Ташкентский государственный транспортный
университет

В.Г. Бабина, Ж.Н.Гулямов

МУЛЬТИМЕДИА СИСТЕМЫ И ТЕХНОЛОГИИ

Методическое пособие к выполнению лабораторных работ по дисциплине «Мультимедиа системы и технологии» для преподавателей и студентов 2,3-курса бакалавриата направления образования 5330200 – Информатика и информационные технологии (на железнодорожном транспорте)

(2-часть)

Ташкент – 2021

УДК 681.3

Мультимедиа системы и технологии. Методическое пособие. 2-часть. **Бабина В.Г., Гулямов Ж.Н.** ТГТУ, Т.: 2021, 82 стр.

Во второй части методического пособия содержатся теоретические и практические сведения по созданию веб-страниц с использованием HTML и CSS, а также основы разработки макета адаптивного сайта с использованием Bootstrap.

Методическое пособие предназначено для преподавателей и студентов института ж.д. транспорта, обучающихся по направлению образования 5330200 – Информатика и информационные технологии (на железнодорожном транспорте).

Методическое пособие рекомендовано к изданию решением Научно-методического совета Ташкентского государственного транспортного университета.

Рецензенты: Равшанов Н. – вед. науч. сот. (ТУИТ);
Ибрагимов Р.И. – к.т.н., доц. (ТГТУ).

ЛАБОРАТОРНАЯ РАБОТА №8

Язык гипертекстов HTML

Цель работы: Научиться создавать HTML страницы, используя различные способы форматирования текста. Уметь задавать фон страницы и фоновый рисунок, вставлять графические изображения.

Задание на лабораторную работу

1. Создать заголовок браузера с произвольным именем (см. рис.1).

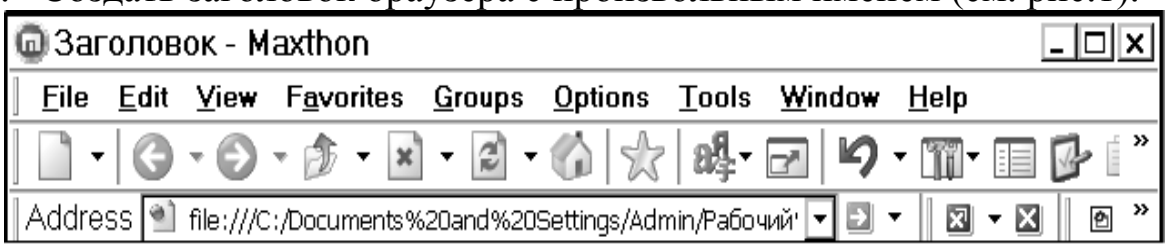


Рис.1. Заголовок страницы

2. Создать HTML-файл, содержащий 6 типов заголовков.

   Например:

   Заголовок 1-го уровня

   Заголовок 3-го уровня, отцентрированный

   Заголовок 6-го уровня

3. Написать четверостишие, используя теги перевода строки, абзаца и центрирования текста.
   Например:

Стихи
В лесу родилась елочка
В лесу она росла
Зимой и летом стройная
Зеленая была

4. Повторить запись, используя теги форматирования текста:

Невозможно отобразить страницу
Эта страница сейчас недоступна
~~Возможно, это вызвано техническими проблемами на веб-узле~~
Я подчеркнутый текст
Буквы одинаковой ширины
$Log_3(x)$
ax^2+bx+c
Этот текст написан Comic Sans MS
Буквы большего размера
Буквы меньшего размера

5. Используя таблицу цветов, раскрасьте ваше четверостишие из 3 задания.
6. Напишите свое резюме (данные о себе).
7. Оформить свое резюме в виде гипертекста на своем сайте. Выделите в гипертексте все фамилии и даты.
8. Вставить картинку в HTML-файл. (Создайте его в графическом редакторе Paint).
9. Вставить картинку как фон.

Теоретическая часть

Структура страницы

Гипертекстовая страница описывается на языке HTML и обрамляется парными маркерами:

<HTML> и </HTML>

Комментарий записывается внутри маркера комментария и не отображается браузером:

<!-- однострочный комментарий -->
<! многострочный
комментарий >

Страница состоит из заголовочной части <HEAD> и тела страницы <BODY>. Вот так выглядит разметка пустой Web-страницы:

<HTML>

```
<HEAD>
 <TITLE> Заглавие страницы </TITLE>
</HEAD>
<BODY>
        <!-- Тело страницы -->

</BODY>
</HTML>
```

Оформление страницы

1. Заголовки

На странице может содержаться одно *заглавие страницы* в заголовочной части страницы, которое отображается в заголовке окна браузера:

```
<HEAD>
  <TITLE>Заглавие страницы</TITLE>
</HEAD>
```

Кроме того, в теле страницы могут располагаться несколько *заголовков текста* различных уровней, которые отличаются размером используемого шрифта, от наибольшего (1-й уровень) до наименьшего (6-й уровень).

```
<Hn> заголовок n-го уровня </Hn>
```

Вместо n подставляются числа от 1 до 6.

Заголовки всегда начинаются с новой строки. Заголовки можно выравнивать по центру, по левому или по правому краю текста, например:

```
<H1 ALIGN="center">Заголовок 1-го уровня, отцентрированный </H1>
```

2. Форматирование

Текст страницы воспринимается браузером как одна длинная строка, повторяющиеся пробелы, переводы строки и другие символы

"пустого места" (white space) отбрасываются. Для вставки необходимых пробелов, а также других специальных символов (латинских букв, <, > и т.д) существуют так называемые ***&-последовательности***.. Каждая такая последовательность должна задаваться маленькими буквами (как всегда, не без исключений) и после них должна идти точка с запятой (;). Наиболее важные символы представлены в *Приложении В.*

Для форматирования текста страницы чаще всего применяются следующие маркеры:

```
<BR>          <! переход на новую строку>
<HR>          <! горизонтальная линия>
<P параметры> Новый абзац в тексте   </P>
<CENTER> Выравненный по центру текст </CENTER>
```

Параметры абзацев:

<P align=justify> - выравнивание справа и слева
<P align=center> - выравнивание по центру
<P align=right> - выравнивание по правому краю
<P align=left> - выравнивание по левому краю

Если требуется сохранить пробелы, сдвиги и другое предварительно отформатированное расположение текста, то применяются маркеры:

```
<PRE>
 Заранее
  отформатированный текст
    с сохранением отступов
  и пробелов
</PRE>
```

Для выделения в тексте разделов используется пара маркеров <DIV> с возможным указанием выравнивания при помощи атрибута ALIGN="LEFT | RIGHT | CENTER":

```
<DIV ALIGN="выравнивание">
Текст раздела
</DIV>
```

3. Начертание текста и шрифты

Теги управления отображением символов можно разбить на два класса: теги, управляющие формой отображения (font style), и теги, характеризующие тип информации (information type). Часто внешне разные теги при отображении дают одинаковый результат.

Для выделения в тексте фрагментов различного начертания применяются следующие маркеры:

<B> **жирное начертание** </B>
<I> *курсив* </I>
<S> ~~зачеркнутый текст~~ </S>
<U> подчеркнутый текст </U>
<TT> моноширинный текст </TT>
_{нижний индекс}
^{верхний индекс}

Часто используются вложения маркеров, чтобы выразить понятия типа "жирный подчеркнутый курсив":

<B> <I><U> ***вот такой текст*** </U></I></B>

Для выделения в тексте фрагментов различными гарнитурами ("шрифтами") используется парный маркер:

<FONT FACE="имя шрифта"> ... </FONT>

Например, так задается отображение текста определенной гарнитурой:

<FONT FACE="Times New Roman"> текст </FONT>

Для указания **относительного размера букв** в тексте применяются маркеры:

<BIG> Буквы большего размера </BIG>
<SMALL> Буквы меньшего размера </SMALL>

Можно также указывать **абсолютный размер букв** числами от 1 до 7 вместо n:

<FONT SIZE=n> текст </FONT>

Базовый размер текста указывается в маркере <BASEFONT SIZE="размер">

4. Цвет

Для указания ***цвета отдельных фрагментов*** текста применяется также тег:

<FONT COLOR=#rrggbb|"цвет"> ... </FONT>

Цвета задаются шестнадцатеричными числами по схеме RRGGBB ("красный - зеленый - синий") или названиями цветов (Приложение С). Например, зеленый цвет можно указать двумя способами:

```
<FONT COLOR=00FF00>зеленый </FONT>
<FONT COLOR="green"> зеленый </FONT>
```

Глобальные цвета текста и фона на странице, а также цвета ссылок (непосещенных, посещенных и выделенных) указываются в маркере описания тела страницы:

```
<BODY BGCOLOR=#rrggbb
  TEXT=#rrggbb
  LINK=#rrggbb
  VLINK=#rrggbb
  ALINK=#rrgbb>
```

5. Вставка изображений (images)

Для встраивания в текст страницы изображения служит маркер <IMG>, в котором указывается гиперссылка на графический файл с изображением, а также необязательные высота и ширина изображения и текст надписи для случая, когда изображение отсутствует.

```
<IMG SRC="адрес файла изображения"
  HEIGHT="высота"
  WIDTH="ширина"
ALIGN="LEFT|CENTER|RIGHT|MIDDLE|TOP|BOTTON"  –
```

выравнивание относительно текста

BORDER=n - толщина рамки

VSPACE=”расстояние между текстом и рисунком по вертикали”

HSPACE=”расстояние между текстом и рисунком по горизонтали”

ALT="текст подсказки">

Например, загрузка из подкаталога IMAGES изображения "yahoo" в формате GIF размером 38 на 30 пикселов и надписью "yahoo", описывается так:

<IMG SRC="images/yahoo.gif" HEIGHT="38" WIDTH="30" ALT="yahoo">

Всеми браузерами поддерживаются стандартные форматы изображений: .GIF и .JPG/.JPEG. Указание размеров изображения позволяет браузеру оптимизировать загрузку файла. Браузер сразу отводит рамку для изображения и продолжает загружать текст на страницу. Пока загружается графика, пользователь может начать читать текст.

На странице можно также поместить фоновый рисунок, указав атрибут BACKGROUND = "изображение" в описании тела страницы:

<BODY BACKGROUND="изображение">

Контрольные вопросы

1. Какова структура html-документа?
2. Как создается заголовок окна браузера?
3. Сколько существует уровней заголовков текста в теле документа?
4. Как создать новый абзац в тексте?
5. Назовите теги, которые используют для изменения начертания и цвета текста.
6. Как задается цвет?
7. Как в документ вставить изображение?
8. Можно ли использовать изображение в качестве фона документа?
9. С помощью какого тега можно расположить горизонтальную линию на странице?
10. Можно ли в браузере отобразить жирный перечеркнутый шрифт?
11. Какие параметры абзацев вы знаете?

ЛАБОРАТОРНАЯ РАБОТА №9

Списки, гиперссылки и карты в HTML

Цель работы: познакомиться с основными тегами создания списков, гиперссылок и графических карт HTML-документов; научиться использовать списки, гиперссылки и графические карты в качестве элементов оформления, навигации и структуризации информации в HTML.

Задание на лабораторную работу

1. Создать новый документ Html, содержащий три списка, различного вида (нумерованный, ненумерованный).

В первом списке привести перечень сделанных лабораторных работ. Второй список организовать самостоятельно.

Третий список сделать вида:

Ресурсы WWW: Содержание

1. Погода
2. Библиотеки
3. Вакансии на работу
4. e-mail
5. hostings

2. Отредактируйте файл, организовав гиперссылки для переходов:

a) по первому списку - на файлы лабораторных работ;

b) по третьему списку на сайты Интернета.

Создайте анкер в начале страницы с именем TOP.

Создайте гиперссылку на начало страницы в конце документа с переходом на анкер TOP.

Создайте гиперссылку ivanov@server.ru для перехода в почтовую программу.

Вставьте в страницу изображение. Сделайте это изображение гиперссылкой на сайт www.yandex.ru.

3. Вставьте на страницу изображение. Создать из изображения карту, области карты должны быть различного вида (круг, прямоугольник и произвольная область) с гиперссылками на поясняющие документы, например, как на рисунке 2.

Теоретическая часть

Списки

Список - это удобная форма представления массива информации, каждый элемент которого можно поместить в одну или несколько строк, а весь массив будет размещен в столбце таких строк.

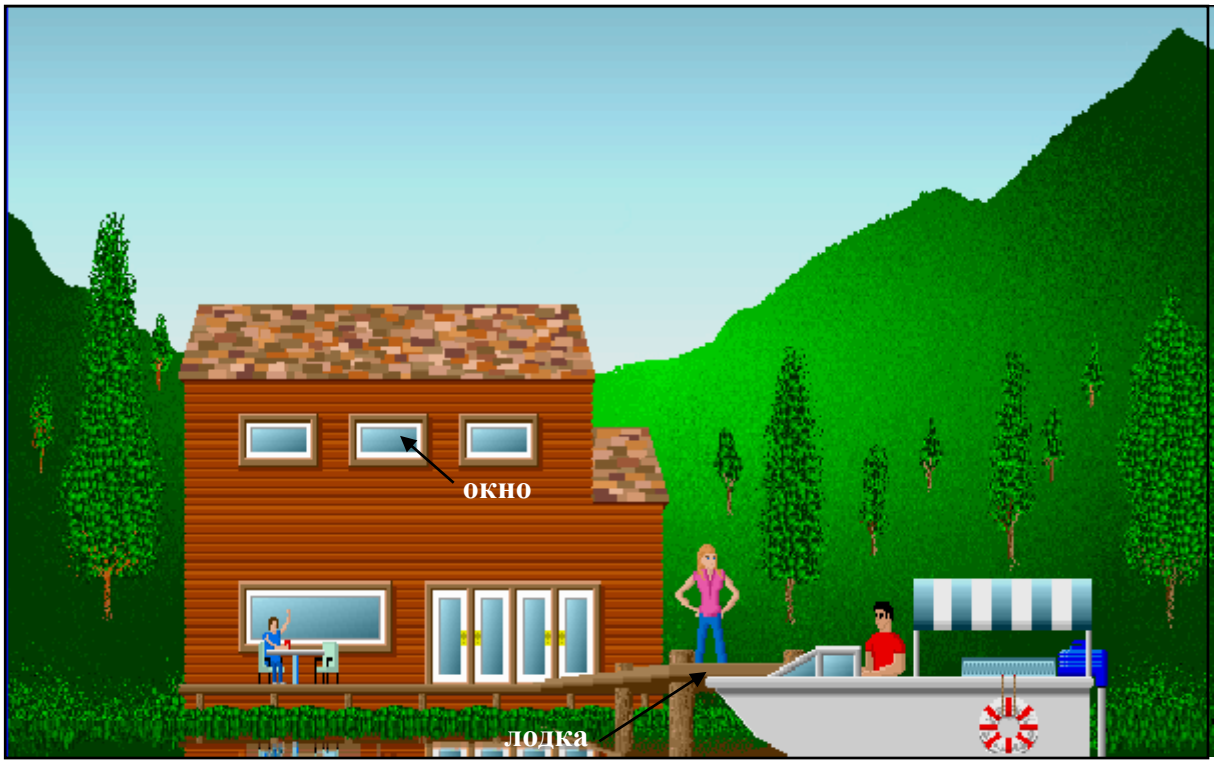


Рис.2. Пример картинки-карты

Форматы списков

Существует несколько форматов списков, позволяющих выделять фрагменты информации в тексте.

Ненумерованные списки: <UL> ... </UL>

Текст, расположенный между метками <UL> и </UL>, воспринимается как ненумерованный список. Каждый новый элемент списка следует начинать с метки <LI>. Ненумерованные или неупорядоченные (от английского **U**nordered **L**ists) списки имеют вид записи:

```
<UL>
  <LH>Название списка</LH>
```

```
  <LI>Первый пункт списка
  <LI>Второй пункт списка
  ....
</UL>
```

Тип маркировки (метки) элемента списка определяется атрибутом **type**. Допустимые значения атрибута и, соответственно, символы маркировки диск, круг, квадрат:

<UL type = "disc|circle|square"> - Для всего списка.

<LI type = "disc|circle|square"> - Этот и последующие элементы, в теги которых включен атрибут type.

Нумерованные списки: <OL> ... </OL>

Нумерованные списки устроены точно так же, как ненумерованные, только вместо символов, выделяющих новый элемент, используются цифры.

Для создания нумерованного списка используется элемент **OL.** Для этого элемента обязательны открывающий и закрывающий теги, которые обеспечивают перевод строки до и после списка, отделяя список от остального текста. Для маркировки заголовка или названия списка применяется элемент языка LH. Для маркировки отдельного элемента списка, как и для ненумерованного списка, применяется элемент языка LI.

