ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АХБОРОТ ТЕХНОЛОГИЯЛАРИ ВА КОММУНИКАЦИЯЛАРНИ РИВОЖЛАНТИРИШ ВАЗИРЛИГИ
ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ
ҚАРШИ ФИЛИАЛИ

“Компьютер инжиниринг” факультети
“Дастурий инжиниринг” кафедраси

“WEB - ДАСТУРЛАШ”

ФАНИ БЎЙИЧА

МАЪРУЗА МАТНЛАРИ

5350400 – “Ахборот технологиялари соҳасида касб таълими” йўналиши талабалари учун

Қарши 2015

Тузувчи: Тошкент ахборот технологиялари университети Қарши филиали “Дастурий инжиниринг” кафедраси катта ўқитувчиси Ч.М. Хидирова

Ушбу маъруза матнлари 5350400 – “Ахборот технологиялари соҳасида касб таълими” йўналишида “WEB-дастурлаш” фани маъруза машғулотларни олиб бориш учун асосий қўлланма ҳисобланиб, назарий маълумотлардан ташкил топган.

“Web дастурлаш” фани бўйича маъруза матнлари Тошкент ахборот технологиялари университети Қарши филиали “Дастурий инжиниринг” кафедра мажлисида кўриб чиқилди ва тасдиқланди (28 август 2015 йил, №1 - сонли баённома) ҳамда, филиал “Компьютер инжиниринг” факультетининг илмий-услубий кенгашида муҳокама қилинган ва чоп этишга тавсия этилган (29август 2015 йил, №1 - сонли баённома).

КИРИШ

Ушбу маъруза матнлари 5350400 – “Ахборот технологиялари соҳасида касб таълими” йўналиши талабаларига “WEB-дастурлаш” фанини ўтиш учун мўлжалланган.

Қўлланманинг мақсади - “WEB-дастурлаш” фани бўйича назарий билимларни, улардаги мавжуд катталик ва тушунчаларни, замонавий технологияларни, WEB-иловалар яратиш тиллари ва уларнинг бир биридан фарқларини ажратишни назарий жиҳатдан ўргатишдир.

Қўлланма назарий малумотлар ва кўрсатмалардан ташкил топган.

Қўлланма “WEB-дастурлаш” курси дастурига тўғри келади ва талабаларнинг замонавий WEB-технологиялар билан ишлаши, танишиши учун бевосита қўл келади.

1-Маъруза

Мавзу: WEB-ДАСТУРЛАШ ФАНИГА КИРИШ

Маъруза режаси:

1. Web дастурлаш фани ва унинг вазифалари
2. HTML белгилашлари: HTML, XML, XHTML, WML.
3. Сценарийли тиллар. "Клиент-сервер" технологияси

Таянч иборалар: *Web технология, HTML, XML, XHTML, WML, Perl, ASP, PHP, JSP, SSI, JavaScript, VBScript, Java.*

1. Web дастурлаш фани ва унинг вазифалари

Гиперматнли маълумотлар тизими маълумотлар бўғинлари тўпламлари, шу бўғинларда аниқланган гиперматнли алоқалар тўпламлари ва бўғинлар ва алоқаларни бошқариш инструментларидан ташкил топган. World Wide Web технологияси бу – гиперматнли тақсимланган системаларни Интернетга киритиш технологияси ва шундан келиб чиқиб, у бундай тизимларнинг умумий таърифига мос келиши керак. Бу шуни билдирадики, гиперматнли тизимларнинг юқорида келтирилган барча компоненталари Web да ҳам бўлиши керак.

Web да гиперматнли тизимга икки хил нуқтаи назардан қараш мумкин. Биринчидан, ўзаро гиперматнли ўтишлар (ANCHOR контейнери) воситасида боғланган ҳолда тасвирланиши керак бўлган саҳифалар тўплами сифатида. Иккинчидан, тасвирланаётган саҳифалар (матн, графика, уяли код ва ҳоказолар) ни ташкил қилувчи элементлар маълумот объектларининг тўплами сифатида. Сўнгги ҳолатда саҳифанинг гиперматнли ўтишлар тўплами – бу матнга ички қўйилган расм каби маълумот бўлаги ҳисобланади.

Иккинчи ёндашувда гиперматнли тизим элементари маълумот объэктлари тўплами учун гиперматнли алоқалар ўрнини ўйновчи HTML-саҳифаларнинг ўзи томонидан аниқланади. Бу ёндашув тайёр компоненталардан тасвирланаётган саҳифаларни қуриш нуқтаи- назаридан анча серҳосилроқдир.

Web да саҳифаларни яратишда “клиэнт-сервер” архитектураси билан боғлиқ муаммо юзага чиқади. Саҳифаларни ҳам клиэнт томонида, ҳам сервер томонида яратиш мумкин. 1995 йилда Netscape компанияси мутахассислари *JavaScript* дастурлаш тилини ишлаб чиқиб, саҳифаларни клиэнт томонида бошқариш механизмини яратишди.

Шундай қилиб, *Javascript* – бу Webни гиперматнли саҳифаларини клиэнт томонида кўриш сенарияларини бошқариш тили. Янада аниқроқ айтадиган бўлса, *Javascript* – бу нафақат клиэнт томонидаги дастурлаш тили. Liveware *Javascript* тилининг авлоди бўлиб, Netscape сервери томонида ишловчи восита бўлади. Аммо *Javascript* тилини машҳур қилган нарса бу клиэнт томонида дастурлашдир.

Javascript нинг асосий вазифаси – *HTML-контейнерлар* атрибутларининг қийматларини ва кўрсатувчи муҳитининг *хоссалирини* HTML-сарлавҳаларини кўриш жараёнида фойдаланувчи томонидан ўзгартириш имкониятлари, бошқача айтганда уларни динамик сарлавҳалар қилиш (DHTML)дир. Яна шуни айтиш жоизки, сарлавҳалар қайта юкланмайди. Амалда буни, масалан, қуйдагича ифодалаш мумкин: сарлавҳанинг фонини, рангини ёки хужжатдаги расмни ўзгартириш, янги ойна очиш ёки огоҳлантириш ойнасини чиқариш ва ҳ.к.

Web-технология классификацияси

2. HTML белгилашлари: HTML, XML, XHTML, WML

Web-технологиянинг (Интернет-технология) Web-дизайн қисмини ўрганишни белгилаш тили таснифи билан бошлаймиз.

Махсус тил мавжуд бўлиб, бу тил ёрдамида матнлар, график маълумотлар Web-саҳифа хужжатга жойлаштирилади ва бу хужжатни барча компьютерда кўриш имконияти мавжуддир. Бундай махсус тиллар белгилаш тиллари (языки разметки) 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;

3. 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** ва **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 файлларни қайта ишлайди. ASP технологияси VBScript, Java ва JavaScript тилларини қўллаб қувватлайди. ASP-код ихтиёрий HTML-хужжатдан, шу билан бирга бошқа ASP-хужжатдан чақирилиши 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.

Назорат саволлари:

1. Web дастурлаш фанининг вазифалари нимадан иборат
2. Web дастурлаштириш тилларини санаб беринг
3. Белгилаш тиллари қайсилар?
4. Клиент томонидан дастурлаш
5. Сервер томонидан дастурлаш

Мавзу: HTML ГА КИРИШ. HTML НИНГ АСОСИЙ ТЕГЛАРИ

Маъруза режаси:

1. HTML хужжатининг тузилиши.
2. HTML асосий теглари.

Таянч иборалар: *html, body, img, align, href, bgcolor, fontcolor, em, kbd*

1. HTML хужжатининг тузилиши

HTML (Hyper Text Markup Language) – белгили тил бўлиб, яъни бу тилда ёзилган код ўз ичига маҳсус рамзларни мужассамлаштиради. Бундай рамзлар ҳужжат кўринишини фақатгина бошқариб, ўзи эса кўринмайди. HTMLда бу рамзларни тэг (тэг – ёрлик, белги) деб аталади. HTMLда ҳамма тэглар рамз-чегараловчилар (< , >) билан белгиланади. Улар орасига тэг идентификатори (номи, масалан B) ёки унинг атрибутлари ёзилади. Ягона истисно бу мураккаб чегараловчилар (<!-- ва -->) ёрдамида белгиланувчи шархловчи тэглардир.

Аксарият тэглар жуфти билан ишлатилади. Очувчи тэгнинг жуфти ёпувчи тэг. Иккала жуфт тэг фақатгина ёпувчи тэг олдидан «слэш» (“/”) белгиси қўйилишини ҳисобга олмаганда, деярли бир хил ёзилади. Жуфт тэгларнинг асосий фарқи шундаки, ёпувчи тэг параметрлардан фойдаланмайди. Жуфт тэг яна контейнер деб ҳам аталади. Жуфт тэглар орасига кирувчи барча элементлар тэг контейнери таркиби дейилади. Ёпувчи тэгда зарур булмаган бир қатор тэглар мавжуд. Баъзида ёпувчи тэглар тушириб қолдирилса ҳам замонавий браузерлар аксарият ҳолларда ҳужжатни тўғри форматлайди, бироқ буни амалда қўллаш тавсия этилмайди. Масалан, расм қўйиш тэги <IMG>, кейинги қаторга ўтиш
, база шрифтини кўрсатиш <BASEFONT> ва бошқалар ўзининг </IMG>, </BR> ва ҳоказо ёпувчи жуфтларисиз ёзилиши мумкин. Нотўғри ёзилган тэгни ёки унинг параметри браузер томонидан рад килинади. (бу браузер танимайдиган тэгларга ҳам тааллуқли). Масалан, <NOFRAME> тэг-контейнери фақатгина фреймларни танийдиган браузер томонидан ҳисобга олинади. Уни танимайдиган браузер <NOFRAME> тэгини тушунмайди.

Тэглар параметр ва атрибутларга эга бўлиши мумкин. Параметрлар йиғиндиси ҳар-бир тэгда индивидуалдир. Параметрлар қуйидаги коида асосида ёзилади:

- Тэг номидан сўнг пробеллар билан ажратилган параметрлар келиши мумкин;
- Параметрлар ихтиёрий тартибда келади;
- Параметрлар ўзининг номидан кейин келувчи «=» белгиси орқали берилувчи қийматларга эга бўлиши мумкин.
- Одатда параметрлар қиймати « » - «кўштирноқ» ичида берилади.
- Параметр қийматида баъзан ёзув регистри мухим.

Шуни эсда тутиш лозимки, ҳамма тэглар ўзининг индивидуал параметрига эга бўлишига карамай, шундай бир қатор параметрлар мавжудки, уларни <BODY> бўлимининг барча тэгларида ишлатиш мумкин. Бу параметрлар CLASS, ID, LANG, LANGUAGE, STYLE ва TITLEлардир.

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> 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.

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

Хужжатнинг HEAD бўлими.

HEAD бўлими сарлавҳа ҳисобланади ва у мажбурий тэг эмас, бироқ мукаммал тузилган сарлавҳа жуда ҳам фойдали бўлиши мумкин. Сарлавҳа қисмининг мақсади ҳужжатни таржима килаётган дастур учун мос ахборотни етказиб беришдан иборат. Ҳужжат номини кўрсатувчи <TITLE> тэгидан ташқари бу бўлимнинг қолган барча тэглари экранда акс эттирилмайди. Одатда <HEAD> тэги дарҳол <HTML> тэгидан кейин келади.

<TITLE> тэги сарлавҳанинг тэгидир, ва ҳужжатга ном бериш учун ҳизмат килади. Ҳужжат номи <TITLE> ва </TITLE> тэглар орасидаги матн қаторидан иборат. Бу ном браузер ойнасининг сарлавҳасида пайдо бўлади (бунда сарлавҳа номи 60 белгидан кўп бўлмаслиги лозим). Ўзгартирилмаган ҳолда бу матн ҳужжатга «закладка» (bookmark) берилганда ишлатилади. Ҳужжат номи унинг таркибини қисқача таърифлаши лозим. Бунда умумий маънога эга бўлган номлар (масалан, Homepage, Index ва бошқалар)ни ишлатмаслик лозим. Ҳужжат очилаётганда биринчи бўлиб унинг номи акс эттирилиши, сўнгра эса ҳужжат асосий таркиби кўп вақт олиб, кенгайиб кетиши мумкин бўлган форматлаш билан бирга юкланишини ҳисобга олган ҳолда, фойдаланувчи хеч булмаганда ушбу ахборот қаторини ўқий олиши учун ҳужжатнинг номи берилиши лозим.

Ҳужжатнинг BODY бўлими.

Ушбу бўлинма ҳужжатнинг таркибий қисмини ўз ичига олади. Бўлинма <BODY> тэгидан бошланиб </BODY> тэгида тугайди. Бироқ ушбу тэглар катъий мавжуд бўлиши шарт эмас, чунки браузерлар матнга қараб ҳужжат таркибий қисмининг ибтидосини аниклаши мумкин. <BODY> тэгининг бир қатор параметрлари мавжуд бўлиб, уларнинг бирортаси ҳам мажбурий эмас.

<BODY> тэги параметрлари:

ALINK – фаол мурожаат (ссылка)нинг рангини белгилайди.

BACKGROUND – фондаги тасвир сифатида фойдаланилувчи тасвирнинг URL-манзилини белгилайди.

BOTTOMMARGIN – ҳужжатнинг қуйи чегараларини пикселларда белгилайди.

BGCOLOR – ҳужжат фонининг рангларини белгилайди.

BGPROPERTIES – агар FIXED қиймати ўрнатилмаган бўлса, фон тасвири айлантирилмайди.

LEFTMARGIN – чап чегараларни пикселларда белгилайди.

LINK – хали кўриб чиқилмаган ссылканинг ранггин белгилайди.

RIGHTMARGIN – ҳужжат ўнг чегарасини пикселларда ўрнатади.

SCROOL – браузер дарчалари харакатлантириш (прокрутка) йўлакларини ўрнатади.

TEXT – матн рангини аниқлайди.

TOPMARGIN – юқори чегарасини пикселларда ўрнатади.

VLINK – ишлатилган мурожаат рангини белгилайди.

BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN ва TOPMARGIN паметрлари матн чегараси ва дарча четлари орасидаги масофани пикселларда белгилайди. (Фақат HTML 4.0 версиясидан бошлаб IE браузерлари бу параметрларни таний олади)

BGPROPERTIES параметри фақатгина битта FIXED қийматига эга. HTML даги ранглар ўн олтилик саноқ тизимида (RGB), ёки ранглар номи ёрдамида берилиши мумкин. Ранглар базаси 3 та рангга – қизил (R) , яшил (G) ва кўк (B) рангларга асосланган бўлиб, у RGB деб белгиланади. Ҳар-бир ранг учун 00 дан FF гача бўлган ўн олтилик саноқ тизимидаги қиймат берилади, бу эса 0 дан 255 гача бўлган диапазонга тўғри келади. Сўнгра бу қийматлар бир сонга бирлаштирилади ва уларнинг олдига “#” белгиси куйилади. Масалан, #800080 сиёҳрангни билдиради.
Мисоллар:
<BODY TEXT = “#000000”> ёки <BODY TEXT = black>
<BODY BGCOLOR = “#ffffff”> ёки <BODY BGCOLOR = WHITE>
<BODY LINK = “#ff0000”> ёки <BODY LINK = RED>
<BODY LINK = “#ooFFFF” ALINK = “#800080”> ёки <BODY VLINK = Aqua ALINK = PURPLE>

Ҳамма бараузлар ўн олтилик саноқ тизимидаги стандарт ранларни танийди. Булар қуйидагилардир:

Black = #000000	Maroon = #800000
Silver = #C0C0C0	Red = #FF0000
Grey = #808080	Purple = #800080
White = #FFFFFF	Fuchsia = #FF00FF
Green = #008000	Navy = #000080
Lime = #00FF00	Blue = #0000FF
Olive = #808000	Teal = #008080
Yellow = #FFFF00	Aqua = #00FFFF

Мисол:

```
<BODY
 BGCOLOR = AQUA
 TEXT = “#848484”
 LINK = RED
VLINK = PURPLE
ALINK = GREEN>
```

Агар BGCOLOR параметри рангни номи ёки унинг таркибий кисмларини ўн олтилик саноқ тизимидаги кодда келтириш вазифаси ёрдамида фон ранггини чиқариш учун ишлатилса, BACKGROUND тасвир ёрдамида саҳифага фон беришда фойдаланилади. Тасвир сифатида GIF ёки JPG форматидаги график файллар ишлатилади. HTML-ҳужжат фонидаги тасвир доимо бутун саҳифани тўлдириб туради. Агар тасвир ўлчами дарча ўлчамидан кичик бўлса, у мозайка тамойилига асосан купайтирилади. Одатда фон тасвири сифатида тармоқ орқали юклаш учун унча кўп вақт кетмайдиган кичик тасвир танлаб олинади, ёки фон сифатида шаффоф рельеф логотипи тасвиридан фойдаланилади.
Мисол:

<BODY BACKGROUND = texture.gif BGCOLOR = gray>.

Саҳифа яратилишида доимо фон рангини бериш тавсия қилинади. Агар фон тасвири ҳам берилаётган бўлса, фон ва тасвир ранглари бир-бирига яқин бўлгани маъқул.
Мисол:

```
<BODY
      TEXT = BLUE
      LINK = RED
```

VLINK = BLUE
ALINK = PINK
BACKGROUND=HYPERLINK("http://www.foo.com/jkorpela/HTML3.2/wave.gif"

Мисол:

```
<HTML>
    <HEAD>
        <TITLE> - саҳифа фонини бериш мисоли </TITLE>
    </HEAD>
        <BODY BGCOLOR = YELLOW
              TEXT = BLACK
              LINK = RED
              VLINK = PURPLE
              ALINK = GREEN>
        </BODY>  </HTML>
```

2. HTML асосий теглари

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 *курсив*да 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.

<STRONG> тэги матнини ажратиб кўрсатишда қўлланилади. Браузерлар одатда бундай матнни ўрта қалинликдаги (полужирный) шрифтда акс эттиради. Мисол:

<STRONG> HEAD </STRONG> бу <STRONG> хизмат </STRONG> табиатига эга ахборотлар ёзиб олинувчи ҳудуддир.

Бу ерда «HEAD» ва «хизмат» сўзлари ўрта қалинликда белгилаб берилади.

Физик форматлаш тэглари ўзларида кўрсатилган матн фрагментини браузер дарчасида акс эттириш форматини белгилайди. Баъзи тэглар HTML 4.0 спецификацияси томонидан бекор қилинган (deprecate) бўлсада, у браузерлар томонидан қўллаб келинади.

<B> тэги матнни ўрта қалинликдаги шрифтда акс эттиради. Мисол:

<B> Мультимедия </B> ўз ичига матн, графика, видео ва товушни камраб олади.

<I> тэги матнни *курсив* шаклда акс эттиради. Мисол:

<I> мультимедия </I> ўзида интерактив интерфейсни акс эттиради.

<U> тэги матннинг тагига чизиб беради (HTML 4.0 версиясида бу бекор қилинган тэгдир. Унинг ўрнига <STRONG> ёки <CITE> тэгларидан фойдаланиш тавсия этилади). Мисол:

WWW протоколи <U> Hyper Text Transfer Protocol </U> деб аталади.

<STRIKE> ва <S> тэглари горизонтал бўйича ўчирилган матнни акс эттиради (4.0 версиясида бу бекор қилинган тэг). Мисол:

Бу <STRIKE> учириб ташланган </STRIKE> матндан бир мисол.

<BIG> тэги матнни барча ҳарфларини катта ўлчамдаги шрифтда чиқариб беради. Масалан:

Бу <BIG> катта </BIG> ўлчамли шрифт.

<SMALL> тэги матнни кичик ўлчамдаги шрифтларда чиқариб беради. Масалан:

Бу <SMALL> кичик </SMALL> ўлчамли шрифт.

