Основы Интернет

Игонин Андрей

2019 г.

История интернета

История интернета начинается в 60-годы XX века. Впервые концепцию описал один американский учёный и эпично назвал её «Галактическая сеть».

В 1969 американское агентство DARPA начало создавать экспериментальную сеть «с коммутацией пакетов». Её назвали ARPANET.

> *Коммутация пакетов - способ передачи данных по сети. Принцип работы очень простой: делим информацию на маленькие пакеты и отправляем их независимо друг от друга. Это нужно для надёжности, скорости и эффективности.*

В декабре 1970 года Network Working Group придумала протокол управления сетью, а в 1971 - 1972 его реализовали в ARPANET. Благодаря этому, появилась возможность создавать сетевые приложения. Первым приложением стала **электронная почта**, её сделали в 1972-м.

Но это всё научные исследования. Интернет, каким мы его знаем, придумал Тим Бернерс-Ли. Он изобрёл технологии URI/URL, HTTP, и HTML.

URL

Первая важная технология, которая позволила появиться интернету - URL. Она применяется для обозначения адресов почти всех ресурсов Интернета: сайты и сервера приложений вроде электронной почты. URL сочетает в себе две технологии:

1. **URI** - стандарт записи уникального адреса. Например, адрес сайта это URI: http://google.com
2. **DNS** - система доменных имён. У любого устройства в сети есть свой числовой адрес, он называется **IP-адрес**. Но людям удобнее запоминать буквы - доменные имена. DNS просто помогает найти IP-адрес по доменному имени. Пример: 173.194.44.40 это IP-адрес, а google.com - доменное имя, которое ему соответствует.

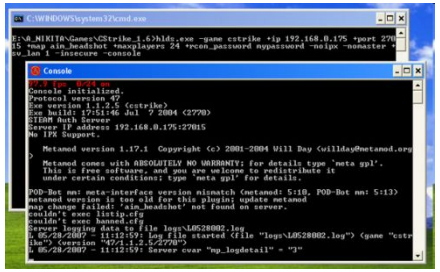
Сервер

Сервер (от англ. server, обслуживающий)

(аппаратное обеспечение) — компьютер повышенной надёжности и производительности для выполнения определённых задач

(сеть) — узел сети, принимающий и обрабатывающий запросы пользователей

(приложение) — приложение, принимающее запросы от клиентов (в архитектуре клиент-сервер)



Сервер (аппаратное обеспечение)

- **Сервер (англ. server от to serve — служить)** — аппаратное обеспечение, выделенное и/или специализированное для выполнения на нем сервисного программного обеспечения (в том числе серверов тех или иных задач). Основное предназначение серверов — предоставление ресурсов для рабочих станций (пользователей).

Серверное оборудование комплектуется более надежными элементами:

1) памятью с повышенной устойчивостью к сбоям,
2) резервированием, в том числе:
   - блоков питания (в том числе с горячим подключением)
   - жестких дисков (RAID; в том числе с горячими подключением и заменой).
3) более продуманным охлаждением.

Конструктивно аппаратные серверы могут исполняться

в настольном, напольном, стоечном потолочном вариантах.

Серверы размещаются в так называемых серверных комнатах.
Управление серверами осуществляют системные администраторы.

Сервер (сеть)

- Сервер (сеть) — узел сети, принимающий и обрабатывающий запросы пользователей

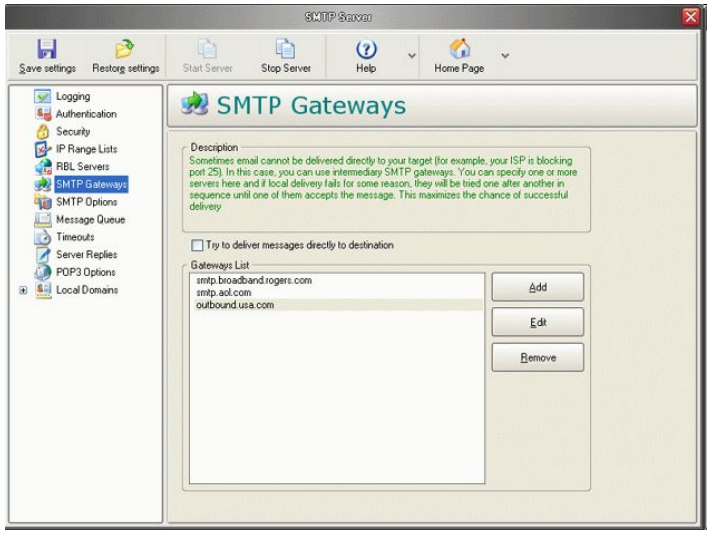
Клиент-сервер (англ. Client-server) — сетевая архитектура, в которой устройства являются либо клиентами, либо серверами. Клиентом (front end) является запрашивающая машина (обычно ПК), сервером (back end) — машина, которая отвечает на запрос. Оба термина (клиент и сервер) могут применяться как к физическим устройствам, так и к программному обеспечению.

Сеть с выделенным сервером (англ. Client/Server network) — это локальная вычислительная сеть (LAN), в которой сетевые устройства централизованы и управляются одним или несколькими серверами. Индивидуальные рабочие станции или клиенты (такие, как ПК) должны обращаться к ресурсам сети через сервер(ы).

1) Пользователь
2) Сервер
3) Данные с сервера в браузере

Сервер (приложение)

- **Се́рвер (англ. server от англ. to serve — служить)** — в информационных технологиях — программный компонент вычислительной системы, выполняющий сервисные функции по запросу клиента, предоставляя ему доступ к определённым ресурсам.

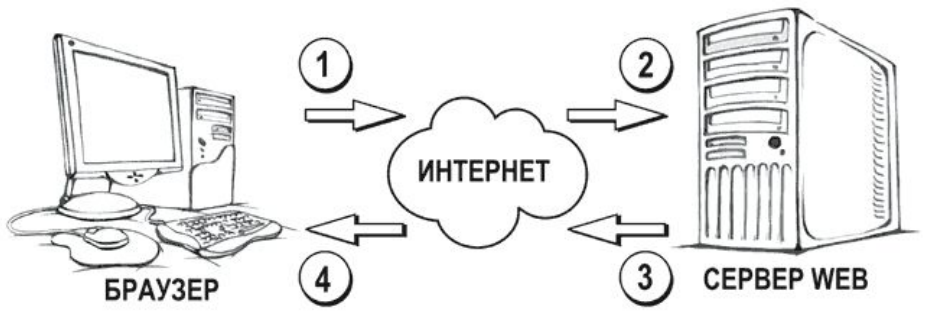


Web сервера

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Веб-серверы — основа Всемирной паутины.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и компьютер, на котором это программное обеспечение работает.

Клиенты получают доступ к веб-серверу по URL адресу нужной им веб-страницы или другого ресурса.



На сегодня двумя наиболее распространёнными веб-серверами являются:

Apache — свободный веб-сервер, наиболее часто используется в Unix-подобных ОС

Microsoft IIS

IIS от компании Microsoft, распространяется с ОС семейства Windows NT

---

nginx — свободный веб-сервер, разрабатываемый Игорем Сысоевым с 2002 года и пользующийся большой популярностью на крупных сайтах

lighttpd — свободный веб-сервер.

