# WEB-DESIGN

## 1. Введение

Веб-сайт состоит из 2 частей:
1) FrontEnd (фактически, интерфейс);
2) BackEnd – набор программ, обрабатывающих пользовательские данные.

Как правило, BackEnd размещается на стороне сервера, хотя некоторые программы могут скачиваться при открытии сайта на компьютер пользователя и выполняться на стороне клиента. FronEnd однозначно скачивается на компьютер пользователя и отображается в окне браузера. Это то, что пользователь видит на экране.

BackEnd – набор программ, написанных на языках программирования (чаще всего на php и JavaScript), он условно говоря, невидим для пользователя (фронт-енд является интерфейсом этих программ) и к веб-дизайну никакого отношения не имеет.

FrontEnd выполняет роль интерфейса. Однако, в отличие интерфейсов от прикладных программ, устанавливаемых в операционную систему пользователя, к "интерфейсам" веб-сайтов предъявляются повышенные требования. Качество FrontEnd веб-сайтов чрезвычайно важно, что и является причиной существования феномена веб-дизайна.

Разработка FrontEnd опирается на три технологии: HTML, CSS, JS (JavaScript).

Назначение их следующее:
1) HTML формирует структуру материала;
2) CSS создает оформление;
3) JS, как тьюринг-полный язык программирования, решает те задачи, которые неспособны решить декларативные языки HTML и CSS, например, сложные анимации или отзывчивость на действия пользователя (чем размывает грань между FrontEnd и BackEnd).

Три языка создания веб-сайтов перечислены в порядке их фундаментальности, так, без оформления и функциональности веб-сайт может обойтись, но содержание записывается в html-файл, то есть, без HTML обойтись нельзя уже никак, это наиболее фундаментальная технология. Исторически языки возникали тоже в этом порядке. Приступать к их изучению следует тоже в этом порядке.

## 2. Язык HTML

### 2.1. Введение

HTML – язык разметки гипертекста. То есть буквально, возможно сначала написать неразмеченный фрагмент гипертекста "сплошным текстом", потом разметить в этом тексте структурные блоки. Понять, что такое структура, легче всего, рассматривая некоторый учебник. Материал учебника всегда структурирован: он разбит на главы, параграфы, абзацы и прочие единицы, образующие иерархическую структуру. В печатном издании эта структура отражена в оглавлении (в наиболее крупном делении, абзацы в оглавлении не отражаются).

С точки зрения веб-дизайна, необходимо еще выделить в отдельные структурные единицы заголовки – они не являются частью текста, это текст совершенно особого рода. Заголовки отличаются от текста не укрупненным шрифтом, а по смыслу, например, они раньше индексируются поисковыми машинами, чем текст. Попытка оформить заголовки, как обычный текст, просто набранный крупным шрифтом, приведет к плохой индексации веб-сайта, такой сайт рискует затеряться в океане ресурсов сети Интернет. Для коммерческих сайтов это верная смерть. Более подробную информацию по оптимизации сайтов для поиска ищите по запросу SEO.

Разнообразный необычный текст: всевозможные правила, примеры, сноски, врезки,

примечания и т.д. также должны быть выделены как структурные единицы. На каждый фрагмент подобного рода в разметке выделяется отдельный блок с пометкой, отражающей суть этого фрагмента, например, что данный блок – это примечание. Сразу оговорим одну распространенную ошибку: неправильно создавать в разметке блоки текста, который будет выделен каким-либо конкретным визуальным способом, например, курсивом. Если было принято решение выделять курсивом, скажем, определения, то следует создавать блоки с пометкой, означающей, что это – определения, а не с пометкой, что эти блоки выделяются курсивом. Разметка не должна содержать сведений об оформлении текста, она должна отражать смысл блоков текста.

Только что сформулированное требование к разметке носит название семанитчности. Выше было замечено, что нарушение семантичности может повлечь катастрофические для проекта последствия.

