Ҳамма гап интерпретатор олинган кодни қандай қайта ишлашида. Биргина истисно шартли равишда аниқланадиган функциядан ташкил топади (шартли операторлар ёки бошқа функциялар ичида). Функция шу тарзда аниқланган тақдирда, уни аниқлаш уни чақиришдан олдин бажарилади.

Агар функция дастур ичида аниқланган бўлса, уни кейин қайта аниқлаш ёки ўчириб ташлаш мумкин эмас. Функция номларига регистр таъсир қилмаслигига қарамасдан, яхшиси функцияни аниқлаш пайтида берилган ном билан чақириш мумкин бўлади.

Функцияларнинг аргументлари

Ҳар бир функцияда, аввал айтганимиздай, аргументлар рўйхати бўлиши мумкин. Бу аргументлар ёрдамида функцияга ҳар хил маълумотлар берилади (масалан, факториали ҳисобланиши керак бўлгансон маъноси). Ҳар бир аргумент ўзгарувчи ва константага эга бўлади.

Аргументлар ёрдамида маълумотлар функцияга уч хил турли усуллар билан ўтказилиши мумкин. Бу аргументларни маъносига кўра (ўзгармас ҳолатда фойдаланилади), иловаларга кўра ва ўзгармас ҳолатда аргументларга маъно беришга кўра ўтказиш . Бу усулларни атрофлича кўриб чиқамиз.

Аргумент функцияга маъносига кўра ўтказилса, функция ичидаги аргумент маъносининг ўзгариши унинг функция ташқарисидаги маъносига таъсир қилмайди. Функцияга унинг аргументларини ўзгартиришга йўл қўйиш учун уларни ҳаволаларга кўра ўтказиш керак. Бунинг учун аргумент номи олдидан функцияни аниқлашда ампенсанд “&” белгисини ёзиш керак.

Функцияда тинч ҳолатда фойдаланилаётган аргументлар маъносини аниқлаш мумкин. Айни пайтдаги маънонинг ўзи константа ифода бўлиши, ўзгартириш ва синф вакили ёки бошқа функция чақируви бўлмаслиги лозим.

Бизда информацион хабар тузувчи функция, унга берилган параметр маъносига мувофиқ тарзда ўзгарувчи имзо бор. Агар параметр маъноси берилмаган бўлса, “Ташкилий қўмита” имзосидан фойдаланилади.

Мисол. Тинч ҳолатдаги аргумент маъноси

```
<?php
function Message($sign="Таш.қўмита."){
// бу ерда параметр sign айни пайтда “Таш.қўмита” маъносига эга
  echo "Кейинги йиғилиш эртага бўлиб ўтади.<br>";
  echo "$sign<br>";
}
Message();
   // Параметрсиз функцияни чақирамиз. Бу ҳолда имзо – Бу Ташкилий қўмита
Message("Ҳурмат билан Камолиддин");
   // Бу ҳолда имзо "Ҳурмат билан Камолиддин." бўлади
?>
```

Бу скрипт ишининг натижаси қуйидагича:

Кейинги йиғилиш эртага бўлиб ўтади.
Ташкилий қўмита.
Кейинги йиғилиш эртага бўлиб ўтади.
Ҳурмат билан Камолиддин.

Агар функциянинг бир неча параметрлари бўлса, тинч ҳолатда маъно берилувчи бу аргументлар функция аниқланишида бошқа барча аргументлардан кейин ёзилиши керак. Акс ҳолда, агар бу аргументлар функцияни чиқариш пайтида кўздан қочирилса хато юзага келиши эҳтимоли бор.

Масалан, биз каталогга мақола тавсифини киритмоқчимиз. Фойдаланувчи мақолага унинг номланиши, муаллифи ва қисқа тавсиф каби характеристикаларни келтириши лозим бўлади. Агар Фойдаланувчи мақола муаллифи исмини киритмади, у Мурод Ёқубов деб олайлик.

```
<?php
function Add_article($title, $description, $author="Мурод Ёқубов"){
echo "Мақолани каталогга киритамиз: $title,";
echo "муаллиф $author";
echo "<br>Қисқа тавсиф: ";
echo "$description <hr>";
}
Add_article("Информатика ва биз","Бу мақола информатикага оид ...",
"Зайниддин Саидов");
Add_article("Характерлар ким", "Бу мақола характерлар ҳақида ...");
?>
```

Скрипт иши натижаси сифатида қуйидагиларни оламиз

Каталогга мақола киритамиз:
Информатика ва биз,
Муаллиф Мурод Ёқубов.
Қисқа тавсиф:

Бу мақола информатикага оид...

Каталогга мақола киритамиз:
Характерлар ким,
Муаллиф Одил Зияев.
Қисқа тавсиф:
Бу мақола характерлар ҳақида...

Агар биз қуйидагича ёзсак:

```
<?php
function Add_article($author="Одил Зияев", $title, $description){
// ... аввалги мисолдаги каби амал
}
Add_article("Характерлар ким", "Бу мақола характерлар ҳақида...");
?>
```

Натижа қуйидагича бўлади:

```
Warning: Missing argument 3 for add_article() in
c:\users\nina\tasks\func\def_bad.php on line 2
```

Ўзгарувчан узунлик аргументлари рўйхатлари

PHP4 да аргументларнинг ўзгарувчан сони билан функция тузиш мумкин. Яъни биз уни неча аргументлар билан чақирилишини билмасдан, функция тузамиз. Бу каби функция ёзиш учун ҳеч қандай махсус синтаксис керак бўлмайди. Ҳаммаси унинг ичига ўрнатилган функциялар func_num_args(), func_get_arg(), func_get_args() ёрдами билан қилинади.

func_num_args() функцияси аргументлар сонини қайтаради. Бу функция фақат фойдаланувчи функциясини аниқлаш мобайнида фойдаланиши мумкин. Агар у функциядан ташқарида пайдо бўлса, интерпретатор огоҳлантириш беради.

func_get_arg функцияси (аргумент_рақами тўлалигича) аргументни ўзгаришлар рўйхатидан аргументлар функциясига қайтаради, унинг тартиб рақами func_get_arg параметри билан берилади. Функция аргументлари нолдан бошлаб ҳисобланади. **func_num_args**() каби бу функция фақат бирон-бир функцияни аниқлашда фойдаланилади.

Аргумент рақами функцияга ўзгарган аргументлар сонидан ортиб кетиши мумкин эмас. Акс ҳолда огоҳлантириш умумлаштирилади ва func_num_args() функциясига False қиймат қайтади.

Маълумотларни текшириш учун функцияга унинг аргументларини тузамиз. Агар функциянинг биринчи аргументи – бутун сон, иккинчиси – қатор бўлса, текшириш муваффақиятли ўтди, деб ҳисоблаймиз.

func_get_args() фунцияси аргументлар рўйхатидан ташкил топган массив қайтаради. Массивнинг ҳар бир элементи аргументга, функция ўзгаришига тўғри келади. Агар функция фойдаланувчи функцияси аниқлигидан ташқарида фойдаланилса огоҳлантириш умумлаштирилади.

Аввалги мисолни кўчирамиз, бу функциядан фойдаланамиз. Функцияни ҳаракатлантирувчи жуфт аргумент бутун сон эканлигини текширамиз:

Мисол:

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
  // функцияга ўзгарган аргументлар сони
  $args = func_get_args();
  // функция аргументлари массиви
  for ($i=0;$i<$n;$i++){
    $v = $args[$i];
    if ($i % 2 == 0){
      if (!is_int($v)) $check = false;
// текширамиз, жуфт аргумент бутунми-йўқми
}
}
return $check;
}
if (DataCheck(array("text", 324)))
   echo "Текширув тўғри ўтди<br>";
else echo "Маълумотлар шартларни қониқтирмайди <br>";
?>
```

Бундан, func_num_args(), func_get_arg() ва func_get_args() функция комбинацияси функциялар ўзгарувчан аргументлар рўйхатига эга бўла олиши учун фойдаланилади. Бу функциялар фақат PHP4 га киритилган. PHP3 да шундай натижага эришиш учун, аргумент сифатида массив функциясидан фойдаланиш мумкин бўлади. **Масалан**, ҳар бир тоқ функциялар параметри бутун сонлигини текширувчи скриптни қуйидагича ёзиш мумкин:

```
<?
function DataCheck($params){
  $check =true;
$n = count($params);
// функцияга ўзгарган аргументлар сони
  for ($i=0;$i<$n;$i++){
    $v = $params[$i];
    if ($i % 2 !== 0){
```

```
// текширамиз, тоқ аргумент бутунми-йўқми
if (!is_int($v)) $check = false;
    }
  }
return $check;
}
if (DataCheck("text", 324))
   echo "Текширув тўғри ўтди<br>";
else echo "Маълумотлар шартларни қониқтирмайди<br>";
?>
```

Функциялар ичида ўзгарувчилардан фойдаланиш

1. Глобал ўзгарувчилар
2. Статистик ўзгарувчилар
3. Қайтарилувчан маънолар
4. Ҳаволани қайтариш

Глобал ўзгарувчилар

Функциялар ичида ундан ташқарида берилган ўзгарувчилардан фойдаланиш учун, бу ўзгарувчиларни глобал деб эълон қилиш керак. Бунинг учун функция танасида унинг номларини global калит сўзидан кейин келтириш лозим бўлади:

