Введение

Что такое HTML?

HTML — это стандартный язык разметки документов в интернете.

Он интерпретируется браузерами, а затем полученный текст отображается на мониторе компьютера или экране мобильного телефона.

Что такое CSS?

CSS (каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (HTML).

Связка двух технологий html/css позволяет создать внешний вид нашего сайта, html это контент нашего сайта, css это стили для данного контента.

Настройка редактора кода

Создание сайта начинается с выбора редактора кода. На начальном этапе я рекомендую выбрать бесплатный редактор, так как их огромное количество и их функционала будет более, чем достаточно для создания сайта любой сложности. Основным преимуществом платных редакторов кода является проверка написанного кода на ошибки, но данная часть может вам пригодится на этапе создания кода, а технологии html и css не являются программированием, поэтому выбор бесплатного редактора более чем оправдан.

Какой же редактор кода выбрать?

Давайте рассмотрим 2 бесплатных редактора кода, основные плюсы и минусы каждого из них

Sublime text

Плюсы

- Условно бесплатный
- Не перегружен лишним функционалом
- Отлично подходит для веб разработки
- Есть возможность установить русский язык меню при необходимости
- Тысячи бесплатных плагинов

Минусы

- Сложно установить расширения для новичков
- Потребуется установка большого количества расширений
- Изредка при сохрани будет спрашивать не желаете ли вы оплатить

Visual studio code

Плюсы

- Бесплатный редактор кода
- Есть возможность установить русский язык меню при необходимости
- Гибкая настройка
- Много предустановленных расширений
- Поддержка большого количества языков программирования

Почти Минусы

- Потребуется настройка и установка расширений при необходимости

В данном курсе мы будем использовать Visual studio code который является самым популярным редактором, самым удобным и как мы можем заметить практически без минусов. В видео вы сможете увидеть как оптимальнее всего настроить данный редактор кода.

Внимание! *Все настройки редактора кода являются индивидуальными, но не возбраняется установка дополнительных расширений.*

Как работает интернет

Для начала давайте разберем определение

Интернет – это множество компьютеров по всему миру, объединенных в единую сеть, которые постоянно обмениваются какой-либо информацией.

Немного предыстории, после создания компьютеров, возникла потребность в передаче информации с одного компьютера к другому, так и появилась идея создания интернета. После того как удалось осуществить обмен информацией между двумя компьютерами, пришли к идее объединения всех компьютеров, для обмена знаниями и информацией.

С какой проблемой столкнулись при создании интернета? Конечно же нужно определить как пронумеровать или определить каждый компьютер в сети. Сам интернет похож на паутину, где в узлах находятся компьютеры, связывающие остальные компьютеры.

Для нумерации данных компьютеров был придумал IP-4 , а в дальнейшем уже IP6 версии.

Существуют два типа IP-адресов:

- Постоянные, закрепленные за определенным компьютером.

- Динамические – присваиваются в тот момент, когда пользователь соединяется с интернетом.

Получается по присвоенному IP адресу мы можем определить в какой стране находится сервер и сам сайт.

Какой минус мы сразу можем заметить, при использовании IP адресов, их очень сложно запомнить, тут к нам на помощь приходит Доменное имя

Доменное имя – уникальное имя, которое данный поставщик услуг избрал себе для идентификации, например mail.ru или google.com.

Доменное имя может иметь несколько уровней. Домен первого уровня обычно определяет страну местоположения сервера (ru – Россия; ua – Украина; uk – Великобритания; de – Германия) или вид организации (com – коммерческие организации; edu – научные и учебные организации; gov – правительственные учреждения; org – некоммерческие организации). С недавнего времени стало возможным использование русскоязычных доменов (рф).

Браузеры

С помощью чего мы просматриваем любую страницу в интернете? Конечно же это браузер. Более чем уверен что с браузером вы уже давным давно знакомы, но для разработки нам нужно помнить что существует огромное количество разных браузеров и нам необходимо сделать сайт отлично отображающимся во всех браузерах.

Надеюсь вам и в голову не придет использовать для разработки Internet Explorer. Он уже давным давно устарел и проверять работу сайтов в нем стоит только, если заказчик этого попросит (такое бывает не часто).

Какой браузер выбрать?

Ответ на данный вопрос максимально прост, тот браузер который вы использовали до обучения идеально подходит и для разработки сайтов. Мы будем разбирать создание сайтов с использование браузера google chrome поэтому внешний вид тех или иных возможностей браузера может незначительно отличаться и тут потребуется найти данный функционал в вашем любимом браузере или же остановиться на google chrome.

Что представляет собой веб-страница

Веб-страница сайта состоит из различных блоков:

- Шапка (header), в которой могут размещаться логотип компании, название сайта, телефоны организации и проч.
- Вертикальная и/или горизонтальная навигация по сайту (меню).
- Основное содержимое (content).
- Нижний колонтитул (footer), где размещается дополнительная информация: автор сайта, счетчики посещаемости.