<SUB> тэги матнни қатор сатхидан пастга, индексга тушуриб, кичик ўлчамли шрифтда чиқариб беради. Бу математик индекслар ёзишда жуда қўл келади. Масалан:

А массивининг элементлари А ₁₁ а ₁₂ дан иборат.

Бу: “А массивининг элементлари A_{11} a_{12} дан иборат” шаклда пайдо бўлади.

<SUP> тэги матнни қатор сатҳидан юқорига суриб, даражага, кичик ўлчамли шрифтда чиқариб беради. Бу сонларнинг математик даражасини ёки сноскаларни ифодалашда асқотади. Мисол:

A ² + b ²

Бу ифода «$A^2 + b^2$» кўринишига эга бўлади.

<TT> тэги матнни бир хил кенгликдаги шрифтда (телетайп ҳолатида) акс эттиради. Масалан:

Бу <TT> бир хил кенгликдаги </TT> матндир.

Форматлаш тэглари бир-бирининг ичига солинган бўлиши мумкин. Бунда бир контейнернинг бутунлай бошқа бирининг ичида бўлиши мажбурий. Масалан,

Бу <B> ўрта қалинликдаги </B> шрифт <P>

Бу <I> курсив </I> шрифт <P>

Бу матн <B> <I> ҳам ўрта қалинликдаги, ҳам курсивдир </B> </I>.

<FONT> тэги шрифт параметрларини кўрсатиб беради. Шрифт параметрларининг бевосита ҳужжат матнида келтирилиши ҳужжат таркибий қисмининг ҳужжатни тақдим этиш шакли таърифидан ажралиб туриши лозимлиги тўғрисидаги асосий ғояни бузади. Шу туфайли HTML 4.0 спецификациясида <FONT> тэги ва <BASEFONT> тэги бекор қилинган. Бироқ, энг оддий ҳужжатлар учун физик форматлашни қўлласа бўлади. Шундай қилиб <FONT> тэги матн даражасидаги тэгларга мансуб ҳисобланиб ўзида блокларни, масалан, <P> <TABLE>ларни мужассамлаштира олмайди. <FONT> тэги учун қуйидаги атрибутлар берилиши мумкин:
FACE
SIZE
COLOR

FACE атрибутлари шрифт турини кўрсатишга ҳизмат қилади. Атрибут қийматини фойдаланувчида бўлган шрифт номи билан аниқ мос тушиши лозим бўлган шрифт номидан иборат бўлади. Агар бундай шрифт бўлмаса, узгартирилмаган ҳолда ўрнатилган шрифтдан фойдаланилади. Бир неча шрифт номларини вергул - “,” билан ажртилган ҳолда келтириш мумкин. Бундай рўйхат чапдан ўнгга қараб кўриб чикилади. Агарда компьютерда жорий шрифт бўлмаса, то мавжуд шрифт топилгунча берилган рўйхатдан кейинги шрифтлар кўриб борилади.
Мисол,

```
<HTML>
<HEAD>
<TITLE> шрифтларни белгилаш </TITLE>
</HEAD>
<BODY>
<I> бу узгартирилмаган шрифтда ёзилган матн </I> <BR>
<FONT FACE = “Verdana”, “Arial”, “Helvetica”>
<B> бу эса шрифтни кўрсатишга мисол </B>
</FONT>
</BODY>
</HTML>
```

SIZE атрибути шрифт ўлчамини 1 дан 7 гача бўлган шрифтли бирликларда кўрсатишга ҳизмат қилади. Шритфнинг аниқ ўлчами фойдаланилаётган браузер турига боғлиқ. Ўртача шрифт ўлчами 3 бирликни ташкил этади деб қабул қилади. Шрифт ўлчами ҳам абсолют катталик (SIZE=2), ҳам нисбий катта (SIZE=+1) тарзида берилиши мумкин.
Мисол:

```
<HTML>
<HEAD>
 <TITLE> шрифт ўлчамини белгилаш </TITLE>
</HEAD>
<BODY>
<FONT SIZE=1> шрифт ўлчами 1 </FONT> <BR>
<FONT SIZE=-1> шрифт ўлчами 2 </FONT> <BR>
<FONT SIZE=3> шрифт ўлчами 3 </FONT> <BR>
<FONT SIZE=4> шрифт ўлчами 4 </FONT> <BR>
<FONT SIZE=5> шрифт ўлчами 5 </FONT> <BR>
<FONT SIZE=+3> шрифт ўлчами 6 </FONT> <BR>
<FONT SIZE=7> шрифт ўлчами 7 </FONT> <BR>
</BODY>
</HTML>
```

COLOR атрибути шрифт ранггини ўзгартиради. Буни ранглар номи ёки #RRGGBB формати ёрдамида амалга ошириш мумкин.
Мисол:

```
<HTML>
<HEAD>
 <TITLE> Шрифт ўлчамини белгилаш </TITLE>
</HEAD>
<BODY>
<FONT>
<FONT COLOR=green>Яшил рангли матн </FONT> <BR>
<FONT COLOR=#FF0000> Қизил рангли матн </FONT> <BR>
</BODY>
</HTML>
```

Масалан:

```
<HTML>
<HEAD>
 <TITLE> Шрифт, ўлчам ва ранггини белгилаш </TITLE>
</HEAD>
<BODY>
<FONT SIZE=+1 COLOR= YELLOW FACE = COURIER>
Матн рангги сариқ, ўлчами катта, шрифт – courier </FONT>
</BODY>
</HTML>
```

<BASEFONT> тэги – бу тэг ҳужжатда бирор ҳаракт амалга оширилганда ишлатиладиган шрифт ўлчами, тури ва ранггини беришга ҳизмат қилади. У бажарилганда бутун ҳужжат учун амал қилади. Бу буйруқлар берилганда керакли жойларда <FONT> тэги ишлатилади. Ҳужжатда бирор ҳаракат амалга оширилганда шрифтлар, атрибутларни аниқлаш, бир неча марта такроран амалга оширилиши мумкин, яъни <BASEFONT> тэги кўп марта ишлатилиши мумкин.

<BASEFONT> тэги учун параметрлар <FONT> тэги атрибутлари билан бир ҳилдир. <BASEFONT> тэги жуфт эмас, бундан ташқари бу тэгни <HEAD> ҳужжат сарлавҳасида ҳам ишлатиш мумкин.

Масалан:

```
<HTML>
<HEAD>
<TITLE> Шрифт ўлчамларини аниқлаш</TITLE>
</HEAD>
<BODY>
    Бу матн ўзгартирилмаган шрифтда ёзилган
<BASEFONT SIZE=2> Шрифт ўлчами 2
<BASEFONT SIZE=4> Шрифт ўлчами 4
</BODY>
</HTML>
```

Хулоса қилиб шуни айтиш мумкинки, жадваллар учун <BASEFONT> тэгини белгилаш ҳеч қандай натижа бермайди.

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:

ALIGN

Горизонтал 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>
<HEARD>
<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:

ALIGN

Тэглар 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.

Назорат саволлари:

1. HTML тилида BODY контейнери;
2. HTML тилининг асосий тегларини санаб беринг;
3. Форманинг HIDDEN киритиш элементи;
4. Форманинг TEXTAREA киритиш элементи;
5. HTML тилида рўйхатлар.

3-Маъруза

Мавзу: HTML ДА ФОРМАЛАР ВА ФРЕЙМЛАР

Маъруза режаси:

1. HTML-формалар
2. HTML-фреймлар
3. Фреймлар орасидаги ўзаро таъсир

Таянч иборалар: *фрейм, type, form, reset, submit, value, input, button, frameset, mainframe, leftframe, rightframe, topframe, buttomframe, calls, rows*

1. HTML-формалар

Формалар WWW да фойдаланувчи томонидан киритилаётган маълумотларни тартибга солиш мақсадида қўлланилган. Форма элементлари тўлдирилиб бўлгач улардаги маълумотлар сервердаги маълумотларни қайта ишловчи дастурга юборилади. Кўп сонли жўнатилаётган маълумотлар жунатиш тугмаси босилгандан сўнг серверда жойлашган Common Gateway Interface (CGI) ёрдамида 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*> тэглари 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 дастерга юборилиш URI (Uniform Resource Identifier) адресини ифодалайди.

<FORM METHOD=post
ACTION=mailto:yourname@your.email.address>

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.

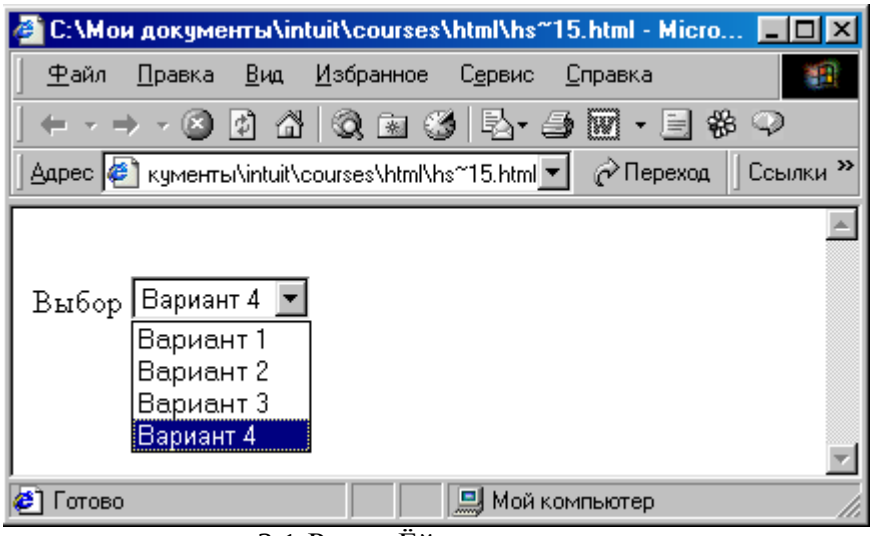
NAME атрибути
Объект номини ифодалайди.

SIZE атрибути
Рўйхатни кўринувчи сатрлари сонини ифодалайди. `SIZE > 1` бўлган ҳолда браузер оддий рўйхатни кўрсатади.
Формада <*OPTION*> теги фақат <*SELECT*> тэглари орасида қўлланилади. Бу тэглар қўшимча иккита атрибутни қўллаб қувватлайди: `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>



3.1-Расм. Ёйилувчи меню

2. HTML да Фреймлар

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 туғулганда 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*”>.
Ушбу мисолда қийматларни белгилашнинг 3 та усулидан ҳам фойдаланилган. Ҳам ROWS, ҳам COLS атрибутлари биргаликда ишлатилганда фреймлар сеткаси яратилади:
<FRAMESET COLS = “2*, *”, ROWS = “*, 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.
Мисол:

```
<HTML>
<HEAD>   </HEAD>
<FRAMESET ROWS = “25%, 50%, 25%”>
<FRAME SRC=“header.html”>
<FRAME SRC=“footer.html”>
<FRAME SRC=“content.html”>
</FRAMESET>
<NOFRAMES>
    Сизнинг браузерингиз фреймларни акс эттирмайди
</NOFRAMES>
</HTML>
```

Фреймларнинг мунтазам сеткасини яратишга мисол:

```
<HTML>
<HEAD>   </HEAD>
<FRAMESET ROWS =“*, 2*, COLS=20%, 30%, 50%”>
<FRAME SRC=“docum1.html” scrolling=“yes”>
<FRAME SRC=“ docum2.html” scrolling=“yes”>
<FRAME SRC=“ docum3.html” scrolling=“yes”>
<FRAME SRC=“ docum4.html” scrolling=“yes”>
<FRAME SRC=“ docum5.html” scrolling=“yes”>
</FRAMESET>
</HTML>
```

3. 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.

Мисол:

```
<HTML>
<HEAD>
<TITLE> Фреймлардан фойдаланиш </TITLE>
</HEAD>
<FRAMESET COLS = “2*, *, *”>
<FRAME SRC=frame_a.html NAME=“A”>
<FRAME SRC=empty.html NAME=“B”>
<FRAME SRC=empty.html NAME=“C”>
</FRAMESET>
</HTML>
```

frame_a.html деб номланган файл қуйидаги кўринишга эга:

```
<HTML>
<HEAD>
<TITLE> Фрейм учун ҳужжат </TITLE>
</HEAD>
<BODY>
<A HREF=“test.html” TARGET=“A”>
1. B </A> <p> фреймига ҳужжатни юклаш

<A HREF=“test.html” TARGET=“B”>
2. C </A> <p> фреймига ҳужжатни юклаш

<A HREF=“test.html” TARGET=“C”>
3. D </A> <p> ойнасига ҳужжатни юклаш

<A HREF=“test.html” TARGET=“_blank”>
4. Янги </A> <p> ойнасига ҳужжатни юклаш

<A HREF=“test.html” TARGET=“_top”>
5. </A> <p> бутун ойнасига файлни юклаш

<A HREF=“test.html” TARGET=“_self”>
6. Жорий </A> фреймига ҳужжатни юклаш
</BODY>
</HTML>
```

test.html файлининг матни:

```
<HTML>
<HEAD>
<TITLE> Матнли ҳужжат </TITLE>
</HEAD>
<BODY>
```

ҳужжатнинг матни
</BODY>
</HTML>

empty.html файлининг матни:

<HTML>
<HEAD>
<TITLE> бўш файл </TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>

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

Назорат саволлари:

1. Фрейм тушунчаси;
2. HTML тилида фреймлар қандай яратилади?
3. HTML тилида формалар қандай яратилади?
4. Форманинг BUTTON киритиш элементи;
5. HTML тилида актив ва оддий тасвирлар билан ишлаш.

4-Маъруза

Мавзу: КЛИЕНТ ТОМОНИДАН ДАСТУРЛАШ. JAVASCRIPT ГА КИРИШ. JAVASCRIPT НИ HTML-ХУЖЖАТГА ЖОЙЛАШТИРИШ.

Маъруза режаси:

1. JavaScript га кириш
2. JavaScript нинг объектли модели тушунчаси
3. JavaScript ни HTML-хужжатига жойлаштириш.
4. JavaScript нинг URL-схемаси
5. Синфлар иерархияси

Таянч иборалар: *Патрик Нотон, Oak, WebRunner, Sun Microsystems, Liveware – Netscape, ECMA, Java, C++, HTML-контейнер, JavaScript, Подстановкалар, скриптли дастурлар, URL-схемаси, синфлар иерархияси.*

1. JavaScript га кириш

1990 йили Sun Microsystems компанияси дастурлар яратиш ходими Патрик Нотон (Patrick Naughton) ташаббуси билан Green, дастурлаш гурухи ташкил этилди . Бу гурухнинг мақсади хар хил турмушда ишлатилувчи аппаратлар: видеомагнитофонлар, лазер диски билан ишловчи қурилмалар стериосистемалар билан ишлаш да ягона ёндошувни амалга ошириш дастурлаш тили ишлаб чиқилишга киришилди. Шу вактда Sun Microsystems компанияси Mosaic системасини ишлаб чиқди бу тизим эса World Wide Web ларнинг асосини ташкил этар эди ва у Internet нинг тез ривожланишига сабаб бўлган эди . Тезда компания Oak-компилятор ва Oak-браузер «WebRunner» яратилди. 1995 йили Sun Microsystems компанияси бу махсулотни такомиллаштириб уни Java деб номлади.

Web саҳифани генерация қилиш жараёнида "клиент-сервер" архитектураси билан боғлиқ равишда дилеммалар ҳосил бўлади. Саҳифалар клиент томонида ҳам сервер томонидаги каби генерация қилинади. 1995 йилда Netscape компанияси мутахассислари клиент

томонидаги саҳифаларни генерация қилиш учун махсус дастурлаш тили яратишди ва уни ***JavaScript*** деб номдашди.

Шундай қилиб, ***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*" номи Netscape нинг ўзигагина тегишлидир. Шунга ўхшаш Microsoft томонидан ишлаб чиқилган тилнинг расмий номи Jscript деб аталади. *JavaScript* нинг бир қатор версиялари JScript нинг версиялари билан биргаликда ишлайди (қисман аммо тўлиқ эмас), яъни *JavaScript* тили JScript тилининг қисм тўплами ҳисобланади.

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. Таъкидлаш керакки, бугунги кунда *JavaScript* нинг барча турлари ҳам ECMA стандартига мос тушавермайди.

Java нинг асосий характеристикалари

1. Соддалиги. Тилнинг соддалиги Java нинг хама характеристикаларига кириши лозим эди : масалан, дастурловчи тилни ўрганишга кўп вақт ажратмаслиги лозим. Тилнинг фундаментал концепциялари дастурловчилар томонидан жуда тез ўрганилиб олинди . Java ни ишлаб чиқувчилар дастурловчиларнинг кўпчилиги C++ билан яхши танишлигини эътиборга олишган. Шу сабабли Java C++ га жуда яқин дастурлаш тилидир .
2. Java хавфсисзликга жуда катта ахамият берилган . Кўпгина шифрлаш алгоритмлари шу тил функцияларини ишлатишади.
3. Барқарорлиги. Java платформаси ўта барқарор ишловчи амалий дастурлар яратишни таъминлашга мўлжалланган. Java да объектлар яратиш new оператори билан амалга оширилади Java да массивлар билан ишлаш бевосита бошқариш системаси назоратида олиб борилади. Хотирани бошқариш модели C++ ва C даги кўпгина хатоларнинг олдини олади.
4. Объектга йўналтирилганлиги. Java да дастурловчилар объектларнинг стандарт библиотекасидан фойдаланишлари мумкин, масалан, киритиш-чиқариш қурилмаларидан фойдаланиш, тармоқ функцияларидан фойдаланиш , график интерфейс яратиш усулларидан фойдаланиш ва ҳ.к. Бу объектлар библиотекаси функционаллиги янада кенгайтирилиши мумкин.
5. Кўп оқимли. Хозирги вактда кўпгина тармоқ иловаларида бир вақтда бир нечта операцияларни биргаликда бажаришни тақазо этади . Java да бундай жараёнларни ёзиш механизми ишлаб чиқилган.
6. Юқори самарадорлик. Java да юқори самарадорлик махсус оптимизация қилинган байт код ишлатилиши билан эришилади, бу кодлар машина кодига жуда осон ўтказилади. Javada иловалар шундай яратилиши мумкинки унинг бир қисми ассемблер илида ёзилиши мумкин.
7. Java нейтрал архитектурага мўлжалланган.
8. Java динамиклиги.

Java нинг такомиллашуви натижасида JavaScript тили вужудга келди.

JavaScript – бу HTML вараги тарибида ишлатилувчи дастурлаш тилидир ,у асосан фойдаланувчи билан мулоқатни Microsoft Internet Explorer ёки Netscape Navigator дастурлари орқали амалга оширади. У Netscape ва Sun Microsystems компанияси билан биргаликда ишлаб чиқилган. JavaScript ёрдамида Web варақларида HTML стандарт теглари ёрдамида қилиш мумкин бўлмаган ишларни бажариш мумкин. Скриптлар эса маълум бир ходисалар бажарилишида рўй беради . JavaScript Java билан тугридан тугри богланишга эга эмас аммо шундай бўлсада Java нинг ташқи хусусиятларига ва усулларидан фойдаланади. Фарқи шундаки Java апплетлари браузер ташқарисида ишлайди , JavaScript да еса браузерлар ичида ишлайди. JavaScript – бу мижозларнинг фаол варақларини яратиш тилидир, унинг ёрдамида HTML хужжатларининг мазмунини ўзгартириш мумкин ,анимацияларни бошқариш , фойдаланувчи томонидан киритилган формаларни текшириш учун хам ишлатилиши мумкин ,мураккаб математик хисоблашларни амалга ошириш мумкин ,шунингдек Web-узелларда излашлар амалга оширишлиши мумкин. Шу сабабли JavaScript тўлигича кўпгина Web мастерлари талабига жавоб бераолади.

