PHP, MySQL жана Bootstrap менен Веб-сайтты түзүү негиздери

Ош 2025

УДК 004.45

ББК 32.973.4

А.90

Окуу усулдук колдонмо Ош мамлекеттик университетинин Окумуштуулар кеңешинин 2025-жылдын март айындагы №5- жыйыдын токтомунун негизинде басмага сунушталган

Рецененттер:

Мамбетов Ж.И. - ОшТУнун Колдонмо информатика кафедрасынын башчысы, физ.-мат.илим.канд., доцент.

Аркабаев Н.К. – Ош мамлекеттик университенин ИСП кафедрасынын физ.-мат.илим.канд., доцент.

Асилбеков Т. М., Орозов М.О.

А.90 PHP, MySQL жана Bootstrap менен Веб-сайтты түзүү негиздери– Ош:2025 – 96 б.

ISBN 978-9967-06-093-7

Бул китепте PHP программалоо тилин, MySQL маалымат базасын жана Bootstrap фреймворкун колдонуп веб-колдонуучулардын CRUD операцияларын аткаруунун негиздери баяндалат. Ар бир бөлүмдө түшүндүрмөлөр, код мисалдары жана практикалык тапшырмалар камтылган

Бул китеп маалымат технологиялар багытындагы окутуучуларга жана студенттерге арналган.

ISBN 978-9967-06-093-7

Киришүү

Заманбап веб-сайттар — бул компьютердик технологиялардын өнүгүшүнүн натыйжасы. Веб-сайттардын көптөгөн түрлөрү пайда болуп, бул бизге маалыматтарды таратуу, байланыш жана кызмат көрсөтүү мүмкүнчүлүгүн берет. Бул китепте биз PHP программалоо тилин, MySQL маалымат базасын жана Bootstrap фреймворкун колдонуп, веб-колдонуучуларга маалыматтарды эффективдүү башкаруунун негиздерин үйрөнөбүз.

PHP – бул динамикалык веб-баракчаларды түзүү үчүн колдонулган сервердик программалоо тили. PHP программасы веб-серверде иштейт, бул сиздин веб-баракчаларыңыздын интерактивдүүлүгүн камсыз кылат. Бул китепте биз PHPдин негиздерин, синтаксисин жана функцияларын түшүндүрөбүз, ошондой эле маалыматтарды веб-баракчалар аркылуу башкаруу ыкмаларын көрсөтөбүз.

MySQL – бул маалымат базасын башкаруу системасы, ал сизге маалыматтарды сактоого жана башкарууга мүмкүнчүлүк берет. Биз MySQLдин негиздерин, таблицаларды түзүү, маалыматтарды кошуу жана SQL командаларын колдонуу аркылуу маалыматтарды башкаруу жөнүндө үйрөнөбүз.

Bootstrap – бул веб-дизайнды жөнөкөйлөтүүчү фреймворк, ал сайттын сырткы көрүнүшүн кооздоп, аны мобильдик түзмөктөргө ылайыкташтырат. Bootstrap колдонуу менен, сиз веб-баракчаларыңыздын дизайнын оңой жана тез өзгөртө аласыз.

Бул китептин максаты – PHP, MySQL жана Bootstrap технологияларын колдонуп, веб-колдонуучулардын CRUD (Create, Read, Update, Delete) операцияларын аткарууну үйрөнүүгө жардам берүү. Ар бир бөлүмдө теориялык материалдар, код мисалдары жана практикалык тапшырмалар камтылган. Биз сизди веб-өнүктүрүүнүн негиздери менен тааныштырууга жана веб-баракчаларды түзүү боюнча жөндөмдөрүңүздү өстүрүүгө жардам беребиз.

Китепти окуу учурунда сиздин кызыгууларыңызды арттырууга, сиздин билимдериңизди кеңейтүүгө жана веб-сайттарды эффективдүү түзүү үчүн маанилүү билимдерди алууга үмүттөнөбүз.

1. Интернет технологиялары

Веб-технологиялар деген эмне?

Веб-технологиялар — бул интернет аркылуу маалыматтарды берүү жана кабыл алуу үчүн колдонулган программалар жана система. Мисалы, веб-сайттар, тиркемелер жана интернетте көргөн баракчаларыбыз веб-технологиялардын жардамы менен иштейт.

Браузер жана сервер деген эмне?

- **Браузер** — бул биз интернетти колдонгондо сайттарды ачкан программа. Браузерлерге Google Chrome, Firefox жана Safari сыяктуу программалар кирет. Браузерлер сайттарды ачып, биз үчүн маалыматтарды көрсөтөт.
- **Сервер** — бул сайттар жайгашкан компьютер. Ал башка компьютерлерден чоңураак болушу мүмкүн жана интернеттеги бардык сайттардын маалыматтарын сактайт. Сервер биз браузерде сайтты ачканда маалыматты берет.

HTTP деген эмне?

HTTP (HyperText Transfer Protocol) — бул браузер менен сервердин бири-бири менен сүйлөшүү жол-жобосу. Сайтты ачканда, браузер серверге HTTP аркылуу "сураныч" жөнөтөт, ал эми сервер ошол суранычка "жооп" кайтарат. Мисалы, биз сайттын атын жазып, Enter басканда, браузер серверден ошол сайттын барагын сурайт, сервер болсо ал баракты браузерге жиберет.

Браузер менен сервердин иштешүүсү

Мисал менен түшүнүп көрөлү:

1. **Браузерде сайт ачуу**: Сайттын атын жазып, Enter басасың. Мисалы, "www.example.com".
2. **Сураныч жиберүү**: Браузер ошол сайтты кайсы серверде сакталганын аныктап, серверге HTTP аркылуу суроо жиберет: "Мага бул сайттын барагын берчи!".
3. **Серверден жооп алуу**: Сервер суроо келип түшкөндөн кийин, ал сайттын маалыматтарын табат да, браузерге жөнөтөт.

4. **Сайттын көрүнүшү**: Браузер серверден келген маалыматтарды окуп, экранда сага сайттын барагын көрсөтөт.

Ошентип, интернетти колдонгон сайын браузер менен сервердин өз ара иштешүүсү ушул жол менен ишке ашат.

Куралдар: PHP, Bootstrap жана MySQL деген эмне жана алар эмнеге пайдалуу?

PHP деген эмне?

PHP — бул веб-сайттарга акыл кошкон программалоо тили. Ал сайттарга динамикалуу мүмкүнчүлүктөрдү берет, башкача айтканда, сайттар жандуу болуп, колдонуучулар менен өз ара иштешет.

Мисалы, PHP менен эмне кылса болот?

- **Катталуу жана кирүү**: PHPни колдонуп, сайтка катталуу жана кирүү формаларын түзө аласыз.
- **Маалыматтарды иштеп чыгуу**: PHP суроо-талаптарды кабыл алып, аларды иштеп чыгат, мисалы, бирөөнүн комментарийин кошуу же бир нерсени сайтта издөө.
- **Маалыматты сактоо**: PHP маалыматтарды сайттын серверинде сактап, аны кайра чакырууга жардам берет.

Эмнеге PHP пайдалуу?

- **Жөнөкөй**: PHP программалоону үйрөнүү жана колдонуу оңой.
- **Ыңгайлуу**: Ал көпчүлүк сайттардын артында иштейт жана ар түрдүү кызматтарды сунуштайт.

Bootstrap деген эмне?

Bootstrap — бул веб-сайттардын көрүнүшүн жакшыртуу үчүн колдонулган даяр дизайн компоненттеринин жыйындысы. Ал CSS (сайттын сырткы көрүнүшүн башкаруучу тил) жана JavaScript (сайттын кыймыл-аракетин камсыздоочу тил) менен иштейт.

Bootstrap эмне кылат?

- **Адаптивдүү дизайн**: Сайтты компьютерде, планшетте жана телефондо туура көрүүгө жардам берет.
- **Даяр шаблондор**: Bootstrapте кнопкалар, менюлар, формалар сыяктуу алдын ала даярдалган элементтер бар, аларды колдонуу менен убакытты үнөмдөөгө болот.

Эмнеге Bootstrap пайдалуу?

- **Тезирээк иштөөгө жардам берет**: Bootstrapти колдонуу менен сайттын сырткы көрүнүшүн оңой жана ылдам түзүп алса болот.
- **Адаптивдүү дизайн**: Сайтты ар кандай түзмөктөрдө туура көрсөтүүнү камсыз кылат.

MySQL деген эмне?

MySQL — бул маалымат базасы системасы, ал маалыматтарды сактоо жана алар менен иштөө үчүн колдонулат. Сайттарга маалымат базасы керек, анткени колдонуучулардын каттоо маалыматтары, комментарийлер жана башка көптөгөн маалыматтар ошондо сакталат.

MySQL эмне кылат?

- **Маалыматтарды сактоо**: Бардык керектүү маалыматтарды иреттеп, чоң көлөмдөгү маалыматтарды сактайт.
- **Маалыматтарды издеп берүү**: Колдонуучу сайттан бир маалыматты издегенде, MySQL маалымат базасынан ошол издеген маалыматты таап, көрсөтүп берет.

Эмнеге MySQL пайдалуу?

- **Ылдам жана күчтүү**: MySQL чоң көлөмдөгү маалыматтарды иштетүүгө жөндөмдүү.
- **Коопсуздук**: Ал маалыматтарды коопсуз сактоого жана сыр сөздөрдү шифрлеп коргоого жардам берет.

Бул куралдар кандайча чогуу иштейт?

- **PHP** программалоо тили менен сайттын иштөө логикасын жазасың.

- **Bootstrap** колдонуп сайтты кооздойсуң жана ар кандай түзмөктөрдө туура иштешин камсыз кыласың.
- **MySQL** сайттын бардык маалыматтарын сактоого жардам берет, PHP болсо ошол маалыматтар менен иштейт.

Мисалы, биз сайтка катталуу системасын жасасак, PHP колдонуучунун атын жана сыр сөзүн алып, MySQLге сактайт. Ал эми Bootstrap ошол формага кооз көрүнүш берет.

2. HTML жана CSSтин негиздери

HTMLдин түзүмү: Тегдер, атрибуттар жана элементтер

HTML (HyperText Markup Language) — гипертексттик белгилөө тили бул веб-баракчалардын негизги түзүлүшүн түзүүчү тил. Веб-сайтты браузерде ачканда, HTML веб-баракчанын бардык тексттерин, сүрөттөрүн жана башка элементтерин көрсөтүүгө жардам берет.

HTML тегдери деген эмне?

Тегдер — гипертексттин негизги түзүүчүлөрү болуп саналат. бул HTML баракчанын элементтерин аныктап турган коддор. Тегдер "ачык" жана "жабык" болуп эки түргө бөлүнөт. Ачык тег элементтин башталышын билдирет, жабык тег болсо элементтин бүтөөрүн билдирет.

Тегдин түзүлүшү:

<тег> Мазмун </тег>

- **Ачык тег**: <тег>
- **Жабык тег**: </тег>

Мисал:

Бул бир параграф.

Бул жерде:

- — бул ачылган тег.
- — бул жабылган тег.
- Тегдин ортосундагы текст ("Бул бир параграф.") — бул мазмун.

HTML атрибуттары деген эмне?

Атрибуттар — бул HTML тегдерине кошумча маалымат берүүчү бөлүк. Атрибуттар ар дайым тегдин ичинде болот жана алар ар кандай кошумча касиеттерди бере алат, мисалы, түстү, өлчөмдү же шилтемени аныктоо үчүн колдонулат.

Атрибуттун түзүлүшү:

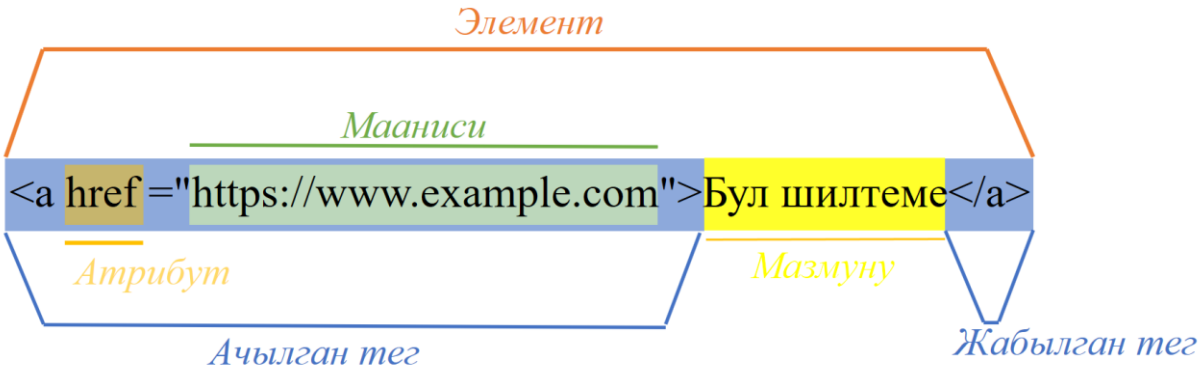
<тег атрибут="маани">Мазмун</тег>

Мисал:

Бул шилтеме

Бул жерде:

- <a> — бул гипершилтеме (link) тег.
- **href="https://www.example.com"** — бул атрибут, ал шилтеменин дарегин көрсөтөт.
- "Бул шилтеме" — бул мазмун, ал экранда чыкчу текст.



HTML элементтери деген эмне?

Элементтер — бул ачылган жана жабылган тегдердин ортосундагы бардык нерсе, анын ичинде текст же башка тегдер. Ар бир HTML теги бир элемент түзөт.

Мисал:

<h1>Саламатсызбы, дүйнө!</h1>

Бул жерде:

- **<h1>** — бул чоң башкы текстти (heading) билдирген тег.
- **Саламатсызбы, дүйнө!** — бул элементтин мазмуну.
- **</h1>** — бул жабылган тег.

Негизги HTML тегдери

Мына HTMLде колдонулган айрым негизги тегдер:

- **<html>**: HTML документтин башталышын жана аягын көрсөтөт.
- **<head>**: Бул тегде мета-маалыматтар жана документтин баштапкы маалыматы сакталат (мисалы, документтин аталышы).
- **<title>**: Веб-баракчанын аталышы, ал браузердин үстүндөгү тилкеде көрсөтүлөт.
- **<body>**: Бул жерде веб-баракчанын негизги мазмуну жайгашкан. Мында тексттер, сүрөттөр, видеолор жана башка элементтер көрсөтүлөт.
- **<p>**: Параграфтарды түзөт.
- **<a>**: Шилтемелерди түзөт.
- **<img>**: Сүрөттөрдү көрсөтөт.
- ** **: Баракчаны бөлүктөргө бөлүп турган тег.
- **<h1> – <h6>**: Башкы тексттер (heading) үчүн колдонулат, <h1> эң чоң, ал эми <h6> эң кичинекей баш текст.

