\* {

    margin: 0;

    padding: 0;

}

Отключает все стандартные отступы в браузере.

<section> - служит для группировки содержимого на странице

<section>

<h1>Заголовок</h1>

<p>

Первый абзац текста.

</p>

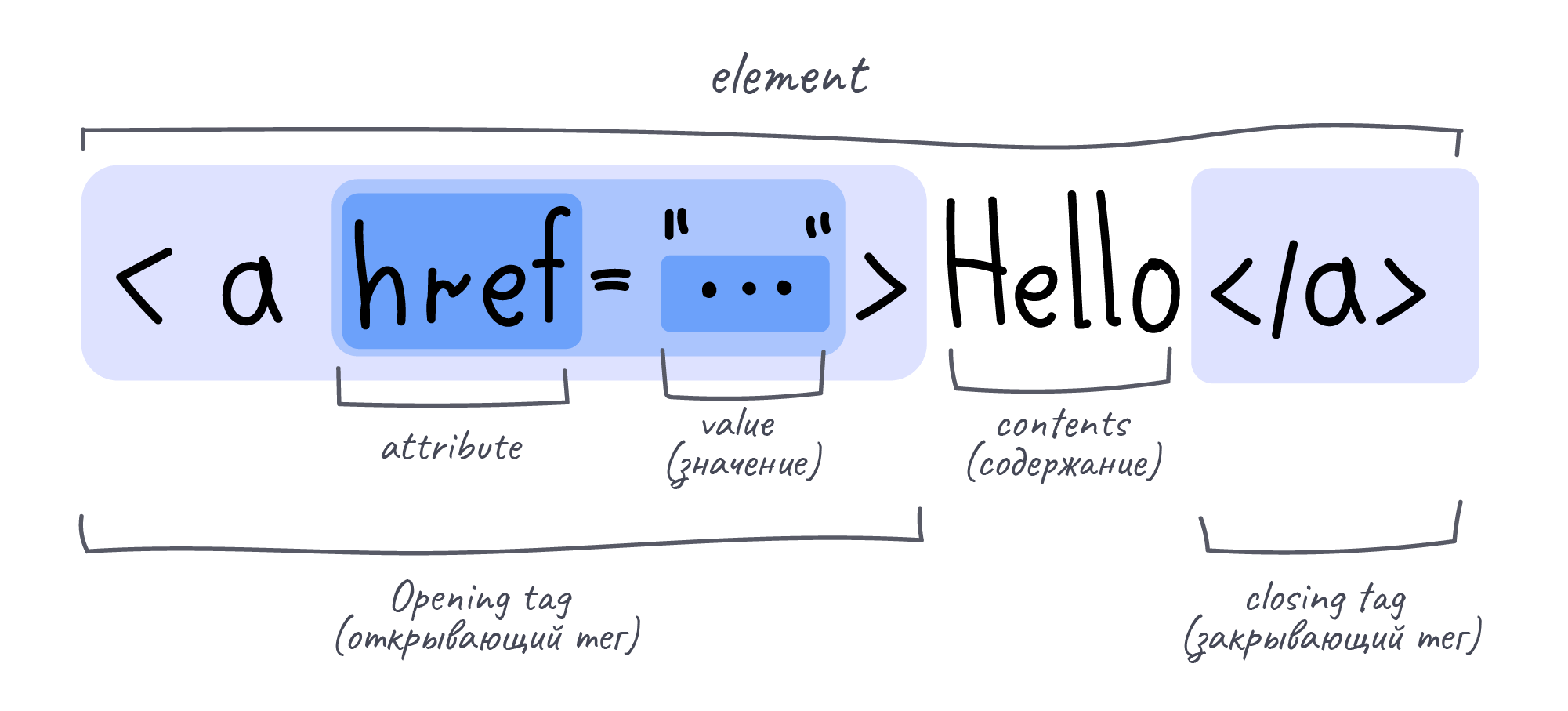
<p>

Второй абзац текста.

</p>

</section>

* <h1> - заголовок первого уровня, отображает текст "Хорошие книги".
* <p> - абзац, содержит текст "Советую прочитать эти книги:".
* <ol> - упорядоченный список, содержит элементы списка книг.
* <ul> - неупорядоченный список (точка).
* <li> - элемент списка, каждый содержит название книги и автора.
* <b>, <strong> - жирное выделение текста.
* <i> - курсивное выделение текста.
* <a> - ссылка, в данном случае на страницу википедии.



<a href=»ссылка»>название ссылки</a>

<a href=»ссылка» title=»ссылка на статью»>название ссылки</a>

title – атрибут, который дает подсказку при наведении на ссылку

target =»\_blank» – позволяет открывать ссылку на новой странице

target =»\_self» - позволяет открывать ссылку на текущей странице

Пример:

<a href=»ссылка» target =»\_blank» title=»ссылка на статью»>название ссылки</a>

Создание якоря (переход к тексту на той же странице). Например, чтобы перейти к заголовку **Смысл жизни**:

<a href=»#point»>название ссылки</a>

<h2 id=»point»>Cмысл жизни</h2 >

<h1>Хорошие книги</h1>

<p>Советую прочитать эти книги:</p>

<ol>

<li><b>Код. Тайный язык информатики</b> <i>(Чарльз Петцольд)</i></li>

<li>

<b>Не заставляйте меня думать</b>

<i>(<a href="https://en.wikipedia.org/wiki/Steve\_Krug">Steve Krug</a>)</i>

</li>

<li><b>Дизайн привычных вещей</b> <i>(Donald Norman)</i></li>

</ol>

<h2>Рекомендуемые книги</h2>

<ul>

<li><b>Мастер и Маргарита</b> <i>(Михаил Булгаков)</i></li>

<li><b>1984</b> <i>(Джордж Оруэлл)</i></li>

<li><b>Преступление и наказание</b> <i>(Федор Достоевский)</i></li>

</ul>

Хорошие книги

Советую прочитать эти книги:

1. **Код. Тайный язык информатики** *(Чарльз Петцольд)*
2. **Не заставляйте меня думать** *([Steve Krug](https://en.wikipedia.org/wiki/Steve_Krug))*
3. **Дизайн привычных вещей** *(Donald Norman)*

Рекомендуемые книги

* **Мастер и Маргарита** *(Михаил Булгаков)*
* **1984** *(Джордж Оруэлл)*
* **Преступление и наказание** *(Федор Достоевский)*

Вы можете попробовать открыть разметку этой страницы с помощью комбинации Ctrl + U

**Для добавления картинки в HTML:**

<img src=»адрес изображения» alt=»описание изображения»>

<div class="article">

<div class="article-header">

<h1>Заголовок статьи</h1>

</div>

<div class="article-body">

<p>А здесь <strong>находится</strong> текст статьи</p>

</div>

</div>

# Заголовок статьи

А здесь **находится** текст статьи

<span>

Длинный текст, который мы обернули в строчный элемент span. Внутри span мы

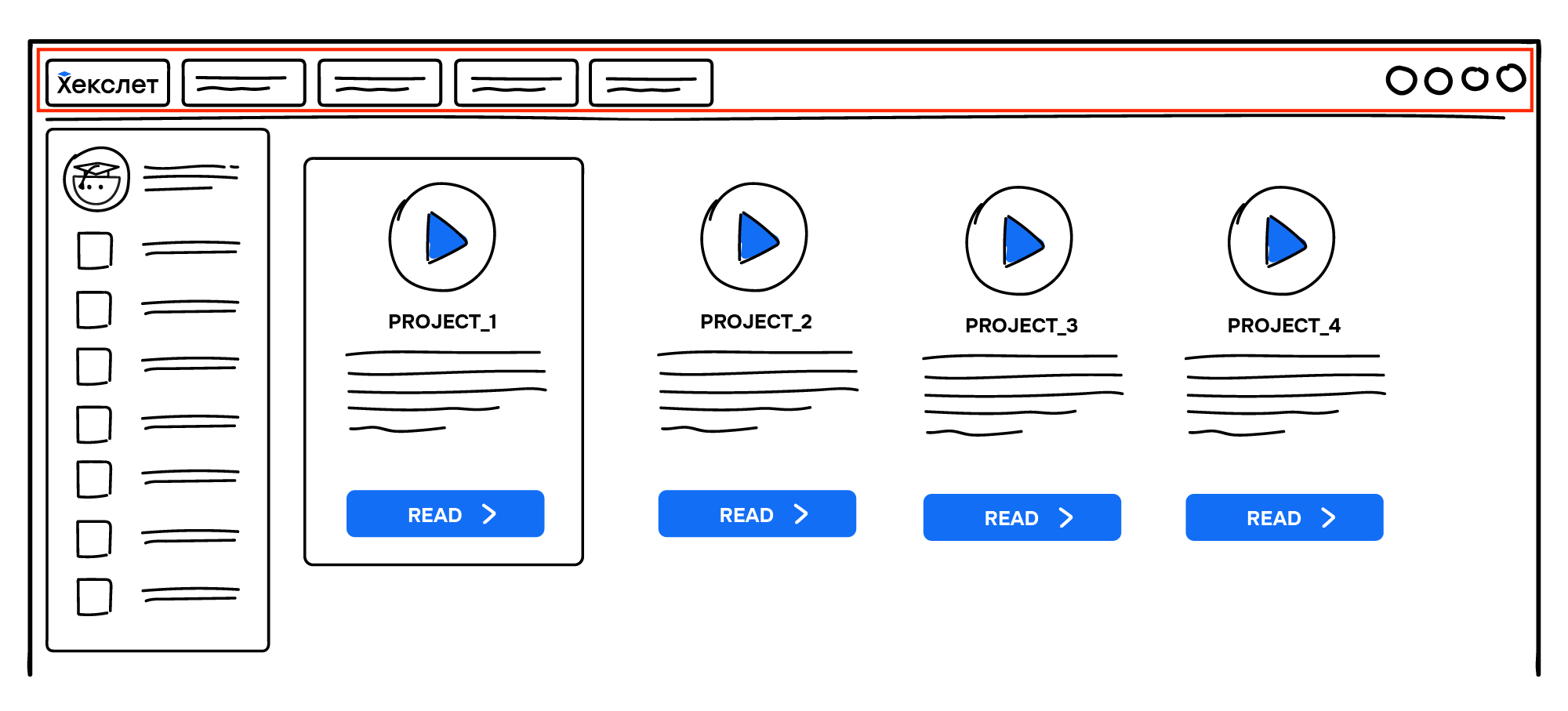
также можем вставить тег <i>i</i>, <strong>strong</strong> и так далее.

</span>

Длинный текст, который мы обернули в строчный элемент span. Внутри span мы также можем вставить тег *i*, **strong** и так далее.

## **Шапка сайта**

Верхняя область макета зачастую называется «шапкой» сайта. Она содержит название компании, основное меню, контактную информацию. Эта область очень важна для быстрой навигации по сайту, так как обычно содержит меню с основными разделами сайта.



Чтобы создать шапку сайта, используется парный тег <header>, внутри которого мы и располагаем необходимую информацию.

Один из примеров шапки сайта:

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<div id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</div>

</header>

Логотип

* [Главная](https://htmlcodeeditor.com/)
* [О нас](https://htmlcodeeditor.com/about)
* [Контакты](https://htmlcodeeditor.com/contacts)

Начало формы

Конец формы

По своему поведению тег <header> работает так же, как и простой <div>. Почти все семантические элементы являются блочными и не имеют стилей по умолчанию. Это позволяет очень быстро добавить семантику в уже существующие проекты. Если стили в этих проектах не завязаны на тегах, то достаточно просто сменить название с <div> на <header>, и мы получим уже семантичную шапку сайта.

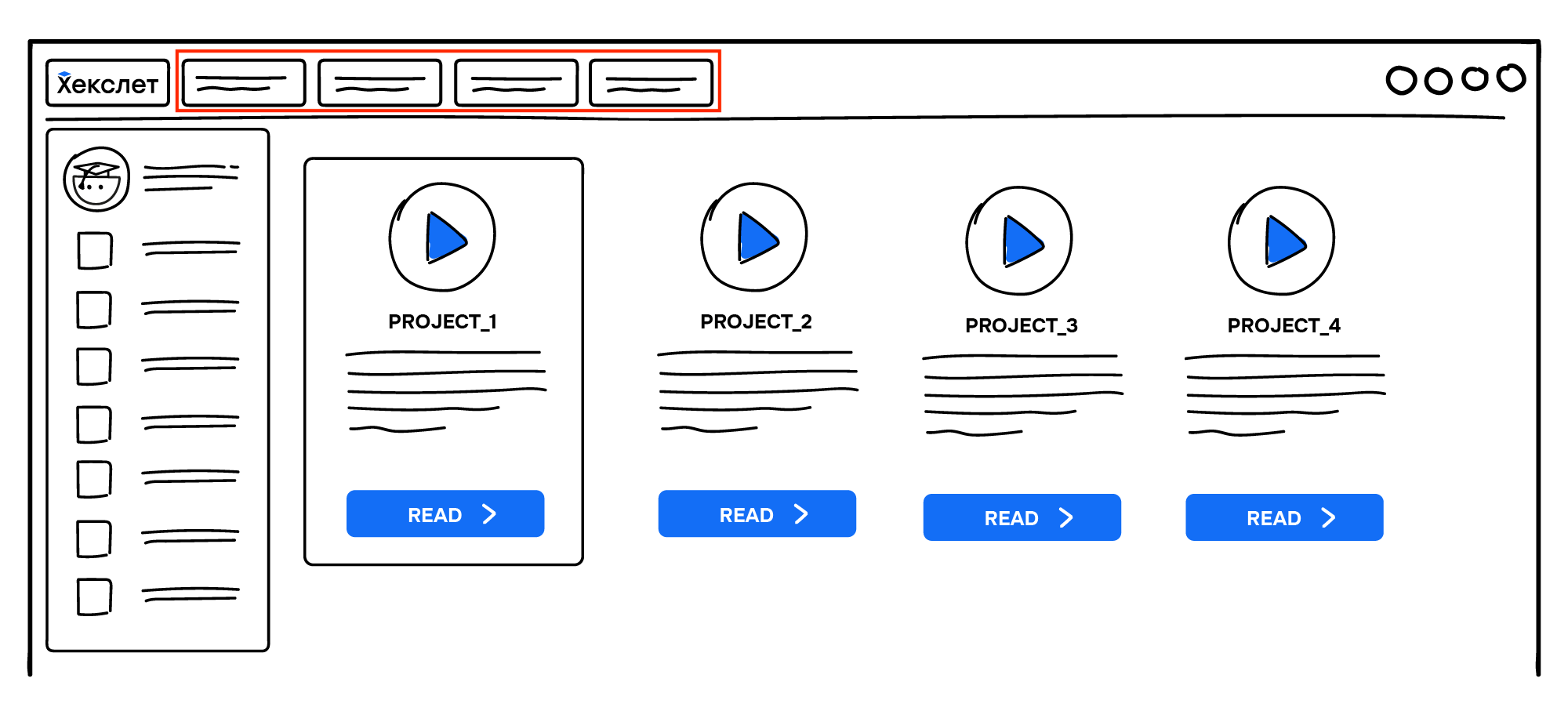
## **Навигация**

Попробуйте взглянуть на шапку сайта из примера выше глазами компьютера. Что он там видит?