Нумерованные или упорядоченные (от английского Ordered Lists) списки имеют вид записи:

```
<OL>
  <LH>Название списка</LH>
  <LI>Первый пункт списка
  <LI>Второй пункт списка
  ....
</OL>
```

Тип нумерации элемента списка определяется атрибутом type.

<OL type = "1|A|a|I|i">-Для всего списка.

<LI type = "1|A|a|I|i"> - Этот и последующие элементы, в теги которых включен атрибут type.

Пять возможных способов нумерации с помощью атрибута type в открывающем теге <OL> или <LI>:

- type=1 Стандартная цифровая нумерация - 1,2,3..

- type="A" Прописные буквы - A, B, C, ...
- type="a" Строчные буквы - a, b, c, ...
- type="I" Римские цифры - I, II, III, IV...
- type="i" Строчные римские цифры - i, ii, iii, iv, ...

Атрибут start позволяет начать нумерацию списка не с единицы. Например:

<OL start=4>

Значение атрибута value элемента LI позволяет изменить номер данного элемента списка. При этом изменяется и нумерация всех последующих элементов списка.

Например:

<LI value=5>

DL (списки определений)

В списке определений для каждого пункта предоставляется не одна, а две строки. Для написания каждого пункта списка применяются два элемента: DT - определяемый термин (Definition Term) и DD - описание определения (Definition Data).

Стандартный список определений:

```
<DL>
  <LH>Название списка</LH>
  <DT>Название термина 1
  <DD>Определение термина 1
  <DT>Название термина 2
  <DD>Определение термина 2
  ....
</DL>
```

Вл оженные списки

Каждый пункт списка может представлять собой самостоятельный список. Вложение списков в списки создает несколько уровней информации. Можно вкладывать друг в друга разные типы списков. Каждый внутренний список должен иметь открывающий и закрывающий теги UL или OL.

Гипертекстовые ссылки

Любой Web-документ может содержать ссылку на другой документ или на определенное место текущего или другого документа. Гипертекстовая ссылка - это подчеркнутый и выделенный определенным цветом фрагмент текста, указывающий на документ в одном из ресурсов Интернета или на метку внутри страницы такого документа.

Шаблон элемента гиперссылки

Шаблон элемента гиперссылки имеет вид:

<A href="Адрес ссылки"> текст для щелчка, отсылающий по адресу ссылки </A>

Каждый документ (файл) в Интернете имеет адрес, называемый Универсальным указателем ресурсов (uniform resource locator -URL) Поэтому атрибут href элемента гиперссылки A, задающий адрес ссылки, в общем случае имеет шаблон:

href="URL"
или
href="Протокол://Адрес ссылки"

В общем случае шаблон гиперссылки будет иметь вид: <A href="http://Адрес ссылки" target="параметр" title="заголовок ссылки"> текст гиперссылки для щелчка</A>

Атрибут target со значением "_self" вызывает загрузку ресурса в то же окно, в котором была активизирована гиперссылка, значение атрибута target - "_blank" вызывает загрузку в новое окно. Значение "_self" действует по умолчанию при отсутствии атрибута target в открывающем теге элемента гиперссылки.

Ссылки к файлам на локальном диске

Если ссылка указывает на файл, который находится на локальном диске, значение атрибута href обязательно должно начинаться со слова file, то есть содержать указание на протокол:

href="file://Имя диска:\Путь к файлу"

или
href="file:///Имя диска:/Путь к файлу"

По умолчанию используется ссылка на файлы текущего каталога (того, где расположена Web-страница). В этом случае просто указывается имя файла, например, рис1.gif; текст3.html.: href="Имя файла.тип"

Если в текущую папку вложена другая с необходимым файлом, шаблон атрибута href будет иметь вид:

href="Имя папки/Имя файла.тип"

Если папка с необходимым файлом находится на том же уровне вложения, что и текущая, шаблон гиперссылки имеет вид: href="../Имя папки/Имя файла.тип"

Ссылки для перехода к определенной части страницы текущего документа

В том случае, когда используются переходы внутри текущей страницы, на ней должны быть расставлены метки или якоря. Для них предусмотрен элемент:

<A name="Метка">ключевое слово или заголовок раздела</A>

В качестве метки рекомендуется использовать короткий набор латинских букв. Шаблон гиперссылки для перехода к метке внутри текущей страницы:

<A href="#Метка"> текст для щелчка</A>

Ссылки на почтовый ящик

Ссылка на почтовый ящик прописывается немного иначе, чем ссылка на другой документ (страницу, сайт):

 pochta@mail.ru

Задание цвета ссылки

Цвет текста ссылки или рамки изображения-ссылки задается с помощью атрибутов тега BODY, управляющих цветом гипертекстовых ссылок:

link="Цвет" Цвет ссылки, неиспользованной посетителем страницы.

vlink="Цвет" Цвет ссылки, использованной посетителем страницы. Visited Link.

alink="Цвет" Цвет ссылки в момент активизации (щелчка мыши). Active Link.

Изображения-карты в HTML-документах

Элемент IMG позволяет использовать изображения, отдельные части которых связаны со ссылками и позволяют выполнять переходы по адресам этих ссылок. Различные фрагменты Изображения-карты адресуют к различным HTML-файлам. Такие изображения называются КАРТАМИ.

Создание изображения-карты. Элементы MAP, AREA

Изображения-карты можно создать из любых графических изображений: пиктограмм, кнопок, рисунков и т.д., если использовать их в качестве фрагментов карты. Для общего определения карты используется элемент MAP, состоящий из открывающего и закрывающего тегов. Внутри открывающего тега задается имя карты с помощью атрибута name="*Имя карты*". Области карты определяются при помощи элементов AREA, которые находятся между открывающим и закрывающим тегами элемента MAP. Для каждой области карты должен быть создан свой элемент AREA. Он не имеет закрывающего тега. Таким образом, элемент MAP имеет шаблон:

<MAP><AREA></MAP>

Атрибуты элемента AREA:

- href="Адрес ссылки"
  Атрибут, определяющий адрес ссылки.
- alt="Текст подсказки"
  Атрибут для задания альтернативного текста, заменяющего фрагмент изображения карты.

Атрибуты, определяющие форму области на карте

Существуют три стандартных вида областей (см. рис.3.): Круг (circle), Прямоугольник (rect), Многоугольник произвольной формы (polygon).

Для круга задаются координаты центра и радиус (r) в пикселях.

Координаты центра отсчитываются от левого края (x) и верхнего края (y) рисунка. Шаблон для задания круговой области изображения-карты:

shape="circle" coords="x,y,r"

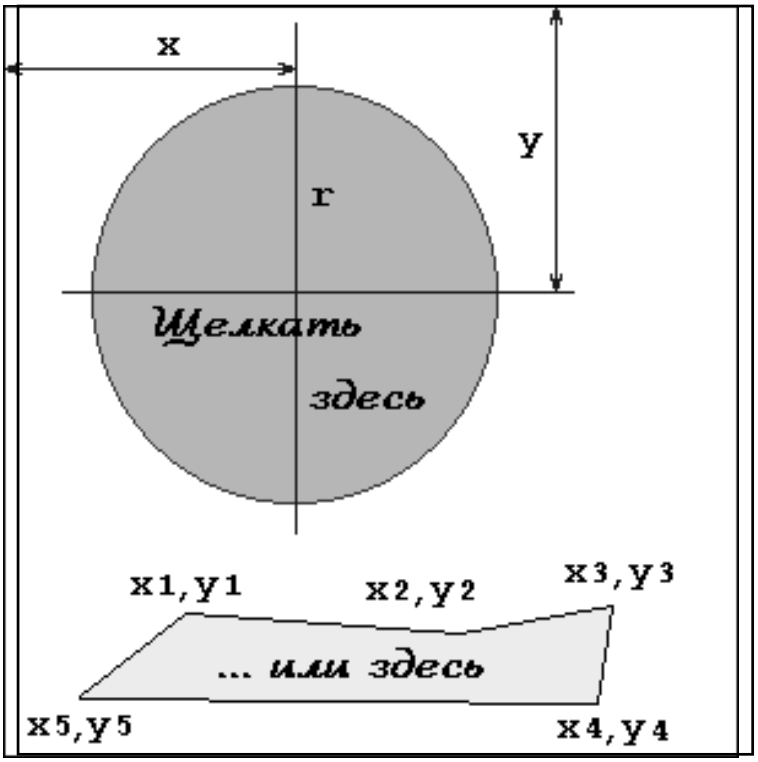


Рис 3. Пример задания координат элементов карты

Для прямоугольной области задаются координаты верхнего левого и правого нижнего углов прямоугольника:
shape="rect" coords="x1,y1, x2,y2"

Для определения области произвольной конфигурации задаются координаты (x,y) каждого из углов многоугольника, который точно или приблизительно соответствует по форме этой области:
shape="poly" coords="x1,y1, x2,y2, x3,y3..."

Для того, чтобы связать карту с картинкой, надо использовать атрибут
usemap="#имя_карты" для картинки
<img src="bluerects.gif" usemap="#karta1">
... Куча текста и всякого содержания, или ничего...
<map name="karta1">
<area href="drugoy_document.html" shape="rect"

coords="25,36,114,98">
</map>

Контрольные вопросы

1. Какие виды списков Вы знаете?
2. Что такое гиперссылка?
3. Что такое изображение-карта?
4. Какое значение атрибута SHAPE элемента AREA задает круговую область на изображении-карте?
5. Какой атрибут в элементе графического изображения придает его фрагментам свойства гиперссылки, то есть возможность осуществления переходов?
6. С какого слова должна начинаться запись значения атрибута HREF, если ссылка указывает на файл, который находится на локальном диске?
7. Каким образом можно задать цвет посещенной ссылки?
8. Как организуются ссылки внутри документа?
9. Можно ли организовать ссылку на файл-изображение?

ЛАБОРАТОРНАЯ РАБОТА №10

Основные элементы таблиц

Цель работы: Создание HTML-страницы с использованием таблиц. Научиться создавать таблицы в HTML документе и позиционировать с помощью них различные блоки.

Задание на лабораторную работу

1. Составить таблицу согласно варианту. Используйте атрибуты выравнивания текста и толщины рамки таблицы.

Вариант 1

<table>
<tr><td colspan="2">Заголовок1</td><td colspan="2">Заголовок2</td><td>Заголовок3</td><td>Заголовок4</td><td>Заголовок5</td></tr>
<tr><td></td><td></td><td colspan="2">По центру</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">Слева</td><td></td><td colspan="2">Справа</td></tr>
</table>

Вариант 2

Яч.1	Заголовок1	Заголовок2	Заголовок3	Яч.2	Заголовок4
	Слева				
		Яч.3			Справа

Вариант 3

Заголовок1	Заголовок2	Яч.1		Заголовок3	Заголовок4
Слева				Справа	
			Яч.2		

Вариант 4

Яч.1	Заголовок1	Заголовок2	Заголовок3	Заголовок4	
		По центру			Яч.2
	Слева			Справа	

Вариант 5

	Заголовок1	Заголовок2	Заголовок3	Заголовок4
Яч.1	Справа		Справа	Яч.2
	По центру	Слева	По центру	

Вариант 6

Яч.1	Заголовок1	Заголовок2	Заголовок3	Заголовок4	
	Слева				
	Справа	Справа	Справа	Справа	Яч.2

Вариант 7

Заголовок1		Заголовок2	Заголовок3	Заголовок4	Заголовок5
	Яч.1	Слева		Яч.2	
		Справа	Справа		

Вариант 8

Заголовок1		Заголовок2	Заголовок3	Заголовок4	
		По центру			
Слева				Справа	

Вариант 9

Заголовок1		Заголовок2	Заголовок3		Заголовок4
		Яч.1			Яч.2

Вариант 10

Яч.1	Яч.1	Заголовок1	Заголовок2	Яч.1	Яч.1
		Слева			
		Справа			

2. Создайте таблицу состоящую из трех вложенных таблиц**,** например, такую, как показано на рисунке 4.

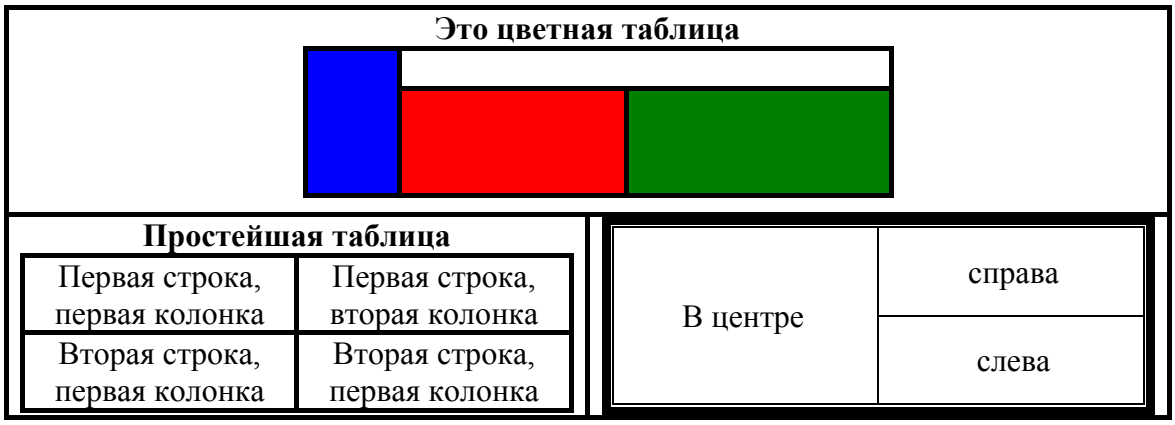


Рис.4. Вложенные таблицы

3. Создайте страничку, содержащую текст и картинки, для позиционирования элементов на ней используйте таблицы. Пример подобной странички показан на рисунке 5.

Теоретическая часть

Таблица представляет собой особую часть документа. Данные в ней организованы в виде прямоугольной сетки, состоящей из вертикальных столбцов и горизонтальных рядов.

Каждая клетка таблицы называется ячейкой. Ячейки могут содержать в себе текст, графику или другую таблицу. Текст и графика внутри ячейки могут быть ссылками.

Таблицы состоят из трех основных частей: названия таблицы, заголовков столбцов и ячеек. Таблица заполняется горизонтальными рядами ячейка за ячейкой слева направо. Заполнение начинается с левого верхнего угла и заканчивается правым нижним. Каждая ячейка должна быть заполнена. Для создания пустых ячеек используются пробелы.

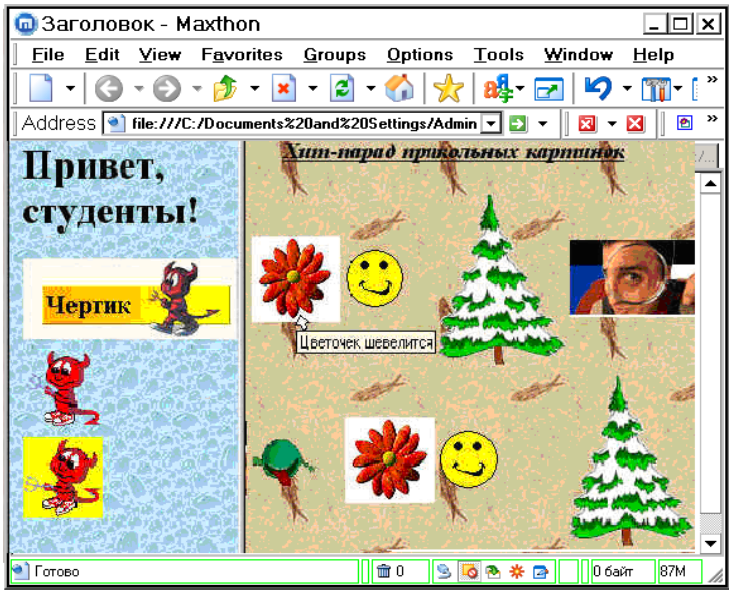


Рис.5 Пример страницы с позиционированием элементов при помощи таблиц

Структура таблицы

```
<TABLE                                                    BORDER>
    <CAPTION  ALIGN=TOP|BOTTOM>  Заголовок  таблицы
</CAPTION>
                                                          <TR>
  <TH>      Заголовок      1-го      столбца      </TH>
  <TH>      Заголовок      2-го      столбца      </TH>
                                                          </TR>
                                                          <TR>
  <TD>      1-ая      ячейка      данных      </TD>
  <TD>      2-ая      ячейка      данных      </TD>
                                                          </TR>
</TABLE>
```

По умолчанию заголовки центрируются и размещаются либо над (<CAPTION ALIGN=TOP>), либо под таблицей (<CAPTION ALIGN=BOTTOM>).