HTML документтин түзүмү

Келгиле, толук HTML документтин түзүлүшүн карап көрөлү:

```
html
<!DOCTYPE html>
<html lang="ky">
<head>
   <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Мисал баракча</title>
</head>
<body>
    <h1>Саламатсызбы, дүйнө!</h1>
    <p>Бул бир параграф.</p>
    <a href="https://www.example.com">Бул шилтеме</a>
</body>
</html>
```

Бул документте:

- **<!DOCTYPE html>** — бул HTML документ экенин билдирген кыстарма.
- **<html>** — HTML документтин башталышы.
- **<head>** — Мета-маалыматтар, аталыштар жана документтин башталышы.
- **<body>** — Баракчанын негизги мазмуну (тексттер, шилтемелер, сүрөттөр).

Мисал:

```
<!DOCTYPE html>
<html lang="ky">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Сиздин Баракчаңыздын Титулу</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Баракчанын Башкы Титулу</h1>
    </header>
```

```
    <nav>
      <ul>
        <li><a href="#home">Башкы</a></li>
        <li><a href="#about">Биз жөнүндө</a></li>
        <li><a href="#contact">Байланыш</a></li>
      </ul>
    </nav>
    <main>
      <section>
        <h2>Секция 1</h2>
        <p>Бул жерде текст же контент болушу мүмкүн.</p>
      </section>
      <section>
        <h2>Секция 2</h2>
        <p>Бул жерде дагы бир текст же контент болушу
мүмкүн.</p>
      </section>
    </main>
    <footer>
      <p>© 2024 Сиздин Атыңыз. Бардык укуктар
корголгон.</p>
    </footer>
</body>
</html>
```

Структуранын Тегдери

1. **<!DOCTYPE html>**: Бул тег HTML5 стандартын колдонуп жаткандыгыңызды билдирет. Ал веб-браузерлерге документтин тибин аныктоого жардам берет.
2. **<html>**: Бул тег HTML документтин башын жана денесин чогулткан тег. lang атрибуту документтин тилин көрсөтөт.
3. **<head>**: Документтин башы. Бул жерде мета маалыматтар, стилдер жана скрипттер жайгашат.

    - **<meta charset="UTF-8">**: Сайттын символдорун аныктайт, бул жерде UTF-8 символдук кодировкасы колдонулат.

- **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Мобилдик түзмөктөргө ылайыкташтыруу үчүн колдонулат.
- **<title>**: Баракчанын аталышын билдирет, бул браузердин табакчасында көрсөтүлөт.
- **<link>**: Стилдик таблицаларды (CSS) байлап, веб-баракчанын сырткы көрүнүшүн жөнгө салат.

4. **<body>**: Документтин негизги мазмуну. Веб-баракчанын көрүнүп турган бөлүгү ушул жерде жайгашат.

    - **<header>**: Баракчанын башкы бөлүгү, адатта, логотип, аталыш жана навигацияны камтыйт.
    - **<nav>**: Навигациялык элементтерди көрсөтүү үчүн колдонулат.
    - **<main>**: Баракчанын негизги контенти. Ар кандай секциялардын (бөлүктөрдүн) болушу мүмкүн.
    - **<section>**: Ар бир секциянын темасын чагылдырган бөлүк.
    - **<footer>**: Баракчанын аягындагы маалымат, автордук укуктар, байланыш маалыматтары жана башка маалыматтар.

HTML тегдерден, атрибуттардан жана элементтерден турат. Ар бир тег HTML документтин түзүлүшүн аныктайт, ал эми атрибуттар жана элементтер менен сайттын мазмунун жана сырткы көрүнүшүн өзгөртүүгө болот.

3. CSS Негиздери: Сырткы көрүнүштү кооздоо жана класстарды колдонуу

CSS (Cascading Style Sheets) — бул веб-баракчанын сырткы көрүнүшүн башкарууга жана кооздоого колдонулган тил. Ал HTML менен иштеп, тексттерди, түстөрдү, фонду, өлчөмдөрдү жана башка көптөгөн визуалдык элементтерди көзөмөлдөйт.

CSS деген эмне жана ал эмнеге керек?

HTML баракчада маалыматты көрсөтсө, **CSS** анын сырткы көрүнүшүн өзгөртүп, ошол маалыматты кооз, уюшкан түрдө берет. CSS веб-баракчанын дизайнын түзүүгө жардам берет, жана ал:

- Шрифттерди, түстөрдү, өлчөмдөрдү өзгөртөт.
- Элементтердин жайгашуусун жана түзүлүшүн контролдойт.
- Колдонуучуга ыңгайлуу жана жандуу дизайн түзүүгө мүмкүндүк берет.

CSSтин негизги синтаксиси

CSSти HTML баракчаларга колдонуу үчүн **стилдик эрежелер** жазылат. Ар бир эреже **селектор** жана **декларациялардан** турат.

CSS синтаксиси:

```
селектор {
  касиет: маани си;
}
```

Мисал:

```
p {
  color: blue;
  font-size: 16px;
}
```

Бул жерде:

- **p** — бул селектор, ал кайсы HTML элементине стиль берилип жатканын көрсөтөт (бул учурда <p> параграф тегине).
- **color: blue;** — бул декларация, ал параграфтын түсүн көк кылат.
- **font-size: 16px;** — бул дагы бир декларация, ал тексттин өлчөмүн 16 пикселге коет.

CSSти HTMLге кошуу ыкмалары

CSSти HTML менен бириктирүүнүн үч негизги ыкмасы бар:

Inline CSS (HTML тегдеринин ичинде):

- Бул стиль бир эле элементке колдонулат. **style** атрибутун колдонуп түз жазылат.

```
<p style="color: red;">Бул кызыл текст.</p>
```

Internal CSS (HTML документтин ичинде):

- HTML документтин башында (head тегинин ичинде) жазылат.

```
<head>
  <style>
    p {
      color: green;
    }
  </style>
</head>
```

External CSS (сырткы файл аркылуу):

- Бул эң көп колдонулган ыкма. Бардык стилдерди өзүнчө CSS файлга жазып, аны HTML документке туташтырып койсо болот.

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

Андан кийин, **styles.css** деген файлга стилдер жазылат:

```
p {
  color: purple;
}
```

Класстарды колдонуу

Класстар — бул көп элементтерге бирдей стилдерди колдонуу үчүн керектелген CSSтин маанилүү бөлүгү. Класс HTML элементке

берилет, жана андан кийин ошол класс үчүн жазылган стиль бардык окшош элементтерге колдонулат.

HTMLде класс колдонуу:

```
<p class="highlight">Бул текст сары фон менен болот.</p>
<p>Бул кадимки текст.</p>
```

CSSте класска стиль берүү:

```
.highlight {
  background-color: yellow;
  color: black;
}
```

Бул жерде:

- **.highlight** — класс селектору. Класс селекторлор ар дайым чекит менен башталат (.).
- Бул стиль кайсы HTML элементке **class="highlight"** берилген болсо, ошого колдонулат.

Идентификаторлорду (ID) колдонуу

Класстар сыяктуу эле, **ID** да элементтерге стилдерди берүү үчүн колдонулат, бирок айырмасы — ID бир баракча ичинде бир эле элементке берилиши керек. Ал уникалдуу.

HTMLде ID колдонуу:

```
<p id="special">Бул текст уникалдуу стиль алат.</p>
```

CSSте IDга стиль берүү:

```
#special {
  color: red;
  font-size: 20px;
}
```

Бул жерде:

- **#special** — бул ID селектору. ID селектор ар дайым тор белгиси менен башталат (#).
- Бул стиль бир гана **id="special"** деген элементке берилет.

Көп колдонулган CSS касиеттери

- **color**: Тексттин түсүн өзгөртөт. color: blue;
- **font-size**: Тексттин өлчөмүн көрсөтөт. font-size: 18px;
- **background-color**: Элементтин арткы (фон) түсүн өзгөртөт. background-color: yellow;
- **margin**: Элемент менен башка элементтердин ортосундагы боштукту белгилейт. margin: 20px;
- **padding**: Элементтин ичиндеги мазмун менен анын четиндеги чек аранын ортосундагы боштукту белгилейт. padding: 10px;
- **border**: Элементтин тегерегине чийилген чек араны көрсөтөт. border: 1px solid black;

CSS менен түзүлгөн мисал

Мисалы, HTML жана CSSти колдонуп, карап көрөлү:

HTML:

```
<!DOCTYPE html>
<html lang="ky">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Мисалы</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Саламатсызбы!</h1>
    <p class="highlight">Бул параграф сары фон менен болот.</p>
    <p>Бул параграф кадимки көрүнүштө болот.</p>
```

```
</body>
</html>
```

CSS (styles.css):

```
h1 {
  color: blue;
  text-align: center;
}

.highlight {
  background-color: yellow;
  padding: 10px;
}

p {
  font-size: 16px;
  color: black;
}
```

Бул мисалда:

- **h1** тегине тексттин түсү көк болуп, борборго жайгаштырылган стиль берилет.
- **highlight** классы менен параграфка сары фон жана 10px боштук кошулат.
- Бардык **p** элементтери кара түстө жана 16px өлчөмүндө болот.

CSS веб-сайттардын көрүнүшүн башкарууга жана аларды колдонуучулар үчүн кооздоого жардам берет. Класстар жана IDлер менен стилдерди ар кандай элементтерге колдонууну жеңилдетсе болот. Туура стилдер менен веб-баракчаны кызыктуу жана функционалдуу кылып көрсөтүүгө мүмкүнчүлүк түзүлөт.

4. Bootstrap Киришүү: Веб-баракчаларды Жасалгалоонун Жөнөкөйлүгү

Bootstrap — бул эң көп колдонулган HTML, CSS жана JavaScriptтен турган алкактар (framework). Ал веб-баракчаларды

жасалгалоону жеңилдетип, адаптивдүү (mobile-first) жана кооз веб-сайттарды тезирээк түзүүгө жардам берет.

Bootstrapти колдонуу менен профессионалдык веб-баракчаларды минималдык CSS жана HTML жазып эле оңой түзсө болот. Анда ар кандай даяр шаблондор, компоненттер жана стилдер бар, мисалы: кнопкалар, формалар, таблицалар, карточкалар, навигациялык панелдер ж.б.

Bootstrap деген эмне жана эмнеге ал маанилүү?

Bootstrap веб-баракчаларды адаптивдүү жана кооз кылып жасоого арналган. Анын негизги артыкчылыктары:

- **Адаптивдүүлүк**: Веб-баракча ар кандай экран өлчөмдөрүндө (компьютер, планшет, телефон) жакшы көрүнөт.
- **Даяр компоненттер**: Bootstrapте көптөгөн даяр элементтер (кнопкалар, формалар, менюлар) бар, аларды жөн гана кошуп койсоңуз, алар иштеп кетет.
- **Жөнөкөйлүк**: Аз код менен көп натыйжа берүүгө жардам берет. Дизайнга көп убакыт коротпой, даяр стилдерди колдонуп ишти тез бүтүрсө болот.

Bootstrapти кантип колдонууга болот?

Bootstrap веб-баракчаны кошуунун эки негизги ыкмасы бар:

1. CDN (Content Delivery Network) аркылуу**.**
2. Bootstrap файлдарын локалдуу түрдө жүктөп алып колдонуу**.**

CDN аркылуу кошуу — бул эң оңой ыкма, анткени файлдарды жүктөөнүн кереги жок. Төмөнкү кодду HTML документиңин <head> тегине кошуу керек.

```
<head>
  <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.
min.css" rel="stylesheet">
```

```
</head>
```

Бул код аркылуу Bootstrapтын стилдерин автоматтык түрдө кошулат. Ал эми JavaScript үчүн төмөнкүдөй код кошуу зарыл:

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.bundle.min.js"></script>
```

Бул ыкма аркылуу Bootstrapтын бардык стилдерин жана компоненттерин дароо колдонсо болот.

Bootstrap менен компоненттерди колдонуу

Bootstrap колдонуп, веб-баракчаларга көптөгөн компоненттерди оңой эле кошсо болот. Төмөндө бир нече мисалды карап көрөлү:

Bootstrap кнопкасы

Bootstrapте кнопкаларды стилдештирүү оңой. Мисалы, төмөнкү код аркылуу ар кандай стилдеги кнопкаларды колдонсо болот:

```
<button class="btn btn-primary">Көк кнопка</button>
<button class="btn btn-warning">Сары кнопка</button>
<button class="btn btn-success">Жашыл кнопка</button>
<button class="btn btn-danger">Кызыл кнопка</button>
```

Бул жерде **btn** — негизги класс, ал эми **btn-primary**, **btn-warning**, **btn-success** сыяктуу класстар ар кандай стилдерди камсыз кылат.

Bootstrap тармак системасы (Grid system)

Bootstrapтын эң чоң артыкчылыктарынын бири — анын адаптивдүү тармак системасы. Ал аркылуу ар кандай экран өлчөмдөрүнө жараша баракчанын түзүлүшүн оңой башкарып, элементтердин туура жайгашуусун камсыздайт.

Мисалы:

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      Биринчи колонка
    </div>
    <div class="col-sm">
      Экинчи колонка
    </div>
    <div class="col-sm">
      Үчүнчү колонка
    </div>
  </div>
</div>
```

Бул мисалда, **container** классы баракчаны ороп алат, ал эми **row** жана **col-sm** класстары баракчанын тармагын (grid) уюштурат. Колонкалар баракчада бирдей жайгаштырылат, жана алар ар кандай экран өлчөмдөрүнө ылайыкташат.

