Практическая работа №1. Работа с текстом. Списки

Тема: Работа с текстом в “HTML5”
Цель: Научиться работать с текстом в “HTML5”

Ход работы

Теоретическая часть

Добавление заголовка на веб-страницу осуществляется при помощи тега <hN></hN>, где N – индекс, от 1 до 6. Чем выше, тем меньше шрифт по умолчанию. Добавление абзацев осуществляется при помощи открывающего и закрывающего тега p, с помощью этого элемента и псевдоэлемента :before и :after можно оформить цитату, также это можно сделать при помощи тега <blockquote>.

Формирование текста осуществляется при помощи следующих тегов:

Полужирный текст

Выделение текста курсивом

<u>Подчёркнутый текст</u>

<s>~~Зачеркнутый текст~~</s>

<small>Маленький шрифт</small>

<big>Большой шрифт</big>

Знак сноски^{сверху},индекс _{снизу} от текста.

Списки создаются при помощи тегов ul/ol, которые создают маркированный и нумерованный списки. Для создания элемента списка используется тег li. Можно делать вложенные списки, комбинировать различными способами.

Практическая часть

Создайте титульную страницу и содержание книги:

1. Создайте титульную страницу книги, которая должна содержать название, автора, год издания;
2. Создайте страницу с цитатой, взятой из любого произведения;
3. Создайте оглавление книги, содержащее название разделов, глав, общее количество страниц, используя при этом маркированные/нумерованные списки.
4. Составить отчет о проделанной работе и ответить на контрольные вопросы

Контрольные вопросы:

1. Какие теги для работы с текстом Вы знаете?
2. Как осуществляется добавление заголовков и абзацев на веб-страницу?
3. Какие существуют способы оформления цитат? Опишите их.
4. Какие существуют виды списков?

Практическая работа №2. Работа с таблицами, создание таблиц.

Тема: Создание таблиц
Цель: Научиться создавать и работать с таблицами.

Ход работы

Теоретическая часть

Любая таблица представляет собой набор сток и столбцов, на пересечении которых находятся ячейки.
Таблица создается при помощи тегов <table></table>,
Название таблицы указывается между тегами <caption></caption>.
Строки таблицы – теги <tr></tr>, столбцы таблицы – теги <td></td>.
Столбцы имеют названия, расположенные в первой строке – теги <th></th>.
У тега <table>существует ряд параметров:

- ✓ width – задает ширину таблицы (в px или % от ширины экрана);
- ✓ bgcolor – задает цвет фона ячеек таблицы;
- ✓ background – заливает фон таблицы рисунком;
- ✓ border – задает рамку указанной ширины в px вокруг всей таблицы;
- ✓ cellpadding – задает отступ в px между границей клетки и ее содержимым.

Также есть атрибуты:
align – задает выравнивание таблицы:слева (left),справа (right),по центру (center);
cellspacing – задает расстояние между ячейками таблицы в px;
cellpadding – задает расстояние между текстом и внутренней границей ячейки теблицы в px.

Пример создания таблицы:

```
<html>
<head>
<title>html table</title>
</head>
<body>
    <table   width="300"   bgcolor="plum"   border="1"   align="center"
    cellspasing="5" cellpadding="10" rules="rows" >
     <caption>Название таблицы</caption>
     <tr><th>1</th><th>2</th><th>3</th></tr>
     <tr><th>11</th><th>12</th><th>13</th></tr>
     <tr><th>21</th><th>22</th><th>23</th></tr>
     <tr><th>31</th><th>32</th><th>33</th></tr>
    </table>
</body>
</html>
```

Результат:

Название таблицы

1	2	3
11	12	13
21	22	23
31	32	33

Тег colspan – объединение столбцов
Тег rowspan – объединение строк

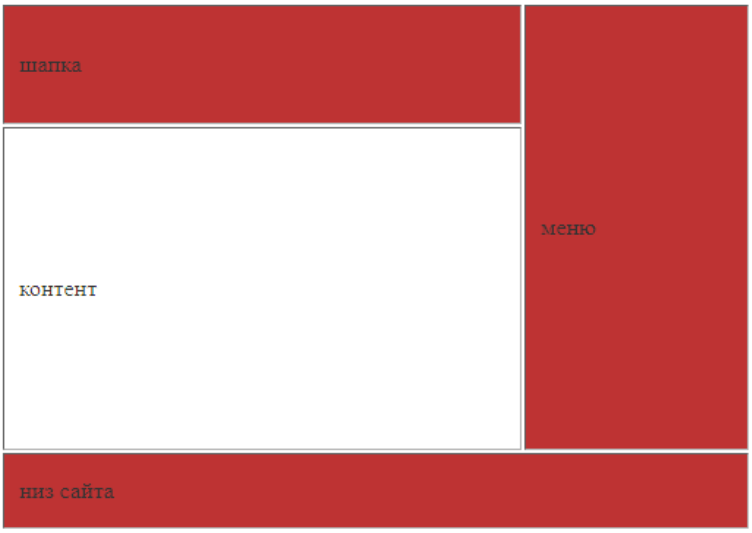
Пример:

```
<html>
<head>
<title>Заголовок</title>
</head>
<body>
    <table    width="715"    border="1"  align="center"  cellspasing="0"
    cellpadding="10"  >
     <tr bgcolor="darkred">
          <td rowspan="2" width="30%">меню</td>
          <td height="60">шапка</td>
     </tr>
     <tr bgcolor="oldlace">
     <td height="200">контент</td>
     </tr>
    </table>
</body>
</html>
```

Результат:


```
<html>
<head>
<title>Заголовок</title>
</head>
<body>
    <table    width="715"    border="1"  align="center"  cellspasing="0"
    cellpadding="10"  >
     <tr bgcolor="darkred">
          <td  width="70%" height="60">шапка</td>
          <td rowspan="2" >меню</td>
     </tr>
     <tr bgcolor="oldlace">
     <td height="200">контент</td>
     </tr>
     <tr bgcolor="darkred">
     <td colspan="2" height="30">низ сайта</td>
     </tr>
    </table>
</body>
</html>
```

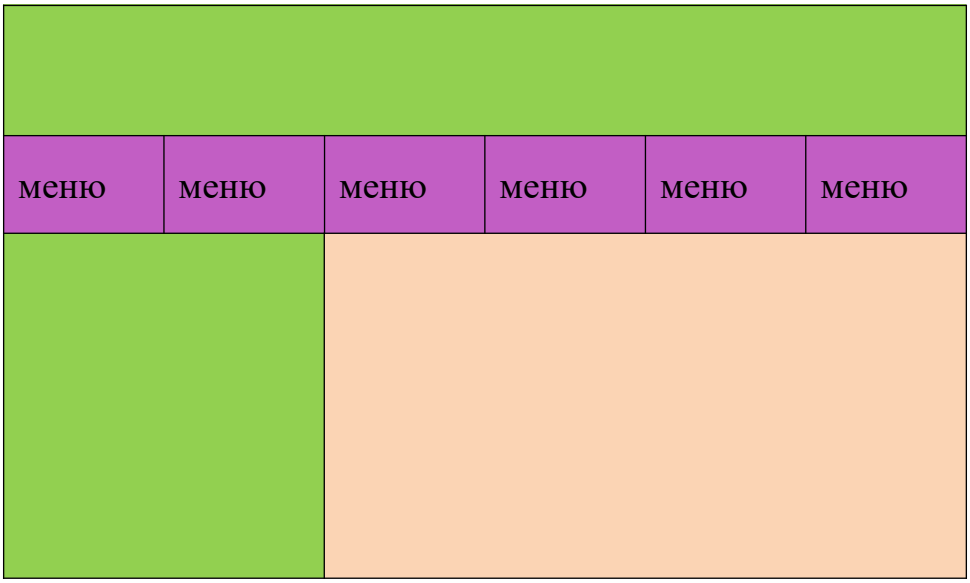
Результат:



Практическая часть

1.Создайте календарь на неделю.

2.Создайте следующую таблицу:



3. Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Какие существуют теги для добавления таблиц на страницу?
2. Как происходит объединение строк и столбцов в таблице?
3. Какие вы знаете атрибуты таблицы?

Практическая работа №3. Работа с мультимедиа и frame

Тема: Добавление мультимедийных файлов на веб-страницу
Цель: Научиться добавлять файлы мультимедиа

Ход работы

Теоретическая часть

В HTML5 существует два тега для добавления аудио и видео файлов: <audio> </audio>, video> </video>. Каждый из них имеет свои атрибуты:

- ✓ src – указывает путь к файлу, путь указывается в двойных ковычках;
- ✓ autoplay – указывает браузеру, что элемент начинает воспроизводится автоматически после загрузки страницы.
- ✓ loop – указывает сколько раз нужно проиграть элемент
- ✓ alt – указывает на альтернативный текст, который отобразится на странице, если не загрузится изображение, видео, аудио.
- ✓ preloader – указывает на ссылку изображения, которое будет отображаться, пока не загрузится видео.

Добавление изображений на страницу возможно при помощи тега img, который также имеет атрибуты src и alt.

Практическая часть

Создайте страницу, содержащую в себе элементы мультимедиа:

1. Создайте страницу, которая будет содержать фото, видео и аудио файлы. Добавьте им различные атрибуты.
2. Добавьте изображение, в котором указан неправильный адрес, чтобы сработал вывод альтернативного текста.
3. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Какие существуют теги для добавления файлов мультимедиа на страницу?
2. Какие атрибуты вы знаете для тегов, добавляющих музыку и видео на сайт?
3. Как добавляются изображения на сайт? Какие существуют теги для группировки содержимого?

Практическая работа №4. Подключение CSS стилей в документ. Форматирование текста.

Тема: Каскадные таблицы стилей
Цель: Научиться работать с таблицами стилей css.

Ход работы

Теоретическая часть

Подключение стилей в документ возможно тремя способами:

1. Внутри открывающего тега добавить атрибут style=”…”;
2. Между тегами head добавить тег style и писать все стили внутри него;
3. Вынести стили в другой документ. Документ каскадных таблиц стилей, который имеет расширение *.css.

Чтобы применить стиль к какому-либо тегу, необходимо сначала обратиться к нему. Это, так называемые селекторы css. Например: h1{color:red;margin:20px;}

Этот код сделает все заголовки первого уровня красными с отступом вокруг них 20px. Также можно обращаться к классам и идентификаторам. Поддерживается наследование. Например, если есть некоторый блок с классом main, в котором находится заголовок первого уровня, то в первом случае необходимо обратиться в к самому блоку, задать его ширину и высоту, а во втором случае необходимо обратиться к заголовку внутри этого блока. (! Первый вариант изменит ВСЕ заголовки первого уровня на сайте, а второй изменит только те, которые будут находиться внутри элемента с классом main).

```
.main {width:20%; height:100px;}
.main h1 {color:red; margin:20px;}
```

Некоторые css свойства элементов:

width – ширина
height – высота
position – позиционирование
margin – внешние отступы
padding – внутренние отступы
font-size – размер шрифта
font-style – начертание шрифта
color – цвет текста
display – свойство дисплей
background – свойство фона
border – границы
vertical-align – вертикальное выравнивание
text-align – выравнивание текста по горизонтали

Подключение шрифтов в документ:

```
@font-face{font-family: название шрифта;
        src:url(путь к шрифту, расширение);}
```

Практическая часть

1. Создайте документ. Создайте заголовок, в котором отобразите название статьи. Сделайте его золотистым;
2. Под ним создайте цитату, задайте ей размер шрифта, цвет и тип шрифта;
3. Вставьте абзац с текстом на три строки (Lorem…..).
4. Задайте ему внешние отступы, цвет;
5. Создайте календарь на текущий месяц при помощи таблицы, оформите его при помощи стилей. Задайте начертание границ, выравнивание текста в ячейках, выходные дни сделайте красным цветом, текущий день курсивом и жирным. Ячейки залейте каким-либо цветом.
6. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Перечислите все 3 варианта подключения стилей.
2. Какой из способов указания стилей приоритетнее?
3. Как подключать шрифты в документ?