Составные ячейки

Несколько ячеек могут объединяться в одну как по горизонтали, так и по вертикали. Этот прием полезен для разработки таблиц, поскольку облегчает восприятие информации.

Для создания составных ячеек используются атрибуты COLSPAN и ROWSPAN, которые можно применять как в теге <TH> (для создания составных ячеек в строке заголовка), так и в теге <TD> (в основном поле таблицы).

Атрибут COLSPAN=

Теги <TD> и <TH> модифицируются с помощью атрибута COLSPAN= .Тогда одна ячейка строки заголовка будет относиться к нескольким столбцам. Количество столбцов задается в качестве аргумента после знака =.

```
<TABLE                    BORDER>
 <TR>
  <TH COLSPAN=2> Заголовок </TH>
 </TR>
 <TR>
```

Заголовок	
1-ая ячейка	2-ая ячейка

```
 <TD>      1-ая      ячейка      </TD>
 <TD>      2-ая      ячейка      </TD>
</TR>
</TABLE>
```

Атрибут ROWSPAN=

Атрибут ROWSPAN=, используемый в тегах <TD> и <TH>, задает число строк, на которые растягивается ячейка. Если указать в атрибуте ROWSPAN= число, большее единицы, то соответствующее количество строк должно находиться под растягиваемой ячейкой. Нельзя поместить ее внизу таблицы.

Регулировка ширины таблицы и ячеек

Общую ширину таблицы можно задавать двумя способами: в пикселах и в процентах от ширины окна. При задании в пикселах таблица всегда будет иметь одну и ту же ширину. Если же она задана в процентах, то всегда будет занимать одну и ту же часть окна броузера (сравните, как выглядит один и тот же документ с таблицами на рисунках 6 и 7).

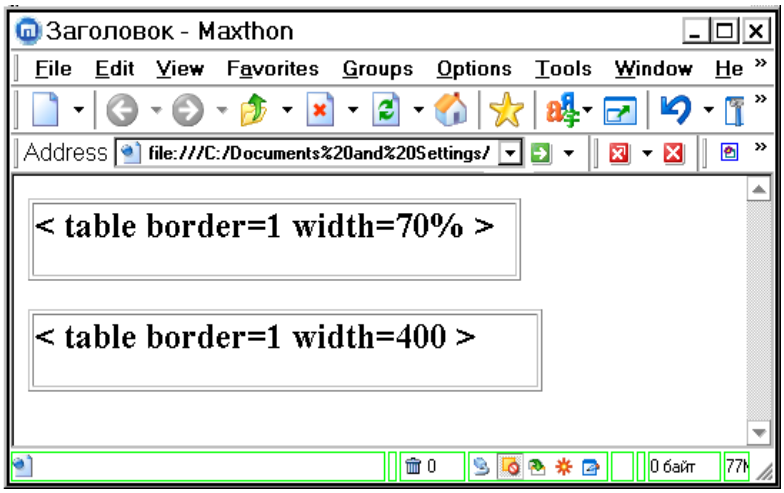


Рис.6. Окно максимального размера

Ширину отдельных ячеек тоже можно задавать двумя способами: в пикселах и процентах от общей ширины таблицы. При задании в пикселах все ячейки всегда будут принимать одну и ту же ширину. Если же размер задан в процентах, то постоянными будут пропорции ячеек в строке.

Пустые ячейки

Если ячейка не содержит данных, она не будет иметь границ. Если необходимо, чтобы у ячейки были границы, но не было содержимого, то нужно поместить в нее что-то, что не будет видно при просмотре. Можно воспользоваться пустой строкой
 или вставить символ пробела . Можно даже задать пустые столбцы, определив их ширину в пикселях или относительных единицах и не введя в полученные ячейки никаких данных. Это средство может оказаться полезным при размещении текста и графики на странице.

Выравнивание в таблицах

С помощью языка HTML можно управлять выравниванием текста и графики в ячейках таблиц. Если выравнивание не задавать, то название таблицы и заголовки центрируются, а содержимое прочих ячеек по горизонтали выравнивается по левому краю, а по вертикали - по средней линии ячейки.

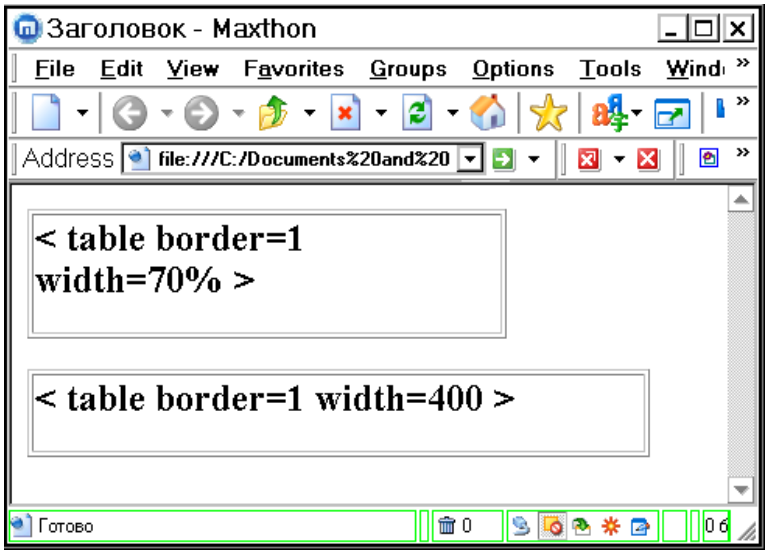


Рис.7. Окно уменьшенного размера

Атрибуты ALIGN= и VALIGN=

Теги <TR>, <TD> и <TH> можно модифицировать с помощью атрибутов ALIGN= и VALIGN=. Атрибут ALIGN определяет выравнивание текста и графики по горизонтали, то есть по левому или правому краю, либо по центру. Горизонтальное выравнивание может быть задано несколькими способами:

ALIGN=BLEEDLEFT - Прижимает содержимое ячейки вплотную к левому краю.

ALIGN=LEFT - Выравнивает содержимое ячейки по левому краю с учетом отступа, заданного атрибутом CELLPADDING=.

ALIGN=CENTER - Располагает содержимое ячейки по центру.

ALIGN=RIGHT -Выравнивает содержимое ячейки по правому краю с учетом отступа, заданного атрибутом CELLPADDING=.

Атрибут VALIGN= осуществляет выравнивание текста и графики внутри ячейки по вертикали. Вертикальное выравнивание может быть задано несколькими способами:

VALIGN=TOP - Выравнивает содержимое ячейки по ее верхней границе.

VALIMG=MIDDLE - Центрирует содержимое ячейки по вертикали.

VALIGN=BOTTOM - Выравнивает содержимое ячейки по ее нижней границе.

```
<TABLE BORDER WIDTH=150>
<TR HEIGHT=80>
 <TD VALIGN=TOP> Яч. 1 </TD>
 <TD VALIGN=MIDDLE> Яч. 2 </TD>
 <TD VALIGN=BOTTOM> Яч. 3 </TD>
</TR>
</TABLE>
```

Яч. 1	Яч. 2	Яч. 3

Оформление таблиц

Внешний вид таблиц можно разнообразить с помощью трех атрибутов: BORDER, CELLSPACING, CELLPADDING.

<TABLE BORDER=1>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

<TABLE BORDER=10>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Если необходимо, чтобы рамка между ячейками отличалась от рамки вокруг всей таблицы, задается атрибут CELLSPACING. После

символа = вводится ширина внутренних рамок в пикселах.

<TABLE BORDER CELLSPACING=1>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

<TABLE BORDER CELLSPACING=10>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Для задания пустых полей между содержимым ячейки и ее рамкой задается атрибут CELLPADDING. Размер полей задается в пикселах после символа =.

<TABLE BORDER CELLPADDING=1>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

<TABLE BORDER CELLPADDING=10>

Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Цветовое оформление таблиц

Цвет всей таблицы, ее рядов и отдельных ячеек задается атрибутом BGCOLOR=. Если нужно задать цвет всей таблицы, этот атрибут ставится в теге <TABLE>. Если нужно изменить цвет всего ряда, то в теге <TR>, а если в одной ячейке, то в теге <TD>.

Цвета задаются либо шестнадцатеричными кодами трех составляющих цветов, либо указанием стандартного названия цвета (Приложение С).

Контрольные вопросы

1. Для чего можно использовать таблицы?
2. Можно ли создавать вложенные таблицы?
3. Что можно изменить в таблице? (форматирование, цвета,…)
4. Как регулируется ширина таблицы, ячеек?
5. Как задаются пустые ячейки в таблице?
6. Как задается расстояние между ячейками?
7. С помощью какого атрибута устанавливается размер полей между содержимым ячейки и рамкой?

8. Можно ли задать цвет для каждой ячейки отдельно?
9. Каким образом можно объединить ячейки в таблице?
10. Каким образом регулируется расположение текста в ячейке?

ЛАБОРАТОРНАЯ РАБОТА №11

Применение каскадных таблиц стилей CSS

Цель работы: научиться форматировать HTML-документ с применением каскадных таблиц стилей, уметь форматировать разметку страниц с помощью блоков (DIV) и CSS

Задание на лабораторную работу

1. Создайте каталог /css/
2. Создайте в нем файл main.css
3. Задайте по умолчанию следующие параметры для всех страниц (переопределив, тег <body>):

– цвет фона
– размер шрифта
– цвет шрифта
– семейство шрифта (например, Arial)

4. Задайте по умолчанию следующие параметры для всех абзацев (переопределив, тег <p> и псевдоклассы тега <p>):

– выравнивание абзаца
– отступ красной строки
– размер и цвет первой буквы

5. Задайте по умолчанию следующие свойства ссылок для всех страниц:

– цвет и оформление ссылки
– цвет и оформление посещенной ссылки
– цвет и оформление активной ссылки
– цвет и оформление ссылки, в момент нахождения курсора мыши над ней

6. Подключите файл main.css к страницам сайта с помощью тега

<LINK>.

7. Создайте новую страницу. Сделайте ее визуальной копией первой страницы, но используйте не таблицы, а блоки и стили.
8. Создайте новую страницу. Сделайте коллаж из картинок или фотографий (не менее 5-ти) применив 2,5 мерное позиционирование.

Теоретическая часть

CSS Cascading Style Sheets (Таблицы каскадных стилей) — это набор правил оформления и форматирования, который может быть применен к различным элементам страницы. В HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходиться каждый раз описывать эти свойства. Применяя CSS, можно один раз описать свойства элементов и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который должен быть оформлен соответствующим образом, должен принять свойства стиля, описанного ранее.

Встраивание CSS в документ

Существует три способа применения таблиц стилей:

Внутренние таблицы стилей (Inline Style Sheets) - при помощи специального атрибута помещаются прямо в HTML теги.

Пример HTML:

 Пример

Пример CSS:

 Пример

Код Inline Style Sheet получился больше чем HTML. Поэтому ISS следует использовать, только если необходимо задать определенному элементу свой индивидуальный стиль. Этот способ действует в пределах лишь одного тега.

Глобальные таблицы стилей (Global Style Sheets) - определяют стиль элементов во всем документе.

Для этого используется тег <STYLE type="text/css">. Он размещается в заголовке документа <head>.

Пример:

<html>

<head>
<STYLE type="text/css">
h1 { color:red; font-style:italic; font-size:32px} - переопределение стандартного тега.
.blue { color:blue} - определение нового класса
</STYLE>
</head>
<body>

Теперь эти стили можно применять в любом месте html-кода. Для этого используются следующие конструкции:

<h1> Этот заголовок написан крупным красным курсивом </h1>
Вот это слово - синие.
</body>
</html>

В данном примере все элементы H1 будут написаны крупным красным курсивом, все элементы с указанным классом BLUE будут синими.

Связанные таблицы стилей (Linked Style Sheets) - могут быть использованы для нескольких документов сразу и хранятся во внешнем файле.

Пример внешнего файла:

Файл main.css

body {background:black; font-size:9pt; color:red; font-family:Arial Black}

.base{color:blue; font-style:italic}

h1 {color:white}

#bold {font-weight:bold}

В самих же HTML документах делается ссылка на этот файл при помощи тега <LINK>.

Выглядит это так:

<LINK rel="STYLESHEET" TYPE="text/css" HREF="путь до файла">

Синтаксис CSS

Селекторы (Selectors)

Синтаксис:
селектор {свойства}
Любой элемент HTML — это возможный CSS селектор. Свойства селектора определяют стиль элемента, для которого он определен.
Например: H1 {color:red; size:20pt;}
Все элементы H1 в документе будут красного цвета, размером в 20 пунктов (типограф.).
Если необходимо придать нескольким элементам веб-страницы одинаковые свойства, то в этом случае селекторы при определении перечисляются через запятую перед блоком свойств.

Классовые селекторы (Class Selectors)

Синтаксис:
селектор. класс {свойства}
CLASS — атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов.

Например:
H1.blue {color:blue; size:20pt;}
Все элементы H1 с атрибутом CLASS="blue" станут синими.
Классы также могут быть описаны без явной привязки к определенным элементам.
Синтаксис:
.класс {свойства}
Например:
.green {color:green;}

В данном случае все элементы с атрибутом CLASS="green" станут зелеными.

Псевдоклассы

В CSS есть такое понятие как псевдокласс. В отличие от обычного класса, действие псевдокласса распространяется не на весь текст, к

которому применен данный стиль, а лишь на его часть и возможно лишь в определенном состоянии. Например эффект, при котором ссылки подчеркиваются лишь при наведении на них курсора. Фрагмент таблицы стилей, который позволяет достигать вышеописанного эффекта:

a { text-decoration: none; }

a:hover { text-decoration: underline; }

Верхняя строчка - это переопределение стандартного тега <a>, которое запрещает подчеркивать ссылки, а вот нижняя - это определение стиля для псевдокласса hover, который описывает стиль ссылки в момент, когда курсор находится над ней.

ID-селекторы (ID Selectors)

Синтаксис:

#id {свойства}

ID - индивидуально именованный стиль. С его помощью можно создавать стилистические исключения среди элементов одного класса.

Идентификаторы используются для придания одному или нескольким элементам одного класса индивидуальных свойств. Например, вы создали класс blue - синий курсив. Но вам понадобился жирный подчеркнутый текст синим курсивом. Конечно, можно создать новый класс, но проще описать ID. Например "bold_under_line". И все элементы класса blue со значением ID "bold_under_line" станут жирным подчеркнутым синим курсивом.

Контекстные селекторы (Contextual Selectors)

Контекстный селектор — это сочетание нескольких обыкновенных селекторов. Стиль задается только элементам в заданной последовательности в зависимости от каскадного порядка.

Например:

P I {color:silver;}

В данном примере все элементы I внутри элементов P будут иметь заданный стиль.

Некоторые параметры CSS

Некоторые параметры шрифта:

font-weight: [bold | normal | number] - жирность шрифта

font-style: [normal | italic | oblique] - наклон шрифта
font-size: number - размер шрифта
font-family: name - гарнитура шрифта
color: number - цвет шрифта
background-color: number - цвет подложки
background: url - текстурная подложка

Некоторые параметры абзаца
text-align: [left | right | center | justify] – выравнивание
text-indent: number - отступ красной строки
line-height: number – интерлиньяж
letter-spacing: number – трекинг
padding-left: number - отступ от текста слева
padding-right: number - отступ от текста справа
padding-top: number - отступ от текста сверху
padding-bottom: number - отступ от текста снизу
margin-left: number - отступ от границы слева
margin-right: number - отступ от границы справа
margin-top: number - отступ от границы сверху
margin-bottom: number - отступ от границы снизу
Основные параметры CSS приведены в приложении D.

Единицы измерения в CSS

В свойствах, которым требуется указание размеров, можно использовать несколько способов для их задания:

- относительный размер в процентах (%);
- относительный размер при помощи словесного описания (larger, smaller, xx-small, x-small, small, medium, large, x-large, xx-large);
- абсолютный размер в типографских единицах - размер может задаваться в пунктах (pt), пиках (pc), средней шириной буквы "m" (em), средней шириной буквы "x" (ex);
- абсолютный размер в стандартных единицах длины - размер может задаваться в сантиметрах (cm), миллиметрах (mm), дюймах (in);
- абсолютный в пикселях (px).

Задание цвета в CSS

Цвет для тех свойств, где это нужно, может быть определен одним из трех способов:

- при помощи названия цвета: yellow, red, green, grey,..
- шестнадцатеричным заданием цвета в формате #RRGGBB: #ff0000, #883490, #ffffff,..
- десятичным заданием составляющих цвета в формате rgb(red, green, blue): rgb(255,0,0), rgb(100,23,78),..