Протоколы передачи данных

Взаимодействие компьютеров, серверов, маршрутизаторов, коммутаторов определяется протоколами. Каждый протокол – четкий и определенный набор правил и соглашений, предписывающий, каким образом производится обмен и обработка информации.

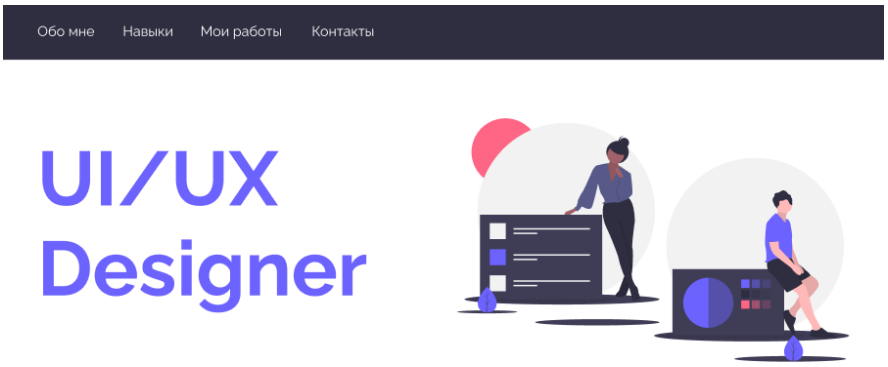
Наиболее известные протоколы, используемые в сети интернет:

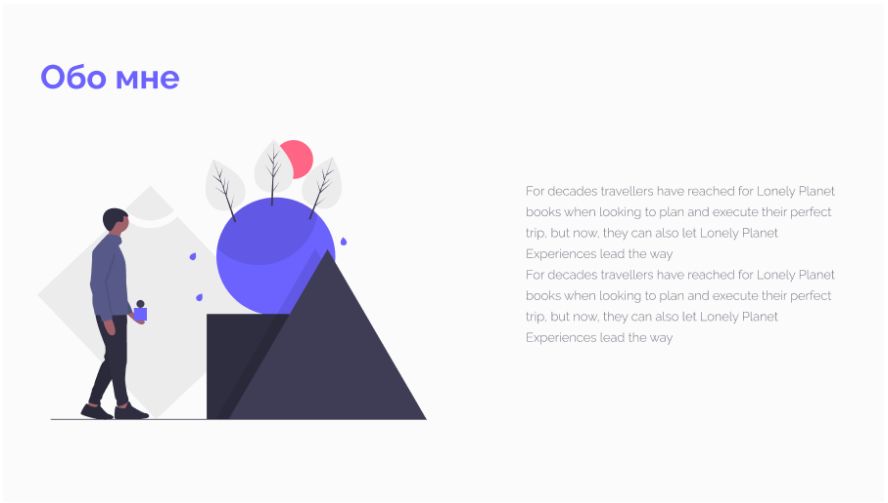
- HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста. Используется при пересылке веб-страниц с одного компьютера на другой.
- HTTPS (HyperText Transfer Protocol Secure) – тоже протокол для передачи гипертекста, но использует дополнительное шифрование данных для более безопасной передачи информации.
- FTP (File Transfer Protocol) – протокол передачи файлов со специального файлового сервера на компьютер пользователя. FTP дает абоненту возможность обмениваться двоичными и текстовыми файлами с любым компьютером сети. Установив связь с удаленным компьютером, пользователь может скопировать файл с удаленного компьютера на свой и наоборот.

Схема HTTP-запроса страницы

Пользователь набирает в браузере адрес нужного ему сайта, после чего посылается HTTP-запрос серверу. Сервер определяет тип страницы по расширению запрашиваемого файла.

