Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► CSS Selectors**

**Using selectors**

**Summary**

This guide looks at CSS selectors — the mechanism you use to select which element receives styles — in detail, the different types of selectors available, and how different kinds of selectors have different priorities.

**Information: Selectors**

CSS has its own terminology to describe the CSS language. In a previous tutorial, you created a line in your stylesheet like this:

strong { color: red; }

In CSS terminology, this entire code section is a *rule*. The rule starts with strong, which is a *selector*. It selects which elements in the DOM the rule applies to.

* The part inside the braces is the *declaration*.
* The keyword color is a *property*, and red is a *value*.
* The semicolon after the property-value pair separates it from other property-value pairs in the same declaration.

This guide refers to a selector like strong as a *tag* selector; you will also see it commonly referred to as an *element* selector. The CSS Specification refers to it as a *type* selector.

In addition to tag names, you can use attribute values in selectors. This allows your rules to be more specific. Two attributes, class and id, have special status for CSS.

**Class selectors**

Use the class attribute in an element to assign the element to a named class. It is up to you what name you choose for the class. Multiple elements in a document can have the same class value.

In your style sheet, type a dot (period) before the class name when you use it in a selector.

**ID selectors**

Use the id attribute in an element to assign an ID to the element. It is up to you what name you choose for the ID. The ID name must be unique in the document.

In your stylesheet, type a number sign (hash) before the ID when you use it in a selector.

**Class and ID selector example**

This HTML tag has both a class attribute and an id attribute:

<p class="key" id="principal">

The **id** value principal must be unique in the document, but other tags in the document can have the same key **class** name.

In a CSS style sheet, this rule makes all the elements with class key green. (They might not all be <p> elements.)

.key { color: green; }

This rule makes the one element with the **id** principal bold:

#principal { font-weight: bold; }

If more than one rule applies to an element and specifies the same property, then CSS gives priority to the rule that has the more specific selector. An ID selector is more specific than a class selector, which in turn is more specific than a tag selector.

**Combining selectors**

You can also combine selectors, making a more specific selector.

For example, the selector .key selects all elements that have the class name key. The selector p.key selects only <p> elements that have the class name key.

You are not restricted to the two special attributes, class and id. You can specify other attributes by using brackets. For example, the selector [type='button'] selects all elements that have a type attribute with the value button.

If the style sheet has conflicting rules and they are equally specific, then CSS gives priority to the rule that is later in the style sheet.

When you have a problem with conflicting rules, try to resolve it by making one of the rules more specific, so that it has priority. If you cannot do that, try moving one of the rules nearer to the end of the style sheet so that it has priority.

**Pseudo-class selectors**

A CSS pseudo-class is a keyword added to selectors that specifies a special state of the element to be selected. For example, :hover will apply a style when the user hovers over the element specified by the selector.

Pseudo-classes, together with pseudo-elements, let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (visited, for example), the status of its content (like :checked on some form elements), or the position of the mouse (like :hover, which lets you know if the mouse is over an element or not). For a complete list of selectors, see css/selectors.

selector:pseudo-class { property: value; }

**Information: Selectors based on relationships**

CSS has some ways to select elements based on the relationships between elements. You can use these to make selectors that are more specific.

|  |  |
| --- | --- |
| Common selectors based on relationships | |
| **Selector** | **Selects** |
| A E | Any E element that is a *descendant* of an A element (that is: a child, or a child of a child, *etc*.) |
| A > E | Any E element that is a child of an A element |
| E:first-child | Any E element that is the first child of its parent |
| B + E | Any E element that is the next *sibling* of a B element (that is: the next child of the same parent) |

You can combine these to express complex relationships. You can also use the symbol \* (asterisk) to mean "any element".

**Example of selectors based on relationships**

This HTML table has an id attribute, but its rows and cells do not have individual identifiers:

<table id="data-table-1">

…

<tr>

<td>Prefix</td>

<td>0001</td>

<td>default</td>

</tr>

…

The following rules make the first cell in each row bold, and the second cell in each row monospaced. They only affect one specific table in the document, the one with the id <data-table-1>:

#data-table-1 td:first-child { font-weight: bold; }

#data-table-1 td:first-child + td { font-family: monospace; }

The result looks like:

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Prefix** | 0001 | default | |

**Increasing specificity**

In the usual way, if you make a selector more specific, then you increase its priority.

If you use these techniques, you avoid the need to specify class or id attributes on so many tags in your document. Instead, CSS does the work.

In large designs where speed is important, you can make your style sheets more efficient by avoiding complex rules that depend on relationships between elements.

**Action: Using class and ID selectors**

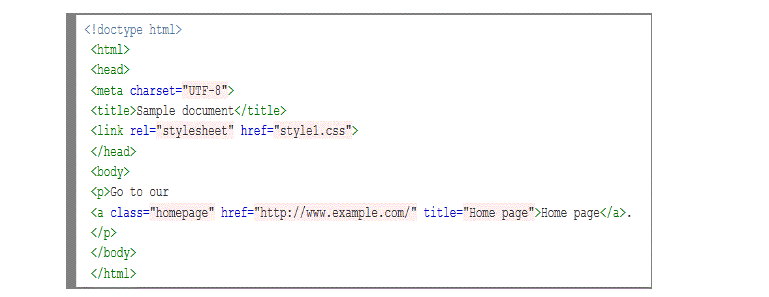
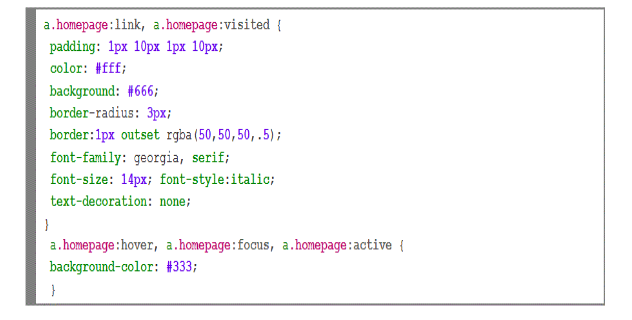


You can try rearranging the lines in your CSS file to show that the order has no effect. The class selectors .carrot and .spinach have priority over the tag selector strong. The ID selector #first has priority over the class and tag selectors.

