1. Интернет. Интернет протокол IP

Интерне́т (англ. *Internet*) — всемирная система объединённых компьютерных сетей для хранения и передачи информации.WWW – набор интернет-сайтов или порталов. Интернет включает в себя и др.службы – e-mail, IP-TV, FTP и тд.

IP – простой протокол, позволяющий передавать данные между двумя ПК. Каждое устройство имеет 32-битный IP-адрес. Адреса – четыре 8-битных числа(от0 до 255). IP можно узнать с помощью сайта 2Ip.ru или с помощью команд для ОС: ipconfig(windows), ifconfing(Mac/Linux). **Протокол IP** отвечает непосредственно за передачу данных по сети и адресацию, т.е. за правильность доставки сообщений по указанному адресу. Иногда пакеты одного сообщения могут доставляться разными путями.

В каждом IP-пакете указывается адрес отправителя и адрес получателя. Маршрутизатор автоматически определяет маршрут передачи пакета и передает следующему маршрутизатору. Пакет передается маршрутизаторами от одного к другому, пока не достигнет сети, в которой расположен компьютер, которому адресован пакет. Маршрутизатор этой сети определяет, какой рабочей станции адресован пакет. Все пакеты доставляются на эту рабочую станцию и из них собирается сообщение.

Пример: 145.10.34.3 ≈ 10 01001.00001010.00000.00000011

2. Интернет. Протокол TCP

Интерне́т (англ. *Internet*) — всемирная система объединённых компьютерных сетей для хранения и передачи информации.WWW – набор интернет-сайтов или порталов. Интернет включает в себя и др.службы – e-mail, IP-TV, FTP и тд.

TCP(Transmission Control Protocol) – протокол передачи данных гарантирует доставку данных по IP, добавляет мультипликсирование(мн-во программ использует один IP). Для этого вводится понятие порт – число, ассоциир-е с каждой программой или службой. Пример: port80 – для браузера, port25 – для почты.

Протокол TCP отвечает за

- правильность разбиения сообщений на пакеты информации
- сборку пакетов в конечном пункте в соответствии с их номерами. Если какой-либо из пакетов утерян или поврежден (передан с ошибками), то его передачу повторяют.
- Обеспечивает надежную доставку данных, так как предусматривает установления логического соединения;
- Делит передаваемый поток байтов на части — сегменты - и передает их нижнему уровню, на приемной стороне снова собирает их в непрерывный поток байтов.

Соединение двух узлов начинается с handshake (рукопожатия):

1. Узел A посылает узлу B специальный пакет SYN — приглашение к соединению

2. B отвечает пакетом SYN-ACK — согласием об установлении соединения

3. A посылает пакет ACK — подтверждение, что согласие получено

После этого TCP-соединение считается установленным, и приложения, работающие в этих узлах, могут посылать друг другу пакеты с данными.«Соединение» означает, что узлы помнят друг о друге, нумеруют все пакеты, идущие в обе стороны, посылают подтверждения о получении каждого пакета и перепосылают потерявшиеся по дороге пакеты.

3.Интернет. Протокол HTTP

Интерне́т (англ. *Internet*) — всемирная система объединённых компьютерных сетей для хранения и передачи информации.WWW – набор интернет-сайтов или порталов. Интернет включает в себя и др.службы – e-mail, IP-TV, FTP и тд.

Протокол HTTP (Hypertext Transfer Protocol – Протокол передачи гипертекста) является протоколом самого верхнего уровня - уровня приложения. Он был разработан для эффективной передачи по Интернету Web-страниц, т.е. является основой системы Word Wide Web. (Гипертекст - Совокупность документов, содержащих текстовую, аудиои видеоинформацию, связанных между собой взаимными ссылками в единый текст). Набор команд, понимаемых web-сервером и посылаемым браузером.

HTTP команды: get filename – скачивание, post filename – отправка web-форм, put filename – загрузка файла на сервер.Коды ошибок: 200 – OK, 301-303 – стр. Перемещена, 403 – доступ закрыт, 505 – ошибка сервера.

URL(uniform resource locator) – единый указатель ресурса, индификатор, показывающий положение документа на сайте. Host(интерпритатор php, asp,jsp)→(HTML)→HTTP→браузер.

Пример: http://www.yandex.ru/..... – протокол//host/путь

4. HTML. Тэги для работы с текстом

HTML (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства. Язык регистронезависимый. Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначается специальными пометками — *тегами*. Элементы могут быть *пустыми*, то есть не содержащими никакого текста и других данных.

```
<html>
   <head>
      <meta charset="utf-8" />
      <title>HTML Document</title>
   </head>
   <body>
      <p>
         <b>
            Этот текст будет полужирным, <i>а этот – ещё и курсивным</i>.
         </b>
      </p>
   </body>
</html>
```

HTML теги для работы с текстом.

1.<H1>, **<H2>**, **<H3>**, **<H4>**, **<H5>**, **<H6>**.Тегу H1 соответствует самый большой заголовок, тегу H6 - самый маленький. Закрывающий тег обязателен.**Атрибуты:**

- **align** - Выравнивает заголовок в соответствии со следующими значениями:
- **center** - По центру.
- **left** - По левому краю.
- **right** - По правому краю.
- **title** - Всплывающая подсказка.

2.Тег <p> создает новый параграф.Атрибуты:

align - Выравнивает параграф относительно одной из сторон документа.
left - выравнивание по правому краю (По умолчанию).
right - выравнивание по правому краю.
center - выравнивание по центру.
justify - выравнивание по ширине.

**3.Контейнер ** выделяет текст жирным шрифтом. Аналогичный тег - , он тоже выделяет текст жирным. Но его не рекомендуется использовать больше 1-2 раз на странице - при большом количестве тегов на странице поисковые системы могут воспринять это как спам. **Контейнер ** выделяет текст жирным шрифтом.Рекомендуется использовать этот тег для выделения наиболее значимого ключевого слова (или словосочетания) для акцентирования на нем внимая посковых систем.

4.Тег <hr> добавляет в документ горизонтальную линию.Закрывающий тег не обязателен.**Атрибуты:**

- **size** - Устанавливает толщину линии.
- **width** - Устанавливает ширину линии в пикселах или процентах.
- **noshade** - Создает линию без тени.
- **color** - Задает линии определенный цвет.

5. **Тег
 переводит текст на новую строку.**
6. **Контейнер ** делает подиндекс. H₂0 = H_2O
7. **Контейнер ** делает надиндекс. X² = 4 = X^2=4
8. **Контейнер <big> </big>** выводит более крупный, чем окружающий текст.
9. **Контейнер <small> </small>** выводит более мелкий, чем окружающий текст.

