

CSS FOR STYLING

CASCADING STYLE SHEETS (CSS)

Describes the appearance, layout, and presentation of information on a web page

- HTML describes **the content** of the page

Describes *how* information is to be displayed, not *what* is being displayed

Can be embedded in HTML document or placed into separate .css file

BASIC CSS RULE SYNTAX

```
selector {  
property: value;  
property: value;  
...  
property: value;  
}
```

CSS

```
p {  
font-family: sans-serif;  
color: red;  
}
```

CSS

A CSS file consists of one or more **rules**

Each rule starts with a **selector**

A selector specifies an HTML element(s) and then applies style **properties** to them

- a selector of * selects all elements

ATTACHING A CSS FILE <LINK>

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

HTML

```
<link href="style.css" type="text/css" rel="stylesheet" />
<link href="http://www.google.com/uds/css/gsearch.css"
rel="stylesheet" type="text/css" />
```

HTML

A page can link to multiple style sheet files

- In case of a conflict (two sheets define a style for the same HTML element), the latter sheet's properties will be used

EMBEDDING STYLE SHEETS: <STYLE>

```
<head>
<style type="text/css">
p { font-family: sans-serif; color: red; }
h2 { background-color: yellow; }
</style>
</head>
```

HTML

CSS code can be embedded within the head of an HTML page

Bad style and should be avoided when possible (why?)

INLINE STYLES: THE STYLE ATTRIBUTE

```
<p style="font-family: sans-serif; color: red;">  
This is a paragraph</p>
```

HTML

This is a paragraph

output

Higher precedence than embedded or linked styles

Used for one-time overrides and styling a particular element

Bad style and should be avoided when possible (why?)

CSS PROPERTIES FOR COLORS

```
p {  
  color: red;  
  background-color: yellow;  
}
```

CSS

This paragraph uses the style above

output

property	description
color	color of the element's text
background-color	color that will appear behind the element

SPECIFYING COLORS

```
p { color: red; }  
h2 { color: rgb(128, 0, 196); }  
h4 { color: #FF8800; }
```

CSS

This paragraph uses the first style above

This h2 uses the second style above.

This h4 uses the third style above.

output

color names: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white (white), yellow

RGB codes: red, green, and blue values from 0 (none) to 255 (full)

hex codes: RGB values in base-16 from 00 (0, none) to FF (255, full)

GROUPING STYLES

```
p, h1, h2 {  
  color: green;  
}  
h2 {  
  background-color: yellow;  
}
```

CSS

This paragraph uses the above style.

This h2 uses the above styles.

output

A style can select multiple elements separated by commas

The individual elements can also have their own styles

CSS COMMENTS /*...*/

```
/* This is a comment.  
It can span many lines in the CSS file. */  
p {  
color: red; background-color: aqua;  
}                                CSS
```

CSS (like HTML) is usually not commented as rigorously as programming languages such as Java

The // single-line comment style is NOT supported in CSS

The <!-- ... --> HTML comment style is also NOT supported in CSS

CSS PROPERTIES FOR FONTS

property	description
font-family	which font will be used
font-size	how large the letters will be drawn
font-style	used to enable/disable italic style
font-weight	used to enable/disable bold style

[Complete list of font properties](https://www.w3schools.com/cssref/default.asp) (<https://www.w3schools.com/cssref/default.asp>)

FONT-FAMILY

```
p {  
  font-family: Georgia;  
}  
h2 {  
  font-family: "Courier New";  
}
```

CSS

This paragraph uses the first style above.

