Ищем в начале строки: [foo^="bar"]

В первом части про селекторы мы разбирали селектор по атрибутам, когда запись `input[type="text"]` выберет все элементы `input`, у которых атрибут `type` равен `text`.

У этого механизма есть дополнительные возможности: можно выбирать элементы по вхождению подстроки в значение атрибута.

Запись вида `[foo^="bar"]` выберет все элементы, у которых значение атрибута `foo` начинается с подстроки `bar`.

Представьте, что у вас есть три класса для задания колонок разной ширины, например: `column-1`, `column-2` и `column-3`.

У этих классов часть свойств повторяется, а разной является только ширина. Чтобы не дублировать CSS-код, вы можете вынести общие свойства колонок в правило с селектором `[class^="column-"]`, а в остальных правилах задать только ширину:

```
[class^="column-"] {

  /* общие свойства: отступы, рамки, фон и так далее */

}

.column-1 { width: 100px; }

.column-2 { width: 200px; }

.column-3 { width: 300px; }
```

То есть, первый селектор выберет все дивы с классами, начинающимися на `column-`:

```
<div class="column-1"></div>

<div class="column-2"></div>

<div class="column-3"></div>
```

Обратите внимание, что селектор чувствителен к регистру.

Ищем в конце строки: [foo$="bar"]

Селектор вида `[foo$="bar"]` выбирает все элементы, значение атрибута `foo` которых оканчивается строкой `bar`.

Представьте, что у вас на сайте есть раздел с файлами для скачивания в разных форматах и вам нужно для каждого типа файлов добавить свою иконку. Пример разметки:

```
<a href="batman.pdf">Скачать</a>

<a href="superman.doc">Скачать</a>
```

В этом случае вы можете назначать иконки в CSS по расширениям файлов:

```
a[href$=".pdf"] {

  /* иконка для PDF */

}

a[href$=".doc"] {

  /* иконка для DOC */

}
```

И снова, селектор чувствителен к регистру.

Поиск подстроки везде: [foo*="bar"]

Следующий вариант записи `[foo*="bar"]`

Будут выбраны все элементы, у которых значение атрибута `foo` содержит подстроку `bar` на любой позиции

Среди трёх элементов:

```
<p class="person-name"></p>

<div class="some-person-info"></div>

<span class="date-person"></span>
```

селектор `[class*="person"]` выберет все.

Обратите внимание, что селектор чувствителен к регистру.

Поиск слов внутри строки: [foo~="bar"]

Следующая запись: `[foo~="bar"]`.

Такой селектор выберет все элементы, у которых значение атрибута `foo` содержит слово `bar`.

Входить должно именно слово, а не просто подстрока. То есть вхождение `bar` должно содержать с обеих сторон разделители: пробелы или начало/конец строки.

Поиск префиксов: [foo|="bar"]

Селектор по атрибутам вида `[foo|="bar"]`

В данном случае будут выбраны все элементы, у которых значение атрибута `foo` содержит префикс `bar`, то есть либо полностью совпадает с `bar`, либо начинается со строки `bar-` (наличие знака переноса существенно). Другими словами, используя уже знакомые записи селекторов, этот можно заменить на два:
1. `[foo="bar"]` — все элементы, у которых значение атрибута `foo` полностью совпадает со значением `bar`.
2. `[foo^="bar-"]` — все элементы, у которых значение атрибута `foo` начинается со значения `bar-`.

Псевдоклассы :enabled и :disabled

В предыдущих частях мы уже научились работать с формами и разными полями форм. Теперь рассмотрим ряд дополнительных селекторов для работы с этими элементами.

Для обращения к элементам, которые являются доступными на сайте (не заблокированными), можно использовать псевдокласс `:enabled`. Заблокированными считаются элементы форм, у которых установлен атрибут `disabled`. Подробнее об этом атрибуте можно посмотреть в этой части.

