CSS Selectors

Selecting elements

Shadi Lahham - Programmazione web - Frontend - HTML e CSS

CSS Recap

```
body {
 color: yellow;
 background-color: black;
}

#container {
 width: 70%;
 padding: 10px;
}

.byline {
 font-size: 12px;
 border-bottom: 1px solid #ccc;
}
```

Body, #container, .byline are selectors

A few CSS selector types

```
selector:pseudo-class::pseudo-element {
   property: value;
}

selector[attribute] {
   property: value;
}

selector relation {
   property: value;
}
```

Basic selectors

Basic Selectors

```
<ul>
   <li id="winner" class="special">item 1</li>
   <li class="special">item 2</li>
   <li>item 3</li>
</ul>
```

ID selector - most specific

```
#winner
```

Class selector

```
.special
```

Element selector - least specific

```
li
```

What does each one select?

Basic Selectors

```
<section id="intro">
   <h1>...</h1>
   <h2 class="tagline">...</h2>
</section>
```

ID Selector: #intro
Selects an element by the ID attribute value
Is unique and may only be used once per page

Class Selector: .tagline
Selects an element by the class attribute value
May be reused multiple times per page

Element Selector: h1
Selects an element by its type (aka Type Selector, Tag Selector)

ID selector

```
#happy-cake {
  color: crimson;
}

<!-- WILL match -->
<div id="happy-cake"></div>

<!-- WILL match -->
<aside id="happy-cake"></aside>

<!-- Will NOT match -->
<div id="sad-cake">Wrong ID!</div>

<!-- Will NOT match -->
<div class="happy-cake">That's not an ID!</div>
```

Class Selector

```
.module {
  color: olive;
}

<!-- WILL match -->
<div class="module"></div>

<!-- WILL match -->
<aside class="country module iceland"></aside>

<!-- Will NOT match -->
<div class=".module">The dot is for CSS, not HTML</div>

<!-- Will NOT match -->
<div class="bigmodule">Wrong class</div>
```

Element Selector

```
h2 {
  color: DarkGoldenRod;
}

<main>
  <div>
    <!-- WILL match -->
    <h2>Anywhere</h2>
  </div>
</main>

<!-- Will NOT match -->
<div class="h2">Wrong tag, can't trick it</div>

<!-- Will NOT match -->
<h2class="yolo">Make sure that tag has a space after it!</h2>
```

* Selector

```
* {
  color: brown;
}

<!-- WILL match -->
<body>
  <!-- WILL match -->
  <div id="happy-cake"></div>
  <!-- WILL match -->
  <aside id="happy-cake">
    <!-- WILL match -->
    <p class="happy-cake">Cake</p>
  </aside>
</body>
```

`* will match any element! Use with care.`

Relational selectors

Relational selectors (aka Combinators)

A CSS selector can contain more than one basic selector
Between the basic selectors, we use one of these relational selectors

1. **Descendant selector (space)**
   Selects an element that resides anywhere within an identified ancestor element

2. **Direct child selector (>)**
   Selects an element that resides immediately inside an identified parent element

3. **General sibling selector (~)**
   Selects an element that follows anywhere after the prior element
   Both elements share the same parent

4. **Adjacent sibling selector (+)**
   Selects an element that follows directly after the prior element
   Both elements share the same parent

Descendant Selector

```
article h2 {
  color: maroon;
}

<!-- Will NOT match -->
<h2>title</h2>
<article>
  <!-- WILL match -->
  <h2>subtitle</h2>
  <div>
    <!-- WILL match -->
    <h2>subtitle</h2>
  </div>
</article>
```

Direct Child Selector

```css
article > p {
 color: greenyellow;
}
```

```html
<!-- Will NOT match -->
<p>Lorem ipsum dolor sit amet</p>
<article>
  <!-- WILL match -->
  <p>This paragraph will be selected</p>
  <div>
    <!-- Will NOT match -->
    <p>Lorem ipsum dolor sit amet</p>
  </div>
</article>
```