This h2 uses the second style above.

output

Enclose multi-word font names in quotes

MORE ABOUT FONT-FAMILY

```
p {  
font-family: Garamond, "Times New Roman", serif;  
}
```

CSS

This paragraph uses the above style.

output

We can specify multiple fonts from highest to lowest priority

Generic font names:

- serif, sans-serif, cursive, ~~FANTASY~~, monospace

If the first font is not found on the user's computer, the next is tried

Placing a generic font name at the end of your font-family value, ensures that every computer will use a valid font

FONT-SIZE

```
p {  
  font-size: 24pt;  
}
```

CSS

This paragraph uses the style above.

output

units: pixels (**px**) vs. point (**pt**) vs. m-size (**em**)

16px, 16pt, 1.16em

vague font sizes: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger

percentage font sizes, e.g.: 90%, 120%

FONT-SIZE

```
p {  
  font-size: 24pt;  
}
```

CSS

This paragraph uses the style above.

output

pt specifies number of point, where a point is 1/72 of an inch onscreen

px specifies a number of pixels on the screen

em specifies number of m-widths, where 1 em is equal to the font's current size

CSS SIZE ELEMENTS

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	Defines a measurement in centimeters.	div {margin-bottom: 2cm;}
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p {word-spacing: .15in;}
mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

FONT-WEIGHT, FONT-STYLE

```
p {  
  font-weight: bold;  
  font-style: italic;  
}
```

CSS

This paragraph uses the style above.

output

Either of the above can be set to normal to turn them off (e.g. headings)

CSS PROPERTIES FOR TEXT

property	description
text-align	alignment of text within its element
text-decoration	decorations such as underlining
line-height, word-spacing, letter-spacing	gaps between the various portions of the text
text-indent	indents the first letter of each paragraph

[Complete list of text properties](https://www.w3schools.com/cssref/default.asp) (<https://www.w3schools.com/cssref/default.asp>)

TEXT-ALIGN

```
blockquote { text-align: justify; }  
h2 { text-align: center; }  
CSS
```

The Gollum's Quote

We wants it, we needs it. Must have the precious. They stole it from us.
Sneaky little hobbitses. Wicked, tricky, false!

output

`text-align` can be `left`, `right`, `center`, or `justify`

TEXT-DECORATION

```
p {  
  text-decoration: underline;  
}
```

CSS

This paragraph uses the style above.

output

can also be overline, ~~line-through~~, blink, or none
effects can be combined:

```
text-decoration: overline underline;
```

THE LIST-STYLE-TYPE PROPERTY

```
ol { list-style-type: lower-roman; }
```

CSS

Possible values:

- i. none : No marker
- ii. disc (default), circle, square
- iii. Decimal: 1, 2, 3, etc.
- iv. decimal-leading-zero: 01, 02, 03, etc.
- v. lower-roman: i, ii, iii, iv, v, etc.
- vi. upper-roman: I, II, III, IV, V, etc.
- vii. lower-alpha: a, b, c, d, e, etc.
- viii. upper-alpha: A, B, C, D, E, etc.
- x. lower-greek: alpha, beta, gamma, etc.
- others: hebrew, armenian, georgian, cjk-ideographic, hiragana...

BODY STYLES

```
body {  
font-size: 16px;  
}
```

CSS

Applies a style to the entire body of your page

Saves you from manually applying a style to each element

CASCADING STYLE SHEETS

Properties of an element *cascade* together in this order:

- browser's default styles
- external style sheet files (in a `<link>` tag)
- internal style sheets (inside a `<style>` tag in the page's header)
- inline style (the style attribute of the HTML element)

INHERITING STYLES

```
body { font-family: sans-serif; background-color: yellow; }  
p { color: red; background-color: aqua; }  
a { text-decoration: underline; }  
h2 { font-weight: bold; text-align: center; }
```

CSS

This is a heading

A styled paragraph. Previous slides are available on the website.

- A bulleted list

output

when multiple styles apply to an element, they are inherited

a more tightly matching rule can override a more general inherited rule

STYLES THAT CONFLICT

```
p, h1, h2 { color: blue; font-style: italic; }  
h2 { color: red; background-color: yellow; }
```

CSS

This paragraph uses the first style above.

