ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АХБОРОТ ТЕХНОЛОГИЯЛАРИ ВА КОММУНИКАЦИЯЛАРИНИ РИВОЖЛАНТИРИШ ВАЗИРЛИГИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

Компьютер инжиниринги факультети

«Мультимедиа технологиялари» кафедраси

«ВЕБ ДАСТУРЛАШ»

фанидан амалий машғулотларини бажариш бўйича услубий қўлланма

Ташкент 2015

МУНДАРИЖА:

1-АМАЛИЙ МАШҒУЛОТ

МАВЗУ: HTML АСОСЛАРИ

Ишнинг мақсади: HTML асосларини ўрганиш, унинг сарлавҳа ва тана қисмига тегишли асосий теглар билан танишиш.

Ишнинг натижасида талаба қуйидагиларга эга бўлиши керак:

Билиши керак: гиперматнли хужжат ташкил этиш асослари, манбаларни, HTML-хужжат ҳақида тушунчаларни;

Қила олиши керак: HTML-хужжат ярата олиш.

НАЗАРИЙ ҚИСМ

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 TITLE лардир.

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

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

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

Ҳужжатнинг HEAD бўлими.

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 бараузер 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 браузерлар 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>

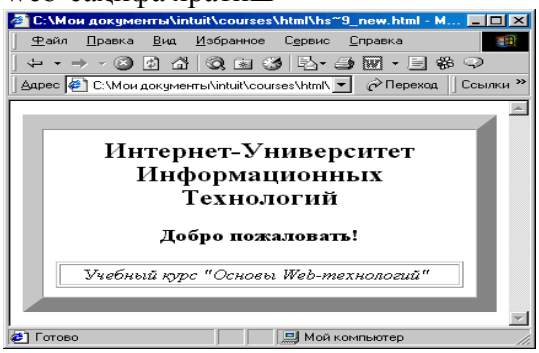
Агар BGCOLOR параметри рангни номи ёки унинг таркибий кисмларини ўн олтилик саноқ тизимидаги кодда келтириш вазифаси ёрдамида фон ранггини чиқариш учун ишлатилса, BACKGROUND тасвир ёрдамида саҳифага фон беришда фойдаланилади. Тасвир сифатида GIF ёки JPG форматидаги график файллар ишлатилади. HTML-ҳужжат 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:
<BODY>

```
<H1 ALIGN=center>Таблица</H1>
<CENTER>
<TABLE BORDER>
<TR> <TD COLSPAN=3>Агарда таблицада иккита TR теги эълон қилинса, у ҳолда иккита қатор ҳосил бўлади.</TD> </TR>
<TR> <TD>Агарда қаторда 2 та бўлса,</TD>
<TD>у ҳолда унда</TD>
<TD>3 та устун пайдо булади</TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

ТОПШИРИҚ

1. қуйидаги кўринишдаги web-саҳифа яратиш



2. Турли ўлчамдаги шрифтлардан фойдаланган ҳолда матн ёзинг. Матн иккала чеккаси бўйича текисланган бўлиши керак.
3. 2 мисолдаги масалани қуйидаги тартибда ёзинг: биринчи сатр чап томонга текислансин, иккинчи сатр ўнг томонга ва ҳ.к.
4. Ўнта рўйхатдан иборат бўлган гиперссилкалар тузинг. Ссилкалар Интернетнинг турли адресларига мурожаат қилсин, жумладан қидирув тизимига.
5. Ўзингизнинг почта қутингизга ссылка яратинг.
6. 2 вариантдаги мисолни ҳар бир сўзини қандайдир хужжатга мурожаат қилувчи гиперссилкалардан ташкил этинг.
7. ихтиёрий расм олинг ва html-хужжат яратинг, расим устида матн мавжуд бўлсин.
8. Учта элементдан иборат бетартиб рўйхат ташкил қилинг. Биринчи элемент айлана билан, иккинчиси квадрат учунчиси доира коринишидаги рўйхатлар бўлсин.
9. Қуйидаги жадвални ҳосил қилувчи HTML-код ёзинг.

<table>
<tr><td colspan="2" rowspan="2">1_1</td><td>1_2</td><td>1_3</td><td colspan="2" rowspan="2">1_4</td></tr>
<tr><td>2_1</td><td>2_2</td></tr>
<tr><td>3_1</td><td>3_2</td><td colspan="2" rowspan="2">3_3</td><td>3_4</td><td>3_5</td></tr>
<tr><td>4_1</td><td>4_2</td><td>4_3</td><td>4_4</td></tr>
<tr><td colspan="2" rowspan="2">5_1</td><td>5_2</td><td>5_3</td><td colspan="2" rowspan="2">5_4</td></tr>
<tr><td>6_1</td><td>6_2</td></tr>
</table>

10. HTML нинг асосий тэгларидан фойдаланган ҳолда WEB саҳифа ташкил этинг.

2-амалий машғулот

Мавзу: HTML-хужжат ташкил қилишда объектлар, форма ва фреймлардан фойдаланиш.

Ишнинг мақсади: HTML-хужжат ташкил қилишда объектлар, форма ва фреймлардан фойдаланиш.

Ишнинг натижасида талаба қуйидагиларга эга бўлиши керак:

Билиши керак: объект асослари, форма ва фреймларни;

Қила олиши керак: HTML-хужжат ярата олиш.

НАЗАРИЙ ҚИСМ

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.

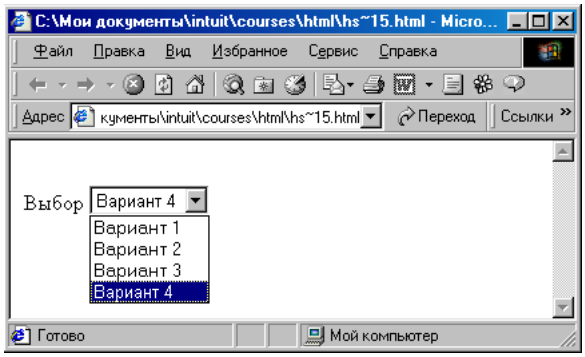
NAME атрибути
Объект номини ифодалайди.

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>



Расм 6.1. Қалқиб чиқувчи меню

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.

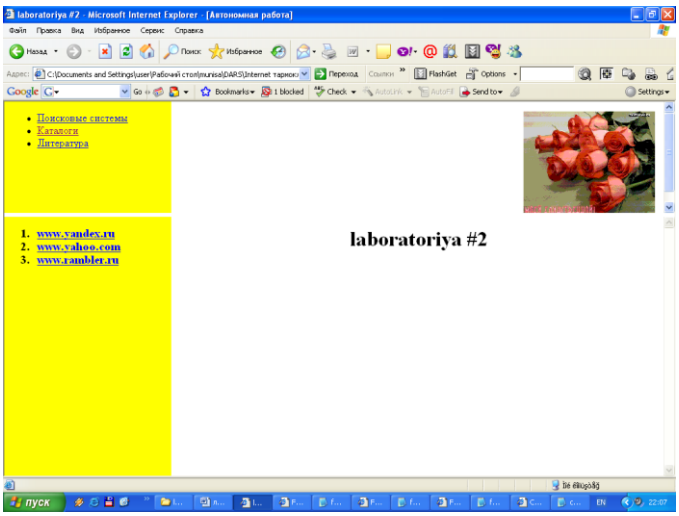
Фреймлар орасидаги ўзаро таъсир

Фреймлар билан ишлаётганда фойдаланувчи учун қўлай бўлган ҳужжат юклаш схемасини яратиш мумкин. Фреймлар орасидаги ўзаро алоқа ҳужжатларни бошқа фреймдаги буйруқлар ёрдамида айнан танланган фреймга юклаш имконини беришидадир. Бу мақсадда <A> тэгининг TARGET атрибутидан фойдаланилади. TARGET атрибути ушбу ссылка кўрсатаётган ҳужжат юкланувчи фрейм ёки браузер ойнаси номини белгилайди. Ўзгартирилмаган ҳолда ушбу параметр йўқ бўлганда ҳужжат жорий фрейм ёки ойнада юкланади. Фрейм номи сифатида мавжуд дарча ёки фрейм номи берилиши ёки бўлмаса янги ойна очиш учун янги ном берилиши мумкин. 4 та захирадаги номлар бор. Улар:
_blank, _self, _top, _parent. Булардан ташқари “_” белгиси билан бошланувчи хар қандай номдан фойдаланиш мақсадга мувофиқ эмас.
TARGET=“_blank” – ҳужжатнинг янги ойнага юкланишини таъминлайди. Бу ойна номга эга бўлмаслиги туфайли унга бошқа ҳужжатни юклашнинг иложи бўлмайди.
TARGET=“_self” дан фойдаланилганда ҳужжат жорий фреймга юкланади.
TARGET=“_top” ҳужжатнинг бутун дарчага юкланишига сабаб бўлади.
TARGET=“_parent” ҳужжатнинг жорий фреймнинг фрейм-ота-онаси томонидан эгалланган сохасига юкланишига олиб келади.

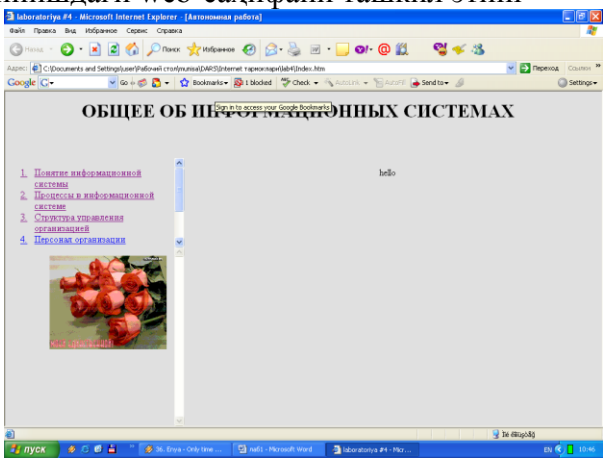
Таркиби “А” фреймига юкланган frame_a.html файлининг биттагина test.html файлига TARGET параметрининг турли қийматларига эга 6 та ссылкаси мавжуд.

ТОПШИРИҚЛАР

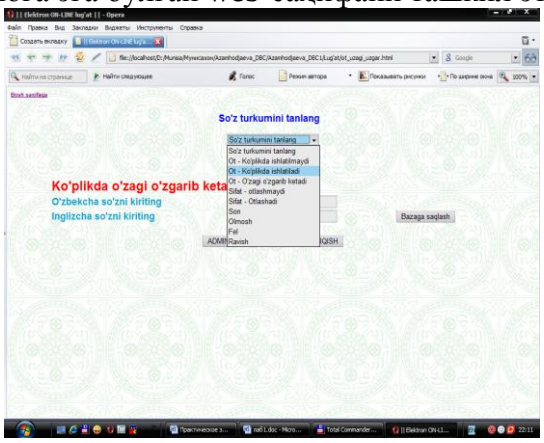
1. Қуйидаги кўринишдаги web-саҳифани ташкил этинг



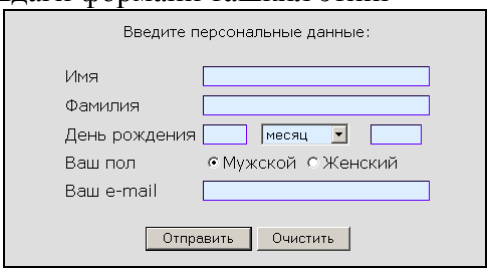
2. Қуйидаги кўринишдаги web-саҳифани ташкил этинг



3. Қуйидагича менюга эга бўлган web-саҳифани ташкил этинг



4. Қуйидаги кўринишдаги формани ташкил этинг



Амалий иш - 3

Мавзу: JavaScript. Клиент томонида дастурлаш. JavaScript ни HTML-ҳужжатларга жойлаштириш. JavaScript да маълумотлар типлари, ўзгарувчилар, ифодалар ва арифметик ифодалардан фойдаланиш.

Ишнинг мақсади: JavaScript ни ўрганиш, клиент томонидаги дастурлар яратиш, JavaScript ни HTML-ҳужжатга жойлаштириш, JavaScript да маълумотлар типларидан фойдаланиш, ўзгарувчилар, амаллар ва арифметик операторлар билан ишлаш.

Ишнинг натижасида талаба қуйидагиларга эга бўлиши керак:

Билиши керак: JavaScript нинг асосий катталиклари;

Қила олиши керак: HTML-ҳужжатда JavaScript ёрдамида клиент томонидаги сценарийлар яратиш.

НАЗАРИЙ ҚИСМ

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

Ўзгарувчилар эълони

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

```
var MyHelloMsg;
```

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

```
MyMsg = "Салом!";
```

MyMsg ўзгарувчи номи ўзлаштирилгандан сўнг рухсат берилади.

Ўзгарувчи номини танлаганда, ыуйидаги оддий ыоидаларни ушлаб ыщйиш керак:

- Ўзгарувчи номи харфлардан ёки "_", "$" белгилардан бошланиш керак ва фақат харфлардан, сонлардан ва "_", "$" белгилардан иборат бўлиши керак;
- Ўзгарувчилар номи JavaScript нинг захираланган калит сўзлари билар мос келмаслиги керак.

Қуйида JavaScript нинг захираланган калит сўзлар келтирилган:

```
break    case      catch    class     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.