Bootstrap навигациялык меню (Navbar)

Веб-сайтта менюларды жасоону да Bootstrap жөнөкөйлөтөт. Мисалы, төмөнкү код меню түзүү үчүн колдонулат:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Менин сайтым</a>
  <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav"
aria-expanded="false" aria-label="Навигацияны ачуу">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="#">Башкы бет <span class="sr-
only">(азыркы)</span></a>
      </li>
      <li class="nav-item">
```

```
        <a class="nav-link" href="#">Биз жөнүндө</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Байланыш</a>
      </li>
    </ul>
  </div>
</nav>
```

Бул код даяр навигациялык менюну жаратып, ар кандай экрандарда туура ачылып-жабыла турган адаптивдүү менюну камсыз кылат.

Bootstraptын артыкчылыктары

- **Ыкчамдык жана жөнөкөйлүк**: Bootstrap аркылуу код жазуу жеңил жана убакытты үнөмдөйт.
- **Адаптивдүүлүк**: Веб-баракчаңыз автоматтык түрдө ар кандай экран өлчөмдөрүнө ылайыкташат.
- **Даяр элементтер**: Bootstrapте көптөгөн даяр элементтер бар, мисалы: кнопкалар, формалар, модалдар, ж.б.
- **Коопсуздук**: Bootstrap менен түзүлгөн веб-сайттар браузерлердин көпчүлүгүндө жакшы иштейт.

Bootstrap — бул веб-сайттарды иштеп чыгууну оңой жана тез жасоого жардам берген күчтүү курал. Ал адаптивдүү дизайнды түзүүгө, визуалдык элементтерди тез кошууга, жана код жазууну жеңилдетүүгө багытталган. Веб-баракчаларды оңой түзүп, сапаттуу дизайндын үстүндө иштөө үчүн Bootstrap абдан чоң жардамчы болушу мүмкүн.

5. PHP программалоо тили

PHPге Киришүү жана OpenServer орнотуу

PHP (Hypertext Preprocessor) — бул сервер тарапта иштеген тил, ал веб-баракчалардын мазмунун динамикалуу кылууга жардам берет. Мисалы, сиз форма толтурганда, аны серверге жөнөтүп, кайра натыйжаны көрсөтүү үчүн PHP колдонулат. Бул тилдин негизги өзгөчөлүгү — ал серверде иштеп, браузерге даяр HTML кодун жөнөтөт.

PHP деген эмне?

PHP — веб-сайттардын жана веб-тиркемелердин сервер жактагы программасын жазуу үчүн кеңири колдонулат:

- **Динамикалык веб-баракчаларды** түзүү.
- **Маалымат базалары** менен иштешүү (мисалы, MySQL менен).
- Веб формалардан маалыматтарды иштетүү (логин, катталуу ж.б.).
- **Файлдарды жүктөө**, сүрөттөрдү иштеп чыгуу сыяктуу функциялар.

OpenServer деген эмне?

OpenServer — бул Windows чөйрөсүндө веб-сервер орнотуу үчүн колдонулган программа. Анын жардамы менен сиз компьютериңизде Apache, PHP жана MySQLди оңой иштете алса болот. Бул башталгычтар үчүн өз веб-сайттарын жергиликтүү серверде иштеп чыгууну баштоого ыңгайлуу курал.

OpenServer аркылуу PHPни Орнотуу

Төмөндө OpenServerди орнотуп, PHPни колдонуу кадамдары берилген.

1) OpenServerди жүктөп алуу

1. **OpenServerдин расмий сайтына** барып, программаны жүктөп алыңыз.
2. Жүктөп алынган архивди ачып, OpenServerди каалаган папкаңызга орнотуңуз (мисалы, D:\OpenServer).

2) OpenServerди орнотуу жана ишке киргизүү

1. OpenServer орнотулгандан кийин, папкаңызга кирип, Open Server.exe файлын иштетиңиз.
2. Программа иштей баштаганда, төмөнкү оң бурчта OpenServerдин кичинекей кызыл иконасы пайда болот. Аны басып, "Запустить сервер" тандаңыз.

3) PHP тестин жүргүзүү

OpenServer орнотулган соң, PHPнин туура иштеп жатканын текшерсеңиз болот. Бул үчүн OpenServerдин долбоорлор папкасында (domains папкасы) файл түзүп, аны браузерде ачып текшерсеңиз болот.

Тест үчүн кадамдар:

1. OpenServer\domains\ папкасына кирип, жаңы папка түзүңүз (мисалы, myproject.local).
2. Бул папкада index.php деген файл түзүп, төмөнкү кодду жазыңыз:

   ```
   <?php
   echo "PHP ийгиликтүү иштеп жатат!";
   ?>
   ```

3. Браузерди ачып, **http://myproject.local/** (**http://localhost/**) деп жазыңыз. Эгерде PHP туура иштеп жатса, анда "PHP ийгиликтүү иштеп жатат!" деген билдирүүнү көрөсүз.

4) Маалымат базасын (MySQL) кошуу

OpenServer MySQL серверин да камтыйт, бул аркылуу маалымат базалары менен иштей алабыз. MySQLди колдонуу үчүн:

1. OpenServerди иштеткенде, MySQL сервери автоматтык түрдө кошулат.
2. Сиз MySQLди башкаруу үчүн **phpMyAdmin** интерфейсин колдоно аласыз. Браузериңизде **http://localhost/phpmyadmin/** дарегине өтүп, MySQL маалымат базалары менен иштешсеңиз болот.

PHP жана OpenServer — бул башталгыч веб-программисттер үчүн динамикалык веб-сайттарды иштеп чыгуунун эң жеңил жана ыңгайлуу жолу. OpenServerди орнотуу жана колдонуу менен PHPде программалоону баштап, веб-сайттарды локалдуу тексттөөдөн өткөрсө болот.

6. PHPге Киришүү

PHPнин негизги өзгөчөлүктөрү

1. **Сервер тараптагы тил**: PHP коду серверде иштетилет, андан кийин натыйжа браузерге жөнөтүлөт.
2. **Динамикалык контент**: PHP менен веб-баракчалардын мазмунун динамикалык түрдө өзгөртүүгө мүмкүнчүлүк берет. Мисалы, колдонуучунун маалыматын алуу же форма аркылуу маалыматтарды иштетүү.
3. **Маалымат базалары менен иштөө**: PHP MySQL сыяктуу маалымат базалары менен оңой иштешет, бул веб-сайттарда маалыматтарды сактоону жана башкарууну жеңилдетет.
4. **Көп платформалуу**: PHP ар кандай операциялык системаларда (Windows, Linux, Mac) иштейт.

PHP менен Иштөө

1. **PHPны Орнотуу**: PHPди компьютерде же веб-серверде (мисалы, Apache же Nginx) орнотуп, иштетсе болот. OpenServer же XAMPP сыяктуу программалар аркылуу жергиликтүү сервер түзүп, PHP кодун тестирлөөгө мүмкүнчүлүк берет.
2. **PHP Файлы**: PHP кодун жазганда, аны .php кеңейтүүсү менен файлда сакталат. PHP коду <?php ... ?> тегдери арасында жазылат.

   Мисал:

   ```
   <?php
   echo "Салам, дүйнө!";
   ?>
   ```

- **Кодду иштетүү**: PHP файлы веб-сервер аркылуу браузерде ачылганда, сервер PHP кодун иштетип, натыйжасын браузерге жөнөтөт. Жогорудагы мисалда, браузерде "Салам, дүйнө!" деген текст көрсөтүлөт.
- **Форма жана маалымат базасы**: PHP менен веб-формаларды түзүп, колдонуучудан маалыматтарды алып,

аларды маалымат базасында сактоого болот. Мисалы, катталуу формасы аркылуу колдонуучунун маалыматтарын чогултуу.

PHPнин тарыхы

PHP 1995-жылы Rasmus Lerdorf тарабынан түзүлгөн. Ал веб-технологияларга байланыштуу жекече долбоорлорду түзүү үчүн иштелип чыккан. Ушул убакка чейин PHP көптөгөн жаңылыктарды жана жакшыртууларды кабыл алды, азыркы учурда ал веб-долбоорлордун жана CMS (Content Management System) платформаларынын негизги тилдеринин бири болуп калды.

PHP — бул веб-программалоо үчүн күчтүү жана кеңири колдонулган тил. Анын көмөгү менен динамикалык веб-сайттарды, веб-тиркемелерди жана маалымат базалары менен иштөөчү системаларды оңой түзүүгө болот. PHPди үйрөнүү веб-өндүрүштөрдү программалоо процессинде жаңы мүмкүнчүлүктөрдү ачат.

7. PHP синтаксиси жана өзгөрмөлөр

PHP программалоо тили веб-тиркемелерди иштеп чыгууда кеңири колдонулат. Анын синтаксиси жана өзгөрмөлөрүн түшүнүү веб-өндүрүштөрдү эффективдүү жазууга жардам берет.

PHP синтаксиси

PHP коду, адатта, <?php ... ?> тегдери арасында жазылат. Тегдердин ичиндеги код серверде иштетилет, ал эми браузерге жөнөтүлгөндө HTML кодун жаратууга мүмкүнчүлүк берет. PHP кодун жазууда төмөнкү негизги эрежелерди эске алуу керек:

- **Файл кеңейтүүсү**: PHP файлдары адатта .php кеңейтүүсү менен болот.
- **Код блоктору**: PHP кодун <?php жана ?> тегдери арасында жазышыңыз керек. Эгерде HTML менен аралаштырсаңыз, PHP кодун тегдер арасында, HTML кодун болсо тышында жаза аласыз.

Мисал:

```
<!DOCTYPE html>
<html>
<head>
   <title>PHP Синтаксиси</title>
</head>
<body>
   <h1>PHP Иштеп жатат!</h1>
   <?php
   echo "Салам дүйнө!";
   ?>
</body>
</html>
```

Өзгөрмөлөр

PHPдеги өзгөрмөлөр — бул маалыматтарды сактоо үчүн колдонулган аталыштар. Өзгөрмөлөргө маалыматтарды сактоого жана алар менен манипуляциялоого мүмкүнчүлүк берет.

Өзгөрмөнүн синтаксиси

- PHPде өзгөрмөлөр $ белгилеси менен башталат.
- Өзгөрмөлөрдүн аттары баш тамгадан же астынкы шрифттен башталып, алфавиттик символдор же сандар менен уланат. Бирок, сандар өзгөрмөнүн атында биринчи болбойт.

Мисал:

```
$myVariable = "Hello, World!";
$age = 25;
```

Маалымат түрлөрү

PHPде төрт негизги маалымат түрү бар:

1. **Строка (String)**: Тексттик маалыматтар. Строка жагында “же” белгиси менен жазылат.

   Мисал:

```
$greeting = "Салам!";
```

2. **Сан (Integer/Float)**: Сандык маалыматтар. Сан Integer бүтүн сандар, Float болсо ондук сандарды билдирет.

   Мисал:

   ```
   $number = 100; // Integer - бүтүн сандар
   $price = 99.99; // Float - ондук сандар
   ```

3. **Бул (Boolean)**: Логикалык маалыматтар. true же false маанилерин кабыл алат.

   Мисал:

   ```
   $isLoggedIn = true; // true же false
   ```

4. **Массив (Array)**: Көптөгөн маанилерди сактоо үчүн колдонулган маалымат структурасы. Массивдин элементтери, индекстер менен уюштурулат.

   Мисал:

   ```
   $colors = array("кызыл", "жашыл", "көк");
   ```

Өзгөрмөлөр менен жумуш

Өзгөрмөлөрдү түзгөндө, сиз алар менен ар кандай математикалык операцияларды же логикалык операцияларды аткара аласыз.

Мисал:

```
$number1 = 10;
$number2 = 5;
$sum = $number1 + $number2; // $sum = 15

$isTrue = ($number1 > $number2); // $isTrue = true
```

Логикалык операциялар:

```
$isTrue = ($number1 > $number2); // $isTrue = true
```

Маалымат түрлөрүн текшерүү

PHPде өзгөрмөлөрдүн түрүн текшерүү үчүн gettype() функциясын колдонсоңуз болот.

Мисал:

```
$myVar = 42;
echo gettype($myVar); // "integer" деп чыгат
```

PHP синтаксиси жана өзгөрмөлөр — программалоонун негизги концепциялары. Өзгөрмөлөр аркылуу маалыматтарды сактап, ар кандай операцияларды ишке ашырууга мүмкүнчүлүк берет. PHPдин маалымат түрлөрүн түшүнүү, динамикалык веб-тиркемелерди жазууда маанилүү роль ойнойт. PHPни үйрөнүү менен, веб-сайттардын логикасын жана функцияларын жазса болот.

8. PHPдеги математикалык операциялар жана камтылган функциялар

PHP программалоо тили математикалык операцияларды жана функцияларды жеңил аткарууга мүмкүнчүлүк берет. Математикалык операцияларды колдонуу программалоо процессинде маанилүү болуп саналат.

Математикалык операциялар

PHPде бир нече негизги математикалык операцияларды аткарууга болот:

- **Кошуу (+)**: Эки же андан көп санды кошот.
- **Алуu (-)**: Бир санды экинчисинен алып салат.
- **Көбөйтүү (*)**: Эки же андан көп санды көбөйтөт.
- **Бөлүү (/)**: Бир санды экинчисине бөлөт.

- **Модулярдык (%)**: Эки санды бөлгөндөн калган калдыкты эсептейт.

Мисалдар:

```
$a = 10;
$b = 5;

$sum = $a + $b;        // 15
$difference = $a - $b; // 5
$product = $a * $b;    // 50
$quotient = $a / $b;   // 2
$modulus = $a % $b;    // 0
```

Каттылган математикалык функциялар

PHP көптөгөн камтылган математикалык функцияларды сунуштайт, аларды сандар менен ар кандай операцияларды аткарууга пайдаланса болот.

Негизги Математикалык Функциялар:

- **abs()**: Сандын абсолюттук маанисин кайтарат (терс сандардын оң мааниси).
  - Мисал:

```
$value = -5;
echo abs($value); // 5
```

- **pow()**: Сандын даражасын (степень) эсептейт.
  - Мисал:

```
echo pow(2, 3); // 8 (2^3)
```

- **sqrt()**: Сандын квадрат тамырын эсептейт.
  - Мисал:

```
echo sqrt(16); // 4
```

- **round()**: Санды жакынкы бүтүн санга тегиздейт.

- Мисал:

```
echo round(4.7); // 5
```

- **max() жана min()**: Көптөгөн сандын эң чоң же эң кичинекей маанисин кайтарат.
  - Мисал:

```
echo max(10, 20, 5); // 20
echo min(10, 20, 5); // 5
```

- **rand()**: Кездешкен санды берет.
  - Мисал:

```
echo rand(1, 100); // 1ден 100гө чейинки кездешкен сан
```

Мисал: кошумча функциялар менен математикалык операциялар

```
$num1 = 15;
$num2 = 4;

// Кошуу
$sum = $num1 + $num2;

// Бөлүү
$division = $num1 / $num2;

// Квадрат тамыр
$sqrt = sqrt($num1);

// Абсолюттук мааниси
$negative = -10;
$absolute = abs($negative);

// Натижаларды чыгарып берүү
echo "Кошуу: $sum\n"; // Кошуу: 19
echo "Бөлүү: $division\n"; // Бөлүү: 3.75
```

```
echo "Квадрат тамыр: $sqrt\n"; // Квадрат тамыр: 3.872983346207417
echo "Абсолюттук мааниси: $absolute\n"; // Абсолюттук мааниси: 10
```

PHPде математикалык операцияларды аткаруу жана камтылган функцияларды колдонуу, программалоо процессин жөнөкөйлөтөт. Математикалык функциялар менен иштөө, маалыматтарды анализдөө жана веб-тиркемелерде ар кандай операцияларды аткарууга жардам берет.

9. PHPде саптык операциялар

PHP программалоо тилиндеги саптык операциялар тексттик маалыматтар менен иштөө үчүн колдонулат. Саптар — бул символдордун тизмеги, жана PHPде аларды иштетүү үчүн көптөгөн функциялар жана операциялар бар. Төмөндө саптык операцияларды жана алардын колдонулушун карап чыгалы.

Сапты инициализациялоо

Сап түзүү үчүн, аларды " же ' белгиси менен жаза алабыз.

Мисал:

```
$text1 = "Салам, дүйнө!";
$text2 = 'PHP программалоо тили!';
```

Сап кошуу (Конкатенация)

Сап кошуу үчүн . (чекит) символун колдонобуз.

Мисал:

```
$firstName = "Тынчтык";
$lastName = "Асилбеков";
$fullName = $firstName . " " . $lastName; // " Тынчтык Асилбеков "
echo $fullName;
```

Сап узундугун текшерүү

Сап узундугун текшерүү үчүн strlen() функциясын колдонсо болот.

Мисал:

```
$text = "PHP программалоо тили";
$length = strlen($text); // Узундук: 24
echo $length;
```

Сапты кесүү

Саптын бөлүгүн алуу үчүн substr() функциясын пайдаланабыз. Бул функцияга сап, башталыш индекси жана узундук параметрлери берилет.

Мисал:

```
$text = "Салам, дүйнө!";
$substring = substr($text, 0, 5); // "Салам"
echo $substring;
```

Сапты издөө

Саптын ичинде текстти издөө үчүн strpos() функциясы колдонулат. Бул функция, эгер текст табылса, анын позициясын кайтарат, табылбаса — false кайтарат.

Мисал:

```
$text = "Салам, дүйнө!";
$position = strpos($text, "дүйнө"); // 7
if ($position !== false) {
    echo "Текст табылды: позициясы " . $position;
} else {
    echo "Текст табылган жок.";
}
```

Сапты алмаштыруу

Саптын бөлүгүн алмаштыруу үчүн str_replace() функциясы колдонулат. Бул функция издеген текстти жана алмаштырган текстти кабыл алат.

Мисал:

```
$text = "Салам, дүйнө!";
$newText = str_replace("дүйнө", "коом", $text); // "Салам, коом!"
echo $newText;
```

Сапты тазалоо

Саптагы боштуктарды тазалоо үчүн trim() функциясын колдонсо болот. Бул функция саптын башындагы жана аягындагы боштуктарды алып салат.

Мисал:

```
$text = "  Салам, дүйнө!  ";
$cleanText = trim($text); // "Салам, дүйнө!"
echo $cleanText;
```

Сапты чыгаруу

Саптарды чакыруу үчүн explode() функциясы колдонулат. Бул функция саптагы белгилүү бир бөлгүчкө бөлүп, массивке кайтарат.

Мисал:

```
$text = "Салам, дүйнө!";
$words = explode(", ", $text); // ["Салам", "дүйнө!"]
print_r($words);
```

Сапка кошумчалоо

Сапты оңой кошумчалоо үчүн sprintf() функциясын пайдалансаңыз болот. Бул функция саптын форматын аныктоого жардам берет.

Мисал:

```
$greeting = sprintf("Салам, %s!", "Тынчтык");
echo $greeting; // "Салам, Тынчтык!"
```

PHPде саптык операциялар тексттик маалыматтарды иштетүү үчүн маанилүү инструменттерди сунуштайт. Саптарды кошуу, кесүү, издөө жана алмаштыруу функциялары программалоо процессинде кеңири колдонулат.

10. PHP логикалык операторлору жан шарттык конструкцялар (if-else)

PHP программалоо тилинде шарттык конструкциялар логикалык шарттарды текшерүүгө жана алардын негизинде ар кандай операцияларды аткарууга мүмкүндүк берет. Эң негизги шарттык конструкция - if жана else операторлору.

if Конструкциясы

if оператору белгилүү бир шарт чын болсо, код блокторун аткарууга мүмкүндүк берет.

Синтаксис:

```
if (шарт) {
   // Шарт чын болсо аткарылуучу код
}
```

Мисал:

```
$age = 18;

if ($age >= 18) {
   echo "Сиз чоң кишисиз.";
}
```

if-else конструкциясы

else оператору if шарт чын болбогон учурда аткарылуучу код блокторун жазууга мүмкүндүк берет.

Синтаксис:

```
if (шарт) {
   // Шарт чын болсо аткарылуучу код
} else {
   // Шарт чын болбосо аткарылуучу код
}
```

Мисал:

```
$age = 16;

if ($age >= 18) {
   echo "Сиз чоң кишисиз.";
} else {
   echo "Сиз жашсыз.";
}
```

if-elseif-else конструкциясы

Эгерде бир нече шарттарды текшерүү керек болсо, elseif операторун колдонсо болот.

Синтаксис:

```
if (шарт1) {
   // Шарт1 чын болсо аткарылуучу код
} elseif (шарт2) {
   // Шарт2 чын болсо аткарылуучу код
} else {
   // Эгерде шарттардын эч бири чын болбосо аткарылуучу код
}
```

Мисал:

```
$score = 75;

if ($score >= 90) {
   echo "Сиз 'A' алдыңыз.";
} elseif ($score >= 75) {
   echo "Сиз 'B' алдыңыз.";
} elseif ($score >= 60) {
```

```
    echo "Сиз 'C' алдыңыз.";
} else {
    echo "Сиз өткөн жоксуңуз.";
}
```

Логикалык операторлор

PHPде логикалык операторлор шарттарды бириктирүүгө же талкуулоого мүмкүндүк берет. Негизги логикалык операторлор төмөнкүлөр:

- **&& (жана)**: Эгерде экөө тең чын болсо, чын болуп калат.
- **|| (же)**: Эгерде кеминде бири чын болсо, чын болуп калат.
- **! (эмес)**: Шарттын маанисин тескери кылат (эгер шарт чын болсо, ал калпка айланат).

Мисал:

```
$age = 20;
$hasLicense = true;

if ($age >= 18 && $hasLicense) {
    echo "Сиз унаа айдаууга укугуңуз бар.";
} else {
    echo "Сиз унаа айдаууга укугуңуз жок.";
}
```

Мисал:

```
$isRainy = true;
$isSunny = false;

if ($isRainy || $isSunny) {
    echo "Бүгүнкү күндүн абалы жакшы.";
} else {
    echo "Бүгүнкү күндүн абалы начар.";
}
```

PHPде шарттык конструкциялар жана логикалык операторлор программалоо процессинде маанилүү роль ойнойт. Алар маалыматтарды анализдөө, шарттарга жараша ар кандай код

блокторун аткаруу жана логикалык шарттарды текшерүүгө мүмкүнчүлүк берет.

11. Switch-Case операторy

PHPде switch оператору бир нече шартты текшерүү үчүн колдонулат. Бул оператор белгилүү бир маанини салыштырууга жана андан кийин шарттарга жараша ар кандай код блокторун аткарууга мүмкүндүк берет. switch-case оператору көп учурда if-else конструкциясынын ордуна колдонулат, анткени ал кодду таза жана окулушу жеңил кылат.

Синтаксис

```
switch (билдирүү) {
   case маани1:
      // value1 үчүн аткарылуучу код
      break;
   case маани2:
      // value2 үчүн аткарылуучу код
      break;
   // Көптөгөн case блокторун кошууга болот
   default:
      // Эгерде эч кандай case жабылган болсо, аткарылуучу код
}
```

Мисал: Сезонду текшерүү

Төмөндө switch операторунун жөнөкөй мисалы:

```
$month = 4;

switch ($month) {
   case 1:
      echo "Январь";
      break;
   case 2:
      echo "Февраль";
      break;
   case 3:
```

```
        echo "Март";
        break;
    case 4:
        echo "Апрель";
        break;
    case 5:
        echo "Май";
        break;
    case 6:
        echo "Июнь";
        break;
    case 7:
        echo "Июль";
        break;
    case 8:
        echo "Август";
        break;
    case 9:
        echo "Сентябрь";
        break;
    case 10:
        echo "Октябрь";
        break;
    case 11:
        echo "Ноябрь";
        break;
    case 12:
        echo "Декабрь";
        break;
    default:
        echo "Бул ай жок.";
}
```

Жыйынтык: Эгер $month 4 болсо, чыкчу натыйжа: "**Апрель**".

break Оператору

break оператору switch конструкциясынан чыгууну камсыз кылат. Эгерде break колдонулбаса, код case блоктору арасында уланат, бул "fall-through" деп аталат.

Мисал: Fall-Through

```
$day = 6;

switch ($day) {
    case 6:
    case 7:
        echo "Убакыт улуу";
        break;
    case 1:
    case 2:
        echo "Бейшемби";
        break;
    default:
        echo "Күн жок.";
}
```

Жыйынтык: Эгер $day 6 болсо, чыкчу натыйжа: **"Убакыт улуу"**.

Default кызмат көрсөтүү

default блогу, эгерде эч кандай case жабылган болсо, аткарылуучу кодду камтыйт.

Мисал:

```
$fruit = "апельсин";

switch ($fruit) {
    case "алма":
        echo "Сиз алма тандадыңыз.";
        break;
    case "банан":
        echo "Сиз банан тандадыңыз.";
```

```
        break;
    default:
        echo "Сиз белгилүү эмес жемиш тандадыңыз.";
}
```

Жыйынтык: Эгер $fruit "апельсин" болсо, чыкчу натыйжа: **"Сиз белгилүү эмес жемиш тандадыңыз."**.

switch-case оператору PHPде кодду жөнөкөй жана окулушу жеңил кылып жазууга мүмкүндүк берет. Бул оператор, айрыкча, бир нече шарттарды текшерүү үчүн колдонулганда, if-else конструкциясынан жакшыраак иштейт. Ошентип, switch-case оператору динамикалык веб-долбоорлорду түзүүдө пайдалуу болот.

12. Массивдер: бир деңгээлдүү жана көп деңгээлдүү

PHP программалоо тилинде массивдер маалыматтарды уюштурууга мүмкүндүк берет. Массивдер бир же бир нече маанилерди сактоого мүмкүнчүлүк берет жана кодду жөнөкөйлөтөт. Массивдер негизинен эки түргө бөлүнөт: бир жана көп денгээлдүү массивдер.

Бир деңгээлдүү массивдер

Бир деңгээлдүү массив – бул бир өлчөмдүү массив, анда маалыматтар бир тизмеде жайгашкан. Массив элементтери индекстер менен белгиленет, ал индекстер нөлдөн (0) башталат.

Мисал:

```
$colors = array("кызыл", "жашыл", "көк");

// Массив элементтерин көрсөтүү
echo $colors[0]; // кызыл
echo $colors[1]; // жашыл
echo $colors[2]; // көк
```

Массивдин өзгөрмөлөрү

Массивдин элементтерин өзгөртүү мүмкүнчүлүгү бар.

Мисал:

```
$fruits = array("алма", "банан", "апельсин");

// Элементти өзгөртүү
$fruits[1] = "жүзүм";

// Жаңы массивди көрсөтүү
print_r($fruits); // Array ( [0] => алма [1] => жүзүм [2] => апельсин )
```

Көп деңгелүү массивдер

Көп денгээлдүү массив – бул массивдин элементтери дагы массивдер болгон массив. Мындай массивдер маалыматтарды татаал структурада сактоого мүмкүндүк берет, мисалы, таблицаларда же матрицаларда.

Мисал:

```
// Көп денгээлдүү массивдин түзүлүшү
$students = array(
    array("Аян", 20, "мамлекеттик университет"),
    array("Айша", 22, "жеке университет"),
    array("Мирлан", 21, "мамлекеттик университет"),
);

// Массив элементтерин көрсөтүү
echo $students[0][0]; // Аян
echo $students[1][2]; // жеке университет
```

Көп деңгээлдүү массивдерди өзгөртүү

Көп денгээлдүү массивдердин элементтерин өзгөртүү дагы мүмкүн.

Мисал:

```
$products = array(
    array("ноутбук", 1000, "Dell"),
    array("смартфон", 500, "Samsung"),
);
```

```
// Элементти өзгөртүү
$products[0][1] = 900; // Ноутбугунун баасын өзгөртүү

// Жаңы массивди көрсөтүү
print_r($products);
/*
Array (
    [0] => Array ( [0] => ноутбук [1] => 900 [2] => Dell )
    [1] => Array ( [0] => смартфон [1] => 500 [2] => Samsung )
)
*/
```

Массивдерди сактоо жана чыгаруу

PHP массивдерди сактоо жана чыгарыш үчүн print_r() же var_dump() функцияларын колдонууга мүмкүндүк берет.

```
Мисал:
$animals = array("ит", "мышык", "жаныбар");

// Массивди чыгарыңыз
print_r($animals);
/*
Array (
    [0] => ит
    [1] => мышык
    [2] => жаныбар
)
*/
```

PHPдеги массивдер - маалыматтарды уюштуруу жана сактоо үчүн маанилүү курал. Бир денгээлдүү массивдер бир типтеги элементтерди камтыса, көп денгээлдүү массивдер структураланган маалыматтарды сактоого мүмкүндүк берет.

13. Циклдер: for, while жана do while

PHPде циклдер программалык логиканы түзүүгө мүмкүндүк берет, бул тактап айтканда, кайталоо операцияларын аткарууга

жардам берет. Циклдерди колдонуу аркылуу кодду кыскартып, бир нече жолу бирдей операцияларды аткарууга мүмкүндүк алабыз. PHPде негизги үч цикл бар: for, while жана do while.

Цикл for

for циклы белгилүү бир шарттарга негизделген кайталоолорду аткарууга жардам берет. Бул цикл үч бөлүктөн турат: баштоо, шарт текшерүү жана кайталоо.

Синтаксис:

```
for (баштоо; шарт; кайталоо) {
   // аткарылуучу код
}
```

Мисал:

```
// 1ден 5ке чейинки сандардын суммасын эсептөө
$sum = 0;

for ($i = 1; $i <= 5; $i++) {
   $sum += $i; // $sum = $sum + $i
}

echo "Сумма: " . $sum; // Сумма: 15
```

Цикл while

while циклы шарт туура болсо, ошол эле учурда код блокторун аткарууга мүмкүнчүлүк берет. Цикл шарт такталганга чейин уланат.

Синтаксис:

```
while (шарт) {
   // аткарылуучу код
}
```

Мисал:

```
// 1ден 5ке чейинки сандардын суммасын эсептөө
$sum = 0;
$i = 1;

while ($i <= 5) {
```

```
    $sum += $i; // $sum = $sum + $i
    $i++; // $i = $i + 1
}

echo "Сумма: " . $sum; // Сумма: 15
```

Цикл do while

do while циклы шартты текшергенден кийин, биринчи жолу код блокторун аткарууга мүмкүнчүлүк берет. Башкача айтканда, бул цикл эң азында бир жолу аткарылат.

Синтаксис:

```
do {
    // аткарылуучу код
} while (шарт);
```

Мисал:

```
// 1ден 5ке чейинки сандардын суммасын эсептөө
$sum = 0;
$i = 1;

do {
    $sum += $i; // $sum = $sum + $i
    $i++; // $i = $i + 1
} while ($i <= 5);

echo "Сумма: " . $sum; // Сумма: 15
```

Циклдерди токтотуу

Циклдерди токтотуу үчүн break операторун колдонсо болот. Ал циклдан дароо чыгууга мүмкүндүк берет. Ошондой эле continue оператору циклды улантууга мүмкүндүк берет, бирок төмөнкү коду аткарбай, кайра шартты текшерүүгө өтөт.

Мисал:

```
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        break; // 5 жеткенде токтотот
```

```
    }
    echo $i . " "; // 1 2 3 4
}

for ($i = 1; $i <= 10; $i++) {
    if ($i % 2 == 0) {
        continue; // Түз сандарды өткөрүп кетет
    }
    echo $i . " "; // 1 3 5 7 9
}
```

PHPдеги циклдер (for, while, do while) кайталаануучу операцияларды аткарууга мүмкүндүк берет. Алар программа логикасын түзүүгө жана кодду кыскартууга жардам берет. Циклдерди туура колдонуу программалык ишмердүүлүктү жакшыртат.

14. Функциялар

Функциялар – программалоо тилинде кодду модулдаштырып, кайра колдонууга мүмкүнчүлүк берген маанилүү компоненттер. Алар бир же бир нече аргументтерди кабыл алып, кайталап иштетүүгө мүмкүндүк берет. Функцияларды колдонуу кодду жөнөкөйлөтөт, аны оңой түшүнүүгө жана иштетүүгө мүмкүндүк берет.

Функциянын түзүлүшү

Функцияны түзүү үчүн function ачкыч сөзүн колдонобуз. Функциянын структурасы төмөнкүдөй:

```
function функция_аталышы($аргумент1, $аргумент2) {
    // аткарылуучу код
    return $нобуд; // функциянын жыйынтыгын кайтаруу
}
```

Мисал:

```
function кошуу($a, $b) {
    return $a + $b; // $a жана $b суммасын кайтарат
}
```

```
// Функцияны чакыруу
$натыйжа = кошуу(5, 10);
echo "Сумма: " . $натыйжа; // Сумма: 15
```

Функциянын областары

Функциянын областары – бул функциянын ичиндеги жана сыртындагы өзгөрмөлөрдүн көрүнүктүүлүгүн аныктоочу концепция. PHPде локалдык жана глобалдык областтар бар.

- **Локалдык өзгөрмөлөр**: Функциянын ичиндеги өзгөрмөлөр локалдык деп аталат. Алар функциядан тышкары жеткиликтүү эмес.

  Мисал:
  ```
  function эсептөө() {
     $x = 10; // Локалдык өзгөрмө
     return $x * 2;
  }

  echo эсептөө(); // 20
  // echo $x; // Бул жерде $x жеткиликтүү эмес, каталар чыгат.
  ```

- **Глобалдык өзгөрмөлөр**: Функциядан тышкаркы өзгөрмөлөр глобалдык деп аталат. Алар программанын бардык бөлүктөрүндө жеткиликтүү.

  Мисал:
  ```
  $z = 20; // Глобалдык өзгөрмө

  function көбөйтүү() {
     global $z; // Глобалдык өзгөрмөнү колдонуу
     return $z * 3;
  }

  echo көбөйтүү(); // 60
  ```

Функцияларды ичинен функцияларды чакыруу

Функциялар бири-биринин ичинде чакырылышы мүмкүн. Бул функциянын кодуна структуралаштырууга жардам берет.

Мисал:

```
function квадрат($n) {
   return $n * $n;
}

function квадрат_жөнүндө($n) {
   return "Квадрат: " . квадрат($n); // Квадрат функциясын чакыруу
}

echo квадрат_жөнүндө(5); // Квадрат: 25
```

Функциянын жыйынтыгын кайтаруу

Функция аткарылганда, жыйынтыкты кайтаруу үчүн return операторун колдонобуз. Функция чагылганда, return операторунан кийин жазылган код аткарылбайт.

Мисал:

```
function текшерүү($n) {
   if ($n > 0) {
      return "Позитивдүү";
   } elseif ($n < 0) {
      return "Негативдүү";
   } else {
      return "Нул";
   }
}

echo текшерүү (3); // Позитивдүү
```

Функциялар программалоо тилинде кодду модулдаштырууга жана кайра пайдаланууга мүмкүндүк берет. Областары функциянын ичиндеги жана сыртындагы өзгөрмөлөрдүн жеткиликтүүлүгүн

аныктайт. Локалдык өзгөрмөлөр функциянын ичиндеги код менен гана иштешет, ал эми глобалдык өзгөрмөлөр программанын бардык бөлүктөрүндө колдонулушу мүмкүн. Функцияларды туура колдонуу коддун структурасын жакшыртат жана аны оңой түшүнүүгө жардам берет.

15. Файлдарды динамикалык түрдө кошуу

PHPде файлдарды динамикалык түрдө кошуу – бул программалык коддун структурасын жана уюштурулушун жакшыртууга жардам берет. Бул метод веб-баракчалардын кодун бөлүп, модулдаштырууга мүмкүндүк берет. Динамикалык файлдарды кошуунун эки негизги методун карап көрөлү: include жана require.

include оператору

include оператору менен белгиленген файлды коддун аткарылышы учурунда кошууга мүмкүнчүлүк берет. Эгер файл табылбаса, PHP катаны көрсөтпөйт, код уланат.

Мисал:

```
// Файлды кошуу
include 'header.php';

// Баракчанын мазмуну
echo "<h1>Мазмун</h1>";

// Тагын файлды кошуу
include 'footer.php';
```

Эгер header.php файлы жок болсо, PHP катаны көрсөтпөйт, бирок footer.php файлы аткарылат.

require оператору

require оператору да файлды коддун аткарылышы учурунда кошот, бирок эгер файл табылбаса, PHP катаны токтотот. Бул методдун жардамында, программалык логикаңызды ишенимдүү кылууга болот.

Мисал:

```
// Файлды кошуу
require 'config.php';

// Баракчанын мазмуну
echo "<h1>Сайт</h1>";

// Тагын файлды кошуу
require 'functions.php';
```

Эгер config.php файлы жок болсо, программа токтоп, катаны көрсөтөт.

include_once жана require_once

Эгерде файл бир нече жолу кошулса, include_once жана require_once операторлору колдонулат. Бул операторлор файлды бир жолу гана кошууга мүмкүндүк берет, бул коддун кайталануусун алдын алууга жардам берет.

Мисал:

```
// Файлды бир жолу гана кошуу
include_once 'functions.php';
include_once 'functions.php'; // Бул файл кайра кошулбайт
```

Файлдарды динамикалык кошуу программалык логикаңызды жакшыраак уюштурууга мүмкүндүк берет. Мисалы, компоненттерди (жана ошол компоненттерди колдонгон) ортосундагы байланышты кыйла жеңилдетип, кодду окууга оңой кылып, коддун түзүмүн жөнөкөйлөтүүгө мүмкүнчүлүк берет.

```
// Файлдарды динамикалык кошуу
$page = 'about'; // Динамикалык баа
include $page . '.php'; // about.php файлынан мазмунду кошуу
```

Файлдарды динамикалык түрдө кошуу – бул PHPде кодту модулдаштыруу жана рефакторинг кылуу үчүн пайдалуу метод. include жана require операторлору файлдарды кошууга мүмкүндүк

берет, ал эми include_once жана require_once операторлору файлдарды бир жолу гана кошуу үчүн колдонулат.

16. Формаларды иштетүү: Маалыматтарды POST жана GET методтору менен берүү.

Формаларды иштетүү веб-дизайнда жана программалоодо маанилүү компонент болуп саналат. Ал колдонуучулардан маалыматтарды жыйноого, аларды серверге жөнөтүүгө жана аларды кайра иштетүүгө мүмкүндүк берет. PHPде формалар аркылуу маалыматтарды өткөрүү үчүн эки негизги ыкма колдонулат: **GET** жана **POST**.

Форманы түзүү

Форманы түзүү үчүн HTML тилин колдонобуз. Форманын элементтери: тексттик талаалар, кнопкалар, радиобатырмалар ж.б. болуп саналат.

Мисал:

```
<form action="process.php" method="POST">
   <label for="name">Аты:</label>
   <input type="text" id="name" name="name" required>

   <label for="email">Электрондук почта:</label>
   <input type="email" id="email" name="email" required>

   <input type="submit" value="Жөнөтүү">
</form>
```

Бул формада колдонуучудан атын жана электрондук почтасын сурайт. Форманын маалыматтары process.php файлына жөнөтүлөт.

GET ыкмасы

GET ыкмасы форманын маалыматтарын URL аркылуу жөнөтөт. Бул ыкма көбүнчө маалыматтарды издөөгө же колдонуучунун маалыматын көрсөтүүгө ылайыктуу.

- **Синтаксис**: URL формасында көрүү.

- **Код**: URL ичинде маалыматтарды өткөрөт (мисалы, process.php?name=John&email=john@example.com).
- **Кимге ылайыктуу**: Кичине, жөнөкөй маалыматтарды өткөрүү үчүн.
- **Коопсуздук**: Сырсөз же сезимдүү маалыматтарды өткөрүү үчүн туура эмес.

Мисал:

HTML формасында method="GET" колдонгон учурда:

```
<form action="process.php" method="GET">
   <label for="name">Аты:</label>
   <input type="text" id="name" name="name" required>

   <input type="submit" value="Издөө">
</form>
```

PHP кодунда маалыматтарды алуу:

```
$name = $_GET['name'];
echo "Сиздин атыңыз: " . htmlspecialchars($name);
```

POST ыкмасы

POST ыкмасы форманын маалыматтарын HTTP денгээлде жөнөтөт, URL ичинде көрүнбөйт. Бул ыкма көбүнчө маалыматтарды жүктөө жана жаңыртуу үчүн колдонулат.

- **Синтаксис**: HTTP денгээлде.
- **Кимге ылайыктуу**: Көп маалыматтарды өткөрүү, файлдарды жүктөө, сезимдүү маалыматтарды өткөрүү үчүн.
- **Коопсуздук**: URL ичинде көрүнбөгөндүктөн, коопсузураак.

Мисал:

HTML формасында method="POST" колдонгон учурда:

```
<form action="process.php" method="POST">
   <label for="name">Аты:</label>
```

```
    <input type="text" id="name" name="name" required>

    <input type="submit" value="Жөнөтүү">
</form>
```

PHP кодунда маалыматтарды алуу:

```
$name = $_POST['name'];
echo "Сиздин атыңыз: " . htmlspecialchars($name);
```

Эки ыкманы тең колдонгон учурда, форманы кандай маалыматтарды өткөрүү керектигин аныктоо үчүн method атрибутун пайдаланабыз.

Формаларды иштетүү – веб-дизайнда маанилүү процесс. **GET** ыкмасы маалыматтарды URL аркылуу жөнөтүүгө, ал эми **POST** ыкмасы HTTP денгээлде жөнөтүүгө мүмкүнчүлүк берет. Формалардын спецификасына жараша, туура ыкманы тандоо маанилүү. Эгер сырдуу маалыматтарды жөнөтүлүп жатса, **POST** ыкмасын колдонуу сунушталат, ал эми жөнөкөй маалыматтарды көрсөтүү үчүн **GET** ыкмасын пайдаланса болот.

17. Убакыт менен иштөө

PHP программалоо тили даталарды жана убакытты иштетүү үчүн күчтүү функциялар менен камсыз кылат. Даталар жана убакыттар менен иштөө веб-программаларды курууда абдан маанилүү, анткени көпчүлүк тиркемелерде убакыт жана дата маанилүү ролду ойнойт.

Датаны жана убакытты алуу

PHPде датаны жана убакытты алуу үчүн date() функциясын колдонобуз. Бул функция күндүн жана убакыттын форматындагы маалыматтарды чыгарат.

Мисал:

```
// Бүгүнкү датаны жана убакытты алуу
$currentDateTime = date('Y-m-d H:i:s');
```

echo "Бүгүнкү дата жана убакыт: " . $currentDateTime;

Формат:

- Y – төрт сан менен жыл (2024)
- m – эки сан менен ай (01-12)
- d – эки сан менен күн (01-31)
- H – 24 саат форматында саат (00-23)
- i – мүнөт (00-59)
- s – секунд (00-59)

Датаны жана убакытты форматтоо

Эгер дата жана убакытты белгилүү форматка келтирүү кере болсо, date() функциясындагы форматтын символдорун пайдаланууга болот.

Мисал:
// Датаны форматы
$formattedDate = date('d.m.Y'); // 31.12.2024
echo "Форматталган дата: ". $formattedDate;

Убакыт зонасы

PHPде убакыт зонасын конфигурациялоо үчүн date_default_timezone_set() функциясын колдонсо болот. Бул функция программаны кайсы убакыт зонасында иштейт экенин аныктайт.

Мисал:
// Убакыт зонасын белгилөө
date_default_timezone_set('Asia/Bishkek'); // Бишкек убакыт зонасы

$currentDateTime = date ('Y-m-d H:i:s');
echo "Бүгүнкү дата жана убакыт: ". $currentDateTime;

Убакытты салыштыруу

Убакытты салыштырууда strtotime() функциясын колдонсок болот. Бул функция датаны Unix timestamp форматына конверттейт, ал эми андан соң салыштырууну жүргүзүүгө мүмкүндүк берет.

Мисал:

```
$date1 = strtotime('2024-10-11');
$date2 = strtotime('2024-11-01');

if ($date1 < $date2) {
   echo "Биринчиси экинчисинен эрте.";
} else {
   echo "Экинчи дата биринчи датадан эрте.";
}
```

Убакытты кошуу жана кемитүү

PHPде даталарды кошуу же жоюу үчүн DateTime классын пайдаланууга болот. Бул класс даталарды жана убакытты жөнөкөй жана интуитивдүү жол менен иштетүүгө мүмкүндүк берет.

Мисал:

```
// Жаңы DateTime объектиси
$date = new DateTime('2024-10-11');

// 10 күн кошуу
$date->modify('+10 days');
echo "10 күндөн кийин: " . $date->format('Y-m-d'); // 2024-10-21

// 15 күн жоюу
$date->modify('-15 days');
echo "15 күндөн кийин: " . $date->format('Y-m-d'); // 2024-10-06
```

Датаны убакытка конверттөө

Кээде датаны убакытка конверттөө керек болушу мүмкүн. Бул үчүн DateTime классынын getTimestamp() методун колдонсо болот.

Мисал:

```
$date = new DateTime('2024-10-11');
$timestamp = $date->getTimestamp();
echo "Unix timestamp: " . $timestamp;
```

PHPде даталарды жана убакыттарды иштетүү программаларды өнүктүрүүдө маанилүү. date() функциясы, DateTime классы жана strtotime() функциясы даталарды алуу, форматтоо, салыштыруу, кошуу жана жоюу үчүн колдонулат. Убакыт зоналарын жөндөп, даталарды туура иштетүү менен веб-тиркемелериңиздин натыйжалуулугун жогорулатууга болот.

18. PHPнин функциялары

PHP программалоо тили бир нече пайдубал функцияларды, же тактап айтканда, **встроенные функции** сунуштайт. Бул функциялар кодду жөнөкөйлөтүп, кээ бир жалпы тапшырмаларды аткарууга жардам берет. PHPдеги встроенные функциялар ар кандай категорияларга бөлүнөт, мисалы, текст, массив, дата жана убакыт, жана математикалык функциялар. Төмөндө бул функциялардын айрымдарына мисалдар келтирилет.

Тексттик функциялар

a. strlen()

Берилген тексттин узундугун эсептейт.

```
$text = "Салам, дүйнө!";
$length = strlen($text);
echo "Тексттин узундугу: " . $length; // 14
```

b. str_replace()

Тексттин ичindeki белгилүү символду башка символго алмаштырат.

```
$text = "Салам, дүйнө!";
$newText = str_replace("дүй", "чөй", $text);
echo $newText; // Салам,чүйнө!
```

c. substr()

Текстгин белгилүү бир бөлүгүн алат.

```
$text = "Салам, дүйнө!";
$substring = substr($text, 0, 5);
echo $substring; // Салам
```

2. Математикалык Функциялар

a. abs()

Сандык абсолюттук маанини алат.