Практическая работа №5. Стилизация веб-страниц. Установка отступов, границ элементов.

Тема: Установка отступов, границ, стилизация элементов.
Цель: Научиться работать с границами и отступами. Создание фигур при помощи границ.

Ход работы

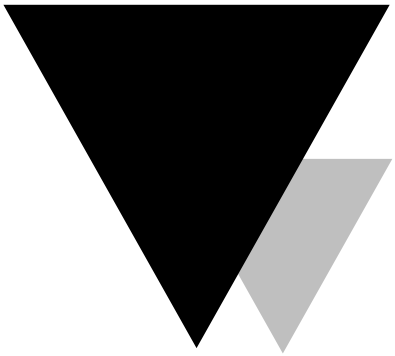
Теоретическая часть

Все блочные элементы обладают двумя типами отступов. Внешние – margin, определяют отступы от соседних элементов, и внутренние – padding, определяют расстояние внутреннего содержимого этого элемента до его границ.
Свойство border отвечает за границы элемента. Можно указать сразу все 4 границы или по одной, например:
`border:10px solid green;`(появится рамка вокруг элемента, толщиной в 10px, зеленого цвета)
`border-top:10px solid red;` (появится нижнее подчеркивание элемента, толщиной в 10px, красного цвета)
Свойства границ:
border-color:; - цвет
border-style:; - начертание границ
border-width:; - толщина границ
Эти три свойства объединяются одним общим свойством border:<толщина><тип начертания><цвет>;
border-radius: значение (в px или %); - скругление. Может быть задано отдельно для каждого угла.
ВАЖНО: при задании внутренних отступов играни элемента, увеличивается либо уменьшается его ширина и высота, из-за чего может «поехать» вся верстка. Чтобы этого избежать, следует использовать свойство box-sizing. Оно определяет тип блочной модели.
Принимает свойства: content-box
Основывается на стандартах CSS, при этом свойства width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.
border-box – свойства width и height включает в себя значения полей и границ, ноне отступов (margin)
padding-box – свойства width и height включает в себя значения полей, но не отступов (margin) и границ (border)
inherit – наследует значение родителя.

Практическая часть

1. Создайте блок с шириной 500px, высотой 300px, задайте внешний отступ слева 100 px, сверху 50 px, и такие же внутренние отступы. Поместите внутрь него текст, посмотрите, что произошло. Добавьте ему границу произвольной толщины и цвета. Тип границы – solid. Посмотрите на элемент. Замените тип границы на dotted.
2. Создайте квадратный блок, залейте его произвольным цветом.
3. Создайте прямоугольный блок, залейте его произвольным цветом.
4. Создайте овальный блок, для этого сделайте прямоугольник, залейте его произвольным цветом и задайте скругление углов в 50%.
5. Создайте треугольник. Для этого создайте блок, дайте ему 0 px ширины и

высоты, и задайте границы. Например, для создания треугольника, смотрящего острием вниз необходимо сделатьдве его противоположные границы прозрачными, а одну (в данном случае верхнюю) цветную.



HTML:

```
<div id=”tr”></div>
```

CSS:

```
#tr { width:0px;
          height:0px;
          border-top:140px solid black;
          border-left: 70px solid transparent;
          border-right:70px solid transparent;}
```

6. Создайте аналогичным образом треугольники, смотрящие влево, вправо, вверх
7. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Опишите как создать прямоугольник, смотрящий вверх?
2. Какое свойство необходимо использовать чтобы создать овал или круг?
3. Перечислите свойства границ.

Практическая работа №6. Создание файла сброса стилей.

Тема: Сброс стандартного отображения элементов веб-страницы в браузерах. Написание файла сброса стилей.
Цель: Научиться приводить стили к общему виду во всех браузерах.

Ход работы

Теоретическая часть

Все браузеры для вывода содержимого используют встроенный движок. Существуют три основных движка, на которых написаны все существующие современные обозреватели.

Каждый такой движок по-своему отображает одни и те же элементы на веб-странице, и чтобы при верстке под один браузер быть уверенным, что весь сайт не поломается в другом браузере, нужно перед версткой привести все основные элементы, которые будут использоваться в нем, к общему виду. Это называется обнулением (сбросом) стилей.

Стили применяются и загружаются так, как они подключены в HTML, поэтому чтобы провести сброс, необходимо подключить файл сброса стилей до всех остальных файлов, так как в противном случае, этот файл обнулит всю верстку.

Файл сброса делается следующим образом:
Выбираются CSS селекторы и группируются в зависимости от того, какие стили нужно сбросить. Далее им присваивается отображение по умолчанию, например:

```
div, section, iframe, figure, figcaption, form, lable { margin:0; padding:0; display:block; box-sizing:border-box; }
```

Таким образом, для всех элементов выше задаются нулевые отступы, одинаковая блочная модель и поведение по умолчанию.

Практическая часть

1. Сверстайте простую веб-страницу
2. Сделайте скриншот о проделанной работе.
3. Напишите файл сброса стилей для элементов, используемых на веб-странице и подключите его к документу.
4. Сделайте скриншот. Сравните полученный результат.
5. Добавьте стили к элементам.
6. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Для чего нужен файл сброса стилей?
2. Какое расширение имеет файл сброса стилей?
3. Какая очередность подключения у файла сброса стилей и файла стилей?
4. Что такое универсальный селектор?

Практическая работа №7. Изучение видов позиционирования элементов при помощи CSS3. Центрирование элементов.

Тема: Позиционирование элементов с помощью CSS3.
Цель: Научиться позиционировать элементы на странице.

Ход работы

Теоретическая часть

Существует четыре вида позиционирования элементов на сайте.

static – стоит по умолчанию, нет смысла применять к элементу, если не нужно обнулить позиционирование.

relative – относительное позиционирование, можно выровнять блок по центру страницы при помощи свойства margin:0 auto; отступы задаются при помощи margin. Элемент может сдвигаться, но место в потоке под него остается.

absolute – абсолютное позиционирование, в этом случае элемент «выдирается» из потока на странице, ниже идущие элементы поднимаются на его место, а сам элемент прижимается в верхний левый угол страницы. Позиционируются элементы при помощи свойств left, right, top, bottom. Чтобы позиционировать относительно границ нужного контейнера, родителю необходимо задать относительное позиционирование.

fixed – так же, как и абсолютное, только элемент при прокрутке всегда будет находиться на своем месте и никогда не покинет экран.

Горизонтальное центрирование:

Относительное позиционирование – margin(отступ сверху и снизу) auto(выравнивает по центру);

Абсолютное и фиксированное – необходимо сдвинуть элемент на 50% влево и задать отрицательный отступ слева, равный половине ширины элемента;

Можно сделать элемент строчным или строчно-блочным, а к родителю применить выравнивание текста по центру.

Вертикальное центрирование:

Необходимо дать родительскому элементу относительное позиционирование, а дочернему следующие свойства:

Родитель{position:relative;},Дочерний{position:absolute;top:50%;transform:translate(-50%);}

При помощи таблицы:

Родительский{display:table;} Дочерний{display:table-cell;vertical-align:middle;}

Практическая часть

1. Создайте четыре блока размером 400x400px. Залейте их разными цветами.
2. Первому задайте статическое позиционирование.
3. Второму задайте абсолютное позиционирование. Посмотрите на то, как повели себя остальные элементы. При помощи свойств right, top прижмите его к верхнему правому углу.
4. Третьему задайте фиксированное позиционирование. При помощи свойств bottom, left прижмите его к левому нижнему углу.
5. Четвертому задайте относительное позиционирование. Центрируйте его по горизонтали, внутри этого блока создайте еще один контейнер, которому задайте абсолютное позиционирование, половину ширины и высоты родительского блока. Залейте его произвольным цветом. Центрируйте по вертикали и горизонтали.
6. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Перечислите и опишите виды позиционирования.
2. Какое позиционирование можно не указывать?
3. При каком позиционировании элемент не исчезает с экрана?
4. Как горизонтально центрировать элемент, которому задано абсолютное позиционирование?

Практическая работа №8. Верстка простой модульной сетки.

Тема: Создание простого двух-колоночного макета.
Цель: Научиться создавать простые макеты страницы.

Ход работы

Теоретическая часть

При создании простого макета веб-страницы можно пользоваться разными способами верстки. Один из них – совместное использование относительного и абсолютного позиционирования. При этом, координаты при абсолютном позиционировании указываются от верхней левой границы родительского блока, обладающим относительным позиционированием, а тот располагается при помощи margin.

1. Перед тем как начать верстать сайт, необходимо продумать его структуру.
2. Нет необходимости заполнять сразу вновь созданный тег. Сперва необходимо продумать основную конструкцию, обычно строение сайтов одинаково – шапка=> какой-то блок-слайдер (не обязательно) => общий блок, в котором располагается основной контент сайта => подвал.
3. После того, как основная структура создана – можно приступать к заполнению незаполненных блоков.

Способ 1

1. Создайте контейнер с id=”main”, которому задайте абсолютное позиционирование. Дайте ему 100% ширины, и 600px высоты. Залейте его любым цветом. Присвойте ему положение: top:0; left:0;
2. Создайте внутри этого контейнера элемент <header>. Задайте ему ширину в 100% и высоту в 60px. Присвойте точно такое же положение, как и блоку выше. Это будет шапка сайта. Залейте произвольным цветом.
3. Создайте контейнер с классом content. Задайте ему абсолютное позиционирование, 100% ширины и 350px высоты и следующее положение: top:70px; left:0; Залейте произвольным цветом.
4. Создайте внутри этого контейнера элемент <footer>. Задайте ему ширину в 100% и высоту в 70px, а также следующее положение: top:430px; left:0; Это будет подвал сайта. Залейте произвольным цветом.

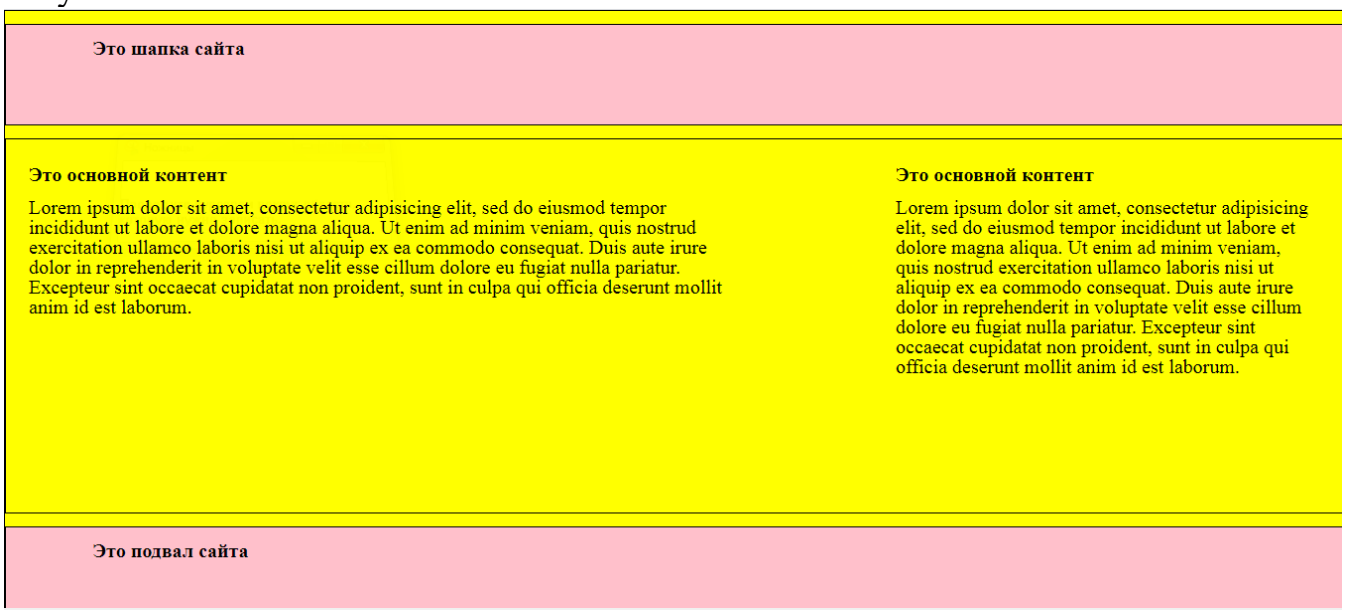
Для расположения элементов таким способом требуется постоянное вычисление отступов и положения. Это неудобно.