Блоки текста очевидным образом вкладываются друг в друга, например, параграфы – в главы, образуя иерархическую структуру. Практически обязательно создавать некоторое наглядное представление этой структуры, возможно, в виде простого рисунка на бумаге. Наиболее ясными способами отобразить структуру являются древовидная схема, схема, изображающая блоки в виде прямоугольников, вложенных друг в друга и "вид сбоку" предыдущей схемы, изображающий блоки скобками, следующими слева направо по уровням вложенности. Древовидная схема, расположенная по иерархическим уровням, носит название документ-объектной модели (DOM) и играет фундаментальную роль в создании программ, вмешивающихся в FrontEnd. "Плоская" схема в виде прямоугольников несколько менее наглядно отражает иерархию, зато дает дополнительный "бонус" в виде частичной информации о пространственном расположении блоков (что относится уже к оформлению, этот тип схемы помогает установить связь между разметкой и будущим оформлением веб-сайта). "Боковая" схема в виде скобок хороша тем, что отражает слоистую структуру веб-сайта: структурные блоки накладываются друг на друга, образуя стопку слоев, более низкие в иерархии блоки располагаются в этой стопке выше.

## 2.2. Синтаксис

HTML распознает только последовательность символов – текста и разметки – и игнорирует их расположение в окне текстового редактора, где мы создаем разметку. Точная формулировка правил:
1) символ окончания строки равносилен пробелу;
2) символ табуляции равносилен пробелу;
3) множество пробелов подряд равносильно одному пробелу.

Конечно, при наборе текста и создании разметки удобно располагать текст на экране не сплошь, а структурировано, с отступами разной глубины, пустыми строками и т.д. Все это очень помогает ориентироваться в тексте и настоятельно рекомендуется, но никак не влияет на чтение текста браузером.

Легко заметить, что с точки зрения браузера код представляет собой одну единственную строку большой длины, линейную структуру. Единственный способ выделить в такой линии блок – поставить метку начала блока и метку конца блока. Эти метки называются тэгами (здесь мы тоже наблюдаем специфическую терминологию). Соответственно, различаются открывающий тэг и закрывающий тэг. Имена тэгов записываются в угловых скобках, закрывающий тэг – такой же, как открывающий, но перед именем тэга ставится слэш:

```
<tag_name> ... </tag_name>
```

Как говорят, образуется контейнер. Его содержимое – контент.

Помимо такого простейшего синтаксиса существует более общий синтаксис, включающий атрибуты тэгов. Атрибуты модифицируют тэги, играя роль параметров настройки с различными значениями. Атрибуты записываются внутри угловых скобок открывающего тэга через пробел, значения атрибутов устанавливаются при помощи знака равенства:

```
<tag_name attribute1="value" attribute2="value2" ...> ... </tag_name>
```

Пример: гиперссылка на Яндекс:

<a href="http://www.yandex.ru">Яндекс</a>

Здесь именно атрибут "href" содержит адрес гиперссылки, слово "Яндекс" пользователь увидит на экране оформленным, как гиперссылка: подчеркнутым и окрашенным в синий цвет, если это не было изменено стилями. Курсор при подведении к этому слову превратится в символ "рука", а щелчок вызовет переход по указанному адресу. Сам же адрес, поскольку он – атрибут, пользователю не виден (точнее, его можно увидеть в левом нижнем углу экрана, если подвести курсор к ссылке).

Кроме тэгов, ограничивающих блоки текста с двух сторон, существуют уединенные тэги, иначе называемые пустыми. Таков, например, перенос строки:

<br>

("break"). Другой пример – вставка изображения:

<img src="picture.jpg">

Этот тэг нельзя употреблять без атрибута "src" ("source"), указывающего путь к файлу с картинкой.

Возможно употребление одинарных кавычек для значений атрибутов, многие авторы настоятельно рекомендуют использовать именно их. Большинство браузеров способны читать код вообще без этих кавычек, однако это нарушение стандарта, и делать так не рекомендуется.