10. **Контейнер   и ** выделяет текст курсивом.
11. **Тег <s>** делает текст зачеркнутым.
12. **Контейнер <u> </u>** делает текст подчеркнутым.
13. **Тег ** определяет цвет, размер и выводимый шрифт.Закрывающий тег обязателен.
    color - определяет цвет текста.
    face - определяет гарнитуру шрифта.
    size - определяет размер текста в пределах от 1 до 7, где 1 - самый мелкий шрифт. По умолчанию равен Пр: .

15.Тег <marquee> заставляет текст перемещаться из стороны в сторону.Закрывающий тег </marquee> обязателен.**Атрибуты:**

- behavior - Определяет вид движения.
- **alternate** - Колебательные движения налево и направо.
- **scroll** - Перемещение текста в направлении, указанном в direction. Достигнув края экрана, надпись появляется снова с противоположной стороны.
- **slide** - Схоже с scroll, но текст перемещается только один раз и останавливается.
- direction - Определяет направление движения.
- **down** - Движение вниз.
- **left** - Движение справа налево (по умолчанию).
- **right** - Движение слева направо.
- **up** - Движение вверх.

Пр:<marquee behavior="alternate" direction="right"></marquee>

16.Контейнер <code> </code> применяют для выделения программного кода, отображаемого на странице.

5. HTML. Тэги для создания таблиц

Для добавления таблицы на веб-страницу используется тег **<table>**. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов **<tr>** и **<td>**. Таблица должна содержать хотя бы одну ячейку. Допускается вместо тега **<td>** использовать тег **<th>**. Текст в ячейке, оформленной с помощью тега **<th>**, отображается браузером шрифтом жирного начертания и выравнивается по центру ячейки. В остальном, разницы между ячейками, созданными через теги **<td>** и **<th>** нет. Тег **<td>** предназначен для создания одной ячейки таблицы. Тег **<td>** должен размещаться внутри контейнера **<tr>**, который в свою очередь располагается внутри тега **<table>**. Тег **<tr>** служит контейнером для создания строки таблицы. Каждая ячейка в пределах такой строки может задаваться с помощью тега**<th>** или **<td>**.**Атрибуты**

- align-Определяет выравнивание таблицы.
- background - Задает фоновый рисунок в таблице.

Синтаксис

```
<table>
  <tr>
    <td>...</td>
  </tr>
</table>
```

- bgcolor - Цвет фона таблицы.
- border - Толщина рамки в пикселах.
- bordercolor-Цвет рамки.
- cellpadding-Отступ от рамки до содержимого ячейки.
- cellspacing-Расстояние между ячейками.
- cols-Число колонок в таблице.
- frame-Сообщает браузеру, как отображать границы вокруг таблицы.
- height-Высота таблицы.
- rules-Сообщает браузеру, где отображать границы между ячейками.
- summary-Краткое описание таблицы.
- width-Ширина таблицы.

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тег TABLE</title>
 </head>
 <body>
  <table border="1" width="100%" cellpadding="5">
   <tr>
    <th>Ячейка 1</th>
    <th>Ячейка 2</th>
   </tr>
   <tr>
    <td>Ячейка 3</td>
    <td>Ячейка 4</td>
  </tr>
 </table>
 </body>
</html>
```

6. HTML. Тэги для создания списков и гиперссылок.
HTML-списки используются для группировки связанных между собой фрагментов информации
Маркированный список представляет собой неупорядоченный список *(от англ. Unordered List)*. Создаётся с помощью парного тега . В качестве маркера элемента списка выступает метка, например, закрашенный кружок.
Каждый элемент списка создаётся с помощью парного тега *(от англ. List Item)*.

```
Пример: <ul>
 <li>Microsoft</li>
 <li>Google</li>
 <li>Apple</li>
 <li>IBM</li>
</ul>
```

Нумерованный список создаётся с помощью парного тега . Каждый пункт списка также создаётся с помощью элемента <li>. Браузер нумерует элементы по

порядку автоматически и если удалить один или несколько элементов такого списка, то остальные номера будут автоматически пересчитаны.

```
Пример: <ol>
  <li>Microsoft</li>
  <li>Google</li>
  <li>Apple</li>
  <li>IBM</li>
</ol>
```

Списки определений создаются с помощью тега <dl></dl>. Для добавления термина применяется тег <dt></dt>, а для вставки определения — тег <dd></dd>

```
Пример: <dl>
  <dt>Режиссер:</dt>
    <dd>Петр Точилин</dd>
  <dt>В ролях:</dt>
    <dd>Андрей Гайдулян</dd>
    <dd>Алексей Гаврилов</dd>
    <dd>Виталий Гогунский</dd>
    <dd>Мария Кожевникова</dd>
</dl>
```

Зачастую возможностей простых списков не хватает, например, при создании оглавления никак не обойтись без вложенных пунктов. Разметка для вложенного списка будет следующей:

```
<ul>
 <li>Пункт 1.</li>
  <li>Пункт 2.
    <ul>
```

```
    <li>Подпункт 2.1.</li>
     <li>Подпункт 2.2.
      <ul>
        <li>Подпункт 2.2.1.</li>
        <li>Подпункт 2.2.2.</li>
        </ul>
     </li>
    <li>Подпункт 2.3.</li>
   </ul>
 </li>
<li>Пункт 3.</li>
</ul>
```

Тег <a> (от anchor- якорь), в него можно заключить текст или рисунок, которые станут ссылкой на те или иные документы. Атрибут тега <a> href задаёт имя и путь к документу на который указывает ссылка.
Здесь мои фотки!! - Такая запись подразумевает, что в директории, где расположен наш первый html документ есть папка stranica в которой расположен файл primer.html
Здесь мои фотки!! - документ расположен на сайте www.site.ru.

