



Front-end | CSS

Session 3

Some properties

Advanced selectors

Compatibility

Length Units

@media

by Mohammad Amin H.B. Tehrani

www.maktabsharif.ir

Some properties



Some properties

- display
- position
- min-height
- max-height
- overflow
- direction
- font-family
- filter
- transform
- box-shadow
- background: linear-gradient()
- flex
- float
- cursor
- border-radius
- background
- border-left
- border-left-width
- bottom
- border-color
- background-color
- background-image
- background-repeat
- align-items
- align-content
- text-align
- margin
- margin-left
- padding-top

Advanced selectors



Syntax

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)

Combinator selectors

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS

- **descendant selector (space)**
- **child selector (>)**
- **adjacent sibling selector (+)**
- **general sibling selector (~)**

```
div p {  
  background-color: yellow;  
}
```

```
div > p {  
  background-color: yellow;  
}
```

```
div + p {  
  background-color: yellow;  
}
```

```
div ~ p {  
  background-color: yellow;  
}
```

Descendant selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all `<p>` elements inside `<div>` elements:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div p {
      background-color: yellow;
    }
  </style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are
descendants of a specified element</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Descendant selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all `<p>` elements inside `<div>` elements:

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div p {
      background-color: yellow;
    }
  </style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are
descendants of a specified element</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```


Child selector

The child selector selects all elements that are the children of a specified element.

The following example selects all `<p>` elements that are children of a `<div>` element:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div > p {
      background-color: yellow;
    }
  </style>
</head>
<body>
<h2>Child Selector</h2>
<p>The child selector (>) selects all elements that are
the children of a specified element</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section> <!--
not Child but Descendant -->
  <p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```

Child selector

The child selector selects all elements that are the children of a specified element.

The following example selects all `<p>` elements that are children of a `<div>` element:

Child Selector

The child selector (`>`) selects all elements that are the children of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div > p {
      background-color: yellow;
    }
  </style>
</head>
<body>
<h2>Child Selector</h2>
<p>The child selector (>) selects all elements that are
the children of a specified element</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section> <!--
not Child but Descendant -->
  <p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```

Pseudo-classes selectors

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

Hover

The `:hover` selector is used to select elements when you mouse over them.

Tip: The `:hover` selector can be used on all elements, not only on links.

Tip: Use the `:link` selector to style links to unvisited pages, the `:visited` selector to style links to visited pages, and the `:active` selector to style the active link.

Note: `:hover` MUST come after `:link` and `:visited` (if they are present) in the CSS definition, in order to be effective!



Test



Test

```
button.my_btn {  
  box-shadow: 0 3px 5px 0 gray;  
  color: white;  
  background: #3434d4;  
  border: 1px solid gray;  
  font-size: 1rem;  
  padding: .5rem 1.5rem;  
  border-radius: 20px;  
  outline: none;  
}  
  
button.my_btn:hover {  
  box-shadow: none;  
  background: #0d0d5c;  
  color: rgba(255, 255, 255, 0.66);  
}
```

```
<button class="my_btn">Test</button>
```

Focus

The `:focus` selector is used to select the element that has focus.

Tip: The `:focus` selector is allowed on elements that accept keyboard events or other user inputs.

Without focus...

focused!

```
input.my_input {  
  border: 1px solid cadetblue;  
  padding: .4rem .4rem;  
  border-radius: 5px;  
  font-size: 1rem;  
  outline: none;  
  color: #3e3e3e;  
}  
  
input.my input:focus {  
  box-shadow: 0 0 6px 2px cadetblue;  
  color: #0e0e0e;  
}
```

```
<input class="my_input">
```

More pseudo-class Selectors

W3Schools: CSS Pseudo-Classes

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> elements that is the first child of its parent
...		

Pseudo-Elements selectors

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax:

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

Examples:

1. **::first-letter** : [Try it!](#)
2. **::first-line** : [Try it!](#)
3. **::before** : [Try it!](#)
4. **::after** : [Try it!](#)
5. ...

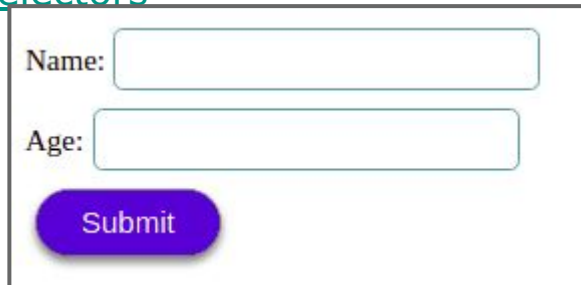
Attribute selectors

It is possible to style HTML elements that have specific attributes or attribute values.

The `[attribute]` selector is used to select elements with a specified attribute.

The following example selects all `<a>` elements with a target attribute:

W3Schools: Attribute Selectors



A form with two text input fields labeled "Name:" and "Age:", and a purple "Submit" button.

```
input[type='submit'] {  
  box-shadow: 0 3px 5px 0 gray;  
  color: white;  
  background: #3434d4;  
  border: 1px solid gray;  
  font-size: 1rem;  
  padding: .5rem 1.5rem;  
  border-radius: 20px;  
  outline: none;  
}  
  
input:not([type='submit']) {  
  border: 1px solid cadetblue;  
  padding: .4rem .4rem;  
  border-radius: 5px;  
  font-size: 1rem;  
  outline: none;  
  color: #3e3e3e;  
}
```


Compatibility



Intro

- **Browser compatibility:** The term browser compatibility refers to the ability of a certain website to appear fully functional on different browsers that are available in the market. This means that the website's HTML coding, as well as the scripts on that website, should be compatible to run on the browsers.
- **Device compatibility (Responsive):** Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device.