Google Web Server — веб-сервер разработанный компанией Google.

Resin — свободный веб-сервер приложений.

Cherokee — свободный веб-сервер, управляемый только через web-интерфейс.

Rootage — веб-сервер, написанный на java.

THTTPD — простой, маленький, быстрый и безопасный веб-сервер.

Open Server — бесплатная программа с графическим интерфейсом использует множество исключительно свободного программного комплекса.

H2O — свободный быстрый веб-сервер, написанный на C.

	Microsoft IIS
Достоинства: Надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д. Поддерживает IPv6.	**Достоинства:** Удобный интерфейс, хорошая производительность и доступность веб-сервера.
Недостатки: Недостатком наиболее часто называется отсутствие удобного стандартного интерфейса для администратора и местами сложность настройки.	**Недостатки:** Сложная настройка. Много проблем с поддержкой. Работает только под Windows. Более высокие требования к производительности аппаратуры.

Взаимодействие сервера и Клиента

Клиент: отправка запроса

Сервер:
- получение запроса
- обработка запроса + подготовка ответа
- отправка ответа

Клиент:получение ответа

Web Браузер

Процесс работы:

- Отправка запроса
- Загрузка заголовков ответа
- Загрузка HTML кода страницы
- Построение DOM
- Загрузка связанных ресурсов
- Отображение
- Исполнение JavaScript

Web Браузер

- Google Chrome
- Mozilla Firefox
- Opera
- Microsoft Edge, Microsoft Internet Explorer
- Apple Safari

Программирование Web Сервера

- PHP
- База данных
- SQL
- Remote API / XML / Rest
- HTML / CSS / JavaScript

HTTP

HyperText Transfer Protocol (*гипертекст трансфер протокол*) - протокол передачи гипертекста.

HTTP (*эйч-ти-ти-пи*) - протокол (то есть регламент, набор условных обозначений), который устанавливает определённый формат общения между *клиентом* и *сервером*. Суть его заключается в том, что клиент *посылает запросы* на URL, а сервер ему отвечает.

Клиент - тот, кто посылает запросы. Когда ты открываешь сайт, ты - клиент.

Сервер - тот, кто принимает запросы и отвечает на них. Когда ты открываешь сайт гугла, сервер гугла посылает тебе веб-страничку - отвечает на твой запрос.

Запрос. Что значит "послать запрос"? Если не углубляться в сложные научные определения, это означает отправить по сети сообщение с каким-нибудь требованием. Это работает так:

1. ты посылаешь запрос
2. между тобой и сервером устанавливается соединение
3. сервер формирует ответ, посылает его тебе
4. соединение закрывается

Так вот, протокол передачи гипер**текста.** "Текста", потому что сервер отвечает на запрос текстом определённого формата. А так как любые данные можно представить в виде текста, то получается, что через HTTP можно пересылать всё: картинки с котиками, гифки с котиками, и даже видео. С котиками.

Структура протокола

- Стартовая строка
- Заголовки
- Тело сообщения

Строка запроса выглядит так:

GET URI - для версии протокола 0.9.
Метод URI HTTP/Версия - для остальных версий.

Метод - название запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен ниже.
URI определяет путь к запрашиваемому документу.
Версия - пара разделённых точкой арабских цифр. Например: 1.0.

Ответ сервера имеет следующий формат:

HTTP/Версия КодСостояния Пояснение

Версия - пара разделённых точкой арабских цифр как в запросе.
КодСостояния - три арабские цифры. По коду статуса определяется дальнейшее содержимое сообщения и поведение клиента.
Пояснение - текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Структура протокола

- Пример запроса:
- GET /wiki/страница HTTP/1.1
- Host: ru.wikipedia.org
- User-Agent: Mozilla/5.0 (X11; U; Linux i686;
- ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
- Accept: text/html
- Connection: close

Структура протокола

- Пример ответа:
- HTTP/1.1 200 OK
- Date: Wed, 11 Feb 2009 11:20:59 GMT
- Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT
- Content-Language: ru
- Content-Type: text/html; charset=utf-8
- Content-Length: 1234
- Connection: close
- (пустая строка)
- (далее следует запрошенная страница в HTML)

MIME Types

- Content-Type: text/html
- Content-Type: application/msword
- Content-Type: image/jpg
- Content-Type: application/zip

Классы кодов состояния.

1xx Informational (Информационный)

В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-сервера подобные сообщения должны отправлять дальше от сервера к клиенту.

2xx Success (Успешно)

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

3xx Redirection (Перенаправление)

Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (жарг. редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4xx Client Error (Ошибка клиента)

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

5xx Server Error (Ошибка сервера)

Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

IDE

- Eclipse
- NetBeans
- PHPStorm
- Plain Text Editors
- Vim

HTML

HyperText Markup Language (*гипертекст маркап лэнгуидж*) - язык разметки гипертекста. В этой аббревиатуре нам интересно слово "разметка".

Разметка - что это вообще такое? Представь, что ты передаёшь текст по сети. Как сделать в тексте заголовок? Выделить абзац? Подчеркнуть слово? Самый простой вариант - пометить начало и конец выделяемого фрагмента условными метками. Например:

```
<заголовок>HTML</заголовок>
<полужирный>HyperText Markup Language</полужирный> <курсив>(гипертекст маркап лэнгуидж)</курсив> - язык разметки
гипертекста.
```

Это разметка.

HTML позволил создавать документы, в которых были заголовки, абзацы, ссылки, таблицы, картинки и куча всего прочего.

HTML

HTML (от англ. HyperText Markup Language — «язык разметки гипертекста») — стандартный язык разметки документов во Всемирной паутине.
Все веб-страницы создаются при помощи языка HTML (или XHTML).
Язык HTML интерпретируется браузером и отображается в виде документа, в удобной для человека форме.

HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879.

Тег – оформленная единица HTML-кода. Например, <HEAD>, <FONT ...>, <BODY>, <HTML> и так далее. Теги бывают начальными (открывающими) и конечными (закрывающими, начинающимися со знака "/"). Например, вышеуказанным тегам соответствуют закрывающие теги </HEAD>, </FONT>, </BODY>, </HTML>.

Элемент – понятие, введенное для удобства. Например, элемент HEAD состоит из двух тегов – открывающего <HEAD> и закрывающего </HEAD>. Следовательно, элемент – более емкое понятие, обозначающее пару тегов и участок документа между тегами, на который распространяется их влияние.

HTML (HyperText Markup Language, язык разметки гипертекста)

HTML — это система верстки веб-страниц, которая определяет, какие элементы и как должны располагаться в документе. Чтобы показать, что мы имеем дело не с обычным текстовым документом, используется термин HTML-документ. Подобные документы открываются под управлением браузера.

HTML-документ
Обычный текстовый файл, который может содержать в себе текст, теги и стили. Изображения и другие объекты хранятся отдельно. Содержимое такого файла обычно называется HTML-код.

