Лабораторная работа №1.
Создание одностраничных Web-проектов с разметкой текста с помощью HTML

Цель работы: создать с помощью текстового редактора Web-страницы с текстом в соответствие с заданиями и применить разметку средствами языка HTML.

Общая постановки задачи

Задание №1

Мы все рано или поздно учимся писать письма. Это хороший способ проверить наши навыки форматирования! В первом задании этой работы предполагается создание письма с применением форматирования с использованием гиперссылок, а также некоторых элементов тега <head>. Это письмо – ответ исследователя будущему PhD студенту о его заявлении на работу в университете.

Во ходе изучения порядка выполнения работы (см. ниже) необходимо найти решения следующим подзадачам задания:

1. Создать новый проект сайта с названием *Letter* в соответствие со стандартной файловой структурой сайта.
2. Скопировать из секции ниже текст письма, который необходимо отформатировать, и вставить его в тело документа в файле *index.html* созданного проекта сайта:

Dr. Eleanor Gaye
Awesome Science faculty
University of Awesome
Bobtown, CA 99999,
USA
Tel: 123-456-7890
Email: no_reply@example.com
20 January 2016
Miss Eileen Dover
4321 Cliff Top Edge
Dover, CT9 XXX
UK
Re: Eileen Dover university application
Dear Eileen,
Thank you for your recent application to join us at the University of Awesome's science faculty to study as part of your PhD next year. I will answer your questions one by one, in the following sections.
Starting dates
We are happy to accommodate you starting your study with us at any time, however it would suit us better if you could start at the beginning of a semester; the start dates for each one are as follows:
First semester: 9 September 2016
Second semester: 15 January 2017
Third semester: 2 May 2017
Please let me know if this is ok, and if so which start date you would prefer.
You can find more information about important university dates on our website.
Subjects of study
At the Awesome Science Faculty, we have a pretty open-minded research facility — as long as the subjects fall somewhere in the realm of science and technology. You seem like an intelligent, dedicated researcher, and just the kind of person we'd like to have on our team.
Saying that, of the ideas you submitted we were most intrigued by are as follows, in order of priority:
Turning H2O into wine, and the health benefits of Resveratrol (C14H12O3.)

Measuring the effect on performance of funk bassplayers at temperatures exceeding 30°C (86°F), when the audience size exponentially increases (effect of 3 × 103 increasing to 3 × 104.)
HTML and CSS constructs for representing musical scores.
So please can you provide more information on each of these subjects, including how long you'd expect the research to take, required staff and other resources, and anything else you think we'd need to know? Thanks.
Exotic dance moves
Yes, you are right! As part of my post-doctorate work, I did study exotic tribal dances. To answer your question, my favourite dances are as follows, with definitions:
Polynesian chicken dance
A little known but very influential dance dating back as far as 300BC, a whole village would dance around in a circle like chickens, to encourage their livestock to be "fruitful".
Icelandic brownian shuffle
Before the Icelanders developed fire as a means of getting warm, they used to practice this dance, which involved huddling close together in a circle on the floor, and shuffling their bodies around in imperceptibly tiny, very rapid movements. One of my fellow students used to say that he thought this dance inspired modern styles such as Twerking.
Arctic robot dance
An interesting example of historic misinformation, English explorers in the 1960s believed to have discovered a new dance style characterized by "robotic", stilted movements, being practiced by inhabitants of Northern Alaska and Canada. Later on however it was discovered that they were just moving like this because they were really cold.
For more of my research, see my exotic dance research page.
Yours sincerely,
Dr Eleanor Gaye
University of Awesome motto: "Be awesome to each other." -- The memoirs of Bill S Preston, Esq

2. Создать в папке styles пустой файл с именем *main* и расширением *.css*.
3. Скопировать из секции ниже и вставить текст CSS-стиля в файл *main.css:*

```
body {
  max-width: 800px;
  margin: 0 auto;
}

.sender-column {
  text-align: right;
}

h1 {
  font-size: 1.5em;
}

h2 {
  font-size: 1.3em;
}

p,ul,ol,dl,address {
  font-size: 1.1em;
}

p, li, dd, dt, address {
  line-height: 1.5;
}
```

4. Подключить созданный CSS-файл со стилем к HTML-документу с текстом письма (*index.html*).

5. Отформатировать письмо, которое должно быть размещено во внутренней сети университета, в соответствие с имеющимися требованиями.

Требования к блочным элементам / структуре:

- Необходимо корректно структурировать весь документ, включив в него элементы doctype, и <html>, <head> и <body>.
- Письмо в целом должно быть размечено используя параграфы и заголовки, за исключением следующих пунктов – один заголовок верхнего уровня (начинается на "Re:") и три заголовка второго уровня.
- Даты начала семестра, изучения предметов и экзотических танцев должны быть помечены используя соответствующие типы списков.
- Два адреса должны быть помещены внутри элементов <address>. Каждая строка адреса должна находиться на новой строке, но не быть новым параграфом.

Требования к строчным элементам:

- Имена отправителя и получателя (как и "Tel" и "Email") должны быть выделены жирным.
- Четырем датам в документе необходимо выбрать правильные элементы содержащие машинно-читаемые даты.
- Первый адрес и первая дата в письме должны иметь атрибут class со значением "sender-column"; CSS стиль, который вы добавите позже, позволит выравнять по правому боку, как оно и должно быть в классической разметке письма.
- Пять акронимов/аббревиатур в главном тексте письма должны быть размечены, чтобы предоставить подсказки для каждого акронима/аббревиатуры.
- Шесть под/надстрочных элементов должны быть оформлены корректно в химической формуле, как и числа 10^3 и 10^4 (степень числа должна быть над числом).
- Для разметки символов градуса и умножения воспользуйтесь справочниками в интернет.
- Постарайтесь выделить как минимум два нужных по смыслу слова в тексте жирным.
- Есть два места, где следует разместить гиперссылки; добавьте нужные ссылки с заголовками. В качестве адреса для ссылок используйте http://example.com.
- Девиз университета и цитата должны быть размечены соответствующими элементами.
- Требования к заголовку документа:
- Кодировка документа должна быть указана как utf-8 с использованием соответствующего мета-тега.
- Автор письма должен быть указан в соответствующем мета-теге.
- Предоставленный CSS должен быть включён в соответствующий тег.

Полученная страница с письмом должна иметь следующий вид:

Dr. Eleanor Gaye
Awesome Science faculty
University of Awesome
Bobtown, CA 99999,
USA
Tel: 123-456-7890
Email: no_reply@example.com

20 January 2016

Miss Eileen Dover
4321 Cliff Top Edge
Dover, CT9 XXX
UK

Re: Eileen Dover university application

Dear Eileen,

Thank you for your recent application to join us at the University of Awesome's science faculty to study as part of your PhD next year. I will answer your questions one by one, in the following sections.

Starting dates

We are happy to accomodate you starting your study with us at any time, however it would suit us better if you could start at the beginning of a semester; the start dates for each one are as follows:

- First semester: 9 September 2016
- Second semester: 15 January 2017
- Third semester: 2 May 2017

Please let me know if this is ok, and if so which start date you would prefer.

You can find more information about important university dates on our website.

Subjects of study

At the Awesome Science Faculty, we have a pretty open-minded research facility — as long as the subjects fall somewhere in the realm of science and technology. You seem like an intelligent, dedicated researcher, and just the kind of person we'd like to have on our team. Saying that, of the ideas you submitted we were most intrigued by are as follows, in order of priority:

1. Turning H_2O into wine, and the health benefits of Resveratrol ($C_{14}H_{12}O_3$.)
2. Measuring the effect on performance of funk bassplayers at temperatures exceeding 30°C (86°F), when the audience size exponentially increases (effect of $3 \times 10^3 > 3 \times 10^4$.)
3. HTML and CSS constructs for representing musical scores.

So please can you provide more information on each of these subjects, including how long you'd expect the research to take, required staff and other resources, and anything else you think we'd need to know? Thanks.

Exotic dance moves

Yes, you are right! As part of my post-doctorate work, I *did* study exotic tribal dances. To answer your question, my favourite dances are as follows, with definitions:

Polynesian chicken dance
: A little known but *very* influential dance dating back as far as 300BC, a whole village would dance around in a circle like chickens, to encourage their livestock or be "fruitful".

Icelandic brownian shuffle
: Before the Icelanders developed fire as a means of getting warm, they used to practice this dance, which involved huddling close together in a circle on the floor, and shuffling their bodies around in imperceptably tiny, very rapid movements. One of my fellow students used to say that he thought this dance inspired modern styles such as Twerking.

Arctic robot dance
: An interesting example of historic misinformation, English explorers in the 1960s believed to have discovered a new dance style characterised by "robotic", stilted movements, being practiced by inhabitants of Northern Alaska and Canada. Later on however it was discovered that they were just moving like this because they were really cold.

For more of my research, see my exotic dance research page.

Yours sincerely,

Dr Eleanor Gaye

University of Awesome motto: “Be excellent to each other.” -- *Bill S Preston, Esq*

Задание №2

Вы работаете в школе. В настоящее время ваши ученики изучают планеты солнечной системы, и вы хотите обеспечить их наглядным пособием для поиска фактов и данных о планетах. Таблица HTML была бы идеальным вариантом – вам необходимо взять необработанные данные, которые у вас есть, и превратить их в таблицу, следуя нижеприведенным инструкциям.

Готовая таблица должна выглядеть так:

Data about the planets of our solar system (Planetary facts taken from Nasa's Planetary Fact Sheet - Metric.

		Name	Mass (10^{24}kg)	Diameter (km)	Density (kg/m^3)	Gravity (m/s^2)	Length of day (hours)	Distance from Sun (10^6km)	Mean temperature (°C)	Number of moons	Notes
Terrestrial planets		Mercury	0.330	4,879	5427	3.7	4222.6	57.9	167	0	Closest to the Sun
		Venus	4.87	12,104	5243	8.9	2802.0	108.2	464	0	
		Earth	5.97	12,756	5514	9.8	24.0	149.6	15	1	Our world
		Mars	0.642	6,792	3933	3.7	24.7	227.9	-65	2	The red planet
Jovian planets	Gas giants	Jupiter	1898	142,984	1326	23.1	9.9	778.6	-110	67	The largest planet
		Saturn	568	120,536	687	9.0	10.7	1433.5	-140	62	
	Ice giants	Uranus	86.8	51,118	1271	8.7	17.2	2872.5	-195	27	
		Neptune	102	49,528	1638	11.0	16.1	4495.1	-200	14	
Dwarf planets		Pluto	0.0146	2,370	2095	0.7	153.3	5906.4	-225	5	Declassified as a planet in 2006, but this remains controversial.