Результат способа 1:

Способ 2

1. Подключите файл сброса стилей.
2. Создайте контейнер с относительным позиционированием. Присвойте ему id=”main”. Дайте ему 100%, высоту пока не трогайте. Задайте фоновый цвет.
3. В стилях создайте класс wrapper. Задайте ему ширину 960px (стандарт расположения контента под все расширения). Выровняйте по центру страницы (margin:0 auto;) Внутри этого блока будет располагаться весь контент.
4. Внутри контейнера main создайте блок с классом header – это будет шапка сайта. Задайте ей 100% ширины, высота – 30-40px, залейте цветом. Внутри шапки создайте блок с классом wrapper. Создайте внутри этого блока заголовок с фразой: «Это шапка сайта».
5. Внутри контейнера main создайте блок с классом content. Задайте ему 100% ширины и 300px высоты. Внутрь него добавьте блок с классом wrapper. В нем создайте два контейнера – 300px и 600px. Дайте им абсолютное позиционирование. Один прижмите вверх и влево, второй вверх и вправо. В каждый из них поместите заголовок с текстом: «Это основной контент» и абзац с текстом: «lorem…..»
6. Создайте блок с классом footer. Сделайте все тоже самое, что и с шапкой сайта, текст: «Это подвал сайта»
7. Каждому блоку, кроме блока с классом wrapper, задайте сплошную границу, толщиной в 1px и разным цветом. Задайте внешние отступы в 10px.

Результат способа 2:



Практическая часть

1. Сверстайте веб-страницы обеими способами
2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Опишите основную конструкцию любого сайта.
2. Какой из способов верстки лучше? Почему?

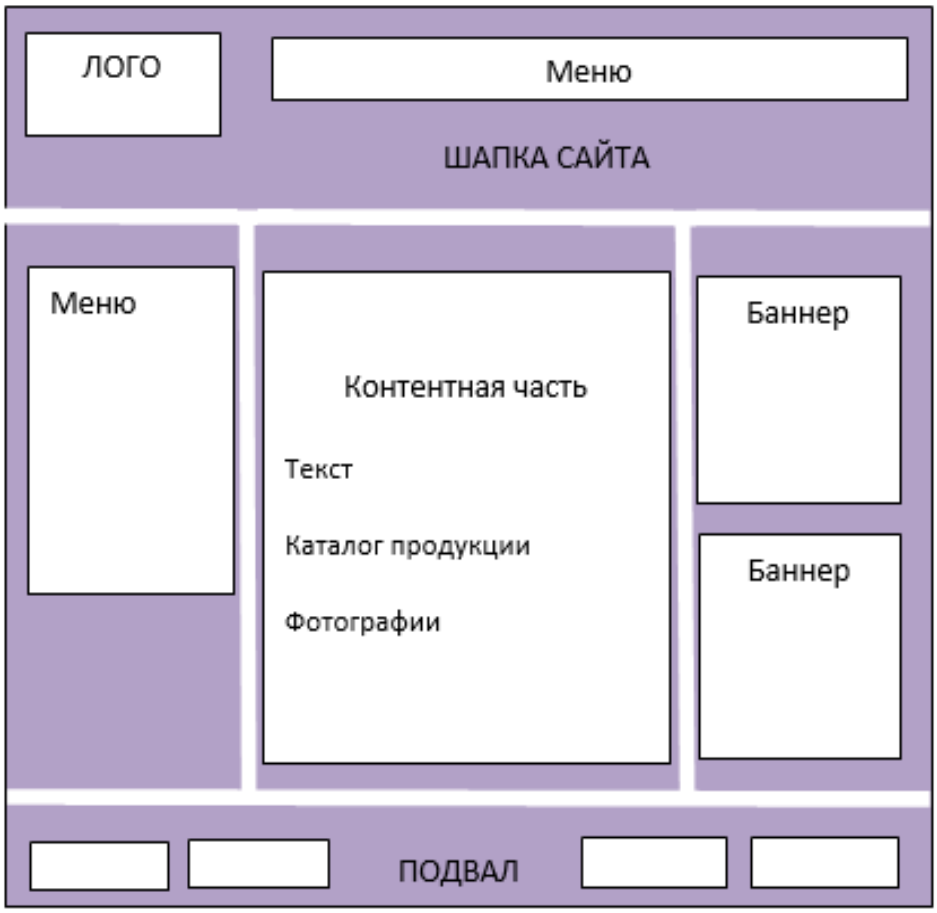
Практическая работа №9. Использование абсолютного и относительного позиционирования. Верстка простого макета сайта.

Тема: Верстка простой веб-страницы.
Цель: Научиться верстать простые веб-страницы.

Ход работы

Теоретическая часть

Перед версткой необходимо условно разбить сайт на компоненты, представить его структуру. Например:



Все содержимое оборачивается в один блок-каркас. Внутри создается еще три-четыре блока. Шапка, контент, подвал. Шапка сайта делится на две части – логотип и меню. Контент делится на три части: меню, главную часть и баннерную рекламу. В свою очередь баннерная часть делится еще на два элемента, в которые помещается реклама. И подвал делится еще на две части, внутри которых находятся два блока, которые в свою очередь содержат еще по два элемента.

Пример создания горизонтального меню:

HTML
```
<ul class:=”clearfix”>
     <li><a href=””>Item1</a></li>
     <li><a href=””>Item2</a></li>
     <li><a href=””>Item3</a></li>
</ul>
```

CSS
```
.clearfix:after{
    content:"";
    display:table;
    clear:both;}
       .li{
    float:right;
    margin:0 10px;}
```
(При помощи float прижимаем все элементы списка вправо и задаем им определенные отступы между собой. Clearfix применяется для того, чтобы отменить схлопывание родителя элементов (тега ul)).

Либо:

HTML
```
<ul>
    <li><a href="">Item1</a></li>
    <li><a href="">Item2</a></li>
    <li><a href="">Item3</a></li>
</ul>
```
CSS
```
.li{
    display:inline-block;
    margin:0 10px;}
```
(Изменяем стандартное отображение элементов списка и делаем их строчно-блочными, что позволяет расположить их в одну строку и при этом задать отступы (при display:inline задать отступы не возможно так как у строчных элементов их нет))

Практическая часть

1. Сверстайте пример страницы сайта, указанный в лабораторной работе, при помощи относительного и абсолютного позиционирования.

**Сверстайте полноценную страницу «Новостной портал» с логотипом, меню, слайдером и пятью новостями с иллюстрациями.

2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Для чего нужен был класс clearfix (в данном примере)?
2. Для чего нужен float?
3. Разница между display:inline и display:inline-block?

Практическая работа №10. Верстка простой модульной сетки при помощи плавающих блоков.

Тема: Верстка простых модульных сеток при помощи свойства float.
Цель: Научиться использовать float для верстки.

Ход работы

Теоретическая часть

Изначально свойство float использовалось для изображений, чтобы задать им обтекание, но его можно использовать и для верстки сайтов. Существует три значения для свойства float:
right – элемент прижимается вправо
left – элемент прижимается влево
none – отмена обтекания
Существует свойство clear для очистки float, оно может принимать три значения:
both – следующий элемент располагается ниже блоков, которым присвоено свойство float
left – элементы располагаются под блоком, у которого значение свойства float задано как left
right – элементы располагаются под блоком, у которого значение свойства float задано как right

Однако, при верстке таким образом, возникает некоторая сложность. Если дочернему контейнеру назначить float, то если у родительского контейнера не будет задана высота – он просто схлопнется и перестанет быть видимым. Чтобы этого избежать необходимо добавить класс clearfix к родителю. Этот класс имеет следующие свойства:

```
.clearfix:after{
     content:”“;
     display:table;
     clear:both;}
```

Применяется очистка float, задается псевдоэлемент, у которого меняется отображение и задается контент. После этого родитель начинает учитывать размер дочерних элементов. Например:

```
<ul class:=”clearfix”>
     <li><a href=””>Item1</a></li>
     <li><a href=””>Item2</a></li>
     <li><a href=””>Item3</a></li>
</ul>
```

```
<style>
     *{margin:0;
         padding:0;
     -webkit-box-sizing:border-box;
     -moz-box-sizing:border-box;
     box-sizing:border-box;}
.clearfix:after{
     content:”“;
     display:table;
     clear:both;}
```

```
.clearfix{clear:both;}
ul{border:1px solid black;}
li{float:left;
    margin:10px;}
</style>
```

Практическая часть

1. Подключите файл сброса стилей. Сразу создайте класс .clearfix для отмены схлопывания.

2. Создайте два блока и абзац с любым текстом. Задайте блокам ширину – 400-500px, высоту – 200-400px и рамку. Одному присвойте float:right. Посмотрите, как ведет себя текст и блок, задайте блоку и тексту очистку float при помощи свойства clear. Попробуйте все три варианта.

3. Удалите содержимое страницы. Создайте общий контейнер с идентификатором main, внутри которого создайте еще три блока с классами header, content, footer.

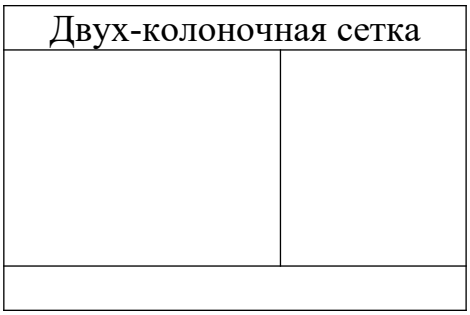
4. Блоку с id main задайте ширину в 960px, относительное позиционирование, центрируйте его по горизонтали.

5. Шапке сайта задайте 100% ширины, 50-60px высоты. Залейте любым цветом. Добавьте в нее заголовок с текстом: «Это шапка сайта».

6. Подвалу сайта, так же, как и шапке задайте 100% ширины, 50-60px высоты. Залейте любым цветом. Добавьте в него заголовок с текстом: «Это подвал сайта».

7. Верстка главной части. Блоку с классом left задайте 600px ширины, float:left; Внутрь него поместите произвольный текст. Блоку с классом right задайте 360px ширины, float:right; Внутрь него поместите произвольный текст.

8. Можно увидеть, что центральная часть схлопнулась и подвал поднялся вверх. Добавьте ранее созданный класс clearfix к центральной части. Должно получиться примерно так: `<……class="main clerfix">` Теперь контент отображается правильно.



9. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Опишите свойства float?
2. Для чего нужно свойство clear? Какие может принимать значения?
3. Что произойдет если не сделать очистку float?

Практическая работа №11. Верстка сложных модульных сеток.

Тема: Верстка сложных модульных сеток.
Цель: Научиться создавать макеты страницы.

Ход работы

Теоретическая часть

static – стоит по умолчанию, нет смысла применять к элементу, если не нужно обнулить позиционирование.

relative – относительное позиционирование, можно выровнять блок по центру страницы при помощи свойства margin:0 auto; отступы задаются при помощи margin. Элемент может сдвигаться, но место в потоке под него остается.

absolute – абсолютное позиционирование, в этом случае элемент «выдирается» из потока на странице, ниже идущие элементы поднимаются на его место, а сам элемент прижимается в верхний левый угол страницы. Позиционируются элементы при помощи свойств left, right, top, bottom. Чтобы позиционировать относительно границ нужного контейнера, родителю необходимо задать относительное позиционирование.

fixed – так же, как и абсолютное, только элемент при прокрутке всегда будет находиться на своем месте и никогда не покинет экран.