```
Пример кода: <html>
<head>
<title>Закладки</title>
</head>
<body>
<h2>А. С. ПУШКИН</h2>
<a href="#skazka1">Сказка о попе и работнике его Балде</a><br>
<a href="#skazka2">Сказка о рыбаке и рыбке</a><br>
<a href="#skazka3">Сказка о царе Салтане</a>
<h3><a name="skazka1">Сказка о попе и работнике его Балде</a></h3>
<pre>
Жил-был поп,
Толоконный лоб.
... ... ...
</pre>
```

```
<h3><a name="skazka2">Сказка о рыбаке и рыбке</a></h3>
<pre>
Жил старик со своею старухой
У самого синего моря.
... ... ...
</pre>
<h3><a name="skazka3">Сказка о царе Салтане</a></h3>
<pre>
Три девицы под окном
Пряли поздно вечерком.
... ... ...
</pre>
</body>
</html>
```

7. CSS. Задание цвета

CSS (Cascading Style Sheets), или каскадные таблицы стилей, используются для описания внешнего вида документа, написанного языком разметки. Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.
Вот список основных ключевых слов, которые можно использовать:

```
aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow
```

Имена цветов обязательно нужно указывать маленькими буквами, т.к. CSS здесь чувствителен к регистру.
Вот пример использования ключевых слов:

```
body {color: black; background: white }

h1 { color: maroon }

h2 { color: olive }
```

В CSS тоже можно работать с RGB моделью цвета.
В случае использования 16-ричного представления цвета, перед указанием цвета ставится знак решетки (#)
Возможно указание цвета в виде десятичных значений:

```
em { color: rgb(255,0,0) }
```

Вместо цифр, возможно указание процентов. В этом случае числу 255 будет соответствовать 100%, а 0 – 0%.

```
em { color: rgb(100%,0%,7%) }
```

Если указать число более 255, то оно будет приведено к 255.

((***em***) — ***это*** масштабируемая единица измерения, которую использует в web. 1em равен текущему размеру шрифт)
В CSS тоже можно работать с RGBA моделью цвета (A – отвечает за прозрачность)
В итоге, модель RGBA выглядит следующим образом:
rgba(r,g,b,opacity)
opacity – значение от 0 до 1 с шагом 0.1.
1 – полностью непрозрачный цвет.
0 – полностью прозрачный цвет.

```
em { color: rgba(255,0,0,0.5)
```

В отличие от модели RGB, в модели RGBA нельзя представлять значение цвета в 16-ричной системе счисления

8. CSS. Задание параметров текста

CSS-текст представляет набор свойств для форматирования текстового содержимого веб-страниц

text-indent	Отступ первой строки текста в блоке
text-align	Выравнивание текста (left, right, center, justify)
text-decoration	Декорирование текста none (нет), underline (подчеркнутый), overline (надчеркнутый), line-through (перечеркнутый), blink (мерцающий)
text-shadow	Эффекты затенения текста
letter-spacing	Межсимвольное расстояние
word-spacing	Расстояние между словами
text-transform	Преобразование текста (none, capitalize (первая буква каждого слова заглавная), uppercase (все буквы заглавные), lowercase (все буквы строчные))
line-height	Межстрочный интервал
white-space	Способ обработки символов пустого пространства внутри блока(normal, pre (не изменять символы пустого пространства), nowrap (не разрывать строку))

Пример: **<html>**

```
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 <title>Шрифт</title>
 <style type="text/css">
  H1 {
   font-family: Arial, Helvetica, Verdana, sans-serif; /* Гарнитура шрифта */
   font-size: 150%; /* Размер текста */
   font-weight: lighter; /* Светлое начертание */
  }
 </style>
</head>
<body>
  <H1>Заголовок</H1>
 <p>Обычный текст</p>
</body>
</html>
```

9. CSS.Группирование и наследование стилей

Благодаря группированию и наследованию появляется возможность упорядочить и оптимизировать документы, описывающие стили элементов вебстраниц сайта. Группирование: общие свойства селекторов можно объединить и не прописывать их в индивидуальных частях кода:

```
h1, h2, h3
{
   font-family: Tahoma, Arial, Helvetica,
sans-serif;

}

h1 {
   font-size:
2.5em;
   color:#fffff
d;
```

```
}
h2 {
   font-size:
1.8em;
   color:
#265a8b;
}
h3 {
  font-size:
1.7em;
  color:#6b6b4
e;
}
```

Наследование предполагает перенос правил стилей для элементов, находящихся внутри других. Такие элементы называются дочерними и они наследуют стилевые свойства своих родителей. Наследование позволяет определять значения один раз, задавая их для родительского элемента верхнего уровня. Благодаря наследованию нет необходимости определять свойства для абсолютно каждого элемента в отдельности.

Пример:

```
body
{
   font-family: Tahoma, Arial, Helvetica, sans-serif; /* Стиль
шрифта */
   color: black; /* Черный цвет
текста */
 }
```

10. CSS. Классы. Псевдо-классы.

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

```
Тег.Имя класса { свойство1: значение; свойство2: значение; ... }
```

Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определённым классом, к тегу добавляется атрибут class="Имя класса"

Пример:

```
<html>

 <head>

  <meta charset="utf-8">

  <title>Камни</title>
```

```
<style>
 table.jewel {
  width: 100%; /* Ширина таблицы */
  border: 1px solid #666; /* Рамка вокруг таблицы */
 }
 th {
  background: #009383; /* Цвет фона */
  color: #fff; /* Цвет текста */
  text-align: left; /* Выравнивание по левому краю */
 }
 tr.odd {
  background: #ebd3d7; /* Цвет фона */
 }
</style>
</head>
<body>
<table class="jewel">
 <tr>
  <th>Название</th><th>Цвет</th><th>Твердость по Моосу</th>
 </tr>
 <tr class="odd">
  <td>Алмаз</td><td>Белый</td><td>10</td>
 </tr>
 <tr>
  <td>Рубин</td><td>Красный</td><td>9</td>
 </tr>
```

```
 <tr class="odd">
  <td>Аметист</td><td>Голубой</td><td>7</td>
 </tr>
 <tr>
  <td>Изумруд</td><td>Зеленый</td><td>8</td>
 </tr>
 <tr class="odd">
  <td>Сапфир</td><td>Голубой</td><td>9</td>
 </tr>
</table>
</body>
</html>
```

В примере класс с именем odd используется для изменения цвета фона строки таблицы. За счёт того, что этот класс добавляется не ко всем тегам **<tr>** и получается чередование разных цветов.

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