HTML

Теперь мы знаем, что такое *разметка*, давай узнаем, что такое HTML. Есть вещи, которые проще показать, чем объяснить. HTML - одна из них.

Всё гениальное просто. Вот как бы выглядел текст этого урока на HTML:

```
<h1>HTML</h1>
<p>Теперь мы знаем, что такое <i>разметка</i>, давай узнаем, что такое HTML. Есть вещи, которые проще показать, чем
объяснять. HTML - одна из них.</p>

<p>Всё гениальное просто. Вот как бы выглядел текст этого урока на HTML:</p>
```

h1 - заголовок первого уровня (header 1)

p - абзац (paragraph)

Теги

Как ты уже догадался, в HTML для разметки используется особый набор символов. Он называется **тег**.

Что такое тег

Тег — это синтаксическая единица языка HTML, которая выделяет или создаёт элемент. Это набор символов, с помощью которого браузер понимает, где элемент создается, начинается и заканчивается. Есть 2 вида тегов: двойные и одинарные.

Двойные теги

Двойные теги показывает начало и конец элемента. Начало элемента обозначается открывающим тегом `<…>` , а конец - закрывающим `</…>` .

Двойной тег обязательно должен быть закрыт. Даже несмотря на то, что современные браузеры умеют в некоторых случаях понимать разметку без закрытых тегов, лучше всегда закрывать их.

Одинарные теги

Одинарные теги просто не имеют пары. Примеры: тег переноса строки `<br>` или горизонтальной линии `<hr>` .

Старые браузеры требовали закрывать одинарные теги: `<br />` , сейчас таких браузеров практически не осталось и допустимо использовать оба варианта синтаксиса.

Вложенность тегов

Теги можно вкладывать друг в друга. Пример:

```
<p>
  <em>Курсив внутри абзаца</em>
</p>
```

Но при вложении тега нужно всегда помнить о том, что **внутренний тег нужно закрывать раньше внешнего.** То есть вот такой код недопустим:

```
<p><em>Я уже говорил тебе, что такое безумие?</p></em>
```

Атрибуты

Атрибуты — это свойства тега. С помощью них мы задаём параметры тега.

Сразу возьмём пример: тег `<a>` — ссылка. Для задания адреса, куда будет вести эта ссылка, нам понадобится атрибут `href`. Вот так будет выглядеть ссылка на страницу ITC Вконтакте:

```
<a href="https://vk.com/itc.digital">ITC Вконтакте</a>
```

Атрибут указывается внутри тега, значение атрибута указывается внутри кавычек. Атрибуты отделяются друг от друга пробелами. Пример ссылки на страницу ITC, которая откроется в новой вкладке:

```
<a href="https://vk.com/itc.digital" target="_blank">ITC Вконтакте</a>
```

У атрибута может не быть значения, тогда наличие атрибута включает какой-то параметр, а отсутствие - отключает. Например, атрибут `disabled`. Если кнопке `<button>` задать атрибут `disabled`, она станет серой и на неё невозможно будет нажать.

```
<button disabled>Нельзя нажимать</button>
```

Результат:

Особенности интерпретации HTML

При преобразовании HTML-кода в веб-страничку есть некоторые особенности, в которых мы сейчас разберёмся.

Перенос строки только через тег

Возможно, у тебя возник вопрос, зачем нужен тег переноса строки, если можно просто нажать энтер. Дело в том, что HTML воспринимает перенос строки как пробел. Это нужно потому, что редакторы кода не переносят строки, которые не помещаются в экран - так удобнее писать код. Поэтому чтобы длинный текст влезал в экран, в коде ставятся переносы строки, которые не нужны, когда страница показывается в браузере.

Несколько пробелов, идущих подряд, считаются за один

Это происходит по той же причине, что и с переносом строки. Так просто удобнее форматировать код в редакторе. Из-за того, что теги вкладываются друг в друга, для удобного восприятия кода вложенность показывают отступами - пробелами. Пример:

```
<article>
  <h1>В Индонезии после землетрясения началось извержение вулкана</h1>
  <p>
    В Индонезии на острове
    Сулавеси проснулся вулкан Сопутан,
    выпустив столб пепла высотой около четырех метров, пишут
    «Новые Известия». Извержение вулкана началось сегодня утром.
    Местным жителям рекомендовали не приближаться к нему и
    использовать респираторы.
  </p>
  <p>
```

Произвольный регистр

`<br>` даст такой же результат, что и `<BR>`, и `<Br>`, и `<bR>`. Несмотря на это, писать разметку лучше в нижнем регистре - это негласное правило.

Перенос строки в теге

При определении тега и его атрибутов можно переносить строку. Это полезно для длинных определений.

Например, для этого изображения:

```
<img
  src="http://example.com/cat.jpg"
  title="Мурка"
  alt="Рыжая кошка валяется в снегу"
  width="640"
  height="480"
>
```

Структура HTML документа

<HTML>

Начало документа.

<HEAD>

Служебная информация.

(Название страницы, ее автор, ключевые слова, и т.д.)

</HEAD>

<BODY>

Тело документа. (Все, что видно на страничке - текст, рисунки, и т.д.)

</BODY>

</HTML>

Конец документа.