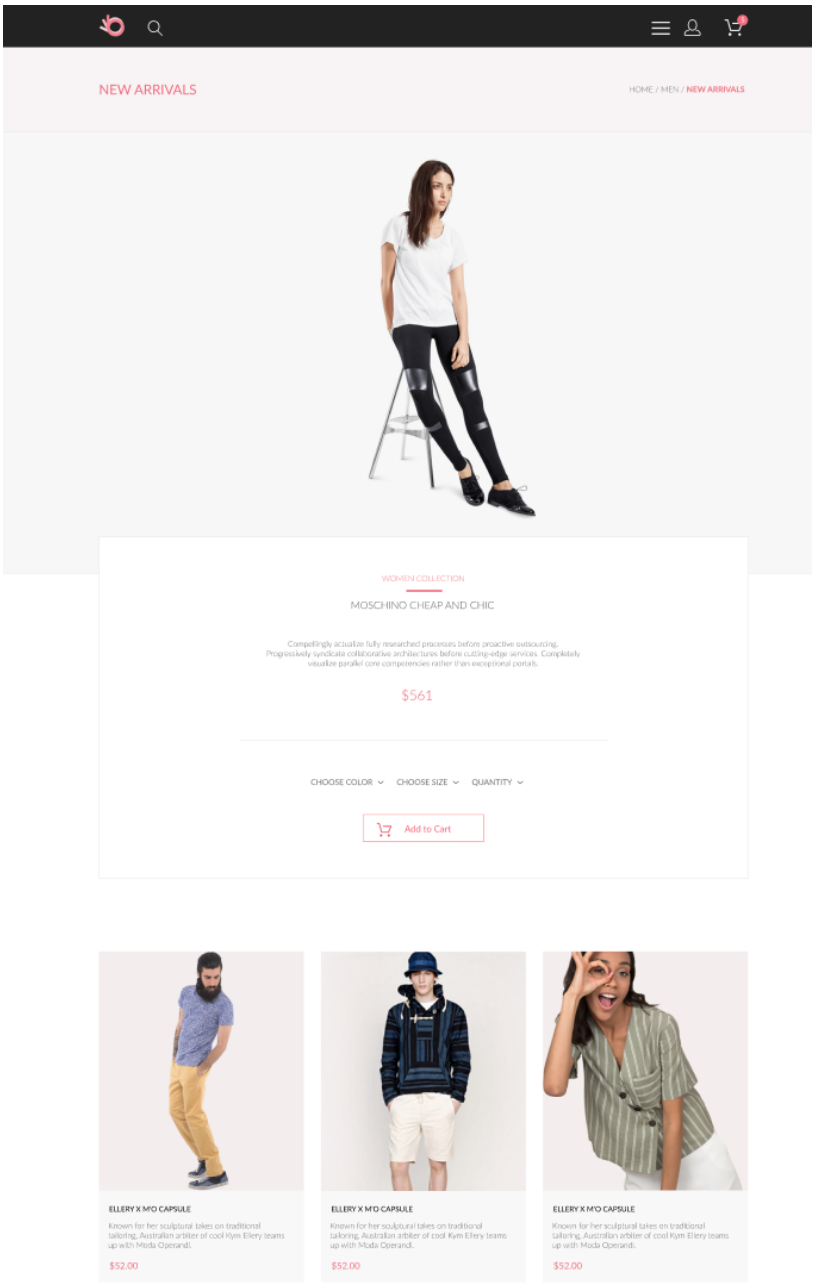
Если наш сайт является статическим и расширение файла .html то сервер находит нужную страницу и отдает “Клиенту” обратно, без обработки или корректировки значений.





В данном примере мы видим статическую страницу портфолио, в которой если нам нужно будет поменять информацию, мы будем менять ее в редакторе кода и в самом коде html.

Если же страница является динамической, когда нам необходимо сформировать новую страницу или получить данные из базы данных, то сначала сервер отправляет sql запрос к базе данных, получает необходимые данные, далее с помощью языка программирования javascript происходит формирование новой страницы и только после этого, пользователь получает свою страницу обратно.



В данном примере мы видим сайт интернет-магазина, чем же он будет отличаться от статического сайта, всё дело в том, что описание товара, название, цена и т.д эти данные хранятся на сервере и первым делом отправляется запрос к базе данных, а уже после этого формируется html страница из контента, который нам предоставляется из базы данных.

Давайте рассмотрим еще один простой пример динамического сайта, тут проще всего представить сайт банка, какие данные на нём постоянно меняются, конечно же это курс валют, получается что пользователь отправляет запрос, сервер получает от базы данных актуальный курс валют, происходит формирование страницы и только после этого пользователь сможет увидеть актуальную информацию на сайте.

Простым языком: если мы обратились к HTML-файлу (.html), браузер возвращает нам HTML-разметку. Если используется другое расширение файлов, например, .php, в данном файле может содержаться программный код, который должен быть обработан (выполнен) интерпретатором языка. На выходе php-интерпретатор должен нам выдать HTML-код, потому что именно его понимает браузер.

Процесс разработки сайта

Давайте детально разберем кто участвует в разработке сайта? Для этого проще всего представить реальный заказ. Допустим к нам пришел заказчик и хочет создать сайт для своей химчистки.

Первым делом нам необходимо собрать всю информацию о функционале и внешнем виде сайта. Для этого на потребуется менеджер проекта, который сможет уточнить все необходимые вопросы, чтобы приступить к разработке

Вторым делом к разработке приступает дизайнер, данный сотрудник не создает сайт, он отрисовывает внешний вид сайта, простыми словами это будет просто изображение сайта, но без дизайнера создать удобный сайт для пользователя будет практически нереально, так что дизайнер это тот сотрудник, который продумывает внешнюю составляющую сайта, плюс удобство для пользователя

Далее к разработке проекта переходит frontend разработчик, данный сотрудник переводит из изображения сайта (из макета, который создал дизайнер) в функционал, который работает в браузере, у нас есть внешний вид сайта, который отображается в браузере, уже будет работать перестроение блоков для мобильных устройств на сайте, будут созданы эффекты наведения, анимации и связь всех страниц друг с другом.

