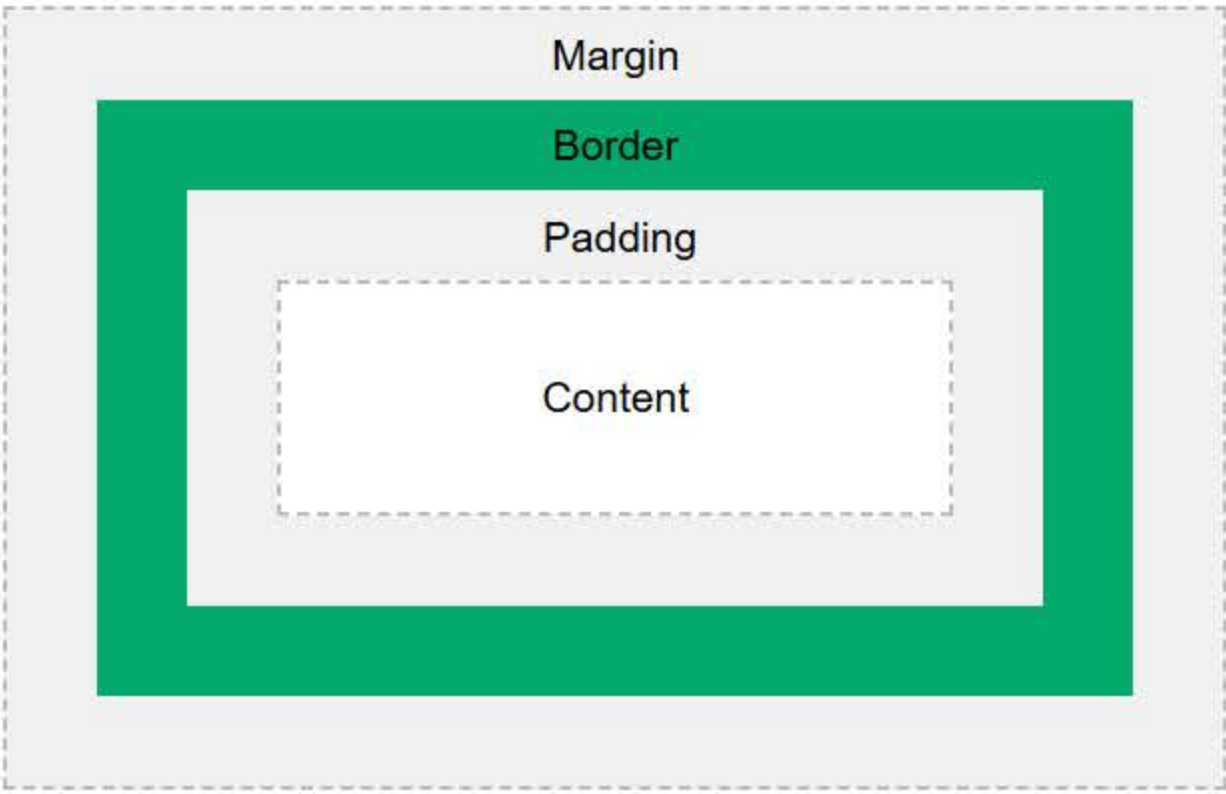


CSS Box Model



**Content** - The content of the box, where text and images appear  
**Padding** - Clears an area around the content. The padding is transparent  
**Border** - A border that goes around the padding and content  
**Margin** - Clears an area outside the border. The margin is transparent

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

CSS has a lot of properties for formatting **text**:  
**alignment decoration transformation spacing shadow**

```
p {
  color: red;
  text-align: center;
}
```

**p** is a selector in CSS (it points to the HTML element you want to style: <p>).  
**color** is a property, and red is the property value  
**text-align** is a property, and center is the property value