2. 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. О*бъектларнинг* 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 оқимини очади, write() усули ушбу ёзишни амалга оширади, close() усули хужжатга ёзиш оқимини 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. Например, button типидаги *объект* билан (INPUT контейнери билан button - "Тугма") click ҳолати амалга ошиши мумкин, яъни фойдаланувчи тугмани босиши мумкин. Бунинг учун INPUT контейнери атрибути click ҳолатни - onClick ҳолатига кенгайтирган. Ушбу атрибут қиймати сифатида HTML хужжат муаллифи томонидан *JavaScipt* да тузилган ҳолатни қайта ишловчи дастур кўрсатилади:

```
<INPUT TYPE=button VALUE="Нажать" onClick="window.alert('Марҳамат, яна бир бор босинг');">
```

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 биттасининг турли версиялари) бир биридан фарқланади, лекин мантиқий таркиби бир ҳилда бўлади.

Кодни HTML-саҳифага жойлаштириш

Дастурлашни энди бошловчилар учун доимо бир савол мавжуд бўлади: "Дастурни қандай жиҳозлаш ва уни қандай бажариш?". Ушбу саволга жавоб бериб кўрамиз, аммо бунда *JavaScript*-коднинг барча хусусият ва катталиклари ҳақида унутмаслик керак.

Биринчидан, *JavaScript*-кодни браузерда бажариш. Браузерда *JavaScript* интерпретатор ўрнатилган. Умуман олганда *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);
- вставка (SCRIPT контейнери).

JavaScript бўйича қўлланмаларда *JavaScipt* ни қўлланилиши SCRIPT контейнери орқали берилади. Аммо дастурлаш нуқтаиназари бўйича бу тўла тўғри эмас, ушбу тартиб қуйидаги саволга жавоб бермайди: *JavaScript*-код бошқарувни қандай қўлга киритади? Яъни *JavaScript* да ёзилган ва HTML ҳужжатга жойлаштирилган дастур қандай бажарилади.

HTML хужжат муаллифининг малакаси ва билим даражасига қараб *JavaScript* нинг бир нечта усулларидан фойдаланиши мумкин.

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.

3. JavaScript нинг HTML да ишлатилиши

JavaScript да ёзилган скриптларни ишга тушуриш учун бизга браузер керак бўлади, масалан, Netscape Navigator ёки Microsoft Internet Explorer (MSIE). Бу иккала браузер хам JavaScript да ёзилган скриптларни ишлаш имконини яратади. JavaScript бизнинг Web сахифамиз кўринишни яхшилашда ишлатилади. Бу тилни ўрганишга киришишдан олдин умуман олганда биз HTML тилини билишимиз яхши бўлади. JavaScript да ёзилган дастур бевосита HTML хужжатларга жойлаштирилади. Бунинг учун махсус тэг <SCRIPT> ва </SCRIPT> ишлатилади.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
...
JavaScript даги дастур
...
//-->
</SCRIPT>
```

LANGUAGE атрибути дастур қайси тилда ёзилганлигини англатади , бизнинг холда JavaScript да ёзилганлигини билдиради . Скриптли дастурларни қўлламайдиган барузерлар уларни ўтказиб юборишлари учун дастурлар комментария блоклари ичига жойлаштирилади.

Одатда дастур функциялари HTML хужжатнинг <HEAD> секциясига жойлаштирилади. Бу HTML хужжат ичида қандай ёзилишини курайлик:

```
<HTML>
<HEAD>
<TITLE> JavaScript даги дастур</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
...
JavaScript даги дастур
...
//-->
</SCRIPT>
</HEAD>
<BODY>
...
HTML-хужжат матни ва JavaScript га функцияларни чақириш
...
</BODY>
</HTML>
```

JavaScript тилида икки хил комментариялар ишлатилади.Биринчисига бир қаторли комментариялари киради , комментария "//" символи билан ажратилади:

```
// Бу - комментария қатори;
ёки
askUser();
//берилмаларни фодаланувчидан олиш.
```

Иккичисига кўп қаторли комментариялар киради:

```
/*
```
Бу –кўп каторли комментариялар бўлиб уларни **JavaScript** интерпретатори эътиборга олмайди

HTML-хужжатга кодни жойлаштиришда 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 хужжат 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();">
```

4. 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
Ушбу ҳолда URL http – билан бошланади ва рухсат схемасини (http схема).

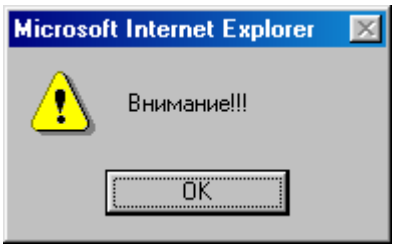
Гипематнли тизимли дастурлаш тилининг асосий вазифаси гипематнли ўтишларни дастурлашдир. Бу шуни англатадики, у ёки бу гипематнли ссилканинг босилиши гипематнли ўтишни амалга оширувчи дастурни ишга тушуради. Web-технологиада шунга ўхшаш стандарт дастурлар саҳифани юклаш дастурлари ҳисобланади. *JavaScript* 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">
```

Ушбу ҳолда "JavaScript_код" матни *JavaScript* даги гипермурожаат босилганда қайта ишлаш дастури ҳисобланади ва кейинги ҳолда расимни юклаш чоғида қўлланилиши келтирилган.

Масалан, Внимание!!! номли гипематнли ссилка босилганда огоҳлантириш ойнасининг очилиши қуйидагича амалга оширилади: (очиш)

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



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

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

URL да мураккаб дастурларни жойлаштириш ва функцияларни чақириш мумкин. Шуни унутмаслик керакки *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('intuit');"></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 l()
{
  str = window.location.href;
  return(str.length);
}
</SCRIPT>
<FORM><INPUT VALUE="&{window.location.href};" SIZE="&{l()};">
</FORM>
<SCRIPT>
<!—Бу изоҳ ...JavaScript-код...// -->
</SCRIPT>
<BODY>
... Ҳужжат танаси ...
</BODY>
</HTML>
```

Биламизки, ҳужжатнинг сарлавҳа қисмидаги матнлар браузер ойнасида кўринмайди. Шунинг учун бу қисимга 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.

Функцияларни жойлаштириш ва фойдаланишга мисол кўрамиз:

```
<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>
```

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

5. 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 турган 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().

Назорат саволлари:

1. Java Script тилида операторлар;
2. Java Script тилида ўзгарувчилар типлари;
3. Java Script тили келиб чиқиш тарихи;
4. Java Script нинг HTML да қўлланилиши.

5-Маъруза

Мавзу: JAVASCRIPT ДА ЎЗГАРУВЧИЛАР, МАЪЛУМОТ ТИПЛАРИ, ИФОДАЛАР ВА АРИФМЕТИК АМАЛЛАР. JAVASCRIPT ДА ФУНКЦИЯ ТУШУНЧАСИ

Маъруза режаси:

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

Таянч иборалар: *ўзгарувчилар, var, маълумотлар типи, сон, матнли қатор, мантиқий маълумот, оператор, унар оператор, бинар оператор, силжувчи оператор, мантиқий оператор, ўзлаштириш оператори, функция, function.*

1. 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, қуйидаги оддий қоидаларга амал қилиш 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     const    continue
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.

2. 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 ёки 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:

```
<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" + "<BR>";
```

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"+ "<BR>";
```

Натижада MyBuf2 ўзгарувчиси MyTextBuf.ўзгарувчиси каби қаторга эга бўлади.

3. JavaScript тили операторлари

Унар операторлари

Унар операторлари 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 узатилмоқда
```

Бинар операторлари

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 кщрилган:
```
>>    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 С 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;
```

4. 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 номи([1-параметр] [,2-параметр] [...,N-параметр])
{  . . .
  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.

6-Маъруза

Мавзу: ДАСТУР МАНТИҚИНИ БОШҚАРИШ ЭЛЕМЕНТЛАРИ ВА ИФОДАЛАРИ

Маъруза режаси:

1. For элементи

2. Break элементи
3. Continue элементи
4. For..in элементи
5. If..else элементи
6. Function элементи
7. New элементи
8. Return ифодаси
9. Var ифодаси
10. This ифодаси
11. While ифодаси
12. With ифодаси

Таянч иборалар: *For, Break, Continue, For..in, Function, If..else элементи, New элементи, Return ифодаси,Var ифодаси, This ифодаси, While ифодаси , With ифодаси.*

1. For элементи

For элементи коднинг бир неча марта такрорланувчи булокларини ташкил этиш учун ишлатилади . Такрорланишлар сони махсус ўзгарувчи қиймати орқали назорат қилинади . Синтаксиси:

```
for( StartVal; Condition; Inc)
```

Бу ерда

- StartVal – цикл ўзгарувчисининг бошлангич қиймати. Одатда у 0 ёки 1 га тенг бўлади , аммо исталган бошқа қийматларга эга бўлиши мумкин. Масалан, Count = 0 ёки i = 1;
- Condition – цикллар сонини назорат қилувчи ифода. Бу ифода қиймати чин бўлса цикл бажарилаверади. Масалан, i<=9 чин қийматга эга бўлса цикл бажарилади;
- Inc – қиймат цикл узгарувчисининг қиймати нечига ўзгаришини аниқлайди . Масалан, i++

  I нинг қийматини бирга ўзгартиради .

1-Мисол:

```
for( i = 0; i <= 9; i++)
{
  document.write( "i = " + i + "<br>" );
}
```

2-мисол :

```
for( i = 0; i < 51; i+=5)
{
  document.write( "i = " + i + "<br>" );
}
```

2. Break элементи

Break элементи бу блокдаги кодларнинг бажарилиши тўхтатилишини ва бу блокдан кейинги биринчи оператордан бошлаб бажарилиши давом этишини аниқлайди. Бу элементи(break) for, for..in ва while элементлари ёрдамида яратилган блокларда ишлатилади. Кўпинча break элементи цикл тугалланишидан олдин чиқиб кетишда ишлатилади. Масалан:

```
for( i=0; i<=19; i++)
{
  if( i == 10 ) break;
  document.write( "i = " + i + "<br>");
}
```

Бу дастурда цикл i=19 гача бажарилиши лозим , аммо break элементи i=10 бўлгандаёқ циклдан чиқишни амалга оширади .

3. Continue элементи

Бу элемент(continue) for, for..in ва while элементлари ёрдамида яратилган блокларда , бу элементдан кейин келаётган ифода бажарилмасдан ўтказиб юборилиши учун ишлатилади. Масалан:

```
for( i=0; i<=19; i++)
{
  if( i == 10 ) continue;
  document.write( "i = " + i + "<br>");
}
```

Цикл 0 дан токи 19 гача ишлайди . Цикл ичида if элементи i=10 лигини текширади , ва агар i=10 бўлса счетчик қиймати чоп қилшга чиқарилмайди . Шундай қилиб экранга қуйидаги рақамлар чиқарилади: 0 дан 9 гача бўлган рақамлар ва 11 дан токи 19 гача бўлган рақамлар .

4. For..in элементи

for..in элементи for элементининг махсус формаси бўлиб объект хусусияти номи ва қийматини олиш учун ишлатилади. Синтаксиси:

```
for( var in object )
{
  ...
}
```

- var – ўзгарувчи номи, у объект хусусияти ва уларнинг қийматларига кириш индекси сифатида ишлатилади;
- object - объект, шу объектга кириш амалга оширилади.

1-Мисол

Масалан, бизга Navigator объектнинг жорий қийматини билиш лозим бўлса , бу кодни шундай ёзиш мумкин:

```
for( i in navigator )
{
  document.write( i + " = " + navigator[i] + "<br>" );
}
```

Бу дастурни Netscape Communicator 4.01 ишлашини амалга оширувчи кодни қуйидагича ёзиш мумкин.

```
userAgent = Mozilla/4.01 [en] (Win95; I)
appCodeName = Mozilla
appVersion = 4.01 [en] (Win95; I)
appName = .Netscape
language = en
platform = Win32
plugins = [object PluginArray]
mimeTypes = [object MimeTypeArray]
```

2-мисол.

Худди шунингдек Screen (объект Communicator 4.x браузерларида ишлатилади) объектнинг хусусиятининг жорий қийматини олиш учун қуйидаги кодни ёзамиз :

```
for( i in screen )
{
  document.write(i + " = " + screen[1] + "<br>");
}
```

Бу дастурнинг ишлаш мисоли :

```
width = 640
height = 480
pixelDepth = 16
colorDepth = 16
availWidth = 640
availHeight = 480
availLeft = 0
availTop = 0
```

3-мисол.

Қуйидаги мисолда for..in элементидан фойдаланиб форма майдонлари қийматларини олиш дастури келтирилган :

```
<script language="JavaScript">
 function foreTest()
 {
  for( i in document.demoform )
  {
   alert( i + " " + document.demoform[i] );
  }
  document.close;
 }
</script>
<FORN NAME="demoform">
<INPUT TYPE="text" NAME="txt1"><br>
<INPUT TYPE="text" NAME="txt2"><br>
<INPUT TYPE="tsxt" NAME="txt3"><br>
<INPUT TYPE="button" VALUE="Test" onClick="formTest()">
</FORM>
```

5. If..else элементи

if..else элементи бирон бир шартга боглиқ бўлган ифодаларни яратиш учун ишлатилади. Бу элементнинг ишлаш алгоритми қуйидагича ёзилади :

- агар ифоданинг қиймати true бўлса , if блокидаги инструкциялар бажарилади ;
- агар ифоданинг қиймати false бўлса , else блокидаги инструкциялар бажарилади . Агар else блоки қатнашмаса , if..else блокидан кейинги элементга бошқарилиши узатилади .

if..else элементи синтаксиси:

```
if( expression )
```

Кўпинча expression ифодасининг қиймати одатда true ёки false га тенг бўлади . Агар if ва else блокларида фақат битта ифода қатнашса , у холда қуйидаги синтаксисдан хам фойдаланиш мумкин :

```
if( Check() )
  Send();
else
  Clear();
```

Агар блокда биттадан ошиқ ифодалар ишлатилаётган бўлса , у холда if ва else блоклари ни ажратиш учун {} қавслари ишлатилади:

```
if( Check() )
{
  alert('Формани жўнатиш');
  Send();
}
```

```
 else
 {
  alert('Формани тозалаш');
  Clear();
 }
```

Шуни қайд қилиш лозимки натижалари текширилаётган ифодаларда исталган огператорлар иштирок этиши мумкин:

```
 function Check()
 {
  var doc = window.document;
  if( doc.forms[0].elements[0].value == '' )
  {
   alert('Майдонлар бўш бўлишлари мумкин эмас');
   return false;
  }
  else
   return true;
 }
```

Юқорида келтирилган мисолда форманинг биринчи майдони қиймати бўш қатор билан таққосланади .

6. Function элементи

Function элементи хусусий функцияларни ва объектларини яратиш учун ишлатилади. Функциялар JavaScript кодидан чақирилиши мумкин бўлган қисм дастурлардан ташкил этилади. Масалан, сервер - Check га жўнатишдан олдин форма майдонларининг тўгри тулдирилишини текширувчи функция қуйидагича яратилади.

```
<script language="JavaScript">
 // Формани серверга жўнатувчи Функция
 function SendForm()
 {
  if( Check() ) window.document.forms[0].submit;
 }
 // Форма қийматларини текширувчи функция
 function Check()
 {
  var doc = window.document;
  if( doc.forms[0].elements[0].value == '' ||
      doc.forms[0].elements[1].value == '' ||
      doc.forms[0].elements[2].value == '' )
  {
   alert( 'Майдонлар бўш бўлиши мумкин эмас' );
   return false;
  }
  else
   return true;
 }
</script>
```

Бу ерда Check функцияси форманинг биринчи учта майдонини текширмоқда, агар улардан бири бўш бўлса у хақида хабар чиқаради.

SendForm (Check текшириш функцияси, шу функцияга чақирилади) функцияси форма билан богланади :

```
<form name="rqform" action="request.dll/doform" method="post">
<font size=4 color="#000090">
 <b>Ахборотни киритинг:</b>
 <br><br>
</font>
<table bgcolor="#fff8bd" border=0 cellspacing=0 cellpadding=1>
 <tr>
  <td><b> Фамилия</b></td>
  <td><input type="text" size=35 value="" name="famname">
 </tr>
 <tr>
  <td><b> Исми</b></td>
  <td><input type="text" size=35 value="" name="mainname">
 </tr>
 <tr>
  <td><b> Шарифи</b></td>
  <td><input type="text" size=35 value="" name="fathname">
 </tr>
 <tr>
  <td><b> Сиз билан қандай богланиш мумкин</b>  </td>
  <td><TEXTAREA name="contact" rows="5" cols="30"></TEXTAREA></td>
 </tr>
 <tr>
  <td width="50%" align="left">
   <input type="Submit" value="Жунатиш" onClick="SendForm()"></td>
  <td width=503 align right>
   <input type="Reset" value="Бекор қилиш" onClick="history.back(-1)"></td>
 </tr>
</table>
</form>
```

7. New элементи

Бу элемент (new) объектнинг янги нусхасини яратади. У икки усул билан фойдаланилиши мумкин:

- Date янги объектини яратиш учун ;
- Янги фойдаланувчи объектини яратиш учун.

Синтаксиси:

```
varName = new objectName(params);
```

- varName – ўзгарувчи номи, шу ўзгарувчида объектнинг янги нусхаси сақланади;
- objectName – объект номи. Date ички жойлаштирилган Date сўзи ишлатилади (Катта харф билан бошланади) ;
- params – битта ёки ундан ортиқ параметрлар, янги объект яратилишида ишлатиладиган параметрлар.

Қуйидаги мисолда new элементининг ички объектнинг янги нусхасини яратишда ишлатилишини кўрамиз :

```
now = new Date();
```

Бу амал бажарилгандан кейин now ўзгарувчиси Date объекти нусхасини сақлайди , ва ундан объект хусусиятини ўзгартириш учун ва унинг усулларини чақириш учун ишлатиш мумкин. Масалан , жорий вақтни билиш учун , биз getHours() усулини чақирамиз :

```
now = new Date();
nowHour = now.getHours();
```

Қуйидаги мисолда new элементининг фойдаланувчи объектининг янги нусхасини(экземплярини) яратишда ишлатишини кўрамиз . Бу объект унга берилган қаторга name хусусияти қийматини беради . Сўнгра бу хусусият қиймати хабарлар панелида ифодаланади :

```
user = new someUser("Alex Fedorov");
alert( user.name );
function someUser(nameParam)
{
  this.name = nameParam;
  return (this);
}
```

8. Return ифодаси

Одатда return ифодаси функция охирини аниқлайди . Яъни функцияга мурожат қилинган қатордан кейинги қатор бажарилади . return ифодасидан бирон бир қийматни қайтариш учун хам фойдаланиш мумкин . Юқоридаги мисолга қайтайлик:

```
function Check()
{
  var doc = window.document;
  if( doc.forms[0].elements[0].value == '' )
  {
    alert('Майдонлар бўш бўлишлари мумкин эмас');
    return false;
  }
  else
    return true;
}
```

Бу ерда Check функцияси false қийматини қайтаради , агарда if блокидаги ифода true қийматга эга бўлса ва акс холда true қийматни қайтаради.

9. This ифодаси

This – бу аслида ифода эмас, балки калит сўздир. У жорий объектга мурожат қилиш учун ишлатилади У икки хил варианта ишлатилиши мумкин:

- жорий формага мурожат қилиш учун ёки ходисаларни ишловчи интерфейс элементида (масалан, onClick ёки onSubmit);
- фойдаланувчи объектининг янги хусусиятини бериш учун .