Чего же нам еще не хватает, конечно же это функционала сайта, за эту часть отвечает backend разработчик, он может приступить к созданию функционала в момент разработки frontend части, или уже после завершения. Что же за функционал сайта? тут нужно помнить что мы должны хранить любые значения, мы сохраняем данные пользователя при регистрации, мы узнаем порядок людей в очереди и тд, всё что должно храниться отдельно на сервере, именно этот функционал реализует backend разработчик.

Думаю на данном этапе уже много информации и теперь давайте соберём как работает связь frontend + backend на примере интернет магазина

1. Пользователь заходит на сайт и просматривает внешний вид, за всё что он видит в браузере отвечает frontend
2. Скорее всего пользователю нужно зарегистрироваться на сайте, он нажимает на ссылку регистрации и переходит на страницу, переход на новую страницу и опять же внешний вид, за это снова отвечает frontend
3. Если пользователь вводит не в нужном формате номер телефона и email, то ему высвечивается подсказка об ошибке или маска, как правильно ввести значения, снова же эта часть работает на стороне пользователя и за нее отвечает frontend
4. После того, как пользователь нажимает на кнопку “Зарегистрироваться” данные улетают на сервер и за сохранение этих данных отвечает уже backend часть
5. Теперь сайт знает имя пользователя и может их передать с сервера, для отображение на главной странице (чаще всего в правом верхнем углу) за работу с сервером отвечает backend составляющая
6. Когда вы хотите залогиниться на портале, проверка корректности введенных данных снова происходит на сервере

Простыми словами, получается очень продуманная схема работы на сайте, так как если бы всё хранилось у нас в браузере и на frontend части, то в обычной ситуации, когда вам необходимо зарегистрироваться на портале, вы сможете это сделать только в вашем браузере и когда вы попытаетесь залогиниться с мобильного телефона, это уже не получится сделать, так как только ваш браузер на вашем компьютере помнит логин и пароль. В ситуации, когда мы используем backend не важно с какого устройства вы прошли регистрацию или с какого устройства в дальнейшем вы будете заходить на портал, все данные уже хранятся на сервере и вы всегда сможете зайти на сайт.

Надеюсь так стало еще понятней за что отвечает видимая часть сайта, а что будет функционалом страницы

Не забываем такие важные моменты, что готовый проект обязательно необходимо тестировать, так же его необходимо продвигать в интернете и в социальных сетях и мессенджерах, плюс возможно на сайте будет какой-то контент, его должен добавить контент менеджер.

Создание первой html страницы

Первое что нам необходимо знать при создании html страницы, это то что html не является языком программирования и в итоге не важно какой у вас стартовый набор знаний или предрасположенность к математике или вы вообще не умеете считать логарифмы, тут вам это не потребуется, вам просто необходимо указать браузеру где и какой контент должен быть, для этого используются тега, о которых мы сейчас максимально подробно разберёмся.

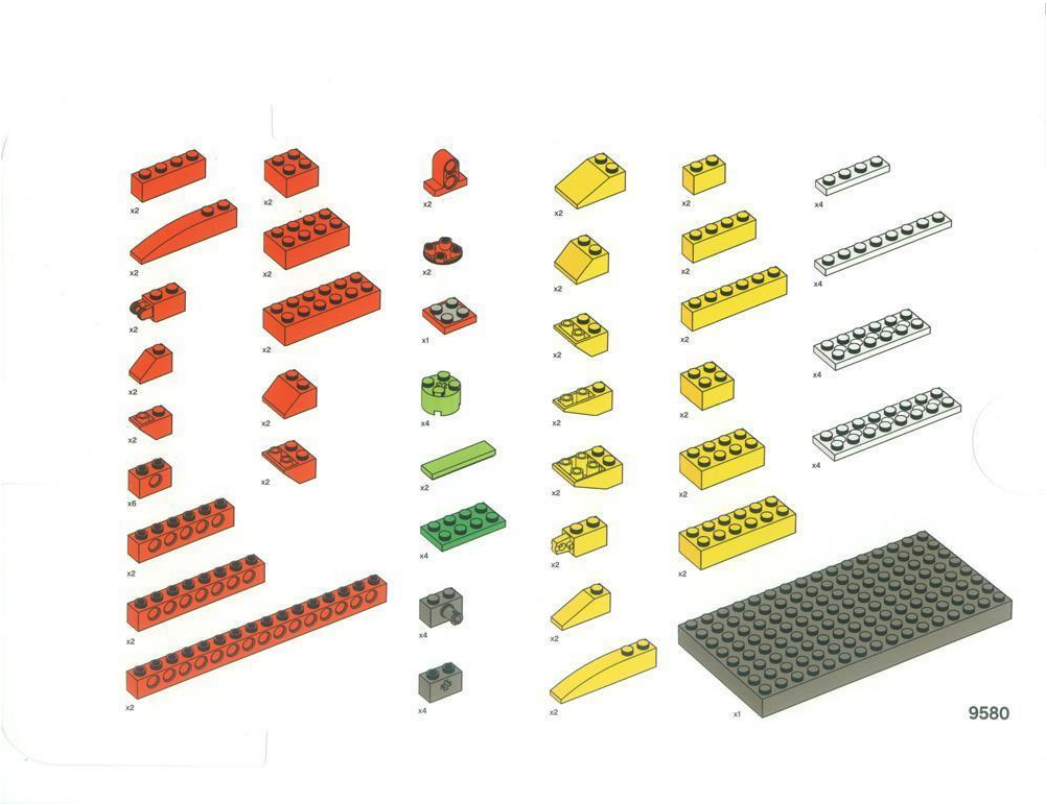
HTML (HyperText Markup Language) — это язык гипертекстовой разметки страницы. Он состоит из тэгов. Каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегом. Есть несколько визуальных сравнений, чтобы лучше представить html.