JavaScript да маълумотлар типи

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

Сонлар

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

```
25         10 асосидаги бутун сон
0137       8 асосидаги бутун сон
0xFF       16 асосидаги бутун сон
386.7      Сузувчи ўнли нуқтали сон
25e5
или 25E5   Илмий нотациядаги сон, 2500000 га тенг.
```

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

Матнли қатор

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

```
"Hello, world!"
""
```

```
"12345"
'Бу матнли қатор'
```
"" қатори –бўшдир. Қуйидаги 2 та ўзлаштириш эквивалент эмаслигини аниқлаймиз:
```
MyStr=""
MyStr1=null
```
Биринчи холда MyStr ўзгарувчисида матнли қатор сақланмоқда (бўш бўлса хам), иккинчисида эса хеч нарса.

Мантиқий маълумотлар

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

Аниқланмаган типли маълумотлар.

Агар ўзгарувчи эълон қилинган бўлса, аммо унга хали қиймат ўзлаштирилмаган бўлса, у холда у аниқланмаган типга бўлади. Мисол учун қуйидаги қаторда аниқланмаган типга эга бўлган MyVariable ўзгарувчиси эълон қилинган:
```
var MyVariable;
```
Агарда бу ўзгарувчини null қиймати билан ўзгартирсак, у холда ўзгарувчи типи ўзгаради ва null қийматга эга бўлган ўзгарувчига айланади:

```
    MyVariable = null;
```

JavaScript тили операторлари

Унар оператори

Унар оператори белгининг ўзгариши учун тўлдириш операциясини бажаришда, инкраментда хамда декрементда ишлатилади:
```
-    тескари холатда белгининг ўзгариши
!        Қушимча. Мантиқий ўзгарувчиларнинг қийматини реверсирования қилиш учун ишлатилади.
++   Ўзгарувчи қийматини ошириш. Ўзгарувчи префикси ёки унинг суффикси бўлиб қўлланиши мумкин.
--   Ўзгарувчи қийматини камайтириш. Ўзгарувчи префикси ёки унинг суффикси бўлиб қўлланиши мумкин.
```
Унар операторини ишлатишга доир мисоллар:
```
i=0;          // i тенг 0 даги ўзгарувчининг бошланғич қиймати
i++;       // i тенг 1 даги қиймат
--i;       // i тенг 0 даги қиймати

var j=3;   // j тенг 3 даги ўзгарувчининг қиймати
i = -j;    // i тенг -3 даги ўзгарувчининг қиймати
```

```
var fYes = true; // fYes тенг true даги ўзгарувчининг қиймати
testFlag(!fYes); // testFlag функциясига false қиймати узатилмоқда
```

Бинар оператори

Бинар оператори 2 та операндни бирлаштиради. JavaScript тилида бинар операторлари айириш, бўлиш, қўшиш, кўпайтириш хамда бўлинмани қолдиғини хисоблаш учун ишлатилади (кўрилади):

```
-     Айириш
+     Қўшиш
*     Кўпайтириш
/     Бўлиш
%     Бўлинмани қолдиғини хисоблаш
```

Бу операторлар C 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 оператори

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
```

Мантиыий операторлар

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

Ўзлаштириш оператори

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

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

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

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

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

Энди буни += оператори ёрдамида бажарамиз:

```
var nCounter = 1;
nCounter += 5;
```
Кўриниб турибдики 2-усул 1-усулга нисбатан қисқа.
Ўзгарувчи қийматини 3 разрядга ўнгга силжитиш учун >>= рператоридан фойдаланиш мумкин ва у қуйидаги матнда кўрсатилган:
```
nCounter >>= 3;
```
Натижа эса қуйидаги матнда кўрсатилганидек бўлади:
```
nCounter = nCounter >> 3;
```

JavaScript тилида функция

Бошланғич матн бўлагини функция кўринишида ёзиш мумкин ва уларни **JavaScript** сценарийсининг турли жойларидан мурожаат қилиш мумкин. Одатда функциялар HTML документини сарлавха бўлимида аниқланади. Функциялар чақирилишидан аввал эълон қилиниши керак ва барча функция эълони HTML документ сарлавхасида жойлаштирилган бўлиши керак.
Функциянинг умумий эълони қуйида келтирилган:
```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{  . . .
  Функция матни қаторлари
  . . .
  [return қиймат]
}
```
Барча параметрлар функцияга қийматига берилади. Шунинг учун функция унга параметр сифатида бериладиган ўзгарувчилар қийматини ўзгартира олмайди.
Return калит сўзи ёрдамида функция қиймати қайтарилади.

Топшириқ

1. Қуйидаги кўринишда шарт оператори таркибини тузинг:
(шарт)? амал1:амал2
Бунда қуйидаги операторлардан фойдаланинг
A) Арифметик амаллр
B) Ўзлаштириш
C) Инкремент
D) Декремент

2. Ҳисобланг
a)12&9= ; b)13&14=; c)10&4=

d)14|10 = e)16^3= f) 14&56|11=

Амалий иш - 4

Мавзу: JavaScript да дастурни бошқариш элементларидан фойдаланиш. Шарт, цикл, танлаш ва бошқа операторлар. Функция ва усуллар яратиш. Объектлар.

Ишнинг мақсади: JavaScript нинг бошкариш элеметларини билиш. Шарт, цикл, танлаш ва бошқа операторлар билан ишлаш. Функция ва усуллар яратиш. Объектларни ўрганиш.

Ишнинг натижасида талаба қуйидагиларга эга бўлиши керак:

Билиши керак: JavaScript нинг бошкариш элементларини;

Қила олиши керак: HTML-хужжатда JavaScript ёрдамида бошкариш элементларидан фойдаланиб илова яратиш. Функция, усуллар ва объектлар яратиш

НАЗАРИЙ ҚИСМ

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.

JavaScript тилида C ва Java дастурлаш тилларидаги каби процедура ва функциялар мавжуд бўлиб, улар қуйидагича эълон килинади:

- function калит сўзи;
- функция номи;
- вергул ва кавс билан ажратилган аргументлар рўйхати;
- фигурали кавс ичига олинган функция танаси.