General Sibling Selector

```html
<section>
  <!-- Will NOT match -->
  <p>Lorem ipsum dolor sit amet</p>
  <h2>title</h2>
  <!-- WILL match -->
  <p>This paragraph will be selected</p>
  <div>
    <!-- Will NOT match -->
    <p>Lorem ipsum dolor sit amet</p>
  </div>
  <!-- WILL match -->
  <p>This paragraph will be selected</p>
</section>
<!-- Will NOT match -->
<p>Lorem ipsum dolor sit amet</p>
```

```css
h2 ~ p {
 color: lightblue;
}
```

Adjacent Sibling Selector

```html
<section>
  <!-- Will NOT match -->
  <p>Lorem ipsum dolor sit amet</p>
  <h2>title</h2>
  <!-- WILL match -->
  <p>This paragraph will be selected</p>
  <div>
    <!-- Will NOT match -->
    <p>Lorem ipsum dolor sit amet</p>
  </div>
  <!-- Will NOT match -->
  <p>Lorem ipsum dolor sit amet</p>
</section>
<!-- Will NOT match -->
<p>Lorem ipsum dolor sit amet</p>
```

```css
h2 + p {
 color: mediumorchid;
}
```

Relational selectors - example

```html
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3
    <ul>
      <li>item a</li>
      <li>item b</li>
      <li>item c</li>
    </ul>
  </li>
  <li class="myclass">item 4 (myclass)</li>
  <li>item 5</li>
  <li>item 6</li>
  <li>item 7</li>
</ol>
```

```css
ol li {
 color: green;
}
ol > li {
 color: purple;
}
li {
 color: yellow;
}
li.myclass {
 color: red;
}
li.myclass ~ li {
 color: blue;
}
li.myclass + li {
 color: orange;
}
```

Attribute Selectors

Attribute Selectors

- Select elements based on:
  - whether an attribute is present
  - what its value may contain
- Attribute selectors are supported in all modern browsers
  - Some very old browsers don't support them, but they are no longer in use

Attribute Present Selector

```
a[target] {
 color: orangered;
}

<!-- WILL match -->
<a href="#" target="_blank">click here</a>
```

Attribute Equals Selector

```
a[href="http://google.com/"] {
 color: darkkhaki;
}

<!-- WILL match -->
<a href="http://google.com/">search on google</a>
```

Attribute Contains Selector

```
a[href*="login"] {
 color: salmon;
}

<!-- WILL match -->
<a href="/login.php">login page</a>
```

Attribute Begins With Selector

```
a[href^="https://"] {
 color: navajowhite;
}

<!-- WILL match -->
<a href="https://www.bbc.com/">The BBC</a>
```

Attribute Ends With Selector

```
a[href$=".pdf"] {
 background-image: url("images/pdf.png");
}

<!-- WILL match -->
<a href="/docs/menu.pdf">download menu</a>
<!-- Will NOT match -->
<a href="/audio/song.mp3">download song</a>
```

Attribute Spaced Selector

```
img[alt~='child'] {
 border: 1px solid orange;
}

<!-- WILL match -->
<img src="child.jpg" alt='a small child'>
<!-- Will NOT match -->
<img src="child.jpg" alt='a-small-child'>
<!-- Will NOT match -->
<img src="child.jpg" alt='asmallchild'>
```

Attribute Hyphenated Selector

```
p[lang|="en"] {
 color: dimgray;
}

<!-- WILL match -->
<p lang="en">English</p>
<!-- WILL match -->
<p lang="en-US">American english</p>
<!-- Will NOT match -->
<p lang="fr">Français</p>
```

Attribute Selectors - example

```css
[data-modal="open"] {
 color: peru;
}
```

```html
<div data-modal="open"></div>

<aside class='closed' data-modal='open'></aside>

<div data-modal="false"></div>

<div data-modal></div>

<div data-modal-open></div>
```

Attribute Selectors - example

```
[data-modal="open"] {
 color: peru;
}

<!-- WILL match -->
<div data-modal="open"></div>

<!-- WILL match -->
<aside class='closed' data-modal='open'></aside>

<!-- Will NOT match - Wrong value -->
<div data-modal="false"></div>

<!-- Will NOT match - No value -->
<div data-modal></div>

<!-- Will NOT match - Wrong attribute -->
<div data-modal-open></div>
```