This heading uses both styles above.

output

when two styles set conflicting values for the same property, the latter style takes precedence

W3C CSS VALIDATOR

```
<p>  
<a href="http://jigsaw.w3.org/css-  
validator/check/referer">  
</a>  
</p>
```

CSS



output

jigsaw.w3.org/css-validator/

checks your CSS to make sure it meets the official CSS specifications

CSS PROPERTIES FOR BACKGROUNDS

property	description
background-color	color to fill background
background-image	image to place in background
background-position	placement of bg image within element
background-repeat	whether/how bg image should be repeated
background-attachment	whether bg image scrolls with page
background	shorthand to set all background properties

BACKGROUND-IMAGE

```
body {  
background-image: url("images/draft.jpg");  
}
```

CSS

This is the first paragraph

This is the second paragraph...

It occupies 2 lines

background image/color fills the element's content area

BACKGROUND-REPEAT

```
body {  
background-image: url("images/draft.jpg");  
background-repeat: repeat-x;  
}
```

CSS

This is the first paragraph

This is the second paragraph...

It occupies 2 lines

can be repeat (default), repeat-x, repeat-y, or no-repeat

BACKGROUND-POSITION

```
body {  
background-image: url("images/draft.jpg");  
background-repeat: no-repeat;  
;  
} background-position: 370px 20px  
CSS
```

This is the first paragraph

This is the second paragraph...
It occupies 2 lines



value consists of two tokens, each of which can be top, left, right, bottom, center, a percentage, or a length value in px, pt, etc.

value can be negative to shift left/up by a given amount

ASIDE: FAVORITES ICON ("FAVICON")

```
<link href="filename" type="MIME type" rel="shortcut icon" />
```

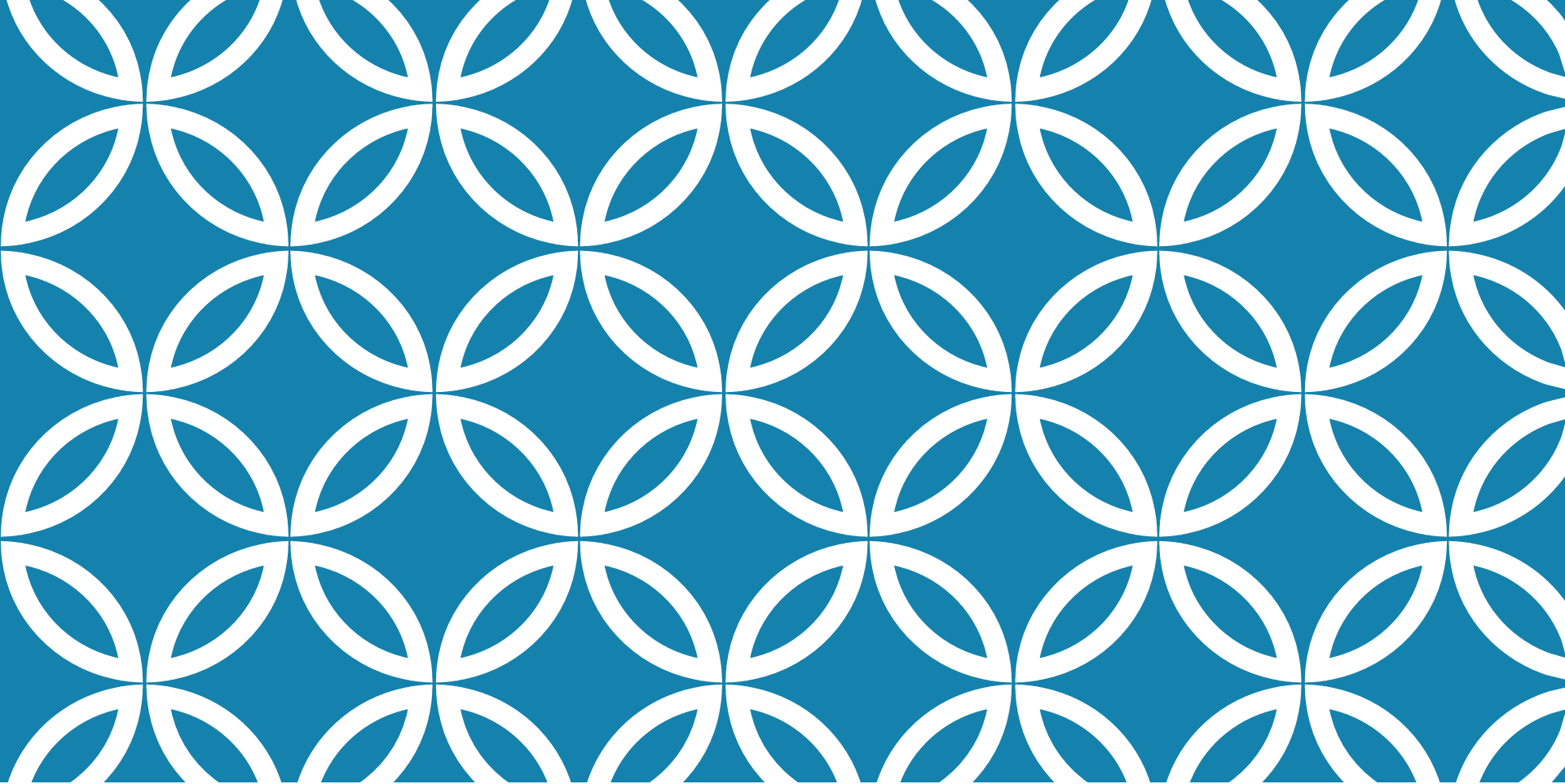
HTML

```
<link href="yahoo.gif" type="image/gif" rel="shortcut icon" />
```