**Exercise questions**

* Without changing the HTML file, add a single rule to your CSS file that leaves the initial letters the same color as they are now, but changes all the other text in the second paragraph to blue.
* Change the rule you just added (without changing anything else), to set the first paragraph to blue.

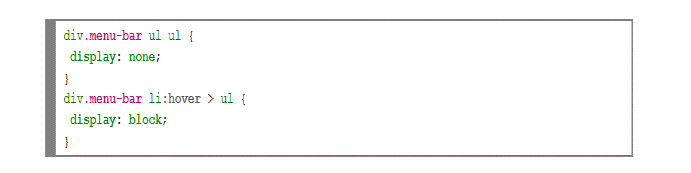
**Action: Using pseudo-classes selectors**

1. Create an HTML page with the following code:
2. 
3. Now edit your CSS file. Replace the entire contents with:
4. 
5. Save the files and refresh your browser to see the result (hover the mouse over the link to see the effect):

The link turns to dark grey.

**Action: Using selectors based on relationships and pseudo-classes**

With selectors based on relationships and pseudo-classes you can create complex cascade algorithms. This is a common technique used, for example, in order to create **pure CSS drop down menus** (that is only CSS, without using JavaScript). The essence of this technique involves creating a rule like this:



to be applied to an HTML structure like the following:



In this way, CSS selectors can be used to create visual effects previously possible only with procedural languages like JavaScript.

**Conclusion**

In this tutorial we examined various types of CSS selectors, how to use them to achieve different styling effects, and explored ways to affect the cascading order (priority) of CSS rules. For more information, see the other CSS tutorials in this section.

**Attribution**

*This article contains content originally from external sources, including ones licensed under the CC-BY-SA license.*

Last modified: Friday, 12 August 2016, 11:46 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=OLBhUtyV0P))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Advanced Selectors**

**Advanced selectors guide**

**Summary**

This guide gives a detailed explanation of most of the advanced CSS selectors available.

**Introduction**

In our Using selectors article, we introduced the most basic of CSS selectors: element, class, and id selectors. With these selectors you can accomplish a lot, but this certainly is not an exhaustive list of selectors — there are other selectors that allow you to select elements to style based on more specific criteria. In this article, most of the advanced CSS selectors are described.

Note: This article discusses *most* of the advanced selectors, because although most modern browsers support all the selectors listed in the CSS3 selectors module, new selectors are being added and modified all the time--keep checking the CSS4 selectors draft for updates. New selectors will be added to this article as they receive more widespread browser support.

You will see many of these selectors used throughout our articles as you read on. Don't worry if you don't understand them all immediately: keep referencing this article as needed.

**Universal selectors**

Universal selectors select every element on a page. For example, the following rule says that every element on the page will display a solid 1 pixel black border:

\* { border: 1px solid #000000; }

**Attribute selectors**

Attribute selectors allow you to select elements based on the attributes they contain. For example, you can select every <img> element with an alt attribute using the following selector:

img[alt] { border: 1px solid #000000; }

Note the square brackets in the example above.

Using the above selector, you could perhaps add a black border around any images that have an alt attribute, and style other images with a bright red border — this is a useful technique for accessibility testing.

**Selecting by attribute value**

Attribute selectors instantly become more useful when you consider that you can select by *attribute value*, not just an attribute's name. The following rule selects all images with an src attribute value of alert.gif:

img[src="alert.gif"] { border: 1px solid #000000; }

Again this style is useful for debugging purposes. You can also use it to style important icons or links without requiring additional classes and IDs.

Note that this code is not supported by IE6 and below.

**Selecting based on substrings within the attribute value**

This is where attribute selectors become even more useful. To begin, you can select and style our <img src="alert.gif"> element using the following rule:

img[src^="alert"] { border: 1px solid #000000; }

The ^ character dictates that this selector affects <img> elements only if they contain the string alert at the start of the src attribute value.

You can also select a <img src="alert.gif"> element using this rule:

img[src$="gif"] { border: 1px solid #000000; }

The $ character dictates that this selector should select <img> elements only if they contain the string gif at the end of the src attribute value. This is really useful for styling links that point to specific types of resources, such as PDF files or word documents.

And finally, you can select a <img src="alert.gif"> element like this:

img[src\*="ert"] { border: 1px solid #000000; }

The \* character dictates that this selector should affect <img> elements only if they have the string ert anywhere within the src attribute value.

Note: These advanced selectors are not supported by IE8 and below.

**Selecting based on delimited items within the attribute value**

If you have an element on a page with a number of classes applied to it, for example:

<article class="featured archive english"></article>

You could select it using any of the following selectors:

article[class~="featured"]

article[class~="archive"]

article[class~="english"]

The ~ character dictates that these selectors should select an <article> element whose class attribute value is a list of whitespace-separated values, but only if one of those values is the value given inside the quotes.

Next, take a look at an element on a page with an ID value in the form of hyphen-separated list:

<article id="english-daily-feature"></article>

You can select it using the following selector:

article[id|="english"]

The | character dictates that this selector should select an <article> element whose id attribute value is a list of hyphen-separated values, but only if the leftmost value is english.

Note: These selectors are not supported by IE8 and below.

**Child selector**

You can use a child selector to select specific elements that are children of other specific elements. For example, the following rule will turn the text of <strong> elements that are children of <h3> elements blue, without affecting other <strong>elements:

h3 > strong { color: blue; }

Note that child selectors are not supported in IE6 or below.

Note also that the > character is often referred to as a combinator in this context — it combines multiple elements together in one selector.

**Descendent selector**

Descendent selectors are similar to child selectors, except that child selectors only select immediate descendants; descendent selectors select matching elements anywhere in the element hierarchy, not just direct descendants. Investigate what this means by examining the following HTML snippet:

<div>

<p>hello</p>

<article>

<p>In this paragraph I will say goodbye</p>

</article>

</div>

In this snippet, the <div> element is the parent of all other elements. It has two children, a <p> and an <article>. The <article> element has a single child element: another <p>.

You could use a child selector to select just the <p> immediately inside the <div>, like so:

div > p { … }

If you instead use a descendent selector, as follows:

div p { … }

Then, both of the <p> elements are selected.