Структура HTML-документа

Структура HTML документа - скелет, на основе которого строится вся страница:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Страница</title>
  </head>
  <body>
    <h1>...</h1>
    <p>...</p>
  </body>
</html>
```

<!DOCTYPE>

Первым тегом в любом HTML документе должен идти тег `<!DOCTYPE>`. Он говорит браузеру, по какому стандарту написана страница. На рассвете веба HTML существовал в разных несовместимых версиях, поэтому для их одновременной поддержки нужно было указывать версию явно. Сейчас все пришли к одному стандарту - HTML5. Поэтому для всех сайтов, которые создаются сегодня, нужно указывать `<!DOCTYPE html>` - так обозначается HTML5.

<html>

Вторым тегом идет `<html>` - контейнер, который содержит два тега - `<head>` и `<body>`. HTML-страница должна заканчиваться закрытым тегом `</html>`.

<head>

В теге `<head>` хранится информация о странице. Здесь указывают кодировку `<meta charset="...">`, имя страницы `<title>...</title>`, специальную информацию для поисковиков, а ещё тут подключаются стилевые файлы и скрипты.

Тег `<head>` не отображается. Его цель — сказать браузеру информацию о странице.

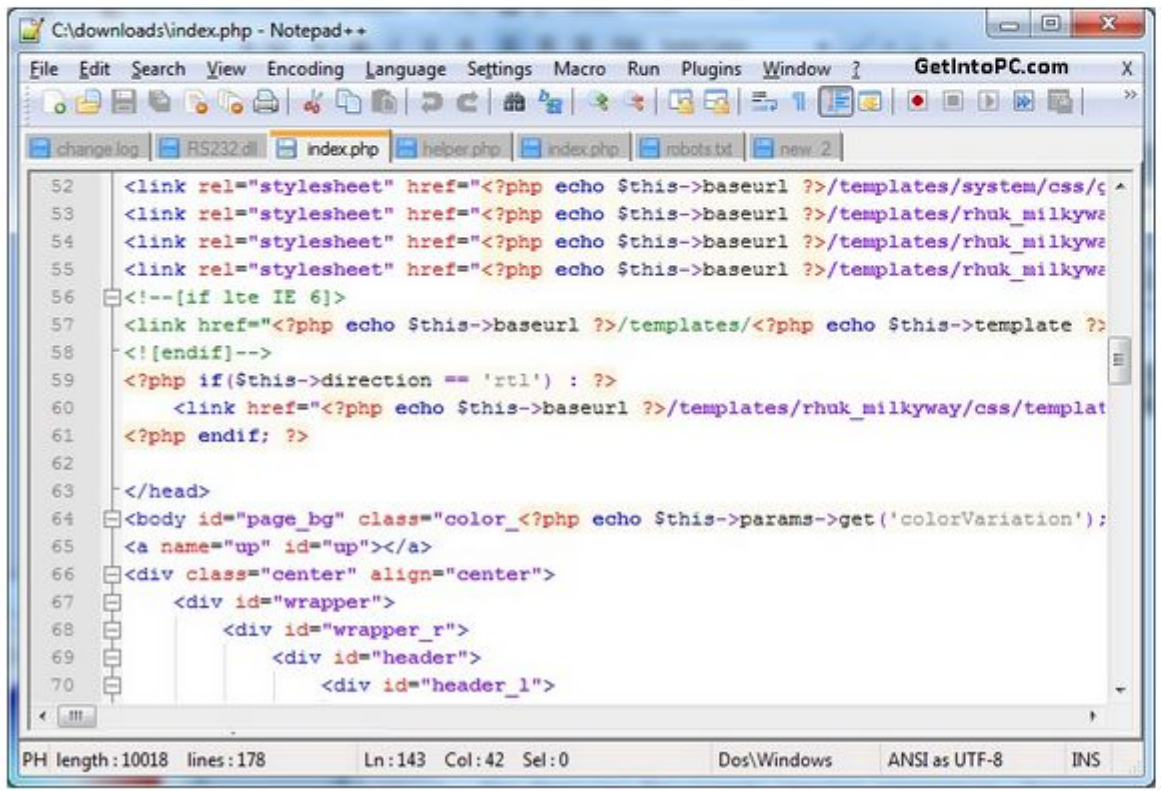
<body>

В теге `<body>` размещается весь контент страницы, который пользователь увидит в браузере.

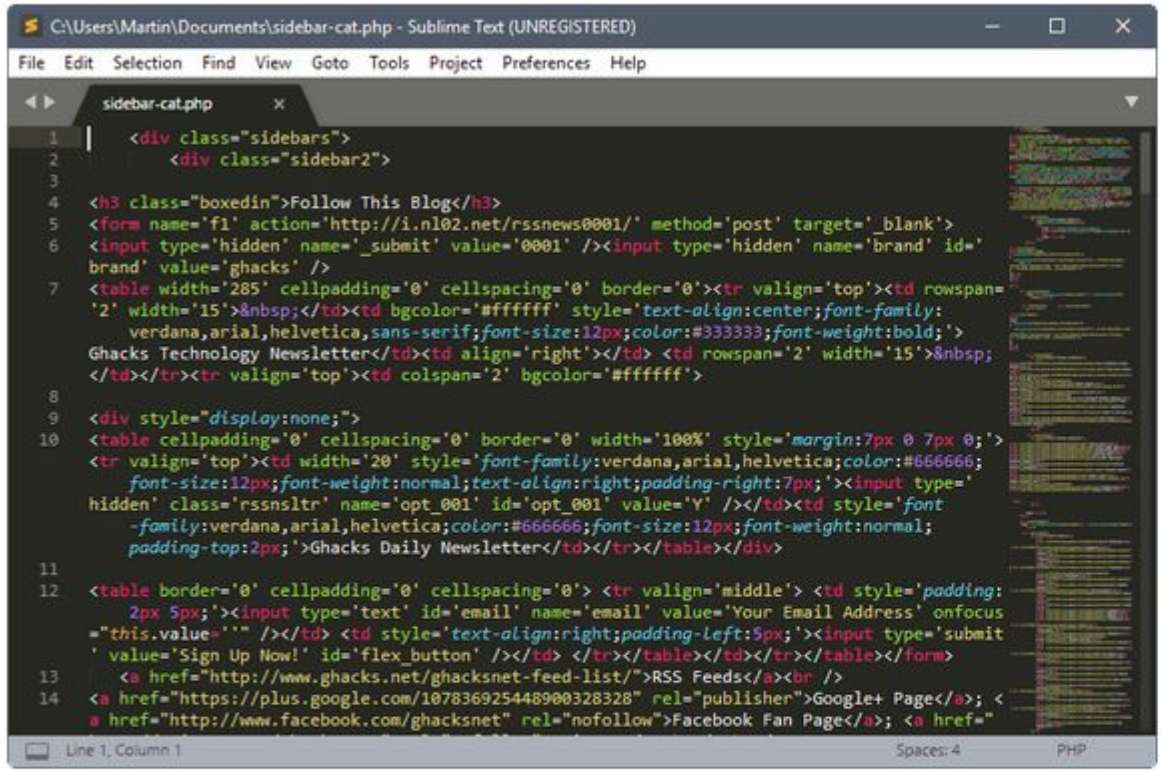
Редакторы кода

Разработчики пишут код в специальных редакторах. Они **отличаются назначением**. Остальные отличия почти всегда вытекают из него. Существует несколько редакторов кода, которые подходят для веб-разработки. Мы рассмотрим несколько из них и порекомендуем два (один из них только для слабых компьютеров). Редакторы кода - холиварная тема, поэтому этот урок содержит субъективное мнение. Если у тебя уже есть любимый редактор, то можешь пользоваться им, потому что **любой разработчик свободен пользоваться тем, что ему нравится.**

Notepad++ - для ветеранов

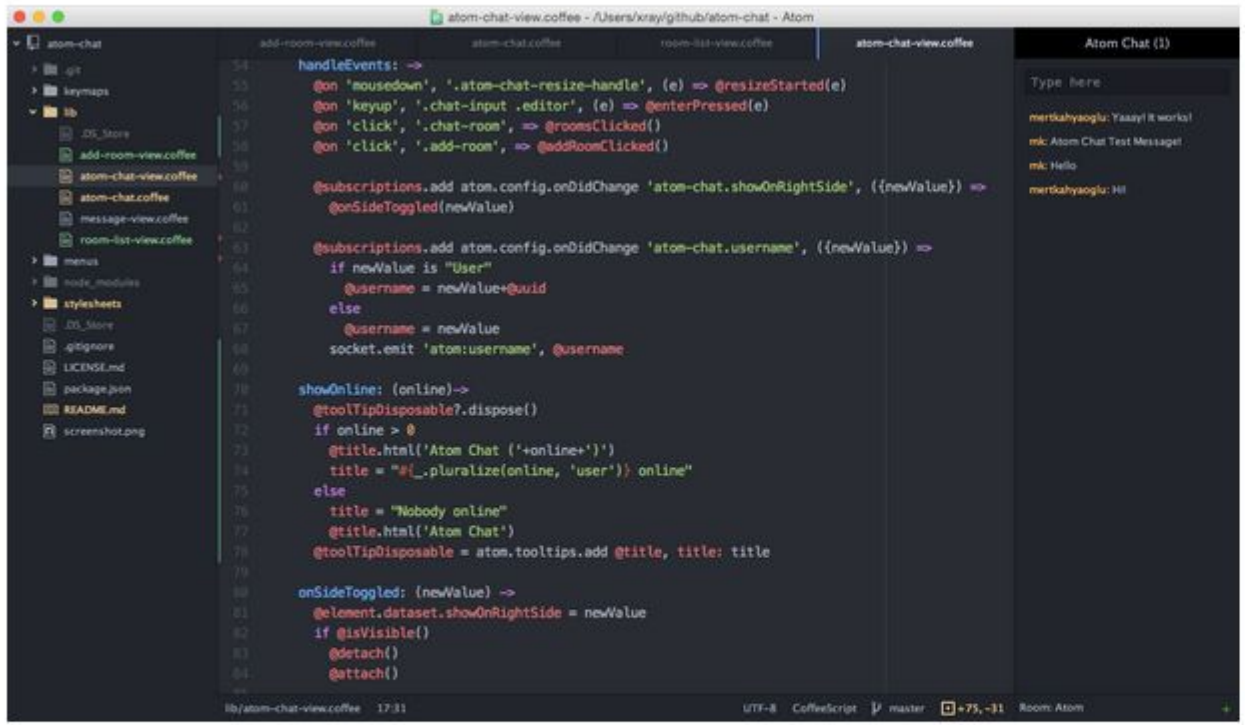


Sublime Text - рекомендуем для слабых компов



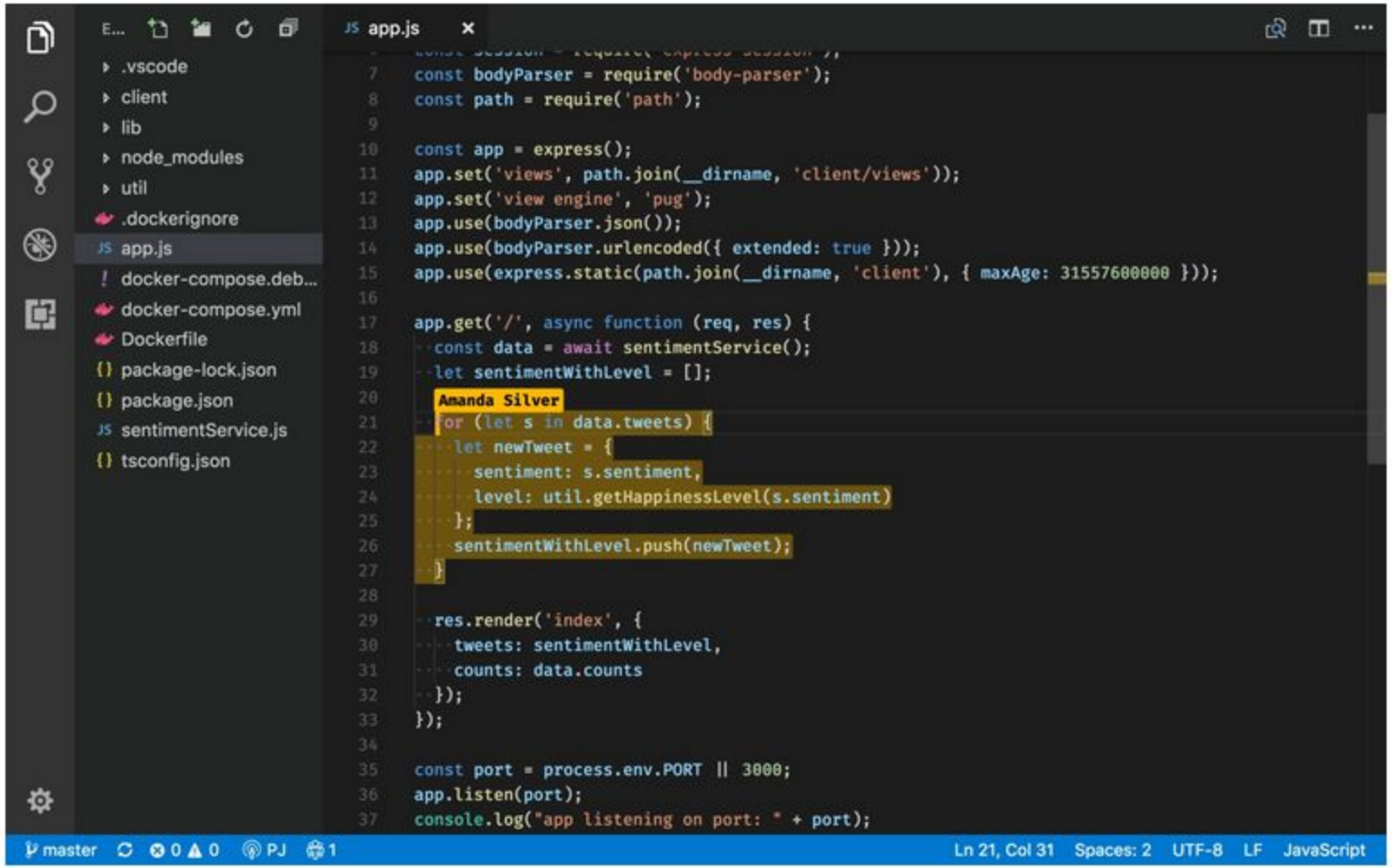
Самый популярный редактор кода у разработчиков на Python. Для веб-разработки тоже подходит. Довольно быстро работает, неплохо выглядит и кастомизируется, имеет несколько полезных плагинов. В целом неплох, но для веб-разработчика есть более подходящий софт. Рекомендуем использовать его только если у тебя слабый компьютер.

Atom - для хипстеров



Хороший редактор кода, заточенный под веб-разработку. Много тем оформления, плагинов. Работает на веб-технологиях, поэтому если ты планируешь развиваться дальше и изучать JavaScript, то в дальнейшем сможешь писать свои расширения. Его минус - скорость работы. Поэтому его использовать не рекомендуем.