```
global $var1, $var2;
<?
$a=1;
function Test_g(){
global $a;
  $a = $a*2;
  echo ' $a=',$a функция ишида натижа;
}
echo 'функциядан ташқарида $a=',$a,', ';
Test_g();
echo "<br>";
echo функциядан ташқарида $a=',$a,', ';
Test_g();
?>
```

мисол. Глобал ўзгарувчилар

Бу скрипт ишидан қуйидаги натижаларни оламиз:

$a=2 функциядан ташқарида, $=2 функция ишида натижа

$a=2 функциядан ташқарида, $=4 функция ишида натижа

Ўзгарувчи глобал деб эълон қилинганда, аниқ глобал ўзгарувчи учун ҳавола тузилади. Бунинг учун бундай ёзув қуйидагига эквивалент

(GLOBALS массиви мавжуд кўриниш соҳаларига мувофиқ барча глобал ўзгарувчиларни ўз ичига олади):
$var1 = & $GLOBALS["var1"];
$var2 = & $GLOBALS["var2"];

Бундан келиб чиқадики, $var1 ўзгарувчини ўчириш $_GLOBALS["var1"] глобал ўзгарувчиини ўчириб ташламайди.

Статистик ўзгарувчилар

Ўзгарувчилардан фақат функция ичида фойдаланиш учун бунда унинг маъносини сақлаган ҳолда ва функциядан чиққандан сўнг, бу ўзгарувчиларни статистик деб эълон қилиш керак. Статистик ўзгарувчилар фақат функциялар ичида кўринади ва дастурни юклаш функция доирасидан ташқарига чиқса ўз маъносини йўқотмайди. Бу ўзгарувчиларни эълон қилиш static калит сўзи ёрдамида амалга оширилади:

static $var1, $var2;

Ҳар қандай маъно статистик ўзгарувчи сифатида талқин қилиниши мумкин, фақат ҳавола эмас.
Мисол. Статистик ўзгарувчилардан фойдаланиш
```
<?
function Test_s(){
static $a = 1;
// ифода ёки ҳаволани ўзлаштириб бўлмайди
  $a = $a*2;
  echo $a;
}
Test_s(); // 2 чиқади
echo $a;  // ҳеч нарса чиқмайди, зеро $a фақат функция ичида кириш йўлаги бор
Test_s(); // $a=2 функция ичида, шунинг учун функция иши натижаси 4 сони бўлади
?>
```

Қайтарилувчан маънолар

Юқорида мисол қилиб келтирилган барча функциялар бирор-бир амал бажаришган. Бундай ҳоллардан ташқари, ҳар қандай функция ўз иши натижаси сифатида қандайдир қиймат қайтаради. Бу return тасдиғи ёрдамида қилинади. Қайтарилувчан қиймат ҳар қандай турда, шу жумладан, рўйхат ва объектлар бўлиши мумкин. Интерпретатор функция танасида return командасига учраганда, у дарҳол уни бажаришни тўхтатади ва функция чақирилган қаторга ўтиб кетади.

Масалан, инсон ёшини қайтарувчи функция тузамиз. Агар инсон вафот этмаган бўлса, ёш жорий йилга мувофиқ ҳисобланади.

```
<?php
/* агар иккинчи параметр true каби ҳисоблаб чиқилса, у вафот этган санадай кўриб чиқилади, */
function Age($birth, $is_dead){
  if ($is_dead) return $is_dead-$birth;
  else return date("Y")-$birth;
}
echo Age(1971, false); // выведет 33
echo Age(1971, 2001); // выведет 30
?>
```

Бу мисолда return функциясидан фойдаланмаса ҳам бўлади, шунчаки уни чиқариш функциясини echo га алмаштирилади. Аксинча, агар биз функция бирор-бир қиймат қайтарадиган қилсак (бу мисолда инсон ёши), биз дастурда ўзгарувчини бу функция қийматини исталган ўзгарувчига ўзлаштиришимиз мумкин.

```
$my_age = Age(1981, 2004);
```

Функция иши натижасида фақат битта қиймат қайтарилиши мумкин. Бир неча қийматни қийматлар рўйхати қайтарилган тақдирда олиш мумкин (бир ўлчамли массив). Биз инсон ёшини кунигача аниқликда олмоқчимиз, деб ҳисоблайлик.

```
<?php
function Full_age($b_day, $b_month, $b_year)
{
	$y = date("Y");
	$m = intval(date("m"));
	$d = intval(date("d"));
	$b_month = intval($b_month);
	$b_day = intval($b_day);
	$b_year = intval($b_year);

	$day = ($b_day > $d ? 30 - $b_day + $d : $d - $b_day);
	$tmpMonth = ($b_day > $d ?  -1 : 0);
	$month = ($b_month > $m + $tmpMonth
	 ? $b_month + $tmpMonth - $m : $m+$tmpMonth - $b_month);
	$tmpYear  = ($b_month >  $m + $tmpMonth ?  -1 : 0);
	if ($b_year > $y + $tmpYear)
	{
		$year = 0; $month = 0; $day = 0;
	}
	else
	{
		$year = $y + $tmpYear - $b_year;
```

```
    }
    return array ($day,$month,$year);
}
$age = Full_age("29","06","1986");
echo "Сиз $age[2] ёш, $age[1] ойлар ва $age[0] кунлар";
?>
```

Функция бир неча қийматларни уларни дастурда қайта ишлаш учун қайтарганда, бир амал билан маънони бирданига бир неча ўзгарувчиларни ўзлаштиришга имкон берувчи list() тил конструкциясидан фойдаланиш қулай бўлади. Масалан, юқоридаги мисолда функцияни, унинг қийматига ўзгартириш киритмай қайта ишлаш қуйидагича бўлиши мумкин:

```
<?
// Full_age() функция киритиш
list($day,$month,$year) = Full_age("07",
   "08","1974");
echo "Сизнинг ёшингиз $year, $month ой ва
   $day кун";
?>
```

list() конструкциясини умуман ўзгарувчини ўзлаштириш учун исталган массив элементи қийматидан фойдаланиш мумкин.

Мисол. list() дан фойдаланиш

```
<?
$arr = array("first","second");
list($a,$b) = $arr;
  // ўзгарувчи $a ўзлаштирилади, биринчи массив қиймати, $b – иккинчи
echo $a," ",$b;
// «first second» қатори келтирилади
?>
```

Ҳаволани қайтариш

Функция ўз иши натижасида шунингдек ҳаволани бирор-бир ўзгарувчига қайтариши мумкин. Бу функцияни қандай ўзгарувчи ҳаволага ўзлаштириш кераклигини аниқлаш учун фойдаланилади. Функциядан ҳавола олиш учун, эълон олдидан амперсанд (&) белгисини ёзиш керак бўлади ва ҳар сафар функция чақируви пайтида унинг номи олдидан ҳам амперсанд (&) ёзиш керак бўлади. Кўпинча функция ҳаволани бирор-бир глобал ўзгарувчига (ёки унинг қисмини – ҳаволани глобал массив элементига), ҳаволани статистик ўзгарувчига (ёки унинг қисмини) ёки ҳаволани аргументлардан бирига қайтаради, агар у ҳавола бўйича берилган бўлса.

Мисол. Ҳаволани қайтариш

```
<?
$a = 3; $b = 2;
function & ref($par){
```

```
global $a, $b;
  if ($par % 2 == 0) return $b;
  else return $a;
}
$var =& ref(4);
echo $var, " и ", $b, "<br>"; // 2 ва 2 келтирилади
$b = 10;
echo $var, " и ", $b, "<br>";  // 10 ва10 келтирилади
?>
```

Ҳавола синтаксисидан фойдаланишда бизнинг мисолдаги $var ўзгарувчи ўзгарувчининг $b қиймати $ref қайтарилган функциясига кўчирилмайди, бу ўзгарувчига ҳавола тузилади. Демак, энди $var ва $b тенг кучли ўзгарувчилар ва улар бир пайтда ўзгартирилади.

Функциянинг ўзгарувчилари

PHP функциялар ўзгарувчиларига кўмаклашади. Бу дегани, агар ўзгарувчи номи оддий қавслар билан тугаса, PHP шу каби номли функцияни қидиради ва уни бажаришга ҳаракат қилади.

Мисол. Функциялар ўзгарувчиларидан фойдаланиш

```
<?
/* Иккита оддий функция тузамиз: Add_sign – қаторга имзо қўшади ва Show_text –матн қаторини чиқариб беради*/
function Add_sign($string,
   $sign="Ҳурмат билан, Мурод"){
   echo $string ." ".$sign;
}
function Show_text(){
  echo "Хабарни почтадан жўнатиш<br>";
}
$func = "Show_text"; // маънога эга ўзгарувчи тузамиз, у функция номига тенг Show_text
$func();  // у  Show_text функцияни чақиради
$func = "Add_sign";  // маънога эга ўзгарувчи тузамиз, у функция номига тенг Add_sign
$func("Ҳаммага салом <br>");
// бу функцияни чақиради Add_sign "Ҳаммага салом" параметрли
?>
```

Бу мисолда Show text функция шунчаки матн қаторини чиқаради. Агар echo махсус функцияси мавжуд бўлса, нега бунинг учун алоҳида функция тузиш керак, дейиш мумкин. Гап шундаки, echo(), print(), unset(), include() каби функциялардан функциялар ўзгарувчилари сифатида фойдаланиб бўлмайди. Яъни биз ёзсак:

```
<?
$func = "echo ";
$func("TEXT");
?>
```
Интерпретатор хатони кўрсатади:

```
Fatal error: Call to undefined function:
echo() in c:\users\nina\tasks\func\var_f.php on line 2
```

Шунинг учун юқорида келтириб ўтилган исталган функциялардан ўзгарувчилар функцияси сифатида фойдаланиш учун юқоридаги мисолдаги йўлни тутдик.

Синфлар ва объектлар

Объектга йўналтирилган дастурлашнинг асосий тушунчалари – синфлар ҳамада объектлардир. Бу тушунчаларни қуйидагича тушуниш мумкин: объект – бу дастурда қўлланиладиган тушунча ёки бирор физик предмет ҳақида маълумот берадиган структураланган ўзгарувчидир, синфлар эса бу объектларнинг тавсифи ва улар устида бажариладиган ҳаракатлардир.

PHP дастурлаш тилида синфлар қуйидаги синтаксис ёрдамида аниқланади:

```
class Синф_номи{
  var $хусусият_номи;
    /* хусусиятлар рўйхати */
  function метод_номи( ){
   /* усулларнинг танаси */
  }
  /*усуллар рўйхати*/
}
```

Синф объектлари хусусиятлари номи var калит сўзи ёрдамида эълон қилинади, берилган синф объектларига қўлланилган усуллар функция сифатида ишлатилади. Синф танаси ичида this калит сўзи ёрдамида тақдим қилинаётган жорий синфга мурожаатни амалга ошириш мумкин.

Масалан, биз мақола категориясини тасвирловчи синф тузишимиз керак. Ҳар бир мақоланинг номи, муаллифи ва қисқа мазмуни каби хусусиятлари бор. Биз мақола билан қандай амал бажармоқчимиз? Биз санаб ўтилган хусусиятларга маъно беришимиз, мақолани браузерда кўрсатишимиз керак бўлади. Шунда бу синфнинг ифодаланиши қуйидагича ҳолатда бўлади:

```
<?
class Articles { // Мақола синфини тузамиз
  var $title;
  var $author;
  var $description;
// мақола атрибути маъносини ўзлаштирувчи усул
```

```
  function make_article($t, $a, $d){
    $this->title = $t;
    $this->author = $a;
    $this->description = $d;
  }
//синф нусхасини ифодалаш учун усул
  function show_article(){
    $art = $this->title . "<br>" .
          $this->description .
      "<br>Муаллиф: " . $this->author;
    echo $art;
} } ?>
```

Шундай қилиб “мақола” туридаги физик объектларни тасвирлаш учун биз уч ўзгартувчидан ташкил топган, мақола характеристикасини ўзида жамлаган Articles номли синф ва муайян мақола тузиш ва уни тасвирлаш учун иккита функция туздик.

Маълумки, PHP билан ишлаш даврий ҳолатда HTML режимида юкланиши мумкин. Бу ҳолда дастур бир неча коднинг бўлаклари(блоклар)дан ташкил топади. Синфни ифодалаш php-коднинг ҳар хил блоклари бўйича ва қолаверса ҳар хил файллар бўйича тарқатилмаслиги керак. Яъни қуйидагича ёзсак:

```
<?php
class Articles { // Синфни тасвирлашнинг боши
  var $title;
?>
<?php   // синфни тасвирлашнинг давоми
   function show_article(){  // усулнинг таркиби
   }
} // синфни тасвирлашнинг якуни
?>
```

бунда дастур тартибли ишлайди.

Синф номи масаласида айрим нарсаларни эътиборда тутиш керак. Синфнинг номи PHP тилидаги объектлар номланиши қоидаларига жавоб бериши лозим, лекин бир қатор номлар борки, техник мутахассислар томонидан ўз мақсади учун захира қилинади. Биринчи навбатда бу номлар “_” қуйи чизиқдан бошланувчилардир. Синфлар ва функциялар тузиш учун бу каби номларни ишлатмаслик керак. Бундан ташқари stdClass номи захира қилинган, зеро у PHP сурилгичи ичида ишлатилади.

Такрорлаш учун саволлар:

1. PHP да функцияларни тушунтириб беринг.
2. PHP да функциялар аргументлари қандай эълон қилинади?
3. PHP да ўзгарувчилар эълонини тушунтиринг.

4. Ичма-ич функцияларни нима?
5. PHP да синфлар ва объектлар нима?

20-маъруза. Класлар ва объектлар. Хатоликлар билан ишлаш

Режа:

1.Шартли функция ичида функцияни аниқлаш

2. Аргументларни ҳаволасига кўра ўтказиш

3. Функциялар ичида ўзгарувчилардан фойдаланиш

Калит сўзлар: Глобал ўзгарувчилар, return, Make_event, Глобал ўзгарувчилар, Объектлар

Функциялар нима учун керак? Бу саволга жавоб бериш учун, функция ўзи нима эканлигини тушуниб олиш лозим бўлади. Дастурлашда, худди математикадаги каби, унга боғлиқ кўпгина аргументларнинг унинг кўпгина маъноларида акс этишидир. Демак, функция аргументнинг ҳар бир маънолари жамланмаси учун унинг бажарган иши натижаси сифатида қандайдир маъно қайтаради. Функциялар нима учун керак, буни мисоллар билан ойдинлаштиришга ҳаракат қиламиз. Дастурлашдаги функцияга классик мисол – бу соннинг факториал аҳамиятини ҳисоблаб берувчи функция. Демак, биз унга сон берамиз, у эса бизга унинг факториалини қайтаради. Бунда биз факториалини олишни хоҳлаган ҳар бир сон учун айнан бир хил кодни қайтаравермаймиз – бу сонга тенг бўлган аргументли функцияни чақиришнинг ўзи кифоя қилади.

Шу йўл билан биз бирон-бир маълумотга боғлиқлик зарурияти туғилган амални бажарганимизда, бу ҳолда ҳам биз айнан шундай амалларни бажаришимиз оширишимиз лозим бўлади, фақат бошқа бошланғич маълумотлардан фойдаланамиз, функциялар механизмидан фойдаланиш– *функция танаси* кўринишидаги амаллар блокини тахт қилиш, ўзгарувчан маълумотларни эса – унинг параметрлари сифатида фойдаланиш қулайроқ бўлади.

Функция топшириғи (эълони) умумий тарзда қандай бўлишини кўрамиз. Функция қуйидаги синтаксис ёрдамида аниқланади:

```
function  Функция_номи (1-параметр, 2-параметр,
... N-параметр){
        Амаллар блоки
return "функцияга айланувчи маъно";
}
```

Агар php-дастурда тўғридан-тўғри ёзилса, ҳеч нарсани ишлаб бўлмайди. Биринчидан, функция номи функция параметрлари номлари (1-параметр, 2-параметр ва б.) PHP да номланиш қоидаларига мувофиқ келиши керак (унда яхшиси кириллча символларни ҳам ишлатмаган маъқул). Функция номлари регистрга нисбатан сезувчан бўлади. Иккинчидан, функция параметрлари – тилнинг ўзгарувчан қисмлари, шунинг учун уларнинг ҳар бирининг номлари олдидан $ белгиси туриши лозим бўлади. Параметрлар рўйхатида ҳеч қандай кўп нуқталарни қўйиш мумкин эмас.

Учинчидан, амаллар блоки сўзи билан бирга функция танасида исталган тўғри PHP-код мавжуд бўлиши керак (параметрларга мувофиқ бўлиши мажбурий эмас). Ва ниҳоят, *return* калит сўзидан сўнг тартибли php-ифода келиши лозим (маънога эга бўлган қандайдир символлар). Бундан ташқари, функцияда қайтарилувчи маъно каби параметрлар бўлмаслиги ҳам мумкин. Функцияни тўғри эълон қилишга мисол – юқорида келтирилган факториални ҳисоблаш функцияси.

Функцияни қачон чақириш мумкин? Бу ғалати савол бўлиб туюлиши мумкин. Функцияни уни аниқлангандан кейин чақириш мумкин, яъни function f_name(){...} блокидан пастда исталган дастур қаторида. PHP3 да бу айнан шундай. Лекин PHP4 да бундай талаб йўқ. Ҳамма гап интерпретатор олинган кодни қандай қайта ишлашида. Биргина истисно шартли равишда аниқланадиган функциядан ташкил топади (шартли операторлар ёки бошқа функциялар ичида). Функция шу тарзда аниқланган тақдирда, уни аниқлаш уни чақиришдан олдин бажарилади.

```
<?
$make = true;
/* бу ерда Make_event() ни чақириш мумкин эмас;
Чунки у ҳали мавжуд эмас, лекин Save_info() ни чақириш мумкин*/

Save_info("Собир","Содиқов",
   "Мен PHP курсини танладим");

if ($make){
// Make_event() функциясини аниқлаш
  function Make_event(){
    echo "<p> Python<br> ни ўрганмоқчиман";
  }
}
// энди Make_event() ни чақириш мумкин
Make_event();
// Save_info функциясини аниқланади
function Save_info($first, $last, $message){
  echo "<br>$message<br>";
  echo "Исм: ". $first . " ". $last . "<br>";
}
Save_info("Мурод","Ёқубов",
        "Мен Lisp ни танладим");
// Save_info ни бу ерда ҳам чақириш мумкин
?>
```

мисол. Шартли функция ичида функцияни аниқлаш

Агар функция дастур ичида аниқланган бўлса, уни кейин қайта аниқлаш ёки ўчириб ташлаш мумкин эмас. Функция номларига регистр

таъсир қилмаслигига қарамасдан, яхшиси функцияни аниқлаш пайтида берилган ном билан чақириш мумкин бўлади.

