**Селекторы**

|  |
| --- |
| [Нелёгкая жизнь без селекторов](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/1) |
| [Селекторы по тегам](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/2) |
| [Селекторы по классам](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/3) |
| [Отрабатываем селекторы по классам](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/4) |
| [Контекстные селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/5) |
| [Соседние селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/6) |
| [Контекстные и соседние селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/7) |
| [Дочерние селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/8) |
| [Псевдоклассы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/9) |
| [Псевдокласс :nth-child](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/10) |
| [:nth-child и контекстные селекторы](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/11) |
| [Псевдокласс :hover](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/12) |
| [Динамические эффекты с помощью :hover](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/13) |
| [Псевдоклассы :link, :visited и :active](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/14) |
| [Псевдокласс :focus](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/15) |
| [Селекторы атрибутов](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/16) |
| [Селектор по id](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/17) |
| [Испытание. Дуэль.](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/18) |

# Нелёгкая жизнь без селекторов [1/18]

Селекторы позволяют очень точно указывать к каким элементам применять CSS-свойства. Селекторы — это наш снайперский прицел, в этом разделе мы будем учиться им пользоваться.

Так как раздел посвящён «стрельбе свойствами по элементам», то тема биатлона подходит как нельзя кстати. В каждом задании вам нужно будет с помощью селекторов закрывать мишени.

Но для начала представим, что механизма селекторов не существует. Как в этом случае задавать CSS-свойства элементам? С помощью атрибута style. Например:

<p style="color: red;">...</p>

# Селекторы по тегам [2/18]

Задавать атрибут style для каждого тега неудобно и долго. Особенно, если тот же результат можно получить с помощью единственного CSS-правила, в котором используется селектор для тега li.

С помощью селекторов по именам тегов можно задать стили для всех элементов списка, изображений, абзацев и так далее. Эти селекторы содержат имя тега без символов < и >. Например:

li {

/\* стили для элементов списка \*/

}

# Селекторы по классам [3/18]

Класс — это один из атрибутов тегов. Выглядит он вот так:

<li class="first"></li>

Этот атрибут особенный, так как в CSS существует возможность выбирать элементы по классу. Делается это с помощью такого селектора: .имя\_класса. Например:

.first {

/\* стили для класса first \*/

}

Имена классов могут состоять из латинских символов, цифр и знаков - и \_. Имя класса должно начинаться с латинской буквы.

# Отрабатываем селекторы по классам [4/18]

В этом разделе ваша задача — раскрашивать мишени в определённые цвета, применяя подходящие селекторы.

Всего будет три типа выстрелов в мишень, каждому из которых соответствует свой цвет:

1. Мишень закрыта — background-color: white;
2. Выстрел мимо — background-color: red;
3. Ошибка техники (попадание есть, но мишень не закрылась) — background-color: yellow;

Потренируемся использовать классы.

# Контекстные селекторы [5/18]

Селектор может состоять из нескольких частей, разделённых пробелом, например:

p strong { ... }

ul .hit { ... }

.footer .menu a { ... }

Такие селекторы называют контекстными или вложенными. Их используют для того, чтобы применить стили к элементу, только если он вложен в нужный элемент.

Например, селектор .menu a сработает для ссылки a только в том случае, если она расположена внутри элемента с классом .menu.

Читать их проще всего справа налево:

/\* выбрать все теги strong внутри тегов p \*/

p strong { ... }

/\* выбрать все элементы с классом hit внутри тегов ul \*/

ul .hit { ... }

/\* выбрать все ссылки внутри элементов с классом .menu,

которые лежат внутри элементов с классом .footer \*/

.footer .menu a { ... }

Таким образом, можно задавать элементам различные стили в зависимости от их контекста. Если ссылка расположена внутри меню, сделать её крупнее, а если внутри основного текста, то задать ей нужный цвет.

В этом задании вы потренируетесь использовать контекстные селекторы.

# Соседние селекторы [6/18]

Контекстные селекторы используются для вложенных друг в друга элементов, а соседние — для расположенных рядом.

Например, теги <li> в списке являются соседними по отношению друг к другу и вложенными в тег <ul>.

Соседние селекторы записываются с помощью знака +, например, селектор1 + селектор2. Стили применятся к элементу, подходящему под селектор2, только если сразу перед ним расположен элемент, подходящий под селектор1.

Пример. Есть два элемента списка:

<li class="hit"></li>

<li class="miss"></li>

Селектор .hit + .miss применит стили к элементу с классом miss, так как перед ним есть элемент с классом hit.

Селектор .hit + li тоже применит стили к элементу с классом miss, а селектор .miss + .hit не сработает.

# Контекстные и соседние селекторы [7/18]

Селекторы в CSS можно очень гибко комбинировать. В частности, можно комбинировать контекстные и соседние селекторы.

Например, селектор .player-1 .hit + .miss сработает для тега с классом miss, если сразу перед ним расположен тег с классом hit и оба тега расположены внутри тега с классом player-1.

В этом задании вы потренируетесь комбинировать контекстные и соседние селекторы. Класс биатлониста будет задавать контекст, а соседние селекторы будут использоваться для выбора мишени.

# Дочерние селекторы [8/18]

Потомком называются любые элементы, расположенные внутри родительского элемента. А дочерними элементами называются ближайшие потомки. Взгляните на пример:

<ul>

<li><span>...</span></li>

<li><span>...</span></li>

</ul>

По отношению к <ul><li> являются дочерними элементами и потомками, а <span> — потомки, но не дочерние элементы.

Контекстные селекторы влияют на всех потомков, что не всегда удобно. Иногда необходимо задать стили только для дочерних элементов. Особенно это полезно при работе с многоуровневыми списками.

Для этого существует дочерний селектор, в котором используется символ >. Например: ul > li или ul > li > span.

В этом задании вы разберётесь, чем отличаются контекстные и дочерние селекторы.

# Псевдоклассы [9/18]

Псевдоклассы — это дополнения к обычным селекторам, которые делают их ещё точнее и мощнее. Обычный селектор — это снайперский прицел, а с псевдоклассом он становится прибором ночного видения.

Псевдокласс добавляется к селектору c помощью символа :, вот так селектор:псевдокласс. Например:

a:visited { ... }

li:last-child { ... }

.alert:hover { ... }

Знакомство с псевдоклассами мы начнём с first-child и last-child.

Псевдокласс first-child позволяет выбрать первый дочерний элемент родителя, а last-child — последний дочерний элемент. Например:

li:last-child { ... }

Этот селектор выберет последний элемент списка.

# Псевдокласс :nth-child [10/18]

Псевдоклассы из предыдущего примера относятся к семейству псевдоклассов, помогающих выбирать элементы по их расположению.

Вспомним [задание 4](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/4). В нём каждому тегу <li> был задан собственный класс. Используя классы, мы могли выбрать любой из пяти тегов. Если бы тегов было десять, то пришлось бы использовать десять разных классов.

С помощью псевдокласса nth-child можно выбирать теги по порядковому номеру, не используя классы. Синтаксис псевдокласса: селектор:nth-child(выражение). Выражением может быть число или формула. Например:

1. li:nth-child(2) { ... }

2. li:nth-child(4) { ... }

3. li:nth-child(2n) { ... }

Первый селектор выберет второй элемент списка, второй селектор — четвёртый элемент списка, третий селектор — все чётные элементы.

# :nth-child и контекстные селекторы [11/18]

Селекторы с псевдоклассами хорошо сочетаются с контекстными селекторами.

Например, селектор:

.shooter-2 li:nth-child(3) { ... }

Выберет третий тег <li> внутри блока с классом shooter-2.

В этом задании вам нужно будет сделать то же, что и в [задании 5](https://npoed.htmlacademy.ru/htmlcss110/course/8/run/5), но без использования классов.

# Псевдокласс :hover [12/18]

Некоторые псевдоклассы позволяют выбирать элементы, с которыми взаимодействует пользователь. Сначала познакомимся с псевдоклассом :hover.

Этот псевдокласс позволяет выбрать элемент, когда на него наведён курсор мыши и кнопка мыши не нажата. Примеры:

1. a:hover { ... }

2. tr:hover { ... }

3 .menu-item:hover { ... }

Первый селектор выбирает ссылку, второй строку таблицы, третий элемент с классом menu-item, но только в том случае, если на них наведён курсор мыши.

Благодаря этому псевдоклассу можно добавлять в интерфейс динамику и интерактивность, так как элементы начинают реагировать на действия пользователя, изменяя свой внешний вид.

# Динамические эффекты с помощью :hover [13/18]

Интересовались ли вы, как с помощью CSS создаются выпадающие меню?

Львиная доля динамических эффектов, создаваемых с помощью CSS, опираются на несколько псевдоклассов, главный из которых, конечно же, :hover. Весь секрет заключается в сочетании контекстных селекторов и псевдоклассов. Посмотрите на пример:

li.top ul.submenu {

display: none;

}

li.top:hover ul.submenu {

display: block;

}

Первое правило прячет список-подменю. Второе правило гласит: «если на верхний пункт меню, в котором находится подменю, наведут курсор, то надо показать подменю». Вот так всё просто.

Общий принцип такой: родительский элемент реагирует на наведение мыши и изменяет свойства элементов-потомков. То есть всё работает на контекстных селекторах вида селектор1:hover селектор2.

# Псевдоклассы :link, :visited и :active [14/18]

Отвлечёмся от биатлона и познакомимся с псевдоклассами для ссылок.

* :link выбирает ещё не посещённые ссылки.
* :visited выбирает посещённые ссылки.
* :active выбирает активные ссылки (кнопка мыши зажата на ссылке).

Пример задания CSS-правил для ссылок:

a:link { ... }

a:visited { ... }

a:hover { ... }

a:active { ... }

Обратите внимание на порядок правил. Если их расположить по-другому, то некоторые могут не сработать.

# Псевдокласс :focus [15/18]

Псевдокласс :focus позволяет выбрать элемент, который в данный момент в фокусе. Например, текстовое поле, в которое установлен курсор, находится в фокусе.

В фокусе могут быть не только текстовые поля. Если вы переключаетесь между элементами веб-страницы с помощью клавиши **tab**, то в фокус будут попадать ссылки.

Пример использования псевдокласса:

input:focus {

/\* стили для поля в фокусе \*/

}

# Селекторы атрибутов [16/18]

Ранее мы познакомились с атрибутом class и специальными селекторами по классу. Существуют селекторы, которые позволяют выбирать элементы по любым атрибутам.

Чаще всего такие селекторы используются при работе с формами, так как поля форм имеют атрибут type с разными значениями.

Селекторы атрибутов записываются с использованием квадратных скобок: элемент[атрибут]. Примеры селекторов:

1. input[checked] { ... }

2. input[type="text"] { ... }

Первый селектор выберет поля формы, у которых есть атрибут checked, второй селектор выберет поля формы, у которых атрибут type имеет значение text.

# Селектор по id [17/18]

Существует ещё один HTML-атрибут, для которого существует специальный селектор. Этот атрибут id (идентификатор), а селектор записывается с помощью символа #, например, #some-id.

На значение id распространяются те же ограничения, что и на имя класса. Также id должен быть уникальным на странице.

Использование селекторов по id при оформлении считается плохой практикой. Существуют редкие исключения из этого правила, например, при оживлении слайдера на чистом CSS.

# Сетка посложнее, добавляем содержание [23/32]

Сетка готова. Осталось протестировать её с настоящим содержанием.

Мы уже добавили контент в разметку и спрятали его. Вам нужно лишь снова отобразить его, а также убрать вспомогательные границы и фоны, чтобы увидеть готовый результат.

Обратите внимание еще на один подход построения сеток. Он заключается в том, что блокам разметки задаются только размеры и внешние отступы. Внутренние же отступы добавляются дополнительным элементам, вложенным в ячейки сетки.

Такой подход упрощает расчет сеток. Как вы помните, блочная модель устроена таким образом, что общий размер элемента складывается из ширины/высоты, маргинов и паддингов, а также ширины рамок. Поэтому, чтобы получить правильные размеры блоков сетки, всегда нужно помнить об этом. Если же блокам сетки задавать только ширину и внешние отступы, то слагаемых становится намного меньше, а шанс совершить ошибку уменьшается.