```
function myFunction(arg1, arg2, ...)
{
...
Операторлар кетма-кетлиги
...
   }
```

Бу ерда myFunction – функция номи, arg1, arg2 – параметрлар.

Мисол:

```
function Factorial(n) {
if((n<0)||(round(n)!=n)) {
alert("Factorial функцияси ушбу аргументда аниқланмади "+n);
return NaN;
} else {
result=(n*Factorial(n-1));
return result;
}}
```

Функцияда return калит сўзи орқали қиймат қайтарилмаслиги ҳам мумкин.

Мисол:

```
function Greeting(s){
document.write("Hello,"+s+"!");
return ;
}
```

Функцияни чақириш аниқ параметрлар билан чақирилади:

Мисол:

Factorial(3);

- бу функция натижаси 6 га тенг,

Greeting("world");

- бу функция экранга "Hello, world!" стрини чиқаради.

Ҳар бир функция, масалан, **myFunction** функцияси myFunction номли объект хисобланади, агарда аргументлар arguments номи билан берилса, унга мурожаат қуйидагича:

myFunction.arguments[i], бу ерда **i** — аргумента номери (рақамлаш 0 дан бошланади).

Функция эълонида аниқ параметрлар формал параметрларга тенг еки кўп сонда бўлиши лозим. Бунда функция ишга туширилганда жунатилаетган аргументлар миқдори myFunction.arguments.length майдони ёрдамида аниқланади ва ушбу майдондаги қийматни қайта ўзлаштиришни динамик ўзгартириш мумкин.

Мисол:

Экранга HTML форматидаги рўёхатни чиқариш.

Бу ерда (ListType) нинг биринчи аргументи тартибланмаган рўйхат учун "o" еки "O", тартибланмаган рўйхат учун "u" еки "U" булиши мумкин.

```
function myList(ListType)  {
document.write("<"+ListType+"L");
for(var i=1; i < myList.arguments.length; i=i+1) {
document.write("<LI>"+myList.arguments[i]);
}
document.write("</"+ListType+"L>");
}
```

HTML ҳужжатида функцияга мурожаат қуйидагича:

```
<script> myList("o", "матн", 2, "3")  </script>
```

Натижа:

матн
2
3

Global класи

Ушбу класс JavaScript нинг функционал қисми бўлиб, бу класс бир объектда бир нечта усул ва хоссаларни бирлаштириш вазифасини бажаради. Усулга мурожаат қилинганда объект кўрсатилмайди, аниқроғи бу усул конструкторга эга бўлмайди. Бундай хосса ва усулларга қуйидагиларни келтириш мумкин:

Хосса	**Мазмуни**
Nan	NaN (Not A Number)
Infinity	Number.POSITIVE_INFINITY қийматни ўз ичига олади

Усул	**Мазмуни**
escape	Қаторни барча платформаларга мос ҳолда тасвирлаш
eval	JavaScript тили функцияси еки усулларини узатиш
isFinite	Аргументнинг охирги рақамлилигини аниқлаш
isNan	Аргументнинг рақам ёки рақам эмаслигини аниқлаш
parseFloat	Қаторни кўчиб юрувчи нуқтали сон кўринишида тасвирлаш
parseInt	Қаторни бутун сонга айлантириш
unescape	Escape функцияси натижасини қайтариш

eval(s) функцияси - s қаторни JavaScript операторлари кетма-кетлиги кўринишида тасвирлаш.

getClass(Jobj) функцияси – JavaObject типидаги аргумент учун JavaClass объектини қайтаради.

Мисол:

var myJavaRClass=new java.awtRectangle()

var myJavaRClass=getClass(myJavaRect)

getClass() Java-методи билан адаштирманг:

var myJavaRCObject=myJavaRect.getClass() - бу java.awt.Rectangle класининг Java тилидаги реализация ҳолати.

isNaN(x) функцияси – x "Not a Number", яъни сон эмаслигини текшириш.

parseFloat(s) функцияси – Float типидаги s рақамни аниқлаш. Агар сон топилмаса у ҳолда NaN ("Not a Number") қиймати қайтарилади.

parseInt(s) – Integer типи учун юқоридаги ҳолат.

eval(s) функцияси

eval(s) функцияси – JavaScript нинг ички функцияси ҳисобланади. Ушбу функция бир ёки бир нечта JavaScript операторларидан иборат бўлган s сатрни аргумент томонидан узатилган кодни бажаради. Бунда s сатридаги операторлар нуқтали вергул ёрдамида ажратилади. Бу функция нафақат операторни бажариш, балки бирор амалларни ҳисоблаш имконини ҳам беради. Бунда у кодда келтирилган амал хисобининг охирги қийматини қайтаради.

isNaN(x) функцияси

Бу функция x аргументнинг "сон эмас" лигини текширади. Натижа NaN қийматга эга эмаслигини, яъни мумкин булмаган сон (масалан, нолни нолга бўлиш натижаси) ни текширади. Ушбу функция JavaScript да литерал кўринишда NaN қийматни бериш мумкин эмаслиги учун муҳимдир. Бундан ташқари parseFloat(s) ва parseInt(s) функциялар натижаларини текшириш (мумкин бўлган сон эканлигини) ва арифметик хатолар мавжудлиги, масалан, нол сонига бўлиш мавжудлигини текширади.

parseFloat(s) функцияси

s сатрини синтактик анализ қилиш ва дастлаб ракамни қайтариш (сатрни рақамна айлантиради). parseFloat(s) да s сатрида рухсат этилмаган рақам элементлари (масалан, белгилар, рақам, ўнли вергуллар, даража кўрсаткичи ва ҳоказо) мавжуд булса анализ тўхтатилади ва қиймат қайтарилади. Агарда s сатрда сон билан бошланмаса, у ҳолда parseFloat(s) функция NaN қийматни қайтаради.

parseInt(s) функцияси

Бу функция сатрни бутун сонга айлантиради. parseInt(s) функциядаги s сатрда ҳисоблаш тизимида кўрсатилмаган қийматларга эга бўлганда синтактик анализ тўхтатилади ва қиймат қайтарилади. Одатда, parseFloat ва parseInt функциялар s сатр сон билан бошланмаганда NaN қиймат қайтаради.

parseInt(s,n) ҳолатида n асос иҳсобланиб, агарда n=10 бўлса, parseInt(s) функция сатрдаги 10 лик санок системасидаги сонларни текширади. n=8 бўлса, 8 лик соноқ тизимидаги сонлар мавжудлигини (бунда n 0 дан 7 гача бўлган сонлар қийматига эга бўлиши мумкин). n=16 бўлса, 16 лик соноқ тизимидаги сонлар мавжудлигини (бунда 0 дан 9 гача бўлган сонлар ва A дан F гача бўлган ҳарфлар қийматига эга бўлинади). Агарда n=0 бўлса ёки қиймат берилмаса, у ҳолда parseInt(s) функция сатрнинг ўзидан асосни аниқлайди. Бу ҳолатда агарда сатр 0x билан бошланса, унда функция сатрнинг қолган қисмини 16 лик саноқ тизимидаги сон сифатида анализ қилади, агарда сатр 0 дан бошланса, сатр 8 лик саноқ тизимидаги қиймат сифатида анализ қилинади.

Math класси

Math – константалар ва методлардан иборат классдир. Улар объект учун одатдагидек мурожаат қилинади:

Math.константа

Math.функция(i..)

Math класси константалари

E – е сони (натурал лагорифм асосли)
LN10 — 10 ли натурал лагорифм (ln10 сони)
LN2 — 2 ли натурал лагорифм (ln2 сони)
LOG10E — 10 асосли е лагорифм (log10e сони)
LOG2E — 2 асосли е лагорифм (log2e сони)
PI — p константаси ("пи" сони)
SQRT1_2 — 2 нинг тескари квадрат илдизи ($1/\sqrt{2}$)
SQRT2 — 2 нинг квадрат илдизи ($\sqrt{2}$)

Math класси методлари

abs(x) (x-сон еки ифода) – абсолют қийматни ҳисоблаш;

acos(x) (x бу ерда [-1.0;1.0] радиан интервалдаги сон еки ифода) – арккосинусни ҳисоблаш. Қайтариладиган қиймат 0 дан p радиан оралиғида бўлади.

asin(x) (x бу ерда [-1.0;1.0] радиан интервалдаги сон еки ифода) – арксинусни ҳисоблаш. Қайтариладиган қиймат –p/2 дан p/2 радиан оралиғида бўлади.

atan (x) (x – сон еки ифода) – арктангенсни радианларда ҳисоблаш. Қайтариладиган қиймат –p/2 дан p/2 радиан оралиғида бўлади.

atan2(x,y)(x,y — тўғри бурчакли кординаната системаси кординаната нуқталари) – қутб кординатасида (x,y) нуқталар бурчагини ҳисоблайди. Қиймати 0 дан 2p радиан оралиғида бўлади.

ceil(x) (x — сон ёки сонли ифода) – сонни бутун сонга йўналтирилган ҳолда яхлитлаш. Манфий сонлар 0 сони йўналишига караб яхлитланади.

cos(x) (x – радиандаги бурчак) – косинусни ҳисоблаш, қайтариладиган қиймат -1.0 дан 1.0 радиан оралиғида бўлади.

sin(x) (x – радиандаги бурчак) – косинусни ҳисоблаш, қайтариладиган қиймат -1.0 дан 1.0 радиан оралиғида бўлади.

Exp(x) (x — сон ёки сонли ифода) — е экспонентсини ҳисоблаш.

Floor(x) (x — сон ёки сонли ифода) – сонни бутун қисмига йўналтириб яхлитлаш, масалан, floor(-1,1) тенг (-2); floor(1,1) тенг 1.

Log(x) (x — мусбат сон ёки ифода) – натурал лагорифмни ҳисоблаш.

max(a,b) (a,b — сон ёки ифода) – икки қийматдан каттасини қайтаради.

min(a,b) (a,b — сон ёки ифода) – икки қийматдан кичигини қайтаради.

pow(x,y) — x ни ҳисоблаш (биринчи аргументни даражага кўтариш).

random — 0 дан 1 гача интервалдаги тасодифий сонларни ҳисоблаш.

round — сонни бутун қисмига қараб яхлитлаш (масалан, round(15.5) натижаси 16 ни беради, round(-15.5) дает -15).

Math.round(x) (x — сон ёки ифода)

Math.sin(x) (x — радианда берилган бурчак)

Math.sqrt(x) (x — 0 га тенг ёки катта бўлган сон ёки ифода)

tan —тангенсни ҳисоблаш.

Math.tan(x) (x — радианда берилган бурчак)

Класс Date

Date() методи аргументсиз берилганда қиймати жорий сана ва вақтга эга Date объекти яратилади. Date() методида янги объект учун аргументи сифатида сана ва зарур ҳолларда вақт кўрсатилади. Date методи JavaScript тили объексти ҳисобланиб, HTML тилида ҳеч қандай аналогга эга эмас. Кўп ҳолларда Date объекти методлари унинг экземпляти ёрдамида чақирилади, масалан:

d=new Date(); // бугунги сана ва вақтни олиш

system.write("Today is: "+d.toLocalString()); // ва уни тасвирлаш

Date объектини яратишнинг юқоридаги синтактикасида кўрсатилгани бўйича, сана ва вақт ҳудудий вакт бўйича берилади. Агарда тузилаётган дастур фойдаланувчи жойлашган часовой поясга боғлиқ бўлмаган ҳолда ишлаши зарур бўлса, у ҳолда Гринвич (GMT) еки универсал координация вақти (UTC) бўйича санани кўрсатиш керак бўлади.

Date объектини яратишда қуйидаги 5 та синтактик вариантдан фойдаланиш мумкин. 3-5 вариантларда вақт ҳудудий тарзда интерпретация қилинади (Гринвич да эмас):

1. new Date();
2. new Date(миллисекунд) – бу ерда миллисекунд жорий сана билан 01.01.1970 сана яримкуни орасидаги сон;
3. new Date(сана сатри) – бунда сана сатри = ой номи, дд, гг [чч:мм[:сс]])
4. new Date(йил, ой, кун) – бунда, йил 2011; ой 0-11; кун 1-31;
5. new Date(йил, ой, кун , соат, минут, секунд) – 24 соатликтизимда.

Date класси методлари

getDate() - Date объектининг 1 дан 31 гача оралиқдаги қийматини беради;

getDay() - Date объектининг 0 [якшанба] дан 6 [шанба] гача оралиқдаги хафта кунлари беради;

getHours() - Date объектининг 0 [ярим тун] дан 23 гача оралиқдаги соат майдони қийматини беради;

getMinutes() - Date объектининг 0 дан 59 гача оралиқдаги минут майдони қийматини беради;

getSeconds() - Date объектининг 0 дан 59 гача оралиқдаги секунд майдони қийматини беради;

getMonth() - Date объектининг 0 [январ] дан 11 [декабр] гача оралиқдаги ойларни беради;

getTime() - Date объекти вақт кўрсаткичининг миллисекундлардаги қийматини беради;

getYear() - Date объекти вақт кўрсаткичининг йиллар майдони қийматини беради; бунда 2011 йил 11 кўринишида берилади;

parse() – сананинг сатр кўринишидаги ҳолатини синтактик анализ қилади ва натижани миллисекунд форматида беради;

setDate() - Date объекти вақт кўрсаткичини ўрнатади;

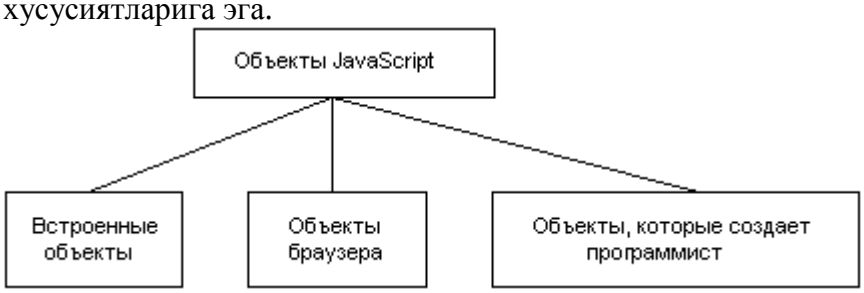
дата.setDate(ой сони) //ой сони 1-31 оралиқда.

toLocalString() – жорий ҳудудий вақт майдонини асосида Date форматини матнли (String) кўринишга келтиради;

UTC() – сана ва вақтнинг рақамли кўринишини миллисекундли форматга айлантиради.

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());
```