Практическая часть

1.Подключите файл сброса стилей. Создайте три документа: index.html, reset.css, main.css.

2.Внутри тега head пропишите два тега <link rel=”stylesheet”href=””>. Подключите сброс, а затем main.css.

3.В файле сброса сначала нужно определить блочную модель, затем убрать отступы, задать единый размер шрифта и цвет, убрать подчеркивания и маркеры у ссылок и списков, убрать обводку у объектов при нажатии или получении ими фокуса.

```
*{    -webkit-box-sizing:border-box;
      -moz-box-sizing:border-box;
      box-sizing:border-box;
      outline:none;}
      h1,h2,h3,h4,h5,h6,p,a{
            font-size:16px;
            color:#000;}
html,body,div,span,n,h1,h2,h3,h4,h5,h6,a,ol,ul,li,table,tr,th,td,article,
aside,figure,figcaption,footer,header,section,audio,video,:after,:before{
      margin:0;
      padding:0;
      border:0;
      font-size:100%;
      font:inherit;
      vertical-align:baseline;}
ol,ul{list-style:none;}
a,li{list-style:none;
      text-decoration:none
      color:#000;}
```

4.Разметка страницы. Создайте главный блок с 100% шириной, внутри него блок с фиксированной шириной в 960px, внутри которого будут шапка сайта, тело и подвал.

5.Элементам пропишите следующие свойства:

Шапка:

Ширина 100%, высота 30px, граница 1px solid #000, внешний отступ сверху 20px.

Главный контейнер:

Ширина 100%, относительное позиционирование, граница 1px solid #000, внешний отступ сверху 20px.

Подвал:

Ширина 100%, высота 50px, заливка rgb прозрачным каналом rgba(0,0,0,0.6), внешний отступ сверху 20px.

6.Внутри главного блока создайте два блока с одинаковыми классами, чтобы применить к ним общие свойства. Задайте им высоту в 300px, ширину в 430px, границу толщиной в 1px, сплошным слоем зеленого цвета, внешние в 20px, относительное позиционирование. Задайте им обрезку границ (overflow:hidden;)

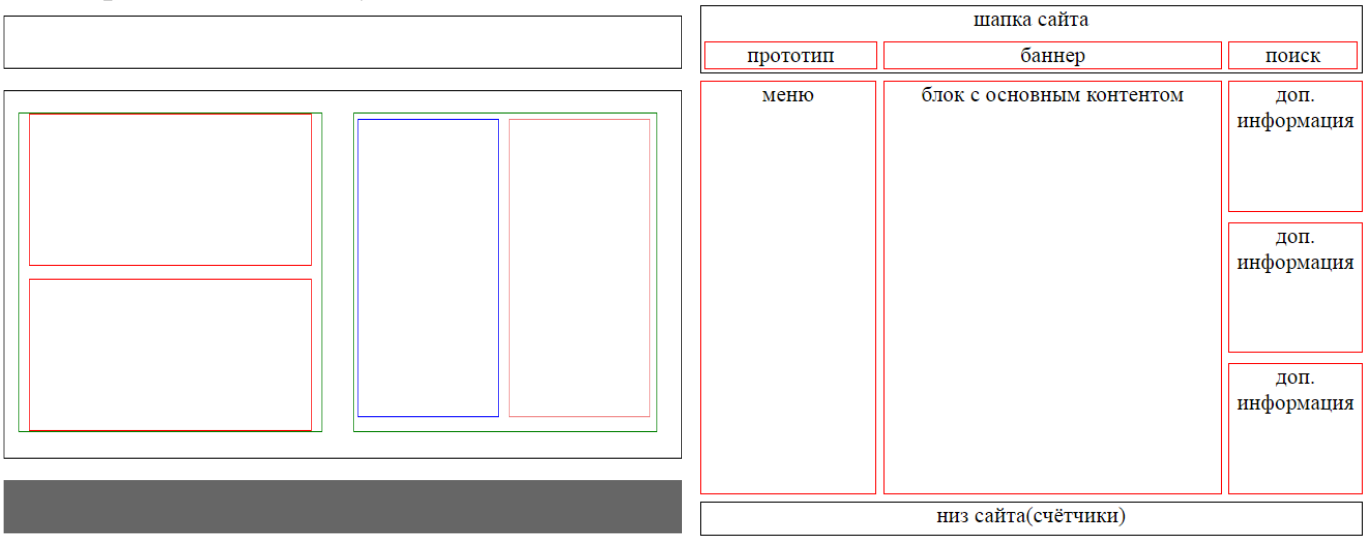
7.Результат не будет удовлетворительным, они расположены друг под другом так как по умолчанию они имеют блочное представление. Чтобы это изменить пропишите им строчно-блочное представление (display:inline-block;) и блоки станут рядом.

8.Создайте внутри первого блока два блока с абсолютным позиционированием. Задайте им высоту в 48% от высоты (около 145px) родительского блока, ширину 400px, один прижмите к верхней, второй – к нижней границе. Оба блока выровняйте по центру.

9.Создайте два блока в пустом контейнере справа, дайте общий класс для общих свойств и уникальный класс: <div class=”our-child one”> и <div class=”our-child two”>

10.Классу our-child присвойте ширину 200px, отступы по 10px, высоту 280px, представление – строчно-блочное. Уникальным классам дайте свои внутренние отступы, свою заливку или фон в виде картинки, задайте свою рамку.

11.Сверстайте две модульные сетки



12.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Опишите все позиционирования элементов на странице, которые знаете?
2. Как задать обрезку границ?
3. Как задать строчно-блочное представление элемента?

Практическая работа №12-13. Верстка простого макета сайта.

Тема: Верстка простого макета сайта.
Цель: Научиться верстать простые веб-страницы.

Ход работы

Теоретическая часть

static – стоит по умолчанию, нет смысла применять к элементу, если не нужно обнулить позиционирование.

relative – относительное позиционирование, можно выровнять блок по центру страницы при помощи свойства margin:0 auto; отступы задаются при помощи margin. Элемент может сдвигаться, но место в потоке под него остается.

absolute – абсолютное позиционирование, в этом случае элемент «выдирается» из потока на странице, ниже идущие элементы поднимаются на его место, а сам элемент прижимается в верхний левый угол страницы. Позиционируются элементы при помощи свойств left, right, top, bottom. Чтобы позиционировать относительно границ нужного контейнера, родителю необходимо задать относительное позиционирование.

fixed – так же, как и абсолютное, только элемент при прокрутке всегда будет находиться на своем месте и никогда не покинет экран.

Горизонтальное центрирование:

Относительное позиционирование – margin(отступ сверху и снизу) auto(выравнивает по центру);

Абсолютное и фиксированное – необходимо сдвинуть элемент на 50% влево и задать отрицательный отступ слева, равный половине ширины элемента;

Можно сделать элемент строчным или строчно-блочным, а к родителю применить выравнивание текста по центру.

Практическая часть

1.Необходимо сверстать полноценную простую веб-страницу по выбранному из предложенных макету, используя все полученные знания.

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Опишите принципы позиционирования элементов на странице?
2. Что такое z-index?
3. Что использовали для центрирования контента на странице?

TOUR OUR LIFE

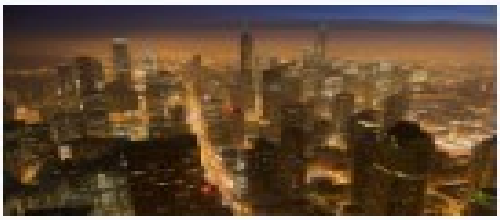
ГЛАВНАЯ | ТУРЫ | БЛОГ | ЗАКАЗАТЬ БИЛЕТ

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Expedita, nemo, repellendus? Rerum nulla distinctio excepturi dolorum suscipit labore vero commodi molestiae, facere nemo quibusdam consectetur tempore soluta, minima perferendis consequuntur.

США

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Recusandae, perferendis minus doloremque autem modi labore quasi ullam velit adipisci quos?



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Corporis tempore, dignissimos officiis eos est, corrupti voluptatibus distinctio beatae, quos voluptates autem molestiae necessitatibus eveniet, expedita maxime! Dolores, officiis officia recusandae?

Просмотреть тур

Blogger

SEARCH

home | about | services | projects | blog | contact

Lorem ipsum dolor sit amet, consectetur adipisc-ing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent aliquet lacus at metus accumsan porta. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin in ligula ut dui adipiscing commodo.

Cras aliquet tellus sed dolor aliquet condimentum. Quisque in ligula orci ... read more

Previous

Next

Nulla tellus tempus magna.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer dictum, neque ut imperdiet pellen- tesque, nulla tellus tempus magna, sed consectetur orci metus a justo. Integer dictum, neque ut imperdiet pellentesque, nulla tellus tempus magna, sed consectetur orci metus a justo. Aliquam ac congue nunc. Mauris a tortor ut massa egestas tempus.
Cras aliquet tellus sed dolor aliquet condimentum. Quisque in ligula orci ... read more

Consectetur elit

Aliquam ac congue nunc. Mauris a tortor ut massa egestas tempus. Pellentesque tincidunt fermentum diam sagittis ullamcorper.
Cras aliquet tellus sed dolor aliquet condimentum ... read more

Практическая работа №14-15. Верстка шапки, контента и подвала сайта.

Тема: Верстка сайта.
Цель: Научиться верстать полноценный сайт.

Ход работы

Теоретическая часть

Шапкой (header) называется фрагмент контента, появляющийся в верхней части сайта. В ней можно расположить: основную систему навигации, вспомогательную систему навигации, поиск, учетную запись и авторизацию.

Навигация/Меню – это вполне самостоятельный компонент. Существует три основные системы навигации на основе меню: вкладки, выпадающее меню, древовидное меню.

Пример написания меню для вашего будущего сайта:

Разметка горизонтального многоуровневого меню базируется на css-позиционировании. Всем элементам списка `li` задается относительное позиционирование, а выпадающему меню `ul` любого уровня — абсолютное позиционирование.

По умолчанию ширина выпадающего меню равна ширине элемента списка, в который оно вложено. Чтобы изменить ширину, нужно её задать при помощи свойства `width`, например, `.submenu {width: 300px;}`.

Выпадающее меню первого уровня вложения не требует задания смещения, абсолютное позиционирование делает его высоту равной нулю, чтобы не оставлять пустое место под элементом списка. Для меню второго уровня задаётся смещение относительно левой и верхней границы элемента списка `ul {left: 100%; top: 0;}`

Выпадающее меню скрывается с помощью `.submenu {visibility: hidden; opacity: 0;}`, показывается `li:hover > .submenu {visibility: visible; opacity: 1;}`.

Можно сместить подменю `.submenu {transform: translateY(10px);}` по горизонтали, скрыть его при помощи `visibility: hidden; opacity: 0;`. При наведении на элемент списка, меню становится видимым и перемещается вверх на 10px.

Нижний колонтитул (footers) обычно относится к одной из двух категорий. Это могут быть как функциональные навигационные инструменты, направляющие пользователя к дополнительному контенту, так и место для той информации, которая обязана присутствовать на сайте, но которую никто не хочет видеть.

Практическая часть

1.Необходимо сверстать полноценный сайт по своему собственному макету, используя все полученные знания.

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Обоснуйте выбор структуры своего сайта?
2. Какие типы сайтов существуют? К какому типу относится ваш сайт?
3. Что такое семантика сайта?

Практическая работа №16-17. Верстка собственного блога.

Тема: Верстка собственного блога.
Цель: Научиться верстать блог.

Ход работы

Теоретическая часть

Блог – это, своего рода, лента новостей. Расскажите в своем блоге о своих увлечениях, достижениях и т.п.

Например:

Практическая часть

1.Необходимо сверстать блог по своему собственному макету, используя все полученные знания (блог должен быть одностраничным сайтом и содержать минимум 6 новостей, 3 статьи о себе и своих увлечениях, 12 фотографий(рисунков), 2-3 колонки на странице).