```
Селектор:Псевдокласс { Описание правил стиля }
```

Псевдоклассы, определяющие состояние элементов

К этой группе относятся псевдоклассы, которые распознают текущее состояние элемента и применяют стиль только для этого состояния.

:active

Происходит при активации пользователем элемента. Например, ссылка становится активной, если навести на неё курсор и щёлкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс :active используется преимущественно для ссылок.

:link

Применяется к непосещенным ссылкам, т. е. таким ссылкам, на которые пользователь ещё не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная хотя бы потому, что по ней был зафиксирован переход ранее.

Запись A {...} и A:link {...} по своему результату равноценна, поскольку в браузере даёт один и тот же эффект, поэтому псевдокласс :link можно не указывать. Исключением являются якоря, на них действие :link не распространяется.

:focus

Применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текс

:hover

Псевдокласс :hover активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

:visited

Данный псевдокласс применяется к посещённым ссылкам. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно

Псевдоклассы, имеющие отношение к дереву документа
К этой группе относятся псевдоклассы, которые определяют положение элемента в дереве документа и применяют к нему стиль в зависимости от его статуса.
:first-child
Применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа. Чтобы стало понятно, о чем речь, разберём небольшой код

Псевдоклассы, задающие язык текста
Для документов, одновременно содержащих тексты на нескольких языках имеет значение соблюдение правил синтаксиса, характерные для того или иного языка. С помощью псевдоклассов можно изменять стиль оформления иностранных текстов, а также некоторые настройки.
:lang
Определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через атрибут lang, он обычно добавляется к тегу **<html>**. С помощью псевдокласса :lang можно задавать определённые настройки, характерные для разных языков, например, вид кавычек в цитатах

Пример: **<html>**

<head>

```
<meta charset="utf-8">
<title>lang</title>
<style>
 P {
  font-size: 150%; /* Размер текста */
 }
 q:lang(de) {
  quotes: "\201E" "\201C"; /* Вид кавычек для немецкого языка */
 }
 q:lang(en) {
  quotes: "\201C" "\201D"; /* Вид кавычек для английского языка */
 }
 q:lang(fr), q:lang(ru) { /* Вид кавычек для русского и французского языка */
  quotes: "\00AB" "\00BB";
 }
</style>
</head>
<body>
 <p>Цитата на французском языке: <q lang="fr">Ce que femme veut,
 Dieu le veut</q>.</p>
 <p>Цитата на немецком: <q lang="de">Der Mensch, versuche die Gotter nicht</q>.</p>
 <p>Цитата на английском: <q lang="en">To be or not to be</q>.</p>

</body>
</html>
```

11. CSS. Логические блоки , id-селекторы
Элемент ** ** является блочным элементом и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.
Как и при использовании других блочных элементов, содержимое тега ** ** всегда начинается с новой строки. После него также добавляется перенос строки.

в CSS есть возможность присваивать стили тегам по их уникальному атрибуту id="".Синтаксис применения селектора идентификатора следующий:

```
#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

Имя идентификатора объявляется символом # и должно начинаться с латинской буквы, может включать в себя символы "-" и "_" (знак подчёркивания). В имени идентификатора запрещено использовать буквы кириллицы.

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS: идентификаторы id.</title>
        <style>
                #idDiv { color: red; }
        </style>
</head>
<body>
        <div>Текст первого блока</div>
        <div id="idDiv">Текст второго блока</div>
        <div id="iddiv">Текст третьего блока</div>
</body>
</html>
```

Выполнив данный пример, мы увидим, что регистр символов имеет значение для работы идентификатора и стиль сработает только для второго абзаца:

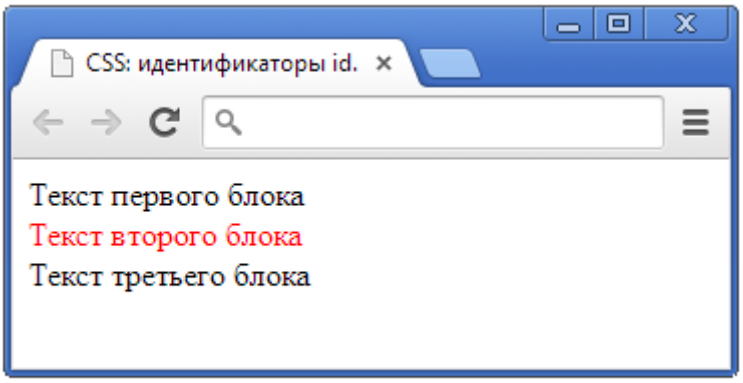


Рисунок 1. Работа примера №1.

Идентификаторы id="idDiv" и id="iddiv" - это разные имена.

12. PHP. Типы данных. Функции

PHP является языком с динамической типизацией. Это значит, что тип данных переменной выводится во время выполнения, и в отличии от ряда других языков программирования в PHP не надо указывать перед переменной тип данных.
PHP поддерживает восемь простых типа данных:

- boolean (логический тип)
- integer (целые числа)
- double (дробные числа)
- string (строки)
- array (массивы)
- object (объекты)
- resource (ресурсы)
- NULL

Integer (целочисленный тип)

Представляет целое число со знаком размером в 32 бита (от -2 147 483 648 до 2 147 483 647).

```
1    $int = -100;
```

```
2    echo $int;
```

Здесь переменная $int представляет целочисленный тип, так как ей присваивается целочисленное значение.

Кроме десятичных целых чисел PHP обладает возможностью использовать также двоичные, восьмеричные и шестнадцатеричные числа. Шаблоны чисел для других систем:

- шестнадцатеричные : 0[xX][0-9a-fA-F]
- восьмеричные : 0[0-7]
- двоичные : 0b[01]

Например:

```
1    <?php

2    // Все числа в десятичной системе имеют значение 28

3    $int_10 = 28; // десятичное число

4    $int_2 = 0b11100; // двоичное число

5    $int_8 = 034; // восьмеричное число

6    $int_16 = 0x1C; // шестнадцатеричное число

7    echo "int_10 = $int_10 <br>";

8    echo "int_2 = $int_2 <br>";

9    echo "int_8 = $int_8 <br>";

10   echo "int_16 = $int_16";

11   ?>
```

Функции – вещь довольно простая. Она представляет собой кусок кода, который принимает определенные параметры и на выходе возвращает какой-либо результат. Можно написать функцию один раз, а затем использовать её в различных местах. Таким образом вам не нужно будет дублировать код, если что-то нужно сделать дважды, или трижды, или сколько угодно раз. Вообще, функции в PHP можно сравнить с функциями в математике.

В PHP изначально содержится огромное число встроенных функций. Это очень сильная сторона этого языка – почти под любую вашу потребность имеется уже готовая функцияНапример, нам нужен косинус числа 3.14. Легко! Для этого в PHP есть функция cos.

```
<?php


echo cos(3.14);
```

Результат:

```
-0.99999873172754
```

Так же функции можно создавать самим. В целом определение любой функции выглядит следующим образом:

```
function имяФункции (аргумент1, аргумент2)

{

   какие-то действия;

   return результат;

}
```

У функции обязательно должно быть имя. **У всех функций должны быть разные имена и имя вашей функции не может называться так же, как и встроенная в PHP функция**
Давайте рассмотрим пример создания простейшей функции в PHP. Пусть она принимает на вход два числа и возвращает их сумму.