Pseudo-classes

Pseudo-classes

- Similar to regular classes in HTML but not directly stated within the markup
- Are dynamically populated as a result of user actions or document structure
- Recognized by the colon prefix (:)

Example:

```
a:hover {
 color: pink;
}
```

Complete list here:

Pseudo-classes - CSS: Cascading Style Sheets

Link Pseudo-classes

:link
Selects only <a> tags that have an href attribute
Is the same as a[href]

:visited
Selects links that have already been visited by the current browser

```
a:link {...}
a:visited {...}
```

User Action Pseudo-classes

:hover
When the mouse cursor rolls over a link, that link is in it's hover state and this will select it

:active
Selects the link while it is being activated (being clicked on or otherwise activated)

:focus
Selected when a user has made an element the focus of the page (e.g. using 'tab' on the keyboard)
Often used on links, inputs and textareas

```css
a:hover {...}
a:active {...}
a:focus {...}

textarea:focus {
 background: pink;
}
```

User Interface State Pseudo-classes

`:enabled`
`selects an input that is in the default state of enabled and available for use`

`:disabled`
`selects an input that has the disabled attribute tied to it`

`:checked`
`selects checkboxes or radio buttons that are checked`

`:indeterminate`
`Selects a checkbox or radio button that has neither been selected nor unselected`

```
input:enabled {...}
input:disabled {...}
input:checked {...}
input:indeterminate {...}
```

Structural & Position Pseudo-classes

`:first-child`
select an element if it’s the first child within its parent

`:last-child`
select an element if it’s the last element within its parent

`:only-child`
will select an element if it is the only element within a parent

```
li:first-child {...}
li:last-child {...}
div:only-child {...}
```

Structural & Position - example

```html
<ul>
  <li>This list item will be selected</li>
  <li>
    <div>This div will be selected</div>
  </li>
  <li>
    <div>...</div>
    <div>...</div>
  </li>
  <li>This list item will be selected</li>
</ul>
```

```css
li:first-child {...}
li:last-child {...}
div:only-child {...}
```

Structural & Position Pseudo-classes

:first-of-type
select the first element of its type within a parent

:last-of-type
select the last element of its type within a parent

:only-of-type
select an element if it is the only of its type within a parent

```
p:first-of-type {...}
p:last-of-type {...}
img:only-of-type {...}
```

Structural & Position - example

```html
<article>
  <h1>...</h1>
  <p>This paragraph will be selected</p>
  <p>...</p>
  <img src="#"><!-- image will be selected -->
  <p>This paragraph will be selected</p>
  <h6>...</h6>
</article>
```

```css
p:first-of-type {...}
p:last-of-type {...}
img:only-of-type {...}
```

Structural & Position Pseudo-classes

:nth-child()
selects elements based on a simple provided algebraic expression (e.g. "2n" or "4n-1")
Can select even/odd elements, "every third", "the first five", etc

:nth-of-type()
works like :nth-child in places where the elements at the same level are of different types

:nth-last-of-type()
like :nth-of-type but counts up from the bottom instead of the top

:nth-last-child()
like :nth-child but counts up from the bottom instead of the top

Useful :nth-child Recipes

Best way to learn is to experiment. Try the :nth Tester

Structural & Position - example

```html
<ul>
 <li>nope</li>
 <!-- WILL match -->
 <li>yep, I'm #2</li>
 <li>nope</li>
</ul>
```

```css
:nth-child(2) {
 color: purple;
}
```

Empty Pseudo-class

```html
<div>Hello</div><!-- darkorange -->
<div><!-- comment --></div><!-- limegreen -->
<div></div><!-- limegreen -->
<div> </div><!-- darkorange -->
<div><b></b></div><!-- darkorange -->
<div>
</div><!-- darkorange -->
```

```css
div:empty {
 border: 1px solid limegreen;
 padding: 10px;
}

div:not(:empty) {
 border: 1px solid darkorange;
 padding: 10px;
}
```

Negation Pseudo-class

```html
<div>content</div><!-- selected -->
<div class="awesome">content</div>
<section>content</section><!-- selected -->
<p class="awesome">hi</p><!-- selected -->
```

```css
div:not(.awesome) {
 color: plum;
}
:not(div):not(body):not(html) {
 color: royalblue;
}
```