Visual Studio Code - наша рекомендация



Не путай с Visual Studio. Редактор кода для веба от Microsoft. По сути, это более быстрый аналог Atom. Он имеет все те же самые плюсы, что и Atom, но работает ощутимо быстрее. Из минусов только майкрософтовый тоталитарный внешний вид, который легко изменить, установив другую тему оформления.

Отдельно про IDE

Редактор кода - не магическая лазерная пушка. По сути, это просто чуть более продвинутый блокнот, который подсвечивает код (по крайней мере, без плагинов). Мы намеренно не рассказываем про IDE, потому что считаем, что для обучения они не годятся. Почему?

- **Скорость работы. Они работают ощутимо медленнее.**
  Скорость работы инструментов - критически важная вещь для продуктивности разработчика.

- **Страшный интерфейс и куча функций, которые никогда не будут использованы.**
  Из-за тысячи возможностей ты знаешь в лучшем случае о 10% из них, тогда как в случае с редакторами ты сознательно ставишь каждый плагин и знаешь, что он делает.

- **Создают зону комфорта, из которой сложно выходить.**
  Разработчики, которые приучаются к IDE, часто неспособны работать в ином окружении - на другом компьютере или удалённом сервере; IDE'шники обычно не очень разбираются в консольных командах и неспособны автоматизировать и оптимизировать рабочее окружение под себя.

Элементы и их виды

Элементы - то, что создаётся тегами. Можно сказать, что теги это текстовое представление элементов. Элементы бывают двух видов:

Блочные элементы

Составляют структуру страницы.

Особенности:

- блоки располагаются друг под другом по вертикали
- запрещено вставлять блочный элемент внутрь строчного
- занимают всё допустимое пространство по ширине
- высота вычисляется автоматически, исходя из содержимого

Примеры:

- абзацы `<p>`
- списки: маркированные (с маркером) `<ul>` и нумерованные (с числами) `<ol>`
- заголовки: от первого уровня `<h1>` до шестого уровня `<h6>`
- статьи `<article>`
- разделы `<section>`
- длинные цитаты `<blockquote>`
- блоки общего назначения `<div>`

Строчные элементы

Используются для форматирования текстовых фрагментов. Обычно содержат одно или несколько слов.

Особенности:

- элементы, идущие подряд, располагаются на одной строке и переносятся на другую при необходимости
- внутрь допустимо вставлять текст или другие строчные элементы, помещать блочные элементы - запрещено

Примеры:

- ссылки `<a>`
- выделенные слова `<em>`
- важные слова `<strong>`
- короткие цитаты `<q>`
- аббревиатуры `<abbr>`

Списки

В HTML существует три вида списков:

Маркированный

Список из неупорядоченных элементов.

Состоит из двух тегов:

- `<ul>` (*unordered list*) - тег начала и конца списка
- `<li>` (*list item*) - пункт списка

Пример:

```
Список ингредиентов:
<ul>
  <li>Картошка</li>
  <li>Морковка</li>
  <li>Свекла</li>
</ul>
```

Нумерованный

Упорядоченный список, каждый пункт имеет свой номер.

Состоит из двух тегов:

- `<ol>` (*ordered list*) - тег начала и конца списка
- `<li>` (*list item*) - пункт списка

Пример:

```
Инструкция по приготовлению:
<ol>
  <li>Довести воду до кипения</li>
  <li>Засыпать ингредиенты</li>
  <li>Варить 10 минут</li>
</ol>
```

Список определений

Используются для создания списка терминов и их определений. В общем случае, каждый пункт — это пара "имя/значение".

Состоит из трёх тегов:

- `<dl>` (*description list*) - тег начала и конца списка
- `<dt>` (*term*) - термин
- `<dd>` (*description*) - определение

Пример:

```
<dl>
  <dt>Гаспачо</dt><dd>лёгкий холодный суп из перетёртых в пюре свежих овощей</dd>
  <dt>Том-ям</dt><dd>кисло-острый суп на основе куриного бульона с креветками, курицей, рыбой или другими
морепродуктами</dd>
  <dt>Борщ</dt><dd>разновидность супа на основе свёклы, которая придаёт борщу характерный красный цвет</dd>
</dl>
```

Вложенные списки

Списки можно вкладывать друг в друга. Для этого в элемент списка вставляется ещё один список. Можно вкладывать списки любого вида друг в друга в любой последовательности.

Пример:

```
<ul>
  <li>Омлет
    <ul>
      <li>Молоко</li>
      <li>Яйца</li>
    </ul>
  </li>
  <li>Блинчики
    <ul>
      <li>Молоко</li>
      <li>Яйца</li>
      <li>Мука</li>
      <li>Сахар</li>
      <li>Соль</li>
      <li>Разрыхлитель</li>
    </ul>
  </li>
  <li>Торт</li>
  <li>Пирог</li>
</ul>
```

Изображения

Для добавления изображения используется тег `<img>`. Это одинарный тег. Вот его основные атрибуты:

- `src` - ссылка на картинку
- `alt` - текст, который отображается вместо картинки, если она не загрузилась
- `title` - текст, который отображается при наведении мыши на картинку
- `width` - ширина картинки в пикселях
- `height` - высота картинки в пикселях

Пример:

```
<img
  src="http://example.com/cat.jpg"
  title="Мурка"
  alt="Рыжая кошка валяется в снегу"
  width="640"
  height="480"
>
```

Семантичные изображения с подписью в HTML 5

В HTML 5 появились теги для оформления объектов с подписями - figure и figcaption. Если твоей картинке нужна подпись - пользуйся ими. Пример кода:

```
<figure>
  <img src="https://www.google.ru/images/branding/googlelogo/2x/googlelogo_color_120x44dp.png">
  <figcaption>
    Лого гугла от 2015 года
  </figcaption>
</figure>
```

Результат:

Ссылки и адреса

Ты уже знаком со ссылками:

```
<a href="https://google.com/">Google</a>
```

Повторим: для создания ссылки необходимо использовать тег <a>. Атрибут **href** указывает адрес, по которому будет совершён переход.

Адреса бывают двух видов:

Абсолютные адреса

Абсолютный адрес, записанный в полной форме. Например,

```
https://google.com/doodles
```

Давай разберём этот адрес:

- https - так называемая «схема», обычно это название протокола. HTTPS - защищённая версия HTTP
- google.com - доменное имя сайта
- /doodles - путь (директория) внутри сайта

Ещё пример:

```
file:///C:/Users/admin/Desktop/Новая%20папка/image.jpg
```

- file - схема URI, предназначенная для того, чтобы адресовать файлы на локальном компьютере или в локальной сети (подробнее на Википедии)
- */C:/Users/admin/Desktop/Новая%20папка/image.jpg* - путь до файла. *%20* - код пробела в URI-кодировании.

Относительные адреса

Относительный - сокращённый адрес. В таком адресе начальная часть опущена и браузер использует текущий адрес для определения полного адреса. Примеры:

- //google.com - ссылка на домен в текущем протоколе: если мы находимся по адресу, который начинается с http, то ссылка будет вести на http://google.com
- /sheets - ссылка на путь внутри текущего домена: если мы находимся на http://google.com, то ссылка будет вести на http://google.com/sheets, а если на http://facebook.com, то на http://facebook.com/sheets.
- page2 - ссылка на путь внутри текущей директории: если мы находимся на http://site.com/routes/page1, то попадём на http://site.com/routes/page2

Пример использования относительного адреса

Файловая система:

```
Новая папка
├──img
│   ├──kisa.jpg
│   └──kot.png
├──index.html
└──style.css
```

Код в index.html:

```
...
<link rel="stylesheet" href="style.css">
...
<img src="img/kisa.jpg">
<img src="img/kot.png">
```

При выполнении заданий с использованием файлов - картинок, шрифтов, веб-страниц, которые находятся локально (то есть у тебя на устройстве), используй относительные ссылки. Потому что при загрузке кода на сервер, ссылке вроде file:///C:/Users/admin /Desktop/Новая%20папка/image.jpg перестанут работать.

Якоря

Ссылки могут ссылаться не только на страницу, но ещё и на конкретное место на странице. Такие ссылки называются *якорными,* а места, на которые они ссылаются -- *якорями.*

Якорному элементу нужно прописать атрибут **id** с именем якоря, например:

```
<h1 id="anchor">Якорь</h1>
```

Теперь, чтобы сослаться на этот якорь, нужно использовать вот такую ссылку:

```
<a href="#anchor">Перейти к якорю</a>
```

При нажатии на ссылку произойдёт переход к нужному месту страницы, а в адресной строке к адресу страницы добавится адрес якоря **#anchor**

Якоря могут использоваться как в относительных ссылках, как в примере выше, так и в абсолютных, например: http://example.com/page#anchor

Таблицы

Таблицы в HTML создаются при помощи тега <table>.

Внутри него размещают **строки таблицы** <tr> (*table row*)

Внутри строк помещают **ячейки строки** <td> (*table data*).

Тегом <th> (*table header*) размечаются **заголовочные ячейки**. Он отличается от <td> тем, что его содержимое будет выделено полужирным и выровнено по центру.

Пример таблицы:

```
<table>
  <tr>
    <th>Имя</th>
    <th>Возраст</th>
    <th>Суперспособность</th>
  </tr>
  <tr>
    <td>Логан</td>
    <td>186</td>
    <td>Повышенная способность к регенерации</td>
  </tr>
  <tr>
    <td>Профессор икс</td>
          <td>94</td>
    <td>Чтение мыслей, вызов иллюзий, временного паралича, способность останавливать время</td>
  </tr>
</table>
```

Результат:

Имя	Возраст	Суперспособность
Логан	186	Повышенная способность к регенерации
Профессор икс	94	Чтение мыслей, вызов иллюзий, временного паралича, способность останавливать время

По умолчанию границы таблиц не отображаются. Чтобы включить отображение границ, нужно использовать атрибут **border**.

Важно!!!

Атрибут border устарел и его применение в HTML5 не приветствуется. Используйте данный атрибут только пока знакомитесь с таблицами. Данный атрибут заменяет свойство border в CSS.

Пример:

```
<table border="1"> ... </table>
```

thead, tbody, tfoot, caption

Существует возможность группировать строки таблицы тегами <thead>, <tfoot>, <tbody>. Они не являются обязательными, но их рекомендуется использовать. Они дают частям таблицы семантический смысл.

<thead> - для заголовка,
<tfoot> - нижний колонтитул таблицы,
<tbody> - основной контент таблицы

Интересный факт: <tfoot> в коде можно расположить перед <tbody> или после него, но браузеры всегда выводят его в конце таблицы.

<thead> и <tfoot> можно использовать только по одному разу в одной таблице, а количество <tbody> может быть любым. С помощью нескольких <tbody> можно разделить контент таблицы на смысловые части. Например, так в таблице могут быть представлены данные за несколько лет, и каждый год будет вынесен в отдельный <tbody>.

Название таблицы можно разместить в теге <caption>. Этот тег располагают внутри тега <table> в самом начале, перед тегами <thead>, <tfoot>, <tbody>

Пример использования тегов

```
<table border="1">
  <caption>Сотрудники отдела поддержки</caption>
  <thead>
    <tr>
      <th>Имя</th>
      <th>Должность</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Петя</td>
      <td>Менеджер</td>
    </tr>
    <tr>
      <td>Маша</td>
      <td>Разработчик</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Вася</td>
      <td>Девопс</td>
    </tr>
  </tfoot>
</table>
```

Результат:

Сотрудники отдела поддержки

Имя	Должность
Петя	Менеджер
Маша	Разработчик
Вася	Девопс

Объединение столбцов

Если количество столбцов в одной строке меньше, чем количество столбцов в другой строке таблицы, то в строке с меньшим количеством ячеек образуется пустое пространство.

Пример:

```
<table border="1">
  <tr>
    <td>Первая ячейка</td>
    <td>Вторая ячейка</td>
    <td>Третья ячейка</td>
  </tr>
  <tr>
    <td>Единственная ячейка</td>
  </tr>
</table>
```

Результат:

Первая ячейка	Вторая ячейка	Третья ячейка
Единственная ячейка		

Как видно, ячейка не занимает свободное пространство. Чтобы это исправить, нужно растянуть её вправо с помощью атрибута **colspan**, и значением указать количество столбцов, которое в сумме будет занимать ячейка.

Пример:

```
<table border="1">
  <tr>
    <td>Первая ячейка</td>
    <td>Вторая ячейка</td>
    <td>Третья ячейка</td>
  </tr>
  <tr>
    <td colspan="3">Единственная ячейка</td>
  </tr>
</table>
```

Результат:

Первая ячейка	Вторая ячейка	Третья ячейка
Единственная ячейка		

Объединение строк

Для объединения строк по аналогии со столбцами используется атрибут **rowspan**. Только в данном случае, ячейка растягивается вниз на указанное количество строк.

Пример:

```
<table border="1">
  <tr>
    <td rowspan="2">Левая ячейка</td>
    <td>Верхняя ячейка</td>
  </tr>
  <tr>
    <td>Нижняя ячейка</td>
  </tr>
</table>
```

Результат:

<table>
<tr><td rowspan="2">Левая ячейка</td><td>Верняя ячейка</td></tr>
<tr><td>Нижняя ячейка</td></tr>
</table>

Объединение строк и столбцов можно сочетать:

```
<table border="1">
  <tr>
    <td rowspan="2">Левая ячейка</td>
    <td>Средняя ячейка</td>
    <td>Правая ячейка</td>
  </tr>
  <tr>
    <td colspan="2">Нижняя ячейка</td>
  </tr>
</table>
```

Результат:

<table>
<tr><td rowspan="2">Левая ячейка</td><td>Средняя ячейка</td><td>Правая ячейка</td></tr>
<tr><td colspan="2">Нижняя ячейка</td></tr>
</table>

Формы

Формы используются для сбора информации, которую пользователь вводит в специально отведённые поля этой формы. Когда он введёт свои данные и нажмет кнопку «Отправить», все эти данные будут отправлены на сервер. Затем они будут обработаны и сервер отправит ответ пользователю. Существует два основных метода отправки данных: **GET и POST**.

GET

Когда ты вводишь в адресной строке браузера какой-либо адрес и переходишь по нему, ты отправляешь серверу запрос, называемый GET. В таком запросе данные могут отсутствовать, как здесь: https://www.google.ru/. А вот запрос https://www.google.ru/search?q=itc+stepik содержит в себе переменную q, которая имеет значение itc stepik. В данном случае запрос отправляется на адрес https://www.google.ru/search, а данные из полей и их названия идут после **?** через знак **&**. Предварительно данные кодируются в URL код, чтобы сервер не перепутал служебные символы (вроде **/**, или **&**) с частью запроса.

POST

Методом POST так же можно отправлять данные в URL. Но, в отличие от GET, он может иметь *тело* - специальную "коробочку", в которую можно положить данные, которые уйдут на сервер.

Этот метод обычно используется для отправки форм и загрузки файлов.

> *Кроме GET и POST существуют методы PUT, PATCH и DELETE. Они отличаются от POST только названием и разработчики выбирают тот или иной метод исходя из смысла действия.*

Создание формы

Элемент формы создаётся парным тегом `<form>`. Внутри размещаются сами поля, причем те, что расположены за пределами элемента, отправлены не будут. При необходимости в элементе формы можно использовать стороннюю разметку. Тип отображения по умолчанию block (блочный элемент). Данный тег имеет два обязательных атрибута: action и method.

В атрибуте action указывается ссылка на обработчик формы. Обработчиком формы является PHP файл (или файл, написанный на других языках программирования), который будет обрабатывать данные этой формы.

Атрибут method предназначен для указания метода отправки данных на сервер (GET или POST).

<input>

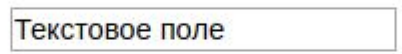
Поле задается одинарным тегом `<input>`. Данный тег стилизуется браузерами по-разному. Тип отображения по умолчанию inline (встроенный элемент), поэтому все поля отображаются в одной строке. Эта проблема решается подключением стилей, а пока мы до них не дошли - используй тег `<br>`.

Типы полей форм

От пользователя нам могут понадобиться самые разные данные. Для удобства их сбора появились типы полей форм. Они задаются через атрибут ***type***. Вот основные:

- *Текстовое поле.* Все просто: в атрибуте пишем ***text***.