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Какие особенности позиционирования элементов при верстке блога?
2. Опишите семантику вашего блога.
3. К чему приводит неправильное формирование SEO-тегов?

Практическая работа №18. Адаптация имеющегося сайта под мобильные устройства.

Тема: Адаптация сайта под мобильные устройства.
Цель: Научиться адаптировать сайт под мобильные устройства.

Ход работы

Теоретическая часть

Адаптивная верстка сайта, или mobile-friendly, заключается в выполнении определенных функций, направленных на создание веб-страницы, способной подстроится под любое разрешение экрана. Адаптивную верстку сайта можно осуществить с помощью CSS и HTML5. Адаптивные макеты могут быть следующих типов: резиновый, использующий перемещение блоков или подключение макетов, панели и элементарная адаптивная верстка.

Пример простого способа адаптировать сайт:

```
<meta name='viewport' content='width=device-width,initial-scale=1'/>
<meta content='true' name='HandheldFriendly'/>
<meta content='width' name='MobileOptimized'/>
<meta content='yes' name='apple-mobile-web-app-capable'/>
```

Свойство width определяет размер окна просмотра. Он может быть установлен на определенное количество пикселей, скажем, width=600 или на специальное значение device-width, которое означает ширину экрана в пикселях CSS в масштабе 100%. Задав мета-тегу viewport значение “device-width” – оповещаем браузер, что ширина области просмотра равняется ширине этого устройства, а не стандартной ширине в 980px, как он может предполагать по-умолчанию.

Свойство initial-scale контролирует уровень масштабирования при первой загрузке страницы. Свойства maximum-scale, minimum-scale и user-scalable определяют, как пользователям разрешено увеличивать или уменьшать страницу.

На экранах с высоким разрешением экрана страницы с initial-scale=1 будут эффективно масштабироваться браузерами. Их текст будет плавным и четким, но их растровые изображения, вероятно, не будут использовать полное разрешение экрана.

Meta-тег HandheldFriendly определяет оптимизирована ли страница сайта под мобильные устройства на Palm и Blackberry, в таком браузере как AvantGo.

Meta-тег MobileOptimized задаёт ширину области просмотра в мобильных браузеров IE Mobile или Pocket IE. Является аналогом атрибута width в meta-теге viewport.

Meta-тег apple-mobile-web-app-capable позволяет странице работать в полноэкранном режиме, актуален для мобильных устройств Apple.

Практическая часть

1.Создайте 4 контейнера в одном родительском контейнере. Подвигайте контейнеры вдоль оси X и вдоль оси Y с помощью Flexbox как в лекционном материале.

2.Необходимо адаптировать один из имеющихся у вас сайтов под мобильные устройства, используя все полученные знания.

3.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Перечислите и опишите типы адаптивных макетов.
2. Как дать понять браузеру, что ширина области просмотра равняется ширине устройства?
3. Для чего нужен Flexbox?

Практическая работа №19. Создание скриптов. Взаимодействие с пользователем. Работа с массивами и циклами.

Тема: Создание и подключение скриптов. Основные типы примитивов. Взаимодействие с пользователем. Работа с объектами.
Цель: Научиться создавать простые скрипты.

Ход работы

Теоретическая часть

В JavaScript существует несколько типов примитивов. Основные из них: строки, числа, объекты, специальные значения, логический тип данных. Все они могут быть записаны в переменную. Существует два способа создания переменных и один способ для создания констант. В общем виде это можно записать так:
тип_переменной имя переменной=значение переменной =>
var name=”Tim”;
let age=30;

const BEHUNGREE = true;

Для создания переменных нет необходимости указывать тип данных, в одну и ту же переменную можно записать число, строку, логический тип и т.д.

Массивы создаются аналогичным образом. В JavaScript массивы динамические, то есть нет необходимости указывать размер массива при его создании. Добавить/удалить элементы можно в любой момент.
var array=[];
var arr=[1,2,”2”,”name”,true,false,4];

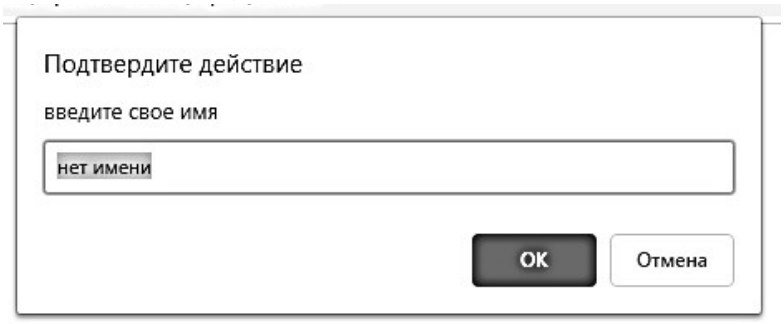
Такие массивы могут содержать любой тип данных и в любой момент могут быть изменены по длине. Перебор массивов осуществляется так же, как и в других языках программирования.

Существует три основных способа сказать что-то пользователю: вывести сообщение, попросить ввести что-то, спросить что-то с вариантами «да» или «нет».

Сообщение выводится командой alert(body), где body – тело сообщения, может быть строчкой, числом, объектом, выражением и т.д.

Запрос ввода чего-либо – prompt(message, variable); – выводит модальное окно, в котором message – тело сообщения, variable – вариант по умолчанию. Например, следующий код попросит ввести пользователя имя:

```
prompt(“введите свое имя”, “нет имени”);
```



При этом, результат запроса можно записать в переменную:

```
prompt("введите свое имя","нет имени");
```

И сразу вывести его:

```
var name=prompt("введите свое имя","нет имени");
alert(name);
```

Модальное окно с вариантами «да» или «нет» выводит текст сообщения и возвращает логическое значение, в зависимости от выбранного пользователем результата.

```
var name=confirm("вам есть 18 лет?")
alert(name);
```

Объекты в джаваскрипт – это что-то вроде ассоциативных массивов, информация в которых хранится в виде ключ – значение. При этом, доступ к значению можно получить по его ключу:

```
var object={name:"Masha",
            age:19,
            student:true};
```

Чтобы вывести имя на экран:

`document.write(object.name)` – выведет Masha

Также можно обратиться к любому другому свойству. Для записи нового свойства:

```
object.second_name=”Новикова”;
```

Для удаления свойства необходимо использовать оператор delete:

```
delete object.name;
```

Для удобного перебора свойств объектов есть свой цикл: `for (key in obj);`

```
var obj={  name:"Masha",
           age:19,
           student:true};
for(key in obj){alert(key+""+obj[key]);};
```

Практическая часть

1.Создайте переменную и присвойте ей какое-то значение. Выведите ее на экран. Создайте константу, в нее запишите свое имя и выведите на экран. Сложите две переменные, результат выведите на экран.

2.Конкатенация строк. Создайте две переменные (хотя бы одной должна быть строка). Выведите на экран сумму, разность, произведение, частное этих переменных.

3.Создайте скрипт, который запросит у пользователя имя, спросит, совершеннолетний он или нет, запишет оба результата в переменные и выведет на экран.

4.Создайте массив из 5 элементов, каждому элементу массива присвойте различное значение (текстовое, численное, специальное). Выведите массив на экран.

5. Создайте скрипт, запрашивающий у пользователя значение и заносящий его в массив. После этого отобразите массив на экране при помощи трех различных циклов.

6.Создайте объект с тремя полями: имя, фамилия, номер группы. Выведите его на экран, обратившись к каждому полю. Удалите поле с группой, замените имя на другое. Выведите его на экран.

7.Создайте объект со следующими свойствами: ваше имя, фамилия, группа, текущий предмет, оценка по предмету. Выведите его на экран при помощи цикла.

8. Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Как подключить сценарии к html-документу?
2. Перечислите циклы в JavaScript. Опишите принцип их работы.
3. Опишите способы взаимодействия с пользователем.

4. Что такое конкатенация строк?

Практическая работа №20. Работа с DOM. Изменение свойств узлов HTML.

Тема: Работа с DOM. Использование условных операторов и операторов множественного выбора.
Цель: Научиться работать с DOM.

Ход работы

Теоретическая часть

Основная задача JavaScript состоит в том, чтобы изменять структуру сайта и его элементов. Для этого JavaScript обладает множеством методов поиска/выбора/удаления/изменения свойств узлов HTML.

Глава всего документа – document. При обращении к нему можно обратиться к любому элементу на странице. Например, необходимо найти кнопку с идентификатором button. Для этого необходимо воспользоваться следующей конструкцией: document.getElementById(‘button’);

Интерпретатор JavaScript обратится к веб-странице, а затем при помощи метода getElementById запросит необходимый элемент по его идентификатору. Этот элемент можно записать переменную: var button= document.getElementById(‘button’);

Теперь переменная button содержит в себе этот элемент и можно его всячески изменять. Например, стиль элемента можно изменить таким образом:

```
var button= document.getElementById(‘button’);
button.style.width=”200px”;
button.style.background=”green”;
button.style.border=”2px solid red”;
```

Также, можно удалить этот элемент: button.remove() или getElementById(‘button’).remove()

Дочерний узел какого-либо элемента. Допустим, есть такой код:

```
<body>
     <div>Начало</div>
     <ul><li>Информация</li></ul>
     <div>Конец</div>
</body>
```

Чтобы получить доступ ко всем дочерним элементам внутри body, можно воспользоваться такой конструкцией:

for(var i=0; i<document.body.childNodes.length; i++){alert(document.body.childNodes[i]);}

То есть через document обращаемся к body и при помощи childNodes получаем псевдомассив дочерних элементов (именно псевдомассив, так как возвращаемые DOM-коллекции, такие как childNodes и другие, не являются JavaScript-массивами. В них нет методов массивов, таких как forEach, map, push, pop и других, но они являются итерируемыми (их можно перебрать при помощи циклов)).

Для того, чтобы получить не дочерний, а родительский элемент тоже есть специальный метод – parrentNode, который получает родителя элемента. Например, для элемента списка <li> родителем будет сам список, то есть

```
<body>
     <ul> - родительский элемент
          <li>Информация</li> - дочерний элемент
     </ul>
</body>
```

Один из способов получения родительского элемента:

document.getElementByTagName(”li”).parrentNode

Но такие методы – только для чтения, то есть нельзя, получив элемент при помощи

parrentNode или childNodes удалить или изменить эти элементы. Для этого необходимо напрямую воспользоваться методом document.getElementBy*, где * - способ получения элемента (по идентификатору, классу, тегу) Список этих методов:
document.getElementByTagName
document.getElementById
document.getElementByClassName и др.
Для получения нескольких элементов с одинаковым классом/названием тега можно воспользоваться методами:
document.getElementsByClassName или document.getElementsByTagName, которые вернут коллекции элементов, подходящих данному условию.
Также есть более универсальный способ получения элементов, сочетающий в себе сразу все вышеупомянутые методы. Используя, document.querySelector можно получить любой элемент, удовлетворяющий селектору CSS, либо его правилу. Например, необходимо получить элемент с идентификатором id или тегом <ul>:
document.querySelector(“#id”) или document.querySelector(“ul”)
Вызов elem.querySelector(css) возвращает не все, а только первый элемент, соответствующий CSS-селектору. Для того, чтобы получить все элементы на странице, соответствующие данному селектору, необходимо использовать метод document.querySelectorAll. Он вернет коллекцию из всех элементов, которые нужны.
Также можно использовать сложные конструкции: document.querySelectorAll(‘ul>li:last-child’)

Практическая часть

1.Создайте html документе список на странице. Соберите все его элементы в коллекцию, а потом каждый второй элемент залейте красным, каждый третий – зеленым. Удалите первый и последний элементы
2.Используя ранее полученные знания, напишите скрипт, который запросит ваше имя и имя соседа по парте, фамилию, возраст, оценку по веб-программированию и поместит эти значения в соответствующие поля таблицы.
3.Создайте html список из 15 элементов. Напишите скрипт, запрашивающий значение (число, номер элемента списка) и удаляющий элемент списка, равный введенному числу.
4.Создайте скрипт, запрашивающий ваше имя и возраст человека, формирует из него массив и выводит в консоль браузера. Перебор можно сделать при помощи любого цикла.
5. Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Как обратиться к кнопке с идентификатором kn?
2. С помощью чего можно получить псевдомассив дочерних элементов?
3. Назовите универсальный способ получения элементов.
4. Как удалить определенный элемент из массива?