CSS Comments

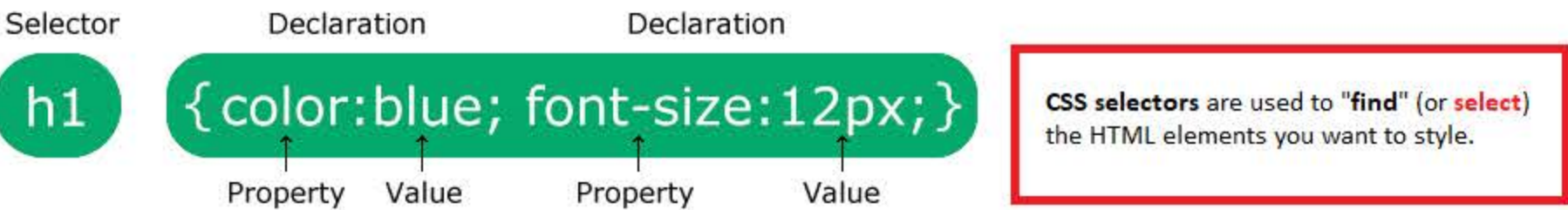
```
/* This is a single-line comment */
p {
  color: red;
}

p {
  color: red; /* Set text color to red */
}

p {
  color: /*red*/blue;
}

<!-- These paragraphs will be red -->
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>
```

CSS Syntax CSS Selectors



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

| simple selectors           |            |   |
|----------------------------|------------|---|
| Selector                   | Example    | Example description                             |
| # <i>id</i>                | #firstname | Selects the element with id="firstname"         |
| . <i>class</i>             | .intro     | Selects all elements with class="intro"         |
| *                          | *          | Selects all elements                            |
| <i>element</i>             | p          | Selects all <p> elements                        |
| <i>element,element,...</i> | div, p     | Selects all <div> elements and all <p> elements |

CSS id Selector

```
#para1 {
  text-align: center;
  color: red;
}
```

<p id="para1">Hello World!</p>  
<p>This paragraph is not affected by the style.</p>

The id of an element is unique within a page, so the id selector is used to select one unique element!

CSS class Selector

```
.center {
  text-align: center;
  color: red;
}
```

p.center {  
  text-align: center;  
  color: red;  
}

The class selector selects HTML elements with a specific class attribute; with a specific class, write a period (.) character, followed by the class name.

CSS Universal Selector

```
* {
  text-align: center;
  color: blue;
}
```

The universal selector (\*) selects all HTML elements on the page.

CSS Grouping Selector

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

How To Add CSS

External CSS  
Internal CSS  
Inline CSS

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules:  
**Inline style** (inside an HTML element)  
**External and internal** style sheets (in the head section)  
Browser **default**

**internal vs. external**

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
```

<h1>This is a heading</h1>  
<p>The style of this document is a combination of an external stylesheet, and internal style</p>

```
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
```

<h1>This is a heading</h1>  
<p>This is a paragraph.</p>

```
</body>
</html>
```

**mystyle.css**

```
body {
  background-color: lightblue;
}
```

h1 {  
  color: navy;  
  margin-left: 20px;  
}

If the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy", not orange, If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange"

**internal**

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

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

<h1>This is a heading</h1>  
<p>This is a paragraph.</p>