Разметка страниц с помощью блоков (DIV) и CSS. Позиционирование

Тэг <DIV> - служит для группирования элементов в блок.

К сгруппированным элементам можно применить стили.

Примеры. Выделение блока бордюром

Границу можно легко разместить вокруг заголовка, списка, абзаца или их группы, поместив их в элемент div.

Прописываем стили для этого блока

Это можно использовать с разметкой следующим образом:

```
<div style="border-color:#FF00FF; border-style:dotted; ">
Содержимое этого элемента DIV будет заключено в прерывистую рамку.
</div>
```

```
<div style="border:15px double #008000; ">
Содержимое этого элемента DIV будет заключено в сплошную рамку шириной 15 пикселов.
</div>
```

```
<div style="border-right: 10px solid #00FF00; border-bottom: 10px solid #00FF00">
Содержимое этого элемента DIV будет заключено в такую рамку (нижняя и правая рамки).
</div>
```

Пример. Выделение блока цветом фона

```
<div style="background-color: #00FFFF">
Содержимое этого элемента DIV будет выделено таким цветом.
</div>
```

Пример позиционирования и задания размеров элементов показан на рис. 8.

Рис.8. Позиционирование и задание размеров элементов

Пример позиционирования элементов в 2,5 мерном измерении показан на рис.9. Порядок наложения (перекрытия) блоков определяется атрибутом z-index.

Код этого примера:

```
<div style="position: relative; width: 200;
    height: 200; z-index: 0; background-color:
    #FFFF00">
    Блок – 1
    </div>
    <div style="position: relative; width: 200;
    height: 200; left: 100; z-index: 1; top: -100;
    background-color: #00FFFF">
    Блок – 2
    </div>
    <div style="position: relative; left: 220; top: -400; width: 200; height:
200; z-index: 2;
    background-color: #00FF00">
```

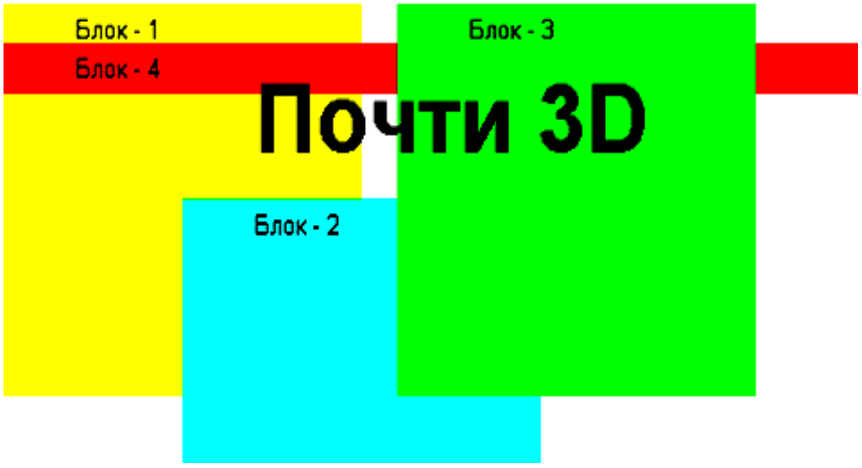


Рис.9. Позиционирование в 2,5 мерном измерении

Блок – 3

<div style="position: relative; width: 750;
height: 20; z-index: 1; top: -580; background-
color: #FE76AF">
Блок – 4

<div style="position: relative; top: -600; z-
index: 3; left: 100">
Почти 3D

Управление переполнением и видимостью

При переполнении следующего блока:
<div style="width: 200; height: 200;
background-color: #00FFFF">

блок будет увеличиваться (т.е. также как и таблица) (рис.10).

Переполнениеееееееееееееееееееееееееееееееееее
Переполнение
Переполнение
Переполнение
переполнение
переполнение переполнение
переполнение
переполнение

Рис.10. При переполнении блока – блок увеличивается

При переполнении следующего блока:

блок не будет увеличиваться, информацию можно просмотреть с помощью прокрутки (рис.11.)

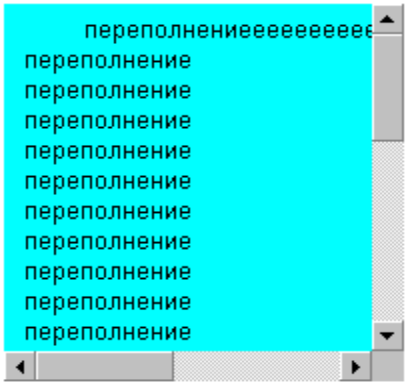


Рис.11. При переполнении блока – появляются полосы прокрутки

При переполнении следующего блока:

<div style="overflow: hidden; width: 200;

height: 200; background-color: #00FFFF">

блок не будет увеличиваться, информация, не поместившаяся в блок, не будет отображена (рис.12).

Комментарии

При создании таблицы стилей можно пользоваться комментариями.

Пример: /* Этот текст является комментарием */

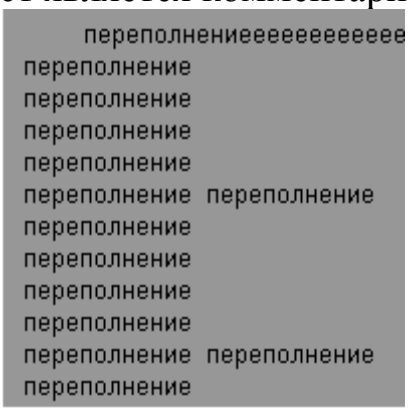


Рис.12. При переполнении блока – информация обрезается

Контрольные вопросы

1. Какие проблемы HTML-разметки решают CSS?
2. Сколько способов размещения описания CSS вы знаете?
3. В чем отличие свойств описания шрифта от свойств описания текста?
4. Каким образом можно переопределить стиль отображения элемента разметки умолчания?
5. Каким образом осуществляется ссылка на внешнее описание стиля?

Чему равно поле отступа текста от левой границы страницы в примере:

```
body {left-padding:20px;} Div {left-padding:0px;margin:5px;}
    p {padding:5px;}
    <body>
<div>
<p> ...</p>
    </div></body> ?
```

ЛАБОРАТОРНАЯ РАБОТА №12

Формы в HTML-документах

Цель работы: Формирование умений создание различного вида форм с помощью HTML.

Задание на лабораторную работу

Создайте новую страницу. Составьте форму-анкету (используя метод **POST**), включающую в себя следующие поля (**все переменные должны быть, читаемы, например: русский язык - langru, а не C1 или T2, по умолчанию в значениях должны быть указаны ваши данные**):

- Фамилия
- Имя
- Отчество
- E-mail

- Выбор страны (обязательно выпадающим **SELECT**, стран не менее 10-ти)
- Выбор города (обязательно с помощью **radio,** переменные должны быть одинаковыми**,** не менее 5-ти)
- Выбор языка (обязательно с помощью **checkbox**, переменные должны быть разными, не менее 5-ти)
- Выбор профессий (обязательно с помощью **SELECT MULTIPLE,** переменные должны быть разными, не менее 10-ти)
- Пароль
- Дополнительная информация (обязательно с помощью **TEXTAREA**)
- В скрытом поле (**hidden**), передайте переменную student со значением "Ваше_имя"(student=Фамилия_имя).
- Кнопка для загрузки информации на сервер
- Кнопка для очистки формы

Данные из формы должны посылаться обработчику на сервер по URI - /internet/test/form.php**.**

Теоретическая часть

Формы передают информацию программам-обработчикам в виде пар [имя переменной]=[значение переменной]. Имена переменных следует задавать латинскими буквами.

Значения переменных воспринимаются обработчиками как строки, даже если они содержат только цифры.

Форма открывается тегом **<FORM>** и заканчивается меткой **</FORM>.** HTML-документ может содержать в себе несколько форм, однако формы не должны находиться одна внутри другой.

Тег **<FORM>** может содержать три атрибута, один из которых является обязательным:

ACTION Обязательный атрибут. Определяет, где находится обработчик формы.

METHOD Определяет, каким образом (иначе говоря, с помощью какого метода протокола передачи гипертекстов) данные из формы будут переданы обработчику. Допустимые значения: METHOD=POST и METHOD=GET. Если значение атрибута не установлено, по умолчанию предполагается METHOD=GET.

- **GET:** методом "get" HTTP браузер берёт значение action, добавляет `?' к нему, затем присоединяет набор данных формы, кодированный с использованием типа содержимого "application/x-

www-form-urlencoded". Затем перенаправляет всё по гиперссылке на этот URL. В этом сценарии данные формы ограничены кодами ASCII (нельзя использовать спецсимволы) и имеют весьма жесткие ограничения на объем вводимой информации.

- **POST:** методом "post" HTTP браузер проводит транзакцию HTTP "post" (в теле HTTP-запроса), используя значение атрибута action и сообщение, созданное в соответствии с типом содержимого, определённым атрибутом enctype.

ENCTYPE Определяет, каким образом данные из формы будут закодированы для передачи обработчику. Если значение атрибута не установлено, по умолчанию предполагается ENCTYPE=application/x-www-form-urlencoded.

Основные элементы форм представлены в табл.2.

"Кнопка", чтобы запустить процесс передачи данных из формы на сервер, создается с помощью тега:

<INPUT TYPE=submit> исполнение==> 

Таблица 2

Основные элементы форм

<form></form>	Создает формы
<select multiple name="NAME" size="?"></select>	Создает скролируемое меню. Size устанавливает кол-во пунктов меню, которое будет показано на экране, остальные будут доступны при использовании прокрутки.
<option>	Указывает каждый отдельный элемент меню
<select name="NAME"> </select>	Создает ниспадающее меню
<textarea name="NAME" cols=40 rows=8></textarea>	Создает окно для ввода текста. Columns указывает ширину окна; rows указывает его высоту.
<input type="checkbox" name="NAME">	Создает checkbox.
<input type="radio" name="NAME" value="x">	Создает radio кнопку.
<input type=text name="foo" size=20>	Создает строку для ввода текста. Параметром Size указывается длина в символах.
<input type="submit" value="NAME">	Создает кнопку "Отправить"
<input type="image" border="0" name="NAME" src="name.gif">	Создает кнопку "Отправить" - для этого используется изображение
<input type="reset">	Создает кнопку "Очистить"

Встретив такую строчку внутри формы, браузер нарисует на экране кнопку с надписью Submit, при нажатии на которую все имеющиеся в

форме данные будут переданы обработчику, определенному в метке.

Надпись на кнопке можно задать любую путем введения атрибута VALUE="[Надпись]"например:

<INPUT TYPE=submit VALUE="Отправить!"> исполнение==> Отправить!

Надпись, нанесенную на кнопку, можно при необходимости передать обработчику путем введения в определение кнопки атрибута NAME=[имя] например:

<INPUT TYPE=submit NAME=button VALUE = "Отправить!"> исполнение==> Отправить!

При нажатии на такую кнопку обработчик вместе со всеми остальными данными получит и переменную button со значением Отправить! (т.е. button=Отправить!, это можно видеть в адресной строке).

В форме может быть несколько кнопок типа submit с различными именами и/или значениями. Обработчик, таким образом, может действовать по-разному в зависимости от того, какую именно кнопку submit нажал пользователь.

Существуют и другие типы элементов **<INPUT>**. Каждый элемент **<INPUT>** должен включать атрибут **NAME=[имя]**, определяющий имя переменной, которая будет передана обработчику. Имя должно задаваться только латинскими буквами. Большинство элементов **<INPUT>** должны включать атрибут **VALUE="[значение]"**, определяющий значение, которое будет передано обработчику под этим именем.

Основные типы элементов <INPUT>:

TYPE=text

Определяет окно для ввода строки текста. Может содержать дополнительные атрибуты SIZE=[число] (ширина поля для ввода, в символах) и MAXLENGTH=[число] (максимально допустимая длина вводимой строки в символах).

Пример:

<INPUT TYPE=text SIZE=30 NAME=student VALUE="Вася Пупкин"> ==> Вася Пупкинfsaafaaaffasfsf

Определяет ширину поля в 30 символов, для ввода текста. По

умолчанию в окне находится текст Вася Пупкин, который пользователь может редактировать. Отредактированный (или неотредактированный) текст передается обработчику в переменной student (student=содержимое_поля). Попробуйте отредактировать поле.

TYPE=password

Определяет окно для ввода пароля. Абсолютно аналогичен типу text, только вместо символов вводимого текста показывает на экране звездочки (*), чтобы посторонний не мог прочесть.

Пример:

<INPUT TYPE=password NAME=pswd SIZE=20 MAXLENGTH = 10> ==> ****

Определяет окно шириной 20 символов для ввода пароля. Максимально допустимая длина пароля — 10 символов. Введенный пароль передается обработчику в переменной pswd (pswd=содержимое_поля). Попробуйте ввести информацию в поле.

TYPE=radio

Определяет радиокнопку. Может содержать дополнительный атрибут checked (показывает, что кнопка помечена). В группе радиокнопок с одинаковыми именами может быть только одна помеченная радиокнопка.

Пример:

<INPUT TYPE=radio NAME=modem VALUE="9600" checked> 9600 бит/с ==> 9600 бит/с

<INPUT TYPE=radio NAME=modem VALUE="14400"> 14400 бит/с ==> 14400 бит/с

<INPUT TYPE=radio NAME=modem VALUE="28800"> 28800 бит/с ==> 28800 бит/с

Определяет группу из трех радиокнопок, подписанных 9600 бит/с, 14400 бит/с и 28800 бит/с. Первоначально помечена первая из кнопок. Если пользователь не отметит другую кнопку, обработчику будет передана переменная modem со значением 9600 (modem=9600). Если пользователь отметит вторую кнопку, обработчику будет передана переменная modem со значением 14400 (modem=14400).

TYPE=checkbox

Определяет квадрат, в котором можно сделать пометку. Может содержать дополнительный атрибут checked (показывает, что квадрат помечен). В отличие от радиокнопок, в группе квадратов с одинаковыми именами может быть несколько помеченных квадратов.

Пример:

<INPUT TYPE=checkbox NAME=comp VALUE="PC"> Персональные компьютеры ==> ☐ Персональные компьютеры

<INPUT TYPE=checkbox NAME=comp VALUE="WS" checked> Рабочие станции ==> ☑ Рабочие станции

<INPUT TYPE=checkbox NAME=comp VALUE="LAN"> Серверы локальных сетей ==> ☐ Серверы локальных сетей

<INPUT TYPE=checkbox NAME=comp VALUE="IS" checked> Серверы Интернет ==> ☑ Серверы Интернет

Определяет группу из четырех квадратов. Первоначально помечены второй и четвертый квадраты. Если пользователь не произведет изменений, обработчику будут передана одна переменная comp с двумя значениями (comp=WS и comp=IS).

TYPE=hidden

Определяет скрытый элемент данных, который не виден пользователю при заполнении формы и передается обработчику без изменений. Такой элемент иногда полезно иметь в форме, в него можно спрятать от пользователя служебные данные.

Пример:

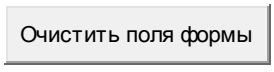
<INPUT TYPE=hidden NAME=id VALUE="1">

Определяет скрытую переменную индексную id, которая передается обработчику со значением 1.

TYPE=reset

Определяет кнопку, при нажатии на которую форма возвращается в исходное состояние (обнуляется). Поскольку при использовании этой кнопки данные обработчику не передаются, кнопка типа reset может и не иметь атрибута name. Пример:

<INPUT TYPE=reset VALUE="Очистить поля формы"> ==>

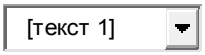


Определяет кнопку Очистить поля формы, при нажатии на которую форма возвращается в исходное состояние.

Элемент <SELECT>:

Меню <SELECT> из n элементов выглядит примерно так:

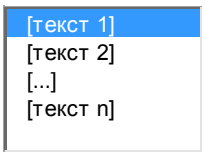
<SELECT NAME="[имя]">
<OPTION VALUE="[значение 1]">[текст 1]
<OPTION VALUE="[значение 2]">[текст 2]
...
<OPTION VALUE="[значение n]">[текст n]
</SELECT>



Метка <SELECT> содержит обязательный атрибут NAME, определяющий имя переменной.

Метка <SELECT> может также содержать атрибут MULTIPLE, присутствие которого показывает, что из меню можно выбрать несколько элементов. Большинство браузеров показывают меню <SELECT MULTIPLE> в виде окна, в котором находятся элементы меню (высоту окна в строках можно задать атрибутом SIZE=[число]). Для выбора нескольких значений одновременно удерживают кнопку "SHIFT" и выбирают значения мышкой.