```
$number = -10;
$absolute = abs($number);
echo "Абсолюттук мааниси: " . $absolute; // 10
```

b. round()

Санды жакындатат.

```
$number = 4.6;
$rounded = round($number);
echo "Жакындатылган сан: " . $rounded; // 5
```

c. rand()

Жалпы диапазондо кездешкен санды түзөт.

```
$randomNumber = rand(1, 100);
echo "Кездешкен сан: " . $randomNumber; // 1ден 100гө чейин
```

Массив функциялар

a. count()

Массивдин элементтеринин санын эсептейт.

```
$array = [1, 2, 3, 4, 5];
$count = count($array);
echo "Элементтердин саны: " . $count; // 5
```

b. array_push()

Массивдин аягына жаңы элемент кошот.

```
$array = [1, 2, 3];
array_push($array, 4);
print_r($array); // [1, 2, 3, 4]
```

c. array_merge()

Эки массивди бириктирет.

```
$array1 = [1, 2, 3];
$array2 = [4, 5, 6];
$mergedArray = array_merge($array1, $array2);
print_r($mergedArray); // [1, 2, 3, 4, 5, 6]
```

Дата жана убакыт функциялар

a. date()

Күндүн жана убакыттын белгилүү форматын кайтарат.

```
$currentDate = date('Y-m-d H:i:s');
echo "Бүгүнкү дата: " . $currentDate; // 2024-10-11 14:00:00
```

b. strtotime()

Тарыхый датаны Unix timestamp форматына конверттейт.

```
$dateString = "2024-10-11";
$timestamp = strtotime($dateString);
echo "Unix timestamp: " . $timestamp; // 1696982400
```

5. Ката Жөнүндө Функциялар

a. error_reporting()

Ката отчеттуулугун жөндөйт.

```
error_reporting(E_ALL); // Бардык каталарды көрсөткөн
```

b. trigger_error()

Ката билдирүүсүн жаратат.

trigger_error("Бул ката", E_USER_WARNING);

PHPнин функциялар программалоо процессин жөнөкөйлөтүп, кодду кайра колдонууга мүмкүнчүлүк берет. Жогорудагы функциялар текст, массив, дата, убакыт жана математикалык операцияларды аткарууга жардам берет. PHPдин мүмкүнчүлүктөрүн кеңейтүү үчүн, ушул функцияларды эффективдүү колдонуу маанилүү.

19. Файлдар менен иштөө

PHP файлдар менен иштөөнүн көптөгөн мүмкүнчүлүктөрүн сунуштайт. Бул функциялар файлдарды окуу, жазуу, жоюу, көчүрүү, жана алардын мазмунун башкарууга жардам берет. Төмөндө PHPде файлдар менен иштөөнүн негизги аспектилери каралат.

1. ***Файлды ачуунун негизги функциялары***

a. fopen()

Файлды ачуу үчүн колдонулат. Ал файлды белгиленген режимде ачат. Режимдер: r (окуу), w (жазуу), a (кошуу), r+ (окуу жана жазуу) ж. б.

```
$filename = 'example.txt';
$file = fopen($filename, 'r'); // Файлды окуу режиминде ачуу
if ($file) {
   echo "Файл ийгиликтүү ачылды.";
} else {
   echo "Файлды ачууда ката пайда болду.";
}
fclose($file); // Файлды жап
```

2. ***Файлдан мазмунду окуу***

a. fgets()

Файлдан бир жардын мазмунун окуу үчүн колдонулат.

```
$file = fopen('example.txt', 'r');
while (($line = fgets($file)) !== false) {
    echo $line . "<br>";
}
fclose($file);
```

b. file_get_contents()

Файлдын толук мазмунун бир жолу окуу үчүн пайдаланылат.

```
$content = file_get_contents('example.txt');
echo $content; // Файлдын толук мазмуну
```

3. Файлга Мазмун Жазуу

a. fwrite()

Файлга мазмун жазуу үчүн колдонулат. Файл биринчи "жазуу" же "кошуу" режиминде ачылышы керек.

```
$file = fopen('example.txt', 'w'); // Жазуу режиминде ачуу
fwrite($file, "Салам, дүйнө!");
fclose($file);
```

b. file_put_contents()

Файлга мазмунду оңой жазуу үчүн колдонулат. Эгер файл бар болсо, анын мазмуну алмаштырылат.

```
file_put_contents('example.txt', "Салам, дүйнө!");
```

4. Файлдарды көчүрүү жана жоюу

a. copy()

Файлды көчүрүү үчүн колдонулат.

```
$source = 'example.txt';
$destination = 'copy_of_example.txt';
if (copy($source, $destination)) {
    echo "Файл ийгиликтүү көчүрүлдү.";
} else {
```

```
    echo "Көчүрүүдө ката пайда болду.";
}
```

b. unlink()

Файлды жоюу үчүн колдонулат.

```
$filename = 'copy_of_example.txt';
if (unlink($filename)) {
    echo "Файл ийгиликтүү жоюлду.";
} else {
    echo "Жоюу процессинде ката пайда болду.";
}
```

5. Файлдын Мазмунун Текшерүү

a. file_exists()

Файлдын бар экенин текшерет.

```
$filename = 'example.txt';
if (file_exists($filename)) {
    echo "Файл бар.";
} else {
    echo "Файл жок.";
}
```

b. is_readable() жана is_writable()

Файлдын окууга же жазууга жарактуу экенин текшерет.

```
if (is_readable($filename)) {
    echo "Файл окууга жарактуу.";
} else {
    echo "Файл окууга жараксыз.";
}

if (is_writable($filename)) {
    echo "Файл жазууга жарактуу.";
} else {
    echo "Файл жазууга жараксыз.";
}
```

6. Файлдын мазмунун тез жазуу

a. fwrite() менен бир нече жырды бир аз жаза аласыз:

```
$file = fopen('example.txt', 'w');
$data = ["Салам,", "Дүйнө!"];
foreach ($data as $line) {
   fwrite($file, $line . PHP_EOL); // PHP_EOL - жаңы жол
}
fclose($file);
```

PHP файлдар менен иштөө үчүн кеңири мүмкүнчүлүктөрдү берет. Файлдарды ачуу, окуу, жазуу, көчүрүү жана жоюу процессин жөнөкөй функциялар аркылуу аткарууга болот. Бул функцияларды туура колдонуу менен, веб-программаларды курууда файлдардын мазмунун натыйжалуу башкарууга мүмкүнчүлүк алууга болот.

20. PHP Функциясы phpinfo() жана массив $_SERVER

Функция phpinfo()

phpinfo() — PHPнин конфигурациясы, версиясы, орнотуу параметрлери жана сервер жөнүндө маалыматтарды көрсөтүүчү функция. Бул функция программалоочуларга PHP орнотууларынын статусун текшерүүгө жана көйгөйлөрдү чечүүгө жардам берет.

Мисал:

```
<?php
phpinfo();
?>
```

Эгер жогорудагы кодду PHP файлы катары серверге жайгаштырса, браузерде PHP орнотуулары жөнүндө толук маалыматты көрсөтө турган бет ачылат. Албетте, бул функцияны түздөн-түз өндүрүштүк чөйрөдө колдонуу сунушталбайт, анткени ал коопсуздук маселелерин жаратышы мүмкүн. Аны текшерүү же диагностика үчүн гана колдонуу керек.

Массив $_SERVER

$_SERVER — глобалдык массив, ал веб-сервер тарабынан автоматтык түрдө түзүлөт жана веб-сервердин аткарылышы жөнүндө маалыматтарды камтыйт. Бул массивдеги элементтер ар кандай сервер жана клиенттин маалыматтарын камтыйт, мисалы, HTTP башталыштары, файлдардын жолдору, сервердин аталышы жана башкалар.

Массивдин негизги элементтери:

- $_SERVER['HTTP_USER_AGENT']: Браузер жана операциялык система жөнүндө маалымат.
- $_SERVER['REMOTE_ADDR']: Кызматты колдонгон колдонуучунун IP-дареги.
- $_SERVER['REQUEST_METHOD']: HTTP ыкмасы (GET же POST).
- $_SERVER['SCRIPT_NAME']: Актуалдуу скрипттин аталышы.
- $_SERVER['SERVER_NAME']: Сервердин аталышы.
- $_SERVER['SERVER_PORT']: Сервердин порту.

Мисал:

```
<?php
echo "Сиздин IP дарегиңиз: " . $_SERVER['REMOTE_ADDR'] . "<br>";
echo "Сиздин браузеринин аталышы: " . $_SERVER['HTTP_USER_AGENT'] . "<br>";
echo "HTTP ыкмасы: " . $_SERVER['REQUEST_METHOD'] . "<br>";
echo "Скрипттин аты: " . $_SERVER['SCRIPT_NAME'] . "<br>";
?>
```

phpinfo() функциясы жана $_SERVER массиви PHP программалоосундагы маанилүү компоненттер болуп саналат. phpinfo() функциясы сервердин конфигурациясын текшерүүгө мүмкүнчүлүк берет, ал эми $_SERVER массиви веб-сервердин маалыматтарын алып, веб-приложенияларды өнүктүрүүнү

жеңилдетет. Бул инструменттерди туура колдонуу программалоо процессин натыйжалуу кылууга жардам берет.

21. Куки жана сессиялар

PHPде кукилер жана сессиялар веб-приложенияларда колдонуучунун маалыматтарын сактоонун маанилүү ыкмалары болуп саналат. Алар колдонуучунун маалыматын же кардардын статусун түзүүнү жана башкарууну жеңилдетет.

1. Куки (Cookies)

Кукилер — бул колдонуучунун браузерине сакталган маалыматтын кичинекей файлдары. Алар сайттан сайтка өтүү учурунда колдонулуучу маалыматтарды сактоо үчүн колдонулат, мисалы, колдонуучунун тандап алган тили, темасы, же аутентификация маалыматы.

Кукини түзүү

Кукини түзүү үчүн setcookie() функциясын колдонсоңуз болот.

Синтаксис:
setcookie(name, value, expire, path, domain, secure, httponly);

- name: Кукинин аталышы.
- value: Кукинин мазмуну.
- expire: Кукинин мөөнөтү (timestamp форматында).
- path: Куки колдонулуучу жол.
- domain: Куки жарактуу домен.
- secure: (тандап алсаңыз) HTTPS аркылуу гана жеткиликтүү болсо true.
- httponly: (тандап алсаңыз) JavaScript аркылуу жеткиликтүү болбойт.

Мисал:
```
<?php
// Кукини түзүү
```

```
setcookie("user", "John Doe", time() + (86400 * 30), "/"); // 30 күнгө сакталат

if(isset($_COOKIE["user"])) {
    echo "Салам, " . $_COOKIE["user"];
} else {
    echo "Сиз кукини кабыл алган жоксуз.";
}
?>
```

Кукини жоюу

Кукини жок кылуу үчүн, анын мөөнөтүн өткөн убакытка коюу керек.

```
setcookie("user", "", time() - 3600, "/"); // Кукини жок кылуу
```

2. Сессиялар (Sessions)

Сессиялар — колдонуучунун серверде сакталган маалыматтары. Сессия, адатта, серверде сакталган уникалдуу идентификатор (сессия ID) аркылуу башкарылат. Сессиялар, кукилерден айырмаланып, серверде колдонуучуга тиешелүү маалыматтарды сактоо үчүн колдонулат.

Сессияны баштоо

Сессияны баштоо үчүн session_start() функциясын колдонушуңуз керек.

```
<?php
session_start(); // Сессияны баштоо

// Сессияга маалыматты сактоо
$_SESSION["user"] = "John Doe";

// Сессиядан маалыматты алуу
echo "Салам, " . $_SESSION["user"];
?>
```

Сессияны жоюу

Сессияны жоюу үчүн, session_destroy() функциясын колдонсоңуз болот. Бирок, бул функцияны колдонуу алдында, сессиянын маалыматтарын жок кылуу керек.

```
session_start(); // Сессияны баштоо
session_unset(); // Сессиядагы маалыматтарды жок кылуу
session_destroy(); // Сессияны жок кылуу
```

3. Кукилердин жана сессиялардын эришүүсү

- **Кукилер**: Колдонуучунун браузерине сакталат жана колдонулуп жаткан сайттан тышкары башка сайттарга жеткиликтүү болушу мүмкүн. Алар браузерди жапканда да сакталат, эгерде мөөнөтү белгиленсе.
- **Сессиялар**: Серверде сакталат жана сессияны жапканда же жабылганда жоюлат. Алар коопсузураак, анткени колдонуучунун маалыматтары серверде сакталат.

Кукилер жана сессиялар PHPде колдонуучунун маалыматтарын сактоонун маанилүү механизмдери болуп саналат. Кукилер браузерде сакталса, сессиялар серверде сакталат. Алар веб-приложенияларда колдонуучунун тажрыйбасын жакшыртуу үчүн кеңири колдонулат.

22. MySQL жана OpenServerге киришүү

1. MySQL негиздери

MySQL — бул ачык коддуу релятивдүү маалымат базасынын башкаруу системасы (RDBMS). Ал веб-приложенияларды жана маалыматтарды сактоо үчүн кеңири колдонулат. MySQL структураланган маалыматтарды (буюмдар, таблицалар, поля) колдонуу менен иштейт. Релятивдүү маалымат базасында маалыматтар таблицаларда (буюмдарда) сакталат жана таблицалар ортосундагы байланыштар негизинде уюштурулат.

MySQLнин негизги өзгөчөлүктөрү:

- **Структураланган Сурау тил (SQL)**: MySQL SQL (Structured Query Language) тилин колдонуучуга маалымат базасын башкаруу үчүн.
- **Коопсуздук**: Пайдаланучулар жана роль негизиндеги коопсуздук системасын колдонуучуга берүү.
- **Жогорку түзүмдүүлүк**: Базадагы маалыматтарды натыйжалуу башкаруу.
- **Көңүлдүү**: Ушул убакка чейин жогорку масштабда маалымат базаларын иштеп чыгуу мүмкүнчүлүгү.

MySQLди колдонуу

1) **MySQLди баштоо**: OpenServerди иштеткенден кийин, MySQL серверин активдештириңиз. OpenServerдин интерфейсинен "MySQL" бөлүмүн табыңыз жана аны активдештириңиз.
2) **phpMyAdminди ачуу**: MySQL маалымат базасын башкаруу үчүн phpMyAdmin колдонсоңуз болот. OpenServer интерфейсинен "phpMyAdmin" опциясын тандап, браузерде ачып, MySQL базасыңыз менен иштешиңиз.
3) **Маалымат базасын жасоо**: phpMyAdmin аркылуу жаңы маалымат базасын түзүңүз жана таблицаларды жазыңыз.

MySQL SSQL запросторду жазуу

MySQL маалымат базасына SQL запрос жазып, маалыматтарды башкарууга болот. Мисалы, таблица түзүү, маалымат кошуу, жана маалыматтарды алуу үчүн SQL командасын колдонуңуз.