```
<?php


function getSum($x, $y)

{

   return $x + $y;

}
```

Окей, функцию написали, теперь можно её вызвать и посмотреть на результат её работы.

```php
<?php


function getSum($x, $y)

{

  return $x + $y;

}


$a = 5;

$b = 10;


echo getSum($a, $b) . '<br>';

echo getSum(-3, 4);
```
Результат:

15

1

13. PHP. Циклы, условия

Циклы позволяют повторять определенное (и даже неопределенное - когда работа цикла зависит от условия) колличество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией.
PHP поддерживает три вида циклов:

- Цикл с предусловием (**while**);
- Цикл с постусловием (**do-while**);
- Цикл со счетчиком (**for**);
- Специальный цикл перебора массивов (**foreach**).

При использовании циклов есть возможность использования операторов **break** и **continue.** Первый из них прерывает работу всего цикла, а второй - только текущей итерации.

Рассмотрим циклы PHP:

Чтобы далеко не ходить – возьмем цикл for отчет начинать будем с нуля $i = 0 , циклов будет 10- $i < 10; и шаг будет плюс один $++

И получим первую строку программы цикла :

for ($i = 0; $i < 10; $++)

Здесь следует отметить, что нужно представлять, некоторые обозначения, которые могут встречаться, (мы сделаем отдельную страницу, посвящённую обозначениям, их тупо нужно выучить!) к примеру:

$i++ — то же самое, что $i+=1, или $i = $i + 1.

Внутри фигурных скобок напишем тело цикла например самое простое вывод номера цикла : echo $i. "
"; // br - перенос строки

Итоговый листинг будет:

```
for ($i = 0; $i < 10; $++)

{

echo $i. "<br>";

}
```

Результат:

```
0
1
2
3
4
5
6
7
8
9
```

Условия в PHP:

Условия - это одно из важнейший операторов в PHP. Без них нельзя создать ниодин нормальный скрипт. С помощью них скрипт может в разных случаях по-разному работать. Принцип работы с ними простой, думаю Вы поймете сразу. Итак обовсем попорядку.

В операторе **if** ставиться условие, если оно выполняется, то выполняется часть кода следующая в фигурных скобках за ним, если нет, то часть кода идущая после оператора **else**. Также можно создавать сложные условия с помощью **elseif**. Схематично выглядит так:

```
if(условие){

...

}elseif(другое условие){

...

}else{

...

}
```

По-русски это звучит так: "Если условие выполняется, то выполняется одно, если нет, то что-то другое". Сдледующий вопрос - это как создавать условия. Для этого используются переменные и операторы сравнения:

Пример	Название	Результат
$a == $b	Равно	True, если $a равно $b
$a === $b	Идентично	True, если $a равно $b и обе переменных принадлежат одному типу
$a != $b	Не равно	True, если $a не равно $b
$a === $b	Не идентично	True, если $a не равно $b и оба их типа не совпадают
$a > $b	Больше чем	True, если $a больше, чем $b
$a < $b	Меньше чем	True, если $a меньше, чем $b
$a >= $b	Больше или равно	True, если $a больше или равно $b
$a <= $b	Меньше или равно	True, если $a меньше или равно $b

Можно создавать сложные условия:

Пример	Название	Результат
$a == $b && $b == $c	И	True, если $a равен $b, и $b равен $c
$a == $b || $b == $c	Или	True, если $a равен $b, или $b равен $c

Вот пример работы условий:

```
<?

$a = 5;
$b = 6;
$c = 5;

if($a != $b && $b > $c) {

echo "Условие выполняется";

}else{

echo "Условие не выполнилось";

}
```

```
?>
```

В качестве результата должно на экране появиться "Условие выполняется".

14. PHP. Массивы

Массивы – это специальные типы данных. В отличии от других простых переменных, массив может хранить более одного значения.

Создание массива

Массив можно создать с помощью функции **array()**, параметры которой и составляют массив. Параметры могут задаваться парами "ключ=>значение". Если при создании массива ключ не указывается, то индекс определяется положением элемента в массиве (начиная с 0). Например:

```
$рост = array (174, 181, 166); //Массив с индексацией, начинающейся с нуля

$цена = array ("помидоры" => 15, "огурцы" => 12); //Ассоциативный массив

$данные = array (

    "Иванов" => array ("рост" => 174, "вес"  => 68),

    "Петров" => array ("рост" => 181, "вес"  => 90),

    "Сидоров" => array ("рост" => 166, "вес"  => 73)); //Двухмерный массив
```

Массивы можно создать и другим способом - непосредственно. Например:

```
$фрукты[] = "яблоко";

$фрукты[] = "груша";

$фрукты[] = "слива";

$цена["помидор"] = 15;

$цена["огурец"] = 12;
```

Индексы элементов неассоциативного массива можно не указывать. PHP автоматически вычислит их. Если же указать индексы таким образом:

```
$фрукты[0] = "груша";

$фрукты[5] = "яблоко";
```

то в массиве будет два элемента, причем последний с индексом 5. Элементы 1 - 4 не инициализируются.

Можно создать массив с помощью функции **array()**, а затем добавить к нему новый элемент:

```
$фрукты = array("яблоко","груша","слива");

...
```

```
$фрукты[] = "персик";
```

Подсчет количества элементов

Количество элементов в массиве можно определить с помощью функций **count()** или **sizeof()**.

Пример 1

```
<html>

<head>

<title>Размер массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик");

echo "Размер массива \$фрукты равен ".count($фрукты)."<br>";

echo "Последний элемент массива \$фрукты - ".$фрукты[count($фрукты)-1]."<br>";

?>

</body>

</html>
```

РЕЗУЛЬТАТ ПРИМЕРА 1:

```
Размер массива $фрукты равен 4

Последний элемент массива $фрукты - персик
```

Для доступа к последнему элементу надо вычесть 1 из размера массива, так как индексация массива начинается с нуля. Для вывода зарезервированного символа "$" перед знаком доллара стоит символ обратной косой черты "\".

Частоту вхождения элементов в массив можно определить с помощью функции **array_count_values()**. Эта функция возвращает массив, в котором ключами являются элементы исследуемого массива, а значениями - частоты их вхождения в исследуемый массив.

15. PHP. Работа с файлами и папками

Функция file_put_contents() отвечает за создание и запись данных в файл, рассмотрим подробнее:

```
file_put_contents(__DIR__.'/text.txt','');//Создаем файл text.txt
```

```
$data = "Доброе утро!";

file_put_contents(__DIR__.'/text.txt',$data);// Создаем файл text.txt и записываем туда данные



file_put_contents(__DIR__.'/text.txt',$data."\r", FILE_APPEND | LOCK_EX);// Создаем файл text.txt и дописывает туда данные
```

Как видите функция file_put_contents может принимать три параметра file_put_contents(Полный путь и название нового файла, Данные для записи в файл, флаг).
Первые два параметра обязательные, поэтому даже если вы хотите создать файл ничего туда не записывая. Вам все равно придется указать второй параметр в виде пустой строки. Третьим параметрам являются так называемые флаги. FILE_APPEND отвечает за дописывание в файл данных , а LOCK_EX - блокирует файл во время записи.
Да если кто не знает , то __DIR__ - это полный путь к текущему каталогу где выполняется данный код.
Едем дальше, с созданием и записью мы разобрались теперь перейдем непосредственно к чтению файлов. Для этого существует функция file_get_contents():

```
$text = file_get_contents(__DIR__.'/text.txt');

echo $text;
```

Как видите мы благополучно считали данные с ранее созданного файла.
file_get_contents(Полный путь к файлу)
Отмечу что с помощью данной функции мы можем считывать данные с сайтов, указав в качестве пути их домен. К примеру

```
$text = file_get_contents('https://twitter.com/');

echo $text;
```

Как видите у нас подгрузились элементы данного сайта, если же мы хотим посмотреть его исходный код , то достаточно заэкранировать специальные символы:

```
echo htmlspecialchars($text);
```

Такой подход бывает используют при парсинге сайтов.
Теперь перейдем к удалению файлов. Для этого используется функция unlink()

```
unlink(__DIR__.'/text.txt');
```

Здесь мы удалили наш файл text.txt , но тут я хочу вас предостеречь если мы повторно выполним данную команду у нас возникнет ошибка. Почему? Правильно, потому что при повторном использовании функции удаления, такого файла у нас уже не будет и именно из за этого произойдет ошибка.
Это можно исправить написав условие для проверки на существование данного файла.

```
if(file_exists(__DIR__ . '/text.txt')) {

   unlink(__DIR__ . '/text.txt');

}
```

здесь с помощью функции file_exists(путь к файлу) мы проверяем существует ли файл с таким путем. Если да , то функция возвращает нам true и условие if выполняется удаляя данный файл , а если же нет ,возвращается false и данное условие уже не выполняется, так как файла с таким путем уже нет.
Создание папок осуществляется с помощью команды mkdir(Полный путь)

```
mkdir(__DIR__."/css");
```

мы создали в текущей директории папку с названием css теперь чтобы к примеру в папке css создать файл style.css, мы можем воспользоваться переходами по папкам.

```
chdir(__DIR__."/css");

file_put_contents('style2.css','');
```

Функции chdir() передается путь относительно которого мы сможем создавать новые файлы и папки.
Да и вот что еще, если мы захотим создать папку в директории где уже есть папка с таким же названием у нас возникнет ошибка. Здесь опять надо проверить на существование в директории папки по пути:

```
if(!is_dir(__DIR__.'/css')) {

   mkdir(__DIR__ . '/css');

}
```

Функция is_dir(путь) проверяет существует ли директория с заданным путем . В нашем условии говорится если директория не существует , то мы создаем ее.

За удаление директорий в php отвечает функция rmdir(путь):

```
if(is_dir(__DIR__.'/css')) {

   rmdir(__DIR__ . '/css');

}
```

Функция rmdir() удаляет папку причем папка на момент удаления должна быть абсолютно пуста иначе возникнет ошибка.
Чтобы проверить содержимое папки мы можем воспользоваться функцией glob()

```
print_r(glob(__DIR__.'/css/*', 0));
```

Здесь функция glob() возвращает массив значений полных путей файлов и папок которые содержаться в директории css. Следовательно когда количество элементов в массиве станет равно нулю только тогда мы сможем удалить данную директорию.
Пример: Получить список файлов директории в виде массива
То же самое делает функция scandir(), разница в том что у scandir() в массиве будут «.» и «..» и есть возможность сортировки.

```
function list_files($path)

{

  if ($path[mb_strlen($path) - 1] != '/') {

     $path .= '/';

  }



  $files = array();

  $dh = opendir($path);

  while (false !== ($file = readdir($dh))) {

     if ($file != '.' && $file != '..' && !is_dir($path.$file) && $file[0] != '.') {

        $files[] = $file;

     }

  }
```

```
    closedir($dh);
    return $files;
}


print_r(list_files(__DIR__));
Array
(
    [0] => favicon.ico
    [1] => index.php
    [2] => image.jpg
    [3] => robots.txt
)
```

16) SQL. SQL-запросы в PHP

SQL (англ. structured query language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

```
SELECT name FROM Student WHERE SID = 456;
INSERT INTO Grade VALUES ('123', 'CPS130', 'C+');
```

Функция, без которой в языке PHP выполнение SQL-запросов было бы просто невозможным:

```
query(запрос)
```

Данная функция посылает запрос к базе данных и возвращает в случае успешного обращения идентификатор ресурса.

```
$query = "SELECT * FROM `my_sql_table`";
$res = mysql_query($query);
```

SELECT

Оператор языка SQL SELECT предназначен для запросов на выборку данных из базы данных. Названия таблиц и столбцов чувствительны к регистру. Для выбора **всех** столбцов таблицы после слова SELECT нужно ставить звёздочку (*).

SELECT * FROM ИМЯ_ТАБЛИЦЫ

SELECT * FROM STAFF

Для выбора **определённых столбцов** таблицы нам потребуется вместо звёздочки перечислить через запятую названия всех столбцов, которые требуется выбрать:

SELECT ВЫБИРАЕМЫЕ_СТОЛБЦЫ FROM ИМЯ_ТАБЛИЦЫ

SELECT DEPT, NAME, JOB FROM STAFF

DISTINCT

Когда для значений строк таблицы не задано условие уникальности, в результатах запроса могут встретиться одинаковые строки. Часто требуется вывести лишь уникальные строки. Это делается при помощи выражения DISTINCT после оператора SELECT.