Pseudo-classes

There are many other useful pseudo-classes

The complete list is here:

Pseudo-classes - CSS: Cascading Style Sheets

Pseudo-elements

Pseudo-elements

- Dynamic elements that don’t exist in the document tree
- When used within selectors allow unique parts of the page to be stylized
- Only one pseudo-element may be used within a selector at a given time
- Recognized by the double colon prefix (::)
  - **Note** single colon (:) is also accepted by modern browsers for retro compatibility because very old browsers didn't support double colon (::)

Textual Pseudo-elements

::first-letter
select the first letter of text within an element

::first-line
select identify the first line of text within an element

```
.alpha::first-letter {...}
.bravo::first-line  {...}
```

Textual Pseudo-elements - example

```css
.alpha::first-letter,
.bravo::first-line {
 color: #ff7b29;
 font-size: 18px;
}
```

```html
<p class="alpha">Lorem ipsum dolor...</p>
<p class="bravo">Integer eget enim...</p>
```

Content Pseudo-elements

`::before`
creates a pseudo-element before, or in front of, the selected element

`::after`
creates a pseudo-element after, or behind, the selected element

These pseudo-elements appear nested within the selected element, not outside of it

Example:

```
a::after {
 color: #9799a7;
 content: " (" attr(href) ")";
 font-size: 11px;
}
```

Content Pseudo-elements - example

```
a::after {
 color: #9799a7;
 content: " (" attr(href) ")";
 font-size: 11px;
}

<a href="http://google.com/">Search the Web</a>
<a href="https://www.bbc.com/">The BBC</a>
```

Content Pseudo-elements - example

```
.sale:after {
 background-color: orange;
 color: white;
 content: "Sale!";
 font-size: 12px;
 margin-left: 5px;
 padding: 1px 6px;
}

<ul>
  <li class="sale">item1</li>
  <li>item2</li>
  <li class="sale">item3</li>
  <li>item4</li>
</ul>
```

Fragment Pseudo-element

::selection
identifies part of the document that has been selected, or highlighted, by a user’s actions

::-moz-selection
Mozilla prefixed fragment pseudo-element has been added to ensure the best support for all browser

```
::-moz-selection {
 background: #ff7b29;
}
::selection {
 background: #ff7b29;
}
```

Combining selectors

Combining selectors

Selectors can be combined together

```
.module.news {
 /* Selects elements with BOTH of those classes */
 color: peachpuff;
}
#site-footer::after {
 /* Adds content after an element with that ID */
 color: aliceblue;
}
section[data-open] {
 /* Selects only section elements if they have this attribute */
 color: bisque;
}
```

Specificity

Specificity: How it works

A weight is applied to a CSS declaration, determined by the number of each selector type

1-0-0:
ID selector

0-1-0:
Class selector, Attribute selector, Pseudo-class

0-0-1:
Element Selector, Pseudo-element

0-0-0:
Universal selector (*), combinators (+, >, ~, ' ', ||) and negation pseudo-class :not()
*Note:*The selectors declared inside :not() *contribute* to the weight

Specificity is usually the reason why CSS-rules don't apply to some elements when think they should
Quick reference: https://specifishity.com/

Your turn

Note for exercises

- Each exercise should include
    - An index.html file
    - A style.css file in a folder named /style
    - A readme.md file in which you document the exercise. It should include
        - Author details
        - Exercise requirements
        - Approach to solution
        - Any other information that you feel is necessary to understand your code and your solution
    - For the first 3 exercises you will need code from a zip file
        - The file is named the same as this unit's name with the **.zip** extension
        - Please download and unzip it