Теги

Для понимания что такое теги, давайте представим себя разработчиками html и подумаем какой контент у нас бывает, тут всего 2 варианта, это когда нам нужно

указать где элемент открывается и где закрывается или точечное действие, поэтому теги бывают 2х видов, **парные** и **одиночные**

Любая html-страница похожа на конструктор, в котором детальки — html-тэги: заголовки, параграфы, списки, ссылки, картинки. В итоге html — это наполнение нашего сайта.



У нас есть все необходимые элементы для создания абсолютно любого контента на сайте.

Чтобы понять, где открывается и закрывается элемент, используются парные тэги.

Закрывающий тег образуется путём добавления слэша «/» перед именем тега: <имя тега>…</имя тега>. Между начальным и закрывающим тегами находится содержимое — контент.

Парные теги

Вам нужно запомнить что все теги являются логическими сокращениями от английских слов, поэтому их так легко запомнить. Для тех кто хочет посмотреть расшифровку тегов, вам всего лишь нужно ввести запрос в любом поисковике “расшифровка тегов html”

Давайте рассмотрим пример

```
<p>Мой первый параграф в html</p>
```

Одиночные теги стоят сами по себе и, как правило, не меняют контент вокруг себя. Пример:

```
<br> -- тег переноса строки
<hr> -- тег горизонтальной линии
```

Здесь проще всего представить, как вы путешествуете на автомобиле. Когда въезжаете в город, видите табличку города «Москва». На выезде из города — такая же табличка, только название города перечеркнуто наклонной чертой «~~Москва~~».

Браузер просматривает (интерпретирует) HTML-документ, выстраивая его структуру. Он отображает её в соответствии с инструкциями, включенными в файл (таблицы стилей, скрипты). Если разметка правильная, то в окне браузера отобразится HTML-страница, содержащая HTML-элементы — заголовки, таблицы, изображения и т. д.

Атрибуты тегов

Очень часто у нас есть теги, которым необходимо знать дополнительную информацию, если это ссылка. то куда переходить, при нажатии на эту ссылку, если это изображение, то где именно находится и в каком формате данное изображение. Именно для таких ситуаций используются атрибуты тегов, давайте рассмотрим пример и сразу всё станет понятно

Пример:

```
<a href="contacts.html">Контакты</a>
<img src="book.png" alt="Книга">
```

При изучении тэгов и их атрибутов я не рекомендую пытаться запомнить их все, вам необходимо понять что все тэги и атрибуты это сокращения от английских слов. Давайте разберём пример выше, только с будем смотреть от сокращений английских слов

a: anchor - якорь

href: hyper reference -- гиперссылка

img: image - картинка

src: source - источник

alt: альтернативный текст

Структура HTML-документа

Каждая страница в HTML-документе состоит из трех обязательных элементов:

1. Тип документа.
2. Раздел <head></head> с технической информацией о странице: заголовок, описание, ключевые слова для поисковых машин, кодировка. Введенная в нем информация в основном не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.
3. Раздел <body></body>, где располагаются все элементы, которые видит пользователь. Именно с этим разделом мы в основном и будем работать.

DOCTYPE отвечает за корректное отображение веб-страницы браузером. DOCTYPE определяет не только версию HTML (например, html), но и соответствующий DTD-файл в интернете.

Элементы, находящиеся внутри тега <html>, образуют дерево документа, так называемую объектную модель документа, DOM (document object model). При этом элемент <html> является корневым.

Пример структуры HTML5:

```
<!doctype html>
<html>
<head>
    <title>Document</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
...
</body>
</html>
```

Разберём, какие ещё элементы могут быть кроме тегов и текстовой информации.

Спецсимволы

Используя спецсимволы в HTML, можно заменять уже имеющиеся на клавиатуре компьютера символы или указать те, что отсутствуют на клавиатуре. Например, значок копирайта.