HTML

The link tag, placed in the HTML page's head section, can specify an icon

- this icon will be placed in the browser title bar and bookmark/favorite



POSITIONING



POSITIONING ELEMENTS WITH CSS

With the exception of floating elements right or left, we have thus far allowed our page elements to appear one after another down the screen. Using CSS, we can depart from this **normal flow** in order to:

- force an element to stay fixed in the same place in the browser window, even while the user continues to scroll further down the page.

- move elements up, down, right, or left relative to their normal position on the page.

- remove elements from their normal position and place them in a precise location within the page content.

- allow elements to overlap and to specify which appears in front and which appears in back.

POSITIONING PROPERTIES AND VALUES

The CSS **position** property has four different values we will use:

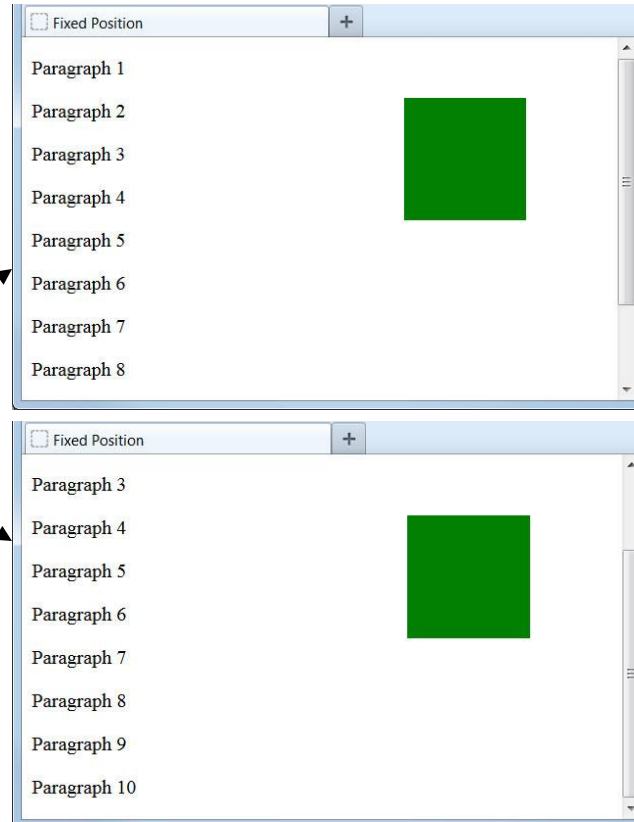
1. **static** - places the element in its usual location in the normal document flow. This is the default value.
2. **fixed** - removes the element from the normal document flow and fixes it to a specific location in the browser window. Even when the page is scrolled, its location will remain the same.
3. **relative** - keeps the element in the normal document flow but allows us to move it around relative to its normal location.
4. **absolute** - removes the element from the normal document flow and allows us to place it in a specific location. It will scroll along with the surrounding page content.