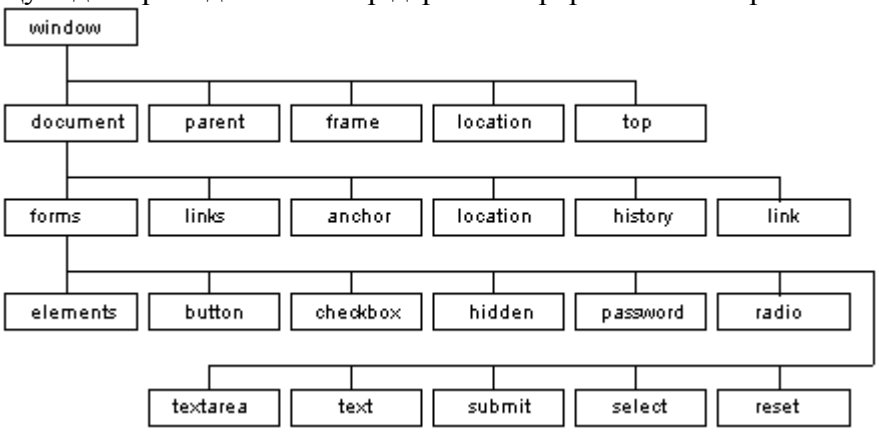
Браузер объектлари

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

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;
}
```

Яратилаётган объектни 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.

Амалий машғулот - 5
Мавзу: PHP га кириш. Маълумотлар типлари билан ишлаш. Ўзгарувчилар. Амаллар. Операциялар

Ишнинг мақсади: PHP ни ўрганиш ва маълумотлар типларидан фойдаланиш, ўзгарувчилар, амаллар операциялар.

Талаба қуйидагиларга эга бўлиши керак:

Билиши керак маълумотлар типлари. ўзгарувчилар, амаллар ва операциялар.

Қилаолиши керак PHPда ишлаш;

НАЗАРИЙ БЎЛИМ

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:

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.

АСОСИЙ СИНТАКСИСЛАР.

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

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

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

Биринчи навбатда PHP дастурлаш тили синтаксисига нисбатан нималарни билиш керак. Бу HTML-код ичига ўрнатилган ва PHP дастурлаш тилидаги коддир, уни интерпретатор фарқлай билади. Аввалги бўлимларда булар ҳақида айтиб ўтгандик. Ҳаммасини қайтариб ўтмаймиз, фақат биз кўп ҳолларда мисолларда `<?php ?>` вариант ўрнига қисқартирилган `<? ?>` теглардан фойдаланишни айтиб ўтамиз.

Инструкцияларни ажратилиши.

PHP дастурлаш тилидаги дастур(ихтиёрий дастурлаш тилидаги) – бу буйруқлар (инструкциялар) тўпламидир. Дастурни қайта ишлаш учун бир буйруқни бошқа буйруқдан фарқини билиш керак. Бунинг учун махсус символлар – ажратгичлардан

фойдаланилади. PHP дастурлаш тилида инструкцияларни худди Си ёки Perl дастурлаш тиллари каби ажратилади, яъни ҳар бир ифода нуқтали вергул (“;”) билан тугайди.

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

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

Комментарийлар.

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

```
<?php
echo "Мени исмим Алишер";
  // Бу бир сатрли комментарий C++ дастурлаш тили стилидаги
echo "Мени фамилиям Болиев";
/* Бу кўп сатрли комментарий. Бу ерга бир қанча сатр ёзиш мумкин. Дастур бажарилиш жараёнида бу ердаги барча ёзувлар (комментарийланган), ўқилмайди. */
echo "Мен PHP дастурлаш тилини INTUIT.ru дан ўрганяпман";
  # Бу комментарий Unix қобиғидаги комментарий.
?>
```

ЎЗГАРУВЧИЛАР, ЎЗГАРМАСЛАР ВА ОПЕРАТОРЛАР.

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

Ўзгарувчилар

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

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

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

*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.

Мисол. Қиймат бўйича ўзлаштириш

```
<?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
?>
```

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:

Мисол. Ссылкалар бўйича ўзлаштириш

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

Бу скриптни натижаси эса қуйидагича бўлади:

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

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

Ўзгармаслар

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

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

Ўзгармаслар номи регистрга сегирлиги катта. Ҳар бир ўзгармасларда уни ўзгартириш мумкин, яъни *регистрга_сезгирлиги_кичик* аргументни қиймати сифатида

True ыйматы кўрсатилади. Ўзгармаслар номи ҳар доим катта регистр билан ёзишга келишиб олинган.

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

Мисол. PHP дастурлаш тилида ўзгармаслар

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

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

Амаллар

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

9.1-жадвал. Арифметик амаллар.		
Белгиланиши	**Номланиши**	**Мисол**
+	Қўшиш	`$a + $b`
-	Айириш	`$a - $b`
*	Кўпайтириш	`$a * $b`
/	Бўлиш	`$a / $b`
%	Бўлишдаги қолдиқ	`$a % $b`

9.2-жадвал. Сатрли амаллар.		
Белгиланиши	**Номланиши**	**Мисол**
.	*Конкатенация* (*сатрларни қўшиш*)	$c = $a . $b (бу $c сатр `$a` ва `$b` сатрлардан

		иборат)

9.3-жадвал. Ўзлаштириш амаллари.

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

9.4-жадвал. Матиқий амаллар.

Белгиланиши	Номланиши	Изох	Мисол
and	ВА	$a ва $b рост (True)	$a and $b
&&	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

9.5-жадвал. Таққослаш амаллари.

Белгиланиши	Номланиши	Изох	Мисол
==	Тенглик	Ўзгарувчилар қийматлари тенг	$a == $b
===	Эквивалентлик	Ўзгарувчилар қийматлари ва типлари тенг	$a === $b
!=	Тенгсизлик	Ўзгарувчилар қийматлари тенг эмас	$a != $b
<>	Тенгсизлик		$a <> $b
!==	Ноэквивалентлик	Ўзгарувчилар эквивалент эмас	$a !== $b
<	Кичик		$a < $b
>	Катта		$a > $b
<=	Кичик ёки тенг		$a <= $b
>=	Катта ёки тенг		$a >= $b

9.6-жадвал. Инкремент ва декремент амаллари.

Белгиланиши	Номланиши	Изох	Мисол
++$a	Пре-*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	

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;  ?>
```