Заметим, что подобный синтаксис характерен для весьма общего стандарта хранения данных под названием XML, язык HTML является подсемейством в XML.

## 2.3. Очень краткий справочник важнейших тэгов HTML

1)  ...  – знаменитый блок div. Велик и всемогущ. Настраивается стилями, без них никак себя не проявляет. Пользуясь одним только этим тэгом и стилями, можно сверстать ЛЮБОЙ дизайн. Недостаток – полная безликость, обратная сторона универсальности.

2)  ...  – то же самое, что div, только строчный элемент. То есть разница между ним и div – только в оформлении, а не в смысле. А стили могут полностью менять оформление, в том числе превращать строчный элемент в блочный и наоборот. Именно поэтому он прозябает на окраинах, в то время, как div – частый и желанный гость в разметке.

3)  ...  – абзац текста. От блока div отличается почти только названием. Однако он вносит семантичность: раз в разметке появился блок , значит, здесь имеется именно какой-то текст, а не все, что угодно, как в случае div. Настоятельно не рекомендуется использовать тэг  для чего-либо, кроме как для хранения абзацев текста.

На самом деле, отличия есть. Например, внутри элемента  не должны появляться заголовки. Применяя трюки и хаки, заголовок можно таки поместить внутрь абзаца, но это будет очень непрямым и неправильным использованием этого элемента.

4) <a href="some_addres"> ... </a> – гиперссылка. Адрес может быть адресом веб-страницы в Интернет по протоколу http, может быть адресом файла по протоколу ftp, может быть путем к файлу в файловой системе нашего компьютера или локальной сети и т.д. Внутри гиперссылки может располагаться текст, тогда этот текст является гиперссылкой и оформляется на экране по правилам оформления гиперссылок. Но можно туда поместить вместо текста картинку – тогда картинка станет гиперссылкой. Можно поместить блок div – тогда весь этот блок станет гиперссылкой, и вообще, все, что мы поместим внутрь гиперссылки, начинает работать, как гиперссылка.

5) <img src="path_to_picture/picture.jpg"> – картинка. Источником картинки может быть файл на нашем компьютере или в сети Интернет, в этом случае указывается интернет-адрес картинки, и она скачивается при открытии сайта с удаленного сервера. Допустимы практически все форматы графических файлов (а вот .xcf — нет!).

6) <br> – принудительный (жесткий) перенос строки. Хорош для создания коротких строчек, например, при написании стихов. Недостаток: в большинстве случаев ухудшает семантичность, рекомендуется избегать этого тэга всюду, где только возможно.

7) <ul> ... </ul> – маркированный список (unordered list). Элементы списка отмечаются слева буллитами, буллиты имеют отрицательное смещение влево, в следствие чего имеют склонность вылазить за границы блоков, это исправляется стилями, если нужно.

8) <ol> ... </ol> – нумерованный список (ordered list). Элементы списка нумеруются автоматически, не надо прописывать номера вручную! Номера обладают теми же особенностями поведения, что и буллиты маркированных списков.

9) <li> ... </li> – элемент любого из списков (list item). Мало чем отличается от блока div, кроме нумерации или буллита. Вполне может существовать самостоятельно, не будучи заключен в контейнер "список", при этом по умолчанию считается элементом маркированного списка и помечается буллитом. Однако, такой неприкаянный элемент списка – все равно, что ящик, вынутый из комода: лучше ему все же находиться в комоде, чем просто так. Общий вид маркированного списка:

```
<ul>
      <li> ... </li>
      <li> ... </li>
      ...
      <li> ... </li>
</ul>
```

10) <h1> ... </h1> – заголовок 1 уровня. Всего имеется 6 уровней заголовков: h1 ... h6.

Несколько новых элементов, появившихся недавно, с появлением стандарта HTML5:

1) <header> ... </header> – "шапка" сайта. Это самый обычный блок div, но озаглавленный, как шапка, и предназначенный именно для этого. Добавляет семантичности. Помещать шапку в простой div или любой другой элемент – ухудшать семантичность.

