HTML 5 tags

**HTML5 <article> Tag:**

element represents a section of content that forms an independent part of a document, such as a blog post, article, or other self-contained unit of information, that may be linked to or included in some other content body

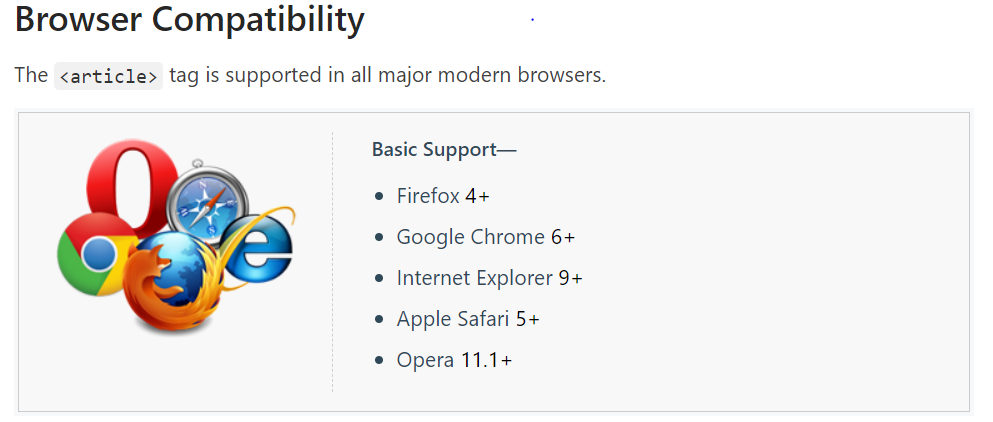
**example:**

<article>

<h1>Introduction to HTML</h1>

<p>HTML is a markup language that is used for creating web pages.</p>

</article>



Similar to this we have aside , section,

**HTML5 <bdi> Tag :**  bi-directional isolation

element represents a span of text that is isolated from other text for the purposes of formatting in a different direction.

This is used while typing code..specially in Arabic etc…

Dir attribute is to specify the direction at display time

<ul>

<li>User <bdi>hrefs</bdi>: 60 points</li>

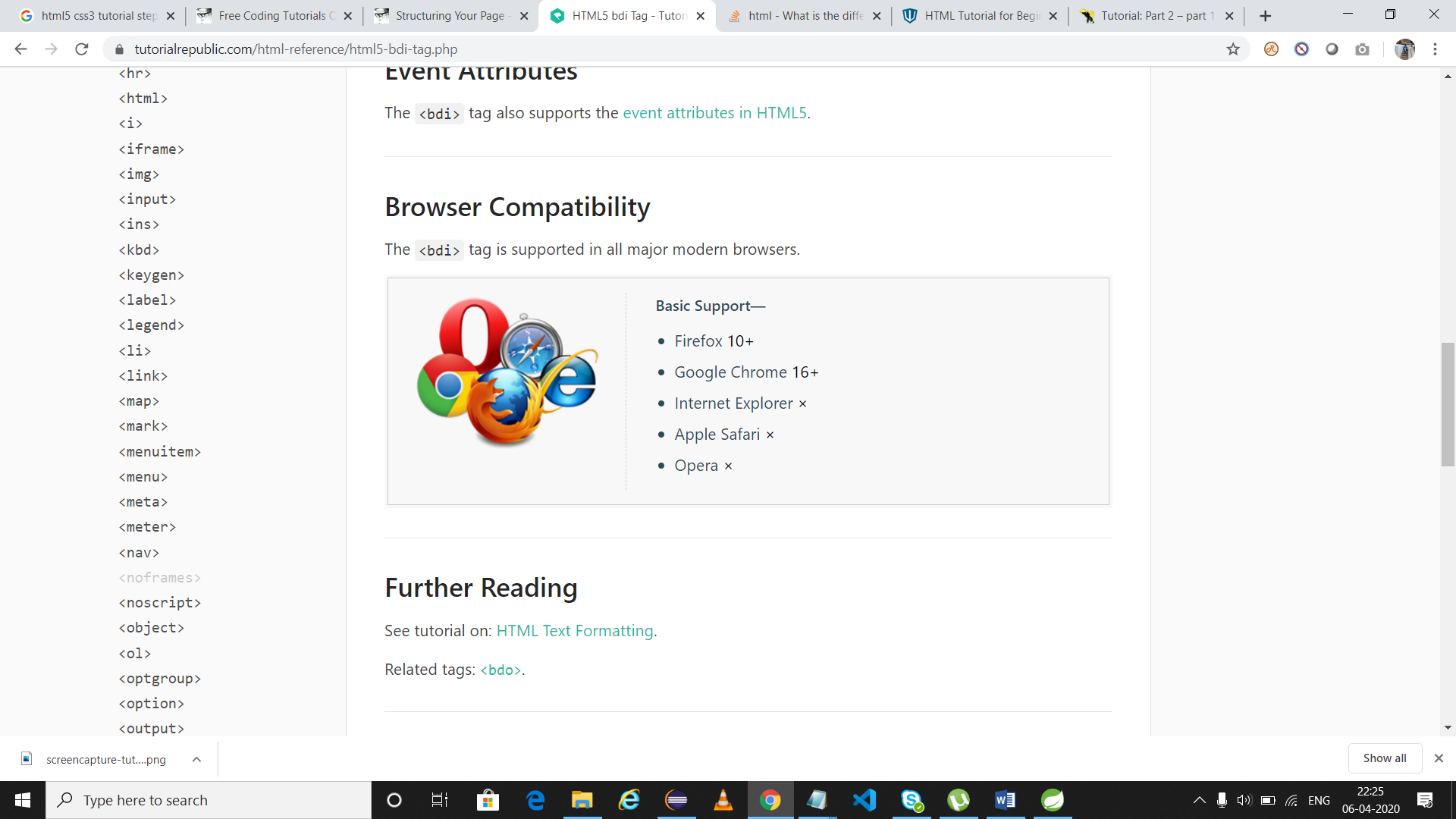
<li>User <bdi>jdoe</bdi>: 80 points</li>

<li>User <bdi>إيان 99</bdi>88 @d 78 points</li>

</ul>

<p dir="ltr">This arabic word <bdi>ARABIC\_PLACEHOLDER</bdi> is automatically displayed right-to-left.</p>

Where dir can be rtl or ltr



<bdo> tag: try

is used to override the current text direction

<blockquote>

tag specifies a section that is quoted from another source.

<blockquote cite=”<http://www.gmail.com>”>sample content</blockquote>

**HTML5 <embed> Tag:**

defines a container for an external application or interactive content (a plug-in).

<embed> tag has not been a part of the HTML 4 specification.

But it is included in html5 version and before html 4

**<embed src="helloworld.swf">**

**Can also have mp4 video**

HTML5 <figcaption> Tag:

 element defines a caption or legend for a figure.

The only permitted parent element for the <figcaption> element is the [<figure>](https://www.tutorialrepublic.com/html-reference/html5-figcaption-tag.php) element. The <figcaption> element must be its first or last child.

mark up a photo in a document,

<figcaption> gives caption for an image

<figure>  
  <img src="logo.png">  
  <figcaption>google logo </figcaption>  
</figure>

<header> , <footer> - tags

HTML5 <hgroup> Tag :

Groups all h*n* tags

The <hgroup> element can be placed anywhere inside the [<body>](https://www.tutorialrepublic.com/html-reference/html-body-tag.php) but it must not be a descendant of an [<address>](https://www.tutorialrepublic.com/html-reference/html-address-tag.php), [<footer>](https://www.tutorialrepublic.com/html-reference/html5-footer-tag.php) or another <header> element

HTML5 <mark> Tag

element defines a marked section of text. You can use this tag if you want to highlight a section of your text for reference purposes.

<p>This is some <mark>highlighted</mark> text.</p>

**HTML <map> Tag :**

tag specifies a client-side image map

<a href=””><img src=””/> </a>

portion of image will act as alink

refer to demo in desktop [(click here)](file:///C:\Users\DELL\Desktop\testimgmap.html)

menu item – try in firefox

**HTML5 <meter> Tag**

element represents a scalar measurement within a known range, or a fractional value. This is also known as a gauge.

<https://www.tutorialrepublic.com/codelab.php?topic=html5&file=meter-tag>

nav – try

The <nav> element is typically a replacement for the <div class="nav">

**Video tag: source tag :track**

<video controls="controls">

<source src="/examples/video/shuttle.mp4" type="video/mp4">

<source src="/examples/video/shuttle.ogv" type="video/ogg">

<track src="subtitles\_en.vtt" kind="subtitles" srclang="en" label="English">

Your browser does not support the HTML5 Video element.

</video>

**Audio tag:**

<audio controls>

<source src="horse.ogg" type="audio/ogg">

<source src="horse.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

**<track> tag:**

element is used to specify supplementary text tracks such as subtitle tracks and caption tracks for [<audio>](https://www.tutorialrepublic.com/html-reference/html5-audio-tag.php) and [<video>](https://www.tutorialrepublic.com/html-reference/html5-video-tag.php) elements.

**HTML5 <output> Tag:**

element represents the result of a calculation.

Typically this element defines a region that will be used to display text output from some calculation that is usually performed by a client-side script (usually JavaScript).

<form>

A: <input type="number" id="a" value="50">

B: <input type="number" id="b" value="100">

Result: <output name="result" value="0"></output>

<input type="button" onclick="add()" value="add">

</form>

<script>

function add(){

var c=parseInt(document.getElementById("a").value)+parseInt(document.getElementById("b").value);

alert('hi'+c);

document.querySelector('output').value=c;

}

</script>

**HTML5 <time> Tag**

element represents a time and/or date.

<p>We open at <time>10:00</time> every morning.</p>

<p>I have a date on <time datetime="2008-02-14 20:00">celeb day</time>.</p>

<time datetime="1914-12-20 08:30:45.687">  <!—year mth date time min secs millisec -->

<time datetime="1914-12">  <!-- means December 1914 -->

<time datetime="1914-W15">  <!-- means week 15 of year 1914 -->

<time datetime="P2D">  <!-- means a duration of 2 days -->  
<time datetime="PT15H10M">  <!-- means a duration of 15 hours and 10 minutes -->

**Progress tag:**

<https://www.tutorialrepublic.com/codelab.php?topic=html5&file=progress-tag>

**HTML5 <canvas> Tag:**

element defines a region in the document, which can be used to draw graphics on the fly via scripting (usually JavaScript).

<https://www.w3schools.com/tags/tag_canvas.asp>

webGL tool:

3D animations with the canvas, we don’t have to get the 3d context,

<https://www.webcodegeeks.com/html5/html5-3d-canvas-tutorial/>

**datalist tag:**

element specifies a set of pre-defined options for an [<input>](https://www.tutorialrepublic.com/html-reference/html-input-tag.php) element. It can be used to provide the quick choices for an input field like an "autocomplete" feature.

Avoids mistakes while typing and gives predefined suggestions

<p>Enter your favorite browser name:</p>

<input type="text" list="browsers">

<datalist id="browsers">

<option value="Firefox">

<option value="Chrome">

<option value="Internet Explorer">

<option value="Opera">

<option value="Safari">

</datalist>

**Rp tag:**

 element is used to provide fall-back parenthesis for browsers that that don't support ruby annotations. Ruby annotations are used for showing pronounciation of East Asian characters, like Chinese or Japanese characters.

<https://www.tutorialrepublic.com/codelab.php?topic=html5&file=rp-tag>

**HTML5 <wbr> Tag:**

(Word Break Opportunity) element specifies a position within text where the browser may optionally break a line if necessary.

**Tip:** When a word is too long, or you are afraid that the browser will break your lines at the wrong place, you can use the <wbr> element to add word break opportunities.

**CSS3:**

* Add layout and design to our pages
* It allows us to share those styles from element to element and page to page.

What is cascade means?

p {

background: orange;

font-size: 24px;

}

p {

background: green;

}

### Cascading Properties:

p {

background: orange;

background: green;

}

**Types of selectors:**

### The type selector has the lowest specificity weight and holds a point value of 0-0-1.

### The class selector has a medium specificity weight and holds a point value of 0-1-0.

### The ID selector has a high specificity weight and holds a point value of 1-0-0.

### #col{

### background:red;

### }

### div{

### background:green;

### }

### .grp{

### background:grey;

### }

### Here id selector takes more precedence,then class and the tagName

## Combining Selectors:

<div class=”news-head">

<p>...</p>

<p>...</p>

<p class="content">...</p>

</div>

## The best practice is to not prefix a class selector with a type selector. Generally we want to select any element with a given class, not just one type of element. And following this best practice, our new combined selector would be better as .news-head .content

#### Absolute Lengths

### Absolute length values are the simplest length values, as they are fixed to a physical measurement, such as inches, centimeters, or millimeters.

### Absolute unit of measurement is known as the pixel and is represented by the px unit notation.

The pixel is equal to 1/96th of an inch; thus there are 96 pixels in an inch. The exact measurement of a pixel, however, may vary slightly between high-density and low-density viewing devices.

With the changing landscape of viewing devices and their varying screen sizes, pixels have lost some of their popularity.

#### Relative Lengths

Relative length values are a little more complicated, as they are not fixed units of measurement; they rely on the length of another measurement.

##### Percentages

.col {

width: 50%;

}

##### Em

The em unit is represented by the em unit notation, and its length is calculated based on an element’s font size.

A single em unit is equivalent to an element’s font size. So, for example, if an element has a font size of 14 pixels and a width set to 5em, the width would equal 70 pixels (14 pixels multiplied by 5).

**The em unit is often used for styling text, including font sizes, as well as spacing around text, including margins and paddings.**

## How Are Elements Displayed?

## Inline-level elements occupy only the width their content requires and line up on the same line, one after the other.

## Block-level elements are generally used for larger pieces of content, such as headings and structural elements.

## Inline-level elements are generally used for smaller pieces of content, such as a few words selected to be bold or italicized.

**Positioning:**

* **Display:**
  + Every element has a default display property value – block
  + few values for the display property are block, inline, inline-block, and none.

<p> more people are affected by this Virus</p>

<p> Content 2 for em</p>

p{ display:block; }

**inline-block:**

**none:**

 using a value of none will completely hide an element and render the page as if that element doesn’t exist. Any elements nested within this element will also be hidden.

## What Is the Box Model?

Every element on a page is a rectangular box and may have width, height, padding, borders, and margins.

* The core of the box is defined by the width and height of an element, which may be determined by the display property, by the contents of the element, or by pecified width and height properties.
* padding and then border expand the dimensions of the box outward from the element’s width and height.
* margin we have specified will follow the border.

div {

border: 6px solid #949599;

height: 100px;

margin: 20px;

padding: 20px;

width: 400px;

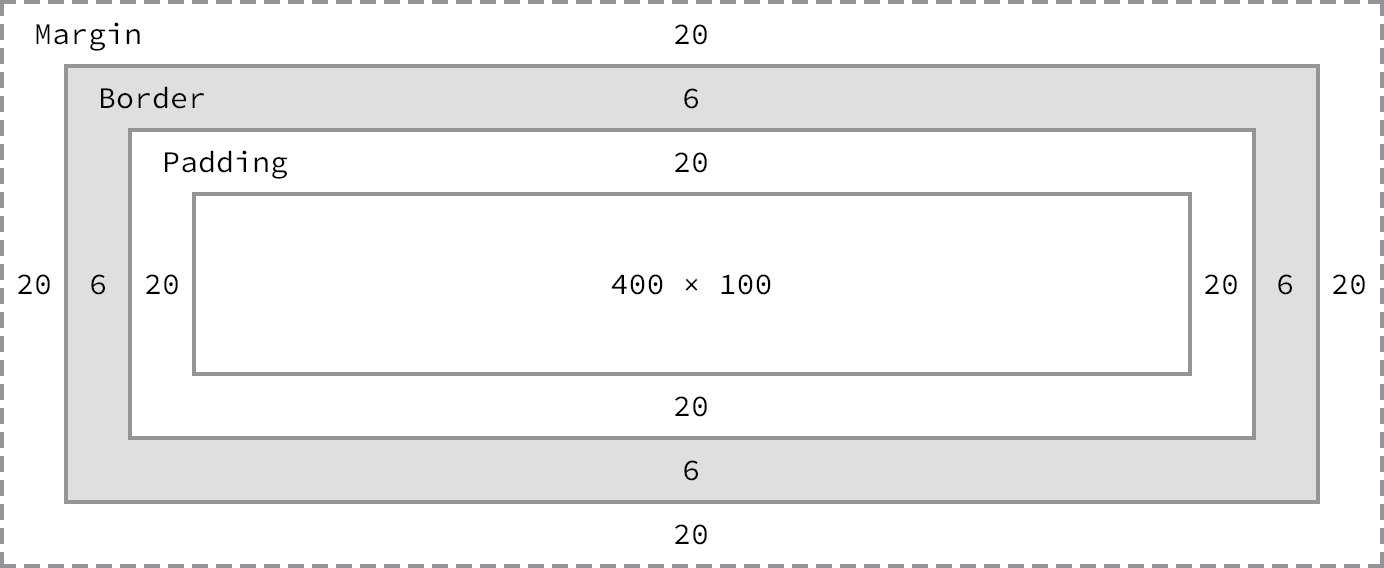
}

According to the box model, the total width of an element can be calculated using the following formula:

|  |  |
| --- | --- |
| 1  2 | margin-right + border-right + padding-right + width + padding-left +  border-left + margin-left |

In comparison, according to the box model, the total height of an element can be calculated using the following formula:

|  |  |
| --- | --- |
| 1  2 | margin-top + border-top + padding-top + height + padding-bottom + border-bottom + margin-bottom |



Using the formulas, we can find the total height and width of our example code.

* **Width:** 492px = 20px + 6px + 20px + 400px + 20px + 6px + 20px
* **Height:** 192px = 20px + 6px + 20px + 100px + 20px + 6px + 20px

The box model is without question one of the more confusing parts of HTML and CSS. We set a width property value of 400 pixels, but the actual width of our element is 492 pixels. By default the box model is additive; thus to determine the actual size of a box we need to take into account padding, borders, and margins for all four sides of the box. Our width not only includes the width property value, but also the size of the left and right padding, left and right borders, and left and right margins.

#### Width

The default width of an element depends on its display value. Block-level elements have a default width of 100%, consuming the entire horizontal space available. Inline and inline-block elements expand and contract horizontally to accommodate their content.

#### Height

The default height of an element is determined by its content.

### Margin & Padding

The default margins and padding for these elements may differ from browser to browser and element to element.

#### Margin

The margin property allows us to set the amount of space that surrounds an element.

Margins can be used to help position elements in a particular place on a page

div {

margin: 20px;

}

This applies for margin-top bottom, left and right

**NOTE:**

**One oddity with the margin property is that vertical margins, top and bottom, are not accepted by inline-level elements. These vertical margins are, however, accepted by block-level and inline-block elements.**

p{

margin-top:40px;

display:inline;

}

<p> hi</p>

<p> hello</p>

#### Padding:

* The padding property is used to provide spacing directly within an element.
* Falls within border
* To set unique values for all four sides of an element, specify those values in the order of top, right, bottom, and left, moving clockwise

div {

margin: 10px 20px 0 15px;

}

div {

margin-top: 10px;

padding-left: 6px;

}

**NOTE:**

**Padding works on all four sides of inline-level elements; however, the vertical padding—the top and bottom—may bleed into the lines above and below an element.**

### Borders

Borders fall between the padding and margin, providing an outline around an element. The border property requires three values: width, style, and color.

Below code is for a 6-pixel-wide, solid, gray border that wraps around all four sides of a <div>:

div {

border: 6px solid #949599;

}

#### Border Radius

div {

border-radius: 5px;

}

**border-radius: 15px 75px; - ellipse, circle border-radius: 50%;**

try - border-top-right-radius

The box model may, however, be changed to support different calculations. CSS3 introduced the box-sizing property, which allows us to change exactly how the box model works and how an element’s size is calculated.

The property accepts three primary values—content-box, padding-box, and border-box—each of which has a slightly different impact on how the box size is calculated.

## Positioning with Floats

* To position elements on a page is with the float property.
* position it to the left or right of its parent element.

#### The float property relies on an element having a display value of block, and may alter an element’s default display value if it is not already displayed as a block-level element.

**Containing float:**

**Relative and static positioning**

Box-shadow

Text-shadow

Text-indent

Text-decoration

text-transform: uppercase;

word-spacing :.25em;

letter-spacing: -.5em;

Font-face

@font-face {

font-family: "Lobster";

src: local("Lobster"), url("lobster.woff") format("woff");

}

body {

font-family: "Lobster", "Comic Sans", cursive;

}

Download that font in to local system , where html file stays

#### Keyword Colors

Keyword color values are names (such as red, green, or blue) that map to a given color. These keyword names and their corresponding colors are determined by the CSS specification. Most common colors, along with a few oddities, have keyword names.

A complete list of these keyword names can be found within the [CSS specification](https://www.w3.org/TR/css3-color/).

| **Color** | **Name** | **Hex Values** | **RGB Values** | **HSL Values** |
| --- | --- | --- | --- | --- |
|  | Black | #000000 | rgb(0, 0, 0) | hsl(0, 0%, 0%) |
|  | Silver | #c0c0c0 | rgb(192, 192, 192) | hsl(0, 0%, 75%) |
|  | Gray | #808080 | rgb(128, 128, 128) | hsl(0, 0%, 50%) |
|  | White | #ffffff | rgb(255, 255, 255) | hsl(0, 100%, 100%) |
|  | Maroon | #800000 | rgb(128, 0, 0) | hsl(0, 100%, 25%) |
|  | Red | #ff0000 | rgb(255, 0, 0) | hsl(0, 100%, 50%) |
|  | Purple | #800080 | rgb(128, 0, 128) | hsl(300, 100%, 25%) |
|  | Fuchsia | #ff00ff | rgb(255, 0, 255) | hsl(300, 100%, 50%) |
|  | Green | #008000 | rgb(0, 128, 0) | hsl(120, 100%, 25%) |
|  | Olive | #808000 | rgb(128, 128, 0) | hsl(60, 100%, 25%) |
|  | Lime | #00ff00 | rgb(0, 255, 0) | hsl(120, 100%, 50%) |
|  | Yellow | #ffff00 | rgb(255, 255, 0) | hsl(60, 100%, 50%) |
|  | Navy | #000080 | rgb(0, 0, 128) | hsl(240, 100%, 25%) |
|  | Blue | #0000ff | rgb(0, 0, 255) | hsl(240, 100%, 50%) |
|  | Teal | #008080 | rgb(0, 128, 128) | hsl(180, 100%, 25%) |
|  | Aqua | #00ffff | rgb(0, 255, 255) | hsl(180, 100%, 50%) |

**Things to know for a developer:**

## 1. Absolute positioning

If you want control over where an element lives on our website at all times, absolute positioning is the key to making this happen. If you think of your browser as one big bounding box, absolute positioning allows you to control exactly where in that box an element will stay. Use top, right, bottom and left, accompanied by a pixel value to control where an element stays.

position:absolute;

top:20px;

right:20px

The CSS above sets the position of an element to stay 20px from the top and right edges of your browser. You can also use absolute positioning inside of a div.

2. \* + selector

The \* enables you to select all elements of a particular selector. For example, if you used \*p and then added CSS styles to that, it would do it to all elements in your document with a <p> tag. This makes it easy to target parts of your website globally.

## 3. Overriding all styles

This should be used sparingly, because if you do this for everything, you’re going to find yourself in trouble in the long run. However, if you want to override another CSS style for a specific element, use !important after the style in your css. For example, if I wanted the H2 headers in a specific section of my site to be red instead of blue, I would use the following CSS:

.section h2 { color:red **!important**; }

## 4. Centering

Centering is tricky, because it depends on what you’re trying to center. Let’s take a look at the CSS of items to be centered, based on content.

### TEXT

Text is centered using the text-align:center;. If you want it to either side, use left or right instead of center.

### CONTENT

A div (or any other element) can be centered by adding the block property to it, and then using auto margins. The CSS would look like this:

#div1 {

display: block;

margin: auto;

width: anything under 100%

}

The reason I put “anything under 100%” for width is because if it was 100% wide, then if would be full-width and wouldn’t need centering. It is best to have a fixed width, like 60% or 550px, etc.

## 5. Vertical alignment (for one line of text)

You will use this in a CSS navigation menu, I can almost guarantee that. The key is to make the height of the menu and the line-height of the text the same. I see this technique a lot when I go back and edit existing websites for clients. Here’s an example:

.nav li{

line-height:50px;

height:50px;

}

## 6. Hover effects

This is used for buttons, text links, bock sections of your site, icons, and more. If you want something to change colors when someone hovers their mouse over it, use the same CSS, but add :hover to it and change the styling. Here’s an example:

.entry h2{

font-size:36px;

color:#000;

font-weight:800;

}

.entry h2:hover{

color:#f00;

}

What this does is it changes the color of your h2 tag from black to red when someone hovers over it. The great thing about using :hover is that you don’t have to declare the font-size or weight again, if it isn’t changing. It only changes what you specify.

### TRANSITION

For hover effects, like with menus or on images in your website, you don’t want colors snapping too quickly to the end result. You ideally want to ease the change in gradually, which is where the transition property comes into play.

.entry h2:hover{

color:#f00;

transition: all 0.3s ease;

}

This makes the change happen over .3 seconds, instead of just instantly snapping to red. This makes the hover effect more pleasing to the eye and less jarring.

## 7. Link states

These styles are missed by a lot of designers, and it really causes usability issues with your visitors. The :link pseudo-class controls all links that haven’t been clicked on yet. The :visited pseudo class handles the styling of all of the links you’ve already visited. This tells website visitors where they have already been on your site, and where they have yet to explore.

a:link { color: blue; }

a:visited { color: purple; }

## 8. Easily resize images to fit

Sometimes you get in a pinch where images need to fit a certain width, while scaling proportionally. An easy way to do this is to use max width to handle this. Here is an example:

img {

max-width:100%;

height:auto;

}

This means that the largest the image could ever be is 100%, and the height is automatically calculated, based on the image width. In some cases, you might have to also have to specify the width at 100%.

## 9. Control the elements of a section

Using the image example above, if you only want to target the images of a certain section, like your blog, use a class for the blog section, and combine it with the actual selector. This will enable you to select only the images of the blog section, and not other images, such as your logo, or social meia icons, or images in any other sections of your site, like the sidebar. Here’s how the CSS would look:

.blog img{

max-width:100%;

height:auto;

}

## 10. Direct children

I wish I’d known this when I first started out using CSS. This would have saved me so much time! Use > to select the direct children of an element. For example:

#footer > a

This will select and style all of the active link elements that are immediately under the Footer ID. It won’t select anything past the active element, or anything else contained in the footer, like plain text. This works great with top level navigation elements, too.

### SPECIFIC CHILD ELEMENTS

Believe me, this is handy when you are styling lists. You just need to count how many items down the element is that you want to style and then apply that style.

li:nth-child(2) {

font-weight:800;

color: blue;

text-style:underline;

}

The CSS above targets the second item in the list and makes it bold, underlined, and blue. Add an “n” after the number in parenthesis and you can target every 2nd list item. Imagine being able to style every other line in a table-style layout for easy reading. The CSS would be:

li:nth-child(2)

## 11. Apply CSS to multiple classes, or selectors

Let’s say you wanted to add an identical border around all images, the blog section and the sidebar. You don’t have to write out the same exact CSS 3 times. Just list those items out, separated by commas. Here is an example:

.blog, img, .sidebar {

border: 1px solid #000;

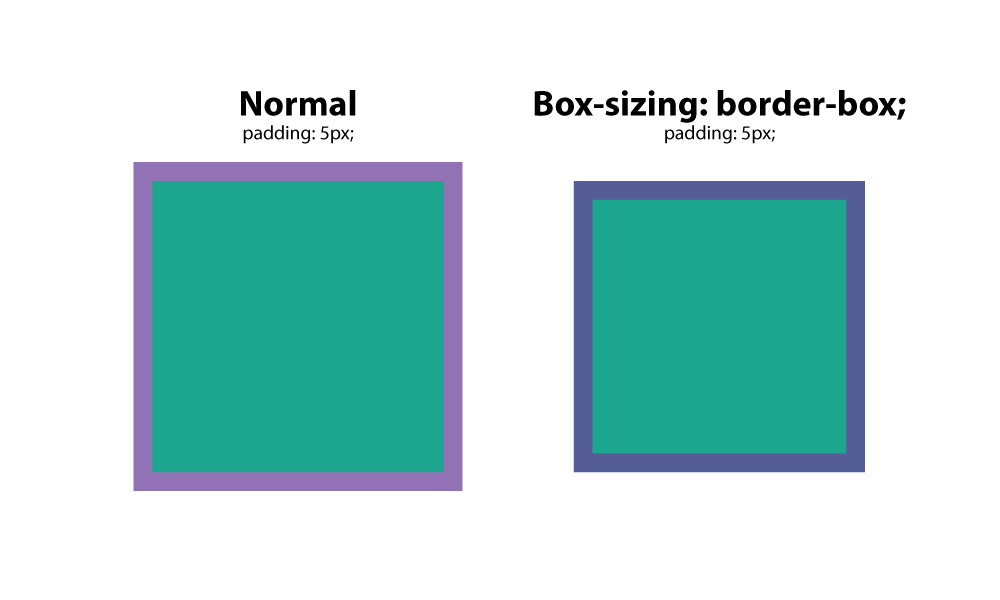
}

Whether you’ve been a web designer for years, or you’re just starting out, learning how to build websites the right way can seem like a rocky, never-ending journey. Once you’ve narrowed down which languages you want to learn, you have to learn and refine your skills.

No matter what you learn, CSS is one of those essential, but daunting skills you have to master. It doesn’t have to be so difficult, though, especially if you know a few handy and lesser-known CSS techniques to get the job done.

## 12. box-sizing: border-box;

This is a favorite among many web designers, because it solves the problem of padding and layout issues. Basically, when you set a box to a specific width, and add padding to it, the padding adds to the size of the box. However, with box-sizing:border-box;, this is negated, and boxes stay the size they are meant to be.



## 13. :before

This CSS is a selector that allows you to choose a CSS element and insert content before every element with a specific class applied to it. Let’s say you had a website where you wanted specific text before every H2 tag. You would us this setup:

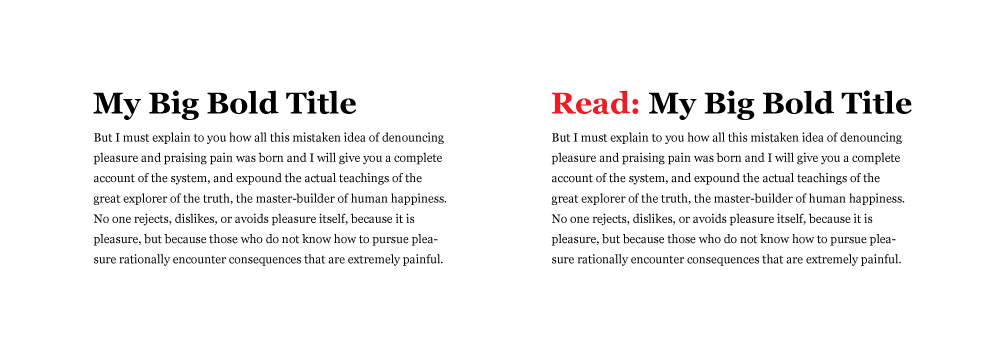
h2:before {

content: "Read: ";

<span class="Apple-converted-space"> color: #F00;</span>

}

This is extremely handy, especially if you are using an icon font. You can place icons before certain elements, and apply it globally.



## 14. :after

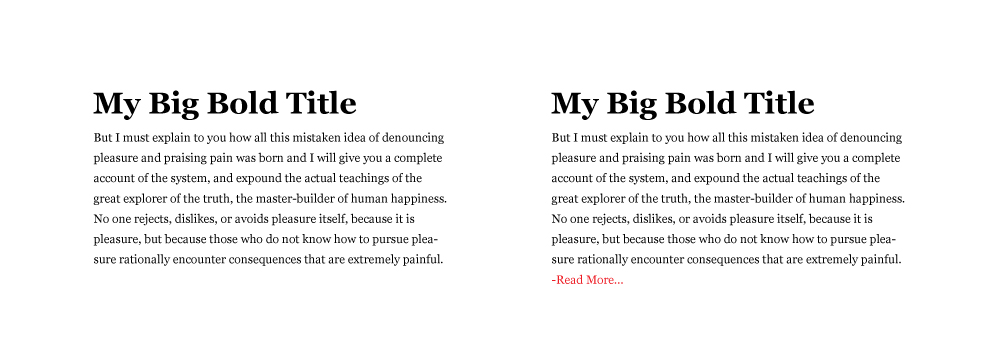
Like the :before selector, you can use :after to insert content globally on specific elements. A practical use would be adding “read more” after every excerpt on a blog. Here’s how you would do that.

p:after{

content: " -Read more… ";

color:#f00;

}



## 15. content

content is a CSS property that comes in handy when you need to insert an element that you want to be able to control. The most common use I’ve seen for this is to insert an icon from an icon font in a specific place. In the examples above, you can see that you have to wrap the text you want to insert in quotation marks.

## 16. CSS reset

Different browsers have default CSS settings, so it is a must to reset those, so you have an even, consistent playing field. Think of it as building a house, and whether you build on the side of a mountain, on a sandy beach, or on the middle of a wooded area, you want that foundation to be level.

This CSS reset method sets a standard base for all of your websites, giving them consistency in their CSS starting point. It removes unwanted borders, preset margins, padding, lines heights, styles on lists, etc. Eric Meyer created [one that works well](http://meyerweb.com/eric/tools/css/reset/).

## 17. Drop caps

Everyone loves drop caps. It reminds us of the traditional printed book, and is a great way to start a page of content. That 1st, large letter really grabs your attention. There’s an easy way to create a drop cap in css, and it’s by using the pseudo element: :first letter. Here’s an example :

p:first-letter{

display:block;

float:left;

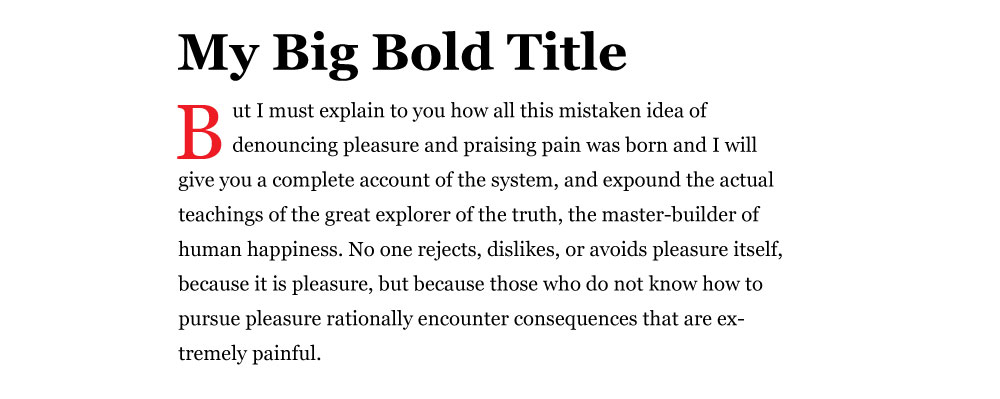
margin:3px;

color:#f00;

font-size:300%;

}

What this does is set the letter to 3x the size of the other letters. It sets 3px of space around the letter to prevent overlapping, and sets the color of the letter to red.



## 18. Force text to be all caps, all lowercase, or capitalized

It would be absurd to type an entire section in all caps. Imagine having to go back and fix that later when the format of the website changes, or it gets updated. Instead, use the following css styles to force text to a certain formatting. This css targets the h2 title tag.

* h2 { text-transform: uppercase; } – all caps
* h2 { text-transform: lowercase; } – all lowercase
* h2 { text-transform: capitalize; } – capitalizes the 1st letter of each word.



## 19. Vertical screen height

Sometimes you want a section to fill the entire screen, no matter what the screen size is. You can control this with vh, or view height. The number before it is a percentage, so if you want it to fill 100% of the browser, you would set it to 100. You might set it to a value like 85% to accommodate a fixed navigation menu.

Create a class for the container and apply the amount of vh you want it to have. One thing you may need to tweak is the media query value for specific screens or orientations like phones in portrait mode. Imagine stretching a landscape image to fit portrait mode. That just wouldn’t look good.

.fullheight { height: 85vh; }

## 20. Style telephone links

If you have a link that calls a phone number when a user taps it on their phone, you may have trouble styling it with the traditional active link selector. Instead, use the following CSS:

a[href^=tel] {

<span class="Apple-converted-space"> color: #FFF;</span>

<span class="Apple-converted-space"> text-decoration: none;</span>

}

## What is JavaScript?

JavaScript was initially created to “make web pages alive”.

The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

initially it was called as “LiveScript”.

Intially JavaScript was executed only in the browser, but now also in server, or on any device that has a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).

The browser has an embedded engine sometimes called a “JavaScript virtual machine”.

Different engines have different “codenames”. For example:

* [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) – in Chrome and Opera.
* [SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey) – in Firefox.
* There are other codenames like “Trident” and “Chakra” for different versions of IE,
* “ChakraCore” for Microsoft Edge,
* “Nitro” and “SquirrelFish” for Safari, etc.

Es5 or es6(es2015)

Since everyone cannot only depend on JS they can use diff languages which can be transpiled.

Transpilers – source to source converters

Traceur – google project

Babel - 7

Examples of such languages:

* [CoffeeScript](http://coffeescript.org/) is a “syntactic sugar” for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
* [TypeScript](http://www.typescriptlang.org/) is concentrated on adding “strict data typing” to simplify the development and support of complex systems. It is developed by Microsoft.
* [Flow](http://flow.org/) also adds data typing, but in a different way. Developed by Facebook.
* [Dart](https://www.dartlang.org/) is a standalone language that has its own engine that runs in non-browser environments (like mobile apps), but also can be transpiled to JavaScript. Developed by Google.

There are more. Of course, even if we use one of transpiled languages

**Demo 1:**

**Auto loading of script in html**

<script>

alert( 'Hello, world!' );

</script>

Keep this code in html

## External scripts: <script src="/path/to/script.js"></script>

<script src="file.js">

alert(1); // the content is ignored, because src is set

</script>

## Task 1:

## Create a page that shows a message “I’m JavaScript!”. Alert & console.log – embedded js

1. Take the solution of the previous task Show an alert. Modify it by extracting the script content into an external file alert.js, residing in the same folder. Open the page, ensure that the alert works.
2. Find the usage of ‘use strict’
3. Noscript tag

Refer other word doc for more contents on js [(click)](file:///D:\All%20tuts\js\js%20tuts%20mine.docx)

## String Conversion

String conversion happens when we need the string form of a value.

For example, alert(value) does it to show the value.

We can also call the String(value) function to convert a value to a string:

let value = true;

alert(typeof value); // boolean

value = String(value); // now value is a string "true"

alert(typeof value); // string

String conversion is mostly obvious. A false becomes "false", null becomes "null", etc.

## [Numeric Conversion](https://javascript.info/type-conversions" \l "numeric-conversion)

Numeric conversion happens in mathematical functions and expressions automatically.

For example, when division / is applied to non-numbers:

alert( "6" / "2" ); // 3, strings are converted to numbers

We can use the Number(value) function to explicitly convert a value to a number:

let str = "123";

alert(typeof str); // string

let num = Number(str); // becomes a number 123

alert(typeof num); // number

Numeric conversion rules:

| **Value** | **Becomes…** |
| --- | --- |
| undefined | NaN |
| null | 0 |
| true and false | 1 and 0 |
| string | Whitespaces from the start and end are removed. If the remaining string is empty, the result is 0. Otherwise, the number is “read” from the string. An error gives NaN. |

Examples:

alert( Number(" 123 ") ); // 123

alert( Number("123z") ); // NaN (error reading a number at "z")

alert( Number(true) ); // 1

alert( Number(false) ); // 0

## [Default values](https://javascript.info/function-basics" \l "default-values)

If a parameter is not provided, then its value becomes undefined.

For instance, the aforementioned function showMessage(from, text) can be called with a single argument:

showMessage("Ann");

That’s not an error. Such a call would output "Ann: undefined". There’s no text, so it’s assumed that text === undefined.

If we want to use a “default” text in this case, then we can specify it after =:

function showMessage(from, text = "no text given") {

alert( from + ": " + text );

}

showMessage("Ann"); // Ann: no text given

Now if the text parameter is not passed, it will get the value "no text given"

Can also call another function :

function showMessage(from, text = anotherFunction())

There is another syntax for creating a function that is called a Function Expression.

It looks like this:

let sayHi = function() {

alert( "Hello" );

};

Console.log(sayHi);

Function declaration:

sayHello()

{ console.log(‘hi’);}

Task : diff between function declaration and function expression