<SELECT MULTIPLE SIZE=3 NAME="[имя]">
<OPTION VALUE="[значение 1]">[текст 1]
<OPTION VALUE="[значение 2]">[текст 2]
<OPTION VALUE="...">[...]
<OPTION VALUE="[значение n]">[текст n]
</SELECT>



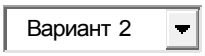
.

Метка <OPTION> определяет элемент меню. Обязательный атрибут VALUE устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Метка <OPTION> может включать атрибут selected, показывающий, что данный элемент отмечен по умолчанию.

Пример:

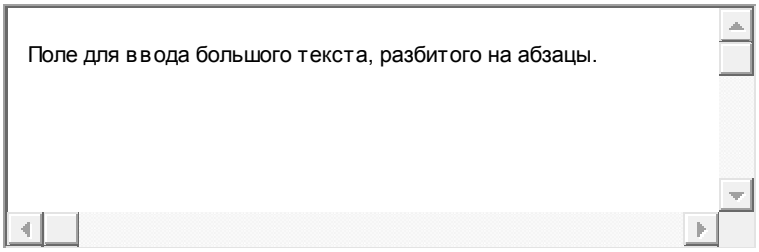
<SELECT NAME="selection">

<OPTION VALUE="option1">Вариант 1
<OPTION VALUE="option2" selected>Вариант 2
<OPTION VALUE="option3">Вариант 3
</SELECT>



Такой фрагмент определяет меню из трех элементов: Вариант 1, Вариант 2 и Вариант 3. По умолчанию выбран элемент Вариант 2. Обработчику будет передана переменная selection (selection=...) значение которой может быть option1 (по умолчанию), option2 или option3.

Элемент <TEXTAREA>:
Пример:
<TEXTAREA NAME=address ROWS=5 COLS=50>
Поле для ввода большого текста, разбитого на абзацы.
</TEXTAREA>



Все атрибуты обязательны. Атрибут **NAME** определяет имя, под которым содержимое окна будет передано обработчику (в примере — address). Атрибут **ROWS** устанавливает высоту окна в строках (в примере — 5). Атрибут **COLS** устанавливает ширину окна в символах (в примере — 50).

Текст, размещенный между метками <TEXTAREA> и </TEXTAREA>, представляет собой содержимое окна по умолчанию. Пользователь может его отредактировать или просто стереть.

Контрольные вопросы

1. В каких случаях используются формы?
2. Какие атрибуты может содержать тег <FORM>?
3. Как создать кнопку?
4. Чем отличается атрибут NAME от атрибута VALUE?
5. Как создается поле для ввода пароля?

6. Для чего используется тег SELECT?
7. Нужно ли устанавливать фиксированные размеры окна для ввода текста?

ЛАБОРАТОРНАЯ РАБОТА №13

Работа с сеткой Grid в Bootstrap

Цель работы: Формирование умений создание различного вида макетов сайта с использованием сетки в Bootstrap.

Задание на лабораторную работу

Разработать макет сайта с использованием библиотеки Twitter Bootstrap. Использовать для размещения отдельных элементов сетку. Макет сайта должен включать следующие элементы:

- Таблицы.
- Элементы HTML-форм.
- Панель навигации (в верхней части страницы).
- Выпадающие списки кнопок (могут быть использованы в панели навигации).
- Индикаторы прогресса.

Теоретическая часть

Bootstrap на данный момент является самым распространенным и уважаемым фреймворком для проектирования веб-приложений. Элементы Bootstrap можно грубо разделить на два типа. К первому типу следует отнести модульную сетку, а ко второму разнообразные составляющие веб-приложений, такие как кнопки или выпадающие меню. 12-ти колоночная модульная сетка, основа Bootstrap, используя её можно строить надёжные адаптивные веб-страницы, подходящие для любого типа задач.

Начало работы

Для начала работы с Bootstrap следует скачать файлы, составляющие библиотеку и разобраться в их структуре. После извлечения файлов из скаченного архива можно увидеть следующие файлы:

- bootstrap.css – основной файл Bootstrap, определяющий css оформление элементов библиотеки;
- bootstrap.min.css – содержит те же стилевые описания, что и предыдущий файл, но сжат для быстрой загрузки;
- bootstrap-theme.css – файл переопределяет стандартное стилевое оформление элементов Bootstrap, можно использовать, как частичную замену bootstrap.css;
- bootstrap-theme.min.css - содержит те же стилевые описания, что и предыдущий файл, но сжат для быстрой загрузки;
- bootstrap-theme.css.map –позволяет работать с файлами тем, так будто они не были сжаты;
- bootstrap.css.map – позволяет работать с файлом bootstrap.min.css, так будто он не был сжат;
- bootstrap.js – файл, отвечающий за динамические возможности библиотеки;
- bootstrap.min.js – сжатый файл bootstrap.js;
- glyphicons-halflings-regular.eot - в данном файле находятся векторные иконки библиотеки для браузера Internet Explorer;
- glyphicons-halflings-regular.svg – в данном файле находятся те же шрифты и иконки, что в предыдущем, но в формате векторной графики svg;
- glyphicons-halflings-regular.ttf – стандартный файл шрифтов;
- glyphicons-halflings-regular.woff – сжатый файл шрифтов.

Для начала работы следует подключить bootstrap.css и bootstrap.js или сжатые альтернативные файлы. Так же для полноценной работы библиотеки следует подключить JQuery. Примерно так должен выглядеть начальный документ, основанный на Bootstrap:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Bootstrap</title>
  <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
 </head>
 <body>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script src="bootstrap/js/bootstrap.min.js"></script>
 </body>
</html>
```

Модульная сетка

Сетка сделана из ***группирования строк и колонок*** внутри одного или нескольких контейнеров. Сетки на Bootstrap могут использоваться отдельно, без Bootstrap JavaScript и других CSS компонентов. Достаточно скачать и сослаться на "bootstrap-grid.css", который включает в себя flexbox классы и классы для сетки.

Bootstrap включает в себя изменяемую, адаптированную под мобильные устройства, масштабируемую до 12 колонок, модульную сетку, которая может подстраиваться под область просмотра.

Модульная сетка используется для создания макета страниц с помощью строк и столбцов, в которой можно размещать содержимое. Основные правила работы модульной сетки:

- Строки должны быть размещены внутри фиксированного контейнера (container) или резинового контейнера (container fluid) для правильного выравнивания и заполнения.
- Для создания горизонтальных групп столбцов используются строки (row).
- Расстояния между колонками задаются с помощью padding.
- Столбцы в модульной сетке создаются с указанием всех 12 доступных столбцов.
- Если разместить более 12 колонок в одной строке, то каждая группа дополнительных столбцов будет единым целым переноситься на новую строку.

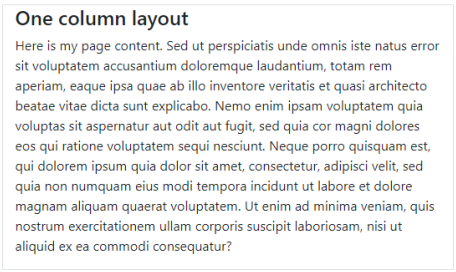
Правила сетки:

- колонки должны быть прямыми потомками Row;
- Row используются только для того, чтобы включать в себя колонки и не для ничего больше;
- Row должны быть помещены внутри контейнера.

Вот самый простой пример применения сетки:

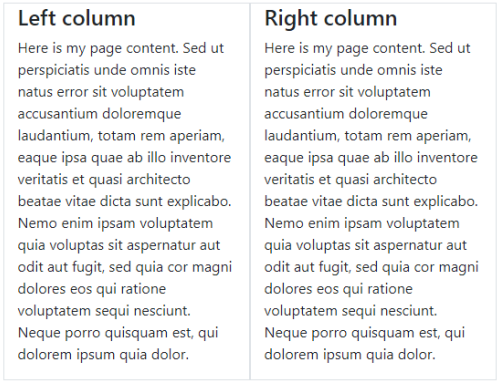
```
<div class="container">
<div class="row">
  <div class="col"> Это контент внутри сетки!</div>
 </div>
</div>
```

Этот код выдаст нам одну большую колонку на всю ширину вьюпорта.



А вот уже две колонки:

```
<div class="container">
  <div class="row">
   <div class="col">Left column</div>
   <div class="col">Right column</div>
  </div>
</div>
```



Контейнер

Контейнер это ключевой элемент сетки в Bootstrap. Контейнер может использоваться для хранения любых элементов и самого контента. Он используется не только для строк и колонок сетки. Для примера, вот идеальная и правильная разметка Bootstrap:

```
<div class="container">
<h2>Заголовок</h2>
<div class="row">    <div class="col"> Это контент внутри сетки! </div>  </div>
</div>
```

С первого взгляда, контейнер может показаться пустяковым и не сильно уж нужным, но он очень важен для контроля ширины шаблона. Контейнер также используется для равномерного выравнивания граней шаблона внутри вьюпорта браузера. Контейнер используется, чтобы противодействовать отрицательным внешним отступам row.

У Bootstrap 4 есть 2 типа контейнера. В примерах использованы

.container, но также есть и полноэкранный .container-fluid. Вы можете использовать любой из них:

1 – Контейнер с фиксированной шириной, для центровки контейнера по середине шаблона.

** **

2 – Контейнер с шириной во весь экран.

** **

.container масштабируется адаптивно по ширине экрана, так что в конце концов он может стать шириной на весь экран, как и .container-fluid, но на маленьких устройствах.

Применяя сетку, более одной строки может быть помещено внутри контейнера. Вы можете иметь их сколько угодно в самом контейнере и вы также можете иметь сколько хотите контейнеров на странице. Все зависит от того, какой шаблон вы пытаетесь сверстать.

У строк (rows) есть отрицательные левые/правые внешние отступы в -15px. Внутренний отступ контейнера в 15px используется для пресечения срабатывания отрицательных внешних отступов в строке контейнера. Это делается для равномерного выравнивания по граням в шаблоне. Если вы не поместите строку (row) в контейнер, то она будет выходить за пределы своего контейнера, вызывая нежелательные горизонтальные прокрутки.

Строки (Rows) и Колонки (Columns)

Строка – это группа колонок (Columns). Это потому, что колонки внутри .row не всегда располагаются горизонтально вдоль вьюпорта. Иногда нам надо, чтобы колонки в шаблоне были горизотальны, а иногда нам надо, чтобы они располагались вертикально. Концепция горизонтального vs. Вертикального шаблона является сущностью адаптивного дизайна. Единственным предназначением “строки”, является содержание одной или более “колонки”.

Не вставляйте контент прямо в “строку”!

Контент размещается уже внутри “колонок”:

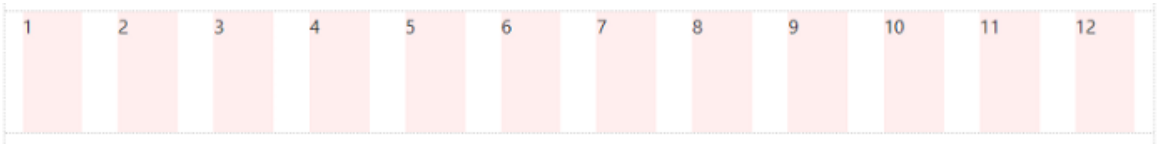
```
<div class="row">
  <div class="col">Happy :-) This is the right way.</div>
</div>
```

Так же очень важно упомянуть, что .row имеет display: flex. А как потомок в Flexbox, “колонка” в каждой строке одной и той же высоты.

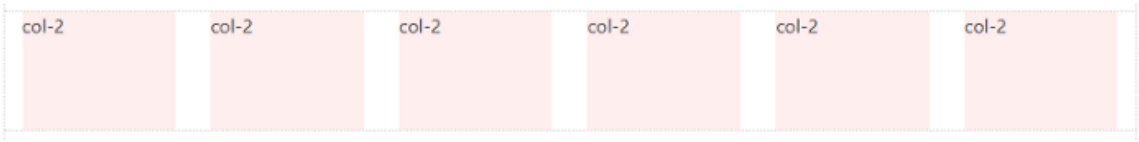
Всё что нужно знать о колонках Bootstrap

Колонки должны быть прямыми потомками строк. Колонки создают горизонтальные деления по вьюпорту. Пространство между колонками называется “gutter”.

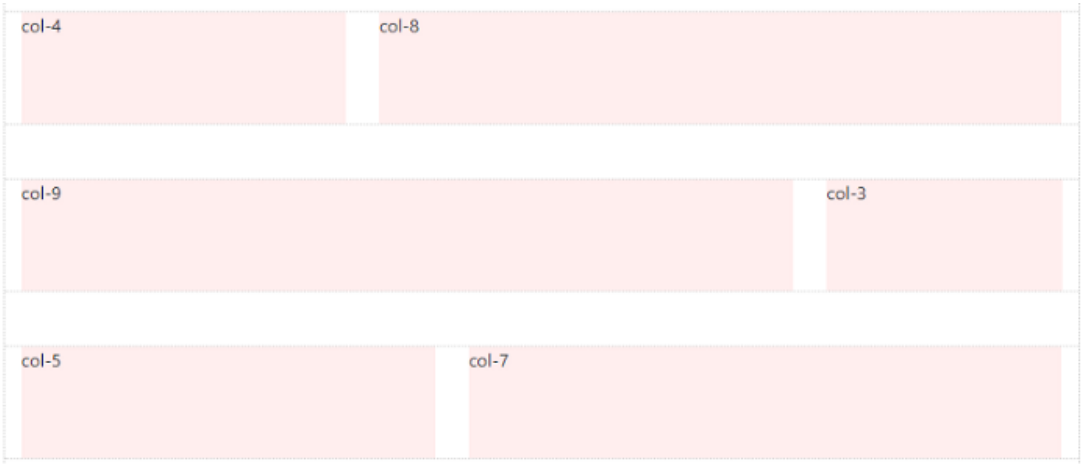
Классическая сетка Bootstrap имеет 12 колонок:



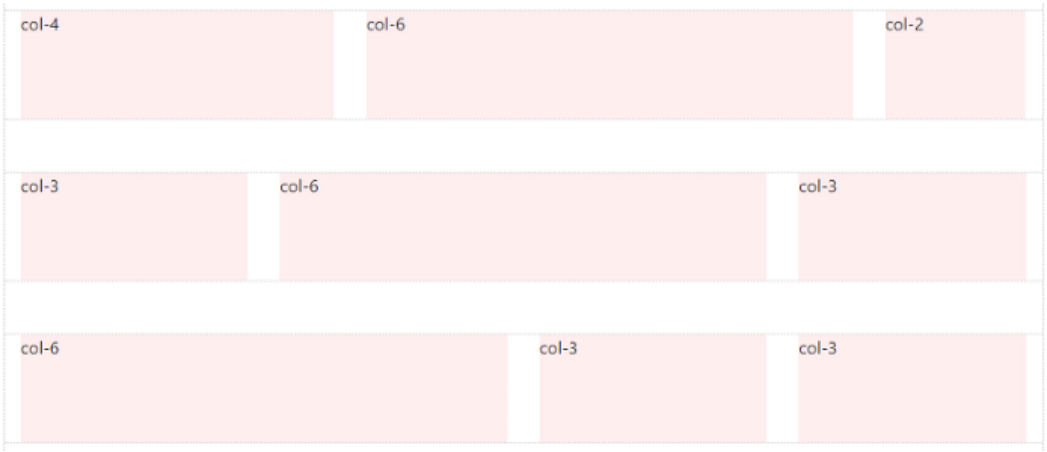
Таким образом, колонки могут быть равномерно разделены на 12 частей. Вот пример, 6 колонок (12/6=2):



Колонки могут быть разделены с использованием любой части из 12 элементов. И это нормально — использовать меньше 12. И также, это нормально — использовать больше 12-ти.



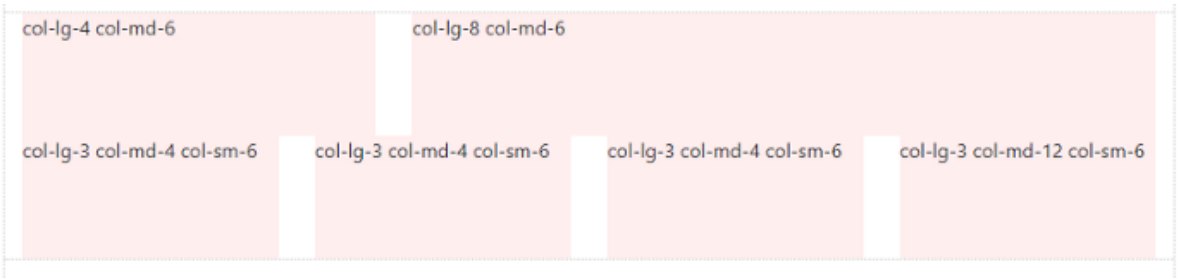
Учитывая такую гибкость, возможности шаблона кажутся бесконечными.



Колонки в одной строке располагаются вдоль горизонтально, а затем встают вертикально вниз. Это *вертикальное сталкивание* или “обертывание”, происходит когда элементы колонок в одном ряду превышают количество двенадцати. Этот процесс известен как “враппинг колонок”

Колонки в одной строке переходят на следующую строку, через каждые 12 элементов:

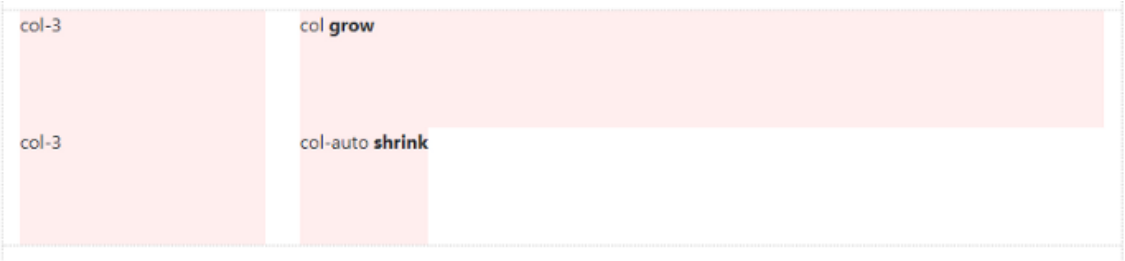
Ширина колонки и “обертывание” может контролироваться, используя разные ряды адаптивной сетки. (a.k.a “Grid Breakpoints”)



Колонки могут менять позицию относительно потомков в том же ряду:

Колонки могут содержать других потомков Rows & Columns. Это называется вложением:

Колонки могут “расти” и “урезаться” по ширине. Это auto-layout колонки:



Контрольные вопросы

1. Что такое Bootstrap?
2. Как подключить библиотеку Bootstrap?
3. Для чего используется модульная сетка в Bootstrap?
4. Для чего нужен контейнер?
5. Каковы правила при использовании модульной сетки?

ЛАБОРАТОРНАЯ РАБОТА №14

Работа с компонентами Bootstrap

Цель работы: Формирование умений использования различных компонент Bootstrap.

Задание на лабораторную работу

Включите в созданный на прошлой лабораторной работе следующие компоненты:

- Элементы HTML-форм.
- Панель навигации (в верхней части страницы).
- Выпадающие списки кнопок (могут быть использованы в панели навигации).
- Индикаторы прогресса.

Теоретическая часть

Десятки полезных компонентов встроены в Bootstrap для навигации, сообщений, загрузки информации и многого другого.

Примеры компонентов:

- Уведомления
- Значки
- Кнопки
- Карты
- Выпадающие элементы
- Формы
- Группа списков
- Модальное окно
- Навигация
- Навигационная панель
- Нумерация страниц
- и др.

Выпадающие элементы

Выпадающие элементы - это переключаемые, контекстные элементы поверхностного наложения для отображения списков ссылок и т.п. Они интерактивны благодаря плагину JavaScript в BS4. Функциональность toggle в выпадающих элементах запускается по клику, а не по наведению. Выпадающие элементы «построены» на сторонней библиотеке Popper.js, которая обеспечивает динамическое позиционирование и определение размера окна просмотра. Обязательно включите popper.min.js перед JavaScript Bootstrap или используйте bootstrap.bundle.min.js / bootstrap.bundle.js, который содержит Popper.js.

Стандарт WAI ARIA описывает и определяет виджет role="menu" как настоящий виджет, но лишь для меню «а-ля приложение», запускающих действия или функции. Меню ARIA могут содержать лишь пункты меню, чекбоксы, «радио-кнопки», группы «радио-кнопок» и подменю.

Выпадающие элементы Bootstrap, с другой стороны, спроектированы для решения множества задач и могут работать в разных структурах разметки. Например, можно создать выпадающие элементы, содержащие дополнительные поля ввода и элементы контроля форм, такие как поиск или поле ввода логина. По этой причине BS4 не «ожидает» (и не добавляет автоматически) ни один из атрибутов (role и aria-), необходимых для меню согласно стандарту ARIA.

Однако Bootstrap всегда добавляет встроенную поддержку для большинства стандартных взаимодействий меню и клавиатуры, таких как возможность двигаться через отдельные элементы класса

.dropdown-item кнопками курсора на клавиатуре и закрывать меню кнопкой ESC.

Примеры

Оберните «контролирующий» элемент (ссылку или кнопку) выпадающего элемента и выпадающее меню классом .dropdown или другим элементом с position: relative;. При необходимости выпадающие элементы можно запускать из элементов <a> или <button>.

Выпадающие элементы одинарных кнопок

Любую одинарную кнопку .btn можно превратить в «контролирующий» элемент (кнопка открытия\скрытия) при помощи некоторых изменений разметки. Вот как вы можете это сделать также и с элементами <button>:

```
<div class="dropdown">
 <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  Кнопка выпадающего списка
 </button>
 <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
 </div>
</div>
```

А вот так - с <a> элементами:

```
<div class="dropdown show">
 <a class="btn btn-secondary dropdown-toggle" href="#" role="button" id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

Выпадающая ссылка
 </a>

 <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
```

```
 <a class="dropdown-item" href="#">Something else here</a>
 </div>
</div>
```

Выпадающие элементы кнопок с разделенными зонами

По такому же принципу создавайте выпадающие элементы в кнопках с разделенными зонами, используя почти такую же разметку, как в пункте выше, но с добавлением класса .dropdown-toggle-split для правильного спейсинга вокруг выпадающего элемента.

Тут используется дополнительный класс, который уменьшает на 25% горизонтальный паддинг padding с обеих сторон выпадающей «каретки» и удаляет margin-left, добавленный для выпадающих элементов обычных кнопок. Эти изменения позволяют центрировать выпадающую «каретку» в разделенной кнопке и обеспечивают более подходящий размер «зоны клика» вблизи главной кнопки.

```
<!-- Example split danger button -->
<div class="btn-group">
 <button type="button" class="btn btn-danger">Action</button>
 <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  <span class="sr-only">Toggle Dropdown</span>
 </button>
 <div class="dropdown-menu">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">Separated link</a>
 </div>
</div>
```

«Выпадающий вверх»

Добавьте класс .dropup и выпадающий элемент будет «выпадать» вверх.

```
<!-- Default dropup button -->
<div class="btn-group dropup">
 <button type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
```

```
 Dropup
</button>
<div class="dropdown-menu">
 <!-- Dropdown menu links -->
</div>
</div>

<!-- Split dropup button -->
<div class="btn-group dropup">
 <button type="button" class="btn btn-secondary">
  Split dropup
 </button>
 <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  <span class="sr-only">Toggle Dropdown</span>
 </button>
 <div class="dropdown-menu">
  <!-- Dropdown menu links -->
 </div>
</div>
```

Добавьте класс .dropright и выпадающий элемент будет «выпадать» вправо. Добавьте класс .dropleft и выпадающий элемент будет «выпадать» влево.

Пункты меню

Исторически содержимым выпадающих элементов всегда были ссылки, но BS4 изменил это. Сейчас вы можете использовать в качестве содержимого выпадающих элементов <button>, а не только <a>.

```
<div class="dropdown">
 <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenu2" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  Dropdown
 </button>
 <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
  <button class="dropdown-item" type="button">Action</button>
  <button class="dropdown-item" type="button">Another
```

```
action</button>
    <button class="dropdown-item" type="button">Something else here</button>
  </div>
 </div>
```

Выравнивание меню

По умолчанию выпадающее меню автоматически расположено в 100% от вершины и на левой стороне родителя. Добавьте класс .dropdown-menu-right к элементу класса .dropdown-menu для выравнивания выпадающего меню по правой стороне.

Внимание! Выпадающие элементы позиционируются благодаря Popper.js (за исключением случаев, когда они содержатся в navbar).

Заголовки меню

Добавьте заголовок, чтобы обозначить секции действий любого выпадающего меню.

```
<div class="dropdown-menu">
 <h6 class="dropdown-header">Dropdown header</h6>
 <a class="dropdown-item" href="#">Action</a>
 <a class="dropdown-item" href="#">Another action</a>
</div>
```

Разделители меню

```
<div class="dropdown-menu">
 <a class="dropdown-item" href="#">Action</a>
 <a class="dropdown-item" href="#">Another action</a>
 <a class="dropdown-item" href="#">Something else here</a>
 <div class="dropdown-divider"></div>
 <a class="dropdown-item" href="#">Separated link</a>
</div>
```

Формы меню

Расположите форму внутри выпадающего меню, и используйте утилиты паддинга или марджина для ее уплотнения.

```
<div class="dropdown-menu">
 <form class="px-4 py-3">
  <div class="form-group">
```

```
        <label for="exampleDropdownFormEmail1">Email address</label>
        <input type="email" class="form-control"
id="exampleDropdownFormEmail1" placeholder="email@example.com">
       </div>
       <div class="form-group">
        <label for="exampleDropdownFormPassword1">Password</label>
        <input type="password" class="form-control"
id="exampleDropdownFormPassword1" placeholder="Password">
       </div>
       <div class="form-check">
        <input type="checkbox" class="form-check-input"
id="dropdownCheck">
        <label class="form-check-label" for="dropdownCheck">
         Remember me
        </label>
       </div>
       <button type="submit" class="btn btn-primary">Sign in</button>
      </form>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">New around here? Sign up</a>
      <a class="dropdown-item" href="#">Forgot password?</a>
     </div>

     <form class="dropdown-menu p-4">
      <div class="form-group">
       <label for="exampleDropdownFormEmail2">Email address</label>
       <input type="email" class="form-control"
id="exampleDropdownFormEmail2" placeholder="email@example.com">
      </div>
      <div class="form-group">
       <label for="exampleDropdownFormPassword2">Password</label>
       <input type="password" class="form-control"
id="exampleDropdownFormPassword2" placeholder="Password">
      </div>
      <div class="form-check">
       <input type="checkbox" class="form-check-input"
id="dropdownCheck2">
       <label class="form-check-label" for="dropdownCheck2">
        Remember me
       </label>
```

```
 </div>
 <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

Активные элементы меню

Добавьте класс .active к элементу выпадающего меню для его **стилизации как «активированного».**

```
<div class="dropdown-menu">
 <a class="dropdown-item" href="#">Regular link</a>
 <a class="dropdown-item active" href="#">Active link</a>
 <a class="dropdown-item" href="#">Another link</a>
</div>
```

Неактивные элементы меню

Добавьте класс .disabled к элементу выпадающего меню для его **стилизации как «деактивированного».**

```
<div class="dropdown-menu">
 <a class="dropdown-item" href="#">Regular link</a>
 <a class="dropdown-item disabled" href="#">Disabled link</a>
 <a class="dropdown-item" href="#">Another link</a>
</div>
```

Использование

Через атрибуты или JavaScript, плагин выпадающих элементов изменяет (показывает скрытоесодержимое) состояние содержимого (выпадающие меню) изменением класса .show элемента родительского списка. Атрибут data-toggle="dropdown" отвечает за закрытие выпадающих меню на уровне приложения, так что будет неплохой идеей всегда использовать данный атрибут.

Параметры

Передавайте параметры через JavaScript или атрибуты. Если используются атрибуты, добавляйте название параметра к data- как в data-offset="".

Название	Тип	По умолч.	Описание
offset	number | string | function	0	Смещение выпадающего элемента относительно его триггера.

Название	Тип	По умолч.	Описание
flip	boolean	true	Позволяет выпадающему элементу «перевернуться», если произошло перекрытие другого элемента.
boundary	string | element	'scrollParent'	Граница ограничения переполнения выпадающего меню. Принимает значения 'viewport', 'window', 'scrollParent' или ссылку HTMLElement (только для JavaScript).

Обратите внимание, что если для boundary установлено значение, отличное от 'scrollParent', позиция position: static применяется к контейнеру .dropdown.

Методы

Метод	Описание
$().dropdown('toggle')	Задействует поведение toggle в выпадающем меню данного навбара или при навигации «TAB»ом.
$().dropdown('update')	Обновляет позицию «выпадения» элемента.
$().dropdown('dispose')	Уничтожает выпадающий элемент.

События

Все события выпадающих элементов наступают в родительском элементе класса .dropdown-menu и несут свойство relatedTarget, значение которого равно элементу «якоря» (ссылка, т.е. <a>), запускающего функциональность toggle.

Событие	Описание
show.bs.dropdown	Это событие наступает немедленно по вызову экземпляра метода show.
shown.bs.dropdown	Это событие наступает, когда выпадающий элемент стал видимым юзеру (будет ждать завершения CSS-переходов).
hide.bs.dropdown	Это событие наступает немедленно по вызову экземпляра метода hide.
hidden.bs.dropdown	Это событие наступает, когда выпадающий элемент стал невидимым юзеру (будет ждать завершения CSS-переходов).

```
$('#myDropdown').on('show.bs.dropdown', function () {
 // do something…
})
```

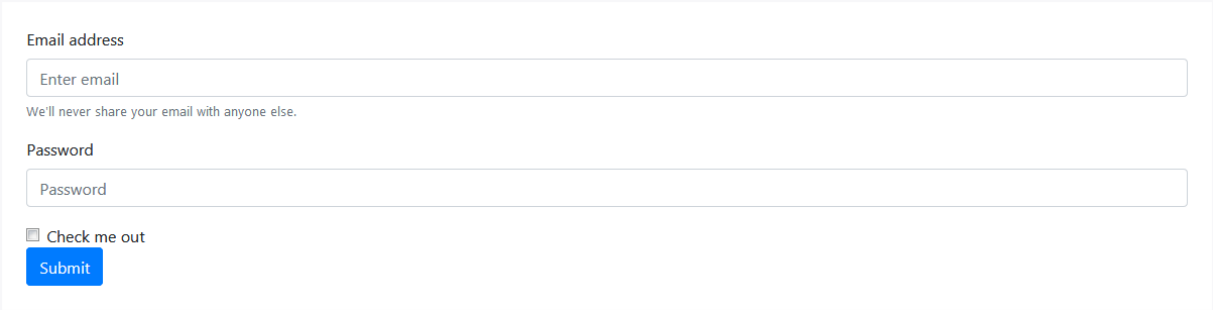
Формы

Формы в Bootstrap по сути являются просто расширением форм

HTML5 с добавлением классов.

Удостоверьтесь, что используете правильный атрибут type во всех формах ввода (т.е., email для почты и number для цифровой информации), это даст вам преимущества в виде новейших инструментов (таких как проверка email, выборка чисел и т.д.) контроля данных ввода.