Практическая работа №21. Введение в события, изменение свойств элементов через события.

Тема: Работа с функциями, динамическое создание узлов DOM, работа с сетевыми запросами.
Цель: Научиться создавать DOM-элементы, программно обращаться к серверу.

Ход работы

Теоретическая часть

Создание элементов DOM (document object model) происходит при помощи метода createElement(tagName) объекта document, где tagName – строковый параметр, обозначающий имя тега. Например, var p – document.createElement('p'); Это создаст пустой абзац, который хранится в памяти.

Для того, чтобы вставить его на страницу, существует множество методов. Самый распространенный – parrentElement.appendChild(elem), где parrentElement – родительский элемент, а elem – только что созданный объект, находящийся в памяти. Данная команда вставляет новый элемент после всех предыдущих DOM узлов и элементов.

Для указания атрибутов существует команда: elem.setAttribute(atrName, value), где atrName – строка, передающая имя атрибута, а value – значение атрибута. Например, что бы создать картинку с атрибутом src, указывающий на какое-либо изображение, а потом вставить его в самый конец потомков тега body:

```
let img = document.createElement("img")
img.setAttribute('src','img/avatar.jpg');
document.body.appendChild(img);
```

Для записи строки или числа в элемент используется метод elem.innerHTML=string/number; Данный метод запишет в тег строку или число. Например, создание заголовка 4 уровня и запись в него текущего дня недели:

```
let h4 = document.createElement("h4")
h4.innerHTML="понедельник";
document.body.appendChild(h4);
```

Подобным образом можно вкладывать элементы друг в друга, например, получим тег div, который будет находится в теге body и содержать в себе теги h4, p:

```
let div=document.createElement('div');
let h4=document.createElement('h4');
div.appendChild(h4);
let p = document.createElement("p");
p.innerHTML="Понедельник – первый день недели";
div.appendChild(p);
document.body.appendChild(div);
```

Для создания функций используется ключевое слово function. Существует как минимум два способа создать функцию: дать ей имя, или создать анонимную функцию, записав ее в переменную.

```
function seyHello(){ alert('hello');}
var seyHello=function(){alert('hello');}
```

Разница в подходах состоит в том, что функции, объявленные вторым способом нельзя вызвать до того, как они будут созданы.

Для реакции на действие посетителя и внутреннего взаимодействия скриптов существуют события.

События – это сигнал от браузера о том, что что-то произошло. Существует много видов событий. Часто используемые:

События мыши:
click – происходит, когда кликнули на элемент левой кнопкой мыши.
mouseover – возникает, когда на элемент наводится мышь.
mousedown/mouseup – когда кнопку мыши нажали и отпустили.
mousemove – при движении мыши.
События документа:
DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

Существует множество способов назначить обработчик события на элемент:
Добавлением тега соответствующего атрибута:

<input value=”Нажми меня” onclick=”alert(‘Клик!’)” type=”button”>

Использованием свойства DOM-объекта:

<input id=”elem” type=”button” value=”Нажми меня”/>
<script> elem.onclick=function(){alert(‘Спасибо’);};</script>

При помощи метода addEventListener(event,callback), где в качестве события передается строка с его названием в качестве callback – функция, которая будет вызвана после активации события. Этот способ является наиболее рекомендуемым к использованию, так как с его помощью можно навесить несколько различных обработчиков на одно и то же событие к одному элементу, в то время, как предыдущие способы просто перезаписали бы уже имеющиеся обработчики. Например:

```
function handler(e){alert(e);}
input.addEventListener(“click”,handler);
```

То есть назначается обработчик события клика на определенный элемент, а потом удаляется. Когда пользователь нажмет на данный элемент, вызовется функция, переданная в качестве callback. В качестве аргумента ей передастся объект события, записанный в переменную e. Из этого объекта можно узнать тип события (e.type), на каком именно элементе произошло событие (e.target), координаты мыши и так далее.

Данные в формате JSON представляют собой JavaScript-объекты {…}, или массивы […], или значения одного из типов: строки в двойных кавычках, число, логическое значение true/false, null.
JSON.parse – читает объекты из строки в формате JSON.
JSON.stringify – превращает объекты в строку в формате JSON, используется, когда нужно из JavaScript передать данные по сети.
Вызов JSON.parse(str) превратит строку с данными в формате JSON в JavaScript объект/массив/значение. Например:

```
var numbers=”[0,1,2,3]”;
numbers= JSON.parse(numbers);
alert(numbers[1]);//1
```

Или:

```
var user=’{“name”:“Vasya”, “age”:35, “isAdmin”:false, “friends”:[0,1,2,3]}’;
user= JSON.parse(user);
alert(user.friends[1]);//1
```

Метод JSON.stringify(value,replacer,space) преобразует («сериализует») значение JSON в строку. Например:

```
var event= {title:”Конференция”
          date:”Сегодня”}
var str=JSON.stringify(event);
alert(str);// {“title”:”Конференция”,”date”:”сегодня”}
```

Запросы к серверу:

```
var xhr=new XMLHttpRequest(); //Создается новый объект XMLHttpRequest
xhr.open('GET','phones.json',false);  //Конфигурация GET-запрос на URL'phones.json'
xhr.onreadystatechange=function(e){if(e.readyState !=4)return};
//по окончании запроса доступны: status, statusText
if(e.status !=200){  //обработать ошибку
alert('ошибка:'+(e.status ? e.statusText:'запрос не удался'));
return;} //получить результат из e.responseText
```

На первой строке создается объект запроса. Здесь происходит его инициализация. На второй строке указывается тип запроса (GET), адрес запроса, синхронный или асинхронный запрос (true – асинхронный, false – синхронный, рекомендуется true).

Далее подписка на событие запроса onreadystatechange, которое может возникать несколько раз, в ходе которого можно проверить текущее состояние запроса (status), и обработать запрос в соответствии с его состоянием. Далее можно получить результат при помощи responseText, который затем обрабатывается. Его также можно передать в какую-либо функцию.

Рассмотрим работу с запросами на примере кода получения списка друзей ВКонтакте. У ВКонтакте есть свой сервер и свое API, обращаясь к которому можно получить те или иные данные. Все запросы к ВКонтакте строятся по следующей схеме:

Синтаксис запроса:

Чтобы обратиться к методу API ВКонтакте, необходимо выполнить POST или GET запрос такого вида:

https://api.vk.com/method/METHOD_NAME?
PARAMETERS&access_token=ACCESS_TOKEN&v=V

Он состоит из нескольких частей:

METHOD_NAME (обязательно) – название метода API, к которому нужно обратиться. Полный список методов доступен на официальной странице для разработчиков vk.developers

PARAMETERS (опционально) – входные параметры соответствующего метода API, последовательность пар name=value, разделенных амперсандом. Список параметров указан на странице с описанием метода.

ACCESS_TOKEN (опционально) – ключ доступа.

V (опционально) – используемая версия API. Использование этого параметра применяет некоторые изменения в формате ответа различных методов. Этот параметр следует передавать со всеми запросами.

Для некоторых методов можно не передавать access_token, так как они могут быть доступны для всех. Рассмотрим пример составления запроса на получение информации о пользователе:

```
var xhr=new XMLHttpRequest();
var url=' https://api.vk.com/method/users.get?user_ids=210700286&fields=bdate&v=5.60' //Конфигурация его – GET-запрос на url-адрес содержащийся в переменной
xhr.open('GET',url);
xhr.onreadystatechange=function(e){ if (e.readyState !=4) return;
                                    if (e.status !=200){//обработать ошибку
```

(‘ошибка:’+(e.status?e.statusText:’запрос не удался’));
return;}//получить результат из e.responseText и передать его функцию f(), определенную где-то в коде f(e.responseText);
В ответ придет такой JSON:
{“response”:[{
“id”:210700286,
“first_name”:”Lindsey”,
“last_name”:”Stirling”,
“bdate”:”21.9.1996”}]}
Если мы его преобразуем при помощи метода JSON.parse, то получим объект, содержащий в себе свойство response, в которой записан массив, содержащий всего один объект с полями first_name, last_name, bdate.

Практическая часть

1.Получите список друзей, преобразуйте его в объект, свойства которых запишите в DOM элементы, созданные динамически из скрипта. В итоге должен получится список, состоящий из элементов ul, в которых будет картинка, имя и фамилия человека. Оформите это при помощи стилей и сделайте похожим на страницу друзей в ВКонтакте. Сделайте это при помощи нескольких функций: одна будет отправлять запрос, вторая – принимать ответ и строить DOM с полученными данными.

3.Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Опишите способы создания функций.
2. Что такое событие? Назовите виды.
3. Что такое callback функции?
4. Что такое ACCESS_TOKEN и V?

Практическая работа №22. Обработка ошибок, исключений, проброс исключений.

Тема: Обработка ошибок, исключений, проброс исключений.
Цель: Научиться обрабатывать логические ошибки в коде.

Ход работы

Теоретическая часть

Конструкция перехвата ошибок try…catch:

```
try { //код…
} catch (err) { // обработка ошибки}
```

Работает эта конструкция таким образом:

1. Выполняется код внутри блока try.
2. Если в нём не встречаются ошибки, то блок catch(err) игнорируется.
3. Если в нём возникнет ошибка, то выполнение try будет прервано на ошибке, и управление передается в начало блока catch(err). При этом переменная err (можно выбрать любое другое название) должна содержать объект ошибки с подробнейшей информацией о произошедшей ошибке.

Поэтому при ошибке в try скрипт не останавливается, и мы имеем возможность обработать ошибку внутри блока catch.

В случае, если грубо нарушена структура кода, не закрыта фигурная скобка или где-то стоит лишняя запятая, то try..catch не поможет. Это синтаксические ошибки, интерпретатор такой код не понимает. JavaScript-движок сначала читает код, а затем исполняет его. Ошибки, которые возникают во время фазы чтения, называются ошибками парсинга. Их нельзя обработать (изнутри этого кода), потому что движок не понимает код. Таким образом, try..catch может обрабатывать только ошибки, которые возникают в корректном коде. Такие ошибки называют «ошибками во время выполнения», «исключениями».

Блок catch должен обрабатывать только те ошибки, которые ему известны, и «пробрасывать» все остальные.

Техника «проброс исключения» выглядит так:

Блок catch получает все ошибки.

В блоке catch(err) {...} анализируется объект ошибки err.

Если не знает как её обработать, тогда делает throw err.

Конструкция try..catch может содержать ещё одну секцию: finally.

Если секция есть, то она выполняется в любом случае:

после try, если не было ошибок, после catch, если ошибки были.

```
try { .. пробуем выполнить код .. }
catch(e) { .. перехватываем исключение .. }
finally { .. выполняем всегда .. }
```

Практическая часть

1. Написать код, который будет содержать секции try, catch(e), finally. В коде необходимо обработать, одну известную ошибку, а остальные – осуществить «проброс исключения».

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Какие ошибки обрабатывает конструкция try…catch?
2. Как работает конструкция try…catch?

3. Что такое «проброс исключения»?
4. Зачем нужна секция finally?

Практическая работа №23. Формы HTML5. Работа с формами с помощью JavaScript.

Тема: Формы HTML5. Работа с формами с помощью JavaScript.
Цель: Научиться работать с формами HTML5 с помощью JavaScript.

Ход работы

Теоретическая часть

Если в HTML-документе определена форма, то она доступна сценарию JavaScript как объект, входящий в объект document с именем, заданным атрибутом NAME тега FORM.