2) <main> ... </main> – блок основного содержания сайта. Как и <header>, это просто блок, озаглавленный надлежащим образом.

3) <footer> ... </tooter> – "подвал" сайта. Тоже – просто блок.

4) <article> ... </article> – блок для помещения в него статьи.

5) <nav> ... </nav> – блок для вставки в него навигации по сайту. Весьма полезен, чтобы выделить навигационные гиперссылки из множества всех гиперссылок.

Есть еще набор тэгов, которые формируют каркас сайта и несут служебную информацию. Приведем их в виде фрагмента кода:

```
<!DOCTYPE html>
<html>
<head>
      <title> ... </title>
      <meta charset='utf-8'>
      <link rel='stylesheet' href='style.css'>
</head>
<body>
      ...
      ...
      ...
</body>
</html>
```

Теперь поясним смысл написанного. Уединенный пустой тэг !DOCTYPE – декларация версии языка HTML. На сегодня имеется пять версий, приведенный здесь доктайп устанавливает версию HTML5, самую последнюю на сей день. Он лаконичен и прост.

Тэг <html> создает контейнер, содержащий весь код веб-страницы. За пределами этого контейнера ничего не должно быть! Кроме доктайпа, конечно. Сам же контейнер состоит из двух контейнеров: <head> и <body>, ничего другого в нем нет.

Контейнер <head> (не путать с <header>!) – голова – стоит вверху, он содержит служебную информацию, которая нужна для правильного отображения сайта, но сама на странице не отображается.

Контейнер <title> – заголовок, отображаемый в ярлыке вкладки браузера.

Пустой тэг <meta charset=" ... "> устанавливает кодировку. Обязателен! Кодировка UTF-8 является правилом хорошего тона в сети, CP1251 не рекомендуется.

Тэг "link" служит для связи html-файла с другими файлами, тип связи указывается в атрибуте "rel", адрес – в атрибуте "href". В данном случае устанавливается связь со стилевым файлом, написанном на языке CSS.

Контейнер <body> – то, что пользователь увидит на экране, текст с разметкой, картинки и все остальное. Все содержимое сайта должно находиться только в этом контейнере.

### 2.4. Классы и идентификаторы

Атрибуты "class= ..." и "id= ..." добавляются к тэгам для привязки к ним программных скриптов и стилевых свойств. Атрибут "id" служит для точной идентификации единственного конкретного элемента, поэтому на странице идентификатор с каким-либо именем может употребляться только один раз. Атрибут "class" создает целый класс элементов, объединенных каким-то общим свойством. Например,

555-555 – телефон главного директора,

<img src=av12jg0.jpg class='portrait'> – одна из картинок с чьими-то портретами.

Новички, особенно с математически складом ума, часто придают классам (и идентификаторам) простейшие имена вроде тех, что используются в математике для обозначения переменных: a1, a2, b, c, ... Опытные верстальщики резко критикуют такой подход и настаивают на подробных именах, четко отражающих сущность элементов веб-страницы, как упомянутый выше "generalDirectorPhone". Существует несколько устоявшихся подходов к формированию имен, один из наиболее продуманных разработан командой Яндекса, составляет часть так называемой методологии БЭМ.

## 3. Язык CSS

### 3.1. Введение

CSS вбирает в себя все, что касается оформления веб-страницы.

Оформление можно разделить на две части:
1) геометрию расположения элементов;
2) цвета, тени, фоновые рисунки, шрифты и прочую декорацию.

Декорировать текст проще, чем управлять геометрией.

Напомним, что для подключения стилевого файла существует специальный тэг

<link rel=stylesheet href='style.css'>,

который указывает на то, что стили записаны в отдельном файле 'style.css' (имя файла 'style' приведено в качестве образца, расширение 'css' обязательно). Таким образом, рядом с файлом HTML надо создать стилевой файл и в него вписывать стили.