1.CSS diner

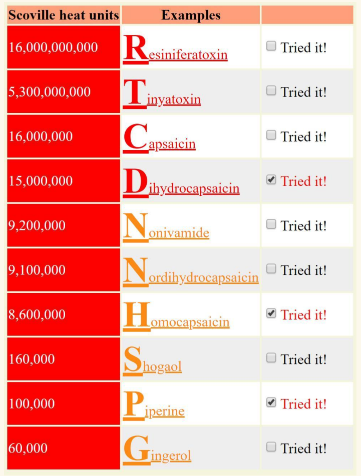
- Open the folder **01-css-diner** from the unit's zip file
- Open the file **index.html**
- Complete all levels of the game
- For each level there can be **many** solutions, try to find as many as you can
- Submit a **css-diner.md** markdown file in which for each level
    - Write all the solutions that you have found
    - For each solution, write the name/type of the all selectors
    - For each solution, explain how the selector works and which HTML tags it targets
- **Note** some of the HTML in this game is not "real"
    - For example the <plate></plate> tags are not valid HTML. This is just a game

2.Selectors Practice

- Open the folder **02-selectors-practice** from the unit's zip file
- Complete the exercise following the instructions in the CSS styles.css file
- Do **not** modify the HTML file

3.Super hot

- Open the folder **03-super-hot** from the unit's zip file
- Complete the exercise following the instructions in the CSS styles.css file
- Do **not** modify the HTML file. Your solution should look like this image

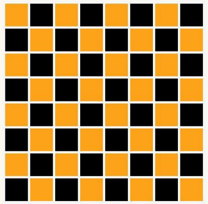


Scoville heat units	Examples	
16,000,000,000	Resiniferatoxin	☐ Tried it!
5,300,000,000	Tinyatoxin	☐ Tried it!
16,000,000	Capsaicin	☐ Tried it!
15,000,000	Dihydrocapsaicin	☑ Tried it!
9,200,000	Nonivamide	☐ Tried it!
9,100,000	Nordihydrocapsaicin	☐ Tried it!
8,600,000	Homocapsaicin	☑ Tried it!
160,000	Shogaol	☐ Tried it!
100,000	Piperine	☑ Tried it!
60,000	Gingerol	☐ Tried it!

4.Chessboard

- Create a <table> with 8 rows and 8 columns
- Your <td> elements should be empty (or contain one letter when needed)
- Use CSS selectors to create a chessboard pattern like in the image below
- Use the following rule to fill the table cells:

```css
td {
  width: 20px;
  height: 20px;
}
```



5.Most specific

For each selector, calculate the specificity and explain it **in detail** in a markdown file:
most-specific.md
Note: Your explanation should be very detailed!

```
ul li {}
ul > li {}
body > #main.mobile a:hover {}
div p > span {}
.users .name {}
[href$='.pdf'] {}
:hover {}
div .name {}
a[href$='.pdf'] {}
.pictures img:hover {}
.name.name.name {}
.user #name {}
#name span {}
nav#nav > li:hover {}
li:nth-child(2n+1):hover {}
```

Bonus

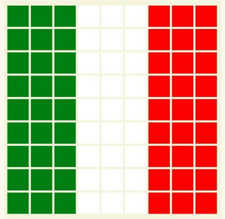
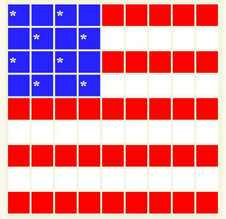
6.We mean business

- Create a table with details of business contacts
  - Columns can include: name, email address, country, etc (max 6 columns)
  - Fill the table with contents
- Add a styles.css file. The HTML file should not contain any inline styles!
- The table header should have a background color
- The table rows should have an alternating white/grey background
  - Row 1 grey, row 2 white, row 3 grey, etc
- When the user hovers over a row with the mouse the color should change

Bonus: modify your CSS so that the table row colors alternate every 2 rows

7.Flag maker

- Same rules as the previous exercise
- Create a <table> with 9 rows and 9 columns for each flag
- Use CSS selectors to create the Italian and US flags like in the images below
- In the same HTML file create as many other flags as you can think of
- Use CSS selectors in creative ways
- **Bonus:** make a flag that changes from one nation to another when the mouse hovers over it

References

CSS Selectors Reference

CSS selectors - CSS: Cascading Style Sheets

Pseudo-classes - CSS: Cascading Style Sheets

References

Specificity

Specificity - CSS: Cascading Style Sheets

Specifics on CSS Specificity

Specifishity :: Specificity with Fish