```
SELECT DISTINCT column(s) FROM table;
```

```
SELECT SID FROM Grade;
```

SID
142
142
123
857
857
456

```
SELECT DISTINCT SID FROM Grade;
```

SID
142
123
857
456

WHERE

Для выбора определённых строк таблицы вместе с оператором SELECT уже потребуется ключевое слово WHERE, указывающее на некоторое значение или несколько значений, содержащиеся в интересующих нас строках.

SELECT ИМЯ_СТОЛБЦА FROM ИМЯ_ТАБЛИЦЫ WHERE УСЛОВИЕ

SELECT DEPT, NAME, JOB

FROM STAFF WHERE DEPT=38

Наиболее простые условия задаются при помощи операторов сравнения и равенства (<, <=, >, >=, =, <> (не равно)).

Условий может быть несколько, тогда они перечисляются с использованием ключевого слова **AND** (так же есть оператор OR):

```
SELECT DEPT, NAME, JOB
    FROM STAFF WHERE JOB='Clerk' AND DEPT=38
```

```
DEPT   NAME       JOB
------ ---------- -----
    38 Naughton   Clerk
    38 Abrahams   Clerk
```

OR

Пусть требуется выбрать из таблицы Staff имена, должности и число отработанных лет сотрудников, работающих в отделах с номерами 20 или 84:

```
SELECT NAME, JOB, YEARS
    FROM STAFF
        WHERE DEPT=20 OR DEPT=84
```

BETWEEN

Выберем из той же таблицы имена, должности и число отработанных лет сотрудников, зарплата которых между 15000 и 17000 включительно:

```
SELECT NAME, JOB, YEARS
    FROM STAFF
        WHERE SALARY BETWEEN 15000 AND 17000
```

LIKE

Предикат **LIKE** используется для выборки тех строк, в значениях которых встречаются символы, указанные после предиката между апострофами (').

Выберем из таблицы имена, должности и число отработанных лет сотрудников, имена которых начинаются с буквы S и содержат любые другие буквы в любом количестве:

```
SELECT NAME, JOB, YEARS
    FROM STAFF
```

```
WHERE NAME LIKE 'S%'
```

NAME	JOB	YEARS
Sanders	Mgr	7
Sneider	Clerk	8
Scoutten	Clerk	-
Smith	Sales	7

‘S%’ – начинается на S. ‘%S’ – заканчивается на S. ‘%S%’ – содержится где-то между.

Пример:

```
<?php
$dbh = mysql_connect($hostname, $username, $pswd) or die("Не могу соединиться с MySQL.");
mysql_select_db($database) or die("Не могу подключиться к базе.");
$query = "SELECT * FROM `my_sql_table`";
$res = mysql_query($query);
while($row = mysql_fetch_array($res))
{
echo "Номер: ".$row['id']."<br>\n";
echo "Имя: ".$row['firstname']."<br>\n";
echo "Фамилия: ".$row['surname']."<br><hr>\n";
}
?>
```

17) SQL. Подключение к базе данных MySQL

```
<?php
  $conn = new mysqli("namehost", "username", "password", "namebd");
  if ($conn->connect_error) {
    die("Ошибка: не удается подключиться: " . $conn->connect_error);
  }
    $result = $conn->query("SELECT name FROM employee");
  echo "Количество строк: $result->num_rows";
  $result->close();
  $conn->close();
?>
```

1)MySQLi означает MySQL Improved.

new mysqli - инициируют новое соединение, используя расширение Mysqli. Эта функция будет принимать четыре аргумента:

1. Имя хоста, где база данных MySQL работает
2. Имя пользователя для подключения MySQL
3. Пароль для пользователя mysql
4. База данных MySQL для подключения.

query - Функция запроса – Используйте ее, чтобы указать ваш запрос MySQL. В этом примере, мы выбираем столбец имени из базы данных employee.

num_rows – возвращает количество строк

2)Подключение на PHP с использованием функций mysql_ (устар)

```
$db = mysql_connect("host", "username", "password");
mysql_select_db("database name");


# connect to Simpsons database on local computer
$db = mysql_connect("localhost", "stepp", "6uldv8");
mysql_select_db("Simpsons");
$results = mysql_query("SELECT * FROM Grade WHERE SID = 142;");

# loop through each of Bart's course grade records
while ($row = mysql_fetch_array($results)) {
    print("Course ID: {$row['cid']}\n");
}
```

mysql_fetch_array – формирует и выдает массив по выборке из таблицы.

18) SQL. Сортировка записей. Объединение таблиц (JOIN)

ORDER BY

Часто требуется отсортировать строки по порядку номеров, алфавиту и другим признакам. Для этого служит ключевое словосочетание ORDER BY. Такие запросы имеют следующий синтаксис:

SELECT ИМЯ_СТОЛБЦА FROM ИМЯ_ТАБЛИЦЫ WHERE УСЛОВИЕ

ORDER BY СТОЛБЕЦ, ПО КОТОРОМУ СОРТИРУЮТСЯ РЕЗУЛЬТАТЫ

Слово **ASC** указывает, что порядок сортировки - возрастающий. Это слово не обязательно, так как возрастающий порядок сортировки применяется по умолчанию.

Слово **DESC** указывает, что порядок сортировки – убывающий:

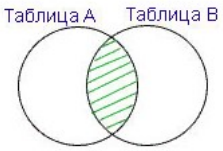
```
SELECT NAME, JOB, YEARS
    FROM STAFF WHERE DEPT=84
        ORDER BY YEARS DESC
```

NAME	JOB	YEARS
Quill	Mgr	10
Edwards	Sales	7
Davis	Sales	5
Gafney	Clerk	5

JOIN

Оператор языка SQL JOIN предназначен для соединения двух или более таблиц базы данных по совпадающему условию.

Запрос с оператором **INNER JOIN** предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON. То же самое делает и **просто JOIN**. Таким образом, слово INNER - не обязательное.



Пример:

Таблица Parts:

Part_ID	Part	Cat
1	Квартиры	505
2	Автомашины	205
3	Доски	10
4	Шкафы	30
5	Книги	160

Таблица Categories:

Cat_ID	Cat_name	Price
10	Стройматериалы	105,00
505	Недвижимость	210,00
205	Транспорт	160,00
30	Мебель	77,00
45	Техника	65,00

Требуется соединить данные этих двух таблиц так, чтобы в результирующей таблице были поля Part (Часть), Cat (Категория) и Price (Цена подачи объявления) и чтобы данные полностью пересекались по условию. Условие - совпадение идентификатора категории в таблице Categories и ссылки на категорию в таблице Parts. Для этого пишем следующий запрос:

```
SELECT PARTS.Part, CATEGORIES.Cat_ID AS Cat, CATEGORIES.Price
```

```
FROM PARTS INNER JOIN CATEGORIES
        ON PARTS.Cat = CATEGORIES.Cat_ID
```

Результатом выполнения запроса будет следующая таблица:

Part	Cat	Price
Квартиры	505	210,00
Автомашины	205	160,00
Доски	10	105,00
Шкафы	30	77,00

LEFT OUTER JOIN (левое внешнее соединение)

Запрос с оператором LEFT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из **первой по порядку (левой)** таблицы, даже если они не соответствуют условию.