* Картинку
* Блочный элемент <div>
* Маркированный список
* Набор ссылок

Если вы еще не сверстали пару своих макетов, то по такому набору можете и не понять, что набор ссылок не что иное, как основное меню сайта. Вот и роботам не всегда просто это сделать. Конечно, они уже достаточно обучены, чтобы в таком простом наборе найти меню, но меню обычно может быть не одно и быть контекстно-зависимым. Тогда робот может принять за основное меню не то, что бы мы хотели.

Как же нам ему помочь? Для этого в стандарте HTML5 появился тег <nav>. Он обозначает навигационную область. Причем помочь отделить основное меню от меню раздела можно с помощью других тегов, речь о которых пойдет ниже.



Заменим простой блочный элемент <div> на его семантичного брата <nav>:

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

Начало формы

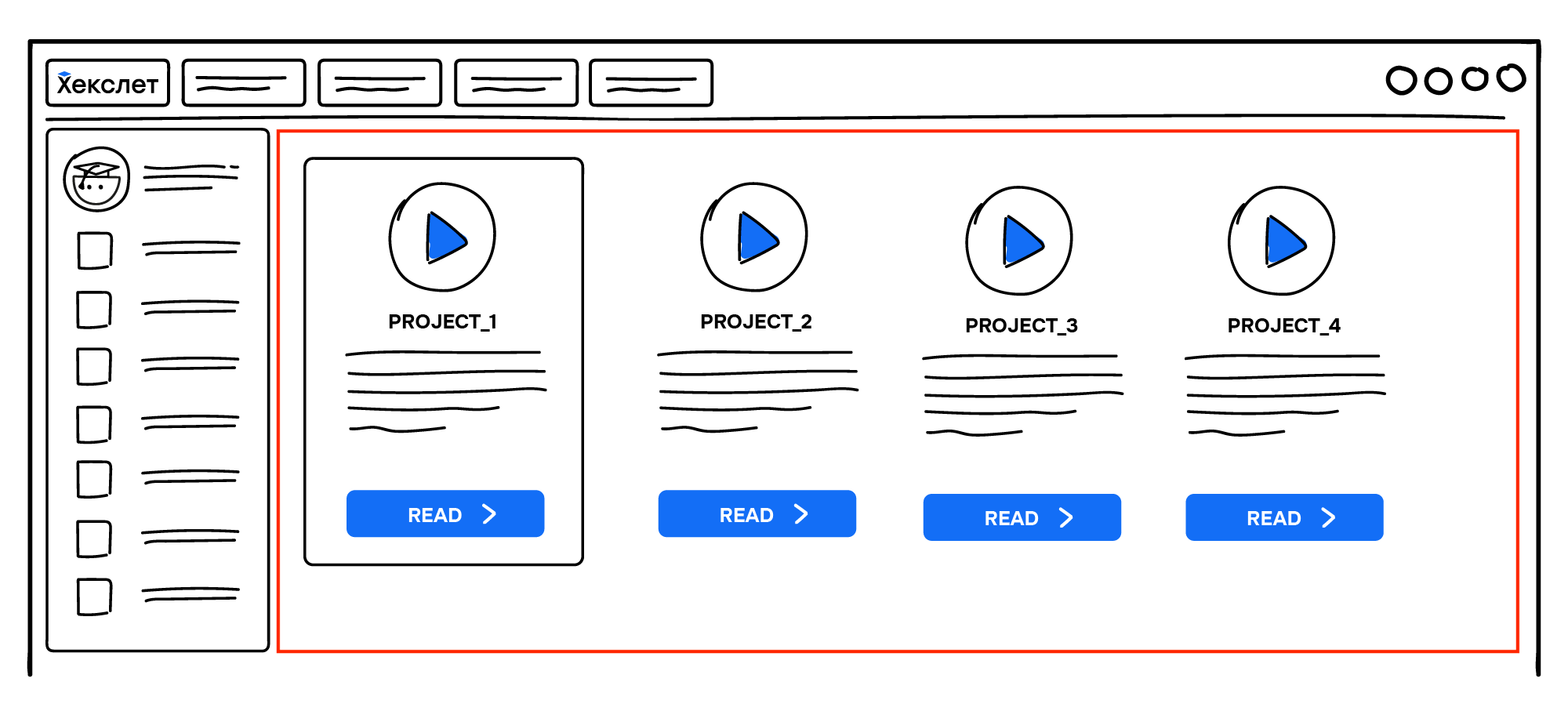
Конец формы

Главной особенностью использования <nav> является то, что не обязательно оборачивать все меню на странице. Обычно достаточно обернуть только главное меню и, например, не оборачивать меню в футере. При этом не запрещается иметь сразу несколько элементов <nav> на странице. Выделяйте ими главные меню на странице.

## **Уникальный контент**

Основной смысл каждой страницы — уникальный контент. Это самое главное, что должно быть на вашей странице. Пользователь вполне справится без меню или футера, но если на странице нет своего уникального контента, то страница бесполезна.

Для разметки уникального контента в стандарте HTML5 появился специальный тег <main>. Именно он поможет обозначить область с самым важным контентом на странице. Старайтесь в нем держать только контент. Обычно меню, боковые панели и футер в эту область не входят. Исключением может быть только ситуация, если эти блоки действительно уникальны для данной страницы. Например, меню может вести по разделам страницы. В таком случае оно на полных правах может быть включено в область уникального контента.



Добавим такую область в нашу верстку:

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

<main>

<h1>Хекслет — практические курсы по программированию</h1>

<p>

Мы считаем, что настоящий программист должен понимать, как работает

компьютер, и обладать вычислительным мышлением. Он должен видеть проблему, а

не задание. Должен уметь анализировать и рассуждать на уровне проблемы и

выше, а не только на уровне кода.

</p>

<p>

При сегодняшнем обилии учебных материалов, курсов и книг, главный вопрос,

который встает перед новичком, это не «где учиться?», а «что и в каком

порядке учить?». Существует множество мнений по этому поводу. Кто-то

советует начинать с математики, кто-то советует конкретные языки и

технологии.

</p>

<p>

Хекслет — это готовый путь от абсолютного новичка до первого

трудоустройства.

</p>

</main>

Начало формы

Конец формы

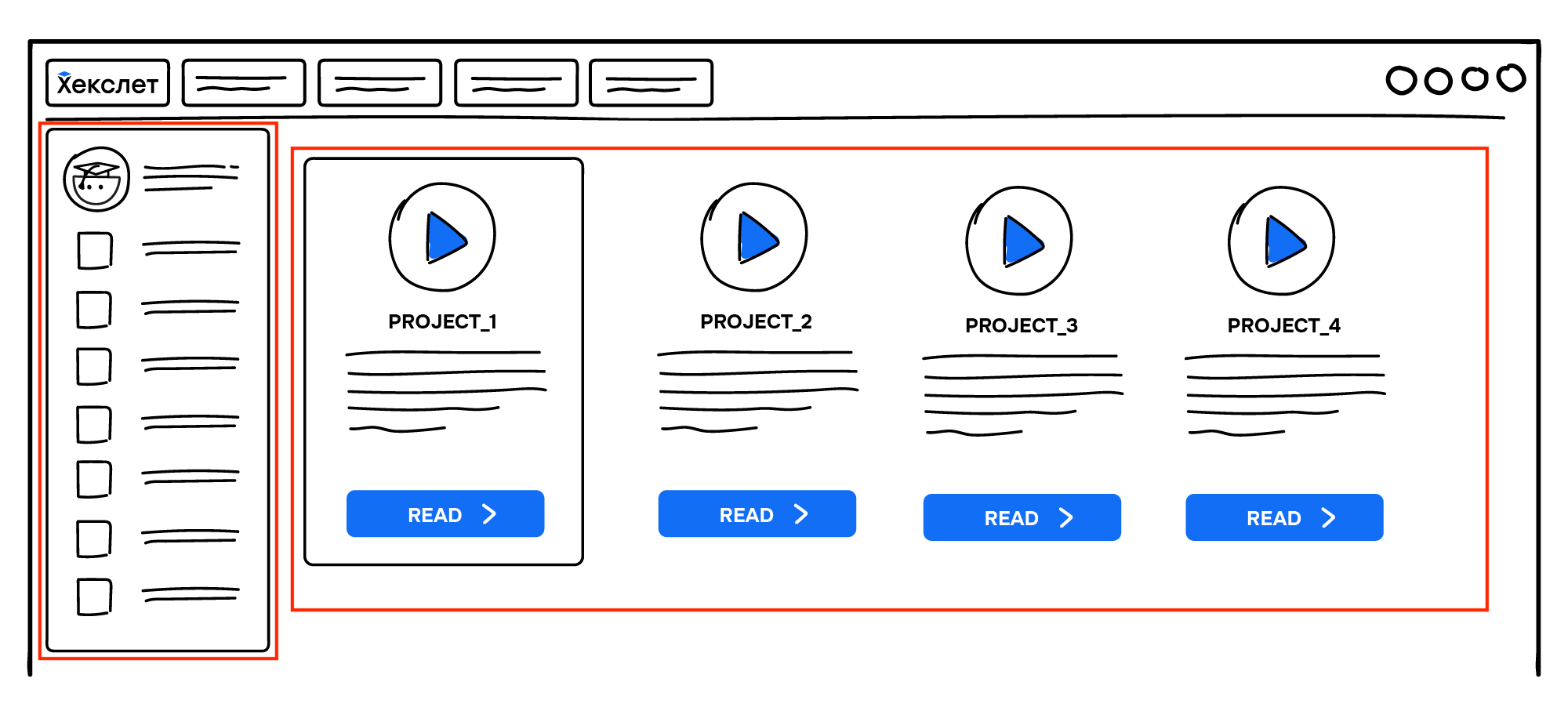
Наличие тега <main> также очень важно для мобильных браузеров. Вы могли видеть, что многие из них имеют функцию «Режим чтения». При его включении браузер автоматически удалит все оформление и все ненужные блоки, оставив только главный контент. Этим контентом и будет являться область, заключенная в теге <main>. Такой режим отлично подходит для людей, у которых в настоящий момент слабое подключение к интернету.

Так как внутри <main> содержится уникальный контент страницы, то разрешено использовать только один такой тег на странице.

## **Секции**

Контент на странице не является однородным. Обычно это цепочка логических областей, каждая из которых описывает что-то конкретное. Например, на странице могут присутствовать область с описанием преимуществ, цены, формы и так далее. Их хочется как-то выделить. И на это есть несколько причин:

* С правильно разделенным контентом удобно работать. Мы легко можем перемещать такие области, менять местами или удалять. В коде их будет легко найти
* Правильная группировка разделов — важная часть при создании доступного web'а



Для таких самостоятельных логических единиц существует специальный тег <section>, который внутри себя может содержать одну конкретную секцию. Добавим ее в наш пример:

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

<main>

<h1>Хекслет — практические курсы по программированию</h1>

<p>

Мы считаем, что настоящий программист должен понимать, как работает

компьютер, и обладать вычислительным мышлением. Он должен видеть проблему, а

не задание. Должен уметь анализировать и рассуждать на уровне проблемы и

выше, а не только на уровне кода.

</p>

<p>

При сегодняшнем обилии учебных материалов, курсов и книг, главный вопрос,

который встает перед новичком, это не «где учиться?», а «что и в каком

порядке учить?». Существует множество мнений по этому поводу. Кто-то

советует начинать с математики, кто-то советует конкретные языки и

технологии.

</p>

<p>

Хекслет — это готовый путь от абсолютного новичка до первого

трудоустройства.

</p>

<section>

<h2>Преимущества</h2>

<ul>

<li>Много теории</li>

<li>Много практики</li>

<li>Много менторов</li>

</ul>

</section>

</main>

Начало формы

Конец формы

Заметьте, как легко стало искать преимущества в коде, ведь они находятся в отдельной секции. Также вы могли обратить внимание на заголовок внутри секции. Так как секция — это самостоятельная единица, то почти всегда она имеет свой заголовок. Хоть это и не всегда так, и стандарт не обязывает нас включать заголовок в секцию, но старайтесь придерживаться этого правила.