Одатда калит сўз this ни объект номи билан биргаликда , ёки аълохида хам ишлатиш мумкин, яъни ёки this, ёки this.object шаклида ишлатилиши мумкин. Мисол. Кнопкага босишни ишловчи ишлаганда бутун форма кайтарилиши талаб қилинса, бу дастур қуйидагича ёзилади:

```
<INPUT TYPE="button" NAME="test" VALUE="Test"
  onClick="test(this.form)">
```

Агар оддий калит сўз this ишлатилса , у холда бутун форма эмас фақат кнопка қайтарилади :

```
<INPUT TYPE="button" NAME="test" VALUE="Test"
  onClick="test(this)">
```

Қуйидаги мисолда хар хил вариантларда калит сўз this нинг ишлатилиши келтирилади:

```
<html>
<head><title>JS - CP1197</title>
<script language="JavaScript">
 function myTest(obj)
 {
  alert(obj.name);
  return;
 }
</script>
</head>
<body>
<p align="center">
Калит сўзнинг ишлатилиши  <b>this</b>
</p>
<center>
<form name="ThisTest">
<INPUT TYPE="button" NAME="test1" VALUE="Test1"
 onClick="myTest(this.form)">
<INPUT TYPE="button" NAME="test2" VALUE="Test2"
 onClick="myTest(this)">
</form>
</center>
</body>
</html>
```

10.Var ифодаси

Бу ифода (var) ўзгарувчиларни бериш учун ишлатилади. Бу холда ўзгарувчи қиймати хам берилиши мумкин . Бу ифода хам ўзгарувчиларнинг харакат сохасини беради , агарда функция ичида тавсифланаётган бўлса .

Синтаксиси:

```
var VariableName;
```

ёки

```
var VariableName = value;
```

- VariableName – ўзгарувчи номи;
- value – ўзгарувчига бериладиган қиймат.

Биринчи холда ўзгарувчи VariableName эълон қилинмокда , аммо унинг қиймати берилмаган. Иккинчи холда ўзгарувчи қиймати унинг эълон қилинишида берилмокда . Фойдаланувчи функциялари ташқарисида қуйидаги икки ифода тенг кучли хисобланади:

```
var VariableName = value;  VariableName = value;
```

Иккала холда хам хамма функциялар ишлатилиши мумкин бўлган глобал ўзгарувчи эълон қилинмокда . Масалан , ўзгарувчи someVar глобал хисобланади:

```
var someVar = 100;
function showVar()
{   alert( someVar );  }
```

Функция ичида фақат шу функция учун ишлатилиши мумкин бўлган локал ўзгарувчини хам бериш мумкин:

```
var someVar = 100;
function showLocalVar()
{
  var someVar = 256;
  alert( 'local var = ' + someVar );
```

```
  showGlobalVar();
 }
 function showGlobalVar()
 {
  alert( 'global var = ' + someVar );
 }
```

11.While ифодаси

While ифодаси цикларни яратишда ишлатилади. Бу цикллар бу ифоданинг қиймати true бўлганда бажарилади. Цикл танаси фигурали қавсга олинади .

Синтаксиси:

```
 while( exprssion )
 {
  // цикл танаси
 }
```

Қуйидаги циклда цикл танаси 10 марта бажарилади. Хар бир итерацияда счетчик қиймати чиқарилади:

```
 loopCount = 0;
 while( loopCount < 10 )
 {
  document.write( "LoopCount = " + loopCount + "<BR>");
  loopCount++;
 }
```

12.With ифодаси

With ифодаси объектлардан оддий ва кўринарли фойдаланишни таъминлайди . Масалан, ички Math объекти усулларига кириш учун қуйидаги кодни ёзамиз:

```
 alert( Math.PI );
 alert( Math.round( 1234.5678 ) );
```

Синтаксиси:

```
 with( object )
 {
  ифода
 }
```

- object – фойдаланилаётган объект;
- ифода – объект ишлатаётган битта ва ундан ортиқ ифодалар.

Шундай қилиб , with ифодасидан фойдаланиб биз объектининг хусусиятлари ва усулларига хар бир мурожат Қилишимизда Math объектини кўрсатишимиз шарт эмас:

```
with( Math )
 {
  alert( PI );
  alert( round( 1234.5678 ) );
 }
```

With ифодаларида ички ва фойдаланувчи объектларидан фойдаланиш мумкин . Масалан , форманинг биринчи элементига мурожат қуйидагича ёзилади :

```
 function Show()
 {
  with( document.forms[0].elements[0] )
  {
```

```
    alert( name );
    alert( value );
  }
 }
```

Бу мисолда with ифодасида форманинг биринчи элементи кўрсатилган , шу сабабли with блокидаги alert функцияси document.forms[0].elements[0] объекти хусусиятига индамасдан мурожат қилмокда .

7-Маъруза

Мавзу: JAVASCRIPT НИНГ УЧ ТУРДАГИ ОБЪЕКТЛАРИ. JAVASCRIPT ОБЪЕКТИ ХУСУСИЯТЛАРИ ВА УСУЛЛАРИ

Маъруза режаси:

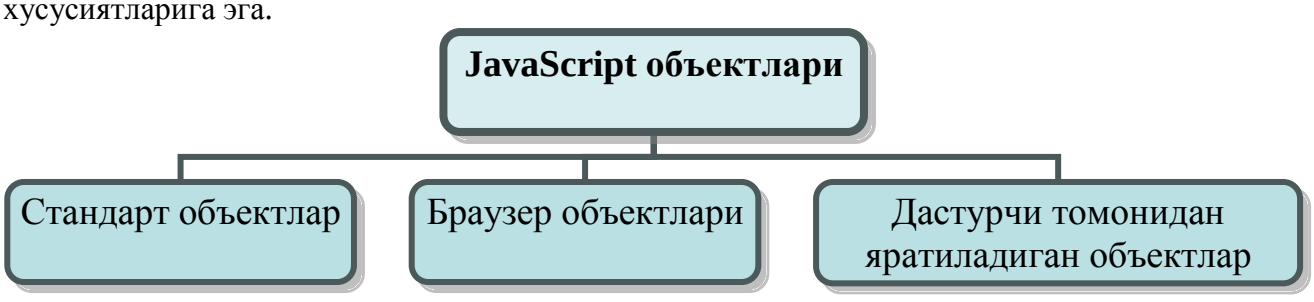
1. JavaScript нинг уч турдаги объектлари
   - Стандарт объектлар
   - Браузер объектлари
   - Дастурчи томонидан яратилувчи объектлар
2. JavaScript объекти хусусияти ва усуллари

Таянч иборалар: *стандарт объектлар, Array, Boolean, Math, Date, Function, Global, Number, Object, String, браузер объектлари, дастурчи томонидан бошқариладиган объектлар, объект, усул.*

JavaScript сценарийли тили объектга-мўлжалланган тилдир. JavaScript объектлари хусусиятлар ва усуллар тўпламини ифодалайди. Объект хусусияти – бу, объектга боғлиқ бўлган маълумотлардир, усуллар эса - объект маълумотларини қайта ишловчи функциялардир. JavaScript сценарийда хусусиятларни адреслаш уларнинг номлари билан ёки уларнинг номерлари билан амалга ошиши мумкин. Кейинги вариант бўйича, ҳар бир хусусият массивнинг бир элементи сифатида олинади ва улар ўзларининг уникал номерларига эга бўладилар.

1. JavaScript нинг уч турдаги объектлари

JavaScript тилида уч турдаги объектлар мавжуд: стандарт объектлар, браузер объектлари ва дастурчи томонидан яратилувчи объектлар. Уларнинг ҳар бири ўзларининг таснифи ва хусусиятларига эга.



Стандарт объектлар

Қуйида JavaScript да қўлланилувчи стандарт объектлар, хусусиятлар ва усуллар келтирилган. Уларни ишлатишда олдиндан эълон қилиш талаб этилмайди.

Объект	Таснифи
`Array`	Массив

Boolean	Мантиқий маълумотлар
Date	Календарли вақт
Function	Функция
Global	Глобал усуллар
Math	Математик константа ва функциялар
Number	Сон
Object	Объект
String	Сатр

Стандарт объектлар билан қандай ишлаш мумкин? Анча оддий. Объектни реализация қилувчи дастур ёзилади ва унинг хусусият ва усулларига мурожаат қилинади. Мисол сифатида жорий вақтни кўрсатувчи HTML хужжатни кўрамиз.

```
<HTML>
  <HEAD>
    <TITLE>Жорий кун ва вақт </TITLE>
  </HEAD>
  <BODY BGCOLOR=WHITE>
    <H1> Жорий кун ва вақт </H1>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

      var dt;
      var MyDate="";

      dt = new Date();
      MyDate = "Date: " + dt.getDate() + "."
        + dt.getMonth() + "." + dt.getYear();

      document.write(MyDate);
      document.write("<BR>");
      document.write("Time: " + dt.getHours()
        + ":" + dt.getMinutes() + ":" + dt.getSeconds());

    // -->
    </SCRIPT>
  </BODY>
</HTML>
```

Бу ерда JavaScript сценарий new калит сўзи ёрдамида Data объектини яратади. Бунда Date конструктори параметрларсиз келтирилади:

```
var dt;
dt = new Date();
MyDate = "Date: " + dt.getDate() + "."
  + dt.getMonth() + "." + dt.getYear();
```

getDate, getMonth ва getYear усуллар ёрдамида жорий сана олинади. Ушбу усуллар dt объекти учун чақирилади.

Матн сатри эса HTML хужжатга write усули ёрдамида босмага чиқарилади. Бу усул document объектининг усули ҳисобланади:

```
document.write(MyDate);
```

Date объекти жорий вақтни ҳам ўз ичига олади. Бу маълумотлар getHours, getMinutes ва getSeconds (соат, минут ва секунд) усуллари ёрдамида кўрилади:

```
document.write("Time: " + dt.getHours()
  + ":" + dt.getMinutes() + ":" + dt.getSeconds());
```

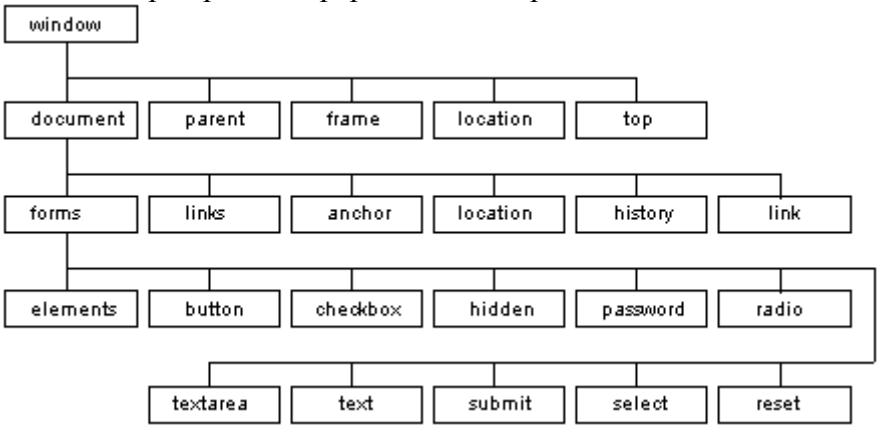
Браузер объектлари

JavaScript сценарий нуқтаи назари бўйича объектлар иерархик дарахт кўринишда ташкил этилади.

Браузер объектлари фойдаланувчи учун яратилган, браузер ойнасида жойлашган объектлар ҳисобланади. JavaScript сценарида браузер объектлари, хусусият ва усулларидан фойдаланиб бир класс асосида бошқа класс яратиб бўлмайди.

Браузер объектлари иерархияси

Қуйидаги расмда объектлар дарахти иерархияси келтирилган.



window объекти бу иерархиянинг илдизи ҳисобланади. Қачонки HTML хужжат юкланса унмнг ичида **document**, **parent**, **frame**, **location** ва **top** бошқа объектлар ҳосил бўлади.

Объектлар билан боғлиқ ҳолатлар

Браузернинг ҳар бир объекти билан аниқ бир ҳолатлар тўпламидан ташкил топади.

Масалан, **window** объекти **onLoad** ва **onUnload** ҳолатлари билан боғлиқ ҳолда ишлайди. Биринчи ҳолат браузер ойнани юклаб бўлгач ишга тушади. Иккинчиси эса браузер ойнани ёпиш вақтида ишга тушади.

Дастурчи томонидан яратилувчи объектлар

JavaScript тилида бошқа дастурлаш тилларида бўлгани каби класслар яратиш мумкин.

Аввало myRecord номли класс яратамиз. Ҳозирча унда усуллар мавжуд эмас, уларни кейинчалик қўшамиз. Бу класс қуйидагича яратилади:

```
function myRecord(name, family, phone, address)
{
  this.name    = name;
  this.family  = family;
  this.phone   = phone;
  this.address = address;
  this.secure  = false;
}
```

Кўриниб турибдики ушбу эълон қилинган классда мураккаб нарсанинг ўзи йўқ.

Яратилаётган обэъктни хусусиятларини кўрсатиш учун махсус this калит сўзидан фойдаланилади. Бу калит сўз объектнинг хусусиятларига бўлган мурожаатини кўрсатади.

Келтирилган классдан қандай фойдаланиш мумкин? Яратилган класс асосида исталган сондаги объектлар яратиш мумкин. Қуйида берилган myRecord классида иккита rec1 ва rec2 объектлари яратилган:

```
var rec1;
var rec2;
rec1 = new myRecord("Иван", "Иванов",
  "000-322-223", "А. Темур кўча, д. 225, кв. 226");
rec2 = new myRecord("Петр", "Петров",
```

```
  "001-223-3334", "Бобур кўча, д. 552, кв. 662");
rec2.secure = true;
```
Объектлар new оператори ёрдамида яратилади.

2. JavaScript объекти ва усуллари

JavaScript дастурлаш тили оддий объектга йўналтирилганлик мисолига асосланган. Объект – бу JavaScript ўзгарувчиси бўлган конструкциядир. Хусусиятлар бошқа функциялар бўлишлари мумкин. Объектлар билан богланган функциялар объект усуллари дейилади.

Объектлар ва уларнинг хусусиятлари

JavaScript объекти хусусиятларга эга. Биз объект хусусиятига қуйидаги белгилашлар тизими орқали мурожат қилишимиз мумкин:

objectName.propertyName

Объект номи ва хусусият номи регистрга нисбатан сезгир хисобланади. Биз хусусиятни қиймат бериш билан аниқлаймиз. Масалан, myCar номли объект берилган бўлсин. Биз бу объектга қуйидаги номларга эга бўлган хусусиятлар беришимиз мумкин: **make**, **model**, **year**:

myCar.make = "Ford"
myCar.model = "Mustang"
myCar.year = 69;

Шунингдек биз бу хусусиятларга юқоридаги жадвалдаги белгилашлар тизимларидан фойдаланиб мурожат қилишимиз мумкин:

myCar["make"] = "Ford
myCar["model"] = "Mustang"
myCar["year"] = 69;

Қуйидаги функция объект хусусиятининг аргумени сифатида ишлатилишини ифодалайди:

Масалан, show_props(myCar, "myCar") функциясига мурожат қилишдан чиқган натижа қуйидагича бўлади :

myCar.make = Ford
myCar.model = Mustang
myCar.year = 67

Биз хусусиятни бирон бир сон билан хам белгилашимиз мумкин, масалан:

temp[0] = 34
temp[1] = 42
temp[2] = 56

Бу мисолда teamp объектининг учта хусусияти ўрнатилмоқда , ва биз бу хусусиятларга temp[*i*] шаклида мурожат қилишимиз лозим, бу ерда *i* 0 ва 2 оралигидаги бутун сон.

Усулларни аниқлаш

Объект билан боглиқ усул - функция. Усулни стандарт функциясини қандай аниқлаган бўлсак , худди шундай йўл билан усулни хам аниқлаш мумкин . Сўнгра мавжуд объект билан функция билан боглаш учун қуйидаги синтаксисдан фойдаланамиз :

object.methodname = function_name

Бу ерда,

object –мавжуд объект,
methodname – усулга бериладиган ном,
function_name – функция номи .

Объект контекстида усулни қуйидагича чақришимиз мумкин :

object.methodname (params);

8-Маъруза

Мавзу: ОБЪЕКТЛАРНИ ЯРАТИШ ВА УЛАРГА МУРОЖАТ ҚИЛИШ

Маъруза режаси:

1. Объектларни яратиш
2. Янги объектларни яратиш
3. Усулларни аниқлаш
4. This калит сўзидан фойдаланиб объектга мурожат қилиш
5. Document объекти ва унинг усуллари

Таянч иборалар: Объектларни яратиш, янги объектларни яратиш, усулларни аниқлаш, this, low, high,объект номунасини яратиш, Document.

1. Объектларни яратиш

JavaScript махсус калит сузи *this* (бу, этот) га эга бўлиб, у жорий объектга мурожат қилиш учун ишлатилади. Масалан, бизда *validate* номли функциямиз бўлсин, бу функция объект қийматнинг тўгрилигини аниқлайди, яъни объект қиймати high(юқори қиймати) ва low(қуйи қиймати) оралигида ёки ундан чиқиб кетишини аниқлайди:

```
function validate(obj, lowval, highval)
{
  if ((obj.value < lowval) || (obj.value > highval))
    alert("Invalid Value!")
}
```

Объект функцияси турини аниқлаш объектларни яратишда ишлатилади.

2. Янги объектларни яратиш

JavaScript нинг клиентлари ва серверлари объектларни аниқлаш қаторига эга. Бундан ташқари биз ўзимизнинг хусусий объектларимизни хам яратишитмиз мумкин. Янги хусусий объектни яратиш икки қадамда бажарилади:

1. Объект функцияси турини аниқлаш.
2. **new** билан объект намунасини яратиш.

Объект турини аниқлаш учун, объект тури функциясини яратишимиз лозим, у унинг номини аниқлайди , ва унинг хусусиятларини ва усулларини аниқлайди . Масалан, биз автомобиллар учун объект турини яратмокчимиз. Бизнинг объектимиз *car* (автомобил) деб номланади, ва у make, model, year хусусиятларни аникласин. Бу ишни амалга ошириш учун қуйидаги функцияни ёзамиз:

```
function avtomobil(rangi, model, yili)
{
  this.rangi = rangi;
  this.model = model;
  this.yili = yili;
}
```

Эслатма, **this** дан фойдаланиб, объектга хусусият қийматини функцияга асосланиб бериш мумкин. Энди *mycar* номли объектни қуйидагича яратишимиз мумкин:

```
mycar = new car("Oq", "Neksiya", 1993);
```

Бу скрипт бажарилганда *mycar* объекти яратилади ва унинг хусусиятига кўрсатилган қиймат берилади.