Любые теги никак не показываются в окне браузера, поскольку воспринимаются им как команды для вставки элементов и изменения их свойств. Но иногда требуется на веб-странице вывести теги, например, для демонстрации HTML-кода. В этом случае используйте спецсимволы **<** и **>** для замены угловых скобок < и >.

Некоторые варианты спецсимволов:

Код в HTML	Внешний вид	Описание
` `		неразрывный пробел
`&copy;`	©	знак copyright
`&quot; и &raquo;`	« и »	двойные кавычки
`< и >`	< и >	символы
`&prime;`	′	одиночный штрих

Спецсимволов много, поэтому есть простой способ добавления их на страницу — типограф.

Типограф

Типограф помогает автоматически расставить неразрывные пробелы, исправить мелкие опечатки, привести кавычки к правильному виду, заменить дефисы на тире в нужных местах. Можно выбирать любой понравившийся типограф.

1. Типограф 1.
2. Типограф 2.
3. Типограф 3.

Принцип работы типографа — простой. Нужно выделить весь html-код с текстом, который вы создали, скопировать его, добавить в типограф и нажать клавишу «типографировать». Далее заменяем всё содержимое на уже оттипографированный код.

Комментарии

В HTML-документах можно оставить какой-либо комментарий, например, название части страницы. Это не отобразится в браузере. Но любой желающий сможет увидеть этот текст, если воспользуется функцией «просмотр кода страницы» в любом браузере.

Комментарии применяются для того, чтобы не отображать теги в браузере. Это может пригодиться при редактировании веб-сать целые блоки кода.

Пример:

```
<!--Текст, который не будет отображаться на сайте-->
<p>Текст основной части сайта</p>
```

Основные теги оформления текста

Заголовки

Как и в газетах и журналах, в HTML-странице любая статья или новость должна начинаться с заголовка. Предусмотрено 6 уровней заголовков: первый из них будет отображен самым крупным шрифтом, а далее чем выше уровень, тем меньше будет размер шрифта.

.Для отображения заголовков существует тег <h> и указывается цифра от 1 до 6, которая соответствует уровню заголовка. Тег заголовка – парный, не забудьте его закрыть.

<h1>Заголовок первого уровня</h1>

<h2>Заголовок второго уровня</h2>

<h3>Заголовок третьего уровня</h3>

<h4>Заголовок четвертого уровня</h4>

<h5>Заголовок пятого уровня</h5>

<h6>Заголовок шестого уровня</h6>

Параграфы

После заголовка обычно следует какой-нибудь текст, который необходимо заключать в параграфы, или абзацы. При составлении документа выделяйте блоки текста в отдельные параграфы, как это сделано в книгах – в противном случае может получиться сплошной текст, который очень трудно будет читать посетителю сайта.

В HTML для параграфов используется парный тег <p>, и внутри него помещается тот текст, который нужно отобразить. Пример:

```
<p>Здесь мы напишем первый параграф</p>
```

Теги выделения текста

Иногда необходимо выделить слово, словосочетание, предложение или целый участок текста, чтобы привлечь внимание читателя или поискового робота. Чтобы это сделать, можно выделить фрагмент жирным, курсивом или подчеркнуть.

Аккуратнее с подчеркиванием текста: по стандартам принято, что подчеркнуты гиперссылки, и многие пользователи уже настолько к этому привыкли, что, видя подчеркнутый текст, хотят «кликнуть» по этому участку текста, ожидая перехода на другую страницу.

Теги **<b>** и **** внешне делают одно и то же: выделяют текст полужирным. Разница в том, что указывает на важность текста, а <b> просто делает текст полужирным. Так, текст в тегах устройство для чтения текста вслух будет выделять интонацией, а <b> – нет.

Аналогично с тегами **** и **<i>**, делающими текст курсивным: указывает на важность текста, а **<i>** нет.

<small> уменьшает размер шрифта на единицу по отношению к обычному тексту.

<sub> используется для создания нижних индексов. Сдвигает текст ниже уровня строки, уменьшая его размер.

<sup> используется для создания степеней и верхних индексов. Сдвигает текст выше уровня строки, уменьшая его размер. .

<del> перечеркивает текст. Используется для выделения текста, удаленного из документа.

<code> служит для выделения фрагментов программного кода. Отображается моноширинным шрифтом.

<pre> позволяет вывести текст на экран, сохранив изначальное форматирование. Пробелы и переносы строк при этом не удаляются. Не нужно использовать данный тег постоянно, так как то изображение, которое вы видите в редакторе кода, может отличаться от того что вы увидите в браузере, плюс текст будет не адаптивен.

<q> используется для выделения коротких цитат. Браузерами заключается в кавычки.

Списки

Списки в HTML-документах бывают двух видов: маркированные и нумерованные. Отличаются они тем, что у маркированных элементы списка начинаются с точек (маркеров), а у нумерованного – с цифр или букв.