```
<input type="text">
```



- *Ввод пароля*. Указывается значение ***password***.

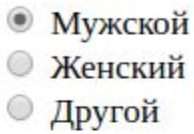
```
<input type="password">
```

- *Флаги (чекбоксы).* Они используются для выбора нескольких вариантов ответа и обозначаются в атрибуте как ***checkbox***. Подпись к галочке нужно писать отдельно.

```
<input type="checkbox">
```

- *Радио-кнопки.* Это выбор одного из нескольких вариантов. Для создания в атрибуте прописываем radio. Переключатели между собой нужно связывать: пишем атрибут name с одинаковыми значениями. Также необходим атрибут value.

```
<input type="radio" name="gender" value="male" checked> Мужской<br>
<input type="radio" name="gender" value="female"> Женский<br>
<input type="radio" name="gender" value="other"> Другой
```



Эти типы создают кнопки:

- *Загрузка файлов.* При значении атрибута file появляется кнопка загрузки.

```
<input type="file">
```

- *Сброс всех значений*. Для создания такой кнопки пишем reset. Чтобы все сработало, кнопка должна находиться внутри формы, в которой нужно сбросить значения.

```
<input type="reset">
```

- *Отправка.* Любая форма бессмысленна, если не отправляется на сервер. Значение для такой кнопки: submit.

```
<input type="submit">
```

- *Просто кнопка.* Чтобы создать кнопку без определенного действия указываем button.
-

```
<button>Нажми на меня</button>
```

Фух! Очень много этих полей, да? Осталось последнее - раскрывающийся список. Это тег `<select>`. У него есть несколько атрибутов: ***name, size, multiple***. Атрибут ***name*** работает как у переключателей. **Size** отвечает за то, сколько строк списка будет одновременно отображено. В качестве значения туда пишется целое положительное число. Атрибут ***multiple*** отвечает за то, можно ли выбрать сразу несколько вариантов из списка (как checkbox); это атрибут без значения.

Каждый элемент списка выделяется в тег `<option>`. Так же, как и с `<input type="checkbox">` и `<input type="radio">` нужно указывать атрибут value со значением, которое будет отправлено на сервер.

Новые типы полей

С появлением стандарта HTML5 добавились и новые типы полей форм. В первую очередь они стали гораздо удобнее для пользователей.

- **Ввод E-Mail**, *type="email"*. Текстовое поле, у которого на клавиатуре мобильных устройств появляется символ @.
- **Номер телефона**, *type="tel"*. На мобильных устройствах открывается клавиатура с числами.
- **Ввод ссылки**, *type="url"*.
- **Числовое поле**, *type="number"*. Помимо клавиатуры с цифрами появляется возможность переключать значения поля. Атрибуты **min** и **max** определяют нижнее и верхнее возможное значение, **step** - шаг изменения, а **value** - начальное значение.
- **Числовой ползунок**, *type="range"*. Появляются уже указанные атрибуты **min**, **max**, **step** и **value.**
- **Поиск**, *type="search"*. Google Chrome добавляет крестик для очистки введенной строки. На мобильных устройствах появляется кнопка поиска.

HTML form: <form>

```
<form action="destination URL">
  form controls
</form>
```

HTML

- атрибут required action задает URL страницы, которая будет обрабатывать данные этой формы.
- когда форма будет заполнена и отправлена, ее данные будут отправлены на адрес действия

Form example

```
<form action="http://www.google.com/search">
  <div>
    Let's search Google:
    <input name="q" />
    <input type="submit" />
  </div>
</form>
```

HTML

- Элементы формы можно помещать в контейнер, например div

Text fields: `<input>`

```
<input type="text" size="10" maxlength="8" /> NetID <br />
<input type="password" size="16" /> Password
<input type="submit" value="Log In" />
```

HTML

- input аттрибуты: `disabled`, `maxlength`, `readonly`, `size`, `value`
- `size` задает ширину
- `maxlength` ограничение вводимых символов

Text boxes: <textarea>

```
<textarea rows="4" cols="20">
Type your comments here.
</textarea>
```

HTML

- пример текста помещенного внутри тега textarea
- атрибуты rows и cols указать высоту/ширину в символах

Radio buttons: <input>

```
<input type="radio" name="cc" value="visa"
checked="checked" /> Visa
<input type="radio" name="cc" value="mastercard" />
MasterCard
<input type="radio" name="cc" value="amex" /> American
Express
```
HTML

- сгруппированы по имени атрибута (только один может быть проверен одновременно)
- необходимо указать значение для каждого из них

Text labels: <label>

```
<label><input type="radio" name="cc" value="visa"
checked="checked" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" />
MasterCard</label>
<label><input type="radio" name="cc" value="amex" />
American Express</label>
```

HTML

- Label связывает соседний текст с элементом управления, так что вы можете щелкнуть на текст, чтобы активировать элемент управления
- может использоваться с флажками или переключателями
- элемент label может оформлен при помощи CSS

Drop down lists: <select>, <option>

```HTML
<select name="favoritecharacter">
  <option>Frodo</option>
  <option>Bilbo</option>
  <option selected="selected">Gandalf</option>
  <option>Galandriel</option>
</select>
```

- элемент option определяет выбор в select

Using: `<select>` for lists

```
<select name="favoritecharacter[]" size="3"
multiple="multiple">
  <option>Frodo</option>
  <option>Bilbo</option>
  <option>Gandalf</option>
  <option>Galandriel</option>
  <option selected="selected">Aragorn</option>
</select>
```

HTML

- `multiple` позволяет выделить несколько элементов с Shift или Ctrl-клик
- необходимо объявить имя параметра с помощью [], если вы разрешаете множественный выбор