Биз **new** фойдаланиб исталган сонданги *car* объектларни яратишимиз мумкин. Масалан,

```
kenscar = new car("kulrang", "Tiko", 1992)
```

Объект ўз навбатида мустақил обкт бўлган хусусиятга эга бўлиши мумкин . Масалан , биз *person* номи билан объект яратамиз :

```
function person(nomi, modeli, rangi, yili)
```

```
{
  this.nomi = nomi;
  tish.modeli= modeli;
  this.rangi = rangi;
  this.yili = yili;
}
```
Сўнгра иккита *person* номи билан объект яратамиз :

```
rand = new person("engil avtomobil",tiko,2002)
ken = new person("engil avtomobil",neksiya,2004)
```

3. Усулларни аниқлаш

Биз объект тури учун усуллар аниқлашимиз мумкин , бунда усулни аниқлаш объект турини аниқлашга ўтиш билан амалга оширилиши мумкин . Масалан , бизда GIF тасвир файллари терилмаси бор дейлик , ва биз car га ахборот берувчи усул яратишимиз лозим , мос тасвирлари билан албатта . Биз функция турини қуйидагича аниқлашимиз мумкин:

```
function displayCar()
{
  var result = "A neksiya " + this.nomi+ tish.modeli+ modeli;
  this.rangi + this.yili ;
  pretty_print(result)
}
```

Бу ерда *pretty_print* – қаторни кўрсатувчи олдиндан аниқланган функция .Усулга тегишли объектга мурожат қилиш учун this дан фойдаланамиз .
Биз функцияни ***car*** га тегишли усул сифатида яратишимиз хам мумкин :

```
This.displayCar = displayCar;
```

У холда car нинг тўлиқ аниқланиши қуйидагича ёзилади:

```
function car(nomi, modeli, rangi, yili)
{
  this.nomi = nomi;
  tish.modeli= modeli;
  this.rangi = rangi;
  this.yili = yili;
  this.displayCar = displayCar;
}
```

Бу янги усулни қуйидагича чақиришимиз мумкин:

```
car1.displayCar ()
car2.displayCar ()
```

4. This калит сўзидан фойдаланиб объектга мурожат қилиш

JavaScript махсус калит сўзи **this** га эга бўлиб, у жорий объектга мурожат қилиш учун ишлатилади. Масалан, бизда *validate* номли функциямиз бўлсин, бу функция объект қийматнинг тўгрилигини аниқлайди, яъни объект қиймати high ёки low эканлигини аниқлайди:

```
function validate(obj, lowval, highval)
{
  if ((obj.value < lowval) || (obj.value > highval))
    alert("Invalid Value!")
}
```

Биз *validate* ни onChange ходисаларни ишлашнинг хар бир форма элементи учун чақиришимиз мумкин.

5. Document объекти ва унинг усуллари

Фараз қилайлик, бизнинг JavaScript да ёзган дастуримиз A.htm файлида сақланиб, у бирон бир нарсани фреймга чиқарсин. Бунинг учун куйидаги кодни ёзиш лозим:

```
<HEAD>
<TITLE>Frames Page A</TITLE>
</HEAD>
<BODY>
  Page A
<SCRIPT LANGUAGE = "JavaScript">
<!--
  parent.frames[1].document.write( "Фреймга чиқариш" );
//-->
</SCRIPT>
</BODY>
</HTML>
```

Бу ерда **parent** объекти фреймлар массивини сақлайди .Биз шу массивнинг бирон бир элементига мурожат қилмокчимиз (масалан, иккинчи элементга , чунки массив 0 дан бошланади) ва **document.write** усулини чақирамиз:

```
<HTML>
<HEAD>
<TITLE>Frames Page A</TITLE>
</HEAD>
<BODY>
  Page A
<SCRIPT LANGUAGE = "JavaScript">
<!--
  parent.frames[1].document.open;
  parent.frames[1].document.write( "Вывод в фрейм" );
  parent.frames[1].document.close;
//-->
</SCRIPT>
</BODY>
</HTML>
```

1–мисол. document объекти усулидан фойдаланиб хужжат фони рангини ўзгартириш. Хужжат фони рангини ўзгартириш учун, **document** объектининг **bgColor** хусусияти қийматини ўзгартириш етарлидир:

```
 parent.frames[1].document.bgColor = "red";
```

ёки

```
 parent.frames[1].document.bgColor = "#c0c0c0";
```

2-мисол. Матн рангини ўзгартириш. Матн рангини ўзгартириш учун **fgColor** хусусияти кийматини узгартириш лозим:

```
parent.frames[1].document.fgColor = "#707070";
```

9-Маъруза

Мавзу: JAVASCRIPT НИНГ СТАНДАРТ ОБЪЕКТЛАРИ

Маъруза режаси:

1. Array объекти. Кўп ўлчамли массивларни ташкил қилиш
2. Boolean объекти
3. Date объекти. Date объекти усуллари
4. Function объекти
5. Math объекти

6. Number объекти
7. String объекти

Таянч иборалар: *Array, Array объекти усуллари, массив, кўп ўлчамли массивлар, Global, Object, Function, String, Boolean, Number, Math, Date.*

JavaScript нинг ички объектларидан фойдаланувчи объектларидан фарқли равишда Microsoft Internet Explorer ва Netscape Navigator ларда хам фойдаланиш мумкин. Ички объектларга қуйидагилар киради : Global, Object, Function, Array, String, Boolean, Number, Math ва Date.

1. Array объекти

JavaScript тили массивлар яратиш учун ички берилма турларига эга эмас , шу сабабли бу турдаги масалаларни ечиш учун Array объектидан фойдаланилади .У массивларни бирлаштириш ва уларни тартиблаштириш , хамда массив ўлчамларини урнатиш усулларига эга. *Массив* – бу тартиблаштирилган қийматлар тўплами бўлиб , уларга мурожат қилиш, уларнинг номлари ва индекслари орқали амалга оширилди. Масалан , дастурда хабарлар массивини яратиш мумкин –A[i]. Array объектини яратиш учун иккита усулдан фойдаланиш мумкин .

1-усул

Биз new конструкторини чақириб массив ўлчамини берамиз (массив элементлар сонини). Массивни тўлдириш кейинчалик амалга оширилади . Мисол.

```
<html>
<head><title>JavaScript. 12-97</title>
<script language="JavaScript">
//янги массивни яратиш
  allStr = new Array(5);
// массивни тўлдириш
  allStr[0] = "Message #1";
  allStr[1] = "Message #2";
  allStr[2] = "Message #3";
  allStr[3] = "Message #4";
  allStr[4] = "Message #5";
// массив элементларини акс эттирувчи функция
  function showMsg(ndx)
  {
    alert(allStr[ndx]);
  }
</script>
</head>
<-- Хужжатни юклашда N4 хабарини кўрсатиш лозим-->
<body onLoad="showMsg(3);">
</body>
</html>
```

Юқорида келтирилган массивда 5 та элементдан иборат бўлган массив яратилмоқда, сўнгра массив элементлари қийматлари аниқланмоқда.

2 усул

Биз бирданига new конструкторини чақирамиз ва массивнинг хамма элементлари қийматларини берамиз . Бу холда массив ўлчами ошкора берилмайди .

Мисол.

```
<html>
<head><title>JavaScript. 12-97</title>
<script language="JavaScript">
// янги массивни яратиш ва қийматларини бериш
  allStr = new Array("Message #1", "Message #2", "Message #3",
    "Message #4", "Message #5");
// массив элементи қийматларини акс эттириш фукцияси
  function showMsg(ndx)
  {
    alert(allStr[ndx]);
  }
</script>
</head>
<-- Хужжатни юклашда N4 хабарини кўрсатиш лозим-->
<body onLoad="showMsg(3);">
</body>
</html
```

Array объекти усуллари

Array объекти қуйидаги усулларга эга.

Усул	Тавсифлаш
join	Массивнинг хамма элементларини битта қаторга бирлаштиради
reverse	Массивдаги элементлар тартибини ўзгартиради - биринчи элемент охирги элемент бўлиб қолади , охирги эса биринчи бўлади .
Sort	Массив элементларини қийматлари бўйича тартиблаштиради

Бир нечта элементли массив ва иккита showAll ва showElement функциялари берилган бўлсин. Биринчи функция массивнинг хамма элементларини акс эттириш учун ишлатилади ва циклда showElement функциясини чақиради:

```
<html>
<head><title>JavaScript. 12-97</title>
<script language="JavaScript">
  myArray = new Array("Mother", "Father", "Sister", "Brother",
        "Uncle");
  function showElement(ndx)
  {
    alert(myArray[ndx]);
  }
  function showAll()
  {
    for( i = 0; i <= myArray.length-1; i++)
    {
      showElement(i);
    }
  }
</script>
</head>
<body onLoad="showAll();">
</body>
</html>
```

Агар биз юкорида келтирилган файлни юкласак , у холда биз хар бирида myArray - Mother, Father, Sister, Brother и Uncle массивнинг битта элементи акс эттирилган панеллар хабарлари кетмакетлигини кўрамиз.

Қуйида test функциясини қуйидагича яратамиз :

```
function test()
{
  alert(myArray.join());
}
```

Ва <body> теги қийматини ўзгартирамиз:

```
<body onLoad="test();">
```

Бу ерда join усули кўпинча массив элементларини ажратишни амалга оширади . Махсус кўрсатилмаганда "," символи ишлатилади .

Масалан:

```
function test()
{
  alert(myArray.join(" _|_ "));
}
```

Reverse усули массив элементларининг ўринларини алмаштириш учун ишлатилади . Бу усулни чақиришни test функциясига қўшимча қилайлик :

```
function test()
{
  myArray.reverse();
  alert(myArray.join(";"));
}
```

Бу код бажарилиши натижасида биринчи элемент охирги ўринга ўтди, иккинчи элемент эса охиридан иккинчи ўринга ўтди ва х.к.з. Sort усули эса массив элементларини қийматлари бўйича тартиблаштириш учун ишлатилади. Бу сулни чақршни test функциясига қўшимча қиламиз:

```
function test()
{
  myArray.sort();
  alert(myArray.join(";"));
}
```

Кўп ўлчамли массивларни ташкил килиш

Array объекти куп улчамли массивларни ташкил қилиш учун хам ишлатилади . Қуйида кўп ўлчамли массивлар яратилиши намойиш этилади.

```
<html>
<head><title>JavaScript. 12-97</title></head>
<body>
<center>
<font size=5><b>Multidimentional Array</b></font><p>
<script language="JavaScript">
 a = new Array(4);
 for( i=0; i < 4; i++)
 {
  a[i] = new Array(4);
  for( j=0; j < 4; j++)
  {
   a[i][j] = "["+i+","+j+"]";
  }
```

```
 }
 for( i=0; i < 4; i++)
 {
  str = "Row "+i+":";
  for( j=0; j < 4; j++)
  {
   str += a[i][j];
  }
  document.write( str, "<br>");
 }
</script>
</center>
</body>
</html>
```

Юқорида келтирилган мисолда тўртта элементдан иборат массив яратилмоқда; хар бир элемент ўз навбатида яна тўрттадан элементга эга бўлган массивдан иборат. Хар бир элементга I,j жуфтлик қийматлар киритилади.

2. Boolean объекти

Ички Boolean объекти мантиқий бўлмаган қийматларни мантиқий қийматга (булева қийматга) ўзгартириш учун ишлатилади. Ички Boolean объектининг экземпилярини яратиш учун new конструктори ишлатилади:

```
bfalse = new Boolean(false);
btrue  = new Boolean(true);
```

ValueOf усули булева ўзгарувчининг қийматини булева қиймат сифатида қайтаради, toString усули эса – қаторли қиймат шаклида қайтаради. Мисол:

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<script language="JavaScript">
 // иккита булева ўзгарувчи яратамиз
 bfalse = new Boolean(false);
 btru = new Boolean(true);
 // уларнинг қийматини чиқарамиз (мантиқий қийматлар)
 document.write(bfalse.valueOf()+"<br>");
 document.write(btrue.valueOf()+"<br>");
 // қаторли кийматни чиқарамиз
 document.write(bfalse.toString()+"<br>");
 document.write(btrue.toString()+"<br>");
</script>
</body>
</html>
```

3. Date объекти

Date объекти ва унинг усуллари скриптли дастурларда вақт ва сана билан ишлаш учун ишлатилади. Бу объект хусусиятларга эга эмас. Объектнинг экземпилярини яратиш учун Date конструктори ишлатилади :

```
MyDate = new Date([параметрлар]);
```

Қуйидаги параметрларни бериш мумкин:

- Хеч канака параметр берилмаса - экземпляр жорий вақт ва санани аниқлайди. Масалан, today = new Date();
- Қатор қуйидаги санани сақласа : "Ой, кун, йил, соат: минутлар:секундлар". Масалан, someDate = new Date("May 15, 1996").

Мисол.

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<script language="JavaScript">
  today = new Date();
  document.write("today="+today+"<br>");
  someDate = new Date("May 16, 1996");
  document.write("someDate="+someDate+"<br>");
  otherDay = new Date( 96, 4, 15);
  document.write("otherDay="+otherDay+"<br>");
  sameDay = new Date( 96, 4, 16, 15, 30, 0);
  document.write("sameDay="+sameDay+"<br>");
</script>
</center>
</body>
</html>
```

Date объектининг усуллари

Date объекти усулларини қуйидаги категорияларга ажратиш мумкин :

- *Ўрнатиш усуллари (set)* - Date объекти билан вақт ва санани ўрнатиш усуллари ;
- *Аниқлаш усуллари (get)* - Date объекти экземпилярида вақт ва санани ўрнатиш усуллари;
- *Ўзгартириш усуллари (to)* – вақт ва санани қаторга айлантириш усуллари;
- *Санани ишлаш усуллари*.

Ўрнатиш ва аниқлаш усуллари секунд, минут, соат, ойдаги кунларни ўрнатиш ва ўзгартириш учун ишлатилади. Хамма усуллар бутун турдаги қийматларни ишлатади:

Қийматлар	Диапазони
Секунд ва минутлар сони	0..59
Соатлар сони	0..23
Хафта кунлари	0..6
Сана	1..31
Ой	0..11 (Январ..Декабр)
Йил	Йиллар сони

Мисол. Биз 15 май 1996 йил санасини берайлик.

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<p>
<script language="JavaScript">
  someDate = new Date( "May 15, 1996" );
  document.write("someDate="+someDate+"<br>");
  document.write("getDay  ="+someDate.getDay()+"<br>");
  document.write("getMonth="+someDate.getMonth()+"<br>");
  document.write("getYear ="+someDate.getYear()+"<br>");
```

```
</script>
</center>
</body>
</html>
```

getTime ва setTime усулларини саналарни таққослаш учун ишлатиш қулайроқдир. Масалан, жорий йилда қанча кунлар қолгананини билиш дастурини келтирамиз.

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<br><br><br>
<script language="JavaScript">
  today = new Date();
  // санани бериш
  endYear = new Date("December 31, 1990");
  // йилни алмаштириш
  endYear.setYear(today.getYear());
  // числони миллий секундларда бир кун учун хисоблаш
  msPerDay = 24 * 60 * 60 * 1000;
  // кунлар сонини олиш
  daysLeft = (endYear.getTime() - today.getTime()) / msPerDay;
  // яхлитлаш
  daysLeft = Math.round(daysLeft);
  // курсатиш
  document.write("Number of days left in the year: "+daysLeft);
</script>
</center>
</body>
</html>
```

Parse усули санани қаторли шаклда ифодалаш учун хам ишлатилиши мумкин :

```
someDate = new Date();
someDate.setTime(Date.parse("May 15, 1996"));
```

Жорий вактни чикарувчи дастур ёзамиз. Вакт «кнопка» интерфейси элементи сарловхаси сифатида чиқарилади.

```
<html>
<head><title>JavaScript 12.97</title>
<script language="JavaScript">
  function showTime()
  {
    // жорий санани оламиз
    var time = new Date();
    // соатлар сонини аниқлаймиз
    var hour = time.getHours();
    // минутлар сонини аниқлаймиз
    var minute = time.getMinutes();
    // секундлар сонини аниқлаймиз
    var second = time.getSeconds();
    // соатларни кўрсатамиз
    var temp = hour;
    // минутларни кўрсатамиз
    temp += ((minute < 10) ? ":0" : ":") + minute;
    // секукндларни кўрсатамиз
```

```
   temp += ((second < 10) ? ":0" : ":") + second;
   // ифодалаймиз
   document.forms[0].clock.value = temp;
   // хар бир секундни кўрсатамиз
   id = setTimeout( "showTime()", 1000);
  }
</script>
</head>
<body onLoad="showTime();">
<center>
<font size=5><b>
 JavaScript Date & Time Demo
</b></font>
<form name="DateDemo">
 <input type="button" name="clock" value="xxxxxxxx">
</form>
</center>
</body>
</html>
```

Санани ифодалавчи функция қуйидагича ёзилиши мумкин:

```
 function showDate()
 {
   // жорий вақтни оламиз
   var D = new Date();
   // хафта кунини биламиз
   var DOW = D.getDay();
   // санани биламиз
   var Day = D.getDate();
   // ойни биламиз
   var Month = D.getMonth();
   // йилни биламиз
   var Year = D.getYear();
   // кўрсатамиз
   temp = Day + "/" + Month + "/" + Year;
   document.forms[0].clock.value = temp;
 }
```

Қуйида биз Array объектидан фойдаланиб хафта кунлари номлари массивини ташкил этамиз :

```
 dayNames = new
       Array("якшанба","душанба","сешанба","чоршанба",
                  "пайшанба","жума","шанба");
```

4. Function объекти

Ички объект Function JavaScript да қатор кодини бериш учун ишлатилади . Объектнинг экземплярини яратиш учун new конструктори ишлатилади:

```
 var newBgColor = new Function("c", "document.bgColor=c");
```

Бу функциядан фойдаланишни намойиш қилиш учун қуйидаги мисолни келтирамиз .

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<script language="JavaScript">
 var newBgColor = new Function("c", "document.bgColor=c");
```

```
 function fnDemo()
 {
  newBgColor("#a7b7c7");
 }
</script>
<form>
<input type="button" value="bgColor" onClick="fnDemo();">
</form>
</center>
</body>
</html>
```

Яна бита мисол кўрайлик. Бизга иккита аргументларни кўпайтирувчи функция яратиш лозим бўлиб қолсин. Бу функцияга myMult деб ном берамиз:

```
 var myMult = new Function("x", "y", "return x*y");
```

Ундан қуйидагича фойдаланиш мумкин :

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<script language="JavaScript">
 var myMult = new Function("x", "y", "return x*y");
 document.write("10*20="+myMult(10, 20));
</script>
</center>
</body>
</html>
```

Биз хатто унча катта бўлмаган форма ясаб унга бу натижаларни (икки сон кўпайтмаси) чиқаришни хам амалга оширадиган скриптни қуйидагича ёзишимиз мумкин:

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<script language="JavaScript">
 var myMult = new Function("x", "y", "return x*y");
 function calc()
 {
  document.forms[0].sum.value =
   myMult(document.forms[0].x.value, document.forms[0].y.value);
 }
</script>
</center>
<form>
<input type="text" name="x"   value=10  size=4>
<input type="text" name="y"   value=10  size=4>
<input type="text" name="mul" value=" " size=4>
<input type="button" value="calc" onClick="calc();">
</form>
</body>
</html>
```

5. Math объекти

Math ички объекти хар хил константаларнинг хусусиятлари ва усулларини олиш учун ишлатилади ва математик функцияларнинг бажарилишини таъминлайди. Бу кийматларни чиқариш дастури:

```
<html>
<head><title>JavaScript 3 Demo</title></head>
<body bgcolor="#a7b7c7">
<center><font size=4><b>Math</b> Object Properties</font><p>
<script language="JavaScript">
 s = "<TABLE BORDER=2>" +
    "<TR><TD><B>E</B></TD><TD>"      + Math.E      + "</TD></TR>" +
    "<TR><TD><B>LN2</B></TD><TD>"    + Math.LN2    + "</TD></TR>" +
    "<TR><TD><B>LN10</B></TD><TD>"   + Math.LN10   + "</TD></TR>" +
    "<TR><TD><B>LOG2E</B></TD><TD>"  + Math.LOG2E  + "</TD></TR>" +
    "<TR><TD><B>LOG10E</B></TD><TD>" + Math.LOG10E + "</TD></TR>" +
    "<TR><TD><B>PI</B></TD><TD>"     + Math.PI     + "</TD></TR>" +
    "<TR><TD><B>SQRT1_2</B></TD><TD>" + Math.SQRT1_2 + "</TD></TR>" +
    "<TR><TD><B>SQRT2</B></TD><TD>"  + Math.SQRT2  + "</TD></TR>" +
    "</TABLE>";
 document.write(s);
</script>
</center>
</body>
</html>
```

Math объекти билан олинувчи усуллар қуйидаги жадвалда келтирилган :

Усул	Тавсифи
abs	Аргументнинг абсолют қийматини қайтаради
sin, cos, tan	Стандарт тригонометрик функциялар, аргументлар радианларда берилади
acos, asin, atan	Тескари тригонометрик функциялар.
exp, log	Экспонента ва натурал логарифма, асоси - e
ceil	Аргументга тенг ёки катта бутун сонни қайтаради
floor	Аргументга тенг ёки кичик бутун сонни қайтаради
min, max	Икки аргументдан каттасини ёки кичигини қайтаради
pow	Аргумент даражасини қайтаради
round	Аргументни энг яқин бутунга яхлитлайди
sqrt	Квадрат илдизни қайтаради

Math объектининг кўплаб хусусиятларини ва усулларидан фойдаланилганда with операторидан фойдаланиш мақсадга мувофиқдир:

```
 with( Math )
 {
   a = PI * r*r;
   b = floor(c);
   x = r * sin( theta );
   y = r * cos( theta );
   z = round( b );
 }
```

6. Number объекти

Number ички объектининг хусусияти максимал қиймат , "not-a-number" (NaN) туридаги ва чексизлик қийматлари туридаги константаларни олиш учун ишлатилади.