Пример:




```
<form>
 <div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email" class="form-control" id="exampleInputEmail1"
aria-describedby="emailHelp" placeholder="Enter email">
  <small id="emailHelp" class="form-text text-muted">We'll never
share your email with anyone else.</small>
 </div>
 <div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password" class="form-control"
id="exampleInputPassword1" placeholder="Password">
 </div>
 <div class="form-check">
  <input type="checkbox" class="form-check-input"
id="exampleCheck1">
  <label class="form-check-label" for="exampleCheck1">Check me
out</label>
 </div>
 <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Инструменты контроля форм

Текстовые инструменты контроля – такие как <input>, <select> и <textarea> - стилизованы классом .form-control, который содержит

основные стили внешнего вида, активного состояния, размерности и т.д.

Для создания формы загрузки файлов замените .form-control на .form-control-file.

Для создания форм заданной высоты используются классы, такие как .form-control-lg и .form-control-sm.

Добавьте атрибут булеанова типа readonly в форму ввода для предотвращения возможности изменения значения ввода. Такие типы ввода выглядят светлее (как неактивные формы ввода), но сохраняют стандартный курсор.

Если в ваших формах вы хотите стилизовать элементы <input readonly> как простой текст, используйте класс .form-control-plaintext для удаления оформления форм по умолчанию и сохранения правильных отступов.

Чекбоксы и кнопки «радио»

Чекбоксы и «радио», существовавшие по умолчанию, теперь модернизированы единым для обоих классом .form-check, **цель которого – в улучшении их расположения и «поведения» их элементов HTML**. Чекбоксы существуют для выбора одного или нескольких параметров из списка, а кнопки «радио» - одного.

Неактивные состояния чекбоксоы и "радио" поддерживаются, но для придания курсору функциональности not-allowed по наведению на родительский <label> вам потребуется добавить в .form-check-input атрибут disabled. Атрибут disabled будет применять более светлый цвет, чтобы указать состояние ввода.

По умолчанию, любое количество идущих один за другим чекбоксов и «радио» кнопок будет располагаться сверху вниз, а класс .form-check правильно отрегулирует пространство между ними.

Группируйте чекбоксы или «радио» кнопки по одной горизонтальной линии, добавив класс .form-check-inline в любой элемент класса .form-check.

Добавьте класс .position-static в формы ввода, которые находятся внутри элемента класса .form-check и не имеют какого-либо пояснительного текста. Не забудьте добавить одну из форм «лейбла» для вспомогательных технологий (например, использовав aria-label).

Расположение

Т.к. Bootstrap применяет display: block и width: 100% почти ко всем органам контроля форм, формы по умолчанию будут выстраиваться

вертикально. Дополнительные классы можно использовать для создания вариаций расположения каждой отдельной формы.

Группы форм

Класс .form-group – самый простой путь придания формам некой структуры. Его единственная цель – создание вокруг надписи margin-bottom и включение контроля. В качестве приятного дополнения: поскольку это обычный класс, его можно использовать с <fieldset>, или практически любым прочим элементом.

Сетка форм

Используйте их для расположения форм, которые требуют множественных колонок, различной ширины и других дополнительных параметров выравнивания.

Вы также можете заменить .row на класс .form-row, который есть разновидность нашего стандартного ряда сетки, который обладает возможностью «перебить» стандартно установленные расстояния между колонками и делает колонки более компактными.

Создайте горизонтальные формы с помощью сеток, добавив класс .row к группам форм и используя классы .col-*-* для задания ширины ваших надписей и элементов контроля. Обязательно добавьте класс .col-form-label также и в ваши <label> для того, чтобы они приобрели вертикальное центрирование относительно связанных с ними элементов контроля форм.

Временами вам может понадобиться классы марджина или паддинга, чтобы создать классное выравнивание. Например, мы удалили padding-top в наших вертикально расположенных лейблах ввода "радио", для лучшего выравнивания текста.

Обязательно используйте классы .col-form-label-sm или .col-form-label-lg в своих <label> для того, чтобы размеры шрифтов названия формы и вспомогательной надписи в пустой форме (т.н. placeholder) ввода совпадали.

Вложенные формы

Используйте класс .form-inline для отображения серии лейблов, органов контроля форм и кнопок на одном горизонтальном ряду. Органы контроля форм внутри строчных форм могут слегка отличаться от своего состояния по умолчанию.

- Органы контроля имеют display: flex, что скрывает пустое пространство в HTML и позволяет вам обеспечить контроль над

выравниванием с помощью утилит спейсинга и флекбокса.

- Органы контроля и группы ввода имеют width: auto, который «перебивает» умолчание Bootstrap width: 100%.
- На **зонах видимости от 576px** органы контроля отображаются как строчные элементы для удобства отображения на узких зонах просмотра.

Вам может понадобиться вручную задать ширину и выравнивание каждого элемента контроля форм, используя утилиты спейсинга (как показано ниже).Всегда включайте <label> в каждый элемент контроля, даже если вы хотите спрятать его с помощью класса .sr-only от читателей, которые не используют экранные читалки.

Индикаторы процесса

Компоненты прогрессбаров созданы из двух элементов HTML, CSS для задания ширины и нескольких атрибутов. Если не использовать элемент HTML5 <progress>, это обеспечивает возможность вертикального расположения прогрессбаров, возможность их анимации и размещения текстовых лейблов над ними.

- используем класс .progress как обертку для индикации максимального значения прогрессбара.
- используем внутренний класс .progress-bar для индикации пройденного прогресса.
- класс .progress-bar требует оформления себя как строчного элемента, обычного класса или CSS для задания своей ширины.
- класс .progress-bar также требует атрибутся role и aria, чтобы стать открытыми к взаимодействию с вспомогательными технологиями.

Все это воплощено в примерах ниже.

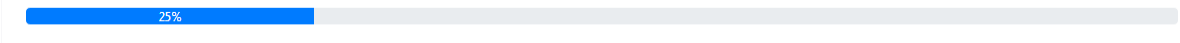
```
<div class="progress">
 <div class="progress-bar" role="progressbar" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar" role="progressbar" style="width: 25%" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
```

```
<div class="progress-bar" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
<div class="progress-bar" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
<div class="progress-bar" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Bootstrap имеет немало утилит для задания ширины. В зависимости от ваших нужд они могут помочь и с быстрой настройкой прогрессбара.

Лейблы

Добавляйте лейблы (т.е. инфо типа цифр) в ваши прогрессбары, размещая текст внутри класса .progress-bar.




```
<div class="progress">
<div class="progress-bar" role="progressbar" style="width: 25%;" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">25%</div>
</div>
```

Высота

В классе .progress задается лишь атрибут height, так что если вы измените это значение, внутренний класс .progress-bar автоматически изменит свой размер соответственно.

```
<div class="progress" style="height: 1px;">
<div class="progress-bar" role="progressbar" style="width: 25%;" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress" style="height: 20px;">
<div class="progress-bar" role="progressbar" style="width: 25%;" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

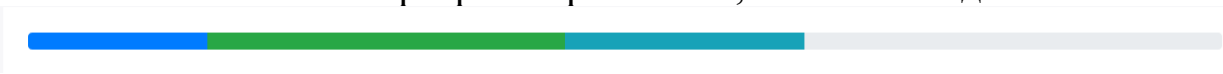
Фон

Используйте обычные классы фона для изменения внешнего вида отдельных полос прогрессбара.

```
<div class="progress">
 <div class="progress-bar bg-success" role="progressbar" style="width: 25%" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar bg-info" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar bg-warning" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar bg-danger" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Множественные полосы

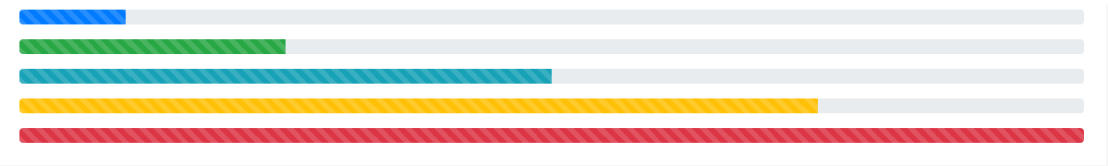
Включайте в свой прогрессбар таковые, если необходимо.



```
<div class="progress">
 <div class="progress-bar" role="progressbar" style="width: 15%" aria-valuenow="15" aria-valuemin="0" aria-valuemax="100"></div>
 <div class="progress-bar bg-success" role="progressbar" style="width: 30%" aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"></div>
 <div class="progress-bar bg-info" role="progressbar" style="width: 20%" aria-valuenow="20" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Полосатые

Добавьте класс .progress-bar-striped к любому элементу класса .progress-bar, чтобы добавить градиентные полосы на CSS к фоновому цвету прогрессбара.



```
<div class="progress">
 <div class="progress-bar progress-bar-striped" role="progressbar" style="width: 10%" aria-valuenow="10" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-striped bg-success" role="progressbar" style="width: 25%" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-striped bg-info" role="progressbar" style="width: 50%" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-striped bg-warning" role="progressbar" style="width: 75%" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100"></div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-striped bg-danger" role="progressbar" style="width: 100%" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100"></div>
</div>
```

Анимированные полосы

Добавьте класс .progress-bar-animated к элементу класса .progress-bar для анимации полосок справа налево анимациями CSS3.

```
<div class="progress">
 <div class="progress-bar progress-bar-striped progress-bar-animated" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100" style="width: 75%"></div>
</div>
```

Навигация

Навигация в Bootstrap имеет общую для подобного типа элементов разметку, от базового класса .nav до активных и «выключенных»

состояний. Заменяйте классы-модификаторы для переключения между стилями.

Базовый компонент класса .nav создан на флексбоксе, что обеспечивает хорошую базу для создания всех типов навигационных компонентов. В него входят несколько стилей, которые «перебивают» остальные (для работы со списками), добавлены паддинги ссылок для увеличения «зоны клика», и базовые стили «выключенных» состояний.

В базовый компонент класса .nav не включено какое-либо состояние .active. Следующие примеры включают класс, главным образом для демонстрации, что данный конкретный класс не подключает какие-либо специальные стили.

Классы используются везде, так что ваша разметка может быть сверх-гибкой. Используйте <ul> как показано выше, или создайте свои; скажем, на основе элемента <nav>. Из-за того, что .nav использует display: flex, ссылки в навигационной панели ведут себя, как если бы были элементами такой же панели, но с меньшим количеством кода.

```
<nav class="nav">
 <a class="nav-link active" href="#">Active</a>
 <a class="nav-link" href="#">Link</a>
 <a class="nav-link" href="#">Link</a>
 <a class="nav-link disabled" href="#">Disabled</a>
</nav>
```

Навигационная панель

Вот что вам надо знать перед использованием навбара:

- Навбары требуют «обертки» из классов .navbar и .navbar-expand{-sm|-md|-lg|-xl} для отзывчивости при «складывании»и классы , а также классы цветовых схем.
- Навбары и их содержимое по умолчанию занимают 100% ширины. Используйте опциональные контейнеры для ограничения их горизонтальной ширины.
- Используйте классы спейсинга и «флекс» для контроля над пространством и выравниванием внутри навбаров.
- Навбары отзывчивы по умолчанию, но вы можете легко изменить это. Отзывчивое поведение зависит от нашего «плагина свертывания» JavaScript.
- Навбары скрыты по умолчанию при печати. Сделайте их печатаемыми, добавив класс .d-print в .navbar. Смотри класс отображения.
- Придайте им доступность использованием элемента <nav>, или,

если используется менее специфический элемент – например : добавьте role="navigation" в каждый навбар для придания ему большей доступности для пользователей вспомогательных технологий.

Дальше вы увидите примеры и список поддерживаемых под-компонентов.

Поддерживаемые типы содержимого

В навбарах присутствуют встроенная поддержка многих суб-компонентов. Выбирайте нужный:

- .navbar-brand для названия вашей компании, продукта или имени проекта.
- .navbar-nav для навигации полной высоты (включая выпадающие элементы).
- .navbar-toggler для использования с нашим JS-«плагином свертывания» и других изменяющихся состояний навигации.
- .form-inline для любых органов контроля форм и действий с ними.
- .navbar-text для добавления вертикально центрированных строк текста.
- .collapse.navbar-collapse для группирования и скрытия содержимого навбара на определенном брейкпойнте родителя.

Контрольные вопросы

1. Какие в Bootstrap есть компоненты?
2. Как создать навигационную панель?
3. Каким образом можно создать меню с помощью Bootstrap?
4. Как создать форму в Bootstrap?
5. Какой базовый класс используется для создания навигации?
6. Какого вида можно создавать индикаторы прогресса?

ЛАБОРАТОРНАЯ РАБОТА №15

Основы создания адаптивного веб-сайта в Bootstrap

Цель работы: Формирование умений создания адаптивного веб-сайта.

Задание на лабораторную работу

Создайте адаптивные страницы из предыдущих лабораторных работ в трех вариантах: для смартфонов, планшетов и ноутбуков.

Теоретическая часть

Различают следующие основные типы макетов сайтов, связанных с шириной:

- фиксированный;
- резиновый (гибкий);
- адаптивный.

Адаптивный макет сайта - это макет, который может «приспосабливаться» под различные устройства (ширину рабочей области окна браузера). Т.е. на одних устройствах он может иметь одну структуру, а на других - другую.

Пример адаптивного макета, состоящего из 2 блоков, который на разных устройствах выглядит по-разному (рис.13):

- на смартфонах и планшетах (устройствах с очень маленьким размером экрана) блоки должны располагаться вертикально, т.е. один под другим;
- на ноутбуках (устройствах со средним размером экрана) блоки должны располагаться горизонтально (1 блок - 33.3%, 2 блок - 66.7%);
- на десктопах (устройствах с большим размером экрана) тоже горизонтально, но с другими размерами (1 блок - 25%, 2 блок - 75%).

```
    <style>
body {
 margin: 0;
}
.container {
 display: flex;
 flex-wrap: wrap;
}
.aside, .main {
 width: 100%;
 min-height: 1px;
}
@media (min-width: 992px) {
 .aside {
```

```
 flex: 0 0 33.333333%;
 max-width: 33.333333%;
 order: 1;
 }
 .main {
 flex: 0 0 66.666667%;
 max-width: 66.666667%;
 order: 2;
 }
}
@media (min-width: 1400px) {
 .aside {
 flex: 0 0 25%;
 max-width: 25%;
 }
 .main {
 flex: 0 0 75%;
 max-width: 75%;
 }
}
</style>
```

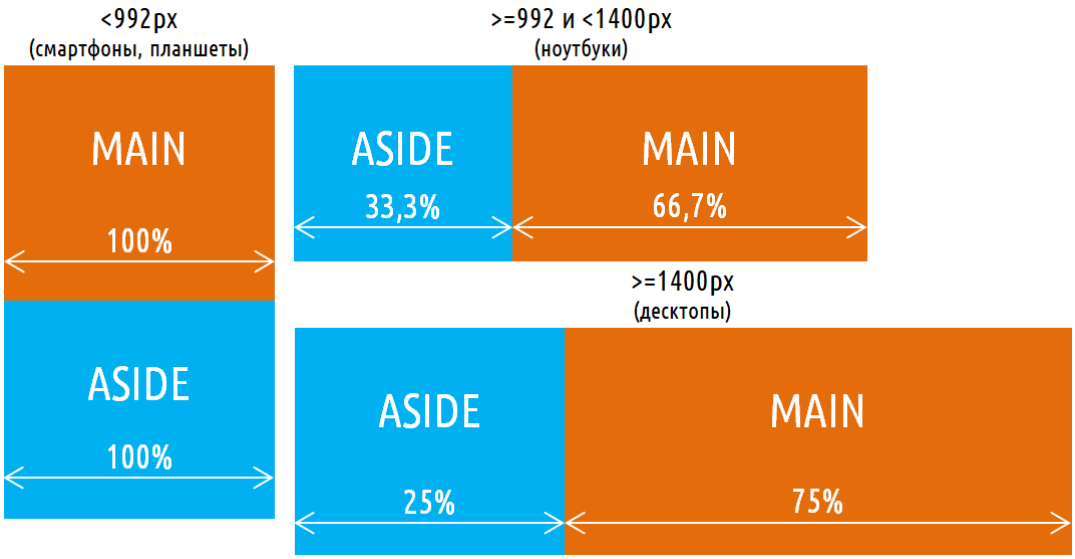


Рис. 13. Адаптивный макет