Мантиқий типлар турли *бошқариладиган конструкциялар*да (цикллар, шартлар ва шунга ўхшаш, булар ҳақида кейинроқ айтиб ўтамиз) қўлланилади. Бир қанча амаллар (масалан, тенглик амали) ҳам мантиқий тип қабул қилиши мумкин, яъни фақат икки қиймат рост ёки ёлғон қийматни қабул қилади. Улар *бошқариладиган конструкциялар*да шартларни текшириш учун қўлланилади. Масалан, шартли конструкторда амаллар ёки ўзгарувчилар қиймати ҳақиқийлигини текширади ва натижадан қатъий назар шу ёки бошқа амалларни бажарилишини текширади. Бу ерда шарт рост ёки ёлғон бўлиши мумкин, чунки *мантиқий тип амаллари* ва *ўзгарувчилар* кўрсатилган.

Мисол. Мантиқий типларнинг қўлланилиши

```
<?php
// '==' амал тенгликка текширади
// мантиқий қийматни
// қайтаради.
if ($know == False) { // агар $know қиймат false бўлса
echo "PHP дастурлаш тилини ўрган!";
}
```

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

Integer (бутун) типи.

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

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

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

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

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

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

Float (қўзғалувчан вергулли сон) типи.

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

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

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

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 Php"; */
$books["lisp"] = 123456; /* Бу массивга янги "lisp" калитли 123456 қиймали янги элемент қўшиш*/
unset($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.

Мисол. PHP дастурлаш тилида объектлар.

```
<?php
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.
?>
```

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 типли фақат битта қиймати мавжуд – регистрга сезгирлиги кичик NULL калит сўзидир.

Масаланинг ечилиши.

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

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

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

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`.

Мисол. if шарт оператори.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($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";
?>
```

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.

Мисол. else оператори.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($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 биринчи эмас?!";
$bax = 30;
if ($account > 100*$bax+3)
   echo " Бу сатр экранга чиқмайди, чунки  шарт бажарилмайди ";
   else echo "Шундай бўлса ҳам экранга чиқди!";
?>
```

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.

Мисол. elseif оператори.

```
<?
$names = array("Карим","Салим","Содиқ");
if ($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?";
}
```

?>

Альтернатив синтаксислар.

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

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

if (ифода) : *бажариладиган_блок* endif;

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

Мисол. Альтернатив синтаксисдан фойдаланиш.

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

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

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

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] ";
}
?>
```

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.

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.

Мисол. foreach оператори.

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

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.

Мисол. Break оператори