```
<html>
<head><title>JavaScript 3 Demo</title></head>
<body bgcolor="#a7b7c7">
<center><font size=4><b>Number</b> Object Properties</font><p>
<script language="JavaScript">
 s = "<TABLE BORDER=2>" +
    "<TR><TD><B>MAX_VALUE</B></TD><TD>"       +
      Number.MAX_VALUE       + "</TD></TR>" +
    "<TR><TD><B>MIN_VALUE</B></TD><TD>"       +
      Number.MIN_VALUE       + "</TD></TR>" +
    "<TR><TD><B>NaN</B></TD><TD>"             +
      Number.NaN             + "</TD></TR>" +
    "<TR><TD><B>NEGATIVE_INFINITY</B></TD><TD>" +
      Number.NEGATIVE_INFINITY + "</TD></TR>" +
    "<TR><TD><B>POSITIVE_INFINITY</B></TD><TD>" +
      Number.POSITIVE_INFINITY + "</TD></TR>" +
    "</TABLE>";
 document.write(s);
</script>
</center>
</body>
</html>
```

7. String объекти

JavaScript тили ички string туридаги ички берилмаларга эга эмас. Шундай бўлсада, String объекти ва унинг усулларидан фойдаланиб, биз скриптли дастурларда қаторлар билан ишлашни амалга оширишимиз мумкин. String объекти қаторларни бошқариш учун кўплаб усулларга эга ва улардан бири (length) хусусияти, у қатор узунлигини аниқлайди.

Экземпляр объекта String объекти экземпиляри new конструктори ёрдамида яратилади:

```
myString = new String("myString Object");
```

String объекти икки турдаги усулларга эга.

String объекти усуллари куйидаги жадвалда келтирилган:

String объекти усуллари

Усул	*Тавсифи*
anchor	Anchor HTML-элементини яратади
big, blink, bold, fixed, italics, small, strike, sub, sup	Мос HTML-элемент яратилади
charAt	Қаторнинг кўрсатилган позициясидаги символни қайтаради
indexOf, lastIndexOf	
link	HTML-мурожатни яратади
split	String объектини қаторлар массивига бўлади
substring	Кўрсатилган қисм қаторни қайтаради

String объекти усулларидан фойдаланишга мисоллар.

```
<html>
```

```
<head><title>JavaScript 3 Demo</title></head>
<body bgcolor="#a7b7c7">
<center>
<script language="JavaScript">
  // янги қатор яратамиз
  var s = new String('Netscape Navigator');
  // янги қисм қатор оламиз
  var n = s.substring(0,8);
  // қисм қатор ва қаторни акс эттириш
  document.write("string="+s+"<br>substring="+n);
  // пастки регистрга ўзгартирамиз
  var l = s.toLowerCase(s);
  // кўрсатамиз
  document.write("<br>"+l);
  // йуқорги регистрга ўзгартирамиз ва кўрсатамиз
  document.write("<br>"+s.toUpperCase(s));
</script>
</center>
</body>
</html>
```

Биз ўзгартириш усулларини кўрдик. Энди қаторнинг HTML версиясини қайтарувчи усулларни кўрамиз:

```
<html>
<head><title>JavaScript 3 Demo</title></head>
<body bgcolor="#a7b7c7">
<center>
<script language="JavaScript">
  // янги қаторни яратамиз
  var s = new String('Netscape Navigator');
  // қўшимча қиламиз   <b> и </b>
  document.write(s.bold(s)+"<br>");
  // қўшимча қиламиз    <i> и </i>
  document.write(s.italics(s)+"<br>");
  // қўшимча қиламиз     <tt> и </tt>
  document.write(s.fixed(s));
  // қўшимча қиламиз     <sub> и </sub>
  document.write(s.sub(s)+" ");
  // қўшимча қиламиз     <sup> и </sup>
  document.write(s.sup(s)+"<br>");
  // қўшимча қиламиз     <strike> и </strike>
  document.write(s.strike(s)+"<br>");
</script>
</center>
</body>
</html>
```

10-Маъруза

Мавзу: БРАУЗЕР ОЙНАСИ ХУСУСИЯТИНИ ДАСТУРЛАШ

Маъруза режаси:

1. Статус майдони (статус қатори, status bar)

2. Location майдони
3. Браузер тури (Navigator объекти)
4. Графикани дастурлаш

Window объектлари класси - JavaScript объектлари иерархиясидаги энг юқорги класс хисобланади. Бу объектлар классига window ва frame объектлари киради. Window объекти браузер дастури ойнаси билан ассоцияция қилинади, frame объекти эса браузер ойнаси ичида ассоцияция қилинади. JavaScript да дастурлашда window туридаги объектларнинг жуда кўп холларда қуйидаги усуллар ва хусусиятлар ишлатилади:

Хусусият	Усуллар
• status • location • history • navigator	• open • close • focus

Window объекти фақатгина ойнани очишда пайдо бўлади.

1. Статус майдони (статус қатори, status bar)

Статус майдони (status bar) деб браузер ойнасининг(HTML сахифасини ифодалаш сохасидан кейинги) пастки қисмининг ўрта майдонига айтилади. Статус майдонида браузернинг ишлаш холати ифодаланади (хужжат юкланиши, графика юкланиши, юклашни тугаллаш, апплетларни ишга тушуриш).

2. Location майдони

location майдони юкланган хужжатнинг URL ни ифодалайди. Агар биз қўлда бошқа бир сахифага ўтмокчи бўлсак (ёки унинг URL ни термоқчи бўлсак), у холда биз бу ишни location майдонида амалга оширимиз лозим. Бу майдон браузер ойнасининг юқорги қисмига жойлашган, инструментлар панелидан кейин. Умуман олганда location – бу объект. Умуман олганда location класси window ва document классларига қисм класс бўлиб киради. Бундан ташқари location – класси URL классларига хам қисм классдир. Location URL URL нинг хамма хусусиятларини меросланади. Location объектининг характеристикалари ва усуллари билан танишамиз.

Қуйидаги код автоматик равишда қуйидаги сахифани юклайди http://google.com :

```
<html>
<head>
</head>
<body>
<script>
document.location="http://google.com";
</script>
</body>
</html>
```

3. Браузер тури (Navigator объекти)

Браузерлар турининг кўпайиши натижасижда сахифаларни конкрет бир браузерга ростлаш муаммоси хам тугулмоқда . Бу холда иккита вариант туради : браузер турини сервер томонида туриб аниқлаш ва браузер турини мижоз томонида туриб аниқлаш . Иккинчи вариант учун JavaScript да Navigator объекти мавжуд . Бу объект window объекти хусусиятига эга.

Оддий кўриб чиқиш дастур турини аниқловчи оддий мисол кўрайлик :

<form><input type=button value="Тип навигатора"
onClick="window.alert(window.navigator.userAgent);"></form>

Кнопкани боссак огохлантирувчи ойна чиқади. Бу ойнада userAgent қаторига браузер мос HTTP-сарловхани жойлаштиради.

Бу қаторни ташкилий компоненталари бўйича ажратиш мумкин:

Navigator хусусияти рўйхати:

navigator.userAgent

4. Графикани дастурлаш

Image объекти ёрдамида биз web сахифалардаги график тасвирларга ўзгартиришлар киритишимиз мумкин. Хусусан бу имконият бизга мультипликация яратиш имконини яратади. Қандай қилиб JavaScript дан web сахифалардаги тасвирларга ўтишни кўрайлик. Бу тилда хамма тасвирлар массивлар сингари ифодаланади. Бу массив images деб аталади ва у document объекти хусусияти хисобланади. Web сахифалардаги хар бир тасвир тартиб рақамга эга бўлади: биринчи тасвир 0 тартиб рақамига эга бўлади, иккинчиси эса 1 номерга ва ҳ.к. Шундай қилиб, биринчи тасвирга document.images[0] ёзув билан мурожат қилишимиз мумкин. HTML хужжатнинг хар бир тасвири Image объекти деб қаралади. Image объекти маълум бир хусусиятларга эга , ва уларга JavaScript тилидан мурожат қилиш мумкин. Масалан, биз тасвир қандай хажмга эга эканлигини биз унинг хусусиятига қуйидагича мурожат қилиб билишимиз мумкин қилишимиз мумкин *document.images[0].width.*

5. Янги тасвирларни юклаш

Биз Web сахифадаги тасвирни алмаштирмоқчимиз ва бу ишда биз src атрибутидан фойдаланамиз. Бу атрибут қаралаётган тасвир адресини сақлайди. Энди биз янги тасвир адресини тайинлашимиз мумкин. Натижада, тасвир янги адресдан юкланади. Мисол:

<img src="img1.gif" name="myImage" width=100 height=100>

Бу ерда img1.gif тасвири юкланмоқда ва у myImage номини олмоқда .Қуйидаги қаторда олдинги тасвир img1.gif янги тасвир img2.gif га алмаштирилмоқда:

document.myImage.src= "img2.src";

Шунингдек янги тасвир олдинги тасвир ўлчамига эга бўлади .Энди биз тасвир жойлашган майдон ўлчамини ўзгартиришимиз мумкин.

11-Маъруза
JAVASCRIPT НИНГ АСОСИЙ СТАНДАРТ ФУНКЦИЯЛАРИ

Маъруза режаси:

1. eval – функцияси.

2. ParseInt ва parseFloat функциялари.
3. Java Script функциялари ва усуллари.
4. Формаларни ишлаш.

Таянч иборалар: *eval, ParseInt, parseFloat, acos, number*

1. Eval функцияси

Бу функция аргументи қатордан иборат бўлади. Қатор деб исталган *JavaScript* ифодаси бўлган қаторга айтамиз. Ифодада ўзгарувчилар ва мавжуд объектларнинг хусусиятлари қатнашиши мумкин. Агар аргумент ифода бўлса, у холда eval ифодани хисоблайди. Бу функция сонли ифодаларни бахолаш билан чекланмаган. Бу аргумент объектларга мурожатларни хам сақлаши мумкин . Масалан, иккита аргументли setValue функциясини аниқлаймиз : объект ва қиймати:

```
function setValue (myobj, myvalue)
{
   eval ("document.forms[0]." + myobj + ".value") = myvalue;
}
```

Энди эса биз уни бу функуцияни "text1" форма элементи қийматини ўрнатиш учун чақирамиз:

```
setValue (text1, 42)
```

2. ParseInt ва parse Float функциялари

Бу икки стандарт функциялар қаторли аргументлар учун сонли қиймат қайтариш учун хизмат қилади. ParseFloat функцияси аргументни текширади ва қаторли аргументни қўзгалувчан вергулли сонга айлантиради. Агар parseFloat функцияси мумкин бўлмаган символларни пайқаса у ишламайди ва бу функциядан кейинги қаторга бошқариш узатилади. Иккинчи функция parseInt эса қаторли аргументни тахлил қилади ва бутун сонли қиймат қайтаради. Масалан, radix параметри , 10 га тенг бўлса, string ўнли сонга айлантирилади, агар 8 га тенг бўлса саккизлик сонга ва 16 – бўлганда ўн олтили сонга ва агар 2 га тенг бўлса иккилик сонга айлантиради.

3. JavaScript функциялари усуллари

abs	forward	setDate
acos	getDate	setHours
alert	getDay	setMinutes
anchor	getHours	setMonth
asin	getMinures	setSeconds
atan	getMonth	setTime
back	getSeconds	setTimeout
big	getTime	setYear
blink	getTimezoneOffset	sin
blur	getYear	small
bold	go	sqrt
ceil	indexOf	strike
charAt	isNaN	sub
clearTimeout	italics	submit
click	lastIndexOf	substring
close (объект document)	link	sup

close (объект window)	log	tan
confirm	max	toGMTString
cos	min	toLocaleString
escape	open (объект document)	toLowerCase
eval	open (объект window)	toUpperCase
exp	parse	unescape
fixed	parseFloat	UTC
floor	parseInt	write
focus	pow	writeln
fontcolor	prompt	
fontsize	random	

abs усули **- Соннинг абсалют қийматини қайтаради.**

Синтаксиси:

Math.abs(*number*)

Number- мавжуд объектнинг исталган сонли ифодаси ёки хусусияти:

Math

acos усули **- соннинг арккосинусини қайтаради (радианларда).**

Синтаксиси:

Math.acos(*number*)

Number – бу -1 ва 1 оралигидаги сонли ифода ёки мавжуд объект хусусияти.

Math усули

Тавсифи:

acos усули 0 ва Пи орасидаги сонли қийматни қайтаради. Агар *number* қиймати бу ораликдан ташқарида бўлса, қайтариладиган қиймат хамма вақт 0 бўлади.

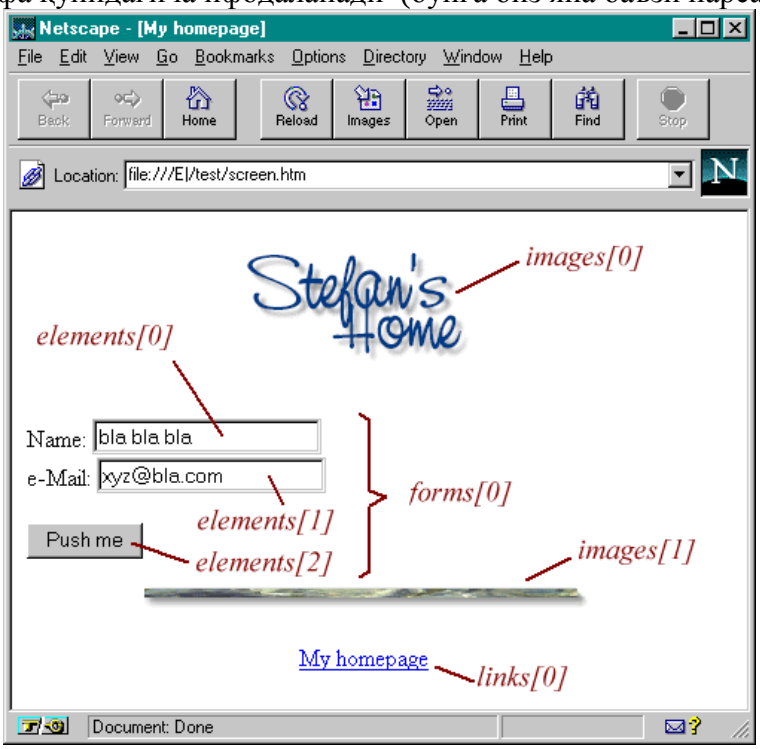
4. Формаларни ишлаш

JavaScript тилининг хамма элементлари web-сахифада иерархик таркибда жойлаштирилади. Хар бир элемент объект сифатида қабул қилинади. Хар бир объект аниқ бир ххусусиятларга ва усулларига эга булади. JavaScript тили Web сахифаларидаги ообъектларни осонгина бошқаришни таъминлайди . Бу ишларин оддий HTML сахифасида кўриб чиқамиз:

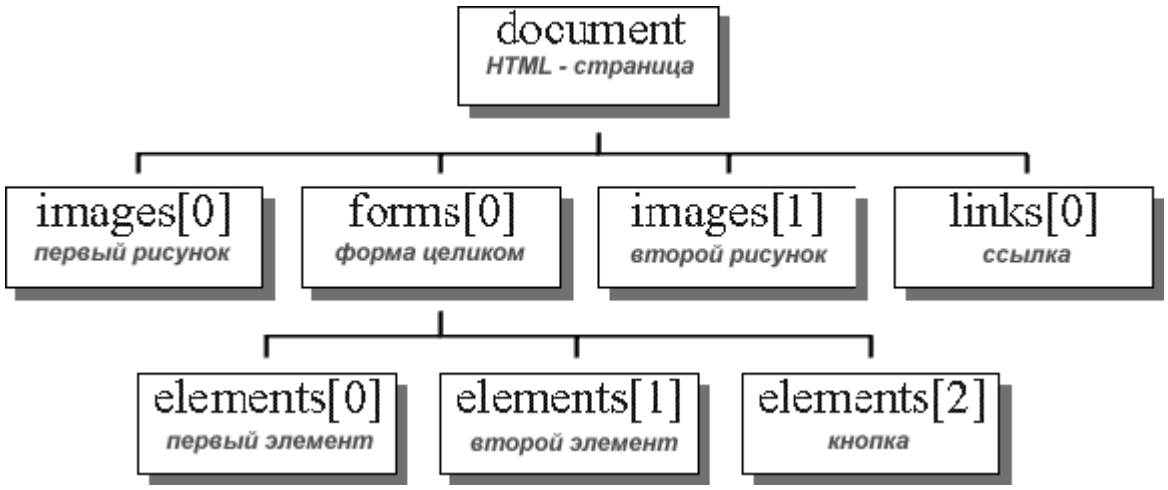
```
<html>
<head>
</head>
<body>
<center>
<img src="home.gif" name="pic1" width=200 height=100>
</center>
<p>
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
e-Mail:
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me" name="myButton" onClick="alert('Yo')">
</form>
<p>
<center>
```

<img src="ruler.gif" name="pic4" width=300 height=15>
<p>
My homepage
*** ***
</body>
</html>

Экранда эса бу сахифа қуйидагича ифодаланади (бунга биз яна баъзи нарсаларни қўшдик):



Шундай қилиб биз, иккита расмга эгамиз, мурожат ва матнни киритиш ва кнопкани киритиш учун икки майдондан иборат формага эгамиз. JavaScript тили нуқтаий назарида браузер ойнаси - бу бирон бир window ойнасидир. Бу объект баъзи бир жихозлаш элементларига эга. Ойна ичига биз HTML хужжатини жойлаштиришимиз мумкин (ёки бирон бир бошқа турдаги файлни). Бундай сахифа умуман олганда document объектидир. Бу дегани, document объекти JavaScript тилида хозирги вақтда HTML хужжатига юклатилган дегани. JavaScript тилида document жуда мухим объект хисобланади. Биз ундан жуда кўп марталаб фойдаланамиз. Document объекти хусусиятларига web сахифа фон ранги хам киради. Биз учун аммо шу нарса мухимки хама HTML объектлари document объекти хусусиятлари хисобланади. Қуйидаги расмда HTML хужжати яратилиш иерархияси келтирилган:



Биз албатта бошқаришни амалга ошириш учун бу иерархиянинг исталган объектлари тўгрисида ахборотлар олишимиз лозим. Бунинг учун биз хар бир объектга JavaScript тилида кириш қандай ташкиллаштрилиганлиги маълум бўлиши лозим. Иерархик таркибдан кўриниб турибтики хар бир объект ўз номига эга. Демак, ундан фойдаланишимиз лозим. Бу ерархияда биринчи элемент document деб аталади. Биринчи расм эса images[0] объекти сифатида ифодаланган. Биз энди бу объектга document.images[0] деб мурожат қиламиз. Масалан, биринчи майдонга матн киритиш учун мурожат қуйидагича ёзилади :

document.forms[0].elements[0]

Фойдаланувчи томонидан киритилган матн қандай аниқланади? Матн киритиш майдонига мос келган элемент value хусусиятига эга бўлади. Шундай қилиб, биз энди излаётган қийматни ўқиш учун хамма воситаларга эгамиз. Бунинг учун JavaScript да қуйидаги қаторни ёзиш лозим:

name= document.forms[0].elements[0].value;

Олинган қатор name ўзгарувчисига киритилади . Энди биз бу ўзгарувчи билан ишлашимиз мумкин. Масалан, биз очилувчи ойна яратишимиз мумкин , бунинг учун қуйидаги буйруқдан фойдаланамиз *alert("Hi" + name)*. Натижада , агар фойдаланувчи бу ойнага 'Ахмад ' сўзини киритса, у холда *alert("Hi" + name)* буйруги билан табрикловчи ' Hi Ahmad ' сўзи чиқади.

Агар биз катта сахифалар билан ишласак, у холда хар хил объектларга ахборот узатиш жуда қийинлашади. Шу сабабли хар хил объектларга одатда ўзимиз уникал ном беришимиз лозим. Бу ишни қуйидаги мисолда курсатамиз:

```
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
...
```

Бу ёзувдан кўриниб турибтики *forms[0] объекти* энди иккинчи ном *myForm* ни олади.

name= document.forms[0].elements[0].value;

Уни биз қуйидагича хам ёзишимиз мумкин

name= document.myForm.name.value;

Бу ишлар JavaScript да катта Web сахифалар билан ишлашда дастурлашни жиддий осонлаштиради . Шуни эсдан чиқармаслик лозимки *myform* ўрнига *myForm* ёзиш мумкин эмас. JavaScript нинг кўпгина объектлари хусусиятларини ўқиш ва унинг қийматини ўзгартириш мумкин:

```
<form name="myForm">
<input type="text" name="input" value="bla bla bla">
<input type="button" value="Write"
onClick="document.myForm.input.value= 'Yo!'; ">
```

Дастлабки код :

```
<html>
<head>
<title>Objects</title>
<script language="JavaScript">
<!-- hide
function first() {
// очилувчи ойна яратилади, унда
// форма майдонида киритилган матн жойлашади
alert("The value of the textelement is: " +
document.myForm.myText.value);
}
function second() {
// бу функция улагичларнинг холатини текширади
var myString= "The checkbox is ";
// улагич уланган ёки уланмаган?
```

```
if (document.myForm.myCheckbox.checked) myString+="checked"
else myString+= "not checked";
// каторни экранга чиқариш
alert(myString);
}
// -->
</script>
</head>
<body bgcolor=lightblue>
<form name="myForm">
<input type="text" name="myText" value="bla bla bla">
<input type="button" name="button1" value="Button1"
onClick="first()">
<br>
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Button2"
onClick="second()">
</form>
<p><br><br>
<script language="JavaScript">
<!-- hide
document.write("The background color is: ");
document.write(document.bgColor + "<br>");
document.write("The text on the second button is:");
document.write(document.myForm.button2.value);
// -->
</script>
</body>
</html>
```

12-Маъруза

Мавзу: PHP - СЕРВЕР ТОМОНДАН ДАСТУРЛАШ. PHP ГА КИРИШ. PHP НИ ЎРНАТИШ ВА ТЕСТЛАШ.

Маъруза режаси:

1. PHP тарихи
2. PHP – веб техногия тил
3. PHP имкониятлари
4. Дастурий воситани созлаш ва ўрнатиш
5. Денвер дистрибутиви.

Таянч иборалар: *PHP, HTML/CSS, JavaScript, Денвер,* CGI, *Apache сервер*

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.

1. 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.

2. PHP – веб техногия тил

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 тезлаштирсада, кодни клиент кўриш имкониятига эга бўлади. Барибир ҳар иккисини бошқаси боса олмайдиган ўз ўрни бор, равшанки бу ўрин PHPда муҳимроқ ва каттароқ.

3. 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.
- 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.

4. 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.

5. 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 компонентларни 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.

13-Маъруза

Мавзу: PHP СТРУКТУРАСИ. ЎЗГАРУВЧИЛАРИ. ИФОДАЛАР. МАЪЛУМОТ ТИПЛАРИ.

Маъруза режаси:

1. PHP скриптлар
2. Асосий синтаксислар
3. Ўзгарувчилар, ўзгармаслар ва амаллар
4. Маълумотлар типлари

Таянч иборалар: *php, $, echo, integer, float, string, array, object, resource, null*.

1. PHP-скриптлар

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:

1-мисол. PHP кодида тузилган содда html-файл.

```
<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.):

```
<? echo "Бу PHP тилида
      оддий қайта ишлашнинг
      инструкцияси"; ?>

<script language="php">
 echo "Бир нечта редакторлар
     (FrontPage) қуйидагича
     қабул қилишади";
</script>

<% echo " ASP технологиясидаги тегдан
      ҳам фойдаланса бўлади"; %>
```

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.

2. 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 буйрукдан 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.

```
<?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.
?>
```

PHP дастурлаш тилида комментарийнинг қўлланилиши.

3. Ўзгарувчилар, ўзгармаслар ва амаллар

Ҳар бир дастурлаш тилида муҳим элементлардан бири бу *ўзгарувчилар*, *ўзгармаслар* ва улар қўлланиладиган *амаллар*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.

*PHP*3 дастурлаш тилида ўзгарувчилар ҳар доим бирор-бир қийматга ўзлаштирилади. Яъни ўзгарувчини бирор-бир ифодага ўзлаштирсак, ифоданинг қиймати ўзгарувчига ўзлашади. Буни қуйидаги мисолда кўриш мумкин, яъни бир ўзгарувчини қиймати бошқасига ўзлаштирилганда улардан бирини қийматини бошқасига таъсир кўрсатмайди.

```
<?php
$first = ' Text '; // $first ўзгарувчига
                   // ' Text ' қийматни
                   //  ўзлаштирдик
$second = $first; // $second ўзгарувчига
                  // $first ўзгарувчи
                  // қиймати ўзлаштирилди
$first = ' New text '; // $first ўзгарувчи
                       // қиймати
                       // 'New text' қийматга
                       // ўзгартирилди
echo "first номли ўзгарувчи қиймати ".
    "$first га тенг. <br>";
       // $first ўзгарувчи қийматини экранга чиқарамиз
echo "second номли ўзгарувчи қиймати ".
    "$second га тенг.";
       // $second ўзгарувчи қийматини экранга чиқарамиз
?>
```

Мисол: Қиймат бўйича ўзлаштириш.

Бу скриптни натижаси қуйидагича бўлади:

```
first номли ўзгарувчи қиймати Text га тенг.
second номли ўзгарувчи қиймати New text га тенг.
```

*PHP*4 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:

```
<?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";
?>
```

Мисол: Ссылкалар бўйича ўзлаштириш.

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 киймати 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.

```
<?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.
?>
```

Мисол: PHP дастурлаш тилида ўзгармаслар.

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:

13.1-жадвал. Арифметик амаллар.

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

13.2-жадвал. Сатрли амаллар.

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

13.3-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 қисқартирилган шакли(даставвал сатрлар қўшилади, сўнгра	$b = "Ҳаммага "; $b .= "салом"; (`$b = $b .`

		ҳосил бўлган сатр ўзгарувчига ўзлашади).	"салом" ифодага эквивалент;) Натижаси: $b="Ҳаммага салом"

13.4-жадвал. Матиқий амаллар.

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

13.5-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

13.6-жадвал. Инкремент ва декремент амаллари.

Белгиланиши	Номланиши	Изох	Мисол
++$a	Пре-*инкремент*	$a қиймати бирга оширилади ва $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	

4. 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.

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

Мисол: Мантиқий тип.

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.

```
<?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";}
?>
```

Мисол: Мантиқий типларнинг қўлланилиши.

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.

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.

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

Мисол: PHP дастурлаш тилида массивлар.

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.

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

Мисол: PHP дастурлаш тилида массивлар.

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 массиввига
                      //  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.

```
<?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);
?>
```

Мисол: Массивни қайта индекслаймиз.

Бу скриптнинг натижаси қуйидагича бўлади:

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

Object (объектлар) типи.

Объектлар – объектга йўналтирилган дастурлашдан кириб келган *маълумот типи*дир. Объектга йўналтирилган дастурлаш тамойилига кўра, синф – аниқ хоссаларга эга ва улар билан ишлайдиган методли объектлар тўплами. Объект эса мос равишда синф нусхасидир. Масалан, дастурчилар – бу дастурни тузувчи, компьютер адабиётларини ўрганадиган одамлар синфи ва бундан ташқари ҳамма одамлар қатори исм ва фамилияси мавжуд. Энди агарда бирор аниқ дастурчи – Азамат Бобоевни олсак, у ҳолда уни шу хоссага эга бўлган дастурчи синфини объекти сифатида қараш мумкин ва у ҳам дастур тузади, ҳамда исми мавжуд ва бошқалар.

PHP дастурлаш тилида *объект методи*га мурожаат -> *амали*дан фойдаланилади. Объектни инициализация қилишда *объектни ўзгарувчан нусхаси*ни яратадиган new ифодасидан фойдаланилади.

```
<?php
// одам синфини яратамиз.
class Person
{
// PHP дастурлаш тилини ўрганадиган одам методи
   function know_php()
      {
         echo "Энди мен PHP дастурлаш тилини биламан!";
      }
}
$bob = new Person; // одам синфини
                // объектини яратамиз.
$bob -> know_php(); // уни PHP тилига ўргатамиз.
?>
```

Мисол: PHP дастурлаш тилида объектлар.

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 қўпол 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.

14-Маъруза

Мавзу: PHP ДА ЖАРАЁНЛАРНИ БОШҚАРИШ

Маъруза режаси:

1. Шарт операторлари
2. Алтернатив синтаксислари
3. Цикллар
4. Бошқарув ўтказувчи операторлар

Таянч иборалар: *if, else, elseif, while, for, foreach, switch, break, continue*

1. Шарт операторлари

if оператори.

Бу *PHP* дастурлаш тилидаги барча дастурлаш тиллари каби жуда муҳим оператордир. У шартга боғлиқ равишда код фрагментини бажаришга мўлжалланган. *if* операторининг структурасини қуйидагича ифодалаш мумкин:

if (ифода) *бажариладиган_блок*

Бу ерда ифода *PHP* дастурлаш тилидаги ихтиёрий тўғри ифодадир (яъни бирор қийматга эга). Скриптни қайта ишлаш жараёнида ифода мантиқий типга ўзлаштирилади. Агар натижада қайта ишланган ифода қиймати рост (True) бўлса, у ҳолда *бажариладиган_блок* бажарилади. Акс ҳолда *бажариладиган_блок* йўқотилади. Агарда *бажариладиган_блок* бир нечта буйруқлардан иборат бўлса, у ҳолда улар фигурали қавсларга { } олиниши шарт.

Ифодани мантиқий типга ўзлаштириш қоидаси:

1. `FALSE` қиймати қуйидаги қийматларга тўғри келади:
   - мантиқий `False`
   - бутун сон – ноль (`0`)
   - ҳақиқий сон – ноль (`0.0`)
   - бўш сатр ва `"0" сатр;`
   - элементларсиз массив
   - ўзгарувчиларсиз объект (объектлар ҳақида кейинги бўлимларда тўлиқ маълумот берилган)
   - махсус тип `NULL бўлганда`
2. Қолган барча қийматларда `TRUE қийматга ўзлаштирилади`.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($names[0]=="Карим") {
   echo "Салом, Азамат!";
   $num = 1;
   $account = 2000;
}
if ($num) echo "Карим рўйхатда биринчи!";
$bax = 30;
if ($account > 100*$bax+3)
   echo "Бу сатр экранга чиқмайди, чунки
   шарт бажарилмайди";
?>
```

Мисол: if шарт оператори.

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 (ифода) *бажариладиган_блок*
else *бажариладиган_блок1*

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

Юқоридаги мисолни бажарилмайдиган шарт ҳолатида қандай кўриниш олишини кўриб чиқайлик.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($names[0]=="Карим") {
   echo "Салом, Азамат!";
   $num = 1;
   $account = 2000;
} else {
   echo "Салом, $names[0].
  Биз Азаматни кутгандик :(";
}
if ($num) echo " Карим рўйхатда биринчи!";
else echo " Карим рўйхатда биринчимасми?!";
$bax = 30;
if ($account > 100*$bax+3)
   echo " Бу сатр экранга чиқмайди, чунки
   шарт бажарилмайди ";
   else echo "Шундай бўлса ҳам экранга чиқди!";
?>
```

Мисол: else оператори.

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 (ифода) *бажариладиган_блок*
elseif (ифода1) *бажариладиган_блок1*
.....
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.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($names[0]=="Карим") {
   // массивда биринчи элемент Карим бўлса
   echo "Салом, Карим!";
}elseif ($names[0] == "Салим"){
   // массивда биринчи элемент
   // Карим эмас Салим бўлса
   echo "Салом, Салим!";
}elseif ($names[0] == "Содиқ"){
   // массивда биринчи элемент
   // ҳам Карим, ҳам Салим эмас Содиқ бўлса
   echo "Салом, Содиқ!";
}else {
```

```
    // массивда биринчи элемент
    // на Карим, на Салим, на Содиқ бўлса
    echo "Салом, $names[0]. Сен кимсан?";
}
?>
```

Мисол: elseif оператори.

2. 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.

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

Мисол: Альтернатив синтаксисдан фойдаланиш.

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;
?>
```

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.

мисолни *switch* операторидан фойдаланган ҳолда қуйидагича ёзиш мумкин:

```
<?
$names = array("Карим","Салим","Содиқ");
switch ($names[0]){
case "Карим":
   echo "Салом, Карим!";
break;
case "Салим":
   echo "Салом, Салим!";
break;
case "Содиқ":
   echo "Салом, Содиқ!";
break;
default:
   echo "Салом, $names[0].
  Сени исминг нима?";
}
?>
```

Агарда берилган мисолда *break* операторини ташлаб кетсак, масалан, *case* "Салим": ҳоли учун, у ҳолда агарда ўзгарувчи сатр қиймати "Салим" бўлса экранга "Салом, Салим!" маълумотини чиқаради ва ишини давом эттириб "Салом, Содиқ!" маълумотни чиқаради ва *switch* операторининг охирига келиб дастур *break* операторини бажаради.

switch операторининг конструкцияси 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.

3. Цикллар

PHP дастурлаш 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.

```
<?
//Бу дастур барча жуфт сонларни экранга чиқаради.
   $i = 1;
   while ($i < 10) {
      if ($i % 2 == 0) print $i;
   // агар у жуфт бўлса экранга чиқаради.
      $i++;
   // $i ўзгарувчи биттага оширилади
   }
?>
```

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.

```
<?
// бу дастур шарт бажарилмаса ҳам
// 12 рақамини экранга чиқаради.
$i = 12;
   do{
      if ($i % 2 == 0) print $i;
   // агар сон жуфт бўлса уни экранга чиқарамиз
      $i++;
   // сонни биттага оширамиз
   }while ($i<10)
?>
```

For

Бу *PHP* дастурлаш тилидаги энг мураккаб циклдир. Улар С дастурлаш тилидаги цикларни эслатади.

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 (бу ҳолда *PHP* дастурлаш тили бу ифодани ҳар доим рост деб ҳисоблайди) бажарилишини билдиради. Бу унчалик бефойда эмас, чунки циклни *break* оператори ёрдамида тўхтатса бўлади.

Масалан, барча жуфт сонларни *for* цикли ёрдамида қуйидагича экранга чиқариш мумкин:

```
<?php
for ($i=0; $i<10; $i++){
   if ($i % 2 == 0) print $i;
```

```
   // жуфт сонларни экранга чиқарамиз
}
?>
```

Агарда иккинчи ифодани ($i<10 шартни) ташлаб кетсак, бу масаладаги циклни ҳам *break* оператори ёрдамида тўхтатса бўлади.

```
<?php
for ($i=0; ; $i++){
   if ($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.
}
?>
```

Барча учала ифодани ҳам тушириб қолдириш мумкин. Бу ҳолда счётчик $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:

```
<?php
$i=0; // счётчикни бошланғич қийматини берамиз
for ( ; ; )
{
   if ($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:

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

Foreach

Яна битта фойдали конструкция. У фақат *PHP*4 дастурлаш тилида учрайди ва массивлар билан ишлашга мўлжалланган.

Синтаксиси қуйидагича:

foreach ($array as $value) { *бажариладиган_блок* }

ёки

foreach ($array as $key => $value)

{ *бажариладиган_блок* }

Биринчи ҳолда берилган $array ўзгарувчи массивнинг элементлари формаллаштирилади. Ҳар бир цикл қадамида массивнинг жорий элементи қиймати $value ўзгарувчига ўзлаштирилади ва массивнинг ички ҳисоблагичи биттага ортади (чунки кейинги қадамда массивнинг кейинги элементи керак бўлади). *бажариладиган_блок* ичидаги

массивнинг жорий элементи қиймати $value ўзгарувчи ёрдамида қийматга эга бўлади. *бажариладиган_блок* $array массивнинг элементлари нечта бўлса шунча марта бажарилади.

Юқорида келтирилган иккинчи тўлдирилган шаклда ҳар бир цикл қадамида массивнинг жорий элементи калити *бажариладиган_блок*да қўлланса ҳам бўладиган $key ўзгарувчига ёзиб борилади.

foreach цикли ишини бошлаганда массивнинг ички кўрсатгичи автоматик равишда биринчи элементни кўрсатади.

```
<?php
$names = array("Карим","Салим","Содиқ");
foreach ($names as $val) {
   echo "Салом, $val <br>";
   // барча саломлашишларни экранга чиқарамиз
}
foreach ($names as $k => $val) {
   // саломлашишдан ташқари
   // рўйхатдаги рақамини, яъни калитини экранга чиқарамиз
   echo "Салом, $val !
    Сен рўйхатда $k – рақамдасан.<br>";
}
?>
```

Мисол: foreach оператори.

4. 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, foreach, while, do while* ёки *switch break* структурани бошқарувчи, тугаллаш кераклигини билдирувчи, унинг таркибига кирувчи рақамли аргумент билан қўлланилади.

```
<?php
$i=1;
while ($i)
{
   $n = rand(1,10);
   // 1 дан 10 гача исталган сонни умумийлаштирамиз
   echo "$i:$n ";
  // итерация рақамини чиқарамиз ва умумийлаштирилган сон
   if ($n==5) break;
/* Агар умумийлаштирилган сон 5 бўлса, цикл ишини тўхтатамиз. Бу ҳолда
бу қатордан кейин цикл ичида нима мавжуд бўлса, амалга оширилмайди */
   echo "Цикл ишламоқда <br>";
   $i++;
}
echo "<br> итерация цикли сони $i ";
?>
```

Мисол: Break оператори

Бу скрипт ишининг натижаси тахминан қуйидагича:----------

```
1:7 Цикл ишлаяпти
2:2 Цикл ишлаяпти
3:5
Цикл итерацияси сони
```

Агар *break* операторидан сўнг сон кўрсатилса, бу цикл операторларидан таркиб топган айнан шундай миқдор бузилади. Модомики, юқорида келтирилган мисолда циклдан фойдаланилмаган экан, бу унчалик тўғри эмас. Скриптимизни бироз ўзгартирамиз:

```
<?php
$i=1;
while ($i) {
  $n = rand(1,10);
  // Исталган сонни умумлаштирамиз
  // 1 дан 10 гача
  switch ($n){
    case 5:
      echo "<font color=blue>
        switch дан чиқиш (n=$n)</font>";
    break 1;
   // switch  ишини тўхтатамиз
    // (breakмавжуд биринчи циклни)
    case 10:
      echo "<font color=red>
       switch дан чиқиш ва
      while (n=$n)</font>";
    break 2;
   // switch ишини тўхтатамиз ва while
    // (иккита break мавжуд цикл)
    default:
      echo "switch ишлаяпти (n=$n), ";
}
echo " while ишлаяпти –$i <br>" амал;
$i++;
}
echo "<br> цикл итерацияси сони $i ";
?>
```

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.