Browser compatibility

Solutions:

While preparing the page style properties, you should also pay attention to preparing the specific properties of each browser engine.

Example:

```
-webkit-align-content: ...; /* For Safari, google chrome, Opera ... */  
-moz-animation: ...; /* For mozilla */  
-ms-behavior: ...; /* For Microsoft (Edge) */
```

Also you can check properties [browser support](#)

Device compatibility: Responsive

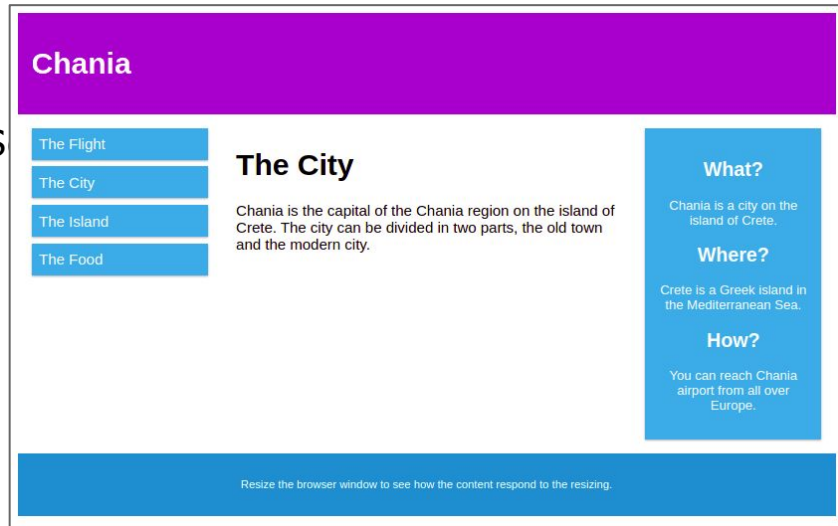
What is Responsive Web Design?

Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Responsive web design is not a program or a JavaS

[Try it!](#)



Viewport

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the `<meta>` tag.

You should include the following `<meta>` viewport element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

Length Units



Intro

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as **width**, **margin**, **padding**, **font-size**, etc.

Length is a number followed by a length unit, such as **10px**, **2em**, etc.

- **Absolute Lengths:** The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.
- **Relative Lengths:** Relative length units specify a length relative to another length property. Relative length units scale better between different rendering medium.

```
h1 {  
  font-size: 60px;  
}  
  
p {  
  font-size: 1cm;  
  line-height: 50%;  
}
```

Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description
cm	Centimeters Try it
mm	Millimeters Try it
in	inches (1in = 96px = 2.54cm) Try it
px *	pixels (1px = 1/96th of 1in) Try it
pt	points (1pt = 1/72 of 1in) Try it
pc	picas (1pc = 12 pt) Try it

Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering medium.

Unit	Description	
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	Try it
ex	Relative to the x-height of the current font (rarely used)	Try it
ch	Relative to the width of the "0" (zero)	Try it
rem	Relative to font-size of the root element	Try it
vw	Relative to 1% of the width of the viewport*	Try it
vh	Relative to 1% of the height of the viewport*	Try it
vmin	Relative to 1% of viewport's* smaller dimension	Try it
vmax	Relative to 1% of viewport's* larger dimension	Try it
%	Relative to the parent element	Try it ²⁵

@media



Intro

The `@media` rule is used in media queries to apply different styles for different media types/devices.

Media queries can be used to check many things, such as

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or port
- resolution

```
@media only screen and (max-width: 600px)
{
  body {
    background-color: lightblue;
  }
}
```

Using media queries are a popular technique for delivering a tailored style sheet (responsive web design) to desktops, laptops, tablets, and mobile phones.

You can also use media queries to specify that certain styles are only for printed documents or for screen readers (mediatype: print, screen, or speech).

In addition to media types, there are also media features. Media features provide more specific details to media queries, by allowing to test for a specific feature of the user agent or display device. For example, you can apply styles to only those screens that are greater, or smaller, than a certain width.

Syntax

```
@media not|only mediatype and (mediafeature and|or|not mediafeature) {  
    CSS-Code;  
}
```

meaning of the not, only and and keywords:

not: The not keyword inverts the meaning of an entire media query.

only: The only keyword prevents older browsers that do not support media queries with media features from applying the specified styles. It has no effect on modern browsers.

and: The and keyword combines a media feature with a media type or other media features.

They are all optional. However, if you use not or only, you must also specify a media type.

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">  
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">  
....
```

Example: Responsive Web Page using

Try it!

Chania

The Flight

The City

The Island

The Food

The City

Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.

What?

Chania is a city on the island of Crete.

Where?

Crete is a Greek island in the Mediterranean Sea.

How?

You can reach Chania airport from all over Europe.

How to

Practice: Registration form

Create a html page like the following sample :

Do it individually



Register user

Enter your profile

Firstname:

Lastname:

Phone:

Gender:

Address:

Example



Pre-reading

- Responsive web page
- Bootstrap
- CSS properties