Все данные, которые необходимы для выполнения задания, находятся в секции ниже.

Rows

Terrestrial planets

Mercury 0.330 4,879 5427 3.7 4222.6 57.9 167 0 Closest to the Sun

Venus 4.87 12,104 5243 8.9 2802.0 108.2 464 0

Earth 5.97 12,756 5514 9.8 24.0 149.6 15 1 Our world

Mars 0.642 6,792 3933 3.7 24.7 227.9 -65 2 The red planet

Jovian planets

Gas giants

Jupiter 1898 142,984 1326 23.1 9.9 778.6 -110 67 The largest planet

Saturn 568 120,536 687 9.0 10.7 1433.5 -140 62

Ice giants

Uranus 86.8 51,118 1271 8.7 17.2 2872.5 -195 27

Neptune 102 49,528 1638 11.0 16.1 4495.1 -200 14

Dwarf planets*

Pluto 0.0146 2,370 2095 0.7 153.3 5906.4 -225 5 Declassified as a planet in 2006, but this remains controversial.

Columns

Name

Mass (10²⁴kg)

Diameter (km)

Density (kg/m³)

Gravity (m/s²)

Length of day (hours)

Distance from Sun (10⁶km)

Mean temperature (°C)

Number of moons

Notes

Caption
Data about the planets of our solar system (Planetary facts taken from Nasa's Planetary Fact Sheet - Metric).

Во ходе изучения порядка выполнения работы (см. ниже) необходимо найти решения следующим подзадачам задания:

1. Создать новый проект сайта с названием *Planets* в соответствие со стандартной файловой структурой сайта.
2. В теле документа в файле *index.html* созданного проекта сайта создать таблицу с внешним контейнером, заголовком и телом таблицы. Нижний колонтитул (footer) не нужен для этого примера.
3. Добавить подпись к таблице ("Caption" в текстовой секции выше).
4. Добавить строку в заголовок таблицы, содержащую все заголовки столбцов ("Columns" в текстовой секции выше).
5. Создать все строки содержимого внутри тела таблицы, помня, что все заголовки строк должны быть *семантическими*.
6. Убедиться, что весь контент помещен в нужные ячейки – в исходных данных каждая строка данных о планете отображается рядом со связанной с ней планетой.
7. Добавить атрибуты, чтобы заголовки строк и столбцов были однозначно связаны со строками, столбцами или группами строк, для которых они выступают в качестве заголовков.
8. Добавить черную рамку вокруг столбца, который содержит все заголовки строк с именами планет.

Порядок выполнения работы

Для создания Web-страницы (Web-сайта) на настольном компьютере (Windows, Mac или Linux) необходимо иметь ряд профессиональных инструментов:

- **Текстовый редактор** для создания кода. Это может быть простой редактор Visual Studio Code, Notepad++, Sublime Text, Atom, GNU Emacs, VIM или гибридный редактор Dreamweaver or WebStorm. Офисные редакторы не подходят для этого, поскольку они зависят от скрытых элементов, которые мешают движкам рендеринга, используемым веб-браузерами.
- **Веб-браузеры** для тестирования кода. В настоящее время наиболее часто используемые браузеры это Firefox, Chrome, Opera, Safari и Internet Explorer. Также необходимо тестировать сайт, как он работает на мобильных устройствах и на любых старых браузерах, которые целевая аудитория может все ещё использовать (например, IE 6-8). Lynx – текстовый веб-браузер для терминала, отлично подходит для того, чтобы увидеть, как ваш сайт воспринимается слабовидящими пользователями.
- **Графический редактор** (The Gimp, Paint.NET, или Photoshop) нужен, чтобы создавать изображения для веб-страниц.
- **Система контроля версий** – для управления файлами на сервере, для совместной работы над проектом с командой, обмена кодом и избегания редакторских конфликтов. На сегодняшний день Git является наиболее популярным

инструментом контроля версий. Репозиторий кода Github, основанный на Git, также является очень популярным.

– **FTP программа** нужна, чтобы загружать веб-страницы на сервер для публичного просмотра. Существует множество таких доступных программ: Cyberduck, Fetch, и FileZilla.

– Шаблоны, библиотеки, фреймворки и т. д., чтобы ускорить написание общей функциональности.

Минимальный набор предполагает наличие текстового редактора и некоторых современных веб-браузеров.

На каждом компьютере уже есть базовый текстовый редактор. По умолчанию Windows включает Блокнот, Mac OS X поставляется с TextEdit. Linux-дистрибутивы могут иметь разные программы. Так Ubuntu поставляется с gedit по умолчанию.

Для веб-разработки, понадобится больше, чем могут предложить Notepad или TextEdit. Рекомендуется начать с Visual Studio Code, который является бесплатным редактором, который предлагает предварительный просмотр и подсказки во время написания кода.

В начале обучения достаточно установить только пару настольных веб-браузеров, чтобы проверять код. В соответствие с имеющейся операционной системой можно выбрать несколько браузеров из предложенных в списке:

– Linux: Firefox, Chrome, Opera, Brave.

– Windows: Firefox, Chrome, Opera, MS Edge, Brave.

– Mac: Firefox, Chrome, Opera, Safari, Brave (Safari поставляется с iOS и OS X по умолчанию)

Веб-сайт (в том числе и одностраничный) состоит из множества файлов: текстового контента, кода, стилей, медиа-контента, и так далее. Когда вы создаете веб-сайт, вы должны собрать эти файлы в рациональную структуру на вашем локальном компьютере. Необходимо убедиться, что файлы могут «общаться» друг с другом, и весь контент выглядит правильно, прежде чем, загружать их на сервер.

Когда вы работаете с веб-сайтом локально на своем компьютере, необходимо держать все связанные файлы в одной папке, которая отражает файловую структуру опубликованного в будущем веб-сайта на сервере. Эта папка может располагаться где угодно, но необходимо разместить её там, где будет легко найти.

1. Выберите место для хранения проектов веб-сайтов. Здесь, создайте новую папку с именем web-projects (или аналогичной). Это то место, где будут располагаться все проекты сайтов.

2. Внутри этой первой папки, создайте другую папку для хранения первого веб-сайта. Назовите ее test-site (или как-то более творчески).

Называйте папки и файлы полностью в нижнем регистре без пробелов. Это необходимо потому что:

1. Многие компьютеры, в частности веб-серверы, чувствительны к регистру. Так, например, если вы положили изображение на свой веб-сайт в test-site/MyImage.jpg, а затем в другом файле вы пытаетесь вызвать изображение как test-site/myimage.jpg, это может не сработать.

2. Браузеры, веб-серверы и языки программирования не обрабатывают пробелы последовательно. Например, если вы используете пробелы в имени файла,

некоторые системы могут отнестись к имени файла как к двум именам файлов. Некоторые серверы заменяют пробелы в вашем имени файла на "%20" (символьный код для пробелов в URI), в результате чего все ваши ссылки будут сломаны. Лучше разделять слова дефисами, чем нижними подчеркиваниями: my-file.html лучше чем my_file.html.

Говоря простым языком, необходимо использовать дефис для имен файлов. Поисковая система Google рассматривает дефис как разделитель слов, но не относится к подчеркиванию таким образом. По этим причинам, лучше всего приобрести привычку писать названия ваших папок и файлов в нижнем регистре без пробелов, разделяя слова дефисами, по крайней мере, пока не поймете, что делаете. Так в будущем вы столкнетесь с меньшим количеством проблем.

Наиболее распространенные вещи, присутствующие в любом проекте сайта: индексный файл HTML и папки, содержащие изображения, файлы стилей и файлы скриптов:

1. **index.html** – файл обычно содержит контент домашней страницы (текст и изображения), которые люди видят, когда они впервые попадают на сайт. Используя текстовый редактор, создайте новый файл с именем index.html сохраните его прямо внутри вашей папки test-site.
2. **Папка images** – будет содержать все изображения, которые используются на сайте. Создайте папку с именем images внутри папки test-site.
3. **Папка styles** – будет содержать CSS код, используемый для стилизации контента (например, настройка текста и цвета фона). Создайте папку с именем styles внутри папки test-site.
4. **Папка scripts** – будет содержать весь JavaScript код, используемый для добавления интерактивных функций на сайте (например, кнопки, которые загружают данные при клике). Создайте папку с именем scripts внутри вашей папки test-site.

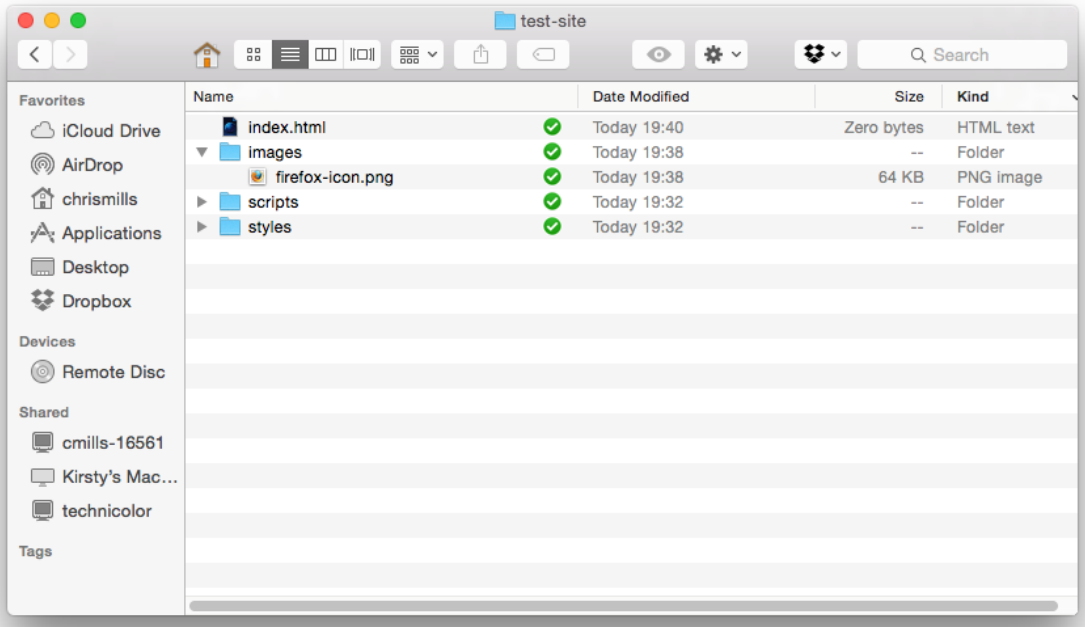
На компьютерах под управлением Windows у вас могут возникнуть проблемы с отображением имен файлов, поскольку у Windows есть опция **Скрывать расширения для известных типов файлов**, включенная по умолчанию. Обычно вы можете отключить ее, перейдя в проводник, выбрать вариант **Свойства папки...** и снять флажок **Скрывать расширения для зарегистрированных типов файлов**, затем щёлкнуть **OK**.