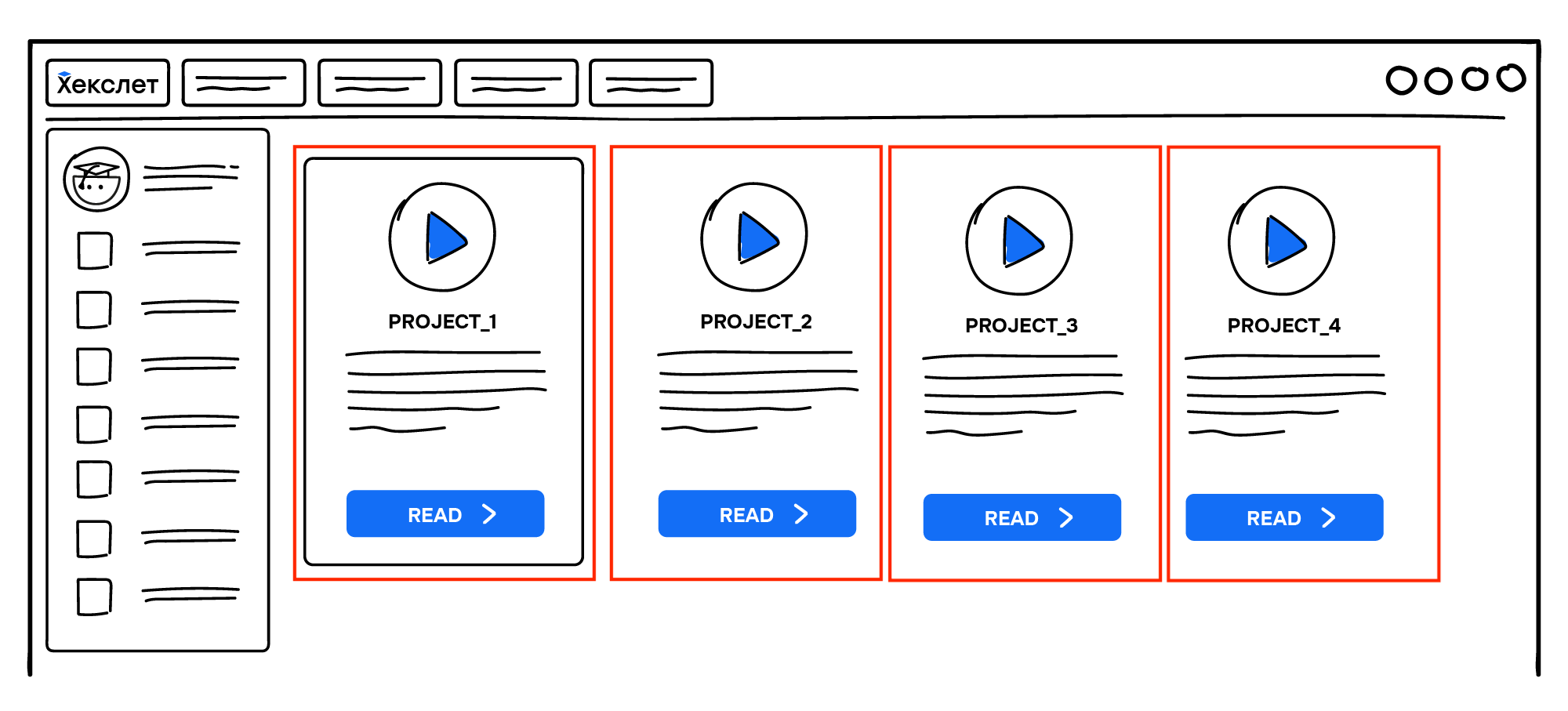
Как быстро определить, стоит ли включать участок контента в отдельную секцию? Все очень просто: если вы можете описать участок контента в одном или двух словах (преимущества, цены, форма заказа, каталог, контакты и так далее), то с большой вероятностью этот участок является самостоятельной секцией.

## **Независимые секции**

Другим способом выделить логический участок текста является использование тега <article>. У вас может возникнуть вопрос: «Зачем иметь два различных тега для выделения одного и того же?»

Между <section> и <article> есть одна существенная разница: <article> является независимой секцией, то есть ее можно перенести на любую страницу сайта или даже на другой сайт, и при этом она не потеряет своего контекста.

Представьте себе блог и отдельную статью в нем. Можем ли мы понять статью, если она вдруг окажется не в блоге, а, например, на странице с услугами? Конечно! Ведь статья — это законченный текст. Следовательно, такую статью можно обернуть тегом <article>.



Добавим колонку новостей в наш пример верстки. Сразу подумаем, как она может быть разделена. Сами по себе новости являются достаточно уникальным элементом, ведь даже если их перенести на другую страницу, то они не потеряют свою актуальность. В этом случае каждую новость можно обернуть в <article>. А что делать с оберткой блока? Она объединяет по смыслу несколько различных новостей, ее можно спокойно назвать одним словом, и она точно будет иметь свой заголовок. Следовательно, ей подойдет тег <section>.

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

<main>

<h1>Хекслет — практические курсы по программированию</h1>

<p>

Мы считаем, что настоящий программист должен понимать, как работает

компьютер, и обладать вычислительным мышлением. Он должен видеть проблему, а

не задание. Должен уметь анализировать и рассуждать на уровне проблемы и

выше, а не только на уровне кода.

</p>

<p>

При сегодняшнем обилии учебных материалов, курсов и книг, главный вопрос,

который встает перед новичком, это не «где учиться?», а «что и в каком

порядке учить?». Существует множество мнений по этому поводу. Кто-то

советует начинать с математики, кто-то советует конкретные языки и

технологии.

</p>

<p>

Хекслет — это готовый путь от абсолютного новичка до первого

трудоустройства.

</p>

<section>

<h2>Преимущества</h2>

<ul>

<li>Много теории</li>

<li>Много практики</li>

<li>Много менторов</li>

</ul>

</section>

<section>

<h2>Новости</h2>

<article>

<h3>Новость 1</h3>

<p>Текст новости 1</p>

<a href="#">Подробнее</a>

</article>

<article>

<h3>Новость 2</h3>

<p>Текст новости 2</p>

<a href="#">Подробнее</a>

</article>

<article>

<h3>Новость 3</h3>

<p>Текст новости 3</p>

<a href="#">Подробнее</a>

</article>

</section>

</main>

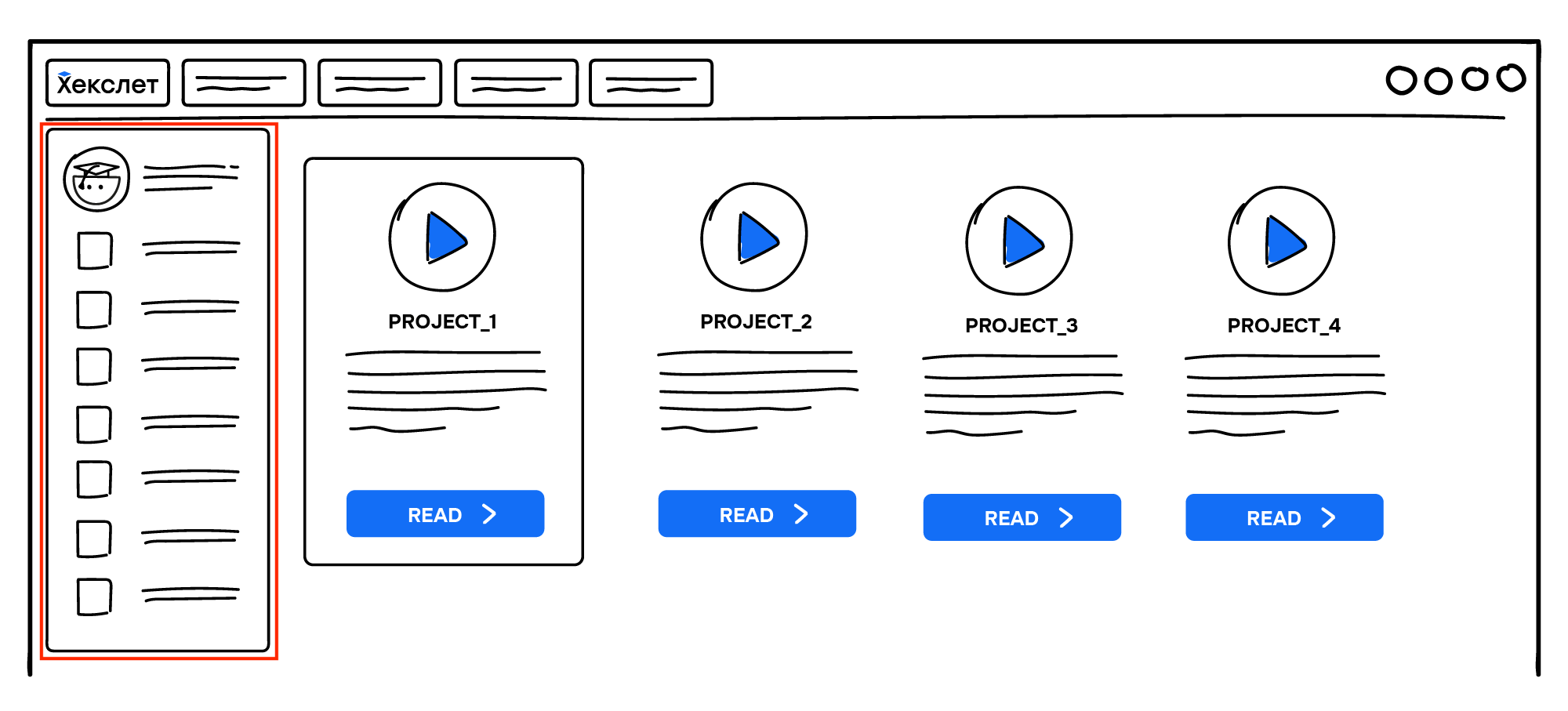
Начало формы

Конец формы

## **Дополняющие секции**

Еще одним крупным контейнером для нашего контента служит тег <aside>. Это область с дополнительной информацией. Она может быть как связана с текущей страницей, так и не очень. Подобные секции вы можете встречать в виде боковых панелей на сайтах. Там содержится дополнительное меню, баннеры, реклама и другая информация.

Обратите внимание, что <aside> не обязан являться боковой панелью по внешнему виду. Это может быть даже дополнительная информация внутри статьи. Но чаще всего внешнее оформление у такого тега именно в виде боковой панели.



Давайте добавим такую информацию в нашу верстку. Внутри этой дополнительной секции будет находиться еще одно меню, которое не будем оборачивать в <nav>, так как оно не является основным.

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

<aside>

<div>

<a href="#">Необязательный пункт 1</a>

<a href="#">Необязательный пункт 2</a>

<a href="#">Необязательный пункт 3</a>

</div>

</aside>

<main>

<h1>Хекслет — практические курсы по программированию</h1>

<p>

Мы считаем, что настоящий программист должен понимать, как работает

компьютер, и обладать вычислительным мышлением. Он должен видеть проблему, а

не задание. Должен уметь анализировать и рассуждать на уровне проблемы и

выше, а не только на уровне кода.

</p>

<p>

При сегодняшнем обилии учебных материалов, курсов и книг, главный вопрос,

который встает перед новичком, это не «где учиться?», а «что и в каком

порядке учить?». Существует множество мнений по этому поводу. Кто-то

советует начинать с математики, кто-то советует конкретные языки и

технологии.

</p>

<p>

Хекслет — это готовый путь от абсолютного новичка до первого

трудоустройства.

</p>

<section>

<h2>Преимущества</h2>

<ul>

<li>Много теории</li>

<li>Много практики</li>

<li>Много менторов</li>

</ul>

</section>

<section>

<h2>Новости</h2>

<article>

<h3>Новость 1</h3>

<p>Текст новости 1</p>

<a href="#">Подробнее</a>

</article>

<article>

<h3>Новость 2</h3>

<p>Текст новости 2</p>

<a href="#">Подробнее</a>

</article>

<article>

<h3>Новость 3</h3>

<p>Текст новости 3</p>

<a href="#">Подробнее</a>

</article>

</section>

</main>

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Моя первая страница</title>

</head>

<body></body>

</html>

В данном коде:

* <!DOCTYPE html> указывает на тип документа (HTML5).
* <html lang="ru"> определяет язык документа (русский).
* <head> содержит метаинформацию о документе, такую как кодировка символов и заголовок страницы.
* <meta charset="UTF-8"> задает кодировку документа (UTF-8).
* <title>Моя первая страница</title> устанавливает заголовок страницы.
* <body></body> содержит содержимое страницы, которое отображается в браузере.

## **DOCTYPE**

Первая конструкция в любом HTML-документе — элемент <!DOCTYPE>. Он не относится к тегам и никаким образом не может отображаться на странице. Его задача — указать браузеру, какой стандарт HTML используется в этом документе. Сейчас это везде стандарт HTML5. Записывается он следующим образом:

**<!DOCTYPE html>**

## **Парный тег html**

Тег <html></html> является основой основ. Именно внутри него располагается вся информация. Благодаря этому тегу браузер понимает, где начинается контент, который необходимо обработать как HTML.

Важной частью тега html является наличие атрибута lang. В нем указывается язык, на котором отображается веб-страница. С помощью этого атрибута браузеры могут корректно считать множество специфичных символов, которые присутствуют в разных языках. Помимо этого, атрибут lang начинает использоваться и в CSS, с которым вы познакомитесь в следующих уроках. В новых стандартах CSS появляются свойства, которые опираются на данный атрибут. Например, позволяют корректно переводить слова в тексте.

В качестве значения атрибут lang принимает знакомые всем коды языков. Для русского — lang="ru", для английского — lang="en", для немецкого — lang="de".

## **Парный тег head**

Тег <head></head> служит для хранения служебной информации. Здесь возможны самые разные сочетания тегов, которые подсказывают браузеру название страницы, описание, ключевые слова и так далее. Такая информация называется метаинформацией. В современном вебе она отвечает не только за служебную информацию для браузера, но и активно используется при продвижении сайта. Поисковые системы считывают всю эту информацию и на основе множества алгоритмов определяют место сайта при разных поисковых запросах.

Любые данные, которые указаны внутри тега <head>, не видны при отображении страницы в браузере. Это значит, что нет необходимости располагать там информацию, которая предназначена для отображения.

Хоть различной информации внутри <head> может быть множество, в этом уроке разберем несколько основных тегов, которые пригодятся при создании любой веб-страницы:

### **Метаинформация**

Метатег <meta>. Он принимает множество разных атрибутов, с которыми вы познакомитесь при создании своих сайтов. В настоящее время важным является метатег <meta> с атрибутом charset. Он позволяет установить кодировку документа.

