HTML5 & CSS3

Presentation by Uplatz

Contact Us: https://training.uplatz.com/

Email: info@uplatz.com

Phone:+44 7836 212635

Uplatz

Table Of Contents:

Changing CSS with JavaScript:

Pure JavaScript:

- It's possible to add, remove or change CSS property values with JavaScript through an element's style property.

var el = document.getElementById("element");

el.style.opacity = 0.5;

el.style.fontFamily = 'sans-serif';

- Note that style properties are named in lower camel case style.
- In the example you see that the css property fontfamily becomes fontFamily in javascript.
- As an alternative to working directly on elements, you can create a <style> or <link> element in JavaScript and append it to the <body> or <head> of the HTML document.

jQuery:

- Modifying CSS properties with jQuery is even simpler.

$('#element').css('margin', '5px');

If you need to change more than one style rule:

```
$('#element').css({
 margin: "5px",
 padding: "10px",
 color: "black"
});
```

- jQuery includes two ways to change css rules that have hyphens in them (i.e. font-size).
- You can put them in quotes or camel-case the style rule name.

```
$('.example-class').css({
 "background-color": "blue",
 fontSize: "10px"
});
```

Styling Lists with CSS:

- There are three different properties for styling list-items: list-style-type, list-style-image, and list-styleposition, which should be declared in that order.
- The default values are disc, outside, and none, respectively.
- Each property can be declared separately, or using the list-style shorthand property.
- list-style-type defines the shape or type of bullet point used for each list-item.

Some of the acceptable values for list-style-type:

- disc
- circle
- square
- decimal
- lower-roman
- upper-roman
- none
- To use square bullet points for each list-item, for example, you would use the following property-value pair:

```
li {
 list-style-type: square;
}
```

- The list-style-image property determines whether the list-item icon is set with an image, and accepts a value of none or a URL that points to an image.

```
li {
 list-style-image: url(images/bullet.png);
}
```

- The list-style-position property defines where to position the list-item marker, and it accepts one of two values:

"inside" or "outside".

```
li {
 list-style-position: inside;
}
```

Structure and Formatting of a CSS Rule:

Property Lists:

- Some properties can take multiple values, collectively known as a property list.

```
/* Two values in this property list */
span {
 text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}
/* Alternate Formatting */
span {
 text-shadow:
 yellow 0 0 3px,
 green 4px 4px 10px;
}
```

Uplatz

Multiple Selectors:

- When you group CSS selectors, you apply the same styles to several different elements without repeating the styles in your style sheet.
- Use a comma to separate multiple grouped selectors.

div, p { color: blue }

- So the blue color applies to all <div> elements and all <p> elements. Without the comma only <p> elements that are a child of a <div> would be red.
- This also applies to all types of selectors.

p, .blue, #first, div span{ color : blue }

This rule applies to:

<p>

elements of the blue class

- element with the ID first
- every <span> inside of a <div>

Rules, Selectors, and Declaration Blocks:

- A CSS rule consists of a selector (e.g. h1) and declaration block ({}).

h1 {}

Comments:

Single Line:

/* This is a CSS comment */

div {

color: red; /* This is a CSS comment */

}

Multiple Line:

```
/* This  is a CSS comment */
div {
 color: red;
}
```

Selectors:

- CSS selectors identify specific HTML elements as targets for CSS styles.
- Selectors use a wide range of over 50 selection methods offered by the CSS language,includingelements, classes, IDs, pseudo-elements and pseudo-classes, and patterns.

Basic selectors:

Selector	Description
*	Universal selector (all elements)

div Tag selector (all <div> elements)

.blue Class selector (all elements with class blue)

.blue.red All elements with class blue and red (a type of Compound selector)

#headline ID selector (the element with "id" attribute set to headline)

:pseudo-class All elements with pseudo-class

::pseudo-element Element that matches pseudo-element

:lang(en) Element that matches :lang declaration, for example <span lang="en">

div > p child selector

- There is no single, integrated CSS4 specification, because it is split into separate modules.

Details:

[attribute]:

- Selects elements with the given attribute.

```
div[data-color] {
 color: red;
}
<div data-color="red">This will be red</div>
<div data-color="green">This will be red</div>
<div data-background="red">This will NOT be red</div>
```

[attribute="value"]:

- Selects elements with the given attribute and value.

```
div[data-color="red"] {
 color: red;
}
<div data-color="red">This will be red</div>
<div data-color="green">This will NOT be red</div>
<div data-color="blue">This will NOT be red</div>
```

[attribute*="value"]:

- Selects elements with the given attribute and value where the given attribute contains the given value anywhere (as a substring).

```
[class*="foo"] {
 color: red;
}
<div class="foo-123">This will be red</div>
<div class="foo123">This will be red</div>
```

```
<div class="bar123foo">This will be red</div>
<div class="barfooo123">This will be red</div>
<div class="barfo0">This will NOT be red</div>
```

[attribute~="value"]:

- Selects elements with the given attribute and value where the given value appears in a whitespace-separated list.

```
[class~="color-red"] {
 color: red;
}
<div class="color-red foo-bar the-div">This will be red</div>
<div class="color-blue foo-bar the-div">This will NOT be red</div>
```

[attribute^="value"]:

- Selects elements with the given attribute and value where the given attribute begins with the value.

```
[class^="foo-"] {
 color: red;
}
<div class="foo-123">This will be red</div>
<div class="foo-234">This will be red</div>
<div class="bar-123">This will NOT be red</div>
```

[attribute$="value"]:

- Selects elements with the given attribute and value where the given attribute ends with the given value.

```
[class$="file"] {
 color: red;
}
```

<div class="foobar-file">This will be red</div>

<div class="foobar-file">This will be red</div>

<div class="foobar-input">This will NOT be red</div>

[attribute|="value"]:

- Selects elements with a given attribute and value where the attribute's value is exactly the given value or is exactly the given value followed by - (U+002D)

[lang|="EN"] {

color: red;

}

<div lang="EN-us">This will be red</div>

<div lang="EN-gb">This will be red</div>

<div lang="PT-pt">This will NOT be red</div>

[attribute="value" i]:

- Selects elements with a given attribute and value where the attribute's value can be represented as Value, VALUE, vAlUe or any other case-insensitive possibility.

```
 [lang="EN" i] {
 color: red;
}
<div lang="EN">This will be red</div>
<div lang="en">This will be red</div>
<div lang="PT">This will NOT be red</div>
```

Specificity of attribute selectors:

0-1-0

Same as class selector and pseudoclass.

***[type=checkbox] // 0-1-0**

- Note that this means an attribute selector can be used to select an element by its ID at a lower level of specificity than if it was selected with an ID selector: [id="my-ID"] targets the same element as #my-ID but with lower specificity.

Combinators:

Overview:

Selector	**Description**
div	span Descendant selector (all <span>s that are descendants of a <div>)
div >	span Child selector (all <span>s that are a direct child of a <div>)
a ~	span General Sibling selector (all <span>s that are siblings after an <a>)

a + span Adjacent Sibling selector (all <span>s that are immediately after an <a>)

Note: Sibling selectors target elements that come after them in the source document.

- CSS, by its nature (it cascades), cannot target previous or parent elements.
- However, using the flex order property, a
- previous sibling selector can be simulated on visual media.

Descendant Combinator: selector selector

- A descendant combinator, represented by at least one space character (), selects elements that are a descendant of the defined element.
- This combinator selects all descendants of the element (from child elements on down).

```
div p {
 color:red;
}
<div>
 <p>My text is red</p>
 <section>
 <p>My text is red</p>
 </section>
</div>
<p>My text is not red</p>
```

Child Combinator: selector > selector:

- The child (>) combinator is used to select elements that are children, or direct descendants, of the specified element.

```
div > p {
 color:red;
}
<div>
 <p>My text is red</p>
 <section>
 <p>My text is not red</p>
 </section>
</div>
```

- The above CSS selects only the first <p> element, as it is the only paragraph directly descended from a <div>.
- The second <p> element is not selected because it is not a direct child of the <div>.

Adjacent Sibling Combinator: selector + selector

- The adjacent sibling (+) combinator selects a sibling element that immediate follows a specified element.

```
p + p {
 color:red;
}
<p>My text is not red</p>
<p>My text is red</p>
<p>My text is red</p>
<hr>
<p>My text is not red</p>
```

General Sibling Combinator: selector ~ selector:

- The general sibling (~) combinator selects all siblings that follow the specified element.

```
p ~ p {
 color:red;
}
<p>My text is not red</p>
<p>My text is red</p>
<hr>
<h1>And now a title</h1>
<p>My text is red</p>
```

Pseudo-classes:

- Pseudo-classes are keywords which allow selection based on information that lies outside of the document tree or that cannot be expressed by other selectors or combinators.
- This information can be associated to a certain state

(state and dynamic pseudo-classes), to locations (structural and target pseudo-classes), to negations of the former (negation pseudo-class) or to languages (lang pseudo-class).

- Examples include whether or not a link has been
- followed (:visited), the mouse is over an element (:hover), a checkbox is checked (:checked), etc.

Syntax

selector:pseudo-class {

property: VALUE;

}

List of pseudo-classes:

Name	**Description**
:active	Applies to any element being activated (i.e. clicked) by the user.
:any	Allows you to build sets of related selectors by creating groups that the

included items will match. This is an alternative to repeating an entire selector.

:target Selects the current active #news element (clicked on a URL containing that anchor name)

:checked Applies to radio, checkbox, or option elements that are checked or toggled into an "on" state.

:default Represents any user interface element that is the default among a group of similar elements.

:disabled Applies to any UI element which is in a disabled state.

:empty Applies to any element which has no children.

:enabled Applies to any UI element which is in an enabled state.

:first Used in conjunction with the @page rule, this selects the first page in a printed document.

:

:first-child Represents any element that is the first child element of its parent.

:first-of-type Applies when an element is the first of the selected element type inside its parent. This may or may not be the first-child.

:focus Applies to any element which has the user's focus. This can be given by the user's keyboard, mouse events, or other forms of input.

:focus-within Can be used to highlight a whole section when one element inside it is focused. It matches any element that the

:focus pseudo-class matches or that has a descendant focused.

:full-screen Applies to any element displayed in full-screen mode. It selects the whole stack of elements and not just the top level element.

.

hover Applies to any element being hovered by the user's pointing device, but not activated.

:indeterminate

- Applies radio or checkbox UI elements which are neither checked nor unchecked, but are in an indeterminate state.
- This can be due to an element's attribute or DOM manipulation.

:in-range

- The :in-range CSS pseudo-class matches when an element has its value attribute inside the specified range limitations for this element.
- It allows the page to give a feedback that the value currently defined using the element is inside the range limits.

- **:invalid** Applies to <input> elements whose values are invalid according to the type specified in the type= attribute.

:lang

- Applies to any element who's wrapping <body> element has a properly designated lang= attribute. For the pseudo-class to be valid, it must contain a valid two or three letter language code.

:last-child Represents any element that is the last child element of its parent.

:last-of-type Applies when an element is the last of the selected element type inside its parent. This may or may not be the last-child.

Class Name Selectors:

- The class name selector select all elements with the targeted class name.

- For example, the class name .warning

would select the following <div> element:

<div class="warning">

 <p>This would be some warning copy.</p>

</div>

- You can also combine class names to target elements more specifically.
- Let's build on the example above to

showcase a more complicated class selection.

CSS

.important {

 color: orange;

}

.

```
warning {
 color: blue;
}
.warning.important {
 color: red;
}
```

HTML

```
<div class="warning">
 <p>This would be some warning copy.</p>
</div>
<div class="important warning">
 <p class="important">This is some really important
warning copy.</p>
</div>
```

- In this example, all elements with the .warning class will have a blue text color, elements with the .important class with have an orange text color, and all elements that have both the .important and .warning class name will have a red text color.
- Notice that within the CSS, the .warning.important declaration did not have any spaces between the two class names.
- This means it will only find elements which contain both class names warning and important in their class attribute.
- Those class names could be in any order on the element.
- If a space was included between the two classes in the CSS declaration

it would only select elements that have parent elements with a .warning class names and child elements with .important class names.

Select element using its ID without the high specificity of the ID selector:

- This trick helps you select an element using the ID as a value for an attribute selector to avoid the high specificity of the ID selector.

HTML:

<div id="element">...</div>

CSS

#element { ... } /* High specificity will override many selectors */

[id="element"] { ... } /* Low specificity, can be overridden easily */

The :last-of-type selector:

- The :last-of-type selects the element that is the last child, of a particular type, of its parent.
- In the example below,the css selects the last paragraph and the last heading h1.

```
p:last-of-type {
 background: #C5CAE9;
}
h1:last-of-type {
 background: #CDDC39;
}
<div class="container">
 <p>First paragraph</p>
 <p>Second paragraph</p>
 <p>Last paragraph</p>
```

```
<h1>Heading 1</h1>
 <h2>First heading 2</h2>
 <h2>Last heading 2</h2>
</div>
```

CSS3 :in-range selector example

```
<style>
input:in-range {
 border: 1px solid blue;
}
</style>
<input type="number" min="10" max="20" value="15">
<p>The border for this value will be blue</p>
```

- The :in-range CSS pseudo-class matches when an element has its value attribute inside the specified rangelimitations for this element.

- It allows the page to give a feedback that the value currently defined using the elementn is inside the range limits.

A. The :not pseudo-class example & B. :focuswithin CSS pseudo-class:

- The following selector matches all <input> elements in an HTML document that are not disabled and don't have the class .example:

HTML:

```
<form>
 Phone: <input type="tel" class="example">
 E-mail: <input type="email" disabled="disabled">
 Password: <input type="password">
</form>
```

CSS:

```
input:not([disabled]):not(.example){
 background-color: #ccc;
}
```

- The :not() pseudo-class will also support comma-separated selectors in Selectors Level 4:

CSS:

```
input:not([disabled], .example){
 background-color: #ccc;
}
```

B. The :focus-within CSS pseudo-class

HTML:

```
 <h3>Background is blue if the input is focused .</p>
 <div>
 <input type="text">
 </div>
```

```
CSS:
div {
 height: 80px;
}
input{
 margin:30px;
```

B. The :focus-within CSS pseudo-class

HTML:

```
 <h3>Background is blue if the input is focused .</p>
 <div>
 <input type="text">
 </div>
```

CSS:

```
div {
 height: 80px; }
```

```
input{
 margin:30px;
}
div:focus-within {
 background-color: #1565C0;
}
```

ID selectors

- ID selectors select DOM elements with the targeted ID. To select an element by a specific ID in CSS, the # prefix is used.

For example, the following HTML div element…

```
<div id="exampleID">
 <p>Example</p>
</div>
```

…can be selected by #exampleID in CSS as shown below:

#exampleID {

 width: 20px;

}

Note: The HTML specs do not allow multiple elements with the same ID

How to style a Range input:

HTML:

<input type="range"></input>

CSS:

Effect **Pseudo Selector**

Thumb input[type=range]::-webkit-slider-thumb, input[type=range]::-moz-range-thumb, input[type=range]::-ms-thumb

Track input[type=range]::-webkit-slider-runnable-track, input[type=range]::-moz-range-track,input[type=range]::-ms-track

OnFocus input[type=range]:focus

Lower part of

the track input[type=range]::-moz-range-progress, input[type=range]::-ms-fill-lower (not possible

in WebKit browsers currently - JS needed)

The :only-child pseudo-class selector example:

The :only-child CSS pseudo-class represents any element which is the only child of its parent.

HTML:

```
<div>
 <p>This paragraph is the only child of the div, it will have the color blue</p>
</div>
<div>
```

```
 <p>This paragraph is one of the two children of the div</p>
 <p>This paragraph is one of the two children of its parent</p>
</div>
CSS:
p:only-child {
 color: blue;
}
```

- The above example selects the <p> element that is the unique child from its parent, in this case a <div>.

Summary:

- Changing CSS with JavaScript
- Structure and Formatting of a CSS Rule
- Comments
- Selectors

Thank You……….

If you have any quries please write to   info@uplatz.com".