```
SELECT PARTS.Part, CATEGORIES.Cat_ID AS Cat, CATEGORIES.Price
    FROM PARTS LEFT OUTER JOIN CATEGORIES
        ON PARTS.Cat = CATEGORIES.Cat_ID
```

Результатом выполнения запроса будет следующая таблица:

Part	Cat	Price
Квартиры	505	210,00
Автомашины	205	160,00
Доски	10	105,00
Шкафы	30	77,00
Книги	160	NULL

RIGHT OUTER JOIN (правое внешнее соединение)

Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из **второй по порядку (правой)** таблицы, даже если они не соответствуют условию.

```
SELECT PARTS.Part, CATEGORIES.Cat_ID AS Cat, CATEGORIES.Price
```

```
FROM PARTS RIGHT OUTER JOIN CATEGORIES
        ON PARTS.Cat = CATEGORIES.Cat_ID
```

Результатом выполнения запроса будет следующая таблица:

Part	Cat	Price
Квартиры	505	210,00
Автомашины	205	160,00
Доски	10	105,00
Шкафы	30	77,00
NULL	45	65,00

FULL OUTER JOIN (полное внешнее соединение)

Запрос с оператором FULL OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из первой (левой) и второй (правой) таблиц, даже если они не соответствуют условию.


```
SELECT PARTS.Part, CATEGORIES.Cat_ID AS Cat, CATEGORIES.Price
    FROM PARTS FULL OUTER JOIN CATEGORIES
        ON PARTS.Cat = CATEGORIES.Cat_ID
```

Результатом выполнения запроса будет следующая таблица:

Part	Cat	Price
Квартиры	505	210,00
Автомашины	205	160,00
Доски	10	105,00
Шкафы	30	77,00
Книги	160	NULL
NULL	45	65,00

Пример:

```
<?php
$dbh = mysql_connect($hostname, $username, $pswd) or die("Не могу соединиться с MySQL.");
mysql_select_db($database) or die("Не могу подключиться к базе.");
$query = " SELECT PARTS.Part, CATEGORIES.Cat_ID AS Cat, CATEGORIES.Price
    FROM PARTS INNER JOIN CATEGORIES
```

```
                ON PARTS.Cat = CATEGORIES.Cat_ID ";
$res = mysql_query($query);
```

19) Механизм Cookie

Ку́ки (англ. cookie, буквально — печенье) — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживания состояния сеанса доступа пользователя;
- ведения статистики о пользователях.

Одни значения cookie могут храниться только в течение одной сессии, они удаляются после закрытия браузера. Другие, установленные на некоторый период времени, записываются в файл. Обычно этот файл называется 'cookies.txt' и лежит в рабочей директории установленного на компьютер браузера.

Для установки Cookies используется функция SetCookie(). Для этой функции можно указать шесть параметров, первый из которых является обязательным:

1) *name* - задает имя (строк), закрепленное за Cookie;
2) *value* - определяет значение переменной (строка);
3) *expire* - время "жизни" переменной (целое число). Если данный параметр не указать, то Cookie будут "жить" до конца сессии, то есть до закрытия браузера. Если время указано, то, когда оно наступит, Cookie самоуничтожится.
4) *path* - путь к Cookie (строка);
5) *domain* - домен (строка). В качестве значения устанавливается имя хоста, с которого Cookie был установлен;
6) *secure* - передача Cookie через защищенное HTTPS-соединение.

Обычно используются только три первые параметра.

Пример установки Cookies:

```php
<?php
// Устанавливаем Cookie до конца сессии:
SetCookie("Test","Value");

// Устанавливаем Cookie на один час после установки:
SetCookie("My_Cookie","Value",time()+3600);
?>
```

Получить доступ к Cookies и их значениям достаточно просто. Они хранятся в суперглобальных массивах и $_COOKIE и $HTTP_COOKIE_VARS.

Доступ к значениям осуществляется по имени установленных Cookies, например:

```
echo $_COOKIE['my_cookie'];

// Выводит значения установленной Cookie 'My_Cookie'
```

Пример установки Cookie и последующего его чтения:

```
<?php
// Устанавливаем Cookie 'test' со значением 'Hello' на один час:
SetCookie("test","Hello",time()+3600);
// При следующем запросе скрипта выводим 'Hello':
echo @$_COOKIE['test'];
?>
```

20) Механизм Сессий

Сессия - это сеанс, во время которого система идентифицирует зашедшего на сайт пользователя. Для этого надо выдать ему уникальный идентификатор и попросить передавать его с каждым запросом. Идентификатор - это обычная переменная. По умолчанию ее имя - PHPSESSID.

Существует несколько способов передачи *идентификатора сессии*:

- С помощью *cookies*.

Cookies были созданы специально как метод однозначной идентификации клиентов и представляют собой расширение протокола HTTP. В этом случае *идентификатор сессии* сохраняется во временном файле на компьютере клиента, пославшего запрос. Метод, несомненно, хорош, но многие пользователи отключают поддержку *cookies* на своем компьютере из-за проблем с безопасностью.

- С помощью параметров адресной строки(сессии).

В этом случае *идентификатор сессии* автоматически встраивается во все запросы (URL), передаваемые серверу, и хранится на стороне сервера.

Например: *адрес* http://green.nsu.ru/test.php превращается в *адрес* http://green.nsu.ru/test.php?PHPSESSID=ac4f4a45bdc893434c95dcaffb1c1811

Для инициализации новой или возобновления ранее созданной необходимо вызвать PHP-функцию *session_start()*. Данную функцию необходимо вызывать до вывода контента в коде. Сессионные параметры находятся в глобальном массиве *$_SESSION*. Рассмотрим пример:

```php
session_start();
if ( ! isset($_SESSION['test']) )
{
    echo 'Сохраняем значение...<br />';
    $_SESSION['test'] = 'Hello, world';
}
echo $_SESSION['test'];
```

При первом выполнении скрипта создастся новая сессия функцией *session_start()*. Условие выполнится, так как сессионной переменной test не существует и при этом будет создана данная сессионная переменная. При последующих выполнениях функция *session_start()* будет возобновлять ранее созданную сессию, условие выполняться не будет, а будет лишь выводиться значение переменной test.

По умолчанию сессия уничтожается при закрытии клиентом браузера. Но можно задать и конкретное время жизни сессии в php.ini (указывается время в секундах):

session.cookie_lifetime = 0

В скрипте сессию можно принудительно уничтожить функцией *session_destroy()*.