```
<?php
$i=1;
while ($i) {
   $n = rand(1,10);
   // исталган сонни умумийлаштирамиз 1 дан 10 гача
   echo "$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 ";
?>
```

Бу скрипт ишининг натижаси 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 (break мавжуд 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:

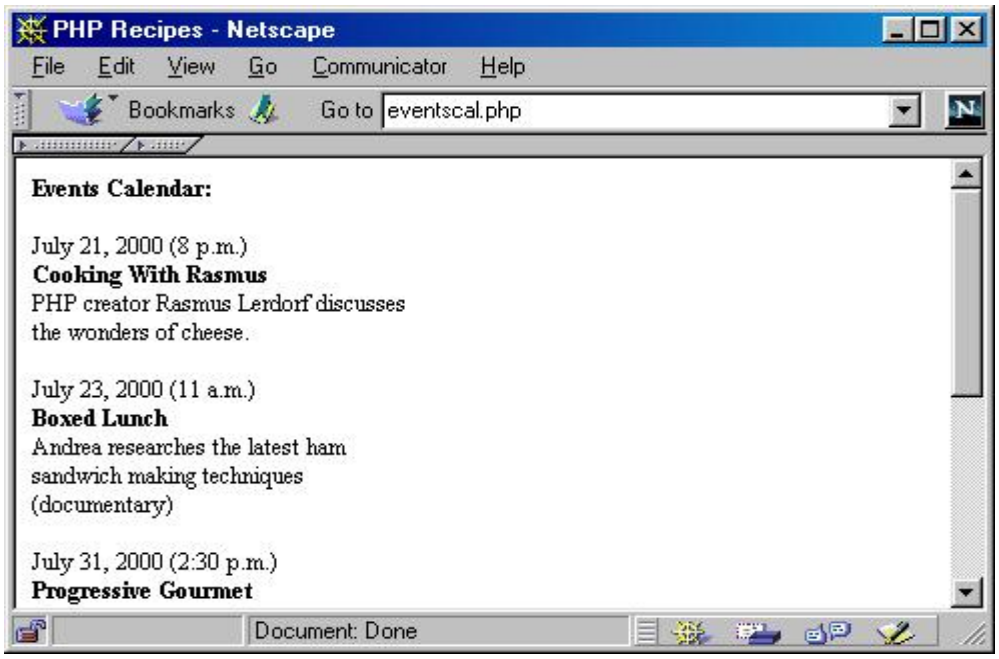
```
<?php
$i=1;
while ($i<4) {
   $n = rand(1,10);
  // исталган сонни умумлаштирамиз 1 дан 10 гача
   if ($n!==5) {
      echo "$i:$n <br>";
  // интерация рақамини чиқарамиз ва умумлаштирилган сон
   $i++;
   }
}
?>
```

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.

ТОПШИРИҚ

Проект: календар

1.Қуйидаги расмда келтирилган натижани берувчи скрипт тузинг



Амалий машғулот - 6
Мавзу: PHPда маълумотлар базалари билан ишлаш. MySQL маълумотлар базаси

Ишнинг мақсади: PHP да маълумотлар базаси билан ишлашни урганиш
Талаба қуйидагиларга эга бўлиши керак:
Билиши керак: маълумотлар базаси билан богланишни ва суровлар билан ишлашни
Қила олиши керак: PHPда MySql билан бирга ишлаш;

НАЗАРИЙ БЎЛИМ

Ушбу бўлим PHP ва MySql МББС (СУБД) ўртасидаги ҳамкорлик усуллари билан танишишга мўлжалланган. Асосий эътибор маълумотлар базаси билан боғланишни ўрнатиш, сўровлар жўнатиш функциялари ва жавобларни (mysql_connect, mysql_query, mysql_result, mysql_num_rows, mysql_close) қайта ишлашга қаратилади. Мисол сифатида виртуал тарих музейи маълумотлар базасининг маъмурияти учун web-интерфейс тузиш масаласини кўрайлик. PHP дистрибутивига MySQL маълумотлар базаси билан ишлаш учун мўлжалланган функцияларни олувчи кенгайтма киради. Бу бўлимда MySQL билан ишлаш учун баъзи бир маълумотлар базасини тасвирлаш ва тўлдириш мақсадида web-интерфейсларни тузиш топшириғини ечиш учун керак бўладиган асосий функциялар билан танишамиз. Савол туғилади: бундай интерфейсларни тузиш нега керак? SQL сўровлар тили билан нотаниш одамлар маълумотни маълумотлар базасига киритиш ва унинг таркибини кўриб туриш имконияти бўлиши учун шундай қилинади. Маълумотлар базасига маълумотларни қўшиш учун web-интерфейс билан ишлашда бу маълумотларни шунчаки html-формага киритиш ва уларни серверга жўнатиш керак бўлади, бизнинг скрипт эса қолган барча амалларни бажаради. Жадвал таркибини кўриб туриш учун ҳавола устига бир марта босиш ва керакли саҳифага кириш кифоя.

Кўриниб туриши учун бу интерфейсларни виртуал музей экспонатлари ҳақидаги маълумотлар жойланадиган Artifacts жадваллари учун тузамиз. Аввалги бўлимда бу коллекцияга структурани ҳамда унинг шахс (Persons) ва тасвирлар (Images) тавсифлари коллекциялари билан алоқасини киритган эдик. Artifacts коллекциясидаги ҳар бир экспонат қуйидаги характеристика ёрдамида тасвирланишини эслатиб ўтамиз:
ном (title);
муаллиф (author);
ифода (description);
ўриндош ном (alternative);
тасвир (photo).

Номланиш ва ўриндош номланиш узунасига 255 белгидан кам сатр (яъни VARCHAR(255)), тасвирлаш – матнли майдон (TEXT турига мансуб) ҳисобланади, “муаллиф” ва “тасвир” майдонларида эса Persons коллекциясидан муаллифнинг идентификаторлари ва Images коллекциясидан экспонат тасвирларига мувофиқ мавжуд бўлади.

Маълумотни қўшиш учун интерфейс тузиш
Демак, бизда маълумотлар базасида бирон-бир жадвал бор. Бу жадвалга маълумотни қўшишга мўлжалланган интерфейс тузиш учун унинг структурасини (яъни унинг майдонлари жамланмасини) html-формада 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.

Алоқа ўрнатиш
Демак, биринчи галдаги вазифа – бу маълумотлар базаси билан алоқа ўрнатиш. **mysql_connect** функциясидан фойдаланамиз. mysql_connect синтаксиси

mysql_connect ресурси ([сервер қатори[, username қатори[, password қатори[, мантиқий **new_link**[, бутун client_flags]]]]])

Бу функция **MySQL** сервери билан алоқа ўрнатади ва бу алоқага кўрсаткич қайтаради ёки муваффақиятсиз чиққанда FALSE кўрсатади. Етишмаётган параметрлар учун қуйидаги маънолар муваққат қўйилади:
server = 'localhost:3306'
username = сервер жараёни эгасидан фойдаланувчи исми
password = бўш парол

Агар функция айнан бир хил параметрлар билан икки марта чақириладиган бўлса, янги уланиш ўрнатилмайди, аксинча ҳавола эски алоқага қайтарилади. Бундан қочиш учун эса, ҳар қандай ҳолатда янги бир алоқа очишга мажбур қилувчи new_link параметридан фойдаланилади.

client_flags параметри – бу қуйидаги константалар комбинацияси:

MYSQL_CLIENT_COMPRESS (сиқиш протоколидан фойдаланилади),
MYSQL_CLIENT_IGNORE_SPACE

(функция номидан сўнг пробел қўйишга рухсат беради),

MYSQL_CLIENT_INTERACTIVE (interactive_timeout бир сония кутиш - wait_timeout – уланиш тугатилишигача). new_link параметри PHP 4.2.0 да мавжуд, client_flags эса – PHP 4.3.0 да.

Сервер билан уланиш, агар у бунгача mysql_close() ёрдамида ёпилмаган бўлса, скриптни амалга ошириш тугалланишида беркилади.

Шундай қилиб, “123” паролли nina фойдаланувчиси учун локал серверда маълумотлар базаси билан уланишни ўрнатамиз.

```
<?
$conn = mysql_connect(
   "localhost", "nina","123")
or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_close($conn);
?>
```

mysql_connect амали
shell>mysql -u nina -p123 буйруғи билан тенг кучли.

Маълумотлар базаларини танлаш
Уланишни амалга оширгандан сўнг бирга ишлашимиз керак бўлган маълумотлар базасини танлашимиз керак бўлади. Бизнинг маълумотларимиз book маълумотлар базасида сақланади.

MySQL да маълумотлар базасини танлаш use буйруғи ёрдамида амалга оширилади:
mysql>use book;

PHP да бунинг учун mysql_select_db функцияси мавжуд
mysql_select_db: синтаксиси
мантиқий mysql_select_db (
  database_name қатори
  [,link_identifier ресурси])

Бу функция TRUE қийматни маълумотлар базасини муваффақиятли танланганда қайтаради ва FALSE ни эса – аксинча бўлганда.

Book маълумотлар базасини ишга туширамиз:

```
<?
$conn = mysql_connect(
   "localhost","nina","123")
or die("Уланишни амалга ошириб бўлмайди: ". mysql_error());
echo "Уланиш амалга ошди";
mysql_select_db("book");
?>
```

Жадвал майдонлари рўйхатини олиш

Энди топшириқни ечиш билан алоҳида шуғулланса бўлади. Жадвал майдонлари рўйхатини қандай олиш мумкин? Жуда оддий. PHP да бу ҳолда ҳам ўзига тегишли буйруқ мавжуд - mysql_list_fields.
mysql_list_fields синтаксиси
mysql_list_fields (ресурси
database_name қатори,
table_name қатори
[, ресурс link_identifier])

Бу функция майдонлар рўйхатини database_name маълумотлар базасидаги table_name жадвалига қайтаради. Келиб чиқадики, маълумотлар базасини танлаш бизга мажбурий эмас, лекин кейинроқ асқотади. Бу функция ишининг натижасини – ресурс типи ўзгарувчисини қандай баҳолаш мумкин. Яъни бу биз олишни хоҳлаган нарса эмас. Бу уларнинг номлари, типлари ва байроқлари 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:

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:

Мисол. Artifacts жадвалига маълумот киритиш учун форма

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

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.

ТОПШИРИҚ

PHP да MySql маълумотлар базасидаги мавжуд жадвалдан маълумотларни руйхат куринишида браузерга чикаринг.