Например:

```
input:enabled {

  /* какие-то стили */

}
```

И наоборот, если нужно обратиться только к заблокированным элементам, то для этого есть псевдокласс `:disabled`

Псевдоклассы :read-only и :read-write

Как мы уже рассматривали в предыдущих частях, есть разные способы запретить редактирование пользователем полей. Одним из таковых является установка атрибута `readonly`. Значение в данном случае доступно для чтения и копирования, но недоступно для редактирования.

В зависимости от этого параметра существует два селектора, которые позволяют выбирать каждую группу полей:

Селектор `:read-only` выберет все поля, доступные только для чтения.

Селектор `:read-write` выберет все поля без атрибута `readonly`, даже если у них есть атрибут `disabled`.

Пример записи:

```
input:read-only {}
```

В случае, если браузер не поддерживает такие селекторы, их можно заменить на аналогичные:

```
input[readonly] {} /* аналог :read-only */

input:not([readonly]) {} /* аналог :read-write */
```

Однако, обратите внимание, что `input:not([readonly])` помимо доступных для редактирования текстовых полей выберет кнопки и другие нетекстовые поля `input`,
например, `input[type="submit"]`.

Псевдокласс :required

Мы уже разбирали, что при помощи специального атрибута `required` можно отметить поля, обязательные для заполнения

Используя селектор `:required` можно задать отдельные стили для этих полей

Например

```
input:required {}
```

Псевдокласс :optional

Помимо `:required` существует селектор `:optional`, выполняющий обратное действие.
То есть выберутся все элементы, у которых НЕ указан атрибут `required`

Пример записи

```
select:optional {}
```

Псевдокласс :checked

При помощи селектора `:checked` можно обратиться ко всем
элементам `input` с типами `checkbox` или `radio`, которые являются выбранными (отмеченными)

Например

```
input:checked {}
```

Псевдоклассы :invalid и :valid

При помощи разных типов полей (email, url и др.) или специфических настроек (pattern, min/max и др.) можно указать браузеру, какие именно данные мы ожидаем от пользователя в том или ином поле.

Селектор `:valid` выберет все элементы, у которых введенное значение удовлетворяет требованиям.

А селектор `:invalid` соответственно выберет элементы, у которых введенное значение некорректно.

Пример записи:

```
input:invalid { }
```

Псевдоклассы :in-range и :out-of-range

В части, посвященной формам, мы разбираем специальный тип поля для ввода числовых значений `<input type="number">`. У этого поля можно определить максимальное и минимальное значение при помощи атрибутов `max` и `min` соответственно.

Селектор `:in-range` выбирает все элементы, значение которых попадает в указанный диапазон.

А селектор `:out-of-range` выбирает все элементы, значение которых НЕ попадает в указанный диапазон.

Например:

```
input:in-range {}
```

Объединяй и властвуй

Все эти новые селекторы, как и любые другие селекторы, можно комбинировать между собой, соединять с селекторами другого типа, псевдоклассами и так далее. Все зависит только от сложности задачи и необходимости использовать то или иное решение

Например

```
input[type="checkbox"]:required:checked {}
```

В данном примере будут выбраны все чекбоксы, которые являются обязательными для заполнения и включены по умолчанию.

Чудеса с ~ и :checked

Благодаря селектору `:checked`, с помощью чистого CSS можно создавать очень много интересных эффектов, так как мы можем не просто выбирать отмеченные поля форм, но и влиять с помощью этих полей на другие элементы.

Для этого нам нужно комбинировать `:checked` и `~`, который позволяет выбрать все элементы, идущие за отмеченным полем. Пример:

```
input:checked ~ .item {

  color: red;

}
```

Такое CSS-правило задаст красный цвет, всем элементам с классом `item`, расположенным после отмеченного поля.

Получается, что мы можем с помощью чекбоксов или радиокнопок управлять внешним видом других элементов. С помощью этого приёма, например, делают переключающиеся вкладки, которые работают без JavaScript.

Давайте разберём пример попроще. Добавим переключатели, которые будут показывать определённых котов.