```
CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
   password VARCHAR(255) NOT NULL
);

-- Маалымат кошуу
INSERT INTO users (username, password) VALUES ('Tynchtyk', 'password123');
```

```
-- Маалымат алуу
SELECT * FROM users;
```

MySQL жана OpenServer веб-приложениени иштеп чыгууда маанилүү куралдар болуп саналат. OpenServer жергиликтүү серверди орнотуу жана башкаруу үчүн жеңил жана ыңгайлуу, ал эми MySQL маалымат базасын натыйжалуу башкарууга мүмкүндүк берет. Бул куралдарды биргеликте колдонуу веб-приложениени жазууда өзгөчө жардам берет.

PhpMyAdmin: негизги маалымат

PhpMyAdmin — бул веб-негиздеги MySQL маалымат базасын башкаруу куралы. Ал PHP тилинде жазылган жана маалымат базалары менен оңой иштешүүгө мүмкүндүк берет. PhpMyAdmin аркылуу колдонуучулар MySQL базасын түзүү, өзгөртүү, маалымат кошуу, чыгаруу, резервдик көчүрмөлөрдү алуу жана башка көптөгөн функцияларды аткара алышат.

PhpMyAdmindин негизги функциялары

1. **Маалымат базасын башкаруу**:
    - Жаңы маалымат базаларын түзүү.
    - Маалымат базаларын жана таблицаларды өзгөртүү, жок кылуу, көчүрүү.
2. **Таблицалар менен иштөө**:
    - Жаңы таблицаларды түзүү, маалымат кошуу.
    - Таблицалардын структурасын өзгөртүү (поля, типтер, индекстер).
3. **SQL Сураулары**:
    - SQL суроолорун түзүп, аткаруу (SELECT, INSERT, UPDATE, DELETE).
    - Сурау жыйынтыктарын визуалдаштыруу.
4. **Маалыматтарды экспорттоо жана импорттоо**:

- Маалыматтарды CSV, SQL же XML форматында экспорттоо.
- Маалыматтарды CSV, SQL же XML форматынан импорттоо.

5. **Резервдик көчүрмөлөр**:

   - Маалымат базасынын резервдик көчүрмөсүн алуу жана калыбына келтирүү.

6. **Пайдалануу жөндөмдүүлүгү**:

   - Колдонуучуларды жана алардын уруксатын башкаруу.
   - Маалымат базасын түзүүдө коопсуздукту камсыз кылуу.

7. **Фильтрация жана сорттоо**:

   - Жыйынтыктарды фильтрациялоо жана сорттоо.
   - Жыйынтыктарды группалоо жана агрегациялоо.

8. **Графикалык интерфейс**:

   - Колдонуучуга түшүнүктүү жана ыңгайлуу графикалык интерфейс.
   - Колдонуучулар үчүн интуитивдик навигация.

PhpMyAdminдин мүмкүнчүлүктөрү

1. **Масштабдолгон чечимдер**: Улуу проекттер үчүн жана көп сандагы маалыматтар менен иштөө мүмкүнчүлүгү.
2. **Эркин долбоор**: Ачык коддуу программа, бул анын бекем жана кеңейүүчү болушун камсыз кылат.
3. **Коопсуздук**: Пароль менен корголгон аккаунттар жана HTTPS колдоо аркылуу коопсуздук.
4. **Кеңейтилүүчү плагиндер**: PhpMyAdmin үчүн ар кандай плагиндерди колдонуп, функционалдуулугун кеңейтүү мүмкүнчүлүгү.

PhpMyAdmin — бул MySQL маалымат базасын башкарууну жөнөкөйлөтүүчү курал. Ал колдонуучуларга маалымат базаларын түзүү, өзгөртүү, маалыматтарды башкаруу жана резервдик көчүрмөлөрдү алуу үчүн керек болгон ар тараптуу функцияларды сунуштайт. Веб интерфейси жана SQL тилдери аркылуу маалымат базасы менен иштөө оңой жана ыңгайлуу.

23. Маалымат базасын жана таблицаны түзүү

MySQLда жаңы маалымат базасын жана таблицаларды түзүү үчүн төмөнкү кадамдарды аткарыңыз.

1. PhpMyAdminге Кириңиз

1. **PhpMyAdmin'ди ачыңыз**: Браузериңизде http://localhost/phpmyadmin дарегин киргизиңиз.
2. **Кирүү**: Эгерде сизде логин жана пароль болсо, аларды киргизиңиз. Эгерде сиз OpenServer колдонуп жатсаңыз, адатта, логин root жана пароли бош (жок).

2. Жаңы маалымат базасын түзүү

1. **Маалымат базасын түзүү**:
    - PhpMyAdmin интерфейсинин негизги бетинде «Создать» (Create) же «Новая база данных» (New Database) секциясына басыңыз.
    - **Атын киргизиңиз**: Жаңы маалымат базасыңызга ат бериңиз. Мисалы, my_database.
    - **Кодировканы тандоо**: Кодировка үчүн utf8_general_ci же utf8mb4_general_ci тандаңыз.
    - **Түзүү** (Create) кнопкасын басыңыз.

3. Жаңы таблицаны түзүү

1. **Маалымат базасын тандаңыз**: Жаңы түзүлгөн маалымат базасын тандаңыз (сол жактагы панелден).
2. **Таблица түзүү**:

- «Создать таблицу» (Create table) секциясына өтүңүз.
- **Атын киргизиңиз**: Жаңы таблицаңызга ат бериңиз. Мисалы, users.
- **Тилке санын киргизиңиз**: Мисалы, 3.
- **Создать** (Create) кнопкасын басыңыз.

3. **Тилкелерди кошуу**:

Таблица үчүн керектүү тилкелердин (структураны) түзүңүз. Мисалы:

1. id — INT, AUTO_INCREMENT, PRIMARY KEY.
2. username — VARCHAR(50), NOT NULL.
3. password — VARCHAR(255), NOT NULL.
4. Ар бир поле үчүн маалымат типин жана мүнөздөмөлөрдү (NOT NULL, UNIQUE ж.б.) тандаңыз.
5. **Сактоо** (Save) кнопкасын басыңыз.

4. Маалымат кошуу

1. **Таблицаны тандаңыз**: Түзүлгөн таблицаны тандаңыз.
2. **Кошуу**: «Вставить» (Insert) секциясына өтүңүз.
3. **Маалымат киргизиңиз**: Полялар боюнча маалыматтарды киргизиңиз.

Мисалы:
username: tynchtyk
password: password123

4. **Сактоо** (Save) кнопкасын басыңыз.

5. Маалыматты текшерүү

1. **Таблицаны тандаңыз**: Кошулган таблицаны тандаңыз.
2. **Текшерүү**: «Просмотр» (Browse) секциясына өтүңүз. Бул жерден киргизилген маалыматты көрө аласыз.

Таблицага маалымат кошуу

MySQL таблицасына маалымат кошуу үчүн PhpMyAdmin интерфейсин колдонуунулат. Бул процессти аткаруу үчүн төмөнкү кадамдарды аткаруу керек:

1. PhpMyAdmin'ге кириңиз

- Браузериңизде http://localhost/phpmyadmin дарегин киргизип, кирүү маалыматтарыңызды киргизиңиз.

2. Маалымат базасын тандаңыз

- Сол жактагы панелде, маалымат базалары тизмесинен маалымат базасын тандаңыз, мисалы, my_database.

3. Таблицаны тандаңыз

- Тандалган маалымат базасындагы таблицаларды көрүү үчүн, таблицалар тизмесинен маалымат кошууну каалаган таблицаны тандаңыз. Мисалы, users.

4. Кошуу (Insert) секциясына өтүңүз

- Таблица бетине киргенден кийин, жогорку панелде «Вставить» (Insert) кнопкасына басыңыз. Бул сизге таблицага маалымат кошуу формасын көрсөтөт.

5. Маалыматтарды киргизиңиз

- Кошууну каалаган поляларга маалыматтарды киргизиңиз. Мисалы:
  username: tynctyk
  password: password123
- Эгерде сиздин таблицаңызда башка тилке болсо, ошол тилке үчүн да маалымат киргизиңиз.

6. Сактоо (Save)

- Маалыматтарды киргизгенден кийин, форманын төмөнкү жагында жайгашкан «Сохранить» (Save) кнопкасына басыңыз. Бул маалыматтарды таблицага кошот.

7. Маалыматтарды текшерүү

- Маалыматтар таблицага ийгиликтүү кошулганын текшерүү үчүн, «Просмотр» (Browse) секциясына өтүңүз. Бул жерде кошулган маалыматтарды көрө аласыз.

24. SQL запрос аркылуу маалымат кошуу

SQL тилин колдонуп маалымат кошуу үчүн, төмөнкү SQL запросту аткаруу керек:

INSERT INTO users (username, password) VALUES ('Tynchtyk', 'password123');

1. PhpMyAdmin интерфейсинде жогорку панелдеги «SQL» секциясына өтүңүз.
2. Жогорудагы SQL суроосун киргизип, «Выполнить» (Execute) кнопкасына басыңыз.

MySQL командалары – бул SQL (Structured Query Language) тилинде жазылган командалар, аларды маалымат базасы менен иштөө үчүн колдонобуз. Төмөндө эң кеңири колдонулган MySQL командаларын жана алардын түшүндүрмөлөрүн мисалдар менен көрсөтүлгөн.

Маалымат базасын жаратуу

CREATE DATABASE my_database;

Түшүндүрмө: Жаңы маалымат базасын түзөт. my_database – маалымат базасынын аты.

Маалымат базасын өчүрүү

```
DROP DATABASE my_database;
```

Түшүндүрмө: Белгиленген маалымат базасын өчүрөт. Эскертүү: Бул команда бардык таблицаларды жана маалыматтарды жок кылат.

Таблицаны жаратуу

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Түшүндүрмө: users аттуу таблицаны түзөт. Бул таблицада id, username жана password деген поляр бар. AUTO_INCREMENT – бул идентификатор автоматтык түрдө өскөн сан.

Таблицаны Өчүрүү

```
DROP TABLE users;
```

Түшүндүрмө: Белгиленген таблицаны өчүрөт.

Маалымат Кошуу

```
INSERT INTO users (username, password) VALUES ('Tynchtyk', 'password123');
```

Түшүндүрмө: users таблицасына жаңы маалымат кошот. username жана password полярларына керектүү маалыматтарды киргизет.

Маалыматтарды жаңыртуу

```
UPDATE users SET password = 'newpassword' WHERE username = 'Tynchtyk';
```

Түшүндүрмө: users таблицасында username полясы Tynchtyk болгон колдонуучунун паролун жаңыртат.

Маалыматтарды өчүрүү

DELETE FROM users WHERE username = 'Tynchtyk';

Түшүндүрмө: users таблицасынан username полясы 'Tynchtyk' болгон жазууну өчүрөт.

Маалыматтарды алуу

SELECT * FROM users;

Түшүндүрмө: users таблицасынан бардык жазууларды алат.

Шарт менен алуу

SELECT username, password FROM users WHERE id = 1;

Түшүндүрмө: users таблицасынан id полясы 1 болгон жазуунун username жана password полярын алат.

Топтоо

SELECT COUNT(*) FROM users;

Түшүндүрмө: users таблицасынан жалпы колдонуучулардын санын эсептейт.

Сорттоо

SELECT * FROM users ORDER BY username ASC;

Түшүндүрмө: users таблицасынан бардык жазууларды username полясы боюнча өсүү тартибинде сорттойт.

Индекс жаратуу

CREATE INDEX idx_username ON users(username);

Түшүндүрмө: username полясы боюнча индексти түзөт, бул издөөнү тездетет.

Топтоо

SELECT username, COUNT(*) FROM users GROUP BY username HAVING COUNT(*) > 1;

Түшүндүрмө: username полясы боюнча топтоп, кайсы колдонуучулар бир нече жолу кошулганын көрсөтөт.

Бул MySQL'деги негизги командалардын кыскача түшүндүрмөсү. Бул командалар маалымат базасын башкарууда, маалыматтарды кошууда, жаңыртууда, өчүрүүдө жана алуу процессинде кеңири колдонулат.

25. PHP аркылуу MySQLге улануу

MySQLге PHP аркылуу улануу үчүн mysqli же PDO (PHP Data Objects) кеңейтмесин колдоно аласыз. Бул жерде mysqli пайдаланууну көрсөтөм.

1. MySQLге улануу

Төмөнкү код MySQLге улануу үчүн керектүү кодду көрсөтөт:

```
<?php
// Кошулуу параметрлерин аныктоо
$host = 'localhost'; // Сервердин адреси
$username = 'root';  // Колдонуучу аты (адатта 'root' болот)
$password = '';      // Пароль (эгер парол жок болсо, бош калтырыңыз)
$dbname = 'my_database'; // Колдонмоңуздагы маалымат базасынын аты

// MySQL серверине улануу
$conn = new mysqli($host, $username, $password, $dbname);

// Уланганды текшерүү
if ($conn->connect_error) {
```

```
    die("Улангандагы ката: " . $conn->connect_error);
}
echo "Ийгилуктүү уланды!";
?>
```

2. Кодду түшүндүрүү

- **$host**: Сервердин адреси. Эгерде сиз жергиликтүү серверде иштеп жатсаңыз, localhost деп жазсаңыз болот.
- **$username**: MySQLдеги колдонуучу аты. Адегенде сиздин root колдонуучусу болушу мүмкүн.
- **$password**: Сиздин MySQL'деги паролуңуз.
- **$dbname**: Улануу каалаган маалымат базанын аты.

3. Улануну Текшерүү

Коддо $conn->connect_error аркылуу уланганын текшерүүгө болот. Эгер ката болсо, программа "Улануда ката" деп көрсөтүп, ката жөнүндө маалымат берет. Эгерде улануу ийгиликтүү болсо, "Ийгиликтүү уланды!" деп билдирет.

4. *Дата Сурауларын Аткаруу*

MySQL'ге улангандан кийин, SQL суроолорун аткарса болот. Мисалы, таблицадан маалымат алуу үчүн:

```
$sql = "SELECT * FROM users"; // SQL суроосу
$result = $conn->query($sql); // Суроону аткаруу

if ($result->num_rows > 0) {
    // Маалыматтарды чыгарып берүү
    while ($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"] . " - Username: " . $row["username"]
. "<br>";
        }
} else {
    echo "Маалымат жок";
}
```

5. Улануўну Жабуу

MySQL’деги улануўну жаппаш керек. Мисалы, программа аягында:

```
$conn->close(); // Улануў жабуу
?>
```

26. PHP’де SQL командаларын колдонуп, маалымат базасына маалыматтарды кошуу, жаңыртуу жана өчүрүү үчүн PDO же mysqli кеңейтмелерин колдонуу менен.

Төмөндө MySQL базасына маалыматтарды кошуу, жаңыртуу жана өчүрүү операциялары үчүн PDO жана mysqli кеңейтмелери менен PHP кодунун мисалдарын көрөлү.

1. MySQLге багыттоо

PDO аркылуу туташуу

```
<?php
$host = 'localhost';
$db = 'my_database';
$user = 'root';
$pass = 'password';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

MySQLi аркылуу Туташуу

```
<?php
```

```
$host = 'localhost';
$db = 'my_database';
$user = 'root';
$pass = 'password';

// MySQLi аркылуу туташуу
$conn = new mysqli($host, $user, $pass, $db);

// Текшерүү
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

2. Маалымат кошуу

PDO аркылуу маалымат кошуу