Кодировка — таблица символов. В ней каждый символ имеет уникальный код, благодаря чему программы, в том числе и браузеры, могут одинаково отображать один и тот же текст. У разных пользователей может стоять различная кодировка по умолчанию. Это приводит к тому, что у некоторых пользователей текст может отображаться в виде «кракозябр», хотя у вас он будет отображаться правильно. Универсальной кодировкой, которая содержит большинство необходимых символов из разных языков, является кодировка UTF-8. Именно ее рекомендуется устанавливать в качестве значения атрибута charset. Теперь браузер будет отображать все символы именно в этой кодировке.

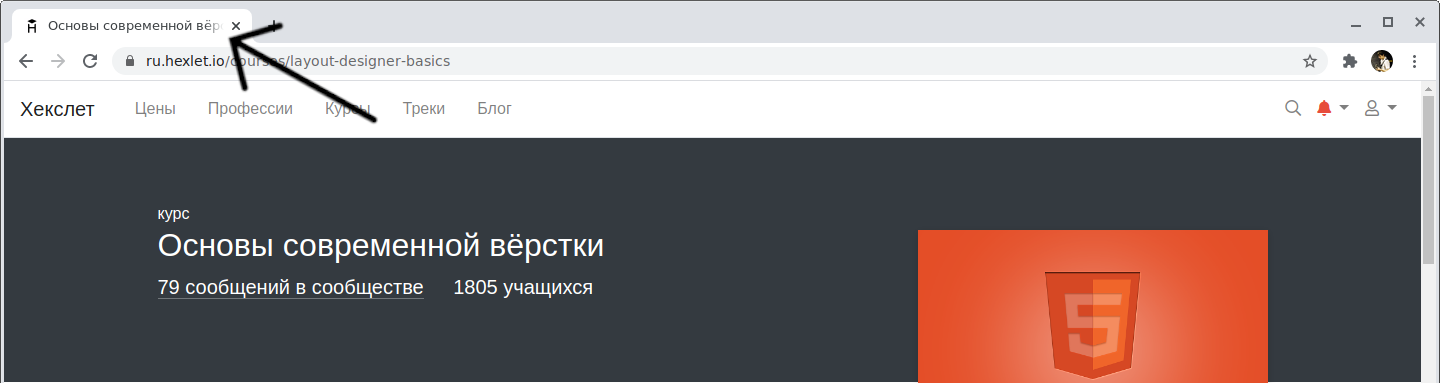
<meta charset="UTF-8">

Начало формы

Конец формы

### **Заголовок страницы**

На любом веб-сайте вы можете заметить заголовок, который отображается на вкладке вашего браузера. Например, на странице курса «Основы современной верстки» вкладка в браузере Google Chrome выглядит следующим образом:



Для указания заголовка страницы используется специальный парный тег <title></title>, внутри которого указывается нужная информация.

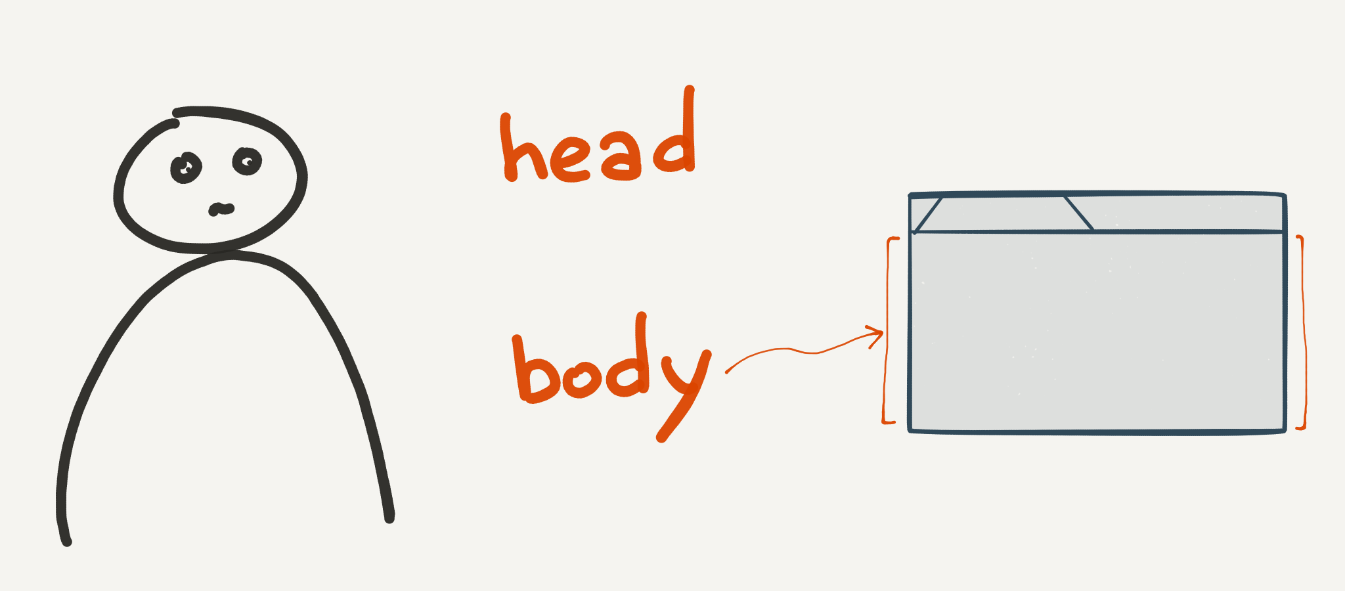
<title>Моя первая страница</title>

Начало формы

Конец формы

## **Тело документа**

После тега <head> в документе указывается парный тег <body></body>, который является «телом» всей страницы. Именно здесь размещается вся информация, которая будет выведена на странице.



Используем один из примеров прошлого урока и добавим все недостающие теги.

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

Код на языке HTML представляет собой верстку шапки сайта. Внутри тега <header> содержатся логотип сайта в виде изображения и навигационное меню.

Ошибка в коде отсутствует.

Объяснение кода:

* Тег <header> используется для обозначения шапки сайта.
* Тег <img> используется для отображения изображения (логотипа) на странице. Атрибут src указывает путь к изображению, а атрибут alt предоставляет текстовое описание изображения для случаев, когда изображение не может быть загружено или отображено.
* Тег <nav> используется для создания навигационного меню на странице. Атрибут id="menu" задает идентификатор для меню, который может быть использован для стилизации или скриптов.
* Тег <ul> создает маркированный список, в данном случае список пунктов меню.
* Тег <li> используется для создания отдельных пунктов списка.
* Тег <a> создает ссылку. Атрибут href указывает адрес, на который будет осуществлен переход при клике на ссылку.

Начало формы

Конец формы

Чтобы полностью соответствовать всем стандартам HTML, добавим необходимую базовую структуру документа.

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Моя первая страница</title>

</head>

<body>

<header>

<img src="/logo.png" alt="Логотип">

*<!-- Логотип сайта -->*

<nav id="menu">

*<!-- Меню -->*

<ul>

<li><a href="/">Главная</a></li>

<li><a href="/about">О нас</a></li>

<li><a href="/contacts">Контакты</a></li>

</ul>

</nav>

</header>

</body>

</html>

Начало формы

Конец формы

Хоть данный набор и является основным, но на самом деле браузеры могут обработать HTML-информацию и без базовой структуры документа. Но не стоит отдавать все на откуп браузеру. Он постарается автоматически обернуть контент в <body>, добавит современный <!DOCTYPE>. Но при этом нет уверенности в том, что все это он добавит корректно.

**CSS** переводится как каскадные таблицы стилей (Cascading Style Sheets).

Этот CSS код применяется к элементам <p> (параграфам) на веб-странице. Он задает красный цвет текста (color: red;) и размер шрифта 20 пикселей (font-size: 20px;).

Пример использования:

**<!DOCTYPE html>**

<html>

<head>

<style>

p {

**color**: red;

**font-size**: 20px;

}

</style>

</head>

<body>

<p>Этот текст будет красным и размером шрифта 20 пикселей.</p>

</body>

</html>

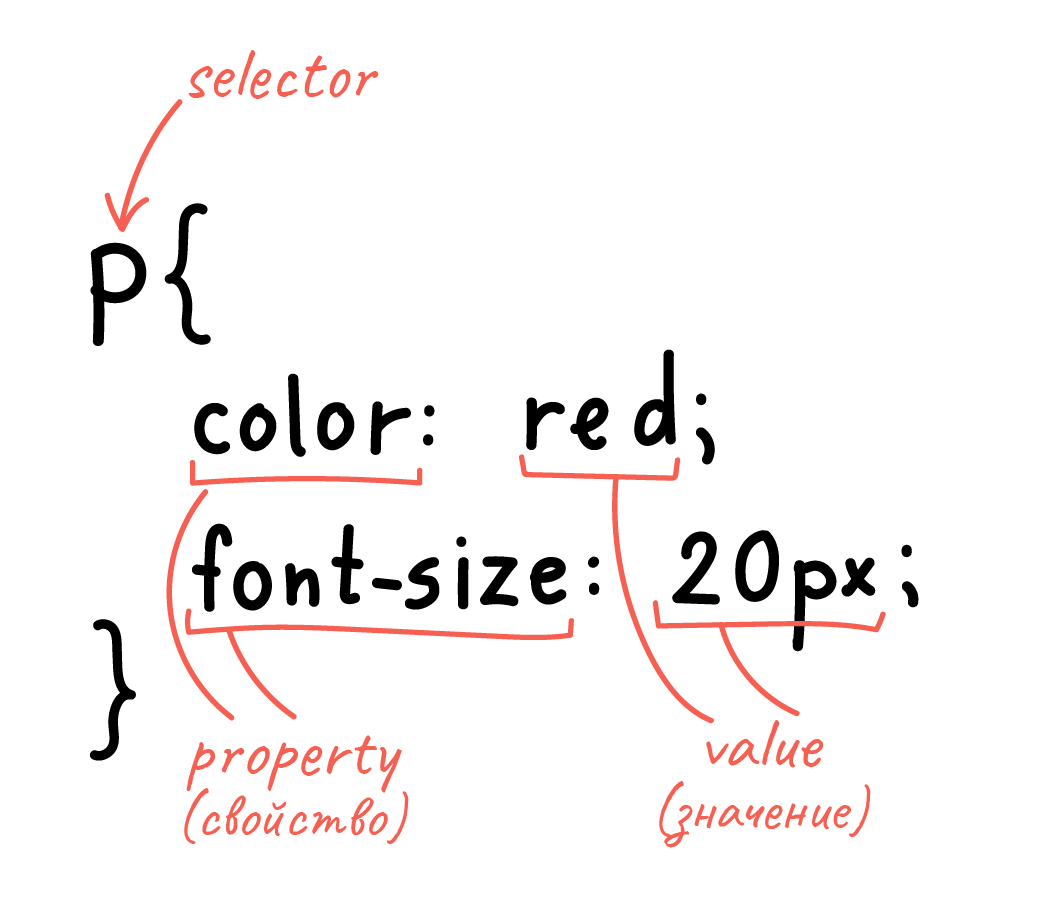
p {

**color**: red;

**font-size**: 20px;

}

1. p — это *селектор*. Здесь мы говорим, чтобы браузер выбрал все параграфы на странице. О том, какие бывают селекторы, мы поговорим чуть позже.
2. Фигурные скобки { }. Они отделяют селектор от правил. Все правила записываются уже внутри этих фигурных скобок.
3. Свойства. Они записываются по схеме, которая представлена ниже. Чем больше у вас будет практики, тем больше различных свойств и их значений вы будете знать. Не пытайтесь выучить их все сразу. Всегда используйте документацию. Даже опытные разработчики прибегают к ней, чтобы вспомнить правильные значения того или иного свойства.



Разберем некоторые свойства, которые помогут вам оформлять текст:

* font-size. Это свойство позволяет задать размер шрифта. Со временем вы узнаете множество различных единиц, в которых можно записать значение. На начальном этапе используйте значения в пикселях. Это достаточно простая и понятная единица измерения, которая не зависит от того, какие еще стили указаны у текста.
* color. Это свойство поможет вам установить новое значение цвета для выбранного текста. Цвет может записываться разными способами. Помимо этого есть большой набор цветов, которые можно записать *словами*. Это отличный способ познакомиться со свойством color. Посмотреть на все такие записи можно здесь.
* text-align. Свойство, устанавливающее выравнивание текста. Оно может принимать одно из следующих значений: left, right, center, justify.

### **Использование отдельного CSS-файла**

Данный способ один из самых удобных в реальной разработке. Так как стилей в проекте обычно много, то держать их в HTML-файле не очень удобно. Связано это с тем, что HTML-файл становится очень большим и ориентироваться в нем становится невозможно. Чтобы этого избежать, можно создать отдельный CSS-файл. Этот файл будет иметь расширение .css, имя же может быть произвольным.

Представим, что на компьютере создана отдельная директория с проектом. Она может выглядеть следующим образом:

site/

├── css/

│ ├── main.css

├── html/

│ ├── index.html

Начало формы

Конец формы

Добавим основную информацию в файл index.html:

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Основной файл index.html</title>

</head>

<body></body>

</html>

Начало формы

Конец формы

Теперь необходимо подключить файл main.css в наш файл index.html. Для этого используется специальный тег <link>. У данного тега необходимо указать атрибут href, значением которого будет путь до файла main.css. Тег <link> указывается в секции <head>.

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Основной файл index.html</title>

<link rel="stylesheet" href="../css/main.css">

</head>

<body></body>

</html>

Начало формы

Конец формы

Дополнительно разберем эту запись: ../css/main.css. Ее можно условно разбить на три составляющие:

1. ../ — переход в директорию выше. Изначально файл index.html находится в директории html/, поэтому мы «выходим» из нее в директорию site/.
2. css/ — переход в директорию css/.
3. main.css — указание имени и расширения файла, который мы хотим подключить.

Если бы наши файлы index.html и main.css находились в одной директории, то подключение выглядело бы следующим образом: href="main.css".

После подключения файла мы можем записывать все нужные стили именно в файле main.css. Они автоматически подключатся на наш сайт.

### **Использование тега style**

Вторым способом использования CSS является использование стилей в специальном теге <style>. Для этого нет необходимости создавать отдельный файл и подключать его к HTML. В этом случае тег <style> обычно указывается внутри секции <head>. Хоть это и не обязательное правило, но желательно придерживаться именно такой структуры.

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Основной файл index.html</title>

<style>

p {

**color**: red;

**font-size**: 20px;

}

</style>

</head>

<body>

<p>Параграф с размером шрифта 20 пикселей. Текст написан красным цветом.</p>

</body>

</html>

Начало формы

Конец формы

Данный способ не является основным, хотя и рекомендуется для некоторых стилей. Это позволит отобразить некоторые стили еще до того, как загрузится CSS-файл. Какие стили рекомендовано так выносить, вы сможете узнать в процессе становления вас как профессионала.

### **Inline-стили**

Последним вариантом использования CSS являются инлайн-стили. Они подключаются с помощью атрибута style у любого тега в разметке. Это наименее предпочтительный способ. Его тяжело читать, особенно если у тега десяток стилей. К тому же вы не сможете записать стили для всех одинаковых тегов. Придется для каждого указывать отдельно. Это приведет к постоянному копированию стилей.

Возьмем прошлый пример и добавим inline стили:

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Основной файл index.html</title>

</head>

<body>

<p style="color: red; font-size: 20px;">

Параграф с размером шрифта 20 пикселей. Текст написан красным цветом.

</p>

</body>

</html>

Начало формы

Конец формы

## **Классы и идентификаторы**

При создании разметки разработчикам удобно отделять одинаковые теги друг от друга с помощью имён. Например, на странице есть два заголовка:

<h2>Как стать программистом</h2>

*<!-- Тут много текста -->*

<h2>Какими навыками обладает программист</h2>

*<!-- Тут много текста -->*

Начало формы

Конец формы

Оба заголовка второго уровня, но в дизайне они могут иметь разные CSS стили. Вы уже познакомились с селектором по тегу и он не поможет дать каждому из заголовков уникальные стили.

Дать имена тегам позволяют атрибуты class и id. Эти атрибуты принимают произвольные имена, с помощью которых можно обратиться к элементу в CSS.

Различие класса и идентификатора в количестве использований:

* class — значение атрибута может устанавливаться сколько угодно раз в рамках одной страницы
* id — значение атрибута устанавливается у одного элемента на странице. Не может быть двух одинаковых идентификаторов на странице

Если значений у атрибута несколько, то они указываются через пробел, например:

<h2 class="title title-big">Как стать программистом</h2>

*<!-- Тут много текста -->*

<h2 class="another-title title-big">Какими навыками обладает программист</h2>

*<!-- Тут много текста -->*

<div id="help">Второго элемента с идентификатором help быть не должно</div>

<div id="another-help">Второго элемента с идентификатором another-help быть не должно</div>

Начало формы

Конец формы

Чтобы обратиться к элементам по их классу или идентификатору, используются специальные селекторы, которые мы сейчас и рассмотрим.

## **Селекторы**

Селекторы — краеугольный камень всего CSS.

«Не так важны стили, как то, к чему они применяются» © Народная мудрость

Селекторы позволяют выбирать элементы, к которым будут применяться стили. Это очень мощный и многогранный инструмент, который позволяет выбирать не только по тегу, но и по классу, идентификатору, наведению мыши, разным атрибутам, выбирать только четные или нечетные элементы и так далее.

Разбор каждого селектора — это тема для целого курса. Сейчас нас интересуют три основных селектора, которые можно назвать базовыми:

1. Селектор по тегу. Такой селектор мы применяли на протяжении всего урока. Цель простая — просто указать тег, для которого будут применяться стили.
2. Селектор по классу. Если нужно выбрать только конкретные элементы в HTML, то можно дать тегу произвольный класс и, используя конструкцию .название-класса, добавить стили.
3. Селектор по идентификатору. Еще один случай выборки по дополнительному атрибуту тега. Главная разница заключается в самом HTML — идентификатор не может повторяться в рамках одной страницы. Одно имя может использоваться только один раз. В связи с этим нет возможности задать стили для идентификатора и использовать его на других элементах. Для выбора идентификатора используется конструкция #имя-идентификатора.

Рассмотрим все три селектора на реальном примере:

**<!DOCTYPE html>**

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Основной файл index.html</title>

<link rel="stylesheet" href="../css/main.css">

</head>

<body>

<main class="content">

<h1 id="main-title">Заголовок с идентификатором</h1>

<p>Просто параграф без класса или идентификатора</p>

</main>

</body>

</html>

Код написан на языке HTML и не содержит ошибок.

Объяснение кода:

* <!DOCTYPE html>: Объявление типа документа, указывает на то, что это HTML5.
* <html lang="ru">: Начало HTML документа с указанием языка страницы (русский).
* <head>: Секция заголовка документа, содержит метаинформацию о странице.
  + <meta charset="UTF-8">: Устанавливает кодировку документа как UTF-8, чтобы корректно отображать текст на разных языках.
  + <title>Основной файл index.html</title>: Устанавливает заголовок страницы, который отображается во вкладке браузера.
  + <link rel="stylesheet" href="../css/main.css">: Подключение внешнего CSS файла для стилизации страницы.
* <body>: Секция содержимого страницы.
  + <main class="content">: Основной контент страницы, помеченный классом "content".
  + <h1 id="main-title">Заголовок с идентификатором</h1>: Заголовок первого уровня с уникальным идентификатором "main-title".
  + <p>Просто параграф без класса или идентификатора</p>: Обычный параграф без класса или идентификатора.

Начало формы

Конец формы

**#main-title** {

**color**: white;

**font-size**: 20px;

**text-align**: center;

}

**.content** {

**background-color**: black;

}

p {

**color**: white;

}

### **Объяснение кода:**

1. Селектор #main-title задает стили для элемента с id "main-title":
   * Цвет текста устанавливается белым (color: white;)
   * Размер шрифта устанавливается 20 пикселей (font-size: 20px;)
   * Выравнивание текста по центру (text-align: center;)
2. Селектор .content задает стили для элементов с классом "content":
   * Цвет фона устанавливается черным (background-color: black;)
3. Селектор p задает стили для всех абзацев на странице:
   * Цвет текста устанавливается белым (color: white;)

Начало формы

Конец формы

После применения этих стилей получится следующая картина:

* Черный фон секции <main>. Для этого использовали класс content и селектор .content;
* Заголовок первого уровня имеет белый цвет, размер шрифта 20 пикселей и расположение по центру. У заголовка есть идентификатор main-title, поэтому для добавления стилей мы смогли использовать селектор по идентификатору #main-title;
* Все параграфы будут иметь белый цвет текста. Для этого использовался селектор по тегу.

### **Вложенность селекторов**

В реальной разработке не всегда удается пользоваться только этими тремя видами селекторов в том виде, в котором вы их только что изучили. Зачастую нам необходимо ограничить область, для которой будут применяться стили. Разберем на примере:

<section class="news">

<h2>Новости</h2>

<article>

<h2>Новость 1</h2>

<p>Текст новости</p>

</article>

<article>

<h2>Новость 2</h2>

<p>Текст новости</p>

</article>

</section>

<section class="stories">

<h2>Интересные истории</h2>

<article>

<h2>История 1</h2>

<p>Текст истории</p>

</article>

<article>

<h2>История 2</h2>

<p>Текст истории</p>

</article>

</section>

Начало формы

Конец формы

Как добиться того, чтобы заголовки новостей были больше? Можно для каждого заголовка в новости добавить свой класс. Это хорошее решение, если новостей немного или нет необходимости для каждой такой секции вводить новые стили для заголовков.

Более простым решением станет использование вложенности селекторов. Вначале посмотрим, как решить нашу задачу:

**.news** article h2 {

**font-size**: 32px;

}

Начало формы

Конец формы

Этот селектор комбинирует два уже известных типа:

1. Селектор по классу .news
2. Селекторы по тегу article и h2

Разделив их пробелом, мы сказали браузеру: «Возьми все заголовки <h2> из <article>, который лежит внутри элемента с классом news». В примере это элемент <section class="news">.

Читать селекторы стоит именно таким образом — справа налево. Такой подход называется вложенностью селекторов. Можно комбинировать что угодно и с какой угодно глубиной.

**Важно:** для удобной работы не стоит использовать очень большую вложенность. Старайтесь использовать комбинацию не более 2 или 3 селекторов

Важно понимать, что такая запись .news article h2 выберет **все** заголовки второго уровня во **всех** <article>, которые лежат внутри блока с классом news. Давайте немного видоизменим верстку, чтобы это проверить.

<section class="news">

<h2>Новости</h2>

<article>

<h2>Новость 1</h2>

<p>Текст новости</p>

</article>

<article>

<h2>Новость 2</h2>

<p>Текст новости</p>

<section>

<h2>Похожие новости</h2>

<article>

<h2>Похожая новость 1</h2>

<p>Текст новости</p>

</article>

</section>

</article>

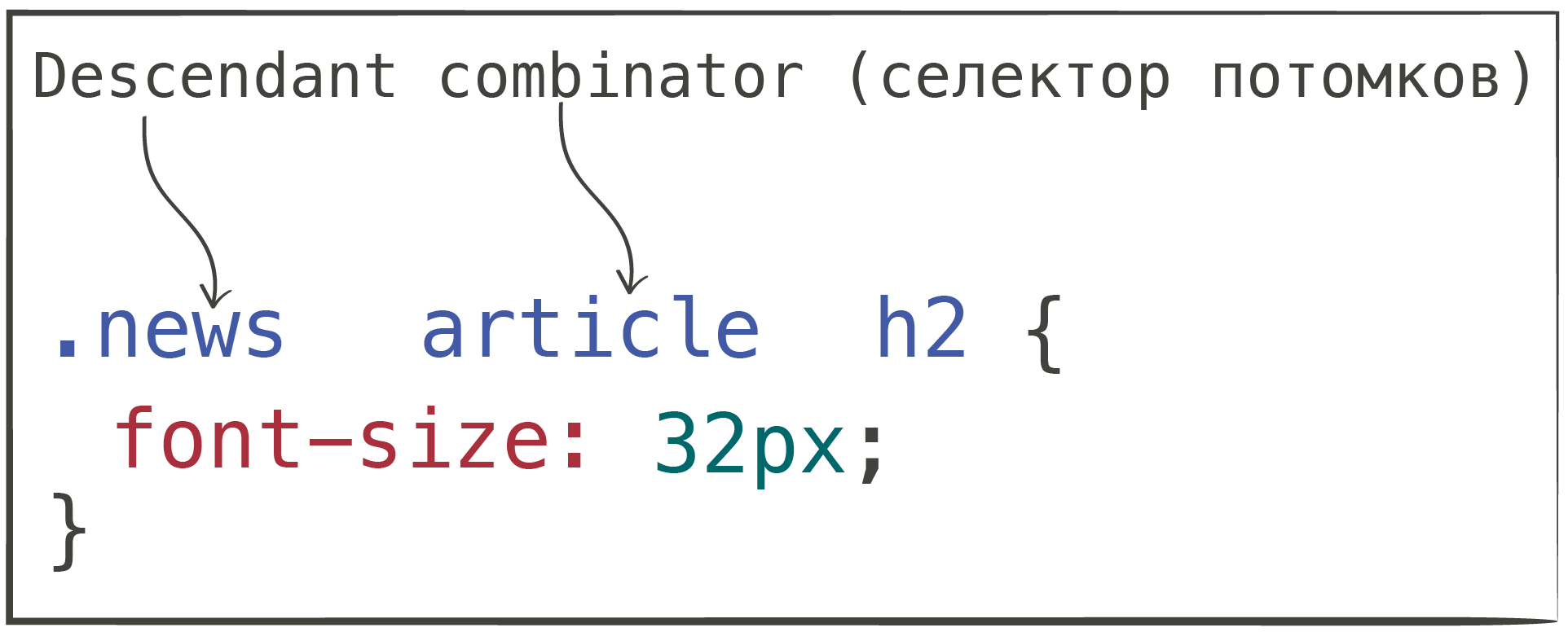
</section>

Начало формы

Конец формы

Ко второй новости мы добавили блок «Похожие новости». Самое интересное, что наше правило font-size: 32px; применится и к заголовку «Похожие новости», и к заголовку «Похожая новость 1». Так как оба заголовка так или иначе лежат внутри <article>, которые находятся в блоке с классом .news.

Такой тип селекторов называется контекстным или селектор потомков. С его помощью мы выбираем всех потомков, которые встречаются внутри самого левого селектора.



Как же нам выбрать только те заголовки, которые относятся непосредственно к новостям, а не к похожим новостям или заголовкам других секций? Здесь нам на помощь придет дочерний селектор. Его суть в выборе только тех элементов, которые непосредственно лежат внутри нужного нам блока. Без учета всех остальных вложенностей. Чтобы указать такой селектор, используется символ >. Укажем, что нам нужны только те <article>, которые лежат внутри .news. И нам нужны заголовки, которые находятся внутри этих <article>. Это делается следующим образом:

**.news** **>** article **>** h2 {

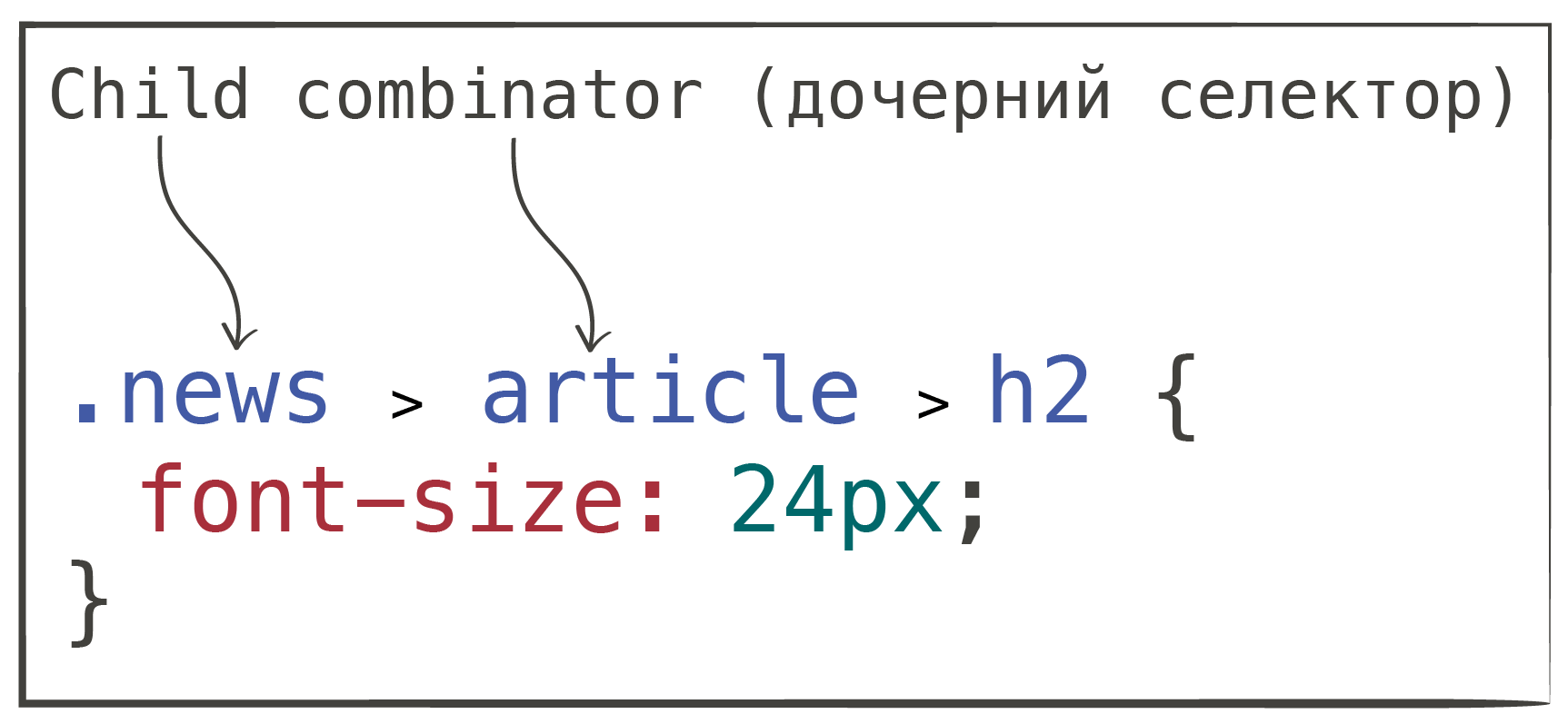
**font-size**: 24px;

}

Начало формы

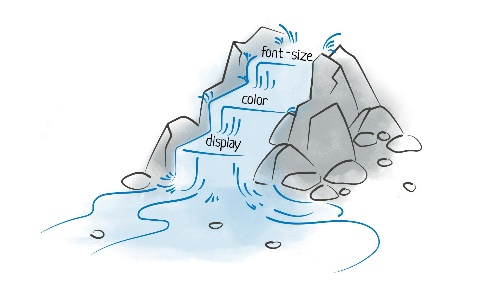
Конец формы

Теперь размер шрифта для заголовков «Похожие новости» и «Похожая новость 1» не будет увеличен, так как мы указали более конкретный селектор для заголовков.



В прошлом уроке была разобрана вторая часть словосочетания Каскадные таблицы стилей, а именно таблицы стилей. Мы изучили, как записываются и применяются стили к различным HTML элементам. В этом уроке поговорим о значении слова каскадные.

Что такое каскадность? Представьте себе каскад водопадов. В них вода перетекает из одного водопада в другой и так до тех пор, пока это все не уйдет в общий бассейн.



В CSS слово каскадность имеет такое же значение. В качестве воды у нас CSS свойства. Они могут накладываться и перекрывать друг друга. Другими словами каскадность — возможность языка CSS накладывать свойства друг на друга, а также расширять свойства в селекторах.

Возможно такое описание достаточно сложное, но давайте посмотрим на примере. Создадим параграф с двумя классами: text-bold и text-dark. Первый класс будет отвечать за жирное начертание текста, а второй за установку темного цвета шрифта. Также добавим селектор по тегу и укажем размер шрифта 24 пикселя.

<p class="text-bold text-dark">

Какой-то очень интересный текст об интересной вещи. Очень интересно.

</p>

Начало формы

Конец формы

В CSS файле укажем следующие стили:

p {

**font-size**: 24px;

}

**.text-bold** {

**font-weight**: bold;

}

**.text-dark** {

**color**: #333333;

}

Начало формы

Конец формы

Попробуйте воспроизвести этот пример, и вы увидите, что текст выводится темного цвета и с жирным начертанием. Стили от классов text-bold и text-dark сложились и применились к нашему параграфу. Также текст имеет размер 24 пикселя, которые мы указали для селектора по тегу.

Итоговыми стилями для нашего параграфа будут:

{

**color**: #333333;

**font-weight**: bold;

**font-size**: 24px;

}

Начало формы

Конец формы

Это и есть каскадность стилей CSS. Хоть мы и использовали разные селекторы и классы для описания, но в конечном итоге они суммировались.

## **Приоритет селекторов**

Важной чертой CSS является то, что разные типы селекторов имеют разный приоритет. Если стили противоречат друг другу в разных селекторах, то вступает в дело принцип приоритета. Разберем на примере. Возьмем параграф с классом red и идентификатором blue. Установим соответствующие цвета для текста. Для всех же параграфов в тексте установим зеленый цвет.

p {

**color**: green;

}

**.red** {

**color**: red;

}

**#blue** {

**color**: blue;

}

Начало формы

Конец формы

<p id="blue" class="red">Какого же цвета будет параграф?</p>

Начало формы

Конец формы

Перед тем, как читать дальше попробуйте выполнить этот пример в любом удобном вам месте.

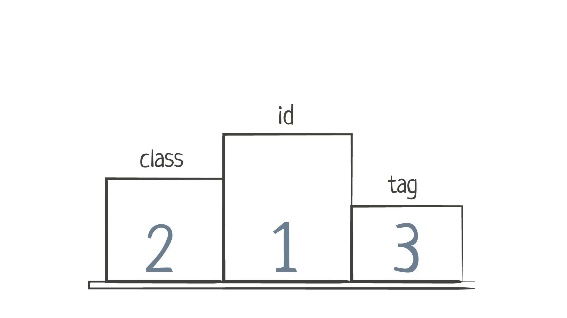
Как вы смогли заметить, текст параграфа будет синим. Но почему так? Может быть, дело в том, что синий цвет в CSS указан самым последним? В данном случае нет. Даже если мы переместим селектор по идентификатору в самый вверх, параграф все равно останется синим.

Все дело в том, что селектор по идентификатору имеет самый высокий приоритет. Встретив его, браузер будет считать эти стили самыми важными для страницы и в случае противоречий возьмет значение в первую очередь из стилей для идентификатора.

Условно можно расставить селекторы в следующем порядке по приоритету:

1. Селектор по идентификатору (#blue)
2. Селектор по классу (.red)
3. Селектор по тегу (p)

Где 1 — самый высокий приоритет, а 3 — самый низкий.



Разберем еще один пример:

p {

**color**: blue;

**font-weight**: bold;

}

**.important** {

**color**: purple;

**font-style**: italic;

}

**#intro** {

**color**: green;

}

Начало формы

Конец формы

<p class="important" id="intro">Индейские племена Манахаттоу и Канарси.</p>

Начало формы

Конец формы

Этот текст будет ***наклонным, жирным и зеленым***. И вот почему:

* Селектор по тегу p:
  + Добавит синий цвет
  + Добавит **жирное начертание**
* Селектор по классу .important:
  + Заменит ~~синий цвет~~ на пурпурный
  + Сделает текст *курсивным*
* Селектор по идентификатору #intro:
  + Заменит ~~пурпурный цвет~~ на зеленый

После всех таких преобразований итоговые стили для параграфа сложатся следующим образом:

{

**color**: green;

**font-weight**: bold;

**font-style**: italic;

}

Начало формы

Конец формы

**Важно:** так как селектор по идентификатору имеет самый высокий приоритет и перебивает все стили классов, то его использование считается не очень хорошей практикой. Указав стили для идентификатора, мы лишаем себя возможности переопределить стили с помощью классов. Это бывает очень важно для создания отзывчивых к действиям пользователя сайтов, когда на его действия мы меняем **margin-top** вид некоторых элементов.

## **Переопределение свойств**

Возможно, у вас возник вопрос, а что случится, если элемент будет иметь несколько классов, свойства которых противоречат друг другу? Здесь каскадность работает по принципу «Кто последний, тот и прав». Проще всего это так же посмотреть на примере:

**.alert** {

**color**: gray;

}

**.alert-error** {

**color**: red;

}

Начало формы

Конец формы

<p class="alert alert-error">Важное сообщение! Сообщение красного цвета</p>

Начало формы

Конец формы

Отталкиваясь от принципа каскадности, браузер вначале увидит селектор .alert и установит серый цвет текста. Далее ему встретится селектор .alert-error, который говорит установить красный цвет. И браузер просто перезапишет серый цвет на красный. Порядок классов в HTML в данном случае не играет никакой роли. Даже в таком случае

<p class="alert-error alert">Важное сообщение! Сообщение красного цвета</p>

Начало формы

Конец формы

Цвет текста будет красным. Это наглядно показывает независимость CSS от HTML. Если же поменять местами селекторы в CSS, то поменяются и итоговые стили.

**.alert-error** {

**color**: red;

}

**.alert** {

**color**: gray;

}

Начало формы

Конец формы

<p class="alert alert-error">Важное сообщение! Сообщение серого цвета</p>

Начало формы

Конец формы

## **Вес селекторов**

Во время разработки вы будете встречать разные ситуации, при которых одни селекторы будут иметь больший вес, чем другие. Причем их месторасположение в CSS файле не имеет особого значения:

<textarea class="form-input"></textarea>

Начало формы

Конец формы

**.form-input** {

**height**: 50px;

}

textarea {

**height**: 200px;

}

Начало формы

Конец формы

Какой высоты будет элемент <textarea>? Правильный ответ — 50px, так как селектор по классу имеет больший приоритет, чем селектор по тегу. Но как же задать высоту в 200px и не вводить новые классы? Достаточно подставить к тегу еще и название класса:

**.form-input** {

**height**: 50px;

}

textarea**.form-input** {

**height**: 200px;

}

Начало формы

Конец формы

Теперь для элемента <textarea> будет установлена высота в 200px. Это произошло благодаря тому, что приоритеты «складываются» в рамках одного селектора. Для большего понимания таких сложений можно представить, что общий вес селектора определяется тремя цифрами, где:

1. Первая цифра показывает количество идентификаторов в селекторе
2. Вторая цифра показывает количество классов в селекторе
3. Третья цифра показывает количество тегов в селекторе

Может звучать сложно, но концепция простая. Разберем прошлый пример:

<textarea class="form-input"></textarea>

Начало формы

Конец формы

**.form-input** {

**height**: 50px;

}

textarea {

**height**: 200px;

}

Начало формы

Конец формы

* Селектор .form-input состоит из одного класса, поэтому в качестве второй цифры поставим единицу и получим: 010
* Селектор textarea состоит из одного тега. Записываем единицу в третий разряд нашего числа: 001

Визуально понятно, что 010 > 001, поэтому свойства внутри такого правила имеют больший приоритет. Мы захотели это исправить и дополнили наш селектор:

**.form-input** {

**height**: 50px;

}

textarea**.form-input** {

**height**: 200px;

}

Начало формы

Конец формы

Теперь порядок сил во вселенной немного изменился:

* Селектор .form-input состоит из одного класса, поэтому в качестве второй цифры поставим единицу и получим: 010
* Селектор textarea.form-input состоит из одного класса и тега. Записываем единицу во второй и третий разряд нашего числа: 011

Получаем, что 010 < 011, а значит свойства внутри селектора textarea.form-input будут иметь больший приоритет.

Такой подход поможет вам лучше понимать, как действуют те или иные правила, а также почему все разработчики стараются максимально уйти от сложных селекторов. При получении опыта вы все реже будете считать вес таким образом, а уже будете изначально понимать, как расставлены приоритеты.

body {

  font-size: 18px;

  color: #333;

}

h1 {

  font-size: 60px;

  text-align: center;

  margin-top: 50px;

  margin-bottom: 10px;

}

h2 {

  margin-top: 0;

  padding-bottom: 30px;

  font-size: 25px;

  text-align: center;

}

section {

  margin-top: 50px;

  margin-bottom: 50px;

  margin-left: 0;

  margin-right: 0;

  padding-top: 0;

  padding-right: 50px;

  padding-bottom: 0;

  padding-left: 50px;

}

section > h2 {

  padding-top: 0;

  padding-right: 0;

  padding-bottom: 0;

  padding-left: 0;

  text-align: left;

  border: none;

  }

  li {

    margin-bottom: 10px;

    }

<!DOCTYPE html>

<html lang="ru">

  <head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles/app.css">

    <title>Резюме</title>

  </head>

  <body>

    <main>

      <h1>Елена</h1>

      <h2>Фокусник</h2>

      <section>

        <h2>Образование</h2>

        <ul>

      <li>Школа волшебства "Хогвартс"</li>

      <li>Курсы фокусников</li>

      <li>Университет "Фокус-покус"</li>

    </ul>

        </section>

        <section>

          <h2>Навыки</h2>

          <ul>

      <li>Достаю кролика из шляпы</li>

      <li>Разрезаю женщин в коробках</li>

      <li>Исчезаю в нужный момент</li>

    </ul>

          </section>

          <section>

            <h2>Дополнительная информация</h2>

            <p>Обладаю даром предвидения, колдую без волшебной палочки.</p>

            </section>

          </main>

  </body>

</html>

**Flex**

**Свойство display: flex** в Flexbox **превращает элемент во flex-контейнер**. В зависимости от заданного значения, оно создаёт инлайновый или блочный контейнер, а также задаёт flex-контекст каждому прямому потомку

Направь этого лягушонка на лилию справа, используя свойство justify-content, которое выравнивает элементы горизонтально и принимает следующие значения:

* flex-start: элементы выравниваются по левой стороне контейнера.
* flex-end: элементы выравниваются по правой стороне контейнера.
* center: элементы выравниваются по центру контейнера.
* space-between: элементы отображаются с одинаковыми отступами между ними.
* space-around: элементы отображаются с одинаковыми отступами вокруг них.

Например, justify-content: flex-end; сдвинет лягушонка вправо.

Теперь используй align-items, чтобы помочь лягушатам добраться до нижней части пруда. Это свойство CSS выравнивает элементы вертикально и принимает следующие значения:

* flex-start: элементы выравниваются по верхнему краю контейнера.
* flex-end: элементы выравниваются по нижнему краю контейнера.
* center: элементы выравниваются вертикально по центру контейнера.
* baseline: элементы отображаются на базовой линии контейнера.
* stretch: элементы растягиваются, чтобы заполнить контейнер.

Лягушатам нужно выстроиться в том же порядке, что и лилии, используя flex-direction. Это свойство CSS задает направление, в котором будут расположены элементы в контейнере, и принимает следующие значения:

* row: элементы размещаются по направлению текста.
* row-reverse: элементы отображаются в обратном порядке к направлению текста.
* column: элементы располагаются сверху вниз.
* column-reverse: элементы располагаются снизу вверх.

Иногда изменения порядка отображения элементов в контейнере недостаточно. В таких случаях мы можем применить свойство order для конкретных элементов. По умолчанию, значение этого свойства у элементов равно 0, но мы можем задать положительное или отрицательное целое число этому свойству.

flex-wrap, которое принимает следующие значения:

* nowrap: размеры элементов устанавливаются автоматически, чтобы они поместились в один ряд.
* wrap: элементы автоматически переносятся на новую строку.
* wrap-reverse: элементы автоматически переносятся на новую строку, но строки расположены в обратном порядке.

Два свойства flex-direction и flex-wrap используются так часто вместе, что было создано свойство flex-flow для их комбинирования. Это свойство принимает их значения, разделённые пробелом.

Например, ты можешь использовать flex-flow: row wrap, чтобы элементы располагались в ряд и автоматически переносились на новую строку.

Попробуй использовать flex-flow, чтобы повторить предыдущий уровень.

**Flex-flow** — это **сокращённая запись свойств flex-direction и flex-wrap**. [2](https://html5book.ru/css3-flexbox/)

**Виды flex-flow в зависимости от контекста**:

* **Только flex-direction**: [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + row; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + row-reverse; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + column; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + column-reverse. [1](https://wp-kama.ru/id_8045/flex-v-css.html)
* **Только flex-wrap**: [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + nowrap; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + wrap; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + wrap-reverse. [1](https://wp-kama.ru/id_8045/flex-v-css.html)
* **Сразу оба значения: flex-direction и flex-wrap**: [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + row nowrap; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + column wrap; [1](https://wp-kama.ru/id_8045/flex-v-css.html)
  + column-reverse wrap-reverse. [1](https://wp-kama.ru/id_8045/flex-v-css.html)

Свойство flex-flow позволяет определить направления главной и поперечной осей, а также возможность переноса flex-элементов при необходимости на несколько строк. [2](https://html5book.ru/css3-flexbox/)

Лягушат раскидало по всему пруду, но лилии сгруппированы в верхней части. Ты можешь использовать align-content, чтобы указать, как несколько рядов должны отделяться друг от друга. Данное свойство принимает следующие значения:

* flex-start: ряды группируются в верхней части контейнера.
* flex-end: ряды группируются в нижней части контейнера.
* center: ряды группируются вертикально по центру контейнера.
* space-between: ряды отображаются с одинаковыми расстояниями между ними.
* space-around: ряды отображаются с одинаковыми расстояниями вокруг них.
* stretch: ряды растягиваются, чтобы заполнить контейнер равномерно.

Это может запутать, но align-content отвечает за расстояние между рядами, в то время как align-items отвечает за то, как элементы в целом будут выровнены в контейнере. Когда только один ряд, align-content ни на что не влияет.

### **Использование gap в сетке**

Все браузеры поддерживают gap в макетах сетки, и в этом контексте их довольно просто понять.

row-gap указывает промежуток между рядами

column-gap указывает промежуток между столбцами

    display: flex;

    justify-content: center;

    grid-gap: 25px;

grid-gap – отступы между элементами

body {

    background-color: white;

}

body:before,

body:after {

    content: "";

    background-color: rgb(135, 135, 187);

    height: 30%;

    position: absolute;

    left: 0;

    width: 100%;

}

body:before {

    top: 0;

}

body:after {

    bottom: 0;

}

Чтобы в HTML сделать фон не на всю страницу, можно использовать **свойство «background-size»**. [1](https://timeweb.com/ru/community/articles/kak-rabotat-s-fonovymi-risunkami-v-css)[4](https://htmlbook.ru/css3-na-primerakh/masshtabirovanie-fona)

В качестве значения этого свойства можно указать **ключевое слово «cover»**. Тогда размер изображения будет таким, чтобы его ширина и высота поместились в заданную область (например, окно веб-страницы). Также можно указать **ключевое слово «contain»**, которое масштабирует картинку так, чтобы хотя бы одна сторона картинки целиком поместилась в заданную область. [4](https://htmlbook.ru/css3-na-primerakh/masshtabirovanie-fona)

Кроме того, допустимо указывать **явные размеры** по горизонтали или вертикали в процентах или других единицах CSS.

**Чтобы сделать фон HTML-страницы закрашенным разными цветами, можно использовать свойство background-color**. [1](https://ru.wikihow.com/%D0%B7%D0%B0%D0%B4%D0%B0%D1%82%D1%8C-%D1%86%D0%B2%D0%B5%D1%82-%D1%84%D0%BE%D0%BD%D0%B0-%D0%B2-HTML)[4](https://doka.guide/css/background-color/) Для этого нужно найти коды нужных цветов с помощью онлайн-инструмента, например, W3Schools HTML Color Picker. Затем добавить нужный цвет фона в свойство background-color. [1](https://ru.wikihow.com/%D0%B7%D0%B0%D0%B4%D0%B0%D1%82%D1%8C-%D1%86%D0%B2%D0%B5%D1%82-%D1%84%D0%BE%D0%BD%D0%B0-%D0%B2-HTML)

Ещё один способ — использовать **линейный градиент**. [2](https://stackoverflow.com/questions/5054415/fill-div-with-2-colors)[5](https://skillbox.ru/media/code/gradient-v-css-chto-eto-i-kak-ego-sdelat/) Для этого в CSS нужно прописать функцию linear-gradient() и передать ей несколько цветов, из которых она составит узор. По умолчанию градиент будет направлен сверху вниз. [5](https://skillbox.ru/media/code/gradient-v-css-chto-eto-i-kak-ego-sdelat/)

Также можно разделить контейнер на два и задать разный цвет фона для каждого из них. [2](https://stackoverflow.com/questions/5054415/fill-div-with-2-colors) Для этого нужно создать два блока с разными классами и задать им background-color.

Верстка сайта

1. в файле html добавляем

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

1. внутри body добавляем div с классами и **footer.**

**Футер (подвал) в HTML** задаётся элементом **<footer>**. В нём может располагаться имя автора, дата документа, контактная и правовая информация.

Чтобы определить подвал для сайта, **<footer> размещают внутри контейнера <body>**. В элементе можно использовать другие элементы, например, <p>, <a> или <span>.

**Есть ограничения для элемента <footer>**. Его нельзя вкладывать внутрь другого <footer>, а также внутрь элементов <address> и <header>. Внутри <footer> не должно быть элемента <main>.

1. Добавить картинку

<img src="./logo.png" alt="logo">

1. Шрифт

**Font-family в HTML** — это свойство, которое **устанавливает семейство шрифта**, которое будет использоваться для оформления текста содержимого. [4](https://hcdev-ru.pages.dev/css/font-family/)

**В значении свойства font-family указывается список названий шрифтов**, перечисленных через запятую. В начале списка располагают самый желаемый шрифт, затем альтернативный, а в самом конце списка — общий тип шрифта. Браузер проходит по списку слева направо и использует первый найденный в системе или на сайте шрифт. [2](https://htmlacademy.ru/blog/css/css-font)

**Если в названии шрифта имеются пробелы или символы**, то оно заключается в кавычки. Это делается для того, чтобы браузер мог понять, где начинается и заканчивается название шрифта. [3](https://html5book.ru/css-shrifty/)

**CSS определяет пять базовых семейств шрифтов**: [3](https://html5book.ru/css-shrifty/)

* **Serif** (шрифты с засечками). Например, Times New Roman, Times, Garamond, Georgia. [3](https://html5book.ru/css-shrifty/)
* **Sans-serif** (рубленые шрифты). Например, Helvetica, Geneva, Arial, Verdana, Trebuchet, Univers. [3](https://html5book.ru/css-shrifty/)
* **Monospace** (моноширинные шрифты). Например, Courier, Courier New, Andele Mono. [3](https://html5book.ru/css-shrifty/)
* **Cursive** (рукописные шрифты). Например, Comic Sans, Gabriola, Monotype Corsiva, Author, Zapf Chancery. [3](https://html5book.ru/css-shrifty/)
* **Аллегорические шрифты** (Western, Woodblock, Klingon). [3](https://html5book.ru/css-shrifty/)

**Значение по умолчанию** устанавливает шрифт, установленный в браузере по умолчанию. Обычно это Times New Roman.

5.Чтобы сделать флекс контейнер, добавляем: **display: flex**

6. Фон

CSS-свойство **background**-**color** задает **цвет** фона элемента, позволяя вам создать сплошной цветной фон.

div с классом container

стили:

**Max-width в CSS — это свойство, которое устанавливает максимальную ширину элемента**.

Значением свойства служат любые единицы для размеров либо ключевое слово

none

, означающее отсутствие значения.

**Max-width: 1180 px;**

**Margin в CSS — это свойство, которое задаёт размер внешнего отступа вокруг элемента**. Иногда внешние отступы называют полями.

**Margin позволяет**:

* задать величину отступа сразу для всех сторон элемента;
* определить её только для указанных сторон.

**Margin: 0 px auto;**

**display: flex**

1. Свойство background-color устанавливает цвет фона элемента. Цвет рисуется за фоновыми изображениями. Для блочных элементов цвет фона распространяется на всю ширину и высоту блока элемента, для строчных — только на область их содержимого.

Чтобы изменить размер фона в HTML, используется **свойство CSS background-size**. [4](https://ru.hexlet.io/qna/html/questions/kak-izmenit-razmer-fona-v-html)

**Вот несколько вариантов значений для этого свойства**:

* **cover**. [1](https://webref.ru/layout/html5-css3/background/size)[4](https://ru.hexlet.io/qna/html/questions/kak-izmenit-razmer-fona-v-html) Размер изображения будет таким, чтобы его ширина и высота поместились в заданную область (например, окно веб-страницы). [1](https://webref.ru/layout/html5-css3/background/size)
* **contain**. [1](https://webref.ru/layout/html5-css3/background/size)[4](https://ru.hexlet.io/qna/html/questions/kak-izmenit-razmer-fona-v-html) Масштабирует картинку так, чтобы хотя бы одна сторона картинки целиком поместилась в заданную область. [1](https://webref.ru/layout/html5-css3/background/size)
* **явные размеры по горизонтали или вертикали** в процентах или других единицах CSS. [1](https://webref.ru/layout/html5-css3/background/size)
* **auto**. Браузер вычисляет размер автоматически, исходя из пропорций картинки. [1](https://webref.ru/layout/html5-css3/background/size)

Если указано два значения размера через пробел, то первое значение определяет ширину по горизонтали, а второе — по вертикали. При этом пропорции игнорируются.

**Виды значений для свойства background-size в CSS**: [1](https://doka-guide.vercel.app/css/background-size/)

1. **Точный размер** в любых единицах измерения, доступных в вебе. [1](https://doka-guide.vercel.app/css/background-size/)
2. **Проценты**. Размер в процентах будет рассчитываться от размеров элемента. [1](https://doka-guide.vercel.app/css/background-size/)
3. **Ключевое слово auto**. Размер изображения остаётся неизменным. Это ключевое слово является значением по умолчанию. [1](https://doka-guide.vercel.app/css/background-size/)
4. **Ключевое слово cover**. Изображение масштабируется без изменения пропорций так, чтобы закрыть фон всего элемента. Короткая сторона изображения подстраивается под соответствующую сторону элемента, а длинная сторона закрывает длинную сторону элемента, а остальное, что не попало в поле видимости, обрезается. [1](https://doka-guide.vercel.app/css/background-size/)
5. **Ключевое слово contain**. Изображение масштабируется без изменения пропорций так, чтобы полностью поместиться в элемент. [1](https://doka-guide.vercel.app/css/background-size/)
6. Высота фона height: 65 px;
7. Свойство justify-content **выравнивает элементы, когда они не используют все доступное пространство по горизонтали.** Оно является подсвойством модуля макет гибкого контейнера. Свойство контролирует выравнивание элементов, когда они выходят за границы.

Свойство justify-content должно быть использовано вместе со свойством display, которое имеет значение "flex". Для вертикального выравнивания элементов нужно использовать свойство align-items.

justify-content: center;

align-items: center;

1. Работа с текстом:

colour: black;

font-size: 12 px;

**Свойство letter-spacing в CSS устанавливает интервал между символами**. [1](https://snipp.ru/html-css/font-distance)[2](https://htmlbook.ru/css/letter-spacing) С его помощью можно сжать или растянуть текст на веб-странице, при этом расстояние между буквами и другими символами уменьшается или увеличивается, а сами символы остаются прежнего размера. [3](https://doka.guide/css/letter-spacing/)

**Синтаксис**: letter-spacing: значение | normal | inherit. [2](https://htmlbook.ru/css/letter-spacing)

**Значения**:

* **normal** — стандартное межбуквенное расстояние, которое задано для шрифта (значение по умолчанию); [3](https://doka.guide/css/letter-spacing/)
* **конкретное значение** — положительное или отрицательное значение в любых единицах величины. Положительное значение добавляется к стандартному расстоянию между буквами, а отрицательное — вычитается из него.

**letter-spacing**: 0,3 px.

**Для вставки прямоугольной секции с закруглёнными углами с помощью CSS используется свойство border-radius**.

#ugolkrug {

width: 200px;

height: 100px;

color: #0000;

background: #FF7F00;

border: 3px #CCCCCC solid;

-moz-border-radius: 10px; /\* закругление для старых Mozilla Firefox \*/

-webkit-border-radius: 10px; /\* закругление для старых Chrome и Safari \*/

-khtml-border-radius:10px; /\*закругление для браузера Konquerer системы Linux \*/

border-radius: 10px; /\* закругление углов для всех, кто понимает \*/

}

**CSS-свойство border** используется для задания стиля, толщины и цвета границы элемента. Граница — это линия, которая окружает элемент и отделяет его контент от соседних элементов.

**Некоторые значения свойства border:**

* **border-width** — задаёт толщину границы. Может быть задано в пикселях (px), процентах (%) или описано ключевым словом — thin, medium, thick. [2](https://htmlacademy.ru/blog/css/css-border)
* **border-style** — устанавливает стиль рамки. Может принимать одно из значений — none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset. [2](https://htmlacademy.ru/blog/css/css-border)
* **border-color** — определяет цвет рамки. Может быть задан в виде ключевого слова, соответствующего названию цвета, hex-кода, rgb или hls значений. Значение currentColor использует текущий цвет текста в качестве цвета границы. [2](https://htmlacademy.ru/blog/css/css-border)

**Для установки границы только на определённых сторонах элемента** можно использовать свойства border-top, border-bottom, border-left, border-right.

Для создания контура вокруг рамки в CSS используются **свойства, начинающиеся со слова «outline»**. [3](https://www.seostop.ru/sozdanie-saita/html-css/stili-ramok.html)

**Некоторые из них**:

* **outline-width**. [1](https://doka.guide/css/outline/)[3](https://www.seostop.ru/sozdanie-saita/html-css/stili-ramok.html) Задаёт толщину контура в тех же значениях, что и border-width. [3](https://www.seostop.ru/sozdanie-saita/html-css/stili-ramok.html)
* **outline-style**. [1](https://doka.guide/css/outline/)[3](https://www.seostop.ru/sozdanie-saita/html-css/stili-ramok.html) Определяет стиль внешнего контура. [2](https://msiter.ru/tutorials/css-nachalnogo-urovnya/css_outline) Может быть таким: dotted (контур состоит из точек), dashed (в виде пунктирной линии), solid (сплошная линия), double (двойная сплошная линия) и другие. [1](https://doka.guide/css/outline/)[2](https://msiter.ru/tutorials/css-nachalnogo-urovnya/css_outline)
* **outline-color**. [1](https://doka.guide/css/outline/)[2](https://msiter.ru/tutorials/css-nachalnogo-urovnya/css_outline) Определяет цвет внешнего контура. Можно задать как имя цвета, HEX-значение, RGB-значение, HSL-значение или инверсию цвета (позволяет сделать внешний контур видимым независимо от цвета фона). [2](https://msiter.ru/tutorials/css-nachalnogo-urovnya/css_outline)
* **outline**. Короткое свойство, объединяющее все предыдущие свойства. [2](https://msiter.ru/tutorials/css-nachalnogo-urovnya/css_outline)

**Flex-grow в CSS** — это **свойство, которое определяет, может ли начальная ширина flex-элемента увеличиваться (расти)**. Увеличение ширины flex-элемента осуществляется за счёт свободного пространства линии. [1](https://itchief.ru/html-and-css/flexbox)

В качестве значения CSS-свойства flex-grow указывается целое число. Именно это значение и определяет (если оно больше или равно 1), какую часть свободного пространства flex-элемент заберёт себе. [1](https://itchief.ru/html-and-css/flexbox)

По умолчанию CSS свойство flex-grow имеет значение 0, что означает, что flex-элемент не может расти (увеличивать свою ширину). [1](https://itchief.ru/html-and-css/flexbox)

Если задать flex-grow значение больше нуля, то флекс-элемент растянется на всё оставшееся во флекс-контейнере свободное пространство.

**Создание сайта Cornelia**

* 1. в файле html нажать !
  2. Title – заголовок сайта
  3. Связь с файлом css:

<link rel="stylesheet" href="main.css">

* 1. Добавить иконку для сайта: добавить внутри тега <head></head> данный код

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

Чтобы менять цвет иконки сайта в зависимости от темы, можно использовать медиафункцию prefers-color-scheme. С помощью медиа-запроса CSS prefers-color-scheme определить светлую или тёмную цветовую тему, которую предпочитает пользователь.

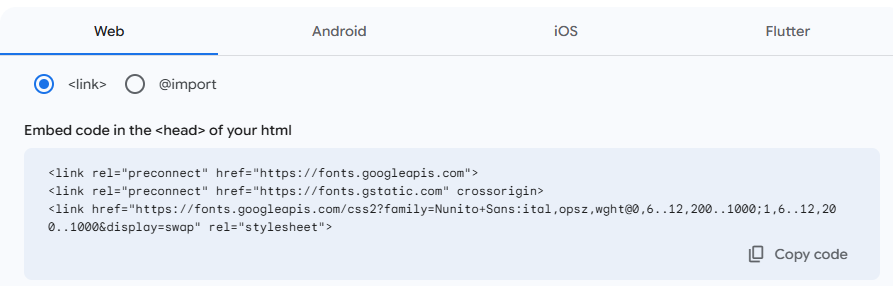
Тег <svg> используется как контейнер для хранения SVG-графики. SVG (Scalable Vector Graphics — масштабируемая векторная графика) — язык разметки, расширенный из XML для описания двухмерной векторной графики.

Тег <svg> парный, содержимое записывается между открывающим (<svg>) и закрывающим (</svg>) тегами.

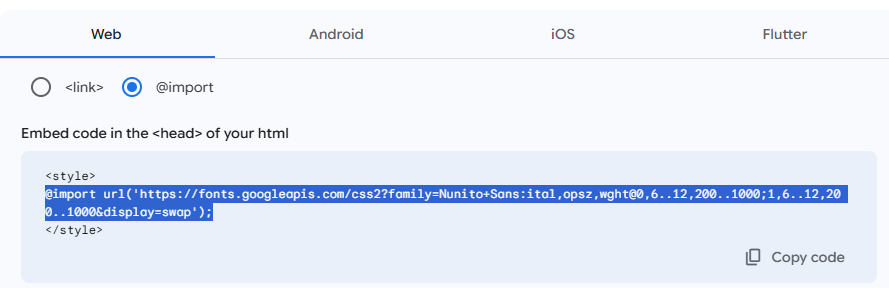
Некоторые атрибуты тега <svg>:

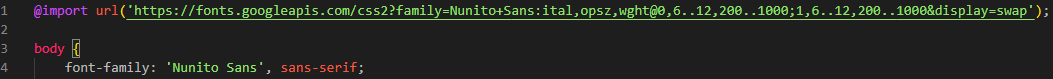
* fill — отвечает за заливку фигуры. Может использоваться как для всего контейнера SVG, так и для отдельных частей.
* <svg>
* <style>
* @media (prefers-color-scheme: dark) {
* path {
* fill: white;
* }
* }
* </style>
* </svg>

5. Если необходим шрифт из google fonts, находим его, нажимаем get font – get embed code, копируем ссылку и вставляем внутри тега <head></head>



Либо копируем отсюда и вставляем в CSS



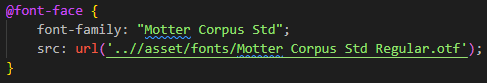


Sans Serif — рублёные шрифты без засечек.

Считаются более современными и могут состоять из букв разной ширины и формы.

1. Если необходимо добавить скаченный шрифт:

Правило **@font-face** в CSS позволяет веб-разработчикам указывать пользовательские шрифты непосредственно в CSS.



Font-family: название шрифта

Srs – ссылка на шрифт

7. Отключает все стандартные отступы в браузере:

\* {

    margin: 0;

    padding: 0;

box-sizing: border-box;

}

{box-sizing: border-box;} Это правило означает, что все элементы в HTML не будут добавлять внутренние отступы и границы к итоговым ширине и высоте.

Убрать подчеркивание ссылок:

a {

    text-decoration: none;

}

Избавиться от лишнего маркера:

li {

    list-style: none;

}

Убрать у кнопки границы (без рамки) и обводку

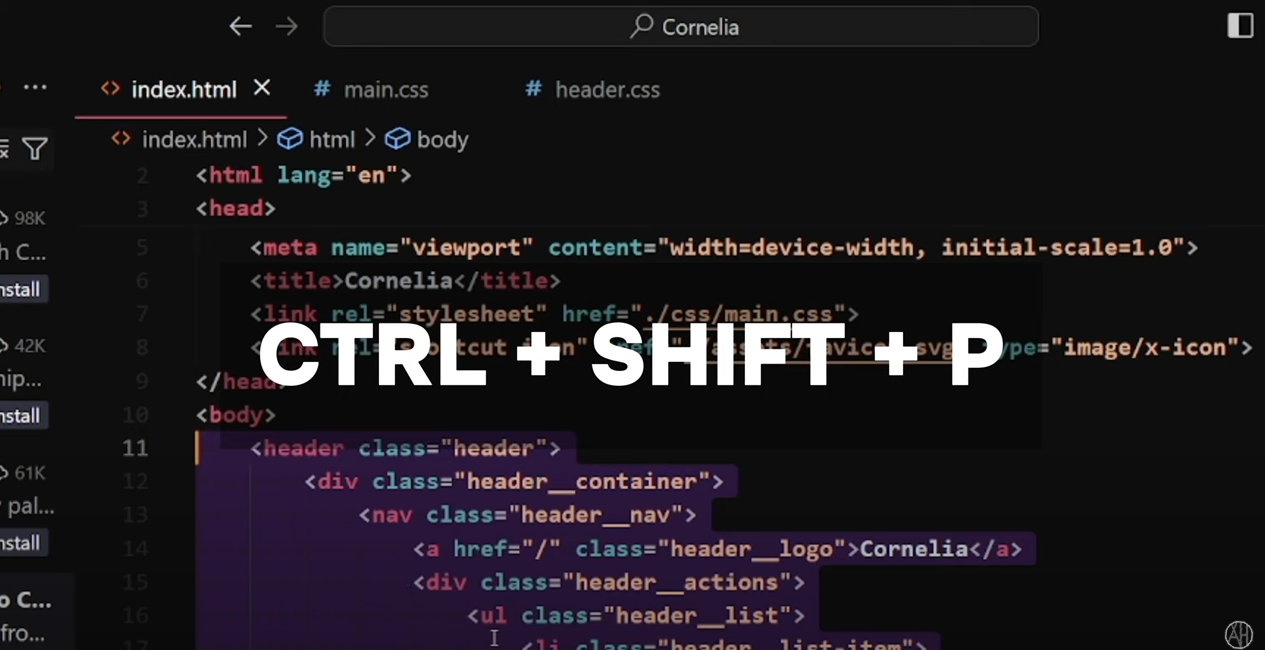
button {

    border: none;

    outline: none;

}

1. Тег <header> в HTML — это контейнер для вводной части сайта или набора навигационных ссылок. Обычно он размещается в верхней части веб-страницы или раздела, к которому принадлежит.
2. **HTML-элемент**<nav> определяет отдельную секцию документа, назначение которой обозначение ссылок навигации (как внутри текущего документа, так и ведущих на другую страницу). В качестве примера такой секции можно привести меню, якорные ссылки.
3. Для переноса тегов в CSS выделяем необходимую часть в html



1. Чтобы любой тег с классом \_container имел одинаковые стили:
2. [class\*='\_container'] {
3. max-width: 88%;
4. margin: 0 auto;
5. }

max-width – максимальная ширина

11. Псевдокласс :root в CSS используется для выбора корневого элемента документа, будь то HTML или SVG. Он часто применяется для определения кастомных свойств (переменных CSS), которые могут быть использованы в дальнейшем в стилях страницы.

Псевдокласс :root соответствует корневому элементу документа. В HTML это элемент <html>, а в SVG — элемент <svg>. Использование :root позволяет применить стили к корневому элементу или задать кастомные свойства, которые могут быть использованы в других селекторах.

:root {

    --primary-green: #0B4141;

}

.header\_logo {

    font-family: "Motter Corpus Std";

    font-size: 26px;

    color: var(--primary-green);

}

12. CSS-свойство gap определяет размер промежутка между строками и столбцами в макете flexbox, grid или с несколькими столбцами.

Это сокращение для свойств row-gap и column-gap.

Свойство gap может принимать два значения: значение строки (то есть пространство между строками элементов) и значение столбца (пространство между столбцами элементов). Если указать только одно значение, оно будет применяться к строкам и столбцам.

13. Чтобы header был всегда сверху страницы:

**position: fixed**— это подвид абсолютного позиционирования, при котором элемент привязывается к координатам окна, а не документа. Когда страницу прокручивают, фиксированный элемент остаётся на своём месте и не прокручивается вместе со страницей.

14. Чтобы header имел прозрачный фон:

background-color: rgba(255, 255, 255, 0.7);

1. Эффект фонового блюра
2. background-filter: blur(18px);
3. z-index: 999;

**z-index в CSS** — свойство, которое определяет порядок наложения элементов в трёхмерном пространстве на веб-странице. Оно управляет тем, как элементы перекрывают друг друга, если они имеют позиционирование, отличное от значения static.