```
<?php
$i=1;
while ($i<4) {
   $n = rand(1,10);
  // исталган сонни умумлаштирамиз
  // 1 дан 10 гача
   echo "$i:$n ";
  // интерация рақамини чиқарамиз ва
   // умумлаштирилган сон
   if ($n==5) {
      echo "Янги интерация ";
      continue;
```

```
/*Агар умумлаштирилган сон 5 бўлса, янги цикл интерациясини бошлаймиз,
$i катталашмайди */
    }
    echo "Цикл ишлаяпти <br>";
    $i++;
}
echo "<br>цикл интерацияси сони $i ";
?>
```

Бу скрипт ишининг натижаси қуйидагича

```
1:10 Цикл ишлаяпти
2:5 Янги итерация
2:1 Цикл ишлаяпти
3:1 Цикл ишлаяпти
4 цикли интерацияси сони
```

continue оператори амалга ошгандан сўнг цикл иши тугалланмаганини ҳисобга оламиз. Мисолда цикл ҳисоблагичи 5 сони олингандан тақдирда у *continue* операторидан кейин бўлса ўзгармайди. Аслида *continue* ёрдамида 5 сони умумлаштирилганда бу ҳолатдан четлашамиз. Шунинг учун *continue* операторини ифода ҳақиқийлигини текширишга алмаштириб ёзиб қўйиш кифоя:

```
<?php
$i=1;
while ($i<4) {
    $n = rand(1,10);
   // исталган сонни умумлаштирамиз
    //  1 дан 10 гача
    if ($n!==5) {
        echo "$i:$n <br>";
   // интерация рақамини чиқарамиз
    // ва умумлаштирилган сон
    $i++;
    }
}
?>
```

PHP да *continue* дан фойдаланишнинг бир хусусияти мавжуд – *switch* конструкциясида у худди *break* каби ишлайди. Агар *switch* цикл ичида жойлашган бўлса ва янги цикл интерациясини бошлаш керак бўлса, continue 2 дан фойдаланиш лозим бўлади.

15-Маъруза

Мавзу: PHP ДА ФУНКЦИЯЛАР. СИНФЛАР ВА ОБЪЕКТЛАР.

Маъруза режаси:

1. PHP да функциялар
2. Функцияларнинг аргументлари
3. Ўзгарувчан узунлик аргументлари рўйхатлари
4. Функциялар ичида ўзгарувчилардан фойдаланиш
5. Статистик ўзгарувчилар
6. Қайтарилувчан маънолар
7. Ҳаволани қайтариш
8. Функциянинг ўзгарувчилари
9. Ички жойлашган (ичма-ич) функциялар.
10. Синфлар ва объектлар.
11. Ўзгарувчиларни инициаллаштириш

12. Объектлар

Таянч иборалар: *function, func_num_args(), func_get_arg(), func_get_args(),echo(), print(), date(), include(), функция, ўзгарувчиб ҳавола, ичма-ич функция, синф, объект*

1. PHP да 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.

Мисол: Натурал сон факториалини ҳисоблаш функцияси

```
<?php
function fact($n)
{
  if ($n==0) return 1;
  else return $fact = $n * fact($n-1);
}
echo fact(3);
   // echo (3*2) деб ёзиш мумкин эди; лекин сон катта бўлса,
echo fact(50);
   // echo (50*49*48*...*3*2) деб ёзгандан функциядан фойдаланиш қулайроқ
  ?>
```

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.

Функция чақириш қандай амалга ошади? Функция номи ва юмалоқ қавслар ичида унинг параметрлари маънолари рўйхати кўрсатилади, агар шундайлари мавжуд бўлса:

```
<?php
  Функция_номи
  ("1-параметр_учун_маъно",
  "2-параметр_учун_маъно ",...);
// Функцияни чақиришга мисол – функцияни чақириш
// факториални ҳисоблаш юқорида бор,
// 3 сони факториалини ҳисоблаш учун у ерда биз fact(3) деб ёзганмиз;
// у ерда fact – чақирилувчи функция номи,
// а 3 –$n номли унинг параметри маъноси
?>
```

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
?>
```

Ушбу 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
   }
}
?>
```

Функция аргументлари, уларнинг маънолари ва ишлатилишини батафсил кўриб чиқамиз.

2. 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.

```
<?php
// қўшимча қилиши мумкин бўлган функцияни ёзамиз checked сўзи қаторига
function add_label(&$data_str){
  $data_str .= "checked";
}
$str = "<input type=radio name=article ";
  // бундай қатор мавжуд бўлсин
echo $str .="><br>";
    // форма элементини келтиради – белгиланмаган радио кнопкасини
add_label($str);
    // функцияни чақирамиз
echo $str .="><br>";
    // бу энди белгиланган радио кнопкани келтиради
?>
```

Ушбу мисол. Аргументларни ҳаволасига кўра ўтказиш

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.

```
<?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
?>
```

Ушбу мисол. Тинч ҳолатдаги аргумент маъноси

Бу скрипт ишининг **натижа**си 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 **натижа**си 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

3. 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();
   // "0 функция аргументлари сони" қаторни келтиради
DataCheck(1,2,3);
   // "3-функция аргументлари сони" қаторни келтиради
?>
```

Ушбу мисол. func_num_args() функциясидан фойдаланиш

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.

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
   /*    функцияга ўзгарган аргументлар сонини текширамиз, биринчи ўзгарган
аргумент бутун сонми-йўқми */
  if ($n>=1) if (!is_int(func_get_arg(0)))
      $check = false;
```

```
  /* текширамиз, иккинчи ўзгарган аргумент қаторми-йўқми */
  if ($n>=2)
     if (!is_string(func_get_arg(1)))
        $check = false;
return $check;
}
if (DataCheck(123,"text"))
   echo "Текширув тўғри ўтди<br>";
else echo "маълумотлар шартларни қондирмайди <br>";
if (DataCheck(324))
   echo "Текширув тўғри ўтди<br>";
else echo "маълумотлар шартларни қондирмайди <br>";
?>
```

Ушбу мисол. Маълумотлар типини, унинг аргументларини текшириш
Ишнинг **натижа**си қуйидагича бўлади.----------

Маълумотлар шартларни қондирмайди. Текширув тўғри ўтди.

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:

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

Аён бўлдики, *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>";
?>
```

4. 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();
?>
```

мисол. Глобал ўзгарувчилар

Бу скрипт ишидан қуйидаги **натижа**ларни 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.

5. 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.

```
<?
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
?>
```

мисол. Статистик ўзгарувчилардан фойдаланиш

6. 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.

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

мисол. *list()* дан фойдаланиш

7. 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.

```
<?
$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
?>
```

мисол. Ҳаволани қайтариш

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.

8. 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.

9. 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 компоненталари керак. Охирги вақтларда бу функциялардан фойдаланиш учун қўшимча компоненталар керак эмас, чунки уларнинг барчаси ҳозирда *PHP* дастурлаш тили ядросига киритилган.

10. 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.

11. 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, *конструктор*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.

12. 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

```
$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
?>
```

Мисол: Хусусиятлар маъносини ўрнатиш

Синфни тузиб, бу синфнинг объекти қандай номга эга бўлишини била олмаймиз, қолаверса объектлар жуда кўп бўлиши ва уларнинг барчаси ҳар хил номга эга бўлиши мумкин. Синфни юзага чиқариш ичида объектга қандай муносабатда бўлишни билмаймиз. Синф юзага чиқиши ичида функциялар ва ўзгарувчиларга эркин кириш учун, $thid ўриндош ўзгарувчисидан фойдаланиш керак. Масалан, $this->title шундай синф объектининг title ини қайтаради. Баъзан бу ўзгарувчини “менинг хусусий мулким” (хусусиятга муносабат тариқасида) деб ўқишни таклиф қилинади.

16-Маъруза

Мавзу: PHP ДА МАЪЛУМОТЛАР БАЗАЛАРИ БИЛАН ИШЛАШ. MYSQL МАЛУМОТЛАР БАЗАСИ

Маъруза режаси:

1. PHP да маълумотлар базалари билан ишлаш
2. Маълумотни қўшиш учун интерфейс тузиш
3. Алоқа ўрнатиш
4. Маълумотлар базаларини танлаш
5. Жадвал майдонлари рўйхатини олиш
6. HTML-формада майдонлар рўйхатининг акс этиши
7. Маълумотлар базасига маълумотлар ёзиш

Таянч иборалар: *MySql, Sql-сўров, МББТ*

1. PHP да маълумотлар базалари билан ишлаш

Ушбу бўлим PHP ва MySql МББТ (Маълумотлар базасини бошқариш тизими) 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.

2. 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-формада майдонларни тасвирлаш.

Бундан кейин формага киритилган маълумотларни маълумотлар базасига киритиш керак бўлади.

Бу топшириқларнинг барчасини тартиб билан кўриб чиқамиз.

3. 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.

4. 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");
?>
```

5. 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 кирадиган 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:
```

6. 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.

Шундай қилиб, маълумотларни Artifacts жадвалига киритиш учун форма тузамиз:

```
<?
$conn=mysql_connect("localhost","nina","123");
        // алоқа ўрнатамиз
$database = "book";
```

```
$table_name = "Artifacts";
mysql_select_db($database); // иш учун
        // маълумотлар базаси танлаймиз
$list_f = mysql_list_fields($database,$table_name);
 // базадан майдонлар рўйхатини оламиз
$n = mysql_num_fields($list_f); // аввалги сўровдаги натижадаги қаторлар сони (Artifacts жадвалида жаъми қанча майдонлар бор)
echo "<form method=post action=insert.php>";
// маълумотларни киритиш учун форма тузамиз
echo " <TABLE BORDER=0 CELLSPACING=0 width=50% ><tr>
   <TD  BGCOLOR='#005533' align=center><font color='#FFFFFF'>
   <b> Add new row in $table_name</b></font></td></tr><tr><td></td></tr></TABLE>";
echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";
// ҳар бир майдон учун унинг номи, тури, узунлиги ва байроғини оламиз
for($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);
   // байроқлар қаторидан массив қиламиз,
   // унда массивнинг ҳар бир элементи – майдон байроғи
   $flags = explode(" ", $flags_str);
   foreach ($flags as $f){
      if ($f == 'auto_increment') $key = $name_f;
         // автоинкремент номини эслатиб ўтамиз
   }
/* автоинкремент ҳисобланган ҳар бир майдон учун, унинг туридан келиб чиқиб мос келувчи форма элементини чиқарамиз*/
if ($key <> $name_f){
echo "<tr><td align=right bgcolor='#C2E3B6'><font size=2>
      <b> ". $name_f ."</b></font></td>";
switch ($type){
      case "string":
         $w = $len/5;
         echo "<td><input type=text name=\"$name_f\"
            size = $w ></td>";
      break;
      case "int":
         $w =  $len/4;
         echo "<td><input type=text name=\"$name_f\"
            size = $w ></td>";
      break;
      case "blob":
         echo "<td><textarea rows=6 cols=60 name=\"$name_f\"></textarea></td>";
      break;
      }
   }
   echo "</tr>";
}
echo "</table>";
echo "<input type=submit name='add' value='Add'>";
echo "</form>";
```

?>

мисол. Artifacts жадвалига маълумот киритиш учун форма

7. Маълумотлар базасига маълумотлар ёзиш

Шундай қилиб форма тузилди. Энди энг асосийсини бажариш қолди – бу формадаги маълумотларни бизнинг маълумотлар базасига жўнатиш. Маълумки, маълумотларни жадвалга ёзиш учун SQL тилидаги INSERT буйруғи ишлатилади. Масалан:

```
mysql> INSERT INTO Artifacts
       SET title='Камолов';
```

PHP скриптда бундай буйруқдан (ёки SQL даги исталган буйруқдан) қандай фойдаланилади, деган савол туғилади. Бунинг учун *mysql_query()* функцияси мавжуд.

mysql_query синтаксиси

```
mysql_query ресурси (query қатори
   [, ресурс link_identifier])
```

mysql_query() SQL-сўровни MySQL маълумотлар базасининг link_identifier кўрсаткичи ёрдамида аниқланадиган актив маълумотлар базасига жўнатади (бу *MySQL* сервери билан бирон-бир алоқага ҳавола). Агар link_identifier параметри ўтказиб юборилган бўлса, сўнгги очиқ алоқа ишлатилади. Агар очиқ алоқа бўлмаса, функция параметрсиз *mysql_connect()* функциясига ўхшаш ҳолда МББТ (СУБД) билан боғланишга уринади. Сўров натижаси буферланади.

Адабиётлар рўйхати

5. Матросов С.Ч. и др. HTML 4.0 в подлиннике. BHV-СПб, 2000. 672 с.
6. Уилтон П. JAVASCRIPT. Основы. Символ-плюс. 2002. 1056 с.
7. Кингли-Хью Э., Кингли-Хью К. JAVASCRIPT 1.5: Учебный курс. Питер. 1-е издание. 2002.
8. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQL. - К.: "ДиаСофт", 2001.
9. Спейнауэр С., Куэрсиа В. Справочник Web-мастера. - К: "BHV", 1997. - 368 с.
10. Бранденбау. JAVASCRIPT. Сборник рецептов для профессионалов. СПб. 2001.
11. Ратшиллер Т., Геркен Т. PHP4: разработка Web-приложений. Питер, 2001. - 384 с.
12. Яргер Р., Риз Дж., Кинг Т. MySQL и mSQL. Базы данных для небольших предприятий и Интернета. - СПб: Символ-Плюс, 2000 - 560 с.
13. Хилайер С., Мизик Д. Программирование Active Server Pages. - М: "Русская редакция", 1999.
14. Холзнер С. Perl: специальный справочник. - СПб: "Питер". 2000. - 496 с.
15. Шварц Р., Кристиансен Т. Изучаем Perl. - К: "BHV", 2000. - 320 с.
16. Мингликулов З.Б., Агзамходжаева М.Р. “WEB-дастурлаш” фани маъруза матнлари /ТАТУ. Тошкент 2009. -125 с.

№	Маъруза мавзулари	Саҳифа
1.	**1-Мавзу: Web-дастурлаш фанига кириш** 1. Web дастурлаш фани ва унинг вазифалари 2. Web дастурлашда фойдаланиладиган асосий дастурлар тўғрисида умумий маълумотлар 3. HTML белгилашлари: HTML, XML, XHTML, WML. 4. Сценарийли тиллар. "Клиент-сервер" технологияси	
2.	**2-Мавзу: HTML га кириш. HTML нинг асосий теглари** 1. HTML хужжатининг тузилиши. 2. HTML асосий теглари.	
3.	**3-Мавзу: HTML да формалар ва фреймлар** 1. HTML да формалар 2. HTML да фреймлар	
4.	**4-Мавзу: Клиент томонидан дастурлаш. JavaScript га кириш. JavaScript ни HTML-хужжатга жойлаштириш.** 1. JavaScript га кириш 2. JavaScript нинг объектли модели тушунчаси 3. JavaScript ни HTML-хужжатига жойлаштириш. 4. JavaScript нинг URL-схемаси 5. Синфлар иерархияси	
5.	**5-Мавзу: JavaScript да ўзгарувчилар, маълумот типлари, ифодалар ва арифметик амаллар. JavaScript да функция тушунчаси** 1. JavaScript да ўзгарувчилар 2. JavaScript да маълумотлар типи 3. JavaScript тили операторлари 4. JavaScript тилида функция	
6.	**6-Мавзу: Дастур мантиқини бошқариш элементлари ва ифодалари** 1. For элементи 2. Break элементи 3. Continue элементи 4. For..in элементи 5. If..else элементи. 6. Function элементи 7. New элементи. 8. Return ифодаси. 9. Var ифодаси. 10. This ифодаси. 11. While ифодаси. 12. With ифодаси.	
7.	**7-Мавзу: JavaScript нинг уч турдаги объектлари. JavaScript объекти Хусусиятлари ва усуллари** 1. JavaScript нинг уч турдаги объектлари 2. JavaScript объекти хусусияти ва усуллари	
8.	**8-Мавзу: Объектларни яратиш ва уларга мурожат қилиш** 1. Объектларни яратиш 2. Янги объектларни яратиш 3. Усулларни аниқлаш 4. This калит сўзидан фойдаланиб объектга мурожат қилиш 5. Document объекти ва унинг усуллари	
9.	**9-Мавзу: JavaScript нинг стандарт объектлари** 1. Array объекти. Кўп ўлчамли массивларни ташкил қилиш 2. Boolean объекти 3. Date объекти 4. Function объекти 5. Math объекти 6. Number объекти	

	7. String объекти	
10.	**10-Мавзу: Браузер ойнаси хусусиятини дастурлаш** 1. Статус майдони (статус қатори, status bar) 2. Location майдони 3. Браузер тури (Navigator объекти) 4. Графикани дастурлаш	
11.	**11-JavaScript нинг асосий стандарт функциялари** 1. eval – функцияси. 2. ParseInt ва parseFloat функциялари. 3. Java Script функциялари ва усуллари. 4. Формаларни ишлаш.	
12.	**12-Мавзу: PHP - Сервер томондан дастурлаш. PHP га кириш. PHP ни ўрнатиш ва тестлаш.** 1. PHP тарихи 2. PHP – веб техногия тил 3. PHP имкониятлари 4. Дастурий воситани созлаш ва ўрнатиш 5. Денвер дистрибутиви.	
13.	**13-Мавзу: PHP структураси. Маълумотлар типлари. Ўзгарувчилари. Ифодалар. Жараёнларни бошқариш.** 1. PHP скриптлар 2. Асосий синтаксислар 3. Ўзгарувчилар, ўзгармаслар ва амаллар 4. Маълумотлар типлари 5. Шарт операторлари ва алтернатив синтаксислари 6. Цикллар ва бошқарув ўтказувчи операторлар	
14.	**14-Мавзу: PHP да Жараёнларни бошқариш** 1. Шарт операторлари 2. Алтернатив синтаксислари 3. Цикллар 4. Бошқарув ўтказувчи операторлар	
15.	**15-Мавзу: PHP да функциялар. Синфлар ва объектлар.** 1. PHP да функциялар 2. Функцияларнинг аргументлари 3. Ўзгарувчан узунлик аргументлари рўйхатлари 4. Функциялар ичида ўзгарувчилардан фойдаланиш 5. Статистик ўзгарувчилар 6. Қайтарилувчан маънолар 7. Ҳаволани қайтариш 8. Функциянинг ўзгарувчилари 9. Ички жойлашган (ичма-ич) функциялар. 10. Синфлар ва объектлар. 11. Ўзгарувчиларни инициаллаштириш 12. Объектлар	
16.	**16-Мавзу: PHP да маълумотлар базалари билан ишлаш. MySQL малумотлар базаси** 1. PHP да маълумотлар базалари билан ишлаш 2. Маълумотни қўшиш учун интерфейс тузиш 3. Алоқа ўрнатиш 4. Маълумотлар базаларини танлаш 5. Жадвал майдонлари рўйхатини олиш 6. HTML-формада майдонлар рўйхатининг акс этиши 7. Маълумотлар базасига маълумотлар ёзиш	