The position property is used in conjunction with the **top**, **bottom**, **right**, and **left** CSS properties. Let's see some examples how these work together.

FIXED POSITION

Let's use the position property to fix an element to a specific location in the browser window:

```
.box1 {  
  height: 100px;  
  width: 100px;  
  background-color: green;  
  position: fixed;  
  top: 50px;  
  right: 75px;  
}  
...  
<div class="box1"></div>  
<p>Paragraph 1</p>  
<p>Paragraph 2</p>  
...
```

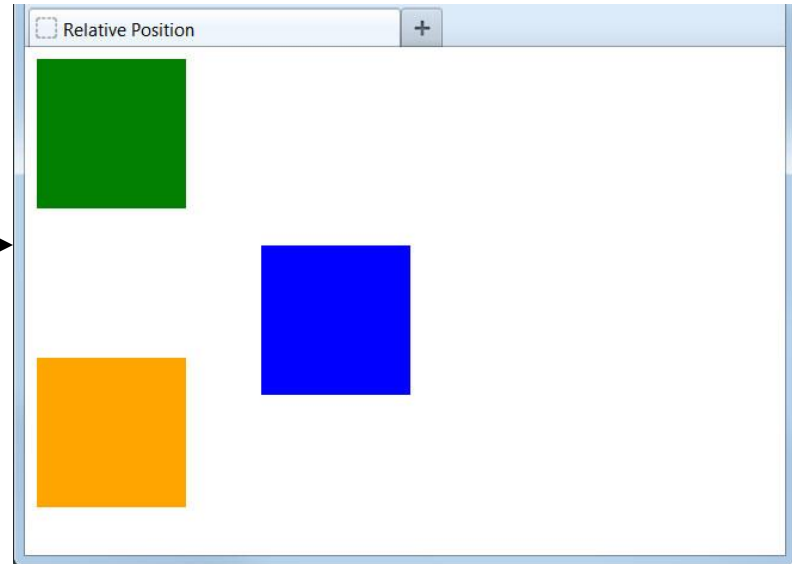


We have pinned this element to a fixed location, 50px from the top edge and 75px from the right edge of the browser window. Even as we scroll the page content, the element remains in the same place.

RELATIVE POSITION

Now let's move an element *relative* to its normal position:

```
.box1 {  
  height: 100px;  
  width: 100px;  
  background-color: green;  
}  
.box2 {  
  height: 100px;  
  width: 100px;  
  background-color: blue;  
  position: relative;  
  top: 25px;  
  left: 150px;  
}  
.box3 {  
  height: 100px;  
  width: 100px;  
  background-color: orange;  
}  
...  
<div class="box1"></div>  
<div class="box2"></div>  
<div class="box3"></div>
```