```
</body>
</html>
```

**inline**

```
<!DOCTYPE html>
<html>
<body>
```

<h1 style="color:blue;text-align:center;">This is a heading</h1>  
<p style="color:red;">This is a paragraph</p>

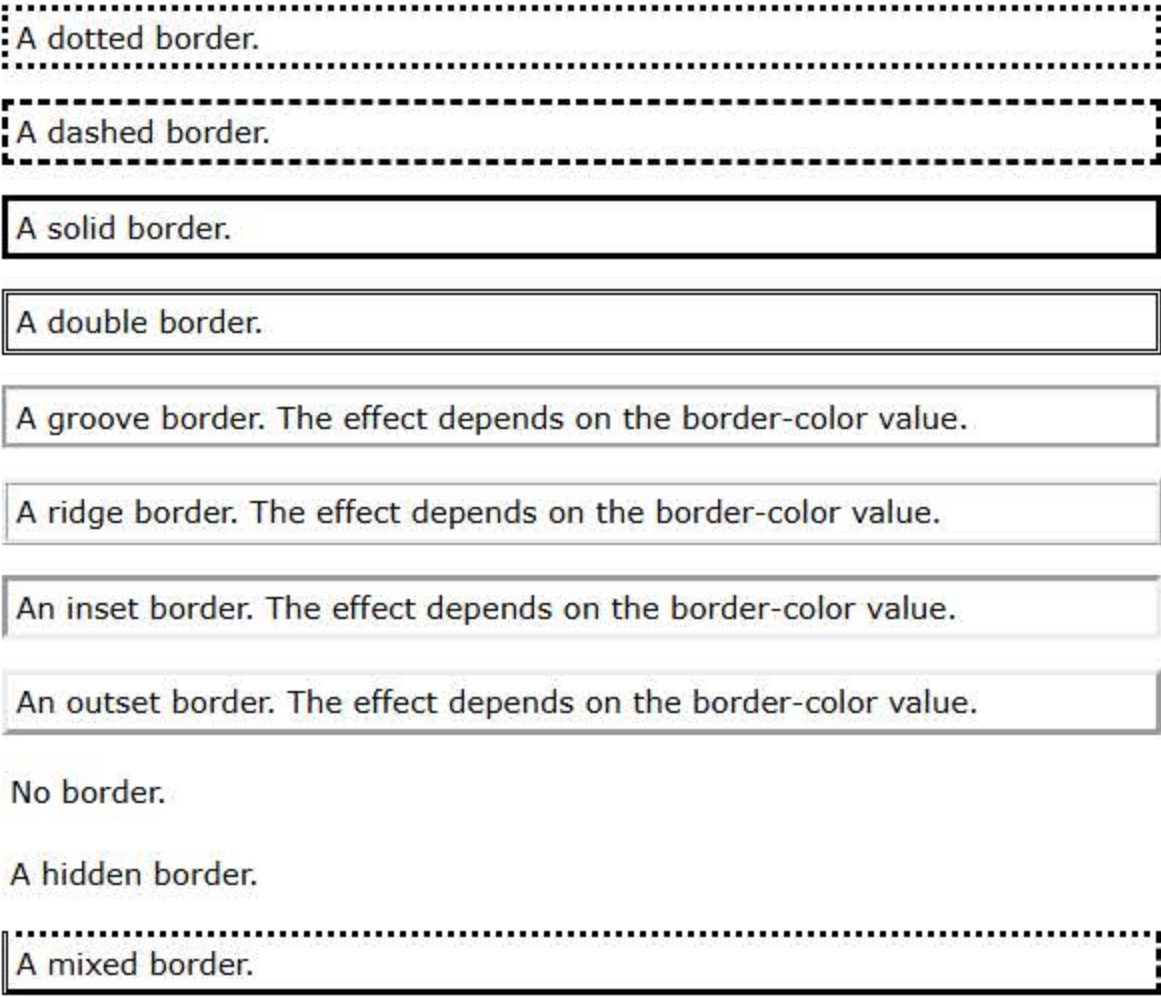
```
</body>
</html>
```

In this example the <p> element will be styled according to class="center" and to class="large":

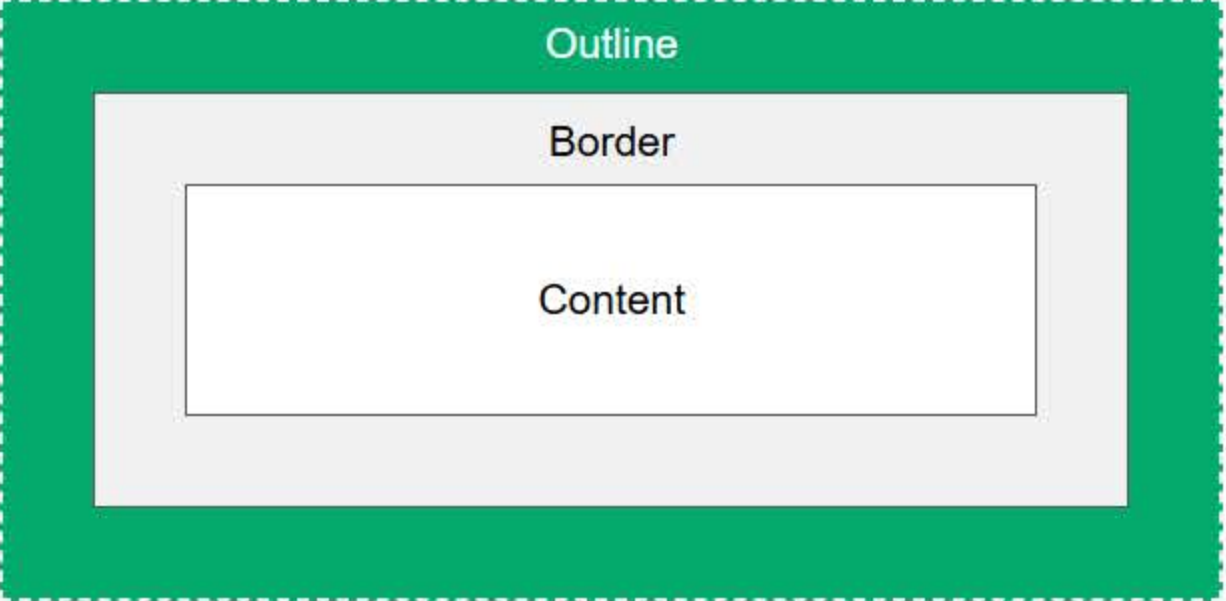
```
<p class="center large">This paragrap</p>
```