```
<?php
/* маълумотларни сақлаш, яъни DataSave() функциясини чақириш мумкин эмас. Унинг тўғрилиги текширилмасдан олдин, яъни DataCheck() функцияси чақирилмасдан олдин бу мумкин эмас.*/

DataCheck();
DataSave();

function DataCheck(){
// маълумотлар тўғрилигини текшириш
  function DataSave(){
  // маълумотларни сақлаймиз
  }
}
?>
```

Функцияларнинг аргументлари

Ҳар бир функцияда, аввал айтганимиздай, аргументлар рўйхати бўлиши мумкин. Бу аргументлар ёрдамида функцияга ҳар хил маълумотлар берилади (масалан, факториали ҳисобланиши керак бўлгансон маъноси). Ҳар бир аргумент ўзгарувчи ва константага эга бўлади.

Аргументлар ёрдамида маълумотлар функцияга уч хил турли усуллар билан ўтказилиши мумкин. Бу аргументларни маъносига кўра (ўзгармас ҳолатда фойдаланилади), иловаларга кўра ва ўзгармас ҳолатда аргументларга маъно беришга кўра ўтказиш .

Бу усулларни атрофлича кўриб чиқамиз.

Аргумент функцияга маъносига кўра ўтказилса, функция ичидаги аргумент маъносининг ўзгариши унинг функция ташқарисидаги маъносига таъсир қилмайди. Функцияга унинг аргументларини ўзгартиришга йўл қўйиш учун уларни ҳаволаларга кўра ўтказиш керак. Бунинг учун аргумент номи олдидан функцияни аниқлашда ампенсанд “&” белгисини ёзиш керак.

Ўзгарувчан узунлик аргументлари рўйхатлари

PHP4 да аргументларнинг ўзгарувчан сони билан функция тузиш мумкин. Яъни биз уни неча аргументлар билан чақирилишини билмасдан, функция тузамиз. Бу каби функция ёзиш учун ҳеч қандай махсус синтаксис керак бўлмайди. Ҳаммаси унинг ичига ўрнатилган функциялар *func_num_args()*, *func_get_arg()*, *func_get_args()* ёрдами билан қилинади.

func_num_args() функцияси аргументлар сонини қайтаради. Бу функция фақат фойдаланувчи функциясини аниқлаш мобайнида фойдаланиши мумкин. Агар у функциядан ташқарида пайдо бўлса, интерпретатор огоҳлантириш беради.

```
<?php
function DataCheck(){
```

```
  $n = func_num_args();
  echo "Функция аргументлари сони $n";
}
DataCheck();
   // қаторни келтиради
   // "0 функция аргументлари сони"
DataCheck(1,2,3);
   // қаторни келтиради
   // "3-функция аргументлари сони"
?>
```

Функциялар ичида ўзгарувчилардан фойдаланиш

Глобал ўзгарувчилар

Функциялар ичида ундан ташқарида берилган ўзгарувчилардан фойдаланиш учун, бу ўзгарувчиларни глобал деб эълон қилиш керак. Бунинг учун функция танасида унинг номларини *global* калит сўзидан кейин келтириш лозим бўлади:

```
global $var1, $var2;
<?
$a=1;
function Test_g(){
global $a;
  $a = $a*2;
  echo ' $a=',$a функция ишида натижа;
}
echo 'функциядан ташқарида $a=',$a,', ';
Test_g();
echo "<br>";
echo функциядан ташқарида $a=',$a,', ';
Test_g();
?>
```

Функциянинг ўзгарувчилари

PHP *функциялар ўзгарувчиларига* кўмаклашади. Бу дегани, агар ўзгарувчи номи оддий қавслар билан тугаса, PHP шу каби номли функцияни қидиради ва уни бажаришга ҳаракат қилади.

```
<?
/* Иккита оддий функция тузамиз:
Add_sign – қаторга имзо қўшади ва
Show_text –матн қаторини чиқариб беради*/

function Add_sign($string,
   $sign="Ҳурмат билан, Мурод"){
   echo $string ." ".$sign;
```

```
}
function Show_text(){
  echo "Хабарни почтадан жўнатиш<br>";
}
$func = "Show_text";
   // маънога эга ўзгарувчи тузамиз,
   // у функция номига тенг Show_text
$func();
   // у  Show_text функцияни чақиради
$func = "Add_sign";
   // маънога эга ўзгарувчи тузамиз,
   // у функция номига тенг Add_sign
$func("Ҳаммага салом <br>");
   // бу функцияни чақиради
   // Add_sign "Ҳаммага салом" параметрли
?>
```

мисол. Функциялар ўзгарувчиларидан фойдаланиш

Бу мисолда Show text функция шунчаки матн қаторини чиқаради. Агар echo махсус функцияси мавжуд бўлса, нега бунинг учун алоҳида функция тузиш керак, дейиш мумкин. Гап шундаки, echo(), print(), unset(), include() каби функциялардан функциялар ўзгарувчилари сифатида фойдаланиб бўлмайди. Яъни биз ёзсак:

```
<?
$func = "echo ";
$func("TEXT");
?>
```

Интерпретатор хатони кўрсатади:

```
Fatal error: Call to undefined function:
   echo() in
c:\users\nina\tasks\func\var_f.php on line 2
```

Шунинг учун юқорида келтириб ўтилган исталган функциялардан ўзгарувчилар функцияси сифатида фойдаланиш учун юқоридаги мисолдаги йўлни тутдик.

Ички жойлашган (ичма-ич) функциялар.

Фойдаланувчи томонидан аниқланадиган функциялар ҳақида гапирганда ички жойлашган функциялар ҳақида гап кетмаслиги мумкин эмас. Юқорида биз echo(), print(), date(), include() каби ички жойлашган функциялар билан танишдик. Бундан ташқари date() функциядан бошқа барча функциялар *PHP* дастурлаш тили конструкциясига эга. Улар *PHP* дастурлаш тили ядросига жойлашган бўлиб, ҳеч қандай модуллар ва қўшимча ўзгартиришлар талаб этмайди. Аммо шундай функциялар мавжудки, уларга турли файл библиотекалари ва мос равишда модулларни

юкламасдан иложи йўқ. Масалан, MySql маълумотлар базаси билан ишлайдиган функциялардан фойдаланиш учун шундай кенгайтмали файлларни қўллаб қувватлайдиган компонентлари керак. Охирги вақтларда бу функциялардан фойдаланиш учун қўшимча компонентлар керак эмас, чунки уларнинг барчаси ҳозирда *PHP* дастурлаш тили ядросига киритилган.

Синфлар ва объектлар.

Объектга йўналтирилган дастурлашнинг асосий тушунчалари – *синфлар* ҳамада *объектлар*дир. Бу тушунчаларни қуйидагича тушуниш мумкин: ***объект*** – бу дастурда қўлланиладиган тушунча ёки бирор физик предмет ҳақида маълумот берадиган структураланган ўзгарувчидир, ***синф***лар эса бу объектларнинг тавсифи ва улар устида бажариладиган ҳаракатлардир.

PHP дастурлаш тилида *синф*лар қуйидаги синтаксис ёрдамида аниқланади:

```
class Синф_номи{
  var $хусусият_номи;
    /*хусусиятлар рўйхати*/
  function метод_номи( ){
   /* усулларнинг танаси */
  }
  /*усуллар рўйхати*/
}
```

Синф объектлари хусусиятлари номи **var** калит сўзи ёрдамида эълон қилинади, берилган синф объектларига қўлланилган усуллар функция сифатида ишлатилади. Синф танаси ичида *this* калит сўзи ёрдамида тақдим қилинаётган жорий синфга мурожаатни амалга ошириш мумкин.

Масалан, биз мақола категориясини тасвирловчи синф тузишимиз керак. Ҳар бир мақоланинг номи, муаллифи ва қисқа мазмуни каби хусусиятлари бор. Биз мақола билан қандай амал бажармоқчимиз? Биз санаб ўтилган хусусиятларга маъно беришимиз, мақолани браузерда кўрсатишимиз керак бўлади. Шунда бу синфнинг ифодаланиши қуйидагича ҳолатда бўлади:

```
<?
class Articles { // Мақола синфини тузамиз
  var $title;
  var $author;
  var $description;
// мақола атрибути
// маъносини ўзлаштирувчи усул
  function make_article($t, $a, $d){
    $this->title = $t;
    $this->author = $a;
    $this->description = $d;
  }
//синф  нусхасини  ифодалаш  учун усул
```

```
  function show_article(){
    $art = $this->title . "<br>" .
          $this->description .
      "<br>Муаллиф: " . $this->author;
    echo $art;
  }
}
?>
```

Шундай қилиб “мақола” туридаги физик объектларни тасвирлаш учун биз уч ўзгартувчидан ташкил топган, мақола характеристикасини ўзида жамлаган *Articles* номли синф ва муайян мақола тузиш ва уни тасвирлаш учун иккита функция туздик.

Маълумки, PHP билан ишлаш даврий ҳолатда HTML режимида юкланиши мумкин. Бу ҳолда дастур бир неча коднинг бўлаклари(блоклар)дан ташкил топади. Синфни ифодалаш php-коднинг ҳар хил блоклари бўйича ва қолаверса ҳар хил файллар бўйича тарқатилмаслиги керак. Яъни қуйидагича ёзсак:

```
<?php
class Articles { // Синфни тасвирлашнинг боши
  var $title;
?>
<?php
// синфни тасвирлашнинг давоми
   function show_article(){
   // усулнинг таркиби
   }
} // синфни тасвирлашнинг якуни
?>
```

бунда дастур тартибли ишлайди.

Синф номи масаласида айрим нарсаларни эътиборда тутиш керак. *Синф*нинг номи PHP тилидаги *объектлар* номланиши қоидаларига жавоб бериши лозим, лекин бир қатор номлар борки, техник мутахассислар томонидан ўз мақсади учун захира қилинади. Биринчи навбатда бу номлар “_” қуйи чизиқдан бошланувчилардир. *Синфлар* ва функциялар тузиш учун бу каби номларни ишлатмаслик керак. Бундан ташқари stdClass номи захира қилинган, зеро у PHP сурилгичи ичида ишлатилади.

Ўзгарувчиларни инициаллаштириш

Баъзан айрим *синф* атрибутларига маънони *синф* иштирокчисини тузиш биланоқ ўзлаштириш керак бўлади. Биз мақола *синф*ини тузганимизда, *синф* атрибутлари (*хусусиятлари*) маъноларини ўзлаштириш учун махсус функция make_article() дан фойдаландик. Умуман олганда, биз тўғри йўл тутмадик, чунки “велосипед ихтироси” билан шуғулландик. Синф

атрибутларининг бошланғич маъноларини бериш учун махсус иккита стандарт *усул* мавжуд. PHP4да маънони var оператори ёки *конструктор* функцияси ёрдамида инициаллаштириш мумкин. var ёрдамида фақат константа маъноларни инициаллаштириш мумкин. Константа бўлмаган маъноларни бериш учун *объект синф*дан ажраб чиққанда ўз-ўзидан ишга тушувчи *конструктор* фунциясидан фойдаланилади. *Конструктор*-функция у ифодаланган бутун *синф*га мос келувчи номга эга бўлиши керак.

Мисол келтирамиз. Айтайлик, “мақола” *объект*ини тузишда биз унинг хусусиятларини қуйидагича белгилайлик: муаллифлар – “Камолов” қаторига тенг, номланиш ва қисқа мазмун - $_POST глобал массиви элементларига мос, мақола нашри – мазкур санада. Унда синфнинг қуйидаги ифодаси PHP4да батартиб бўлмайди:

```
<?
class Articles { // мақола синфини тузамиз
  var $title= $_POST["title"];
  var $author = "Камолов";
  var $description = $_POST["description"];
  var $published = date("Y-m-d");
// синф атрибути
// маъносини ўзлаштирувчи усул
  }
?>
```

Синфнинг қуйидаги ифодаси эса PHP4да кераклича йўсинда ишлайди:

```
<?
class Articles { // мақола синфини тузиш
  var $title;
  var $author = "Камолов";
  var $description;
  var $published;
// синф атрибути
// маъносини ўзлаштирувчи усул
 function Articles(){
  $this->title = $_POST["title"];
  $this->description = $_POST["description"];
  $this ->published = date("Y-m-d");
 }
}
?>
```

PHP3 ва PHP4 да *конструкторлар* ҳар хил ишлашини ҳисобга олиш керак. Функция PHP3 да, агар у синфники каби номга эга бўлса, *консртуктор*га айланади, PHP4 да эса – агар у ифодаланган синфники каби номга эга бўлса шундай бўлади. Бир синф бошқасини кенгайтирганда ва *хусусиятларнинг ва база синфлар усулларининг эргашишида* усуллар орасидаги фарқ кўриниб турибди. Лекин биз бу ҳақда бироз кейинроқ

гапирамиз. PHP5да *синф конструктори* _construct деб номланади. Бундан ташқари, PHPда деструкторлар – *объект*ни йўқ қилишда ўз-ўзидан ишга тушувчи функциялар пайдо бўлди. PHP5 да функция-деструктор destruct деб номланиши керак бўлади.

Объектлар

Биринчи маърузалардан бирида биз PHP да объект деб аталувчи тип мавжудлиги ҳақида айтиб ўтгандик. ***Синф*** – бу объект типидаги маълумотларнинг бир туридаги ифодаланишидир. Синфлар реал ўзгарувчилар учун шаблон вазифасини ўтайди. Керакли типдаги ўзгартувчи new оператори ёрдамида *синф*дан тузилади. Объектни тузиб, биз барча усулларни қўллашимиз ва барча синф ифодасида кўрсатиб ўтилган хусусиятларни олишимиз мумкин бўлади. Бунинг учун қуйидагича синтаксисдан фойдаланилади: $объект_номи->хусусият ёки _номи$объект_номи->усулнинг_номланиши(аргументлар рўйхати). *Хусусиятлар* ёки *усулар* номлари олдидан $ белгиси қўйилмайди.

```
<?php
$art = new Articles;
   // объект тузамиз $art
echo ($art ->title);
   // объекга номланиш берамиз $art
$another_art = new Articles;
   // объект тузамиз $another_art
$another_art->show_article();
   // объектнинг браузердаги ифодаси учун
   // усулни чақирамиз
?>
```

Мисол: Объект усуллари ва хусусиятларига эркин кириш (доступ)

Синфнинг ҳар бир объекти айнан бир хил хусусиятлар ва усулларга эга бўлади. Демак, $art объектда ва $another_art объектда title, description, author хусусиятлари ва Articles(), show_article() усуллари мавжуд. Лекин булар икки хил объектлар. Объектни файллар системасидаги директория деб ҳисоблаймиз, унинг характеристикаси эса – бу директориядаги файллар сингари бўлсин. Аниқки, ҳар бир директорияда бир хил файллар ётиши мумкин, лекин шундай бўлса-да, улар ҳар хил директорияларда сақланаётгани учун ҳар хил ҳисобланиши мумкин. Худди шунингдек, хусусиятлар ва усуллар ҳам, агар улар турли объектларга қўлланиладиган бўлса, ҳар хил ҳисобланади. Юқори босқичдаги директориядан керакли файлни олиш учун бу файлга йўлни батафсил ёзиб чиқамиз. Синфлар билан ишлаш мобайнида биз чақиришни истаган функциянинг номини тўлиқ ёзишимиз керак бўлади. PHP даги юқори босқич директорияларига глобал

ўзгарувчиларнинг бўш ўрни бўлади, йўл эса -> тақсимловчиси ёрдамида кўрсатилади. Шу тарзда $art->title ва $another_art->title номлари икки хил турли ўзгарувчиларни англатади. PHP да ўзгарувчи ном олдидан фақат битта доллар белгисига эга бўлади, шунинг учун $art->$title кўринишида ёзиш мумкин эмас. Бу конструкция $art объектининг title хусусиятига мурожаат сифатида кўриб чиқилмайди, $title ўзгартувчи кўринишида берилган номли хусусият сифатида кўрилади (масалан, $art->"").

```
<?php
$art->title = " Internet га кириш";
   // объект хусусияти маъносини
   // шундай ўрнатиш мумкин
$art->$title = "Internet га кириш";
   // объект хусусияти маъносини
   // бундай ўрнатиб бўлмайди
$property = "title";
$art->$property = "Internet га кириш";
   // объект хусусияти маъносини
   // шундай ўрнатиш мумкин
?>
```

Такрорлаш учун саволлар:

1. Ички жойлашган (ичма-ич) функциялар.

2. Ўзгарувчиларни инициаллаштириш

3. Объект усуллари ва хусусиятларига эркин кириш

21-маъруза. PHPда маълумотлар базалари билан ишлаш. Саҳифани малумотлар базаси билан боғлаш

Режа:

1. Маълумотлар базаси ҳақида тушунча
2. Маълумотлар базаси интерфейси
3. Маълумотлар базаси билан боғланиш
4. Маълумотлар базаси устида амаллар

Калит сўзлар:маълумотлар базаси, MySql дастури, PHP да MySql билан боғланиш

Маълумотлар базаси ҳақида тушунча

Ушбу бўлим PHP ва MySql МББС ўртасидаги ҳамкорлик усуллари билан танишишга мўлжалланган. Асосий эътибор маълумотлар базаси билан боғланишни ўрнатиш, сўровлар жўнатиш функциялари ва жавобларни (mysql_connect, mysql_query, mysql_result, mysql_num_rows, mysql_close) қайта ишлашга қаратилади. Мисол сифатида виртуал тарих музейи маъмурияти учун web-интерфейс тузиш масаласини кўрайлик. PHP дистрибутивида MySQL маълумотлар базаси билан ишлаш учун мўлжалланган функциялар мавжуд. Бунда бу функцияларнинг MySql даги баъзи бир маълумотлар базасини тасвирлаш ва тўлдириш мақсадида web-интерфейсларни тузиш имконини берувчи функциялар билан танишамиз. Маълумотлар базасига маълумотларни қўшиш учун web-интерфейс билан ишлашда бу маълумотларни шунчаки html-формага киритиш ва уларни серверга жўнатиш керак бўлади.

Намойиш этишда бу интерфейсни виртуал музей экспонатлари ҳақидаги маълумотлар сақланадиган Artifacts жадваллари учун тузамиз. Artifacts коллекциясидаги ҳар бир экспонат қуйидаги характеристика ёрдамида тасвирланишини эслатиб ўтамиз:

ном (title);
муаллиф (author);
ифода (description);
ўриндош ном (alternative);
тасвир (photo).

Номланиш ва ўриндош номланиш узунасига 255 белгидан кам сатр (яъни VARCHAR(255)), тасвирлаш – матнли майдон (TEXT турига мансуб) ҳисобланади, “муаллиф” ва “тасвир” майдонларида эса Persons коллекциясидан муаллифнинг идентификаторлари ва Images коллекциясидан экспонат тасвирларига мувофиқ мавжуд бўлади.

Маълумотлар базаси интерфейси

Маълумотлар базасидаги мавжуд жадвал структурасини (яъни унинг майдонлари жамланмасини) html-формада тасвирлаш учун қуйидаги таркибий топшириқларни режалаштириш мумкин:

- МБ билан уланишни ўрнатиш;
- МБ ишини танлаш;
- Жадвал майдонлари рўйхатини олиш;
- html-формада майдонларни тасвирлаш.

Бундан кейин формага киритилган маълумотларни маълумотлар базасига киритиш мумкин.

Маълумотлар базаси билан боғланиш (MySQL дастури мисолида)

Алоқа ўрнатиш

Маълумотлар базаси билан алоқа ўрнатиш учун **mysql_connect** функциясидан фойдаланилади.

mysql_connect синтаксиси

mysql_connect ресурси (“сервер қатори”, “username”, “password”)

Бу функция MySQL сервери билан алоқа ўрнатади ва бу алоқага кўрсаткич қайтаради ёки муваффақиятсиз чиққанда FALSE кўрсатади. Одатда қуйидаги параметрлар қиймати эълон қилинади:

server = 'localhost:3306'

username = сервер жараёни эгасидан фойдаланувчи исми

password = бўш парол

Сервер билан уланиш, агар у бунгача mysql_close() ёрдамида ёпилмаган бўлса, скриптни амалга ошириш тугалланишида база билан алоқа ёпилади.

Мисол:

```
<?
$conn = mysql_connect("localhost", "nina","123") or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_close($conn);
?>
```

mysql_connect амали

shell>mysql -u nina -p123 буйруғи билан тенг кучли.

Маълумотлар базаси устида амаллар

Маълумотлар базаларини танлаш

MySQL да маълумотлар базасини танлаш use буйруғи ёрдамида амалга оширилади:

mysql>use book;

PHP да бунинг учун mysql_select_db функцияси мавжуд.

mysql_select_db: синтаксиси

мантиқий mysql_select_db (database_name қатори);

Бу функция TRUE қийматни маълумотлар базасини муваффақиятли танланганда қайтаради ва FALSE ни эса – аксинча бўлганда.

Мисол: Book маълумотлар базасини танлаш

```
<?
$conn = mysql_connect("localhost","user","123") or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_select_db("book");
?>
```

Жадвал майдонлари рўйхатини олиш

PHP да маълумотлар базаси билан боғланилгандан сўнг, ундаги жадваллар рўйхатини олиш мумкин. Бу функция - mysql_list_fields.

mysql_list_fields синтаксиси

mysql_list_fields (database_name қатори, table_name қатори)

mysql_field_name функцияси сўров амалга оширилиши натижасида олинган майдон номини қайтаради. mysql_field_len функцияси майдон узунлигини қайтаради. mysql_field_type функцияси майдон типини қайтаради, mysql_field_flags функцияси эса пробел билан ёзилган майдон байроқлари рўйхатини қайтаради. Майдон типлари int, real, string, blob ва ҳ. бўлиши мумкин. Байроқлар not_null, primary_key, unique_key, blob, auto_increment ва ҳ. бўлиши мумкин.

Бу барча буйруқлар синтаксиси бир хил:

mysql_field_name (result қатори, бутун field_offset) ресурси;
mysql_field_type (result қатори, бутун field_offset) ресурси;
mysql_field_flags (result қатори, бутун field_offset) ресурси;
mysql_field_len (result қатори, бутун field_offset)

Бу ерда result – бу сўров натижаси идентификатори (масалан, mysql_list_fields ёки mysql_query функциялар билан жўнатилган сўров), field_offset эса – натижадаги майдоннинг тартиб рақами.

mysql_num_rows(result ресурси) буйруғи result нинг кўпгина натижалари қаторлари миқдорини қайтаради.

Мисол: Artifacts (экспонатлар коллекцияси) жадвали майдонлари рўйхатини олиш.

```
<?
$conn = mysql_connect("localhost","user","123") or die("Алоқа ўрнатиб бўлмайди: ". mysql_error());
echo "Алоқа ўрнатилди";
mysql_select_db("book");
$list_f = mysql_list_fields ("book","Artifacts",$conn);
$n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
  $type = mysql_field_type($list_f, $i);
  $name_f = mysql_field_name($list_f,$i);
  $len = mysql_field_len($list_f, $i);
  $flags_str =  mysql_field_flags ($list_f, $i);
```

```
echo "<br>Майдон номи: ". $name_f;
echo "<br>Майдон тури: ". $type;
echo "<br>Майдон узунлиги: ". $len;
echo "<br>Майдон байроқлари қатори: " . $flags_str . "<hr>";
}
?>
```

Натижа сифатида тахминан қуйидагиларни олиш мумкин (албатта, жадвалда иккита майдон бўлганда):

- Майдон номи: id
- Майдон тури: int
- Майдон узунлиги: 11
- Майдон байроқлари қатори:
- not_null primary_key auto_increment
- Майдон номи: title
- Майдон тури: string
- Майдон узунлиги: 255
- Майдон байроқлари қатори:

html-формада майдонлар рўйхатининг акс этиши

Майдон ҳақидаги маълумотни html-форма элементида акс эттирамиз. BLOB туридаги элементларни textarea га ўтказамиз (TEXT турида биз тузган description майдони BLOB типига эга), рақамлар ва қаторларни <input type=text> матнли қаторларида акс эттирамиз, автоинкремент белгисига эга элементни эса умуман акс эттирмаймиз, чунки унинг маъноси ўз-ўзидан ўрнатилади.

Бунинг учун explode функциясидан фойдаланилади:

explode: синтаксиси

explode массиви (separator қатори, string қатори , int limit)

Бу функция string қаторини separator тақсимлагичи ёрдамида қисмларга бўлади ва олинган қаторлар массивини қайтаради.

Бизнинг ҳолатда тақсимлагич сифатида пробел “ ” ни олиш кера, бўлиш учун бошланғич қатор сифатида эса – майдон байроқлари қаторини.

Мисол. Artifacts жадвалига маълумот киритиш учун форма (index.php)

```
<?php
  $dblocation = "localhost";
  $dbname = "book";
  $dbuser = "root";
  $dbpasswd = "";
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
@mysql_select_db($dbname,$dbcnx);
if(isset($_POST['save_hide'])) {
$title=($_POST['title']);
$query = "INSERT INTO Artifacts (title) VALUES ('$title')";
 if(@mysql_query($query))
```

```
 {
  echo "<HTML><HEAD><META HTTP-EQUIV='Refresh' CONTENT='0;
URL=index.php'></HEAD></HTML>";
 } else {  print mysql_error();  error("Маълумотни базага езишда хатолик");}
} else {
?>
<TABLE width="100%" align=center border=0><TR><TD>
<form method="POST" name="save" action="" enctype="multipart/form-data" >
<input type=hidden name=save_hide value=save_hide>
<TABLE width="100%" align=center border=0><TR>
<TD>Экспонат номини киритинг </TD>
<TD><INPUT class="input" size=“35” name=“title” value=""></TD>
</TR></TABLE>
<input type="submit" value="Saqlash" class="batton" />
</form></TD></TR></TABLE>
<? } ?>
```

Маълумотлар базасига маълумотлар ёзиш

Маълумки, маълумотларни жадвалга ёзиш учун SQL тилидаги INSERT буйруғи ишлатилади:

```
mysql> INSERT INTO Artifacts SET title='Eksponat nomi';
```

PHP скриптда бундай буйруқдан фойдаланиш учун mysql_query() функцияси мавжуд.

mysql_query синтаксиси

mysql_query ресурси (query қатори)

mysql_query() SQL-сўровни MySQL маълумотлар базасининг маълумотлар базасига жўнатади. Агар очиқ алоқа бўлмаса, функция параметрсиз mysql_connect() функциясига ўхшаш ҳолда МББТ билан боғланишга уринади.

Сўров натижаси буферланади.

Такрорлаш учун саволлар:

1. Маълумотлар базаси деганда нимани тушунасиз?
2. Қанака маълумотлар базаси дастурларини биласиз ва уларни имкониятлари?
3. PHP да MySql билан боғланиш функциясини тушунтириб беринг?
4. PHP да MySql сўровларини амалга оширишга мисол келтиринг?

22-маъруза. MySQL малумотлар базаси устида амаллар.

Режа:

1. PHP ва MySql МББС (СУБД) ўртасидаги ҳамкорлик
2. Маълумотни қўшиш учун интерфейс тузиш
3. Маълумотлар базаларини танлаш
4. html-формада майдонлар рўйхатининг акс этиши

Калит сўзлар: МБ билан уланишни ўрнатиш, МБ ишини танлаш, Жадвал майдонлари рўйхатини олиш:

Ушбу бўлим PHP ва MySql МББС (СУБД) ўртасидаги ҳамкорлик усуллари билан танишишга мўлжалланган. Асосий эътибор маълумотлар базаси билан боғланишни ўрнатиш, сўровлар жўнатиш функциялари ва жавобларни (mysql_connect, mysql_query, mysql_result, mysql_num_rows, mysql_close) қайта ишлашга қаратилади. Мисол сифатида виртуал тарих музейи маълумотлар базасининг маъмурияти учун web-интерфейс тузиш масаласини кўрайлик. PHP дистрибутивига MySQL маълумотлар базаси билан ишлаш учун мўлжалланган функцияларни олувчи кенгайтма киради. Бу бўлимда MySQL билан ишлаш учун баъзи бир маълумотлар базасини тасвирлаш ва тўлдириш мақсадида web-интерфейсларни тузиш топшириғини ечиш учун керак бўладиган асосий функциялар билан танишамиз. Савол туғилади: бундай интерфейсларни тузиш нега керак? SQL сўровлар тили билан нотаниш одамлар маълумотни маълумотлар базасига киритиш ва унинг таркибини кўриб туриш имконияти бўлиши учун шундай қилинади. Маълумотлар базасига маълумотларни қўшиш учун web-интерфейс билан ишлашда бу маълумотларни шунчаки html-формага киритиш ва уларни серверга жўнатиш керак бўлади, бизнинг скрипт эса қолган барча амалларни бажаради. Жадвал таркибини кўриб туриш учун ҳавола устига бир марта босиш ва керакли саҳифага кириш кифоя.

Кўриниб туриши учун бу интерфейсларни виртуал музей экспонатлари ҳақидаги маълумотлар жойланадиган Artifacts жадваллари учун тузамиз. Аввалги бўлимда бу коллекцияга структурани ҳамда унинг шахс (Persons) ва тасвирлар (Images) тавсифлари коллекциялари билан алоқасини киритган эдик. Artifacts коллекциясидаги ҳар бир экспонат қуйидаги характеристика ёрдамида тасвирланишини эслатиб ўтамиз:

- ном (title);
- муаллиф (author);
- ифода (description);
- ўриндош ном (alternative);
- тасвир (photo).

Номланиш ва ўриндош номланиш узунасига 255 белгидан кам сатр (яъни VARCHAR(255)), тасвирлаш – матнли майдон (TEXT турига мансуб) ҳисобланади, “муаллиф” ва “тасвир” майдонларида эса Persons

коллекциясидан муаллифнинг идентификаторлари ва Images коллекциясидан экспонат тасвирларига мувофиқ мавжуд бўлади.

Маълумотни қўшиш учун интерфейс тузиш

Демак, бизда маълумотлар базасида бирон-бир жадвал бор. Бу жадвалга маълумотни қўшишга мўлжалланган интерфейс тузиш учун унинг структурасини (яъни унинг майдонлари жамланмасини) *html-форма*да тасвирлаш керак бўлади.

Бу топшириқни қуйидаги таркибий топшириқларга бўлиб чиқамиз:

- *МБ билан уланишни ўрнатиш*;
- *МБ ишини танлаш*;
- *Жадвал майдонлари рўйхатини олиш*;
- *html-форма*да майдонларни тасвирлаш.

Бундан кейин формага киритилган маълумотларни маълумотлар базасига киритиш керак бўлади.

Бу топшириқларнинг барчасини тартиб билан кўриб чиқамиз.

Алоқа ўрнатиш

Демак, биринчи галдаги вазифа – бу маълумотлар базаси билан алоқа ўрнатиш. *mysql_connect* функциясидан фойдаланамиз. *mysql_connect* синтаксиси

mysql_connect ресурси ([сервер қатори[, username қатори[, password қатори[, мантиқий new_link[, бутун client_flags]]]]])

Бу функция MySQL сервери билан алоқа ўрнатади ва бу алоқага кўрсаткич қайтаради ёки муваффақиятсиз чиққанда FALSE кўрсатади. Етишмаётган параметрлар учун қуйидаги маънолар муваққат қўйилади:

server = 'localhost:3306'

username = сервер жараёни эгасидан фойдаланувчи исми

password = бўш парол

Агар функция айнан бир хил параметрлар билан икки марта чақириладиган бўлса, янги уланиш ўрнатилмайди, аксинча ҳавола эски алоқага қайтарилади. Бундан қочиш учун эса, ҳар қандай ҳолатда янги бир алоқа очишга мажбур қилувчи new_link параметридан фойдаланилади.

client_flags параметри – бу қуйидаги константалар комбинацияси:

MYSQL_CLIENT_COMPRESS (сиқиш протоколидан фойдаланилади), *MYSQL_CLIENT_IGNORE_SPACE*

(функция номидан сўнг пробел қўйишга рухсат беради),

MYSQL_CLIENT_INTERACTIVE (interactive_timeout бир сония кутиш - wait_timeout – уланиш тугатилишигача). new_link параметри PHP 4.2.0 да мавжуд, client_flags эса – PHP 4.3.0 да.

Сервер билан уланиш, агар у бунгача *mysql_close()* ёрдамида ёпилмаган бўлса, скриптни амалга ошириш тугалланишида беркилади.

Шундай қилиб, “123” паролли nina фойдаланувчиси учун локал серверда маълумотлар базаси билан уланишни ўрнатамиз.

```
<?
$conn = mysql_connect(
   "localhost", "nina","123")
or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_close($conn);
?>
```

***mysql_connect* амали**

shell>mysql -u nina -p123 буйруғи билан тенг кучли.

Маълумотлар базаларини танлаш

Уланишни амалга оширгандан сўнг бирга ишлашимиз керак бўлган маълумотлар базасини танлашимиз керак бўлади. Бизнинг маълумотларимиз book маълумотлар базасида сақланади.

MySQL да маълумотлар базасини танлаш use буйруғи ёрдамида амалга оширилади:

```
mysql>use book;
```

PHP да бунинг учун *mysql_select_db* функцияси мавжуд

mysql_select_db: синтаксиси

```
мантиқий mysql_select_db (
   database_name қатори
   [,link_identifier ресурси])
```

Бу функция TRUE қийматни маълумотлар базасини муваффақиятли танланганда қайтаради ва FALSE ни эса – аксинча бўлганда.

Book маълумотлар базасини ишга туширамиз:

```
<?
$conn = mysql_connect(
   "localhost","nina","123")
or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_select_db("book");
?>
```

Жадвал майдонлари рўйхатини олиш

Энди топшириқни ечиш билан алоҳида шуғулланса бўлади. Жадвал майдонлари рўйхатини қандай олиш мумкин? Жуда оддий. PHP да бу ҳолда ҳам ўзига тегишли буйруқ мавжуд - mysql_list_fields.

mysql_list_fields синтаксиси

```
   mysql_list_fields ( ресурси
   database_name қатори,
   table_name  қатори
```

[, ресурс link_identifier])

Бу функция майдонлар рўйхатини database_name маълумотлар базасидаги table_name жадвалига қайтаради. Келиб чиқадики, маълумотлар базасини танлаш бизга мажбурий эмас, лекин кейинроқ асқотади. Бу функция ишининг натижасини – ресурс типи ўзгарувчисини қандай баҳолаш мумкин. Яъни бу биз олишни хоҳлаган нарса эмас. Бу уларнинг номлари, типлари ва байроқлари киradиган жадвал майдонлари ҳақидаги маълумотларни олиш учун фойдаланиш мумкин бўлган ҳавола. *mysql_field_name* функцияси сўров амалга оширилиши натижасида олинган майдон номини қайтаради. *mysql_field_len* функцияси майдон узунлигини қайтаради. *mysql_field_type* функцияси майдон типини қайтаради, *mysql_field_flags* функцияси эса пробел билан ёзилган майдон байроқлари рўйхатини қайтаради. Майдон типлари int, real, string, blob ва б. бўлиши мумкин. Байроқлар not_null, primary_key, unique_key, blob, auto_increment ва б. бўлиши мумкин.

Бу барча буйруқлар синтаксиси бир хил:

mysql_field_name (result қатори, бутун field_offset) ресурси;
mysql_field_type (result қатори, бутун field_offset) ресурси;
mysql_field_flags (result қатори, бутун field_offset) ресурси;
mysql_field_len (result қатори, бутун field_offset)

Бу ерда *result* – бу сўров натижаси идентификатори (масалан, mysql_list_fields ёки *mysql_query* функциялар билан жўнатилган сўров(бу ҳақда кейинроқ гапириб ўтилади)), field_offset эса – натижадаги майдоннинг тартиб рақами. Умуман олганда, mysql_list_fields ёки *mysql_query* туридаги функцияларни қайтарувчи жадвални, аниқроғи унинг унга бўлган кўрсаткичини акс эттиради. Бу жадваллардан аниқ маънолар олиш учун махсус бу жадвални остки қатор сифатида ўқувчи функцияларни ишга тушириш керак. Бу функцияларга *mysql_field_name* ва шу кабилар ҳам киради. Сўровни амалга ошириш натижаси жадвалидаги барча қаторларни саралаш учун бу жадвалдаги қаторлар миқдорини билиш керак.

mysql_num_rows(result ресурси) буйруғи result нинг кўпгина натижалари қаторлари миқдорини қайтаради.

Энди эса Artifacts (экспонатлар коллекцияси) жадвали майдонлари рўйхатини олишга уриниб кўрамиз.