**Adjacent sibling selector**

This selector allows you to select a specific element that comes directly after another specific element, on the same level in the element hierarchy. Review the following example:

<h2>My heading</h2>

<p>My first paragraph</p>

<p>My second paragraph</p>

<p>My third paragraph</p>

<p>My fourth paragraph</p>

<p>My fifth paragraph</p>

If you wanted to select just the <p> element that comes immediately after the <h2> element (and any other such <p> elements that might appear later in the document) you could use the following rule:

h2 + p { ... }

Note that the adjacent sibling selector is not supported in IE6 or below.

Note also that the + character is often referred to as a combinator in this context — it combines multiple elements together in one selector.

**General sibling selectors**

The general sibling selector is very similar to the adjacent sibling selector, except that it allows you to select all siblings of the specified element type, not just the one immediate next to the element on the left side. The CSS syntax looks like this:

h2 ~ p { ... }

Returning to our previous example, this ruleset would select all five paragraph elements, not just the first one. It would also select the paragraph shown below:

<h2>My heading</h2>

<h3>My sub heading</h3>

<p>My paragraph</p>

Note that the general sibling selector is not supported in IE8 or below.

Note also that the ~ character is often referred to as a combinator in this context — it combines multiple elements together in one selector.

**Pseudo-classes**

Pseudo-classes are used to provide styles not for elements, but for various states of elements.

**Link and user action pseudo-classes**

The most common pseudo-classes you will come across are those used to style link states (they are described in full usage in Styling lists and links):

* :link — the normal, default state of links, as displayed when the page first loads in a browser.
* :visited — selects links that you have already visited in the browser you are currently using.
* :focus — selects links that currently have the text indicator (location of cursor) within them.
* :hover — selects links that the cursor is currently hovering over.
* :active — selects links that are currently being clicked.

Note that the last three listed above (called "user action pseudo-classes" — the first two are the link pseudo-classes) can be used to style the states of any element. They are often used when styling a user interface. For example, you can set a different style of a form input field when it is tabbed into, or you might want an information box to appear only when the cursor hovers over a certain area of the screen.

Check out the following examples. The CSS rules below applies a link style to display blue links (which is the default link style in most browsers anyway). When hovered over, the default link underline disappears. To achieve the same effect when the link is focused via the keyboard, the :hover rule is duplicated with the :focus selector. When a link has already been visited, it turns gray. Finally, when a link is active, it is bold, as an extra clue to visitors that something significant is happening.

a:link { color: blue; }

a:visited { color: gray; }

a:hover, a:focus { text-decoration: none; }

a:active { font-weight: bold; }

Take care to specify these rules in the same order as they are shown in above, because otherwise they might not work as you expect. This is due to the way specificity causes later rules in the style sheet to override earlier rules. Specificity is covered in more detail in the article titled Inheritance and cascade.

As another example, the :focus pseudo-class is also useful as a usability aid in forms. For example, you can highlight the input field that has the active blinking cursor inside it with a rule like this:

input:focus {

border: 2px solid black;

background color: lightgray;

}

**The negation (not) pseudo-class**

The negation pseudo-class can be used to explicitly apply styles to elements that **are not** selected by a simple selector. For example, imagine that you want to apply some styling to a number of content blocks, all except one. The blocks could look like this:

<section id="abstract">... </section>

<section id="experiment"> ... </section>

<section id="tests"> ... </section>

<section id="results"> ... </section>

<section id="conclusion"> ... </section>

<section id="references"> ... </section>

You can apply the styling to all sections except the references, with this rule:

#abstract, #experiment, #tests, #results, #conclusion { ... }

Or instead, you can use the negation selector, like this:

section:not(#references) { ... }

Which is much shorter and easy to read.

Note: The negation selector is not supported by IE8 and below.

**The language (lang) pseudo-class**

The :lang pseudo-class selects elements whose languages have been set to the specified language using the lang attribute. For example, the following HTML element:

<p lang="en-US">A paragraph of American text, gee whiz!<p>

Could be selected using the following:

p:lang(en-US) { ... }

**The target pseudo-class**

The target pseudo-class allows you to select an element if it is the **target** of the current page URL. This is really useful and allows for some cool effects, because it effectively allows you to set styles to be applied when links are clicked. For example:

<a href="#target">Click me</a>

<div id="target">Woot!</div>

This is accomplished with a simple link followed by a <div> — the link references the <div> via its ID. The current URL only targets the <div> upon the link being clicked. To style the <div> only when the link is clicked, you could use the following rule:

div:target { ... }

To see a much more involved example of :target usage, read the article titled CSS3 target-based interfaces by Corey Mwamba.

**UI element state pseudo-classes**

These pseudo-classes are all concerned with styling advanced states of UI elements. You'll most commonly use them to style HTML form elements, particularly when some of the new features of HTML5 forms are being used, such as built in validation. (See [HTML5 form additions] for more details.)

Imagine you are styling a basic form input with a valid attribute for validation:

<input type="text" required>

You can style it only when the information entered into it is valid or invalid using these two rules:

input:valid {}

and

input:invalid {}

You can also style it depending on whether it is enabled (default) or disabled (using the disabled attribute), using this style:

input:enabled {}

and this style:

input:disabled {}

Finally, you can style a checkbox only when checked, like this:

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

**Structural pseudo-classes**

Structural pseudo-classes are advanced selectors that enable you to target specific elements based on their position in the document hierarchy. These were introduced in CSS3, and they are built on selectors previously discussed, such as the adjacent sibling selector.

:root selects the root element of the document, which is usually the <html> element. For example, this rule:

html:root{ ... }

:nth-child[[no]] allows you to select a repeating pattern of elements inside an continuous set of like elements, such as several list items, or several paragraphs or articles next to one another. Review this example:

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

<li>Fourth</li>

<li>Fifth</li>

<li>Sixth</li>

<li>Seventh</li>

<li>Eighth</li>

<li>Ninth</li>

<li>Tenth</li>

</ul>

n is set to the pattern we want to select. In this case, to select list items, use this code:

li:nth-child(n)

To select just the odd or even list items, add this rule:

li:nth-child(odd) li:nth-child(even)

Or you can use this rule to accomplish the same result:

li:nth-child(2n+1) li:nth-child(2n+0)

Take a look at some other formula examples:

* li:nth-child(5): select the fifth adjacent list item.
* li:nth-child(4n+1): select every fourth list item, and then add 1 to each result. So numbers 5 and 9.
* li:nth-child(3n-2): select every third list item, and subtract 2 from each result. So numbers 1, 4 and 7.

You can also use nth-last-child[[no]]. This does the same thing as nth-child[[no]], but it counts from the last element in the sequence, not the first element.

nth-of-type[[no]] and nth-last-of-type[[no]] accomplish almost exactly the same goals as nth-child[[no]] and nth-last-child[[no]], but there is one important difference: the former two ignore any rogue elements interspersed with the repeated sequence of like elements because they select by type of element, not by child number. The latter selects by child number.

Here is another example:

<div>

1. <article class="abstract"> ... </article>

2. <article class="abstract"> ... </article>

3. <article class="abstract"> ... </article>

4. <article class="abstract"> ... </article>

5. <article class="abstract"> ... </article>

6. <blockquote><p> ... </p></blockquote>

7. <article class="abstract"> ... </article>

8. <article class="abstract"> ... </article>

9. <article class="abstract"> ... </article>

</div>

In this example we've got a <element> with eight child <article> elements, and a single <blockquote> element in the middle of them: this is child number six. There are a total of nine child elements.

If you use article:nth-child(2n+0) as the selector, to select all the even-numbered children of the <div>, you'd select only the <article>s in positions 2, 4 and 8: the <blockquote> (position number six) is not selected because it is not an <article>.

If you use article:nth-of-type(2n+0) as the selector, you would select the <article>s in positions 2, 4, 7 and 9: this occurs because it selects by the type of element, not the child position. Therefore in this case, the <blockquote> is completely ignored and the even numbered <article>s are selected. Yes, two of them are odd numbered according to my original numbering scheme because in reality the <blockquote> exists and offsets their position, but article:nth-of-type(2n+0) ignores the <blockquote>, effectively counting positions 7 and 9 as 6 and 8.

Next, check out the :first-child and :last-child selectors — these pseudo-classes select only the first or last instance of an element that is the first or last child element of its parent. So, considering the above example again, we could use the following to select - respectively - the first and last <article> element:

article:first-child { ... }

article:last-child { ... }

blockquote:first-child would select nothing, because the first child is not a <blockquote>.

first-of-type and last-of-type again work in a very similar way, but they select based on the type of element, not the position of child it is. So article:first-of-type selects exactly the same <article> element as article:first-child, butblockquote:first-of-type selects the single <blockquote> in the example, because it is the first of its type, even though it is the sixth child.

There are a few others to consider:

* only-child: Selects an element only if it is the only child of its parent. The selector article:only-child wouldn't select anything in our example above, because there is more than one <article> child.
* only-of-type: Selects an element only if it is the only sibling of its type inside the parent element. So blockquote:only-of-type would select the <blockquote> in the above example because it is the only one of its type that exists.
* empty: selects an element only if it has no children whatsoever (including text nodes). For example div:empty would select <div></div>, but not <div>1</div> or <div><p>Hi!</p></div>

**Pseudo-elements**

Pseudo elements differ from pseudo-classes in that they don't select states of elements; they select parts of an element.

**:first-letter**

You can select the first letter inside a given element using the following rule:

p:first-letter {

font-weight: bold;

font-size: 300% background-color: red;

}

The first letter of every paragraph will now be bold, 300% bigger than the rest of the paragraph, and have a red background. This is a good start towards creating a decent drop cap effect.

**:first-line**

To make the first line of every paragraph bold, you can use the following rule:

p:first-line { font-weight: bold; }

**Generated content using :before and :after**

You can use the :before and :after pseudo-elements to specify that content should be inserted before and after the selected element. You then specify the content that you want to insert or generate. For example, you can use the following rule to insert a decorative image after every link on the page:

a:after { content: " " url(flower.gif); }

You can also use the attr() function to insert the values of attributes of the elements after the element. For example, you could insert the target of every link in your document in brackets after each one using the following:

a:after { content: "" "(" attr(href) ")"; }

This is a great technique to use in a print style sheet, where it is important to display the URLs in the document rather than have them hidden inside text links (which are useless on a printed page).

You can also insert incremented numerical values after repeating elements (such as bullets or paragraphs) using the counter() function. This is explained in much more detail in the article titled [Automatic numbering with CSS counters] by David Storey.

These selectors are not supported in IE 7 or below. Also note that you should not insert important information with CSS, because that content will be unavailable to assistive technologies. It will also be unavailable if a visitor chooses not to use the style sheet. The golden rule is that CSS rules are for styling; HTML is for displaying important content.

**CSS3 pseudo-element double colon syntax**

Please note that the new CSS3 way of writing pseudo-elements is to use a double colon, such as a::after { … }, to set them apart from pseudo-classes. You may see this sometimes in CSS. CSS3 however also still allows for single colon pseudo-elements, for the sake of backwards compatibility. For the immediate future, it is recommended to continue using the single colon syntax.

Last modified: Wednesday, 15 July 2015, 10:44 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

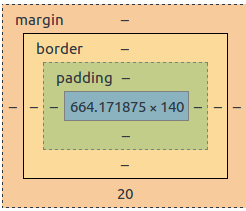
Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Improving the CSS Box Model**

Each HTML element is represented as a rectangular box, it consists of margins, borders, padding, and the actual content. The default CSS box model used to calculate widths and heights of elements, the width/height of an element gives the width/height of the content of the box, excluding padding and border.



Box Model

and they are calculated like this:

Width = width + padding-left + padding-right + border-left + border-right  
Height = height + padding-top + padding-bottom + border-top + border-bottom

so if we take this as an example :

.box {

width: 300px;

padding: 10px;

border: 5px solid #e9573f;

margin: 10px;

}

the width of the rendred box will be calculated like this :

300px (width) + 20px (left + right padding) + 10px (left + right border) + 20px (left + right margin) = 350px

which will give us a box with 350px width, so in this case you have to change the width of the box to 270px, and this is where the box-sizing property comes to the rescue.

box-sizing allows you to switch box models :

* content-box: This is the default style as specified by the W3C. The width and height properties are measured including only the content, but not the border, margin, or padding.
* border-box: The width and height properties include the padding and border, but not the margin. This is the box model used by IE when the document is in Quirks mode.
* padding-box : The width and height properties include the padding size, and do not include the border or margin (only in Firefox).

To save you time, you can apply it using the universal selector

\* {

-moz-box-sizing: border-box;

-webkit-box-sizing: border-box;

box-sizing: border-box;

}

box-sizing is pretty well supported but only Firefox supports the three models, the other browsers only support content-box and border-box, since the padding-box value has been added to the spec very recently.

Copyright © 2015 Adnane Belmadiaf  
This work is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](http://creativecommons.org/licenses/by-sa/3.0/)

Last modified: Wednesday, 24 June 2015, 09:28 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► CSS and Vendor Prefixes**

**CSS3 Compatibility & Vendor Prefixes**

CSS3 includes a handful of properties that allow us to use visual effects on the web that print designers have had access to for ages. For many years we couldn't fully use these properties because of various browser incompatibilities. Browsers have come a long ways since then and we can now use many of these properties.

**Checking Compatibility**

Up until now, most of the CSS you've learned is fully supported in all browsers. For newer properties, from CSS3 and later, that's not always going to be the case.

It's important to check the compatibility of a property to know if you should use it. You should consider the following things:

* Is this feature essential to make my website/app work or is it a visual enhancement
* Which browsers is this feature not supported in
* Can I provide a reasonable fallback option?
* Define a "breaking point" for the feature. How far back should support go?

There are many resources online that collect information on browser compatibilities. Sometimes it's hard to tell how recent the information is. It's important to get up to date information. One of the best resources is Can I Use \_\_\_\_?

* [caniuse.com](http://caniuse.com/)

caniuse.com gives you current information on all relevant browser support. You also can check common known issues and other resources listed for each property. caniuse.com also features information on compatibility for HTML5, JavaScript and other things.

**CSS Vendor Prefixes**

The process of introducing new CSS properties and HTML elements can be a long and convoluted process. Sometimes changes are proposed by standards committees (like the W3C) and other times browser vendors create their own properties.

A property created by the W3C doesn't actually work until browser vendors implement them in new versions of their browsers. Additionally, sometimes there are disagreements in how a standard should be implemented. Other times a browser vendor creates a new property which later becomes a standard, but with a slightly different syntax. And even worse, if end users never upgrade their browsers none of the new features work at all.

Browser vendors needed a way to add support for new features that were not yet standardized, but without messing up later changes or creating incompatibles. To solve this issue *Vendor Prefixes* were created. A vendor prefixes is a special prefix added to a CSS property. Each rendering engine has it's own prefix which will only apply the property to that particular browser.

|  |  |
| --- | --- |
| **Browser** | **Vendor Prefix** |
| Internet Explorer | -ms- |
| Chrome | -webkit- |
| Safari | -webkit- |
| Firefox | -moz- |
| iOS | -webkit- |
| Andriod | -webkit- |
| Opera | -o- |

Note: Both Chrome and Opera now use a forked version of webkit called Blink as their rendering engine. This continuing using the -webkit prefix for now, but in the future will not use prefixes at all and will require 'experimental' features to be turned on via a preference setting. Firefox will be doing a similar thing.

Vendor Prefix Example Vendor Prefixes are best understood by example. In CSS, you can create rounded corners by using the border-radius property. Here it is in it's simplest form:

.rounded { border-radius: 10px; }

All four corners are rounded by 10 pixels. Today, the border-radius is sufficiently supported to make vendor prefixes unnecessary. However, only a few years ago, you would need to use vendor prefixes to ensure support. You would write this:

.rounded {

-moz-border-radius: 10px; /\* support for Firefox \*/

-webkit-border-radius: 10px;

/\* support for webkit based browsers (Chrome, Safari, iOS, etc.) \*/

-o-border-radius: 10px; /\* support for Opera \*/

border-radius: 10px; /\* standardized property \*/

}

/\* Example only. You do not need to use vendor prefixes for

border-radius \*/

* In CSS, a browser simply ignores properties it does not understand.
* Vendor Prefixes are used by their specified browsers and ignored by others.
* Always put the standardized property last. Any browser that understands it will use the last definition, overwriting any previous one.

Last modified: Wednesday, 15 July 2015, 10:48 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

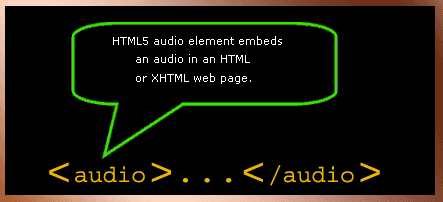
Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► HTML5 Audio**



**Description**

The HTML5 audio element embeds audio in an HTML or XHTML web page. The HTML5 audio element supports .ogg Vorbis, .MP3 and .WAV audio formats.

**Usage**

<audio src="example.ogg" autoplay></audio>

Using the autoplay attribute, you can play the audio automatically.

**What an audio element can contain**

Contains a src attribute or one or more source element, which may be followed by either text or embedded content or text content enclosed by markup.

Any element which can contain text or embeded content or any element which can contain text content enclosed by markup.

**Attributes**

The element supports global attributes (accesskey, class, contenteditable, contextmenu, dir, draggable, dropzone, hidden, id, lang, spellcheck, style. tabindex, title).

**controls**

If specified, the user will be allowed to control audio playback, including volume, seeking, and pause/resume playback by browser.

**loop**

If specified, the audio will automatically be played from the start after reaching the end. This is a boolean attribute.

**preload**

Using this attribute, the author can leave a hint to the browser about loading the associated audio file. Three different values can be used :

none : No preloading is done.

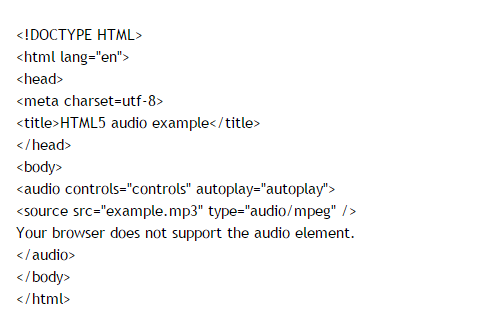
metadata : preload metadata (information about the audio) only.

auto : preload audio automatically.

**src**

The alue of this attribute holds the location of the audio file.

**Example**



**Browser Compatibility**

The above example was tested with the following browser versions:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Feature** | **Chrome** | **Firefox (Gecko)** | **Internet Explorer** | **Opera** | **Safari** |
| Basic support | 12.0.742.122 | 5.0 | 8.0 | 11.50 | 4.0 |

Last modified: Tuesday, 14 July 2015, 03:36 PM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► HTML5 Video**

**Introduction**

In modern browsers, adding a video to your page is as easy as adding an image. No longer do you need to deal with special plug-ins or require crazy markup, you can do it with a single element.

**The Markup**

Let's jump in with a really simple example:

<video src="video.webm" controls> </video>

That's all you need to embed a simple video on your page and show the basic controls so that a user can play, pause or otherwise control the video.

**Specifying Sources**

You can specify multiple source files by using the <source> element. The source element lets you specify multiple formats as a fallback in case the user's browser doesn't support one of them. For example:

<video controls>

<source src="devstories.webm"

type='video/webm;codecs="vp8, vorbis"'/>

<source src="devstories.mp4"

type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"'/>

</video>

When the browser parses the <source> tag, it uses the optional type attribute to help decide which file to download and play. If the browser supports WebM and has the VP8 and Vorbis codecs, it will play devstories.webm, if not, it will check if it can play MPEG-4 videos with the avc1.42E01E and mp4a.40.2 codecs, and so forth.

To improve performance, you should always include the type attribute in the source element. Otherwise the browser will need to load each video file until it can find one that it can play!

It's also a good idea to make sure that your videos are being served with the right MIME type. In some cases, the browser won't play the video if the MIME type isn't set properly.

**Media Fragments**

Adding a media fragment to the media URL, you can specify the exact portion you want to play. To add a media fragment, you simply add #t=[start\_time][,end\_time] to the media URL. For example, to play the video between seconds 10 through 20, you could specify:

<source src="devstories.webm**#t=10,20**"

type='video/webm;codecs="vp8, vorbis"' />

You can also specify the times in hours:minutes:seconds, such as #t=00:01:05 to start the video at one minute, five seconds in. Or, to only play the first minute of the video, you would specify #t=,00:01:00. You need to make sure Range Requests are supported by your server: check for Accept Ranges: bytes. It's on by default for Apache and many other servers, but worth checking.

**Providing captions and subtitles**

The <track> element provides a simple, standardized way to add subtitles, captions, screen reader descriptions and chapters to your video, which improves accessibility but also makes it possible for search engines to understand what's in the video. As well as subtitles and captions, it's possible to put metadata in cues, for example in JSON format. This can enable innovative use cases such as DOM manipulation synchronised with video playback.

<video controls style="width:640px;height:360px;"

poster="poster.png">

<source src="devstories.webm"

type='video/webm;codecs="vp8, vorbis"' />

<source src="devstories.mp4"

type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"' />

**<track src="devstories-en.vtt" label="English subtitles"**

**kind="subtitles" srclang="en" default>**

**</track>**

</video>

The <track> element functions like a <source> element within the <video> element, and has a src attribute that points to a file in WebVTT format. You can specify the label that will be displayed in the UI to the user, as well as the source language (srclang) and if there are multiple track elements, which one should be used as the default.

Here's the first few lines of devstories-en.vtt:

WEBVTT FILE 1 00:00:00.500 -->

00:00:02.000 D:vertical

A:start The Web is always changing 2 00:00:02.500 -->

00:00:04.300 and the way we access it is changing 3 00:00:05.000 -->

00:00:07.000 The source of that change is

<c.highlight>you</c>

**Attributes**

The <video> element has several special attributes that can change or enhance its default behavior.

|  |  |
| --- | --- |
| autoplay\* | Tells the browser to immediately start downloading the video and play it as soon as it can. Note that mobile browsers generally do not support this attribute, the user must tap the screen to begin video playback. |
| preload | Hint to the browser about whether optimistic downloading of the video itself or its metadata is considered worthwhile.  Options are:   * **none** - Hints to the browser that the user likely will not watch the video, or that minimizing unnecessary traffic is desirable. * **metadata** - Hints to the browser that the user is not expected to need the video, but that fetching its metadata (dimensions, first frame, track list, duration, and so on) is desirable. * **auto** - Hints to the browser that optimistically downloading the entire video is considered desirable. |
| poster | Provides an image to show before the video loads |
| controls\* | Shows the default video controls (play, pause, etc) |
| height & width | Sets the width and height of the video in CSS pixels |
| loop\* | Tells the browser to automatically loop the video |
| muted\* | Mutes the audio from the video |

\*indicates a binary attribute, which enables that behavior when the attribute is present, or has it's value set to anything.

**Styling**

Because the <video> element is just another HTML element, you can style it like any other element. You can add borders, set the opacity, apply a filter or even do a 3D transform on the video. For example, by applying filter: grayscale(100%); to the video element, you can turn your video into a black and white video:

***As of the January 2014, the filter effect is only supported in WebKit and Blink based browsers.***

**By** Pete LePage. Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License.

Last modified: Wednesday, 15 July 2015, 10:56 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

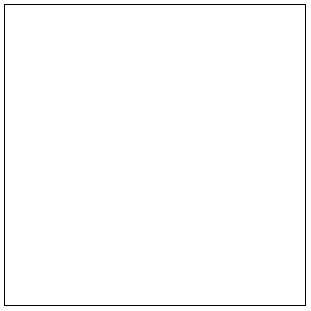
* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► HTML5 Canvas API**

**Introduction**

Canvas is one of the most sought after feature in HTML5. Developers like to use it for creating rich web applications. Users may use those applications without using proprietary browser plug-ins like Adobe's flash player. Most of the modern browsers like Chrome, Firefox, Safari, Opera, IE9 and 10 support it. In a moment we will see what canvas is capable of and how you may use it.

**What's the Canvas Element For?**

Officially a canvas is "a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly". Simply saying, with the help of JavaScript and HTML5 canvas element you may render 2D shapes and bitmap images. The image below shows the canvas with a black border.



If you don't find this much exciting, please read on. We will see how this boring rectangle drawn with canvas may be supercharged to provide you with an awesome graphics.

A webpage may contain multiple canvas elements. Each canvas may have an id using which you may target a specific canvas through JavaScript. Each canvas element has a 2D Context. This again has objects, properties and methods. Tinkers these, you may draw your stuff. To draw on a canvas, you need to reference the context of the canvas. The context gives you access to the 2D properties and methods that We’ll dive deeper into the context later.

Last modified: Wednesday, 15 July 2015, 10:57 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

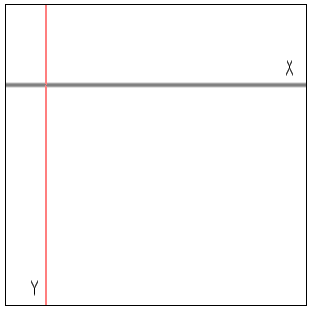
Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► SVG vs. Canvas**

Every canvas element has x and y coordinates. X being the horizontal coordinate and y being the vertical coordinate. The following image shows these coordinates on a canvas.



**Clarifying the SVG – Canvas Relationship**

It’s important to understand the differences between SVG and canvas elements. SVG is an XML based vector graphics format. You can add styles to it with CSS and add dynamic behavior to it using the SVG DOM. Canvas is bitmap based. It allows you to draw graphics and shapes through JavaScript. Like SVG, you may add style and dynamic behavior to it. Here are some reasons to use the canvas over SVG.

* When it comes to draw complex graphics, canvas is faster
* you can save images off the canvas whereas you can’t using SVG
* everything in the canvas is a pixel.

The SVG has some advantages too.

* Being resolutions independent, it can scale for different screen resolutions
* Since it is xml under the hood, targeting different elements is easier
* it’s good at complex animations

So which one to choose over another? to develop a resolution dependent, highly interactive and vector based graphics, choose SVG. If you to render graphics really fast, like in a game, or don’t want to deal with XML, choose the canvas. Actually, they complement each other while delivering real world applications.

**Canvas and Hardware Acceleration**

With Hardware Acceleration enabled browsers, as a developer you will be pleased enough. Because that renders the image/animations with the speed you desire your users should experience. Let's dive a bit deep.

Modern desktops have a GPU (Graphical Processing Unit) along with a CPU(Central Processing Unit). When it comes to delivering fast image/animation, the operation is taken care by GPU, CPU continue server for the rest of the task, resulting in accelerated graphics performance.

Chrome 27, Firefox 22, IE10, And Opera Next supports hardware acceleration and shows significant improve in rendering graphics.

Since JavaScript is the workhorse behind the Canvas, several performance tweaks can be used to user experience by rendering image/animation faster.

Last modified: Wednesday, 15 July 2015, 10:58 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Canvas 2D**

**Canvas 2D API**

The canvas 2D API is an object that allows you to draw and manipulate images and graphics on a canvas element. To reference the context of the canvas, you call getContext, which is a method on the canvas element. It has one parameter, which currently is 2d. Here’s the snippet of code for referencing the context.

Each canvas has its own context, so if you’re page contains multiple canvas elements; you must have a reference to each individual context that you want to work with.

Aside from getContext, there are plenty of other functions (functions of object are called methods in JavaScript) at your disposal in the canvas 2D API. Some of the notable ones are outlined below.

Transformation Functions :

* scale - allows you to scale the current context.
* rotate - allows you to rotate the x and y coordinates of the current context.

State Functions :

* save - allows you to save the current state of the context.
* restore - allows you to restore the state of the context from a previously saved state.

Text Functions

* font - gets or sets the font for the current context.
* fillText - renders filled text to the current canvas.
* measureText - measures the current width of the specified text.

**Working with Canvas**

As we become familiar with the basics of Canvas, we will now explore some of the features Canvas we may use. In this we will focus on basic usage of Canvas, i.e. generating fast, dynamic graphics using JavaScript. But it's worth mentioning that beside that, there lies several other aspects of using Canvas, e.g. working with Text API for Canvas, WebGL - 3D Graphics with Canvas, Full Screen API, Canvas blend modes - Method of defining the effect resulting from overlaying two layers on a Canvas element and CSS Canvas Drawings - Method of using HTML5 Canvas as a background image.

**Setting Up the Canvas**

To set up a canvas for drawing, your must add a <canvas> tag in HTML and assign a 2D drawing context to it. All the drawing operations are performed on the context

**The <canvas> element**

In your HTML, include the following codes that defines the canvas element, giving it a width and height.

<canvas id="myCanvas" height="300" width="400">  
</canvas>

If a width or height is not specified, the default width of 300 pixels and the default height of 150 pixels are used. The canvas is initially empty and transparent.

**The rendering context**

<canvas> creates a fixed-size drawing surface that exposes one or more rendering contexts, which are used to create and manipulate the content shown. We'll focus on the 2D rendering context. Other contexts may provide different types of rendering; for example, WebGL uses a 3D context ("experimental-webgl") based on OpenGL ES.

The canvas is initially blank. To display something, a script first needs to access the rendering context and draw on it. The <canvas> element has a method called getContext(), used to obtain the rendering context and its drawing functions. getContext() takes one parameter, the type of context. For 2D graphics, such as those covered by this tutorial, you specify "2d".

var canvas = document.getElementById('tutorial');

var ctx = canvas.getContext('2d');

The first line retrieves the DOM node for the <canvas> element by calling the document.getElementById() method. Once you have the element node, you can access the drawing context using its getContext() method which returns an object that provides methods and properties for drawing and manipulating images and graphics on a canvas element in a document.

**Checking for support**

The fallback content is displayed in browsers which do not support <canvas>. Through JavaScript you can check for support programmatically by simply testing for the presence of the getContext() method.

Last modified: Wednesday, 15 July 2015, 10:59 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

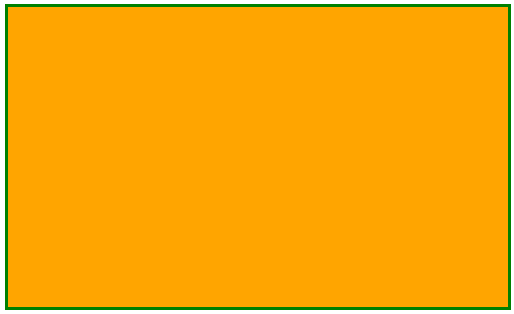
Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Creating a Simple Canvas**

**Create a Simple Canvas:**

<!DOCTYPE html>  
<html>  
<head>  
<title>HTML5 Canvas Demo</title>  
<style>  
#FirstCanvas{  
width: 500px;  
height: 300px;  
border: 3px solid green;  
background-color: orange;  
}  
</style>  
</head>  
<body>  
<canvas id="FirstCanvas"></canvas>  
</body>  
</html>  
  
 

Last modified: Monday, 15 June 2015, 10:15 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

# RESEARCH STYLE GUIDE!!!

# <http://alistapart.com/article/creating-style-guides>

Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Documentation**

**Getting Started With Style Guide Boilerplate**

**Download the Style Guide Boilerplate**

You can clone, fork, or download the repo from GitHub. Once you have the files for **Style Guide Boilerplate**, you'll create a directory on your site for them.

**Set up a directory on your site for the style guide**

I recommend creating a directory named style-guide in your site's root directory. I think it would be awesome if I could go to anysite.com/style-guide/ and check out that site's style guide.

**Upload the Style Guide Boilerplate files**

**Style Guide Boilerplate** is currently PHP based so you will need a server that supports PHP. Just upload the files from the GitHub repo to your newly created directory and your almost done.

**Hook up your own CSS into the style guide**

In the <head> of **Style Guide Boilerplate** are custom styles for the boilerplate itself. These have all been prefixed with sg- so they hopefully shouldn't cause any conflicts with your website's own styles.

Below the custom styles for the boilerplate, you will add in your own custom stylesheet(s) which you use on your live site.

<!-- Style Guide Boilerplate Styles -->

<link rel="stylesheet" href="css/sg-style.css">

<!-- Replace below stylesheet with your own stylesheet -->

<link rel="stylesheet" href="css/style.css">

**Review your live site CSS**

You should be able to go to yoursite.com/style-guide/ and see how your live site's CSS affects base elements. The last step is creating your sites custom patterns/modules.

**Create custom patterns**

To create custom patterns like buttons, breadcrumbs, alert messages, etc., create a new .html file and add your HTML markup into the file.

Save the file as pattern-name.html into the markup/patterns directory inside of your style-guide directory.

You should now be able to see the new patterns at yoursite.com/style-guide/

**Create personalized documentation**

You can use markdown or html to create personalized documentation for your examples. Create a new .md or .html file and name it whatever your markup snippet file is named.

Save the file as markup-name.md or markup-name.html into the doc/base or doc/patterns directory inside of your style-guide directory.

For example, if you want to create doc for markup/patterns/breadcrumbs.html, create a file called breadcrumbs.md or breadcrumbs.html and save it into doc/patterns.

You should now be able to see the new doc at yoursite.com/style-guide/

**Running the app**

You can run the application with PHP's built in web server. Simply run the following command:

php -S localhost:8000

Now, browse to [http://localhost:8000](http://localhost:8000/) to see the website.

**Generating static HTML style guide**

You can generate a static index.html version of style guide boilerplate by running the following command:

php index.php > index.html

**Browser Support**

I've built **Style Guide Boilerplate** with progressive enhancement in mind to work on a wide range of browsers.

Known supported browsers include:

* Chrome
* Firefox
* Safari
* Opera
* IE8+
* Safari for iOS
* Stock Android Browser (4.0+)

Last modified: Tuesday, 18 October 2016, 07:57 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

# Advanced WWW Authoring HTML and CSS

Top of Form

◄ Previous activity

Bottom of Form

Top of Form



Bottom of Form

Top of Form

Next activity ►

Bottom of Form

* [Home](http://www.webmastercert.com/moodle)
* [Courses](http://www.webmastercert.com/moodle/course/index.php)
* [Calendar](http://www.webmastercert.com/moodle/calendar/view.php?view=month)

## You are here

* [**AWS**](http://www.webmastercert.com/moodle/)
* / **►**[**ITNW-1072**](http://www.webmastercert.com/moodle/course/view.php?id=39)
* / **► Documentation**

## Getting Started With Style Guide Boilerplate

### Download the Style Guide Boilerplate

You can clone, fork, or download the repo from GitHub. Once you have the files for **Style Guide Boilerplate**, you'll create a directory on your site for them.

### Set up a directory on your site for the style guide

I recommend creating a directory named style-guide in your site's root directory. I think it would be awesome if I could go to anysite.com/style-guide/ and check out that site's style guide.

### Upload the Style Guide Boilerplate files

**Style Guide Boilerplate** is currently PHP based so you will need a server that supports PHP. Just upload the files from the GitHub repo to your newly created directory and your almost done.

### Hook up your own CSS into the style guide

In the <head> of **Style Guide Boilerplate** are custom styles for the boilerplate itself. These have all been prefixed with sg- so they hopefully shouldn't cause any conflicts with your website's own styles.

Below the custom styles for the boilerplate, you will add in your own custom stylesheet(s) which you use on your live site.

<!-- Style Guide Boilerplate Styles -->

<link rel="stylesheet" href="css/sg-style.css">

<!-- Replace below stylesheet with your own stylesheet -->

<link rel="stylesheet" href="css/style.css">

### Review your live site CSS

You should be able to go to yoursite.com/style-guide/ and see how your live site's CSS affects base elements. The last step is creating your sites custom patterns/modules.

### Create custom patterns

To create custom patterns like buttons, breadcrumbs, alert messages, etc., create a new .html file and add your HTML markup into the file.

Save the file as pattern-name.html into the markup/patterns directory inside of your style-guide directory.

You should now be able to see the new patterns at yoursite.com/style-guide/

### Create personalized documentation

You can use markdown or html to create personalized documentation for your examples. Create a new .md or .html file and name it whatever your markup snippet file is named.

Save the file as markup-name.md or markup-name.html into the doc/base or doc/patterns directory inside of your style-guide directory.

For example, if you want to create doc for markup/patterns/breadcrumbs.html, create a file called breadcrumbs.md or breadcrumbs.html and save it into doc/patterns.

You should now be able to see the new doc at yoursite.com/style-guide/

## Running the app

You can run the application with PHP's built in web server. Simply run the following command:

php -S localhost:8000

Now, browse to [http://localhost:8000](http://localhost:8000/) to see the website.

## Generating static HTML style guide

You can generate a static index.html version of style guide boilerplate by running the following command:

php index.php > index.html

## Browser Support

I've built **Style Guide Boilerplate** with progressive enhancement in mind to work on a wide range of browsers.

Known supported browsers include:

* Chrome
* Firefox
* Safari
* Opera
* IE8+
* Safari for iOS
* Stock Android Browser (4.0+)

Last modified: Tuesday, 18 October 2016, 07:57 AM

You are logged in as [edris safari](http://www.webmastercert.com/moodle/user/view.php?id=4668&course=39) ([Logout](http://www.webmastercert.com/moodle/login/logout.php?sesskey=cOvzlaZmwR))

[ITNW-1072](http://www.webmastercert.com/moodle/course/view.php?id=39)

# 

# 