```
<?php
$username = 'jane_doe';
$password = 'password123';
$email = 'jane@example.com';

$sql = "INSERT INTO users (username, password, email)
VALUES (:username, :password, :email)";
$stmt = $pdo->prepare($sql);

$stmt->bindParam(':username', $username);
$stmt->bindParam(':password', $password);
$stmt->bindParam(':email', $email);

if ($stmt->execute()) {
    echo "User added successfully.";
} else {
    echo "Error adding user.";
}
?>
```

MySQLi аркылуу маалымат кошуу

```
<?php
$username = 'jane_doe';
```

```
$password = 'password123';
$email = 'jane@example.com';

$sql = "INSERT INTO users (username, password, email)
VALUES (?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("sss", $username, $password, $email);

if ($stmt->execute()) {
   echo "User added successfully.";
} else {
   echo "Error adding user.";
}
?>
```

3. Маалымат жаңыртуу

PDO аркылуу маалымат жаңыртуу

```
<?php
$newPassword = 'newpassword456';
$usernameToUpdate = 'jane_doe';

$sql = "UPDATE users SET password = :password WHERE
username = :username";
$stmt = $pdo->prepare($sql);

$stmt->bindParam(':password', $newPassword);
$stmt->bindParam(':username', $usernameToUpdate);

if ($stmt->execute()) {
   echo "User updated successfully.";
} else {
   echo "Error updating user.";
}
?>
```

MySQLi аркылуу маалымат жаңыртуу

```
<?php
$newPassword = 'newpassword456';
```

```
$usernameToUpdate = 'jane_doe';

$sql = "UPDATE users SET password = ? WHERE username =
?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ss", $newPassword, $usernameToUpdate);

if ($stmt->execute()) {
   echo "User updated successfully.";
} else {
   echo "Error updating user.";
}
?>
```

4. Маалымат өчүрүү

PDO аркылуу маалымат өчүрүү

```
<?php
$usernameToDelete = 'jane_doe';

$sql = "DELETE FROM users WHERE username = :username";
$stmt = $pdo->prepare($sql);

$stmt->bindParam(':username', $usernameToDelete);

if ($stmt->execute()) {
   echo "User deleted successfully.";
} else {
   echo "Error deleting user.";
}
?>
```

MySQLi аркылуу маалымат өчүрүү

```
<?php
$usernameToDelete = 'jane_doe';

$sql = "DELETE FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $usernameToDelete);
```

```
if ($stmt->execute()) {
    echo "User deleted successfully.";
} else {
    echo "Error deleting user.";
}
?>
```

27. PHP аркылуу MySQL базасынан маалыматтарды алуу (выборка)

1. MySQLге багыттоо

PDO аркылуу туташуу

```
<?php
$host = 'localhost';
$db = 'my_database';
$user = 'root';
$pass = 'password';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

MySQLi аркылуу туташуу

```
<?php
$host = 'localhost';
$db = 'my_database';
$user = 'root';
$pass = 'password';

// MySQLi аркылуу туташуу
$conn = new mysqli($host, $user, $pass, $db);

// Текшерүү
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}
?>
```

2. Маалыматтарды алуу

PDO аркылуу маалыматтарды алуу

```
<?php
$sql = "SELECT * FROM users";
$stmt = $pdo->prepare($sql);
$stmt->execute();

// Натыйжа алуу
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);

foreach ($results as $row) {
    echo "Username: " . $row['username'] . ", Email: " .
$row['email'] . "<br>";
}
?>
```

MySQLi аркылуу маалыматтарды алуу

```
<?php
$sql = "SELECT * FROM users";
$result = $conn->query($sql);

// Натыйжа алуу
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        echo "Username: " . $row['username'] . ", Email: " .
$row['email'] . "<br>";
    }
} else {
    echo "No results found.";
}
?>
```

3. Шарттуу выборка

PDO аркылуу шарттуу выборка

```
<?php
$usernameToFind = 'jane_doe';
```

```
$sql = "SELECT * FROM users WHERE username = :username";
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':username', $usernameToFind);
$stmt->execute();

$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
if (count($results) > 0) {
    foreach ($results as $row) {
        echo "Username: " . $row['username'] . ", Email: " .
$row['email'] . "<br>";
    }
} else {
    echo "User not found.";
}
?>
```

MySQLi аркылуу шарттуу выборка

```
<?php
$usernameToFind = 'jane_doe';
$sql = "SELECT * FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $usernameToFind);
$stmt->execute();

$result = $stmt->get_result();
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        echo "Username: " . $row['username'] . ", Email: " .
$row['email'] . "<br>";
    }
} else {
    echo "User not found.";
}
?>
```

4. Жыйынтык

Бул жерде Bootstrap, PHP жана MySQL колдонуп CRUD (Create, Read, Update, Delete) операцияларын аткаруучу мисалдын коддору берилет. JavaScript колдонбостон, форма маалыматтарын жөнөтүү үчүн PHP коддору колдонулат.

MySQL базасын даярдоо

Эмне үчүн MySQL базасын түзүү.

Базаны жана таблицаны түзүү

```
CREATE DATABASE my_database;

USE my_database;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

Bootstrap HTML түзүмү

Bootstrap колдонуп, интерфейс түзүү.

HTML коду

```
<!DOCTYPE html>
<html lang="ky">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.mi
n.css">
    <title>CRUD мисал</title>
</head>
<body>
<div class="container mt-5">
    <h2>Пайдаланучулар</h2>
    <button class="btn btn-primary" data-toggle="modal" data-
target="#addUserModal">Пайдаланучу кошуу</button>

    <table class="table mt-3">
      <thead>
```

```
                <tr>
                    <th>ID</th>
                    <th>Колдонуучу Аты</th>
                    <th>Email</th>
                    <th>Чаралар</th>
                </tr>
            </thead>
            <tbody id="userTable">
                <!-- Пайдаланучулар жүктөлөт -->
                <?php include 'get.php'; ?>
            </tbody>
        </table>
    </div>

    <!-- Модал - Пайдаланучу кошуу -->
    <div class="modal" id="addUserModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Пайдаланучу кошуу</h4>
                    <button type="button" class="close" data-
dismiss="modal">&times;</button>
                </div>
                <div class="modal-body">
                    <form action="add.php" method="POST">
                        <div class="form-group">
                            <label for="username">Колдонуучу Аты:</label>
                            <input type="text" class="form-control"
name="username" required>
                        </div>
                        <div class="form-group">
                            <label for="email">Email:</label>
                            <input type="email" class="form-control"
name="email" required>
                        </div>
                        <button type="submit" class="btn btn-
primary">Кошуу</button>
                    </form>
                </div>
```

```
            </div>
        </div>
    </div>

    <!-- Модал - Пайдаланучу жаңыртуу -->
    <div class="modal" id="editUserModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Пайдаланучу жаңыртуу</h4>
                    <button type="button" class="close" data-
dismiss="modal">&times;</button>
                </div>
                <div class="modal-body">
                    <form action="update.php" method="POST">
                        <input type="hidden" name="id" id="editUserId">
                        <div class="form-group">
                            <label for="editUsername">Колдонуучу
Аты:</label>
                            <input type="text" class="form-control"
name="username" id="editUsername" required>
                        </div>
                        <div class="form-group">
                            <label for="editEmail">Email:</label>
                            <input type="email" class="form-control"
name="email" id="editEmail" required>
                        </div>
                        <button type="submit" class="btn btn-
success">Жаңыртуу</button>
                    </form>
                </div>
            </div>
        </div>
    </div>

    <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
    </body>
    </html>
```

PHP коддору

MySQL туташуу

db.php файлы:

```
<?php
$host = 'localhost';
$db = 'my_database';
$user = 'root'; // MySQL колдонуучусунун аты
$pass = ''; // MySQL колдонуучусунун паролу

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Пайдаланучу кошуу (add.php)

```
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $email = $_POST['email'];

    $sql = "INSERT INTO users (username, email) VALUES
('$username', '$email')";

    if ($conn->query($sql) === TRUE) {
```

```
            header("Location: index.php"); // Башка веб-баракчага
жөнөтүү
        } else {
            echo "Ката: " . $sql . "<br>" . $conn->error;
        }

        $conn->close();
    }
    ?>
```

Пайдаланучуларды жүктөө (get.php)

```
    <?php
    include 'db.php';

    $sql = "SELECT * FROM users";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            echo "<tr>
                    <td>{$row['id']}</td>
                    <td>{$row['username']}</td>
                    <td>{$row['email']}</td>
                    <td>
                        <button class='btn btn-success' data-toggle='modal'
data-target='#editUserModal' data-id='{$row['id']}' data-
username='{$row['username']}' data-
email='{$row['email']}'>Редактировать</button>
                        <a href='delete.php?id={$row['id']}' class='btn btn-
danger'>Удалить</a>
                    </td>
                </tr>";
        }
    } else {
        echo "<tr><td colspan='4'>Жыйынтык жок</td></tr>";
    }

    $conn->close();
    ?>
```

Пайдаланучу жаңыртуу (update.php)

```
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $id = $_POST['id'];
    $username = $_POST['username'];
    $email = $_POST['email'];

    $sql = "UPDATE users SET username='$username',
email='$email' WHERE id=$id";

    if ($conn->query($sql) === TRUE) {
        header("Location: index.php");
    } else {
        echo "Ката: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
}
?>
```

Пайдаланучу өчүрүү (delete.php)

```
<?php
include 'db.php';

if (isset($_GET['id'])) {
    $id = $_GET['id'];

    $sql = "DELETE FROM users WHERE id=$id";

    if ($conn->query($sql) === TRUE) {
        header("Location: index.php");
    } else {
        echo "Ката: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
}
```

```
?>
```

Модалдардагы жаңыртуу функциясы

HTML кодунда жаңыртуу модалын толтуруу үчүн PHP кодун колдонобуз. Модалдын иштеши үчүн кнопкаларды жүктөө коду төмөндөгүдөй болот:

```
<script>
   $('#editUserModal').on('show.bs.modal', function(event) {
      var button = $(event.relatedTarget); // Кнопканы ал
      var id = button.data('id');
      var username = button.data('username');
      var email = button.data('email');

      // Модалдагы инпуттарга маалыматтарды жайгаштыруу
      var modal = $(this);
      modal.find('#editUserId').val(id);
      modal.find('#editUsername').val(username);
      modal.find('#editEmail').val(email);
   });
</script>
```

Натыйжа

Бул код мисалы менен сиз Bootstrap, PHP жана MySQL колдонуп, веб-колдонуучу интерфейсин түзүп, CRUD операцияларын аткара аласыз. JavaScript колдонбостон, формалар аркылуу PHP кодун колдонуп, маалыматтарды жүктөө, кошуу, жаңыртуу жана өчүрүү функцияларын аткарууга болот.

Бул китепте PHP, MySQL жана Bootstrap технологияларын колдонуп веб-өнүктүрүүнүн негиздерин кеңири карап чыктык. Сиздер веб-колдонууларды эффективдүү түзүү үчүн зарыл болгон маалыматтар менен таанышып, CRUD (Create, Read, Update, Delete) операцияларын кантип ишке ашырууну үйрөндүңүздөр.

PHP – веб-дизайнын интерактивдүү кылууга мүмкүндүк берген күчтүү сервердик программалоо тили. Сиздин веб-баракчаларыңыздын динамикалык болушу, колдонуучулар менен өз ара аракеттешүүсү PHPнин натыйжасында мүмкүн болуп жатат. Бул китепте PHP синтаксисин, өзгөрмөлөрдү жана функцияларды түшүнүп, веб-колдонмолорду түзүү боюнча негизги билимдерди алдыңыздар.

MySQL – маалыматтарды эффективдүү сактоо жана башкаруу үчүн эң жакшы инструмент. Сиз таблицаларды түзүп, маалыматтарды кошуп, SQL командаларын колдонуп, маалыматтарды манипуляциялоону үйрөндүңүздөр. Бул базаны колдонуп, чоң көлөмдөгү маалыматтарды башкарууга мүмкүнчүлүк алдыңыздар.

Bootstrap – веб-сайттын дизайн жана сырткы көрүнүшүн жакшыртуучу фреймворк. Сиз веб-баракчаларды тез арада жана оңой эле кооздоп, аларды ар кандай түзмөктөргө ылайыкташтыра алдыңыздар. Bootstrap колдонуу менен, веб-дизайныңызды жана колдонуучулардын тажрыйбасын жогорулатууга жол ачтыңыздар.

Бул китепте көрсөтүлгөн концепциялар менен технологияларды колдонуу сизге веб-өнүктүрүү чөйрөсүндө тажрыйба топтоого жана келечекте өз долбоорлоруңузду ишке ашырууга жардам берет. Веб-өнүктүрүү — бул туруктуу өнүгүү жана жаңы технологияларды үйрөнүү процесс болуп саналат.

Ошондуктан, изилдөөлөрүңүздү улантууга жана өзүңүздүн чыгармачылык жөндөмүңүздү өнүктүрүүгө чакырабыз.

Сиздерге веб-өнүктүрүүдө ийгилик жана жаңы жетишкендиктерди каалайбыз!

Адабияттар

1. **Дакетт, Дж.** HTML и CSS: Разработка и дизайн веб-сайтов. — СПб.: Питер, 2018.
2. **Дакетт, Дж.** JavaScript и JQuery: Интерактивная веб-разработка. — СПб.: Питер, 2018.
3. **Янки, К.** PHP и MySQL для начинающих. — Москва: ДМК Пресс, 2021.
4. **Фримен, Э., Робсон, Э.** Изучаем программирование на JavaScript. — СПб.: Питер, 2021.
5. **Веллинг, Л., Томпсон, Л.** PHP и MySQL. Разработка Web-приложений. — Москва: ДМК Пресс, 2022.
6. **Фридман, Дж.** Bootstrap для начинающих. — Москва: ДМК Пресс, 2020.
7. **Зандстра, М.** PHP. Объектно-ориентированное программирование. — СПб.: Питер, 2019.
8. **Дюбуа, П.** Эффективное использование MySQL. — Москва: Вильямс, 2017.
9. **Веру, Л.** Секреты CSS. — Москва: ДМК Пресс, 2021.
10. **Лоусон, Б., Шарп, Р.** Изучаем HTML5. — СПб.: Питер, 2017.
11. **Сиверс, С.** Полное руководство по MySQL. — Москва: Вильямс, 2020.
12. **Стоукс, Р.** Node.js и JavaScript. — Москва: ДМК Пресс, 2019.
13. **Фехли, К., Гольдштейн, Э.** SQL: Полное руководство. — СПб.: Питер, 2021.

Мазмуну

Илимий басылма

Асилбеков Тынчтыкбек Майрамбекович, Орозов Максатбек Омурбекович

PHP, MySQL жана Bootstrap менен Веб-сайтты түзүү негиздери

Печатка берилген күнү 17.04.2025-ж.

Формат 60x84 1/16 Көлөмү 6 б.т.

Тираж: 200 д.

“Билим” редакциялык-басма сектору