```
<?
$conn = mysql_connect(
   "localhost","nina","123")
or die("Алоқа ўрнатиб бўлмайди: ". mysql_error());
echo "Алоқа ўрнатилди";
mysql_select_db("book");
$list_f = mysql_list_fields (
  "book","Artifacts",$conn);
  $n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
```

```
  $type = mysql_field_type($list_f, $i);
  $name_f = mysql_field_name($list_f,$i);
  $len = mysql_field_len($list_f, $i);
  $flags_str =  mysql_field_flags (
            $list_f, $i);
echo "<br>Майдон номи: ". $name_f;
echo "<br>Майдон тури: ". $type;
echo "<br>Майдон узунлиги: ". $len;
echo "<br>Майдон байроқлари қатори: ".
   $flags_str . "<hr>";
}
?>
```

Натижа сифатида тахминан қуйидагиларни олиш мумкин (албатта, жадвалда иккита майдон бўлганда):

```
Майдон номи: id
Майдон тури: int
Майдон узунлиги: 11
Майдон байроқлари қатори:
  not_null primary_key auto_increment
Майдон номи: title
Майдон тури: string
Майдон узунлиги: 255
Майдон байроқлари қатори:
```

html-формада майдонлар рўйхатининг акс этиши

Энди бироз юқоридаги мисолни текшириб кўрамиз. Майдон ҳақидаги маълумотни шунчаки чиқармаймиз, уни муносиб *html-форма* элементида акс эттирамиз. BLOB туридаги элементларни textarea га ўтказамиз (TEXT турида биз тузган description майдони BLOB турига эга эканлигини эътибордан қочирмаймиз), рақамлар ва қаторларни <input type=text> кириш қисми матнли қаторларида акс эттирамиз, автоинкремент белгисига эга элементни эса умуман акс эттирмаймиз, чунки унинг маъноси ўз-ўзидан ўрнатилади.

Буларнинг барчаси анчайин оддий ҳал қилинади, байроқлар рўйхатидаиг auto_increment байроғи бундан мустасно. Бунинг учун explode функциясидан фойдаланилади:

explode: синтаксиси

```
explode массиви (separator қатори,
   string қатори [, int limit])
```

Бу функция string қаторини separator тақсимлагичи ёрдамида қисмларга бўлади ва олинган қаторлар массивини қайтаради.

Бизнинг ҳолатда тақсимлагич сифатида пробел “ ” ни олиш кера, бўлиш учун бошланғич қатор сифатида эса – майдон байроқлари қаторини.

Такрорлаш учун саволлар:

1. html-формада майдонлар рўйхатининг акс этиши
2. Жадвал майдонлари рўйхатини олиш
3. Маълумотни қўшиш учун интерфейс тузиш

23-маъруза. Cookie, сеанслар, FTP ва e-mail технологиялари.

Режа:
1. Cookie ни ўрнатиш
2. Cookie ни ўқиш
3. FTP
4. E-mail

Калит сўзлар: Cookie, сеанс, FTP, E-mail

Cookie ни ўрнатиш

Cookie — клиент компьютерида сақланувчи ва у ҳар сафар серверга мурожаат қилаётганда web-серверга юбориладиган матн сатридир. Шу тариқа маълумотлар турли скриптлар аро формалар ёки URL адресларсиз узатилиши мумкин. Cookie қийматини ўрнатиш учун қуйидаги синтаксисдан иборат бўлган setcookie функциясидан фойдаланилади:

bool setcookie (string name [, string value [, int expire [, string
path [, string domain [, bool securej]]]])

Функция клиент компьютерида сақланувчи cookie ларни тасниф этади. Қуйида ушбу функция параметрларининг таснифи кетирилган:
• name. Cookie нинг номи.
• value. Cookie нинг қиймати.
• expire. Cookie нинг амал қилиш муддати. Агар уберилса, у тугагач cookie ўчириб ташланади. Агар кўрсатилмаса, cookie браузер ойнаси ёпилгандан сўнг ўчириб ташланади
• path. Сервердаги cookie га рухсат этилган адрес.
• domain. cookie га рухсат этилган домен.
• secure. HTTPS протокол орқали боғланишда cookie нинг ҳавфсизлик белгиси. Стандарт ҳолатда cookie HTTPS учун ҳам HTTP даги каби ишлайди.

Кўриниб турибдики, cookie HTTP-сўровнинг қисми ҳисобланади ва у браузерга юборилади. Шу сабаб HTML-код формалаштиришдан олдин унинг қиймати ўрнатилиши керак. Бу шуни англатадики, setcookie функциясининг қиймати HTML-тегдан олдин ва echo операторигача ўрнатилади.

Агар бу қоида сақланмаса setcookie функциясини чақириш FALSE қиймат қайтаради ва бу cookie формировка қилинишида хатолик бўлганлигини англатади. cookie нинг тўғри формировка қилиниши функциянинг TRUE қиймат қайтаришига олиб келади. Бу cookie клиентга қабул қилинди дегани эмас албатта. Чунки браузерни созлашда cookie ўчириб ташлаш ёки серверга жўнатмаслик параметрлари ўрнатилган бўлиши мумкин.

Cookie ни ўқиш

cookie қиймати ўрнатилганидан сўнг, саҳифа қайта юкланмагунича дарҳол ундаги скрипт учун cookie га рухсат мавжуд бўлмайди. Чунки cookie фойдаланувчи компьютерида сақланади ва браузер орқали web-серверга жўнатилади. Бундан ташқари, cookie қачонки унинг домени сервер домени билан мос тушгандагина жўнатилади.

cookie га рухсатни олиш учун махсус **$_COOKIE** суперглобал массивидан фойдаланилади. Массивнинг қиймати сифатида олдин ишлатилган cookie номи олинади. Массив одатда скрипт юкланаётган вақтда **$_GET**, **$_POST** ва **$_REQUEST** массивлари билан автоматик тарзда тўлдирилади

cookie дан фойдаланишдан олдин унинг қиймати ўрнатилганлигига ишонч ҳосил қилинг. Бунинг учун isset функциясидан фойдаланиш жуда қулай. Қуйидаги мисолда message номли cookie нинг қиймати текширилиши ва кўрсатилиши келтирилган.

Мисол:

```
<HTML>
<HEAD> <TITLE> Cookie нинг қийматини ўқиш </TTTLE> </HEAD>
<BODY>
<CENTER>
<H1> Cookie нинг қийматини ўқиш </Hl>
Cookie нинг қиймати:
<?php
if (isset ($_COOKIE ['message']))
{
echo Cookie нинг қиймати:' . $_COOKIE ['message' ] ;
}
else
{
echo 'Cookie ўрнатилмаган' ;
}
?>
</CENTER;.
<BODY>
</HTML>
```

Cookie массивларда ҳам ташкил этилган бўлиши мумкин. Масалан қуйида учта cookie ўрнатилган:

```
setcookie ("cookie[one]" , "Бугун");
setcookie ("cookie[two] ", "Ҳаёт");
setcookie ("cookie[three] ", "гўзал!") ;
```

Натижада $_COOKIE['cookie'] массив қийматлари қуйидаги тарзда босмага чиқарилиши мумкин:

```
if (isset ($_COOKIE ['cookie'])) {
foreach ($_COOKIE'['cookie'] as $data)
```

```
{
echo "$data <BR>";
…
```

FTP

Қуйида FTP нинг баъзи атамаларини кўриб чиқамиз:

1. **FTP**

File Transfer Protocol (файлларни алмашиш протоколи); тармоқ орқали (хусусан Интернет) файлларни узатиш протоколи. Буни бир компьютердан бошкасига файл нусхасини кўчириш каби тушуниш мумкин.

2. **FTP Server**

Бу файлни жунатилиши сўровини кутувчи компьютер ёки сервер.

3. **FTP Client**

Бу FTP серверга сўров жўнатувчи компьютер. Сўров текшируви ёки тасдиқланишидан сўнг FTP клиент компьютери маълумотларни серверга юклаши ёки сервердан юклаб олиши мумкин.

4. **Аноним FTP**

Ушбу серверга FTP клиент компьютери орқали авторизациядан ўтмасдан боғланиши мумкин. Бундай имкониятни бир нечта веб сайтларда кўришимиз мумкин, қайсики рўйхатдан ўтмасдан маълум файлларни юклаб олиш имкониятини беради.

5. **FTP Host**

Ушбу хизмат FTP сайт сифатида, яъни сайтда рўйхатдан ўтиб, сайтда келтирилган маълумотлар файлларини юклаб олиш имкониятини берувчи компьютер. Бу фойдаланувчилар сони чекланган пуллик хизмат ҳисобланади.

6. **FTP сайт**

FTP хост компьютери тасарруфидаги фойдаланувчи логин пароли талаб қилинувчи веб саҳифа. Хост серверда бир нечта веб-саҳифа жойланиши мумкин, бунда ҳар бир сайт учун алоҳида фойдаланувчи авторизацияси мавжуд.

7. **FTP Proxy**

FTP Proxy бу шунақа сервер компьютерки, бунда сўров FTP серверга жўнатилади, ушбу сўров прокси орқали ўтиб, кейин зарур манзилга йўналтирилади.

E-mail

Бу хизмат интернет тизими орқали ўзаро почта хабарлари алмашинуви усулидир. Бундай тизим оралиқ сақланиш методи асосида йўлга қўйилган. Замонавий интернет протоколлар хабарлар жўнатиш иконияти бўйича юқори даражага кўтарилди. Почта жўнатишнинг энг кенг тарқалган протоколи SMTP (Simple Mail Transport Protocol) ҳисобланади.

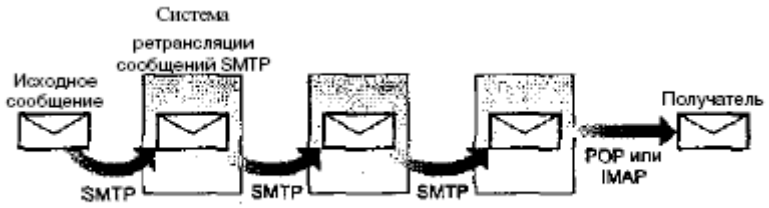


Рис. 22.2. Для пересылки и получения почты используются различные протоколы

Почтани қабул қилиш почта протоколи POP (Post Office Protocol), ёки Интернет хабарларини қабул қилувчи протокол IMAP (Internet Message Access Protocol) дан фойдаланилади.

Такрорлаш учун саволлар:

1. Cookie нима?
2. Сеанс нима?
3. FTP протокол ва унинг вазифаси?
4. E-mail ва унинг вазифаси?

МУНДАРИЖА: