Table of Contents

[What are semantic elements in HTML5? 3](#_Toc526872591)

[When do you use which semantic element? 3](#_Toc526872592)

[Why is the correct hierarchy of headlines important? 4](#_Toc526872593)

[Should you use multiple h1 headlines in one HTML document? 4](#_Toc526872594)

[How do you include webfonts? 4](#_Toc526872595)

[Which webfont formats could be used? 4](#_Toc526872596)

[Why are there different formats? 5](#_Toc526872597)

[How can you provide different font weights and different font styles for a webfont? 5](#_Toc526872598)

[How are CSS stylesheets embedded? 6](#_Toc526872599)

[What is the purpose of normalize.css? 6](#_Toc526872600)

[What's the effect of specifity of CSS selectors? 6](#_Toc526872601)

[What has to be considered when declaring CSS selectors? 6](#_Toc526872602)

[What does !important do and why shouldn't you use it? 6](#_Toc526872603)

[What are vendor-prefixes and which ones do exist? 7](#_Toc526872604)

[What are pseudo elements? What is their purpose? 7](#_Toc526872605)

[What are shorthand properties? Which ones do exist? 8](#_Toc526872606)

[When do you use img tags and when CSS background images? 8](#_Toc526872607)

[What are the different position properties and what's their effect on surrounding elements? 8](#_Toc526872608)

[What is z-index and what's the effect? 9](#_Toc526872609)

[How can you place two block elements side by side? 9](#_Toc526872610)

[What is CSS box-model? 10](#_Toc526872611)

[What is CSS flex-box? When do you use it? 10](#_Toc526872612)

# **What are semantic elements in HTML5?**

Semantic elements explains clearly what it is. Examples: <form> <article> <table>

# **When do you use which semantic element?**

**<article>**

Independent content. It should make sense on its own. For example: forum post, blog post, news article



**<aside>**

Content aside from the content it is placed (like a sidebar). It should relate to the surrounding content.

**<details>**

Additional details that the user can view or hide.

**<figcaption>**

You can add a visual explanation to an image.

**<figure>**

Illustrations, diagrams, photos, code listings etc.

**<footer>**

A footer for a document or section. It should contain information about its containing element. For example: author of the document, copyright information, links to terms of use, contact information.

**<header>**

A header for a document or section. Use as a container for introduction of content.

**<main>**

The main content of a document.

**<mark>**

Marked or highlighted text.

**<nav>**

A set of navigation links.

**<section>**

a part in a document. A thematic grouping of content, typically with a heading. A homepage could be split in the sections for introduction, content and contact information.

**<summary>**

A visible heading for a <details> element.

**<time>**

Date and time.

# **Why is the correct hierarchy of headlines important?**

Search engines use this to index the structure and content of your web pages. And users skim your page via the headings. It is important to use this to show the structure for your users but also for your colleague developers.

# **Should you use multiple h1 headlines in one HTML document?**

Yes you can, it should be used for main headings.

# **How do you include webfonts?**

You have to install the font on your webserver. After this users will automatically download the font when they are on your page. At font-family you give the font a name.

@font-face {  
    font-family: myFirstFont;  
    src: url(sansation\_light.woff);  
}

# **Which webfont formats could be used?**

**TrueType Fonts (TTF)**

Developed by Apple and Microsoft and is most common format for fonts on Mac and Windows operating systems. All major broswers have supported it. Larger file size because not compressed.

**OpenType Fonts (OTF)**

Evolution of TTF. Made by adobe and Microsoft. Contains the screen and printer font data in one component. Has exclusive capabilities including support for multiple platforms and expanded character sets. Can be used by Mac and Windows operating systems. Permits storage of up to 65.000 characters giving designers the freedom to include add-ons.

**The Web Open Font Format (WOFF)**

Essentially OTF or TTF with metadata and compression supported by all major browers. A collaboration by Mozilla, Microsoft and Opera. It’s metadata allows for the inclusion of license data within the font file to address copyright issues. Compressed so they load faster.

**The Web Open Font Format (WOFF 2.0)**

Next generation of WOFF. Offering 30% average compression gain over the original WOFF. It is a WWW consortium recommendation and is clearly the future of font formats.

**SVG Fonts/Shapes**

They use the SVG’s ‘font’ element. Contains glyph outlines as standard SVG elements and attributes as if they were a single vector object in the SVG image. It is not great for body text. If you are targeting Iphone and Ipad users, SVG fonts are your only choice.

**Embedded OpenType Fonts (EOT)**

Designed by Microsoft. Uses compression and subsetting to make the font files smaller.An attempt to address the copyright shortcomings of TTF and OTF when publishing on the web. Uses encryption for further protection. Only supported by Internet Explorer.

# **Why are there different formats?**

They are created by different companies and none of them is usable on all browsers/device.

# **How can you provide different font weights and different font styles for a webfont?**

Within the @fontface you should use font-weight and font-style. You can use these different weights and styles:

@font-face {  
    font-family: myFirstFont;  
    src: url(sansation\_bold.woff);

    font-weight: bold;

normal;  
 bold;  
 100;  
 200;  
 300;  
 400;  
 500;  
 600;  
 700;  
 800;  
 900;

font-style: normal;

italic;  
 oblique;  
}

# **How are CSS stylesheets embedded?**

There are 3 ways to embed CSS stylesheets.

**External style sheet**

<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>

**internal style sheet**

<head>  
<style>  
body {  
    background-color: linen;  
}  
  
h1 {  
    color: maroon;  
    margin-left: 40px;  
}   
</style>  
</head>

**Inline style**

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

# **What is the purpose of normalize.css?**

It is an alternative for CSS reset. It is a small CSS file that provides better cross-browser consistency in the default styling of HTML elements. It’s a modern, HTML5-ready, alternative to the traditional CSS reset.

# **What's the effect of specifity of CSS selectors?**

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more. The element selector selects elements based on the element name. If you use specified CSS selectors as good as possible, then it’s also easier to change all those elements based on the same element name.

# **What has to be considered when declaring CSS selectors?**

If you are going to change one specific selector in CSS. Every element with that name will be changed. So be careful to use the right selector for the right things.

# **What does !important do and why shouldn't you use it?**

A rule that has the !important directive is always applied no matter where that rule appears in the CSS document. If you lean too heavily on the !important declaration to achieve your desired styles, you will eventually have a style sheet littered with !important styles. You will be fundamentally changing the way that page's CSS is processed. It is a lazy practice that is not good from a long-term management standpoint. Use !important for testing or, in some cases, when you absolutely must override an inline style that is part of a theme or template framework.

# **What are vendor-prefixes and which ones do exist?**

CSS vendor prefixes, also sometimes known as or [CSS](https://www.lifewire.com/what-is-css-3466390) browser prefixes, are a way for browser makers to add support for [new CSS features](https://www.lifewire.com/css2-vs-css3-3466978) before those features are fully supported in all browsers. This may be done during a sort of testing and experimentation period where the browser manufacturer is determining exactly how these new CSS features will be implemented.

Android:

-webkit-

Chrome:

-webkit-

Firefox:

-moz-

Internet Explorer:

-ms-

iOS:

-webkit-

Opera:

-o-

Safari:

-webkit-

For example, if you want to add a CSS3 transition to your document, you would use thetransition property as shown below:

-webkit-transition: all 4s ease;  
-moz-transition: all 4s ease;  
-ms-transition: all 4s ease;  
-o-transition: all 4s ease;  
transition: all 4s ease;

# **What are pseudo elements? What is their purpose?**

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

Example The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

p::first-line {  
    color: #ff0000;  
    font-variant: small-caps;  
}

**All Pseudo Elements**

::after p::after Insert something after the content of each <p> element

::before p::before Insert something before the content of each <p> element

::first-letter p::first-letter Select the first letter of each <p> element

::first-line p::first-line Select the first line of each <p> element

::selection p::selection Selects the portion of an element that is selected by a user

# **What are shorthand properties? Which ones do exist?**

are CSS properties that let you set the values of multiple other CSS properties simultaneously.

Example:

Font-style: italic;

Font-weight: bold;

Font-size: .8em;

Line-height: 1.2;

Font-family: Arial, sans-serif;

This can be shortened to

Font: italic bold .8em/1.2 Arial, sans-serif;

# **When do you use img tags and when CSS background images?**

An img tag defines a img in an html page. When it is a CSS background image it specifies an image to use it as the background of an element.

# **What are the different position properties and what's their effect on surrounding elements?**

The position property specifies the type of positioning method used for an element.

There are five different position values:

**Static**

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

**Relative**

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

**Absolute**

This is a very powerful type of positioning that allows you to literally place any page element exactly where you want it. You use the positioning attributes top, left, bottom. and right to set the location. Remember that these values will be relative to the next parent element with relative (or absolute) positioning. If there is no such parent, it will default all the way back up to the <html> element itself meaning it will be placed relatively to the page itself.

The trade-off (and most important thing to remember) about absolute positioning is that these elements are removed from the flow of elements on the page. An element with this type of positioning is not affected by other elements and it doesn't affect other elements.

**Fixed**

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

**Sticky**

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

<https://www.youtube.com/watch?v=P6UgYq3J3Qs>

# **What is z-index and what's the effect?**

The z-index defines the order/layer of an element. Does it need to go to the background or to the front?

# **How can you place two block elements side by side?**

Using inline-block. Inline-block elements are like inline elements but they can have a width and height. Let's look at examples of both approaches.

Example:

.box2 {

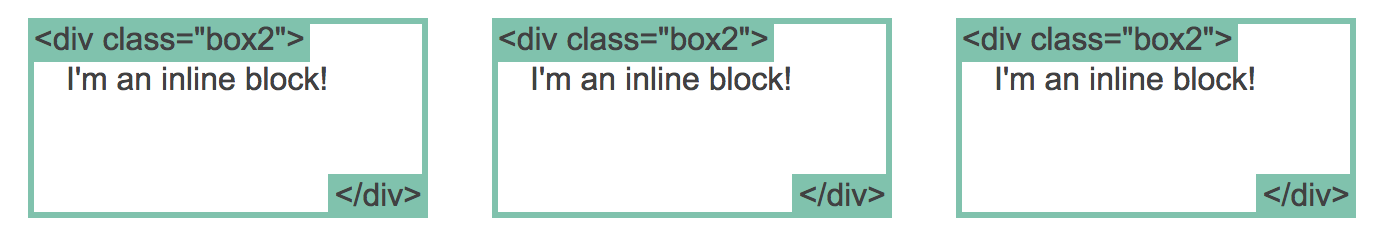
display: inline-block;

width: 200px;

height: 100px;

margin: 1em;

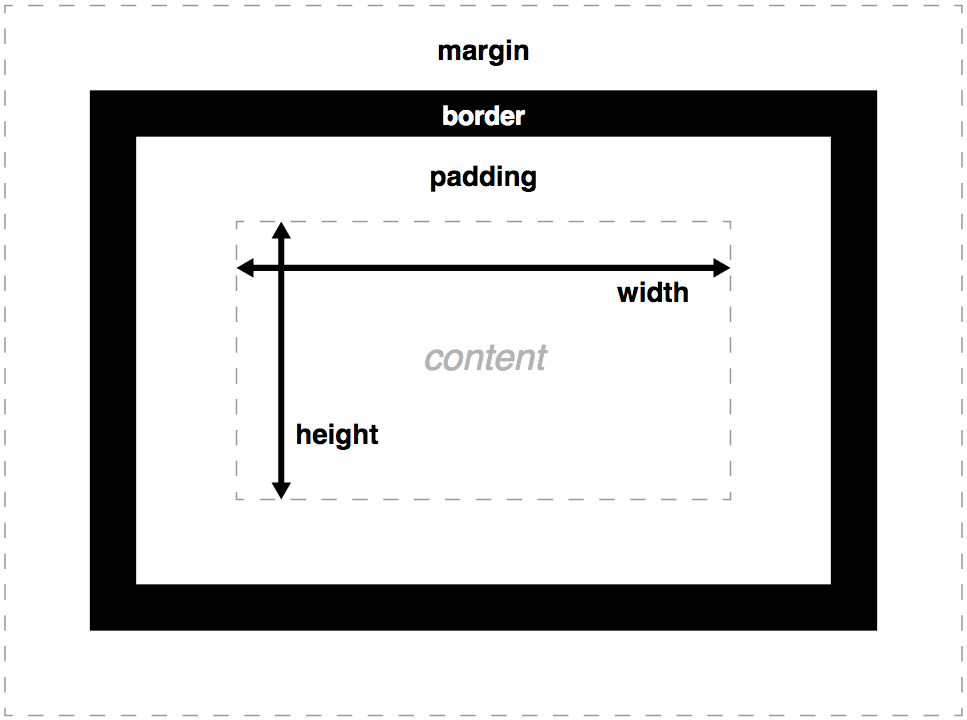
}



# **What is CSS box-model?**

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



# **What is CSS flex-box? When do you use it?**

It makes it easier to design flexible responsive layout structure without using float or positioning. Within the flexbox you can easily manipulate the elements

# **Which semantic element should be used for page navigation?**

<nav>

# **How can you add semantic meaning to any HTML elements?**

Name them correctly.

# **Why should semantic attributes be added to html elements?**

It’s useful for developers, for you, and for users (with screen reading applications) to navigate through your website.