```
<div class="container">
 <main class="main">
 MAIN
 </main>
```

```
 <aside class="aside">
  ASIDE
 </aside>
</div>
```

Адаптивный дизайн

Все компоненты Bootstrap являются гибкими, а не которые из них ещё и адаптивными как, например, Navbar. Данный компонент (Navbar) может изменять своё представление, т.е. находится в мобильном или десктопном представлении в зависимости от того какую в данный момент viewport ширину имеет браузер.

Сетка не всегда может быть из 12 элементов. Спасибо flexbox, у Bootstrap 4 есть новые “auto-layout” колонки. Такие безразмерные колонки дают вам больше гибкости при разработке шаблонов.

Как вы видели выше, колонки могут быть разной ширины:

Ширина колонки может изменяться в зависимости от ширины экрана. Это называется адаптивным дизайном

В Bootstrap 4 есть 5 адаптивных рядов (ну или брейкпоинтов), которые вы возможно заметили в предыдущем примере. (ie; col-lg-4, col-md).

Адаптивные брейкпоинты, основаны на ширине экрана:

(xs) — ширина экрана < 576px. Это стандарт.

sm — ширина экрана ≥ 576px

md — ширина экрана ≥ 768px

lg — ширина экрана ≥ 992px

xl — ширина экрана ≥ 1200px

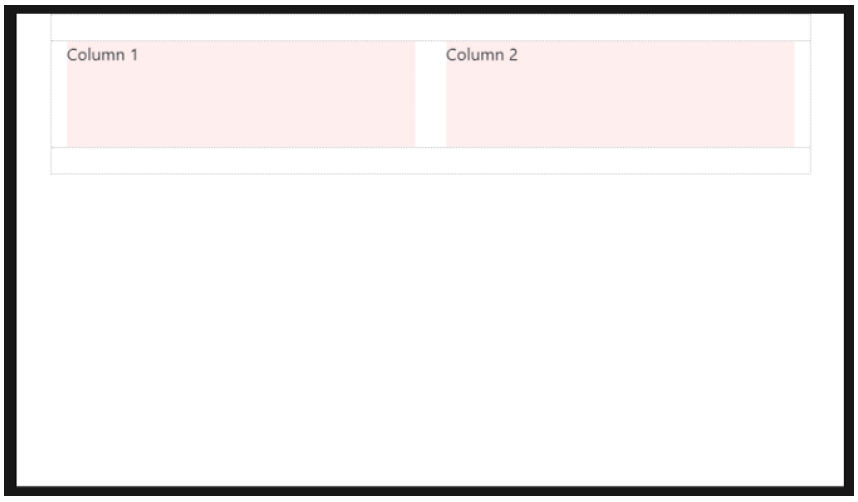
Так как xs это дефолтное прерывание, -xs инфикс, который использовался в Bootstrap 3, больше не используется в Bootstrap 4. Так что вместо col-xs-6, просто col-6.

Bootstrap использует медиа запросы из CSS, что установить адаптивные точки прерываний. Они дают вам возможность контролировать поведение колонок при разных размерах экрана.

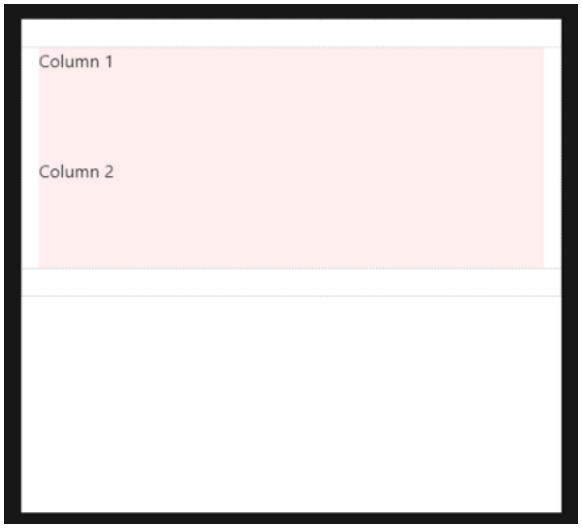
Для примера: вот 2 колонки, каждая шириной 50%:

```
<div class="container">
  <div class="row">
   <div class="col-sm-6">Column 1</div>
   <div class="col-sm-6">Column 2</div>
  </div>
</div>
```

col-sm-6 означает использование 6 колонок из 12, то есть 50% ширины на типичных маленьких размерах экранов. Которые больше или равны 768px:



На экранах меньше, чем 768px, 2 колонки станут шириной 100% и встанут вертикально:



Это происходит, потому что (xs) по дефолту или подразумевает брейкпоинт. Так как я не указывал дефолтную ширину колонки, 50% ширина была применена только на 768px и шире для sm брейкпоинта.

Mobile-first!

Так как (xs) это дефолтный брейкпоинт, то подразумевает col-12. Следовательно:

Column

Работает так же, как и:

Column

xs(default) > переписывает sm > переписывает md > переписывает lg > переписывает xl

Или в обратном порядке… xl > переписывает lg > переписывает md > переписывает sm > переписывает (xs)

Следовательно, col-sm-6 реально означает 50% ширины на мелких и выше дисплеях. Для одинаковой ширины колонок на всех брейкпоинтах, просто выставите ширину для самого маленького ряда как хотите:

.. тоже самое, что и:

..

Для разной ширины колонки на разных уровнях, используйте подходящие брейкпоинты для перезаписи мелких прерываний. Для примера, 3 колоночная ширина на sm и 4 колоночная ширина на md и выше:

..

Auto-layout колонки в Bootstrap 4 также работают адаптивно. Из-за их простоты, теперь я предпочитаю использовать их, а не классические элементы 12 колонок. Auto-layout колонки идеальны для любых сценариев шаблонов, где необходима равная ширина колонок. Но не забудьте, что 12-ти колоночные юниты могут быть смешаны при необходимости.

Давайте посмотрим на auto-layout сетку

Итак, 3 равные колонки. ‘col’ остаются горизонтальным на всей широте и не встают вертикально, так как xs прерывание дефолтно:

```
<div class="container">
  <div class="row">
   <div class="col">1</div>
   <div class="col">2</div>
   <div class="col">3</div>
  </div>
</div>
```

3 равные колонки, адаптивные. В этом примере, ‘col’ остаются

горизонтальным до прерывания sm на 576px, а затем они становятся вертикальными. Помните, что вы можете заменять sm на каком угодно брейкпоинте(md,lg,xl), если нужно:

```
<div class="container">
  <div class="row">
   <div class="col-sm">1</div>
   <div class="col-sm">2</div>
   <div class="col-sm">3</div>
  </div>
</div>
```

2 колонки, левый сайдбар. А вот пример комбинирования классически определнной ширины колонок с колонками auto-layout. Правая колонка будет автоматически расти, чтобы занять ширину, так как мы используем auto-layout .col. Сайдбар будет 16.6% ширины на больших экранах и затем встанет над контентом при sm брейкпоинте с 576px:

```
<div class="container">
  <div class="row">
   <div class="col-sm-2">sidebar</div>
   <div class="col">main content</div>
  </div>
</div>
```

3 колонки, правый сайдбар (сокращение, чтобы уместиться): В этом примере есть левый сайдбар, центральная область контента и правый сайдбар, который сокращается по ширине, чтобы подстроиться под свой контент.

```
<div class="container">
  <div class="row">
   <div class="col-sm-2">left</div>
   <div class="col">main content</div>
   <div class="col-auto">right</div>
  </div>
</div>
```

Ключевые моменты адаптивного дизайна используемые в сетке Bootstrap 4:

Колонки встанут вертикально и станут шириной во весь экран на устройствах с маленьким разрешением, если вы не используете col-*

класс в HTML разметке. Используйте col-* для предотвращения такого вертикального выстраивания.

Классы сеток поменьше, также применяются на больших экранах, пока не перепишутся конкретно под ширину большего экрана. Следовательно, в сущности тоже самое, что и . Следовательно, вам только надо использовать класс для самых маленьких разрешений, которые вам нужно поддерживать.

Строки (row) —имеют display: flex и следовательно колонки имеют равную высоту в одном и том же ряду. Используйте auto-margin или Flexbox align-item и justify-content для горизонтального или вертикального выравнивания.

Создание адаптивного макета с помощью Bootstrap 3

Создания адаптивного макета в Bootstrap 3 осуществляется под различные устройства. По умолчанию в Bootstrap 3 проектирование выполняется под 4 контрольные точки (xs, sm, md и lg). Область контрольной точки xs – это смартфоны, sm – планшеты, md – ноутбуки, а lg – десктопы.

Для примера сверстаем с помощью сетки Bootstrap 3 макет страницы, изображенный на рис.14. В качестве вида макета выберем, например, адаптивно-гибкий.

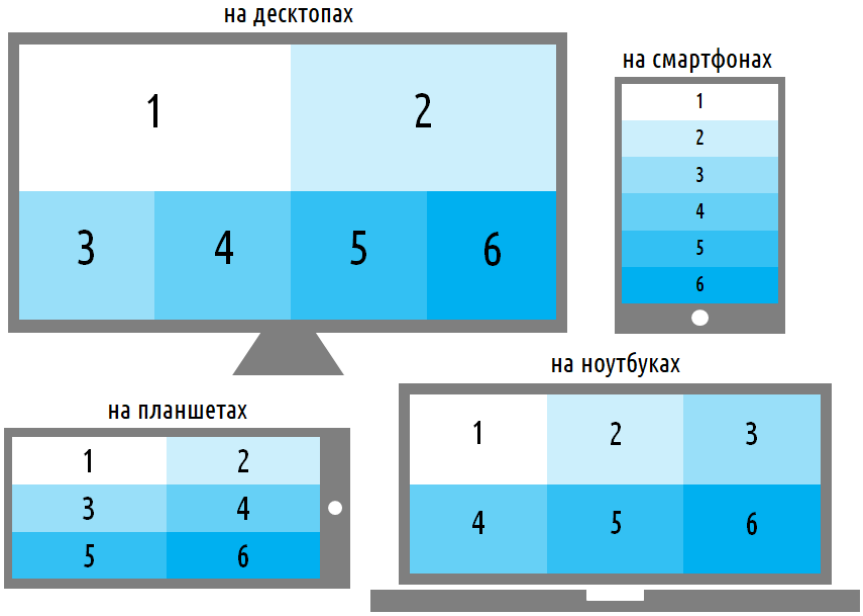


Рис. 14. Пример макета страницы

Разработку адаптивного макета обычно начинают с самых маленьких устройств (смартфонов), по отношению к Bootstrap 3 – это область xs.

На xs блоки должны располагаться вертикально и иметь ширину, равную ширине родительского контейнера (т.е. 12 колонок Bootstrap).

```
<div class="container-fluid">
 <div class="row">
  <div class="col-xs-12">1</div>
  <div class="col-xs-12">2</div>
  <div class="col-xs-12">3</div>
  <div class="col-xs-12">4</div>
  <div class="col-xs-12">5</div>
  <div class="col-xs-12">6</div>
 </div>
</div>
```

На sm блоки должны располагаться на 3 строках по 2 блока в каждой строке. Каждый блок должен иметь ширину, равную 50% ширины родительского элемента (6 колонок Bootstrap).

```
<div class="container-fluid">
 <div class="row">
  <div class="col-xs-12 col-sm-6">1</div>
  <div class="col-xs-12 col-sm-6">2</div>
  <!--Перед блоком sm, который должен начинаться с новой строки-
->
  <div class="clearfix visible-sm-block"></div>
  <div class="col-xs-12 col-sm-6">3</div>
  <div class="col-xs-12 col-sm-6">4</div>
  <!--Перед блоком sm, который должен начинаться с новой строки-
->
  <div class="clearfix visible-sm-block"></div>
  <div class="col-xs-12 col-sm-6">5</div>
  <div class="col-xs-12 col-sm-6">6</div>
 </div>
</div>
```

На md блоки должны располагаться на 2 строках по 3 блока в каждой строке. Каждый блок должен иметь ширину, равную 33.3% ширины родительского элемента (4 колонки Bootstrap).

```
<div class="container-fluid">
 <div class="row">
```

```
      <div class="col-xs-12 col-sm-6 col-md-4">1</div>
      <div class="col-xs-12 col-sm-6 col-md-4">2</div>
      <!--Перед блоком sm, который должен начинаться с новой строки-
->
      <div class="clearfix visible-sm-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4">3</div>
      <!--Перед блоком md, который должен начинаться с новой строки-
->
      <div class="clearfix visible-md-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4">4</div>
      <!--Перед блоком sm, который должен начинаться с новой строки-
->
      <div class="clearfix visible-sm-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4">5</div>
      <div class="col-xs-12 col-sm-6 col-md-4">6</div>
     </div>
    </div>
```

На lg блоки должны располагаться на 2 строчках. На первой строчке 2 блока, а на второй – 4 блока.

```
    <div class="container-fluid">
     <div class="row">
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-6">1</div>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-6">2</div>
      <!--Перед блоком sm и lg, который должен начинаться с новой
строки-->
      <div class="clearfix visible-sm-block visible-lg-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">3</div>
      <!--Перед блоком md, который должен начинаться с новой строки-
->
      <div class="clearfix visible-md-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">4</div>
      <!--Перед блоком sm, который должен начинаться с новой строки-
->
      <div class="clearfix visible-sm-block"></div>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">5</div>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">6</div>
     </div>
    </div>
```

Оптимизируем код, уберем, где возможно классы col-*-12, т.к.

адаптивные блоки Bootstrap по умолчанию занимают ширину, равную 100%.

```
<div class="container-fluid">
 <div class="row">
  <div class="col-sm-6 col-md-4 col-lg-6">1</div>
  <div class="col-sm-6 col-md-4 col-lg-6">2</div>
  <!--Перед блоком sm и lg, который должен начинаться с новой строки-->
  <div class="clearfix visible-sm-block visible-lg-block"></div>
  <div class="col-sm-6 col-md-4 col-lg-3">3</div>
  <!--Перед блоком md, который должен начинаться с новой строки-->
  <div class="clearfix visible-md-block"></div>
  <div class="col-sm-6 col-md-4 col-lg-3">4</div>
  <!--Перед блоком sm, который должен начинаться с новой строки-->
  <div class="clearfix visible-sm-block"></div>
  <div class="col-sm-6 col-md-4 col-lg-3">5</div>
  <div class="col-sm-6 col-md-4 col-lg-3">6</div>
 </div>
</div>
```

Создание адаптивного макета с помощью Bootstrap 4

Для примера сверстаем следующий макет (рис. 15).

```
<section class="a">
 <div class="container">
  <div class="row">
   <div class="a1 col-lg-6">A1</div>
   <div class="a2 col-lg-6">A2</div>
  </div>
 </div>
</section>

<section class="b">
 <div class="container">
  B
 </div>
</section>
```

```
<section class="c">
 <div class="container">
  <div class="row">
   <div class="c1 col-sm-6 col-lg-4">C1</div>
   <div class="c2 col-sm-6 col-lg-4">C2</div>
   <div class="c3 col-lg-4">C3</div>
  </div>
 </div>
</section>

<section class="d">
 <div class="container">
  <div class="row">
   <div class="d1 col-sm-6 col-lg-3">D1</div>
   <div class="d2 col-sm-6 col-lg-3">D2</div>
   <div class="d3 col-sm-6 col-lg-3">D3</div>
   <div class="d4 col-sm-6 col-lg-3">D4</div>
  </div>
 </div>
</section>
```

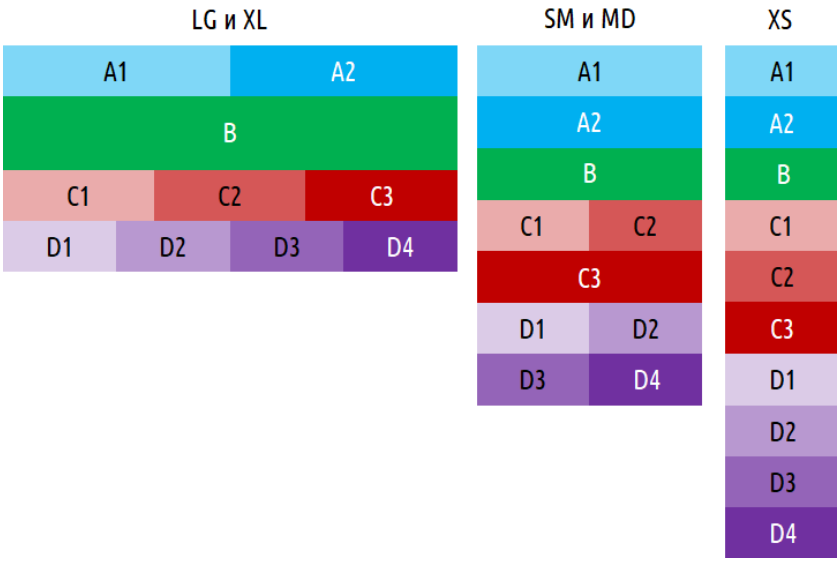


Рис. 15. Пример адаптивного макета

Контрольные вопросы

1. Какие виды дизайна веб-страниц существуют?
2. Что такое адаптивный дизайн?
3. Чем может отличаться макет страницы для смартфонов, планшетов и десктопов?

Список использованной литературы:

1. Левин А. Самоучитель КГ. Adobe Photoshop, включая Adobe Photoshop CS4. М. - 2009
2. Жарков В.А. Компьютерная графика, мультимедиа и игры на Visual C# 2005. М. - 2005
3. Rixsiboyev T. Kompyuter grafikasi. Ўқув қўлланма/2009
4. Методические указания к выполнению лабораторных работ по предмету «Мультимедиа технологии». Сост. Бабина В.Г., ТашИИЖТ, 2007 – 115 с.: ил.
5. Чумаченко И.Н «Corel Draw12» 2-издание Москва-2005. НТ Пресс
6. Ю.Гурский, И.Гурская, А.Жвалевский «Трюки и эффекты Corel Draw12» Питер-2005

Содержание:

Бабина Виктория Геннадьевн,
Гулямов Жавлон Нуруллаевич

МУЛЬТИМЕДИА СИСТЕМЫ И ТЕХНОЛОГИИ

(2-часть)

Редактор: С.А.Мулламухамедов
Технический редактор и верстка: М.Х.Ташбаева

Подписано в печать: 12.03.2021 г.
Формат бумаги 60×84/16. Объем 6 п.л.
Тираж 8 экз. Заказ №15-4/2019
Распечатано в типографии ТГТУ
г.Ташкент, ул. Темирйулчилар, 1

Ташкентский государственный транспортный университет, 2021 г.