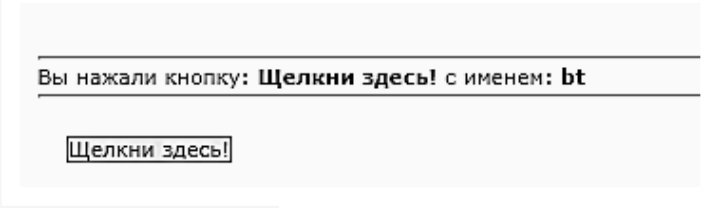
Элементы форм:

1. Кнопки (BUTTON, RESET, SUBMIT)

Свойства: name – имя объекта, value – надпись на кнопке.

Метод:click( ) – вызов этого метода тождественен щелчку мышкой по кнопке, например:

```
    <script>
        <!--
        function btnClick()
        {
            var Txt1 = "";
            var Txt2 = "";
            Txt1 = document.Test.bt.value;
            Txt2 = document.Test.bt.name;
            document.getElementById('ex1').innerHTML="<HR>"+
                "Вы нажали кнопку: " + Txt1.bold() +
                " с именем: "  + Txt2.bold() +"<HR>";
        }
         //-->
    </script>
</head>
<body>
<H1>Нажатие кнопки</H1>
<div id="ex1"></div>
<FORM NAME="Test">
    <INPUT TYPE="button" NAME="bt" VALUE="Щелкни здесь!"
        onClick="btnClick();">
</FORM>
```

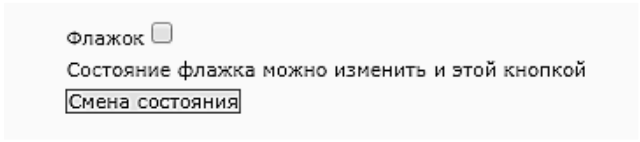


2. Флажки (CHECKBOX)

Свойства: name – имя объекта, value – надпись на кнопке, checked – состояние флажка: true - флажок установлен, false - флажок не установлен, defaultChecked – отражает наличие атрибута CHECKED: true - есть, false - нет.

Метод:click( ). Вызов этого метода меняет состояние флажка, например:

```
<H1>Метод click флажка</H1>
<FORM NAME="Test">
    флажок <INPUT TYPE="checkbox" NAME="ch">
    <BR>Состояние флажка можно изменить и этой кнопкой
    <INPUT TYPE="button" VALUE="Смена состояния"
           onClick="document.Test.ch.click();">
</FORM>
```



3. Переключатель (RADIO)

Свойства: name – имя объекта, value – надпись на кнопке, length – количество переключателей в группе, checked – состояние переключателя: true - переключатель включен, false – выключен, defaultChecked – отражает наличие атрибута CHECKED: true - есть, false - нет.

Метод:click( ). Вызов этого метода включает переключатель. Так как группа переключателей имеет одно имя NAME, то к переключателям надо адресоваться как к элементам массива. Например:

```
    <script><!--
        function btnClick()
        {
            if(document.Test1.Sex[0].checked){
                document.Test1.Sex[1].click();
            }else{
                document.Test1.Sex[0].click();
            }
        }
         //-->
    </script>
</head>
<body>
<H1>Метод click группы переключателей</H1>
<FORM NAME="Test1">
    <INPUT TYPE="RADIO"  NAME="Sex" VALUE ="Man" CHECKED>Мужской
    <INPUT TYPE="RADIO"  NAME="Sex" VALUE ="Woman">Женский
    <BR>Состояние переключателей можно изменить и этой кнопкой
    <INPUT TYPE="button" VALUE="Смена состояния" onClick="btnClick();">
</FORM>
```

4. Список (SELECT)

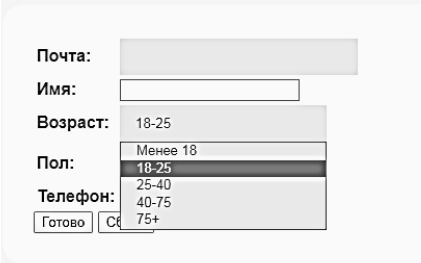
Свойства: name – имя объекта, selectedIndex – номер выбранного элемента или первого среди выбранных (если указан атрибут MULTIPLE), length – количество элементов (строк) в списке, options – массив элементов списка, заданных тегами OPTION.

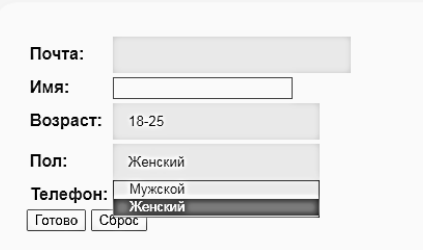
Каждый элемент массива options является объектом со следующими свойствами: value – значение атрибута VALUE, text – текст, указанный после тега OPTION, index – индекс элемента списка, selected – присвоив этому свойству значение true, можно выбрать данный элемент, defaultSelected – отражает наличие атрибута SELECTED: true - есть, false - нет.

Методы: focus( ) – передает списку фокус ввода, blur( ) - отбирает у списка фокус ввода.

Пример:

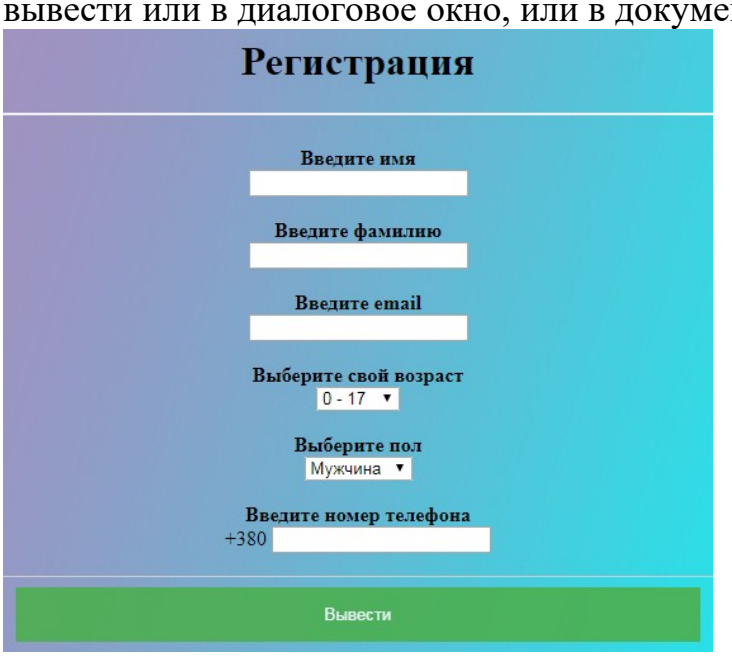
```
    <FORM NAME="Sel1">
        <!-- Анкета -->
        <TABLE>
            <TR><TD><B>Почта:<B></TD>
                <TD><INPUT NAME="Emeil" type="email"  ></TD></TR>
            <TR><TD><B>Имя:<B></TD>
                <TD><INPUT NAME="Name" ></TD></TR>
            <TR><TD><B>Возраст:<B></TD>
                <TD> <SELECT NAME="Age" SIZE=1>
                    <OPTION VALUE="Менее 18" SELECTED>Менее 18
                    <OPTION VALUE ="18-25">18-25
                    <OPTION VALUE ="25-40">25-40
                    <OPTION VALUE ="40-75">40-75
                    <OPTION VALUE ="75+">75+
                </SELECT></TD></TR>
                <TR><TD><B>Пол:<B></TD>
                    <TD> <SELECT NAME="Pol" SIZE=1>
                        <OPTION VALUE="Мужской" SELECTED>Мужской
                        <OPTION VALUE ="Женский">Женский
                    </SELECT></TD></TR>
            <TR><TD><B>Телефон:<B></TD>
                <TD><INPUT NAME="Phone" SIZE=10></TD></TR>
        </TABLE>
    <INPUT TYPE="button" VALUE="Готово" onClick="Complete();">
    <INPUT TYPE="reset" VALUE="Сброс">
</div>
<script>

function Complete()
    {
        var Elem="Почта: " + document.Sel1.Emeil.value +
            "\nИмя: " + document.Sel1.Name.value +
            "\nВозраст "+document.Sel1.Age.value +
            "\nПол: " + document.Sel1.Pol.value+
            "\nТелефон: " + document.Sel1.Phone.value;
        alert(Elem);
        console.log(Elem);
    }

</script>
```





Практическая часть

1. Реализовать форму в виде анкеты. Поля возраст и пол, должны быть реализованы в виде выпадающего списка. Информацию, введенную пользователем необходимо вывести или в диалоговое окно, или в документ, или в консоль. Например:



2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Назовите свойства переключателя (RADIO).
2. За что отвечает свойство length?

3. Перечислите методы списка (SELECT).

Практическая работа №24. Использование fullPage.js.

Тема: Использование fullPage.js..
Цель: Научиться использовать fullPage.js.

Ход работы

Теоретическая часть

fullPage.js – это jquery плагин, который позволяет создать постраничный скроллинг.
Перед началом работы, необходимо загрузить следующие библиотеки:

- ✓ jQuery (≥1.6.0)
- ✓ файл fullPage.css jquery.fullPage.css
- ✓ файл j fullPage.js query.fullPage.js

Файлы javascript подключаютя перед закрывающим тегом:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jquery.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/fullPage.js/2.6.6/jquery.fullPage.min.js"></script>
```

Создание полноэкранных секций:

Сначала необходимо выделить секции, обернув их в тег section. Позже каждой секции будет присвоен уникальный id. По умолчанию, плагин считает активной первую секцию. Чтобы сделать активной другую секцию, добавьте к ней класс action.

Пример HTML структуры:

```
<div id="fullPage">
    <section class="vertical-scrolling">
        <h2>fullPage.js</h2>
        <h3>Первая секция</h3>
        <div class="scroll-icon">
            <p>Перейти к последнему слайду</p>
            <a href="#fifthSection/1" class="icon-up-open-big"></a>
        </div>
    </section>
    <section class="vertical-scrolling">
        <!-- контент здесь -->
    </section>
    <section class="vertical-scrolling">
        <!-- контент здесь -->
    </section>
 </div>
```

Создание горизонтальных слайдов:

Вертикально расположенные секции можно скроллить горизонтально. Применим для секций класс horizontal-scrolling:

```
<section class="vertical-scrolling">
    <div class="horizontal-scrolling">
        <h2>fullPage.js</h2>
        <h3>Это пятый раздел и содержит первый слайд</h3>
    </div>
    <div class="horizontal-scrolling">
        <h2>fullPage.js</h2>
        <h3>Это второй слайд</h3>
        <p class="end">Спасибо!</p>
    </div>
</section>
```

Настройка навигации

Плагин предлагает множество встроенных опций для перемещения по разделам и слайдам. Некоторые из этих опций изначально включены. Например, если необходимо добавить в проект дополнительную навигацию в виде точек, скрыть правую и левые стрелки, то после включения точек навигации, можно изменить их внешний вид путем перезаписи стилей по умолчанию. Вот новые стили:

```
#fp-nav ul li a span,
.fp-slidesNav ul li a span {
    background: white;
    width: 8px;
    height: 8px;
    margin: -4px 0 0 -4px;}
#fp-nav ul li a.active span,
.fp-slidesNav ul li a.active span,
#fp-nav ul li:hover a.active span,
.fp-slidesNav ul li:hover a.active span {
    width: 16px;
    height: 16px;
    margin: -8px 0 0 -8px;
    background: transparent;
    box-sizing: border-box;
    border: 1px solid #24221F;}
```

Практическая часть

1. Добавьте fullPage.js к любому уже сверстанному вами сайту.
2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Что такое fullPage.js?
2. Какая особенность есть у fullPage.js?
3. Что нужно сделать для того чтобы активной была 4 секция?

Практическая работа №25. Анимация элементов на веб-странице.

Тема: Анимация элементов на веб-странице.
Цель: Научиться анимировать элементы на странице.

Ход работы

Теоретическая часть

CSS-анимация - это возможность сделать страницу интерактивной и добавить ей определенной привлекательности.

Hover-эффекты с помощью свойства transition:

Чаще всего свойство transition применяется для создания hover-эффектов, или эффектов при наведении курсора мыши. Его суть заключается в том, что оно просчитывает изменения между свойствами, заданными для обычного состояния элемента и при наведении на него курсора мыши, которое задается с помощью псевдокласса :hover.

Свойство transition является составным (обобщенным, или универсальным) и состоит из ряда свойств, которые можно задавать по отдельности. Синтаксис:

```
transition: transition-property transition-duration transition-timing-function transition-delay;
```

Свойство transition-property задает название css-свойства для анимации перехода.

Свойство transition-duration задает промежуток времени за который произойдет изменение элемента, измеряется либо в секундах (1s, 1.5s, 0.8s, .5s) или в миллисекундах (1000ms, 1500ms, 800ms, 500ms).

Свойство transition-timing-function задаёт временную функцию, которая описывает способ расчета скорости перехода свойств(а) html-элемента от одного значения к другому. По умолчанию свойство имеет значение ease, то есть анимация происходит с некоторым замедлением. Варианты свойства:

```
transition-timing-function: ease| ease-in| ease-out| ease-in-out| linear | cubic-bezier |step-start| step-end | steps
```

Свойство transition-delay позволяет указать задержку в секундах или миллисекундах, после которой будет запущена анимация.

Практическая часть

1. Используя полученные знания, создайте на веб-странице анимированные элементы любой сложности, используя все свойства transition.

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Какие значения может принимать свойство transition?
2. Опишите свойство transition-property.
3. Опишите свойство transition-duration.
4. Опишите свойство transition-timing-function.
5. Опишите свойство transition-delay.

Практическая работа №26. Применение фильтров к изображениям.

Тема: Применение фильтров к изображениям.
Цель: Научиться применять различные фильтры к изображениям с помощью CSS.

Ход работы

Теоретическая часть

Filter – это свойство в CSS3, которое может видоизменять картинки. Накладывать размытость, увеличивать контраст и яркость, добавлять тень, менять цвета и многое другое. Всего у Filter есть 10 значений:

Фильтр размытие – blur: указывается в пикселях, чем больше значение, тем больше проявляется размытость картинки.

Фильтр яркость — brightness: регулируется цвет между черным и оригинальным цветом по мере добавления параметров, от 0% и более (при 0% изображение будет черным, при 100% - оригинальным, а при 200% - станет ярче в два раза). Значения задаются тремя способами: целые числа, дробные числа и проценты.

Фильтр контрастность — contrast: повышает контраст картинки, регулируя разницу между светлыми и темным тонами изображения. Значения задаются тремя способами: целые числа, дробные числа и проценты. (100% - значение по умолчанию, 0% - черное, все, что больше 100%, добавляет контраст).

Фильтр насыщенность — saturate: делает изображения более яркими и насыщенными. Значения: целые и дробные числа, проценты.

Фильтр прозрачность — opacity: устанавливает прозрачность, значения: целые и дробные числа: от 0 до 1, проценты: от 0% до 100.

Фильтр Инверсия – invert: позволяет "переворачивать" цвета, значения: целые и дробные числа: от 0 до 1, проценты: от 0% до 100.

Фильтр сепия — sepia: позволяет изменить цвет, добавив оттенок сепия, значения: целые и дробные числа: от 0 до 1, проценты: от 0% до 100.

Фильтр оттенки серого — grayscale: позволяет превращать цвета в оттенки серого, значения: целые и дробные числа: от 0 до 1, проценты: от 0% до 100.

Фильтр оттенок освещения — hue-rotate: позволяет менять цвет исходного изображения, изменяя угол освещения.

Фильтр тень - drop-shadow: задается несколькими значениями – по оси X, по оси Y (смещение тени по оси X и Y), далее указывается радиус тени и ее цвет.

Практическая часть

1. Выберите любое изображение и примените к нему все изученные фильтры.
2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Что делает фильтр saturate? Как задается?
2. Какая особенность у фильтров opacity, invert, sepia, grayscale?
3. Зачем нужен фильтр hue-rotate?
4. Какие значения и в какой последовательности задаются для фильтра drop-shadow?

Практическая работа №27. Использование временной анимации и правила @keyframes.

Тема: Использование временной анимации и правила @keyframes.
Цель: Научиться анимировать элементы с помощью правила @keyframes.

Ход работы

Теоретическая часть

Ключевые кадры используются для указания значений свойств анимации в различных точках анимации. Они определяют поведение одного цикла анимации; анимация может повторяться ноль или более раз.

Ключевые кадры указываются с помощью правила @keyframes, определяемого следующим образом:

@keyframes имя анимации { список правил }

Создание анимации начинается с установки ключевых кадров правила @keyframes. Кадры определяют, какие свойства на каком шаге будут анимированы. Каждый кадр может включать один или более блоков объявления из одного или более пар свойств и значений. Правило @keyframes содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

Ключевые кадры создаются с помощью ключевых слов from и to (эквивалентны значениям 0% и 100%) или с помощью процентных пунктов, которых можно задавать сколько угодно. Также можно комбинировать ключевые слова и процентные пункты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление.

Например:

```
@keyframes bounce {
  from { top: 100px;    animation-timing-function: ease-out;  }
  25% {top: 50px;    animation-timing-function: ease-in;  }
  50% {    top: 100px;    animation-timing-function: ease-out;  }
  75% {    top: 75px;    animation-timing-function: ease-in;  }
  to {    top: 100px;  }}
```

Практическая часть

1. Создайте анимацию на веб-странице при помощи правила @keyframes (анимация должна включать в себя не менее 10 ключевых кадров).

2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Что такое ключевые кадры?
2. Чем можно заменить ключевые слова?
3. Какой будет последний кадр в процентном эквиваленте?

Практическая работа №28. Установка виртуального сервера. Установка сайта на виртуальный сервер, его настройка.

Тема: Установка сайта на виртуальный сервер, его настройка.
Цель: Научиться устанавливать свой сайт на виртуальный сервер.

Ход работы

Теоретическая часть

Сервер – это физический компьютер, который работает без перерывов, чтобы ваш сайт был доступен всё время для тех, кто хочет его посетить.

Open Server — это портативный программный комплекс, созданный для того чтобы помочь веб-мастерам в разработке, отладке и тестировании сайтов непосредственно на компьютере, локально.

Практическая часть

1. Установите Open Server.
2. Запустите Open Server, в открывшемся окне выберете язык интерфейса сервера, далее сервер попросит Вас установить библиотеки Microsoft Visual C++ , Runtime и пач , установите их.
3. После установки всех необходимых библиотек в трее появится значок Open Server в виде красного флажка, нажмите на него и выберите «запустить».
4. Сервер должен успешно запуститься и красный флажок в трее станет зеленым.
5. Зайдите в папку с установленным Open Server, там находится 3 папки: userdata, modules, domains. В папку domains скопируйте папку с вашим готовым сайтом, и дайте ей название – адрес сайта, например, sayt.com.
6. В трее, нажмите на значок Open Server и выберите «настройки» => «домены». В папке домена укажите свой сайт и нажмите «добавить», сохраните изменения, перезапустите сервер.
7. В трее, нажмите на значок Open Server и выберите «Мои сайты» — откройте свой сайт в браузере, кликнув на название со своим сайтом.
8. Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Для чего нужен локальный сервер?
2. В чем отличие между локальным сервером и хостингом?
3. Назовите типы хостинга.

Практическая работа №29. Установка сайта на WordPress. Управление контентом. Изучение админпанели.

Тема: Установка сайта на WordPress.
Цель: Научиться устанавливать сайт на WordPress.

Ход работы

Теоретическая часть

CMS WordPress — система управления содержимым сайта с открытым исходным кодом; написана на PHP; сервер базы данных — MySQL; сфера применения — от блогов до достаточно сложных новостных ресурсов. Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать проекты широкой функциональной сложности.

С помощью WordPress можно добавлять страницы, записи, менять внешний вид сайта, добавлять все возможные материалы (фото, видео, аудио и прочее…).

С WordPress, можо создавать сайты любого уровня сложности. Делать это очень легко, так как система с подсказками позволяет легко ориентироваться и интуитивно понимать следующий шаг, не будучи профессионалом.

Практическая часть

1. Зайдите на сайт wordpress.com, создайте учетную запись.
2. Выберите тип сайта – блог.
3. Выберите тему вашего блога.
4. Дайте название своему сайту.
5. Выберите адрес сайта, бесплатный тарифный план.
6. Добавьте медиафайлы, стилизуйте ваш сайт на свой выбор.
7. Составьте отчет о проделанной работе и ответьте на контрольные вопросы.

Контрольные вопросы:

1. Что такое CMS WordPress?
2. Приведите примеры подобных систем.

Практическая работа №30. Добавление плагинов к сайту.

Тема: Добавление плагинов к сайту.
Цель: Научиться добавлять плагины к сайту.

Ход работы

Теоретическая часть

Плагин — это дополнение (расширение возможностей) для какой-либо программы на компьютере или движка сайта в интернете. Разработчикам трудно предусмотреть все пожелания пользователей, поэтому они дают возможность сторонним разработчикам удовлетворять эти пожелания при помощи написания плагинов. Кроме того, если все возможные вещи предусмотреть в одном приложении, то оно станет тяжелым и будет работать медленно, а благодаря плагинам этого не происходит, ибо каждый пользователь получает базовый функционал, а все остальное он сможет получить, скачав или установив нужное ему расширение.

Плагин WordPress — с технической стороны, как и тема для WordPress, состоит из набора файлов . php, внутри которых содержится программный код, добавляющий новые возможности.

Практическая часть

1. Установите любой плагин на ваш сайт на WordPress.
2.Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Что такое плагин?
2. Целесообразно ли использовать плагины?

Практическая работа №31. Установка сайта на Joomla. Управление контентом. Изучение админпанели.

Тема: Установка сайта на Joomla.
Цель: Научиться устанавливать сайт на Joomla.

Ход работы

Теоретическая часть

Joomla представляет собой бесплатную систему для создания веб-сайтов. Это проект с открытым исходным кодом, который позволяет создавать сайты любого уровня сложности, как и WordPress.

Практическая часть

1. Реализуйте свой сайт используя Joomla.
2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Какие принципиальные различия между Joomla и WordPress?
2. Что бы вы выбрали Joomla или WordPress? Обоснуйте выбор.

Практическая работа №32. Добавление плагинов к сайту. Создание фотогалереи.

Тема: Создание фотогалереи.
Цель: Научиться создавать фотогалерею.

Ход работы

Теоретическая часть

Фотогалерея для сайта – это:

✓ Возможность продемонстрировать все изображения (фото) сразу – иногда пользователям лень нажимать кнопки для перелистывания. А в классической фотогалерее все объекты сразу доступны для просмотра;

✓ Простота реализации – стандартный образец можно легко создать с помощью html;

✓ Открытость – фотогалерея (по сравнению с альбомом) обладает большей «открытостью», что на подсознательном уровне позволяет вызвать доверие со стороны пользователей.

Некоторые способы создания фотогалереи:

1. Классическая фотогалерея – простая фотогалерея для сайта, представляет собой набор ссылок, к каждой из которых с помощью тега <img> привязана фотография.
2. Фотогалерея на CSS – реализована таким способом, что нажатие на превью фотографии приведет к увеличению масштаба. Кроме этого фотогалерея подстраивается под размеры окна браузера, пропорционально изменяя длину и ширину превью.
3. Фотогалерея на основе Jquery – обладает многофункциональностью и приятным внешним видом.

Практическая часть

1. Используя полученные знания, создайте сайт с фотогалереей.
2. Составьте отчет о проделанной работе и ответьте на контрольные вопросы

Контрольные вопросы:

1. Для чего можно использовать фотогалерею на сайте? Для сайтов с какой тематикой это необходимо?
2. Какой способ создания фотогалереи на сайте лучше, что бы не «утяжелять» сайт?