```
<style>
p.center {
  text-align: center;
  color: red;
}

p.large {
  font-size: 300%;
}
</style>
```



CSS Outline



An outline is a line that is drawn around elements, **OUTSIDE** the borders, to make the element "stand out".

**dotted** - Defines a dotted outline  
**dashed** - Defines a dashed outline  
**solid** - Defines a solid outline  
**double** - Defines a double outline  
**groove** - Defines a 3D grooved outline  
**ridge** - Defines a 3D ridged outline  
**inset** - Defines a 3D inset outline  
**outset** - Defines a 3D outset outline  
**none** - Defines no outline  
**hidden** - Defines a hidden outline

CSS Fonts

**Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.  
**Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.  
**Monospace** fonts have the same fixed width. They create a mechanical look.  
**Cursive** fonts imitate human handwriting.  
**Fantasy** fonts are decorative/playful fonts.

Tip: The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. Start with the font you want, and end with a generic family (to let the browser pick a similar font in the generic family, if no other fonts are available). The font names should be separated with a comma.

Font Awesome Icons

```
<!DOCTYPE html>
<html>
<head>
<script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>
</head>
<body>
```

<i class="fas fa-cloud"></i>

<p>Styled Font Awesome icons (size and color):</p>  
<i class="fas fa-cloud" style="font-size:24px;"></i>  
<i class="fas fa-cloud" style="font-size:36px;"></i>  
<i class="fas fa-cloud" style="font-size:48px;color:red;"></i>  
<i class="fas fa-cloud" style="font-size:60px;color:lightblue;"></i>

```
</body>
</html>
```

| Generic Font Family | Examples of Font Names                  |
|---------------------|---|
| Serif               | Times New Roman<br>Georgia<br>Garamond  |
| Sans-serif          | Arial<br>Verdana<br>Helvetica           |
| Monospace           | Courier New<br>Lucida Console<br>Monaco |
| Cursive             | Brush Script MT<br>Lucida Handwriting   |
| Fantasy             | Copperplate<br>Papyrus                  |

Styling Links

The four links states are:

**a:link** - a normal, unvisited link  
**a:visited** - a link the user has visited  
**a:hover** - a link when the user mouses over it  
**a:active** - a link the moment it is clicked

```
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

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

/* selected link */
a:active {
  color: blue;
}
```



CSS Lists

**unordered lists (<ul>)** - the list items are marked with bullets  
**ordered lists (<ol>)** - the list items are marked with numbers or letters

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

The list-style property is a shorthand property. It is used to set all the list properties in one declaration:

```
ul {  
  list-style: square inside url("sqpurple.gif");  
}
```

CSS Tables

```
table, th, td {  
  border: 1px solid;  
}
```

Table properties:  
table borders  
table size  
table alignment  
table style  
table responsive

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.  
The default display value for most elements is **block** or **inline**.

Block-level Elements

A block-level element ALWAYS starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

**Note:** Setting the display property of an element only changes how the element is displayed, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements


An inline element DOES NOT start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:


- <span>
- <a>
- <img>

display:none




Remove

visibility:hidden



Hide

Reset



Reset All

| Property   | Description   |
|------------|---|
| display    | Specifies how an element should be displayed          |
| visibility | Specifies whether or not an element should be visible |

```
ul.a {  
  list-style-type: circle;  
}  
  
ul.b {  
  list-style-type: square;  
}  
  
ol.c {  
  list-style-type: upper-roman;  
}  
  
ol.d {  
  list-style-type: lower-alpha;  
}
```

list property

| Property            | Description   |
|---------------------|---|
| list-style          | Sets all the properties for a list in one declaration           |
| list-style-image    | Specifies an image as the list-item marker                      |
| list-style-position | Specifies the position of the list-item markers (bullet points) |
| list-style-type     | Specifies the type of list-item marker                          |

The **list-style-image** property specifies an image as the list item marker:

```
ul {  
  list-style-image: url('sqpurple.gif');  
}
```

```
ul.a {  
  list-style-position: outside;  
}  
  
ul.b {  
  list-style-position: inside;  
}
```

A common example is making **inline <li> elements for horizontal menus:**

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
  li {  
    display: inline;  
  }  
</style>  
</head>  
<body>
```

<p>Display a list of links as a horizontal menu:</p>

```
<ul>  
  <li><a href="/html/default.asp" target="_blank">HTML</a></li>  
  <li><a href="/css/default.asp" target="_blank">CSS</a></li>  
  <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>  
</ul>
```

```
</body>  
</html>
```

Display a list of links as a horizontal menu:

[HTML](#) [CSS](#) [JavaScript](#)

The element **will still take up the same space as before**, it will be hidden, but still affect the layout:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
  h1.hidden {  
    visibility: hidden;  
  }  
</style>  
</head>  
<body>  
  
  <h1>This is a visible heading</h1>  
  <h1 class="hidden">This is a hidden heading</h1>  
  <p>Notice that the hidden heading still takes up space.</p>  
  
</body>  
</html>
```

This is a visible heading

Notice that the hidden heading still takes up space.

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the **width** of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This <div> element has a width of 500px, and margin set to auto.

**Note:** The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using **max-width** instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This <div> element has a max-width of 500px, and margin set to auto.

This <div> element has a width of 500px, and margin set to auto.

**Note:** The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using **max-width** instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This <div> element has a max-width of 500px, and margin set to auto.

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}
```

```
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```

CSS Layout - The position

There are five different position values:  
**static relative fixed absolute sticky**

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

```
div.sticky {  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}
```

```
<style>  
.container {  
  position: relative;  
}
```

```
<div class="container">  
    
  <div class="center">Centered</div>  
</div>
```

```
div.static {  
  position: static;  
  border: 3px solid #73AD21;  
}
```

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

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).

```
img {  
  width: 100%;  
  height: auto;  
  opacity: 0.3;  
}
```

Here is the CSS that is used:

```
div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}
```

Setting the top, right, bottom, and left properties of a **relatively-positioned element** will cause it to be adjusted away from its normal position.

Positioning Text In an Image

Example



This <div> element has position: relative;

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, it **does not leave a gap in the page** where it would normally have been located.

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```

```
<div class="container">  
    
  <div class="topleft">Top Left</div>  
</div>
```

```
<style>  
.container {  
  position: relative;  
}
```

```
.topleft {  
  position: absolute;  
  top: 8px;  
  left: 16px;  
  font-size: 18px;  
}
```

```
img {  
  width: 100%;  
  height: auto;  
  opacity: 0.3;  
}
```