Структура маркированного списка:

```
<ul>
    <li>Первый элемент списка</li>
    <li>Второй элемент списка</li>
    <li>Третий элемент списка</li>
</ul>
```

Структура нумерованного списка:

```
<ol>
    <li>Первый элемент списка</li>
    <li>Второй элемент списка</li>
    <li>Третий элемент списка</li>
</ol>
```

Из примеров видим — структура обоих типов списков одинаковая. Единственное различие в том, что в случае с маркированным списком контейнер будет называться <ul>, а в нумерованном — . В нумерованном списке иногда требуется начать список с какой-нибудь определённой цифры или буквы. Для этого существует атрибут start. В его значении указывается, с какого элемента будет начинаться список. В списках есть возможность менять вид маркеров. Например, арабские цифры изменить на римские, либо вообще убрать маркеры.

Структура вложенного списка

```
<ol>
    <li>Первый элемент списка</li>
    <li>Второй элемент списка
          <ul>
               <li>Первый элемент вложенного списка</li>
               <li>Второй элемент вложенного списка</li>
               <li>Третий элемент вложенного списка</li>
          </ul>
    </li>
    <li>Третий элемент списка</li>
</ol>
```

Вложенные списки предназначены для организации сложной иерархической структуры текста, обычно юридических или технических документов. Такие списки используются для создания многоуровневых меню и навигации по сайту.

Гиперссылки

Гиперссылка — активный элемент (текст, изображение, кнопка и т. п.) одного документа. При нажатии на неё происходит переход к другому документу или к его элементу. Документы, содержащие гиперссылки, называются гипертекстовыми. Гиперссылка может быть добавлена к любому элементу гипертекстового документа и обычно выделяется графически.

В HTML-документах текстовые ссылки по умолчанию выделяются синим цветом. При наведении на них курсора мыши в окне браузера меняют цвет или выделяются подчёркиванием. Посещённая ранее ссылка обычно выделяется цветом, отличным от цвета непосещённой ссылки.

Тег ссылки называется <a>. У него есть обязательный атрибут href, в значении которого мы указываем на файл, место в документе или страницу другого сайта, куда хотим перейти. В содержимом этого тега нужно написать текст, который увидит пользователь.

Виды ссылок

1. Относительные ссылки.
2. Абсолютные ссылки.
3. Якоря.

Относительные ссылки

Используются для перемещения внутри документа или сайта.

```
<a href="file_name.html">текст ссылки, который видит пользователь</a>
```

Абсолютные ссылки

Используются для перехода на страницы внешнего сайта. Для этого в значении атрибута href нужно указать полный путь до той страницы, на которую мы хотим перейти, включая тип протокола.

```
<a href="https://mail.ru" target="_blank">
    страница mail.ru откроется в новой вкладке
</a>
```

У тега ссылки существует атрибут target. Если указать target="_blank", то страница откроется в новой вкладке браузера. Раньше это значение официально не поддерживалось в HTML. Но люди всё равно использовали его, поскольку оно работало. Теперь в HTML5 это полностью поддерживается.

Когда использовать target="_blank":

- если нужно открыть сторонний ресурс и не потерять страничку вашего сайта;
- если пользователь работает над чем-то, что может потеряться при изменении текущей страницы;
- если ссылка ведёт на сторонний ресурс в интернете.

Когда не использовать target="_blank":

- в меню, иначе его пункты будут открываться в новых вкладках;
- из-за личных предпочтений;
- если клиент думает, что так будет лучше.

Якоря

При помощи гиперссылок можно перемещаться не только между страницами документа, но и внутри страницы. Это полезно, когда ваша статья состоит из нескольких разделов и имеет объемное содержание.

В этом случае разумно использовать так называемые якоря, чтобы быстро переходить в интересующий раздел. В том элементе, на который следует перейти, нужно определить уникальный идентификатор.

Для этого существует атрибут id, который можно определять практически для любого тега. Название этого атрибута может быть любым и обязательно уникальным. В значении атрибута href тега-ссылки ставится символ решётки. После него указывается имя идентификатора, к которому нужно обратиться.

```
<a href="#p10">Прочитать 10 параграф</a>
<p id="p1">Текст параграфа №1</p>
<p id="p2">Текст параграфа №2</p>
...
<p id="p10">Вот и долгожданный параграф №10</p>
```

Пример работы якоря по ссылке.

Изображения

В настоящее время в веб-документах используются четыре формата изображений: jpeg, gif, png, svg. Рассмотрим каждый из них подробнее.

JPEG

Плюсы:

- небольшой размер файла;
- поддерживает 16 млн цветов;
- можно управлять качеством изображения при сохранении.

Минусы:

- при сжатии размеров теряется качество;
- не поддерживает прозрачность.

GIF

Плюсы:

- поддерживает анимацию;
- поддерживает прозрачность;
- при сжатии не теряет в качестве.

Минусы:

- количество цветов до 256.

PNG

Растровый графический формат PNG, становясь популярнее, появился в 1995 году как замена GIF. Формат PNG подразделяется на форматы PNG-8 — аналог формата GIF, кроме того, что PNG-8 не поддерживает анимацию — и PNG-32.

PNG – 8

Плюсы:

- поддерживает прозрачность;
- при сжатии не теряет в качестве.

Минусы:

- количество цветов до 256.

PNG – 32

Плюсы:

- При сжатии не теряет в качестве.
- Использует 16 млн цветов.
- Плавный переход от прозрачной области к цветной.

Минусы:

- Большой размер файла.

SVG

SVG – формат векторной графики. Файлы SVG можно читать и редактировать (если есть некоторые навыки) при помощи обычных текстовых редакторов. Есть возможность увеличить любую часть изображения SVG без потери качества.

Растровая графика состоит из разноцветных точек, а векторная — из геометрических фигур. Примечательно то, что векторную графику можно легко перевести в растровую.

Доступно использование растровой графики в SVG-документах. Имеется возможность вставлять элементы с изображениями в форматах PNG, GIF или JPG. Текст в графике SVG —

это текст, а не изображение, поэтому его можно выделять и копировать. Он индексируется поисковыми машинами, не нужно создавать дополнительные метафайлы для поисковых роботов.

Плюсы:

- небольшие размеры файлов, отличное сжатие;
- возможно увеличить любую часть изображения SVG без потери качества.

Минусы:

- если скопировать фоновое изображение в этом формате, получится большой размер файла (подходит для иконок на сайте);
- сложность использования.

Для сохранения фотографий с чёткими краями лучше всего подходит формат JPEG, так как в нём размер файла получается небольшим. Для анимированных изображений единственный вариант — формат GIF. А для сохранения качественных изображений, в которых используется прозрачность, лучше всего подойдёт формат PNG-32. Векторной графике подходит SVG.

Загрузка изображений на страницу

Для отображения изображений на странице есть тег <img>, он одиночный. У него имеется обязательный атрибут src, в значении которого указывают путь к изображению. Этот путь может быть как относительным, так и абсолютным. Второй обязательный атрибут — alt.

Атрибут alt означает альтернативный текст. Используется, чтобы:

1. Если картинка не загрузится на страницу, пользователь увидит текст, который указали в этом атрибуте.
2. Альтернативный текст нужен при продвижении сайта. В нём можно указать ключевые слова, и тогда это изображение будет участвовать в поиске по картинкам.

```
<img src="img/my_foto.jpg" alt="это моя фотография" title="это моя фотография" width="300">
```

У тега <img> можно указать дополнительные атрибуты.

title универсален, его можно использовать практически для любого тега. То, что вы напишете в значении, будет выводиться в виде всплывающей подсказки, при наведении на изображение.

Любому изображению можно задать ширину и высоту, указав эти значения в атрибутах **width** и **height** соответственно. При помощи этих атрибутов изображение лучше не увеличивать, иначе

получится плохое качество. Если задать ширину и высоту изображению, браузер при загрузке страницы будет сразу выделять заданную область под картинку. Эти атрибуты используются редко, ведь вся стилизация находится в css.

Важно! *Всегда храните все изображения в отдельной папке, чтобы знать, где искать то или иное изображение.*

Формы и их элементы

Формы — один из важных элементов любого сайта. Они нужны, чтобы получать от пользователя информацию, отправлять эти данные на сервер, а там уже их обрабатывать. Документ может содержать любое количество форм. Но одновременно на сервер может быть отправлена только одна. По этой причине данные форм должны не зависеть друг от друга.

Допускается внутрь контейнера <form> помещать другие теги. При этом сама форма никак не отображается на веб-странице, видны только её элементы и результаты вложенных тегов.

Важно! *В html нет возможности реализовать опции форм, для этого используется серверный язык программирования.*

Элементы форм

В HTML существует три основных тега элементов форм:

1. Тег <input> в зависимости от значения атрибута type будет выглядеть по-разному и иметь различные значения. Тег <input> одиночный.
2. <textarea> используется, чтобы сформировать многострочное поле ввода для информации. С помощью атрибута cols можно задать ширину этого поля, а с помощью атрибута rows — указать число строк. Тег <textarea> парный.
3. <select> представляет собой выпадающий список, элементы которого указываются в теге <option>. У этого тега есть несколько атрибутов. Чтобы сделать возможность выбора нескольких пунктов, нужно задать атрибут multiple, в значении которого указывается multiple. В атрибуте size определяется число строк выпадающего списка. Их увидит пользователь в браузере. Если необходимо, чтобы какой-то из элементов выпадающего списка был выбран, нужно этому элементу в теге <option> задать атрибут selected. Тег <select> также парный.