Курс «Основы веб-программирования»
Лабораторная 2. CSS

Целью данной лабораторной является создание таблицы стилей и добавление навигации на веб-страницы из лабораторной 1 (HTML).

Порядок работы

1. Добавь на страницу **"Об авторе"** правило CSS для отображения заголовка первого уровня (h1) определенным цветом.
2. Добавь правило для отображения всех заголовков заглавными буквами.
3. Создай правило для использования в качестве шрифта страницы шрифта без засечек.
4. Создай правило для изменения фона страницы на какой-либо повторяющийся рисунок.
5. Чтобы текст на странице легко читался, помести все блоки текста в подходящие теги-контейнеры и создай для них одно правило, задающее светлый фоновый цвет, расположение по центру и ширину в 75% от ширины окна браузера.
6. Отредактируй HTML-код с цитатой согласно рекомендациям MDN.
7. Создай правило для отображения текста цитаты в рукописном стиле. Найди подходящий рукописный шрифт и сошлись на него на странице **"Об авторе"**.
8. Размести созданную таблицу стилей в отдельном файле `style.css` и сошлись на неё в каждой из страниц проекта.
9. Создай навигационный блок и добавь его на каждую страницу, созданную в ходе первой лабораторной.
10. Обнови таблицу на странице **"Дневник разработки"**, заменив "OK" на символ ✓ (Check Mark). Удали атрибут `border`, если он имеется, и добавь аналогичное по действию правило в таблицу стилей `style.css`.

Месяц	День	Тема	Урок	Лабораторная
Декабрь 2017	13	HTML и поиск	✓	✓
	20	CSS и Bootstrap	✓	✓
Январь 2018	3	Переменные и функции		
	10	Структура программы		
	17	Массивы и объекты		
	24	Взаимодействие с веб-страницей		
	31	Взаимодействие с сервером		

11. Обнови **"Дневник разработки"**, добавив сведения о второй лабораторной.
12. Попробуй поменять порядок следования правил и их селекторы, понаблюдай за эффектом и отмени изменения. Просмотри страницу с отключенным CSS.
13. Проверь страницы и таблицу стилей с помощью validator.w3.org и исправь ошибки (предупреждения допустимы).

Описание действий

1. Добавь на страницу **"Об авторе"** правило CSS для отображения заголовка первого уровня (h1) определенным цветом.

Внутренние таблицы стилей задаются в теге `style` (который служит контейнером для стиля), находящемся в теге `head`.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Example</title>
  <style>
    body {
      margin: 0;
      background-color: blue;
    }
  </style>
</head>
```

NB! Крайне рекомендуется явно указывать кодировку документа, задавая тег `meta` с атрибутом `charset`. В таком случае исключается возможность, что браузер неправильно определит кодировку и выдаст нечитаемый текст.

Для определения цвета используется свойство `color`. В качестве значения указывается название цвета (например, `red`) или его код в различных системах (например, `rgb(255, 0, 0)` или `#ff0000`).

2. Добавь правило для отображения всех заголовков заглавными буквами.

Действие правила можно распространить на несколько селекторов, перечислив их через запятую. Например:

```
h1, h2, h3, h4, h5, h6 {
  color: red;
}
```

Для изменения регистра текста используется свойство `text-transform`.

3. Создай правило для использования на странице шрифта без засечек.

Шрифт без засечек (sans serif) считается оптимальным для чтения с экрана, тогда как для чтения с листа бумаги предпочитают шрифт с засечками – serif. Для определения шрифта элемента используется свойство `font-family`. В качестве значения указывается перечень подходящих шрифтов, начиная от конкретных названий (`"Times New Roman"`, `Helvetica`) до типовых семейств (`sans-serif`, `cursive`).

При работе над оформлением веб-страницы разработчикам приходится считаться с изрядной долей неопределенности, так как характеристики конечного устройства остаются неизвестными. Например, на компьютере пользователя может не оказаться необходимого шрифта. В этом случае (как и во многих прочих проблемных ситуациях) браузер действует самостоятельно – подбирает шрифт исходя из имеющихся. По этой причине всегда следует указывать в качестве последнего значения в списке подходящее *семейство* шрифтов. В связи с этой проблемой возникло понятие "безопасных шрифтов для веб-дизайна" (Web Safe Fonts).

Свойство `font-family` определяет не форматирование самого блока (такие свойства, как `отступы` или `ширина`), а `форматирование содержимого`, поэтому его действие распространяется на все вложенные элементы (теги, размещенные внутри выбранного тега). Этот механизм называется *наследованием*.

В данном случае имеет смысл использовать в качестве селектора `html` или `body`, так как именно в них содержатся остальные элементы веб-страницы.

Помимо наследования, важную роль играет принцип *каскадности* — правила, указанные позже, отменяют ранее определенные правила для конкретного элемента. Так, если в таблице стилей имеется следующий набор правил в указанном порядке следования, все элементы `h1` в документе будут иметь синий цвет, в то время как остальные заголовки останутся красными:

```
h1, h2, h3, h4, h5, h6 {
  color: red;
}

h1 {
  color: blue;
}
```

Однако если поменять правила местами, все заголовки будут красного цвета.

4. Создай правило для изменения фона страницы на какой-либо повторяющийся рисунок.

Для определения фона элемента используется свойство `background-image`. Так как видимое пользователю содержимое страницы находится в теге `body`, его и следует использовать в качестве селектора.

5. Чтобы текст на странице легко читался, помести все блоки текста в подходящий тег-контейнер и создай для него одно правило, задающее светлый фоновый цвет, расположение по центру и ширину в 75% от ширины окна браузера.

Тег `div` является контейнером, чье единственное назначение — группировка элементов в отдельный блок. В остальном он не имеет побочных эффектов и идеально подходит в качестве элемента для оформления с помощью CSS. Тег `span` является аналогичным контейнером, но в отличие от `div`, не разрывает поток элементов и не создает перехода на новую строку.

При составлении HTML-кода страницы стоит активно пользоваться тегами-контейнерами для семантического разграничения информации. В дальнейшем это облегчит работу над оформлением.

В идеальном случае, при работе над оформлением не должно возникать необходимости править HTML-код.

Чтобы задать правило для тегов-контейнеров, следует использовать *классы*. Если элемент уникален, можно использовать *идентификатор*.

CSS
```
.note {
  border: 5px solid red;
  background-color: yellow;
}

#note15 {
  font-weight: bold;
}
```

HTML
```
<div id="note15" class="note">
  <img src="note.png" alt="">
  <p>Примечание: классы и идентификаторы можно совмещать.</p>
</div>
```

В пределах одного документа не должно быть нескольких элементов с одинаковым идентификатором!

Стандарт HTML5 содержит целый набор специализированных тегов-блоков, единственное назначение которых — разделить и обозначить семантически единые структуры. Такими тегами являются `article` (предназначен для содержимого статьи), `main` (для основной части веб-страницы), `nav` (для навигации), `header`, `footer` и т.д.

Если ранее для задач подобного рода использовались элементы div с заданным классом:

CSS
```
.article {
  margin: 0 auto;
}
```

HTML
```
<div class="article">…содержание статьи…</div>
```

То теперь достаточно воспользоваться специальным элементом:

CSS
```
article {
  margin: 0 auto;
}
```

HTML
```
<article>…содержание статьи…</article>
```

Ширина элемента задается с помощью свойства `width`. Расположение элемента по центру задается с помощью свойства `margin` со значением `auto`. Логика заключается в том, что значение `auto` заставляет браузер подобрать одинаковые значения как для правого, так и для левого внешнего отступа, что и приводит к центровке элемента. При этом обязательно должна быть указана ширина (иначе элемент займёт всё доступное по ширине пространство и эффект не будет заметен).

При работе с цветом можно использовать эффект прозрачности, выбрав в качестве значения цвет с альфа-каналом — `rgba` или `hsla`. Данная возможность поддерживается не всеми старыми браузерами, поэтому следует разместить перед свойством с альфа-каналом свойство без него:

```
background-color: rgb(255, 0, 0);
background-color: rgba(255, 0, 0, 0.75);
```

Если браузер не может прочитать какое-то свойство, он использует предыдущее распознанное.

Для уточнения, какие браузеры поддерживают желаемую возможность, следует использовать справочники MDN Web Docs, W3Schools или специализированный портал caniuse.com

6. Отредактируй HTML-код с цитатой согласно рекомендациям MDN.

Mozilla Developers Network (MDN, https://developer.mozilla.org/en-US/) является одним из наиболее авторитетных ресурсов для веб-разработчика. Материалы сайта наполняются силами сообщества и зачастую содержат наиболее актуальную информацию (а также включают действующие ссылки на спецификации и стандарты).

Поиск по тегу `cite` выдаст исчерпывающую статью, предлагающую три одобряемых способа оформления цитат. В первом случае текст цитаты автоматически обрамляется кавычками (в более новых браузерах) и остается на той же строке:

```
Science, according to Donald Knuth, is <q cite="https://
www.math.upenn.edu/~wilf/AeqB.pdf">what we understand well enough to
explain to a computer. Art is everything else we do</q>.
```

Если требуется разместить более внушительный по размеру отрывок, используется второй вариант, при котором текст размещается на новой строке и отбивается отступом:

```
<article>The joy of programming as described by Fred Brooks:<blockquote
cite="https://books.google.com/books?id=Yq35BY5Fk3gC&pg=PT24">The
programmer, like the poet, works only slightly removed from pure thought-
stuff. He builds his castles in the air, from air, creating by exertion
of the imagination. […]
Yet the program construct, unlike the poet's words, is real in the sense
that it moves and works, producing visible outputs separate from the
construct itself. […] The magic of myth and legend has come true in our
time. One types the correct incantation on a keyboard, and a display
screen comes to life, showing things that never were nor could be.
</blockquote></article>
```

Следует отметить, что в этих случаях источник цитирования должен быть действующим URI. Браузером он не отображается, но может использоваться поисковыми системами. Если стоит задача продемонстрировать источник информации, используется тег `cite`:

```
<p>As Edsger W. Dijkstra pointed in <cite>The Humble Programmer</cite>,
<q cite="http://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/
EWD340.html">We must be very careful when we give advice to younger
people: sometimes they follow it!</q></p>
```

7. Создай правило для отображения текста цитаты в рукописном стиле. Найди подходящий рукописный шрифт и сошлись на него на странице **"Об авторе"**.

Поскольку перечень безопасных шрифтов весьма скуден, с развитием сетевых технологий появилась возможность загружать в браузер пользователя необходимый шрифт. Одним из наиболее крупных каталогов веб-шрифтов является fonts.google.com.

В правом меню можно выбрать начертание шрифта и поддерживаемый язык. После обнаружения желаемого шрифта следует нажать на значок "+" в правом верхнем углу его карточки. В нижней части страницы появится плашка "1 Family Selected". При щелчке на ней откроется панель с краткими инструкциями по использованию и ссылкой на руководство.

8. Размести созданную таблицу стилей в отдельном файле `style.css` и сошлись на неё в каждой из страниц проекта.

Для создания внешнего файла CSS достаточно скопировать всё содержимое тега `style` в отдельный файл и сохранить его с расширением `.css`. Для использования внешней таблицы стилей следует поместить в теге `head` тег `link` с атрибутами `href` и `rel`:

```
<link href="style.css" rel="stylesheet">
```

При использовании нескольких таблиц стилей и сторонних ресурсов необходимо помнить о принципе каскадности – таблица, указанная последней, имеет большее значение. Если на страницу импортируются шрифты, то тег link, ссылающийся на веб-шрифт, должен быть указан до таблицы стилей, которая его использует!

Если веб-шрифт используется практически на всех страницах проекта, имеет смысл указать его в таблице стилей, а не на конкретных страницах — в этом случае браузер загрузит шрифт один раз и далее будет брать его из кеша (памяти). Для этой цели нужно сослаться на шрифт в файле CSS с помощью директивы `@import` (необходимый код можно также найти на fonts.google.com).

Директива `@import` должна находиться в самом начале файла CSS, иначе она не будет работать!

9. Создай навигационный блок и добавь его на каждую страницу, созданную в ходе первой лабораторной.

Навигационный блок является наиболее ярким примером разграничения семантики и оформления. Так навигация является по своей сути списком ссылок, для её создания используется тег `ul`. Для большей ясности вся конструкция помещается в тег `nav`, чтобы четко обозначить назначение списка.

Рассмотрим для нашего проекта ситуацию, когда навигационный блок располагается в верхней части страницы. Для этого разместим nav самым первым элементом в теге body.

Следующий этап — изменение внешнего вида блока. Для начала зададим фон:

```
nav {
  background-color: #333; /* краткая форма записи значения #333333 */
}
```

При этом сразу станет заметно, что вокруг блока остается пустое пространство. Причина заключается в том, что по умолчанию браузер добавляет отступы практически для всех блочных элементов. Чтобы изменить данное поведение, требуется указать нулевые внешние отступы для тегов body, nav и ul:

NB! Рекомендуется добавлять правила CSS по одному и наблюдать за эффектом.

```
body {
  margin: 0;
  …
}

nav {
  margin: 0;
  background-color: #333;
}

nav ul { /* правило распространяется только на элемент ul внутри nav */
  margin: 0;
  list-style: none;
}
```

Изменим отображение элементов списка, разместив их на одной линии:

```
nav li {
  display: inline;
}
```

Изменим отображение ссылок, заставив их приобрести свойства блока (кроме разрыва строки). Затем добавим объем, указав внутренние отступы:

```
nav a {
  display: inline-block;
  padding: 0.7em 1em;
  text-decoration: none;
  color: #ccc;
}
```

NB! Рекомендуется посмотреть назначение используемых свойств в справочнике CSS.

Чтобы пользователь почувствовал, что имеет дело со ссылкой, было бы хорошо затемнять её цвет при нахождении под указателем мыши. В CSS для этой задачи имеются псевдоклассы:

```
nav a:hover {
  background-color: #000;
}
```

Наконец, остается выделить цветом ту страницу, на которой пользователь сейчас находится. С этой целью стоит создать специальный класс и добавить его к активной ссылке:

CSS
```
nav .active {
  background-color: #777;
  color: #fff;
}
```

HTML
```
<nav>
  <ul>
    <li><a class="active" href="about.html">Об авторе</a></li>
    <li><a href="dev-diary.html">Дневник разработки</a></li>
    …
```

Ещё одним важным аспектом CSS является *специфичность*. Так как один и тот же элемент может быть затронут разными правилами (в том числе через наследование), для разрешения спорных ситуаций был разработан специальный механизм. При наличии нескольких правил-кандидатов на применение к элементу побеждает то, чей селектор имеет больший вес (среди правил с одинаковым весом побеждает указанное последним).

В нашем примере селектор `nav .active` проигрывает селектору `nav a:hover`, поэтому активная ссылка меняет свой цвет при наведении мыши (хотя правило для активной ссылки указано позднее). Если мы желаем, чтобы при наведении мыши цвет активной ссылки не менялся, следует повысить специфичность селектора `nav .active`. К примеру, можно дополнительно указать, что правило распространяется только на ссылки:

```
nav a.active {
  background-color: #777;
  color: #fff;
}
```

За счет дополнительного элемента мы сравняли вес селекторов и браузер предпочел последнее правило. Можно заподозрить, что данное решение не является ни изящным, ни надёжным — его можно "сломать", случайно изменив вес селекторов. В связи с этим многие веб-дизайнеры стараются не смешивать в селекторах теги, классы и идентификаторы и отдают предпочтение практически исключительно классам — в этом случае правила действуют более точечно и их взаимодействие проще отслеживать.

В разработке часто имеет место эффект сообщающихся сосудов — к примеру, упрощение таблицы стилей ведёт к усложнению HTML-кода страницы.

В данном случае решение с использованием классов может выглядеть следующим образом (заметна простота селекторов, достигнутая за счет усложнения HTML):

CSS
```
body, nav, ul {
  margin: 0;
}

#nav {
  background-color: #333;
}

.nav-list {
  list-style: none;
}

.nav-item {
  display: inline;
}

.nav-link {
  display: inline-block;
  padding: 0.7em 1em;
  text-decoration: none;
  color: #ccc;
}

.nav-link:hover {
  background-color: #000;
}

.nav-link.active {
  background-color: #777;
  color: #fff;
}
```

HTML
```
<nav id="nav">
  <ul class="nav-list">
    <li class="nav-item">
      <a class="nav-link active" href="about.html">Об авторе</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="dev-diary.html">Дневник разработки</a>
    …
```

Теперь остается скопировать блок `nav` на остальные страницы проекта, не забыв при этом правильно указать класс `active` в зависимости от страницы.

10. Обнови таблицу на странице **"Дневник разработки"**, заменив "OK" на символ ✓ (Check Mark). Удали атрибут `border`, если он имеется, и добавь аналогичное по действию правило в таблицу стилей `style.css`.

При необходимости использования на странице специальных символов, можно воспользоваться мнемониками HTML (HTML Entities). В частности, мнемоники требуется использовать в том случае, если требуемый символ имеет специальное значение в языке HTML. Например, знак "<" используется для оформления тегов, потому для обозначения математического неравенства "5 < 2" требуется применить мнемонику `<` (имеющую расшифровку "less than"):

```
<span class="math">5 &lt; 2</span>
```

При использовании менее распространённых мнемоник надёжнее указывать числовой код символа (как правило, в каталогах он указывается вместе с мнемоникой). Например, символ ♪ можно обозначить с помощью мнемоники:

```
<span>&sung; My songs</span>
```

но надёжнее использовать HTML-код:

```
<span>&#9834; My songs</span>
```

Вместо атрибута `border` тега `table` следует создать отдельное правило для тегов таблицы и указать для них одноимённое свойство:

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse; /* заменить двойную рамку на одинарную */
}
```

11. Обнови **"Дневник разработки"**, добавив сведения о второй лабораторной.

Добавь на страницу раздел **"Лабораторная 2. CSS"** и помести в него материалы и ссылки, которые помогли в выполнении этой лабораторной. Преврати текст ячейки **"CSS и Bootstrap"** в ссылку на раздел "**Лабораторная 2. CSS"**.

Порядок действий аналогичен рассмотренному в Лабораторной 1 (HTML).

12. Попробуй поменять порядок следования правил и их селекторы, понаблюдай за эффектом и отмени изменения. Просмотри страницу с отключенным CSS.

Как было отмечено ранее, при подсчете итогового правила для каждого элемента учитывается вес селекторов, а для правил с одинаковым весом селекторов учитывается взаимное расположение (побеждает правило, указанное позднее).

При грамотном подходе к созданию и оформлению страницы, её содержание не должно страдать от отсутствия таблицы стилей.

Отключение CSS в браузере Firefox доступно в меню View > Page Style > No Style. В Chrome требуется проявить смекалку и, к примеру, открыть Developer Tools и вручную стереть

содержимое CSS файлов во вкладке Resources. Также можно открыть панель Elements и удалить теги `style` и `link` с атрибутами `rel="stylesheet"`.

Дополнительный эксперимент:
Замени элемент `img` на странице **"Об авторе"** на элемент `div` с атрибутом `style` (обрати внимание на одинарные кавычки — их пришлось использовать, чтобы не "разорвать" содержимое атрибута `style`):

```
<div id="pic1" style="background-image: url('lemons.jpg'); height: 300px; width: 300px;"></div>
```

В качестве значения атрибута `style` указывается выражение CSS (без селектора, так как правило будет применено непосредственно к самому элементу). Результат будет аналогичным использованию тега `img` с той разницей, что мы использовали *встроенный* стиль CSS. Размеры пришлось задать вручную, так как элемент не имеет содержимого и по умолчанию приобретает размеры равными 0.

Так как встроенный стиль элементов можно редактировать с помощью JavaScript, данный подход часто используется для обновления изображения (по существу – для изменения фона элемента) без перезагрузки страницы. Как побочный эффект, копирование изображения через контекстное меню станет неосуществимым.

В общей сложности существует три типа таблиц стилей – внешние, внутренние и встроенные. При подсчете итогового правила для элемента сначала просматриваются внешние и внутренние таблицы в порядке их следования в `head` и в последнюю очередь просматриваются встроенные. Соответственно, встроенные стили имеют наибольший приоритет.

Встроенные стили следует использовать только в случае крайней необходимости, так как они являются специализированным средством и теряют часть преимуществ, предлагаемых технологией CSS.

Верни изображение к исходному состоянию (использование тега `img`).

13. Проверь страницы и таблицу стилей с помощью validator.w3.org и исправь ошибки (предупреждения допустимы).

Соответствие правилам языка — один из признаков хорошей веб-страницы.

Рекомендуемые материалы

1. CSS | MDN
   https://developer.mozilla.org/en-US/docs/Web/CSS

2. W3Schools CSS Tutorial
   https://www.w3schools.com/css/

3. W3Schools CSS Reference
   https://www.w3schools.com/cssref/

4. Google Fonts
   https://fonts.google.com

5. Web Technology for Developers | MDN
   https://developer.mozilla.org/en-US/docs/Web

6. HTML Symbols, Entities, Characters and Codes – HTML Arrows
   https://www.toptal.com/designers/htmlarrows/

7. W3C Markup Validator
   http://validator.w3.org/

Дополнительные материалы

1. CSS Zen Garden - The Beauty of CSS Design
   http://www.csszengarden.com/

2. Decoupling HTML From CSS
   https://www.smashingmagazine.com/2012/04/decoupling-html-from-css/

Лицензия