Для того, чтобы файлы общались друг с другом, нужно указать файлам пути друг к другу – обычно один файл знает, где находится другой. Некоторые общие правила о путях к файлам:

- Для ссылки на целевой файл в той же директории, что и вызывающий HTML файл, просто используйте имя файла, например, my-image.jpg.
- Для ссылки на файл в поддиректории, напишите имя директории в начале пути, плюс косую черту (forwardslash, слеш), например: subdirectory/my-image.jpg.
- Для ссылки на целевой файл в директории **выше** вызывающего HTML файла, напишите две точки. Например, если index.html находится внутри подпапки test-site, а my-image.png - внутри test-site, вы можете обратиться к my-image.png из index.html, используя ../my-image.png.
- Вы можете комбинировать их так, как вам нравится, например ../subdirectory/another-subdirectory/my-image.png.

Файловая система Windows стремится использовать обратный слеш (backslash), а не косую черту (forwardslash), например C:\windows. Это не имеет значения, даже если вы разрабатываете веб-сайт на Windows, все равно можно использовать обычные слеши в вашем коде.

К настоящему моменту структура вашей папки должна выглядеть примерно так:



HTML (Hypertext Markup Language) – это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.

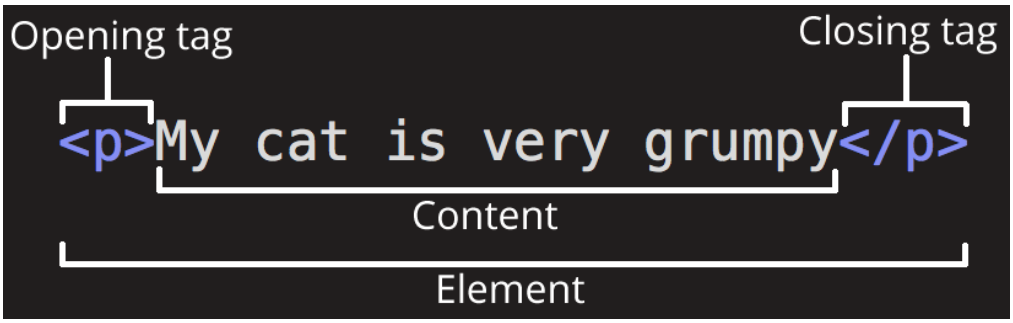
HTML не является языком программирования; это *язык разметки*, который используется, чтобы сообщать браузеру, как отображать открываемые веб-страницы. Он может быть сложным или простым, в зависимости от того, как хочет веб-разработчик. HTML состоит из ряда **элементов**, которые используются для создания вложений или обертки различных частей контента, чтобы заставить его отображаться или действовать определенным образом. Ограничивающие теги могут сделать слово или изображение ссылкой на что-то еще, могут выделить слова курсивом, сделать шрифт больше или меньше. Например, рассмотрим следующую строку контента:

```
Моя кошка очень раздражена
```

Если мы хотим, чтобы строка располагалась отдельно, можно сделать ее в виде абзаца, заключая его в теги абзаца:

```
<p>Моя кошка очень раздражена</p>
```

Рассмотрим элемент абзаца более подробно.



Главными частями нашего элемента являются:

1. **Открывающий тег (Opening tag)**: состоит из имени элемента (в данном случае, "p"), заключенного в открывающие и закрывающие **угловые скобки**. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае – где начинается абзац.
2. **Закрывающий тег (Closing tag):** то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае – где заканчивается абзац. Отсутствие закрывающего тега является одной из наиболее распространенных ошибок начинающих и может приводить к странным результатам.
3. **Контент (Content)**: содержимое элемента, которое в данном случае является просто текстом.
4. **Элемент(Element)**: открывающий, закрывающий теги и контент вместе составляют элемент.

Элементы также могут иметь атрибуты, которые выглядят так:

Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте. В данном случае, class – это *имя атрибута,* а editor-note – это *значение атрибута*. Класс позволяет дать элементу идентификационное имя, которое может позже использоваться, чтобы обращаться к элементу с информацией о стиле и прочих вещах.

Атрибут всегда должен иметь:

1. Пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).
2. Имя атрибута, за которым следует знак равенства.
3. Значение атрибута, заключенное с двух сторон в кавычки.

Иногда встречаются атрибуты, написанные без значения – это допустимо. Такие атрибуты называются булевые, и они могут иметь только одно значение, которое в основном совпадает с его именем. В качестве примера возьмем атрибут disabled, который можно назначить для формирования элементов ввода, если вы хотите, чтобы они были отключены (неактивны), так что пользователь не может вводить какие-либо данные в них.

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

Для краткости совершенно допустимо записывать их следующим образом (мы также для справки разместили не деактивированный элемент input, чтобы дать вам большее понимание происходящего):

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

<input type="text">
```

Структура веб-контента обычно предполагает расположение элементов внутри других элементов. Эта операция называется **вложением**. Если мы хотим заявить, что наша кошка **очень** раздражена, мы можем заключить слово "очень" в элемент , который указывает, что слово должно быть сильно акцентированно:

```
<p>Моя кошка <strong>очень</strong> раздражена.</p>
```

Однако нужно убедиться, что \ элементы правильно вложены: в примере выше мы открыли первым элемент <p>, затем элемент , потом мы должны закрыть сначала элемент , затем <p>. Приведенное ниже неверно:

```
<p>Моя кошка <strong>очень раздражена.</p></strong>
```

Элементы должны открываться и закрываться правильно, поэтому они явно располагаются внутри или снаружи друг друга. Если они перекрываются, как в примере выше, веб-браузер будет пытаться сделать наилучшее предположение на основе того, что вы пытались сказать, что может привести к неожиданным результатам. Так что не стоит этого делать!

Некоторые элементы не имеют контента, и называются **пустыми элементами**. Возьмем элемент <img>, который уже имеется в созданном ранее HTML:

```
<img src="images/firefox-icon.png" alt="Моё тестовое изображение">
```

Он содержит два атрибута, но не имеет закрывающего тега </img>, и никакого внутреннего контента, потому что элемент изображения не оборачивает контент для влияния на него. Его целью является вставка изображения в HTML страницу в нужном месте.

В спецификации HTML до версии 4.01 выделялось две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы – блочные и строчные элементы.

С помощью блочных элементов можно создавать структуру веб-страницы. Элементы блочного уровня формируют видимый блок на странице – они окажутся на новой строке после любого контента, который шёл до них, и любой контент после них также окажется на новой строке. Чаще всего элементами блочного уровня бывают структурные элементы страницы, представляющие собой, например, параграфы (абзацы), списки, меню навигации, футеры, или подвалы, и т.

п. Элементы блочного уровня не вкладываются в строчные элементы, но иногда могут вкладываться в другие элементы блочного уровня.

Строчные элементы – это те, которые содержатся в элементах блочного уровня и окружают только малые части содержимого документа, не целые абзацы и группировки контента. Строчные элементы не приводят к появлению новой строки в документе: они обычно встречаются внутри абзаца текста, например, элемент <a> (ссылка) или акцентирующие элементы вроде или . Строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов <area> и <img>).

В HTML5 эти понятия заменены более сложным набором категорий контента, согласно которым каждый HTML-элемент должен следовать правилам, определяющим, какой контент для него допустим.

Вернемся к коду, который мы записывали в наш index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Моя тестовая страница</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="Моё тестовое изображение">
  </body>
</html>
```

Здесь мы имеем:

– <!DOCTYPE html> – доктайп. В прошлом, когда HTML был молод (около 1991/1992), доктайпы должны были выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать, чтобы считаться хорошим HTML, что могло означать автоматическую проверку ошибок и другие полезные вещи. Однако в наши дни, никто не заботится об этом, и они на самом деле просто исторический артефакт, который должен быть включен для того, чтобы все работало правильно.

– <html></html> – элемент <html>. Этот элемент оборачивает весь контент на всей странице, и иногда известен как корневой элемент.

– <head></head> – элемент <head>. Этот элемент выступает в качестве контейнера для всего, что вы пожелаете включить на HTML страницу, но *не являющегося* контентом, который вы показываете пользователям вашей страницы. Элемент head HTML-документа не отображается на странице в веб-браузере. Он содержит такую информацию, как:

– заголовок (title) страницы;

– ссылки на файлы CSS (если вы хотите применить к вашему HTML стили CSS);

– ссылки на иконки;

– другие метаданные (данные о HTML: автор и важные ключевые слова, описывающие документ).

Задача <head> – хранить метаданные документа. В приведенном выше примере <head> совсем небольшой:

```
<head>
  <meta charset="utf-8">
  <title>Моя тестовая страница</title>
</head>
```

Однако на больших страницах блок <head> может быть довольно объемным. Попробуйте зайти на какие-нибудь из ваших любимых сайтов и посмотреть содержимое <head> с помощью инструментов разработчика.

Метаданные – данные, которые описывают данные. У HTML есть «официальное» место для метаданных документа – элемент <meta>. Существует множество разновидностей <meta>. Не станем пытаться охватить их все сразу, рассмотрим несколько самых популярных.

В заголовке примера выше есть следующая строка:

```
<meta charset="utf-8">
```

В этом элементе указана кодировка документа – набор символов, которые в нём можно использовать. utf-8 – универсальный набор символов, который включает почти все символы со всех языков человечества. Такая веб-страница сможет работать с любым языком. Установить эту кодировку всех веб-страницах, которые вы создаёте – отличная идея! Страница в такой кодировке прекрасно отображает как английские, так и японские символы:

Если использовать, скажем, кодировку ISO-8859-1 (набор символов для латиницы), текст страницы «испортится»:

Некоторые браузеры (например, Chrome) автоматически исправляют неправильную кодировку, поэтому, в зависимости от используемого вами браузера, вы можете не увидеть эту проблему. Несмотря на это вам всё равно необходимо указывать кодировку UTF-8 для вашей страницы, чтобы избежать возможных проблем в других браузерах.

У элементов <meta> часто есть атрибуты name и content:

- name – тип элемента, то есть какие именно метаданные он содержит.
- content – сами метаданные.

Два полезных элемента метаданных – указание автора страницы и краткое описание её содержимого. Рассмотрим эти элементы на примере:

```
<meta name="author" content="Крис Миллс">
<meta name="description" content="Задача MDN – в том, чтобы обучить новичков всему тому, что нужно им для разработки веб-сайтов и приложений.">
```

По указанному имени автора (author) можно найти человека, который написал страницу, и связаться с ним. Некоторые системы управления содержимым (CMS) автоматически обрабатывают эту информацию и делают её доступной для таких целей.

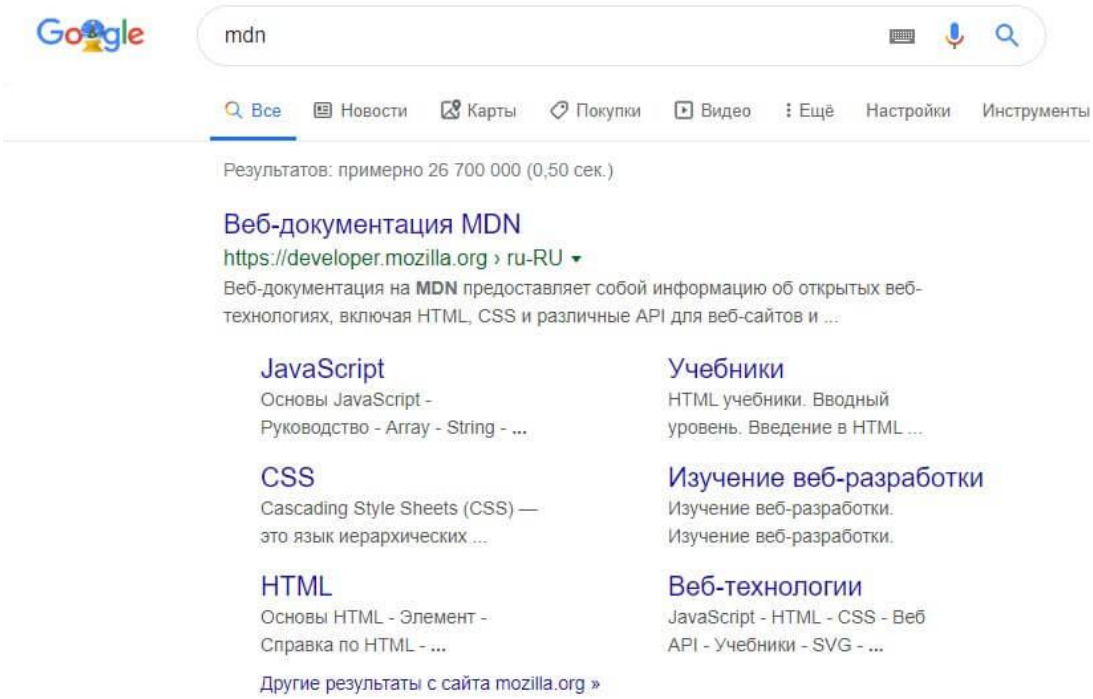
Краткое описание (description) содержимого страницы учитывается поисковыми системами при совпадении ключевых слов. Такое называют поисковой оптимизацией, или SEO.

Описание из <meta name="description"> используется на страницах поисковой выдачи. Проведём небольшое исследование такого сценария.

1. Перейдите на главную страницу Mozilla Developer Network.
2. Откройте исходный код страницы (кликните правой кнопкой мыши и выберите *Просмотреть код* в контекстном меню).
3. Найдите тег meta с описанием. Он выглядит так:

```
<meta name="description" content="Веб-документация на MDN предоставляет собой информацию об открытых веб-технологиях, включая HTML, CSS и различные API для веб-сайтов и прогрессивных веб-приложений. Также на сайте содержатся материалы для разработчиков о таких продуктах Mozilla, как Инструменты разработчика Firefox.">
```

4. Теперь найдите "Mozilla Developer Network" в своём поисковике (мы использовали Google). Обратите внимание, что описание и название из <meta> и <title> используется в результатах поиска, – мы не зря указали их!



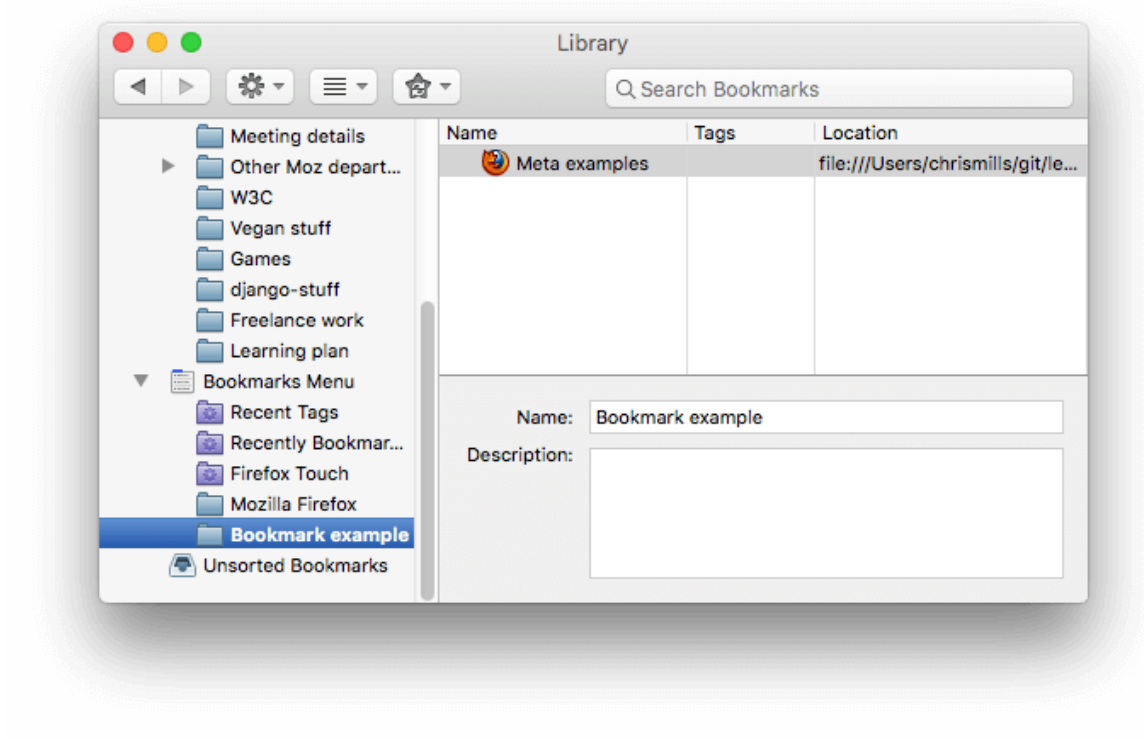
Google также показывает важные страницы MDN под ссылкой на главную страницу. Такие ссылки называются sitelinks, и их можно настроить через Google Search Console, чтобы пользователи могли сразу перейти к ним со страницы поиска.

Многие типы <meta> больше не используются. Так, поисковые системы больше не используют данные из элемента <meta type="keywords" content="ваши, ключевые, слова, введите, здесь">, в котором указывали ключевые слова, по которым можно найти страницу: спамеры засовывали туда все слова, какие могли придумать, чтобы их сайты почаще появлялись в поиске.

В сети вы найдете также другие типы метаданных. Многие из них – это собственные форматы, созданные для предоставления определенным сайтам (например, социальных сетей) специальной информации, которую они могут использовать.

Чтобы добавить своему сайту узнаваемости, можно указать в метаданных разные иконки.

Favicon, один из старожилов интернета, стал первой из таких иконок. Браузеры показывают её в заголовке вкладки и в списке избранных страниц.



Чтобы добавить на страницу favicon:

1. Сохраните изображение в формате .ico (многие браузеры поддерживают и в более привычных форматах, таких как .gif или .png) в папку со своим документом. Старые браузеры, например, Internet Explorer 6, поддерживают только формат .ico
2. Добавьте ссылку на иконку в <head> документа:

```
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```

Для разных устройств можно указывать разные иконки. Например, на главной странице MDN:

```
<!-- Для iPad 3 с Retina-экраном высокого разрешения: -->
<link rel="apple-touch-icon-precomposed" sizes="144x144"
href="https://developer.cdn.mozilla.net/static/img/favicon144.a6e4162070f4.png
">
<!-- Для iPhone с Retina-экраном высокого разрешения: -->
<link rel="apple-touch-icon-precomposed" sizes="114x114"
href="https://developer.cdn.mozilla.net/static/img/favicon114.0e9fabd44f85.png"
>
<!-- Для iPad первого и второго поколения: -->
<link rel="apple-touch-icon-precomposed" sizes="72x72"
href="https://developer.cdn.mozilla.net/static/img/favicon72.8ff9d87c82a0.png"
>
<!-- Для iPhone, iPod Touch без Retina и устройств с Android 2.1+: -->
<link rel="apple-touch-icon-precomposed"
href="https://developer.cdn.mozilla.net/static/img/favicon57.a2490b9a2d76.png"
>
```

```
<!-- Для других случаев - обычный favicon -->
<link rel="shortcut icon"
href="https://developer.cdn.mozilla.net/static/img/favicon32.e02854fdcf73.png"
>
```

В комментариях указано, для чего используется каждая иконка — например, при добавлении страницы на домашний экран iPad будет использована иконка в высоком разрешении.

Современные сайты используют CSS, чтобы выглядеть привлекательнее, и добавляют интерактивные функции через JavaScript: видеоплееры, карты, игры. Обычно связанные стили добавляют на страницу через элемент <link>, а скрипты – через элемент <script>.

- Элемент <link> помещают в заголовок документа. У него есть два атрибута: rel="stylesheet" показывает, что мы указываем *стиль* документа, а в href указан путь к файлу:

```
<link rel="stylesheet" href="my-css-file.css">
```

- Элемент <script> не обязательно находится в заголовке – на самом деле лучше поместить его в самом конце страницы, прямо перед закрывающем тегом </body>. Так браузер сначала отобразит саму страницу, а уже затем загрузит и запустит скрипт – иначе скрипт может обратиться к ещё не созданному элементу страницы и сломаться.

```
<script src="my-js-file.js"></script>
```

Элемент <script> кажется пустым, но это не всегда так, и указывать закрывающий тег обязательно. Вместо того чтобы ссылаться на внешний скрипт, код можно писать прямо внутри этого элемента – так можно не тратить время на загрузку отдельного скрипта, но зато не выйдет сослаться на один js-файл с нескольких страниц.

Наконец, стоит отметить, что вы можете (и действительно должны) установить язык для своей страницы. Это можно сделать, добавив атрибут lang в открывающий HTML-тег:

```
<html lang="en-US">
<html lang="ru">
```

Это полезно во многих случаях. Ваш HTML-документ будет более эффективно индексироваться поисковыми системами, если его язык установлен (что позволяет ему правильно отображаться в языковых результатах), и он полезен людям с нарушением зрения, которые используют программы, читающие страницы вслух (например, слово "шесть" пишется одинаково как на французском, так и на английском языках, но произносится по-разному.).

Можно также указать язык для части документа. Например, мы могли бы установить язык для части страницы на японском:

```
<p>Пример на японском: <span lang="jp">ご飯が熱い。</span>.</p>
```

Коды языков определены в стандарте ISO 639-1.

– <body></body> – элемент <body>. В нем содержится *весь* контент, который вы хотите показывать пользователям, когда они посещают вашу страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки или что-то еще.

– <title></title> – элемент <title>. Этот элемент устанавливает заголовок для вашей страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда вы добавляете ее в закладки/избранное.

В спецификации HTML5 некоторые правила разметки были ослаблены. В частности, использование элементов `<html>`, `<head>` и `<body>` уже не является обязательным для разметки HTML5. Тем не менее браузер все равно считает, что они существуют, в чем можно убедиться, просмотрев разметку веб-страницы в **Mozilla Firebug** или в **Google Chrome Inspector**.

Эти элементы всегда «подразумеваются», но если они будут использоваться в таблице CSS-стилей или в JavaScript-сценариях, то их надо прописывать в явном виде.

Также разрешается не использовать закрывающую обратную косую черту в элементах без содержимого.

В HTML символы <, >, ", ' и & являются специальными символами. Они часть HTML синтаксиса сами по себе. Если вы хотите использовать амперсанд или знак "меньше чем", и вам не нужно, чтобы эти знаки интерпретировалось браузером как часть разметки, нужно использовать ссылки-мнемоники – специальные коды, которые отображают спецсимволы, и могут быть использованы в точных позициях. Каждая ссылка-мнемоник начинается с амперсанда (&), и завершается точкой с запятой (;) (некоторые мнемоники приведены в таблице ниже).

Буквенный символ	**Символьный эквивалент**
<	&lt;
>	&gt;
"	&quot;
'	&apos;
&	&amp;

В следующем примере вы видите два абзаца, которые рассказывают о веб-технологиях:

```
<p>В HTML вы определяете параграф элементом <p>.</p>

<p>В HTML вы определяете параграф элементом &lt;p&gt;.</p>
```

В живом выводе ниже вы можете заметить, что первый абзац выводится неправильно, так как браузер считает, что второй элемент <p> является началом нового абзаца! Второй абзац нашего кода выводится правильно, потому что мы заменили угловые скобки на ссылки-мнемоники.

В HTML, как и в большинстве языков программирования, есть возможность писать комментарии в коде. Комментарии игнорируются обозревателем и не видны пользователю, их добавляют для того, чтобы пояснить, как работает написанный код, что делают отдельные его части и т. д. Такая практика полезна, если вы возвращаетесь к коду, который давно не видели или когда хотите передать его кому-то другому.

Чтобы превратить часть содержимого HTML-файла в комментарий, нужно поместить её в специальные маркеры <!—и -->, например:

```
<p> Меня нет в комментариях( </p>

<!-- <p>А теперь есть!</p> -->
```

Рассмотрим теперь некоторые из основных HTML элементов, которые используются для разметки текста.

Элементы заголовка позволяют указывать определенные части контента в качестве заголовков или подзаголовков. Точно так же, как книга имеет название, названия глав и подзаголовков, HTML документ может содержать то же самое. HTML включает шесть уровней заголовков <h1>–<h6>, хотя обычно используется не более 3-4 :

```
<h1>Мой главный заголовок</h1>
<h2>Мой заголовок верхнего уровня</h2>
<h3>Мой подзаголовок</h3>
<h4>Мой под-подзаголовок</h4>
```

Например, в рассказе <h1> будет представлять заглавие рассказа, <h2> обозначит название каждой главы, <h3> будет обозначать подзаголовки в каждой главе и так далее.

Всё это действительно зависит от вас – что именно будут представлять собой элементы, пока существует иерархия. Вам просто нужно иметь в виду несколько хороших правил при создании таких структур.

- Предпочтительнее использовать <h1> только один раз на странице – это заголовок самого верхнего уровня, и все остальные заголовки располагаются ниже его в иерархии.
- Убедитесь, что вы используете заголовки в правильном порядке в иерархии. Не используйте <h3> для создания подзаголовков при одновременном использовании <h2> для представления под-подзаголовков – это не имеет смысла и приведет к странным результатам.
- Из шести доступных уровней заголовка вы должны стремиться использовать не более трех на странице, если только вы не чувствуете, что это необходимо. Документы с большим количеством уровней (то есть с глубокой иерархией) становятся громоздкими и трудными для навигации. В таких случаях рекомендуется распределять контент по нескольким страницам, если это возможно.

Семантика проявляется всюду вокруг нас – мы полагаемся на опыт, который рассказывает нам, какова функция бытовых предметов; когда мы что-то видим, мы знаем, какова его функция. Так, например, мы ожидаем, что красный свет на

светофоре означает «стоп», а зеленый свет означает «идти». Жизнь станет очень сложной, если применяется неправильная семантика (какие-либо страны используют красный цвет для обозначения «идти»? Надеюсь, что нет.)

В подобном ключе нам нужно убедиться, что мы используем правильные элементы, придавая нашему контенту правильное значение, функцию или внешний вид. В этом контексте элемент <h1> также является семантическим элементом, который даёт тексту, который он обёртывает, роль (или значение) «заголовка верхнего уровня на вашей странице».

```
<h1>Это заголовок верхнего уровня</h1>
```

По умолчанию браузер придаст ему большой размер шрифта, чтобы он выглядел как заголовок (хотя вы можете стилизовать его как угодно, используя CSS). Что ещё более важно, его семантическое значение будет использоваться несколькими способами, например, поисковыми системами и программами чтения с экрана (как упоминалось выше).

С другой стороны, вы можете сделать любой элемент похожим на заголовок верхнего уровня. Рассмотрим следующее:

```
<span style="font-size: 32px; margin: 21px 0;">Это заголовок верхнего уровня?</span>
```

Это элемент . У него нет семантики. Вы используете его, когда хотите применить к контенту CSS (или сделать что-то с ним с помощью JavaScript), не придавая ему никакого дополнительного значения. Мы применили CSS, чтобы он выглядел как заголовок верхнего уровня, но поскольку он не имеет семантического значения, он не получит никаких дополнительных преимуществ, описанных выше. Рекомендуется использовать соответствующий элемент HTML на практике.

Теперь попробуйте добавить подходящее название для вашей HTML страницы, чуть выше элемента <img>.

Как было сказано раньше, элемент <p> предназначен для абзацев текста; вы будете использовать их регулярно при разметке текстового контента:

```
<p>Это одиночный абзац</p>
```

Добавьте свой образец текста в один или несколько абзацев, расположенных прямо под элементом <img>.

Большая часть веб-контента является списками и HTML имеет специальные элементы для них. Разметка списка всегда состоит по меньшей мере из двух элементов. Наиболее распространенными типами списков являются нумерованные и ненумерованные списки:

1. **Ненумерованные списки** – это списки, где порядок пунктов не имеет значения, как в списке покупок. Они оборачиваются в элемент <ul>.
2. **Нумерованные списки –** это списки, где порядок пунктов имеет значение, как в рецепте. Они оборачиваются в элемент <ol>.

Каждый пункт внутри списков располагается внутри элемента <li> (list item, элемент списка).

Например, если мы хотим включить часть следующего фрагмента абзаца в список:

```
<p>Mozilla, мы являемся мировым сообществом технологов, мыслителей и строителей, работающих вместе ... </p>
```

Мы могли бы изменить разметку на эту:

```
<p>Mozilla, мы являемся мировым сообществом</p>

<ul>
  <li>технологов</li>
  <li>мыслителей</li>
  <li>строителей</li>
</ul>

<p>работающих вместе ... </p>
```

Вполне нормально вложить один список в другой. Возможно, вы захотите, чтобы один список располагался внутри другого. Возьмем список из примера рецепта:

```
<ol>
  <li>Очистите чеснок от кожуры и крупно нарежьте.</li>
  <li>Удалите стебель и семена у перца; крупно нарежьте перец.</li>
  <li>Добавьте все ингредиенты в пищевой комбайн.</li>
  <li>Измельчите все ингридиенты до состояния пасты.</li>
  <li>Если вы хотите "грубый" хумус, измельчайте пару минут.</li>
  <li>Если вам нужен гладкий хумус, измельчайте дольше.</li>
</ol>
```

Поскольку последние две строки очень тесно связаны с тем, что было до них (они читаются как вспомогательные инструкции или варианты, которые подходят под этой маркой), может иметь смысл вложить их в свой собственный неупорядоченный список и поместить этот список внутри текущего. Это будет выглядеть так:

```
<ol>
  <li>Очистите чеснок от кожуры и крупно нарежьте.</li>
  <li>Удалите стебель и семена у перца; крупно нарежьте перец.</li>
  <li>Добавьте все ингредиенты в пищевой комбайн.</li>
  <li>Измельчите все ингридиенты до состояния пасты.
    <ul>
      <li>Если вы хотите "грубый" хумус, измельчайте пару минут.</li>
      <li>Если вам нужен гладкий хумус, измельчайте дольше.</li>
    </ul>
  </li>
```

```
</ol>
```

Попробуйте добавить упорядоченный или неупорядоченный список на свою страницу.

В обиходе мы часто подчеркиваем определённые слова, чтобы изменить смысл предложения и мы часто хотим отметить некоторые слова как важные или разные в некотором роде. HTML предоставляет различные семантические элементы, позволяющие нам добавлять текстовые материалы с такими эффектами, и в этом разделе мы рассмотрим несколько наиболее распространенных.

Когда мы хотим добавить акцент в разговорный язык, мы подчеркиваем определенные слова, тонко изменяя смысл того, что мы говорим. Точно так же на письменном языке мы склонны подчеркивать слова, выделяя их *курсивом*. Например, следующие два предложения имеют разные значения.

Я рад, что ты не опоздал.

Я *рад*, что ты не *опоздал*.

В первом предложении звучит искреннее облегчение, что человек не опоздал. Во втором, напротив, звучит сарказм или пассивная агрессия: так выражена досада от того, что человек немного опоздал.

В таких случаях в HTML используется элемент (выделение). Кроме того, чтобы сделать документ более интересным для чтения, они распознаются программами, считывающими с экрана, и произносятся другим тоном. Браузеры стилизуют это по умолчанию курсивом, но вы можете не использовать этот тег, чтобы получить курсив. Для выделения курсивом вы можете использовать элемент и CSS, или, возможно, элемент <i>.

```
<p>Я <em>рад</em>, что ты не <em>опоздал</em>.</p>
```

Чтобы подчеркнуть важные слова, мы склонны подчеркивать их в устной речи и **выделять жирным** на письменном языке. Например:

Эта жидкость **очень токсична**.

Я рассчитываю на вас. **Не опаздывай**!

В таких случаях в HTML используется элемент (важное значение). Помимо того, что документ становится более полезным, они распознаются программами, считывающими с экрана, и говорят другим тоном. Браузеры стилизуют это как полужирный текст по умолчанию, но вы можете не использовать этот тег, чтобы получить жирный шрифт. Для получения жирного шрифта вы можете использовать элемент и CSS, или, возможно, элемент <b> (смотрите ниже).

```
<p>Эта жидкость <strong>очень токсична</strong>.</p>

<p>Я рассчитываю на тебя. <strong>Не </strong>опаздывай!</p>
```

При желании вы можете вложить важные и акцентированные слова друг в друга:

```
<p>Эта жидкость <strong>очень токсична</strong> —
```

```
если ты выпьешь её, <strong>то можешь<em>умереть</em></strong>.</p>
```

Элементы, которые мы обсуждали до сих пор, имеют четкую привязку к семантике. Ситуация с <b>, <i> и <u> несколько сложнее. Они появились в эпоху, когда CSS поддерживался плохо или вообще не поддерживался, чтобы люди могли писать **жирный** текст, *курсив* или подчеркнутый текст. Такие элементы, которые влияют только на внешний вид, а не на семантику, известны как элементы представления и больше не должны использоваться, поскольку, как мы видели ранее, семантика очень важна для доступности людям с ограниченными возможностями, SEO и так далее.

HTML5 переопределил <b>, <i> и <u> с новыми, несколько запутанными, семантическими ролями.

Вот хорошее правило: предпочтительней использовать <b>, <i> или <u> для передачи значения, традиционно передаваемого жирным шрифтом, курсивом или подчеркиванием, при условии, что нет более подходящего элемента. Тем не менее, всегда важно сохранить менталитет доступности. Концепция курсива не очень помогает людям, использующим устройства для чтения с экрана, или людям, использующим систему письма, отличную от латинского алфавита.

- <i> используется для передачи значения, традиционно передаваемого курсивом: иностранные слова, таксономические обозначения, технические термины, мысли ...
- <b> используется для передачи значения, традиционно передаваемого жирным шрифтом: ключевые слова, названия продуктов, предложения ...
- <u> используется для передачи значения, традиционно передаваемого подчеркиванием: имя собственное, орфографическая ошибка ...

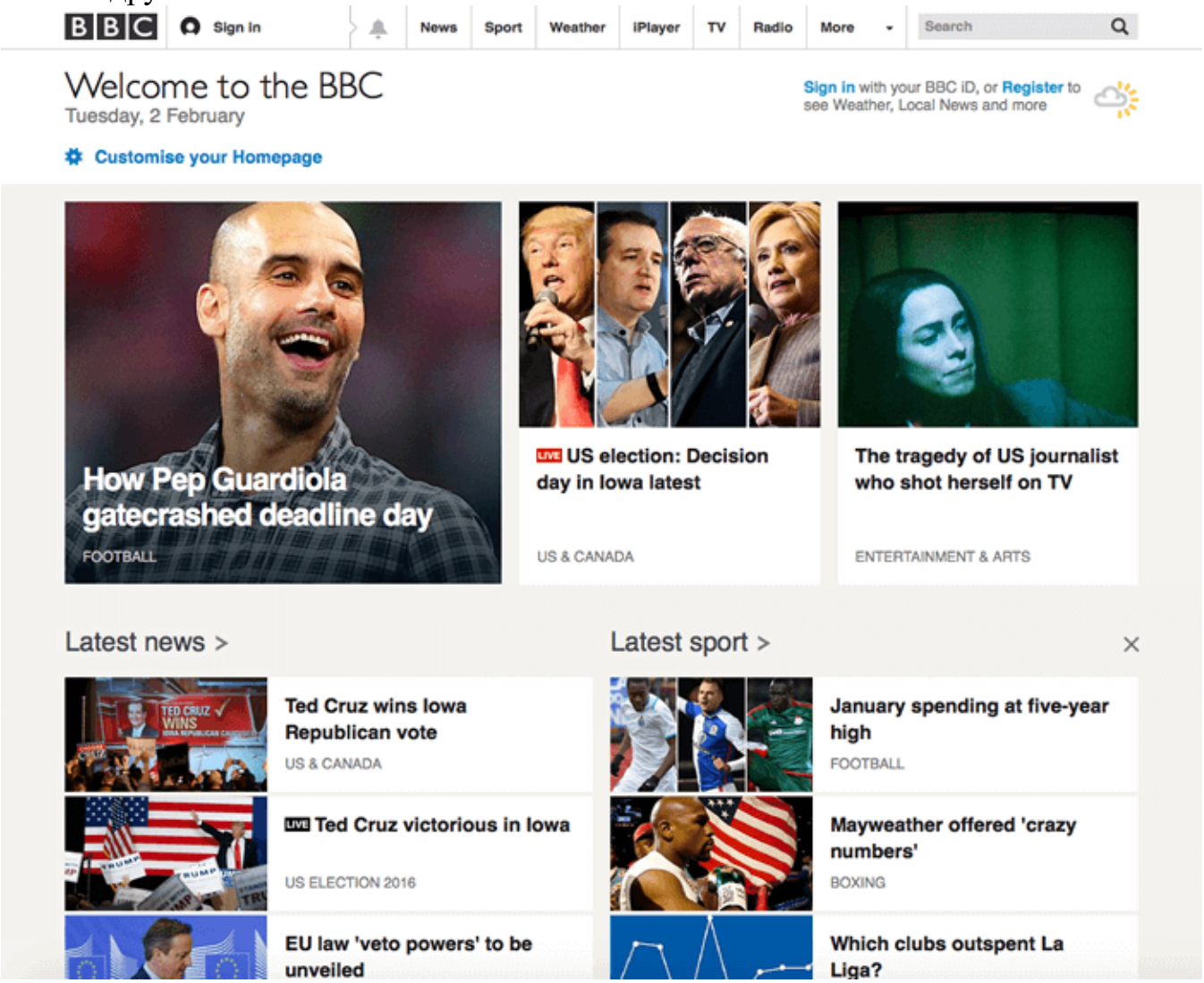
Предупреждение о подчеркивании: **люди сильно ассоциируют подчеркивание с гиперссылками**. Поэтому в Интернете лучше всего подчеркнуть только ссылки. Используйте элемент <u>, когда он семантически подходит, но подумайте о том, чтобы использовать CSS для изменения подчеркивания по умолчанию для чего-то более подходящего в Интернете.

Гиперссылки – одно из самых интересных нововведений Интернета. Они были особенностью Сети с самого начала, но именно они превращают Интернет в Интернет. Они позволяют нам связывать наши документы с любым другим документом (или ресурсом), с которым мы хотим. С их помощью мы также можем связывать документы с их конкретными частями, и мы можем сделать приложения доступными на простом веб-адресе (сравните это с локальными приложениями, которые должны быть установлены, и другими такими же вещами). Почти любой веб-контент может быть преобразован в ссылку, так что когда вы кликаете по ней (или иным образом активируете), она заставляет веб-браузер перейти на другой веб-адрес (URL).

URL-адрес может указывать на файлы HTML, текстовые файлы, изображения, текстовые документы, видео и аудиофайлы и все остальное, что может жить в Интернете. Если веб-браузер не знает, как отображать или обрабатывать файл, он спросит вас, хотите ли вы открыть файл (в этом случае обязанность открытия или обработки файла передаётся в соответствующее локальное приложение на

устройстве) или загрузить файл (в этом случае вы можете попытаться разобраться с ним позже).

Например, домашняя страница BBC содержит большое количество ссылок, которые указывают не только на множество новостей, но и на различные области сайта (меню), страницы входа / регистрации (пользовательские инструменты) и многое другое.



Простая ссылка создаётся путём обёртывания текста (или другого содержимого), который вы хотите превратить в ссылку, в элемент <a> *(a* это сокращение от "anchor", "якорь"), и придания этому элементу атрибута href (который также известен как **гипертекстовая ссылка,** или **цель**), который будет содержать веб-адрес, на который вы хотите указать ссылку.

```
<p>Я создал ссылку на
  <a href="https://www.mozilla.org/ru/">домашнюю страницу Mozilla</a>.
</p>
```

Другим атрибутом, который вы можете добавить к своим ссылкам, является – title. Он предназначен для хранения полезной информации о ссылке. Например, какую информацию содержит страница или другие вещи, о которых вам нужно знать. Например:

```
<p>Я создал ссылку на
  <a href="https://www.mozilla.org/ru/"
     title="Лучшее место для поиска дополнительной информации
     о миссии Mozilla и о том, как внести свой вклад">домашнюю страницу
Mozilla
  </a>.
</p>
```

Описание из атрибута title отображается только при наведении курсора, значит люди, полагающиеся на клавиатурные элементы управления для навигации по веб-страницам, будут испытывать трудности с доступом к информации, которую содержит title. Если информация заголовка действительно важна для удобства использования страницы, то вы должны представить ее таким образом, который будет доступен для всех пользователей, например, поместив её в обычный текст.

Чтобы текст в вашем абзаце стал ссылкой, выполните следующие действия:

1. Выберите некоторый текст. Мы выбрали текст "Манифест Mozilla".
2. Оберните текст в элемент <a>, например так:

```
<a>Манифест Mozilla</a>
```

3. Задайте элементу <a> атрибут href, например так:

```
<a href="">Манифест Mozilla</a>
```

4. Заполните значение этого атрибута веб-адресом, на который вы хотите указать ссылку:

```
<a href="https://www.mozilla.org/ru/about/manifesto/details/">Манифест
Mozilla</a>
```

Можно получить неожиданные результаты, если в самом начале веб-адреса опустить https:// или http:// часть, называемую *протоколом*. После создания ссылки, кликните по ней, чтобы убедиться, что она направляет вас туда, куда вы хотели.

Href сначала может выглядеть довольно непонятым выбором для имени атрибута. Если у вас возникли проблемы с тем, чтобы запомнить его, можете запомнить, что атрибут href образуется как ***h****ypertext* ***ref****erence* ("гипертекстовая ссылка").

Можно превратить любой элемент в ссылку, даже блочный элемент. Если у вас есть изображение, которые вы хотели бы превратить в ссылку, вы можете просто поместить изображение между тегами <a>.

```
<a href="https://www.mozilla.org/ru/">
  <img src="mozilla-image.png" alt="логотип mozilla со ссылкой на их
домашнюю страницу">
</a>
```

Чтобы полностью понять адреса ссылок, нужно понять несколько вещей про URL-адреса и пути к файлам. Этот раздел даст вам информацию, необходимую для достижения этой цели.

URL-адрес (Uniform Resource Locator, или единый указатель ресурса, но так его никто не называет) – это просто строка текста, которая определяет, где что-то находится в Интернете. Например, домашняя страница Mozilla находится по адресу https://www.mozilla.org/ru/.

URL-адреса используют пути для поиска файлов. Пути указывают, где в файловой системе находится файл, который вас интересует.

Можно ссылаться на определенную часть документа HTML (известную как **фрагмент документа**), а не только на верхнюю часть документа. Для этого вам сначала нужно назначить атрибут id элементу, с которым вы хотите связаться. Обычно имеет смысл ссылаться на определённый заголовок, поэтому это выглядит примерно так:

```
<h2 id="Почтовый_адрес">Почтовый адрес</h2>
```

Затем, чтобы связаться с этим конкретным id, вы должны включить его в конец URL-адреса, которому предшествует знак решётки, например:

```
<p>Хотите написать мне письмо? Используйте наш
  <a href="contacts.html#Почтовый_адрес">почтовый адрес</a>.
</p>
```

Вы даже можете использовать ссылку на фрагмент документа отдельно для ссылки на *другую часть того же документа*:

```
<p>
  <a href="#Почтовый_адрес">Почтовый адрес кампании</a>
  можно найти в нижней части этой страницы.
</p>
```

При написании ссылок рекомендуется следовать некоторым правилам.

На вашей странице легко добавить ссылки. Но этого не совсем достаточно. Мы должны сделать наши ссылки *доступными* для всех читателей, независимо от их возможностей и инструментов просмотра страницы, которые они предпочитают. Например:

- Пользователям программ читающих с экрана нравится переходить по ссылкам на странице, читая адрес ссылки в тексте.
- Поисковые системы используют текст ссылки для индексирования файлов, поэтому рекомендуется включать ключевые слова в текст ссылки, чтобы эффективно описывать, куда ведет ссылка.
- Пользователи часто бегло просматривают страницу, не читая каждое слово, и их глаза будут привлечены к тексту, который выделяется, например, ссылки. Им будет полезно описание того, куда ведет ссылка.

Взгляните на этот пример:

***Хороший** текст ссылки:* Скачать Firefox

```
<p><a href="https://firefox.com/">
  Скачать Firefox
</a></p>
```

Плохой *текст ссылки:* Нажми сюда, чтобы скачать Firefox

```
<p><a href="https://firefox.com/">
  Нажми сюда
</a>
чтобы скачать Firefox</p>
```

Советы:

– Не пишите URL-адрес как часть текста ссылки. URL-адреса выглядят сложными, а звучат ещё сложнее, когда программы чтения с экрана читают их по буквам.

– Не пишите «ссылка» или «ссылки на» в тексте ссылки – это лишнее. Программы чтения с экрана сами проговаривают, что есть ссылка. На экране пользователи также видят, что есть ссылка, потому что ссылки, как правило, оформлены в другом цвете и подчеркнуты (подчёркивая ссылки, вы соблюдаете правила хорошего тона, поскольку пользователи привыкли к этому).

– Следите за тем, чтобы текст ссылки был как можно короче. Длинный текст ссылки особенно раздражает пользователей программ чтения с экрана, которым придётся услышать всё, что написано.

– Минимизируйте случаи, когда несколько копий одного и того же текста ссылок указывает на разные страницы. Это может вызвать проблемы для пользователей программ чтения с экрана, которые часто вызывают список ссылок – несколько ссылок, помеченных как «нажмите здесь», «нажмите здесь», «нажмите здесь», будут путать.

Когда вы создаёте ссылку на файл, нажав на который можно загрузить документ PDF или Word или открыть просмотр видео, прослушивание аудио файла или перейти на страницу с другим, неожиданным для пользователя результатом (всплывающее окно или загрузка Flash-фильма), добавляйте четкую формулировку, чтобы уменьшить путаницу. Отсутствие описания может раздражать пользователя. Приведем пример:

– Если вы используете соединение с низкой пропускной способностью и вдруг нажмёте на ссылку без описания, начнётся загрузка большого файла.

– Если у вас нет установленного Flash-плеера и вы нажмёте ссылку, то внезапно перейдёте на страницу с Flash-контентом.

Посмотрите на примеры, чтобы увидеть, как добавить описание:

```
<p><a href="http://www.example.com/large-report.pdf">
  Скачать отчет о продажах (PDF, 10MB)
</a></p>

<p><a href="http://www.example.com/video-stream/">
  Посмотреть видео (видео откроется в отдельном окне, HD качество)
```

```
</a></p>

<p><a href="http://www.example.com/car-game">
  Играть в гонки (необходим Flash)
</a></p>
```

Когда создаёте ссылку на файл, который должен быть загружен, а не открыт в браузере, можете использовать атрибут download, чтобы создать имя файла по умолчанию для сохранения. Приведем пример ссылки для загрузки браузера Firefox:

```
<a href="https://download.mozilla.org/?product=firefox-39.0-
SSL&os=win&lang=en-US"
   download="firefox-39-installer.exe">
  Скачать Firefox 39 для Windows
</a>
```

Можно создавать ссылки или кнопки, которые при нажатии открывают новое исходящее сообщение электронной почты, а не ссылку на ресурс или страницу. Для этого используется элемент <a> и mailto: - *адрес почты*.

Самыми простыми и часто используемыми формами mailto: являются *subject*, *cc*, *bcc* и *body*; дальше прописываем адрес электронной почты. Например:

```
<a href="mailto:nowhere@mozilla.org">Отправить письмо для nowhere</a>
```

Сам адрес электронной почты не является обязательным для заполнения. Если оставить это поле пустым (в поле href оставить только "mailto:"), откроется новое исходящее сообщение почтовой программой, в поле получателя будет пусто. Это можно использовать для кнопки "Поделиться".

Помимо адреса электронной почты, вы можете предоставить другую информацию. Фактически, любые стандартные поля для отправки почты могут быть добавлены к указанному вами адреса mailto. Часто используемыми из них являются «subject», «cc» и «body» (которые не являются истинным полем заголовка, но позволяют указать дополнительную информацию для нового сообщения электронной почты). Каждое поле и его значение задаются в качестве условия запроса.

Вот пример который включает cc(кому отправить копию сообщения, все получатели письма видят список тех кто это письмо получит), bcc(скрытый адрес получателя, никто из получателей не будет видеть полный список получателей письма), subject(тема письма) и body(текст сообщения):

```
<a
href="mailto:nowhere@mozilla.org?cc=name2@rapidtables.com&bcc=name3@
rapidtables.com&amp;subject=The%20subject%20of%20the%20email
&amp;body=The%20body%20of%20the%20email">
  Send mail with cc, bcc, subject and body
</a>
```

Значение каждого поля должно быть написано в URL-кодировке (то есть с непечатаемыми символами и пробелами). Обратите внимание на знак вопроса (?) для разделения основного адреса и дополнительных полей, амперсанд (&) для разделения каждого поля mailto: URL. Для этого используется стандартное описание URL запроса.

Теперь добавьте ссылки в вашей странице с письмом, если вы еще не сделали этого.

Таблица – это структурированный набор данных, состоящий из строк и столбцов (**табличных данных**). Таблицы позволяют быстро и легко посмотреть значения, показывающие некоторую взаимосвязь между различными типами данных, например - человек и его возраст, или расписание в плавательном бассейне.

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Public Swim 06:30 - 10:30	Public Swim 06:30 - 09:00	Public Swim 06:30 - 09:00	Public Swim 06:30 - 11:15	Public Swim 06:30 - 09:00	Lane Swim 08:00 - 09:00	Lane Swim 08:00 - 09:00
Aquacise 10:30 - 11:15	Aqua Jog 09:15 - 10:00	Education Swimming Lessons 09:00 - 12:00	Aquacise 11:15 - 12:00	Education Swimming Lessons 09:00 - 12:00	Oldham Active Kids Swimming Lessons 09:00 - 13:00	Public Swim 09:00 - 11:00
Lane Swim 11:30 - 13:00	Parent & Baby Class 09:30 - 10:15	Lane Swim 12:00 - 13:00	Lane Swim 12:00 - 13:00	Lane Swim 12:00 - 13:00	Parent and Baby 12:00 - 12:45	Aquacise 11:00 - 11:45
Education Swimming Lessons	Public Swim 10:00 - 11:45	Public Swim 13:00 - 16:00	Education Swimming Lessons	Oldham Active Kids Swimming	Public Swim 13:00 - 17:00	Public Swim 11:45 - 13:00

Так что не удивительно, что создатели HTML включили в него средства для структурирования и представления табличных данных в сети.

Смысл таблицы в том, что она жесткая. Информацию легко интерпретировать, визуально сопоставляя заголовки строк и столбцов.

HTML-таблицы следует использовать для табличных данных – это то, для чего они предназначены. К сожалению, многие используют таблицы HTML для оформления веб-страниц, например, одна строка для заголовка, одна для содержимого, одна для сносок, и тому подобное. Это происходило из-за плохой поддержки CSS в разных браузерах; в наше время такое встречается гораздо реже, но иногда все же попадается.

Использование таблиц в целях разметки вместо методов CSS является плохим решением по следующим причинам:

1. **Таблицы, используемые для оформления, уменьшают доступность страниц для людей, имеющих проблемы со зрением**. Скринридеры (Screenreaders), используемые ими, интерпретируют HTML-теги и читают содержимое пользователю. Поскольку таблицы не являются средством для представления структуры таблицы, и разметка получается сложнее, чем при использовании методов CSS, скринридеры вводят пользователей в заблуждение.
2. **Таблицы создают путаницу тегов.** Как уже упоминалось, оформление страниц с помощью таблиц дает более сложную структуру разметки, чем специально предназначенные для этого методы. Соответственно, такой код труднее писать, поддерживать и отлаживать.

3. **Таблицы не реагируют автоматически на тип устройства**. У соответствующих контейнеров (например, <header>, <section>, <article>, или) ширина по умолчанию равна 100% от их родительского элемента. У таблиц же размер по умолчанию подстраивается под их содержимое, поэтому, чтобы они одинаково хорошо работали на разных типах устройств, необходимо принимать дополнительные меры.

Содержимое любой таблицы заключается между двумя тегами: **<table></table>**. Добавьте их в тело HTML. Самым маленьким контейнером в таблице является ячейка, она создается элементом **<td>** ('td' - сокращение от 'table data'). Введите внутри тегов table следующее:

```
<td>Hi, I'm your first cell.</td>
```

Чтобы получить строку из четырех ячеек, необходимо скопировать эти теги три раза. Обновите содержимое таблицы так, чтобы она выглядела следующим образом:

```
<td>Hi, I'm your first cell.</td>
<td>I'm your second cell.</td>
<td>I'm your third cell.</td>
<td>I'm your fourth cell.</td>
```

Как видите, ячейки не располагаются одна под другой, на самом деле они автоматически выравниваются по отношению к другим ячейкам той же строки. Каждый элемент <td> создает отдельную ячейку, а все вместе они создают первую строку. Каждая добавленная ячейка удлиняет эту строку.

Чтобы эта строка перестала расти, а новые ячейки перешли на вторую строку, необходимо использовать элемент **<tr>** ('tr' - сокращение от 'table row'). Попробуем, как это получится. Поместите четыре уже созданных ячейки между тегами <tr> как здесь показано:

```
<tr>
  <td>Hi, I'm your first cell.</td>
  <td>I'm your second cell.</td>
  <td>I'm your third cell.</td>
  <td>I'm your fourth cell.</td>
</tr>
```

Теперь, когда одна строка уже есть, добавим еще – каждую строку надо вложить в дополнительный элемент <tr>, а каждая ячейка должна быть внутри элемента <td>.

В результате получится таблица, которая будет выглядеть примерно так:

Hi, I'm your first cell.	I'm your second cell.	I'm your third cell.	I'm your fourth cell.
Second row, first cell.	Cell 2.	Cell 3.	Cell 4.

Теперь обратимся к табличным заголовкам – особым ячейкам, которые идут вначале строки или столбца и определяют тип данных, которые содержит данная строка или столбец. Чтобы опознавать заголовки таблицы в качестве заголовков,

визуально и семантически, можно использовать элемент **<th>** ('th' сокращение от 'table header'). Он работает в точности как <td>, за исключением того, что обозначает заголовок, а не обычную ячейку. Замените в своем HTML все элементы <td>, содержащие заголовки, на элементы <th>.

По умолчанию к заголовкам таблицы примеряется определенный стиль – они выделены жирным шрифтом и выровнены по центру, даже если вы не задавали для них стиль специально.

Заголовки дают дополнительное преимущество – вместе с атрибутом scope они помогают улучшить связь каждого заголовка со всеми данными строки или столбца одновременно, что довольно полезно.

Иногда нам нужно, чтобы ячейки распространялись на несколько строк или столбцов. Возьмем простой пример, в котором приведены имена животных. Иногда бывает нужно вывести имена людей рядом с именами животных. А иногда это не требуется, и тогда мы хотим, чтобы имя животного занимало всю ширину. Табличные заголовки и ячейки имеют атрибуты colspan и rowspan, которые позволяют это сделать. Оба принимают безразмерное числовое значение, которое равно количеству строк или столбцов, на которые должны распространяться ячейки. Например, colspan="2" распространяет ячейку на два столбца.

HTML позволяет также указать, какой стиль нужно применять к целому столбцу данных сразу – для этого применяют элементы **<col>** и **<colgroup>**. Их ввели, поскольку задавать стиль для каждой ячейки в отдельности или использовать сложный селектор вроде :nth-child() было бы слишком утомительно.

Возьмем простой пример:

```
<table>
  <tr>
    <th>Data 1</th>
    <th style="background-color: yellow">Data 2</th>
  </tr>
  <tr>
    <td>Calcutta</td>
    <td style="background-color: yellow">Orange</td>
  </tr>
  <tr>
    <td>Robots</td>
    <td style="background-color: yellow">Jazz</td>
  </tr>
</table>
```

Что дает нам:

Data 1	Data 2
Calcutta	Orange
Robots	Jazz

Он не идеален, поскольку нам пришлось повторить информацию о стиле для всех трех ячеек в столбце (в реальном проекте, возможно, придется вводить class на

всех трех и вводит правило в таблице стилей). Вместо этого, мы можем задать информацию один раз, в элементе <col>. Элемент <col> задается в контейнере <colgroup> сразу же за открывающим тегом <table>. Эффект, который мы видели выше, можно задать так:

```
<table>
  <colgroup>
    <col>
    <col style="background-color: yellow">
  </colgroup>
  <tr>
    <th>Data 1</th>
    <th>Data 2</th>
  </tr>
  <tr>
    <td>Calcutta</td>
    <td>Orange</td>
  </tr>
  <tr>
    <td>Robots</td>
    <td>Jazz</td>
  </tr>
</table>
```

Мы определяем два "стилизующих столбца". Мы не применяем стиль к первому столбцу, но пустой элемент <col> ввести необходимо – иначе к первому столбцу не будет применен стиль.

Если бы мы хотели применить информацию о стиле к обоим столбцам, мы могли бы просто ввести один элемент <col> с атрибутом span, таким образом:

```
<colgroup>
  <col style="background-color: yellow" span="2">
</colgroup>
```

Подобно colspan и rowspan, span принимает безразмерное числовое значение, указывающее, к какому количеству столбцов нужно применить данный стиль.

Вы можете добавить заголовок для таблицы установив его в элементе <caption> и этот элемент необходимо поместить внутрь элемента <table>. Причем вам нужно поместить его сразу после открытия тега <table>.

```
<table>
  <caption>Dinosaurs in the Jurassic period</caption>


  ...
</table>
```

Как можно понять из короткого примера выше, заголовок отражает в себе описание контента таблицы. Это полезно для всех читателей просматривающих

страницу и желающих получить краткое представление от том полезна ли для них таблица, что особенно важно для слепых пользователей. Вместо того чтобы читать содержимое множества ячеек чтобы понять о чем таблица, он или она могут полагаться на заголовок и принимать решение читать ли таблицу более подробно.

Заголовок помещают сразу после тега <table>.

Атрибут summary также может быть использован в <table> элементе предоставляя описание – это также читается скринридерами. Однако мы рекомендуем вместо этого использовать <caption> элемент, так как summary deprecated в HTML5 спецификации и не может быть прочитан зрячими пользователям (он не отображается на странице).

Когда таблицы становятся более сложными по структуре полезно дать им более структурированное определение. Отличный способ сделать это используя <thead>, <tfoot> и <tbody>, которые позволяют вам разметить header, footer и body секции таблицы.

Эти элементы не создают дополнительной доступности для пользователей со скринридерами и не приводят к какому-то визуальному улучшению при их использовании. Зато они очень полезны при стилизации и разметке, как точки для добавления CSS к вашей таблице. Вот несколько интересных примеров, в случае длинной таблицы вы можете сделать header и footer таблицы повторяемый на каждой печатной странице, или вы можете сделать body таблицы отображаемое на одной странице и иметь доступ ко всему содержимому контенту прокручивая вверх и вниз.

Использование:

- Элемент <thead> применятся к той части таблицы, которая относится к заголовку – обычно это первая строка содержащая заголовки колонок, но это не обязательно. Если вы используете <col>/<colgroup> элемент, тогда заголовок должен находиться ниже его.
- Элемент <tfoot> применятся к той части, которая относится к footer таблицы – например, это может быть последняя строка в которой отображаются суммы по столбцам таблицы.
- Элементом <tbody> оборачивают остальную часть содержимого таблицы, которая не находится в header или footer таблицы. Этот блок располагают ниже заголовка таблицы или иногда footer таблицы, зависит от того какую структуру вы решите использовать.

<tbody> всегда включен в каждой таблице, неявно если не укажете его в коде. Вы могли бы задаться вопросом почему мы должны волноваться о его включении, но это необходимо, потому что это дает больше контроля над структурой таблицы и стилем.

В одну таблицу вкладывать другую таблицу возможно, если вы используете полную структуру включая элемент <table>. Это как правило не рекомендуется, так как делает разметку более запутанной и менее доступной для пользователей скринридеров, так в большинстве случаев вы можете просто вставить дополнительные ячейки/строки/столбцы в существующую таблицу. Однако, иногда это необходимо, например, если вы хотите легко импортировать контент из других источников.

Существует еще один атрибут scope, который может быть добавлен к элементу <th>. Он сообщает скринридеру, какие ячейки точно являются заголовками – например, заголовок строки или столбца, в которых он находится. Например в таблице с записями расходов, можно однозначно определить заголовки столбцов:

```
<thead>
  <tr>
    <th scope="col">Purchase</th>
    <th scope="col">Location</th>
    <th scope="col">Date</th>
    <th scope="col">Evaluation</th>
    <th scope="col">Cost (€)</th>
  </tr>
</thead>
```

И у каждой строки может быть определен заголовок, как здесь (если мы добавили заголовки строк и заголовки столбцов):

```
<tr>
  <th scope="row">Haircut</th>
  <td>Hairdresser</td>
  <td>12/09</td>
  <td>Great idea</td>
  <td>30</td>
</tr>
```

Скринридер распознает разметку структурированную таким образом, что позволяют пользователям прочитать весь столбец или строку целиком.

Атрибут scope имеет еще два возможных значения – colgroup и rowgroup. Они используются для заголовков, которые располагаются вверху ваших столбцов или строк.

Альтернатива атрибута scope – это использование атрибутов id и headers задавая ассоциации между заголовками и ячейками. Этот способ выглядит следующим образом:

1. Вы устанавливаете уникальный id для каждого <th> элемента.
2. Вы устанавливаете атрибут headers для каждого <td> элемента. Каждый headers атрибут должен содержать список всех id , разделенный пробелами, ко всем <th> элементам, которые действуют как заголовок для этой ячейки.

Это обеспечивает явное определение позиции для каждой ячейки вашей HTML таблицы, определяет заголовки столбцов и строк таблицы. Для того чтобы это работало реально хорошо таблице нужно определить и заголовки столбцов, и заголовки строк.

Вернемся к нашему примеру с расчетом затрат, его можно написать следующим образом:

```
<thead>
  <tr>
```

```
    <th id="purchase">Purchase</th>
    <th id="location">Location</th>
    <th id="date">Date</th>
    <th id="evaluation">Evaluation</th>
    <th id="cost">Cost (€)</th>
  </tr>
</thead>
<tbody>
<tr>
  <th id="haircut">Haircut</th>
  <td headers="location haircut">Hairdresser</td>
  <td headers="date haircut">12/09</td>
  <td headers="evaluation haircut">Great idea</td>
  <td headers="cost haircut">30</td>
</tr>

  ...

</tbody>
```

Этот метод создания очень точного определения ассоциаций между заголовками и данными в ячейках, но использует **гораздо** больше разметки и оставляет обширное пространство для ошибок. Атрибута scope обычно достаточно для большинства таблиц.