Замечание 1. Существует возможность вписать стили прямо в файл HTML: существует тэг <style> ... </style>, образующий контейнер для стилевых свойств и атрибут style=" ... ", который можно добавлять к любому тэгу и вписывать стилевые свойства в качестве значения этого атрибута. Оба этих способа применять категорически не рекомендуется!

Замечание 2. Кто-то может вспомнить старые тэги <font> ... </font> и т.д., с помощью которых тоже можно оформлять написанный текст. В настоящее время использовать их ЗАПРЕЩЕНО!

## 3.2. Общий синтаксис стилевых свойств

селектор {свойство: значение; свойство: значение; ...}

где селектор указывает, к каким элементам веб-страницы применить данный набор свойств. Подробно этот вопрос будет освещен позже. В простейшем случае селектором выступает имя тэга или имя класса, например:

h1 {color: blue}

установит синий цвет всем заголовкам первого уровня,

.portrait {width: 400px}

установит ширину 400 пикселей всем элементам класса 'portrait' (необязательно картинкам! Любым элементам, которые имеют атрибут class='portrait'). Обратите внимание на точку перед именем класса!

Замечание 1. Все пробелы – необязательны, нужны только для удобства.

Замечание 2. Существуют мультисвойства – это сокращенные записи для целой группы свойств, например:

h1 {border: 1px solid red}

(сплошная красная рамка толщиной 1 пиксель для заголовка 1 уровня), здесь значения отделяются друг от друга пробелом; существуют свойства, способные принимать несколько значений одновременно, тогда значения перечисляются через запятую. В некоторых случаях синтаксис бывает более сложный, чем здесь описано.

## 3.3. Декорация текста

1) color — цвет текста. Например,

h1 {color: blue}

– устанавливает синий цвет для заголовков 1 уровня;

2) font — шрифт. Например,

a {font: bold italic 17px Arial}

– устанавливает для гиперссылок шрифт жирный курсив ариал, размер 17 пикселей (кегль можно измерять и в типографских пунктах, как это принято в издательском деле);

Замечание 1: font — это мультисвойство, объединяющее несколько свойств, таких, как font-size, font-family и т.д. Оно очень чувствительно к порядку расположения значений, значения надо располагать только в указанном порядке! В случае возникновения непонятных ошибок следует обратиться к литературе.

Замечание 2: в операционной системе пользователя может не быть установлен указанный шрифт. В этом случае ОС подставляет вместо него подходящий (с ее точки зрения!) субститут. Это может нарушить дизайн, следует иметь в виду такую возможность!

3) background — фон элемента. Например,

header {background: red}

– устанавливает красный цвет шапке сайта. Мультисвойство, объединяет множество отдельных свойств, фоном могут служить изображение, градиент и т.д, есть возможность установить несколько фонов на разные части одного элемента, есть и другие экзотические трюки;

4) text-decoration — декорации текста, такие, как подчеркивание, надчеркивание, зачеркивание и т.д. Например,

a {text-decoration: none}

– убирает подчеркивание с гиперссылок;

## 3.4. Введение в геометрические свойства

С точки зрения геометрии, различают два типа элементов:

1) блочные (block);
2) строчные (inline).

Примером строчного элемента является слово в тексте. Слова выстраиваются в линию (в строку), которая будет продолжаться, пока не кончится место, после чего продолжится с новой строки и т.д. Другим примером строчного элемента является изображение. Изображения необходимо как-то отделять по вертикали от текста, иначе они встраиваются в строки текста подобно словам.

Примерами блочных элементов являются заголовки. Они сами, без каких-то команд, отделяются от текста по вертикали. Если создать несколько заголовков подряд, без обычного текста между ними – такая ситуация разумна, если это заголовки разных уровней – то заголовки выстроятся друг под другом. Итак, строчные элементы встают друг за другом в строку, блочные – друг под другом в столбец.

Это не единственное геометрическое различие между строчными элементами и блочными. У них по-разному определяются размеры.

Для расположения элементов на странице есть 3 возможности:

1) поток;
2) позиционирование;
3) флоаты.

Замечание: строго говоря, поток является одним из видов позиционирования – статическим, но, во-первых, он задан по умолчанию, то есть, если не позиционировать элементы никак, мы и получим поток, во-вторых, он резко отличается от других видов позиционирования и употребляется много чаще других.

## 3.5. Боксовая модель CSS

Размеры элемента веб-страницы определяются его шириной (width) и высотой (height), рамкой (border), отступами (padding) и полями (margin), например:

```
h1 {
      width: 800px;
      height: 40px;
      border: 1px solid;
      padding: 20px;
      margin: 30px;
 }
```

Ширина и высота элемента задают область, доступную для содержания (текста, картинки), padding – это внутренний отступ, отступ между содержимым и рамкой, margin – это внешний отступ (поле), отступ между рамкой и внешним окружением. Если у элемента есть свой собственный фон, он распространяется на внутренний отступ (доходит до рамки). Если у элемента нет ни рамки, ни собственного фона, различия между внешними и внутренними отступами не существует.

При верстке страниц всегда необходимо помнить о том, что ширина рамки и оба отступа добавляются к указанной ширине и высоте элемента, например, следующая конструкция

```
h1 {width: 100%; margin: 10px}
```

на 20 пикселей не поместится в отведенном для нее месте ни при каких размерах этого отведенного места. Это означает, что надо проявлять осторожность при явном задании ширины и высоты.

Подробно о единицах измерения можно прочитать в специальной литературе, заметим только, что существуют относительные (например, проценты) и абсолютные (например, пиксели) единицы.

## 3.6. Поток

Поток образуется набором блочных элементов, имеющими свойство

```
position: static,
```

которое нет нужды указывать – оно является свойством по умолчанию. Элемент с таким свойством называется статическим. Статические элементы выкладываются на страницу один под другим в том порядке, в каком они прописаны в разметке – эта последовательность и носит название потока.

Размеры элемента в потоке, если их не задавать явно, будут вычислены браузером по следующим правилам:

1) ширина элемента вычисляется так, чтобы он вместился в отведенное ему место вместе с отступами, полями и рамкой;
2) высота элемента определяется по его содержимому.

Способы выхода из ситуации, когда содержимое элемента не позволяет ему вместиться в указанное место, различны, например, добавление свойства

```
overflow: hidden
```

скроет выступающие части содержимого, а свойство

```
overflow: scroll
```

создаст для элемента отдельный фрейм со скроллбарами.

Поток имеет два неочевидных свойства, способных поставить новичков в тупик:

1) схлопывание вертикальных полей;
2) вываливание вертикальных полей за границы родительского элемента.

Схлопывание полей означает, что вертикальные поля двух соседних блоков "проваливаются" друг в друга. Проще пояснить на примере. Допустим, есть набор идущих друг за другом абзацев текста (это блочные элементы), причем верхний отступ установлен в 30 пикселей, нижний – в 10

```
p {
  margin-top: 30px;
  margin-bottom: 10px;
}
```

Расстояние между абзацами будет не 40, а 30 пикселей (размер самого большого из них). Существуют способы заставить блоки располагаться рядом по горизонтали, при этом горизонтальные поля не схлапываются.

Вываливание полей за границы родительского элемента – сущий ад для новичков. Если в разметке есть статический блок, внутри которого помещен другой (дочерний) статический блок и этот дочерний блок имеет верхнее поле, то верхний край дочернего блока будет совпадать с верхним краем родительского блока, а указанное верхнее поле дочернего блока выйдет за границы родительского. Изменить такое поведение можно множеством способов, например, придать родительскому блоку внутренний отступ сверху, он "не пропустит" поле дочернего блока за границу. Другой способ: назначить родительскому блоку свойство

```
overflow: hidden
```