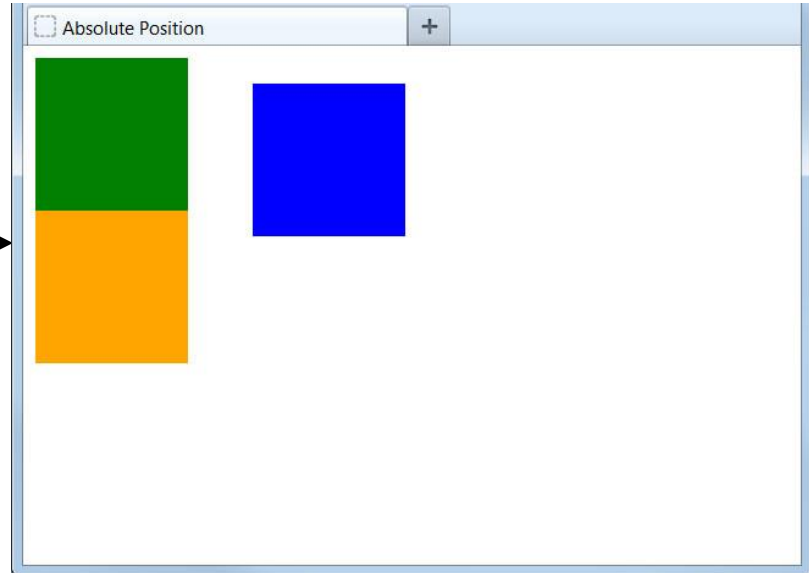
Using relative position, we can nudge elements or even move them to the other side of the page.

Remember that relative positioning keeps the element in the normal document flow. No matter where we move the element, its original location will be reserved on the screen.

ABSOLUTE POSITION

Here we will change the element to have an *absolute* position:

```
.box1 {  
  height: 100px;  
  width: 100px;  
  background-color: green;  
}  
.box2 {  
  height: 100px;  
  width: 100px;  
  background-color: blue;  
  position: absolute;  
  top: 25px;  
  left: 150px;  
}  
.box3 {  
  height: 100px;  
  width: 100px;  
  background-color: orange;  
}  
...  
<div class="box1"></div>  
<div class="box2"></div>  
<div class="box3"></div>
```



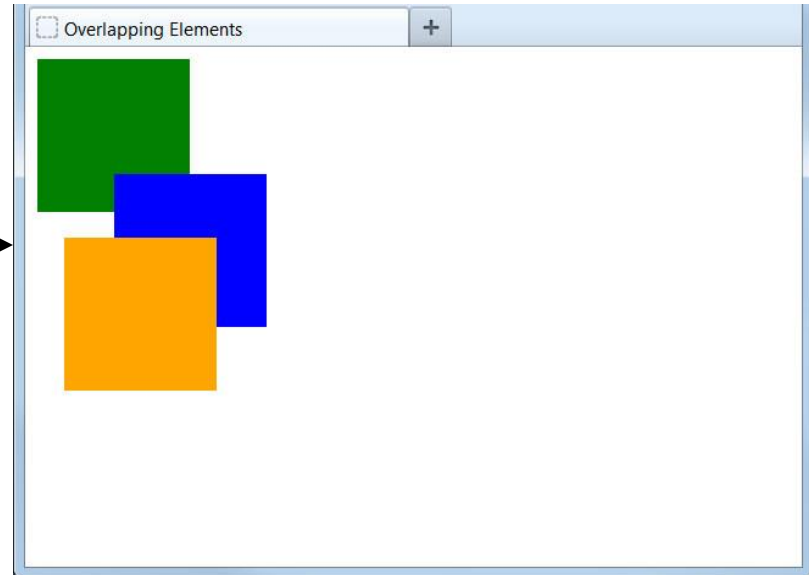
Absolute positioning places the element not relative to its normal position but from the top left corner of the page.

Like fixed positioning, absolute positioning removes the element from the normal document flow. Other elements will appear on the page the same as if this element did not exist.

OVERLAPPING ELEMENTS

Positioning elements can cause them to overlap on the page:

```
.box1 {  
  height: 100px;  
  width: 100px;  
  background-color: green;  
}  
.box2 {  
  height: 100px;  
  width: 100px;  
  background-color: blue;  
  position: relative;  
  top: -25px;  
  left: 50px;  
}  
.box3 {  
  height: 100px;  
  width: 100px;  
  background-color: orange;  
  position: absolute;  
  top: 125px;  
  left: 25px;  
}  
...
```



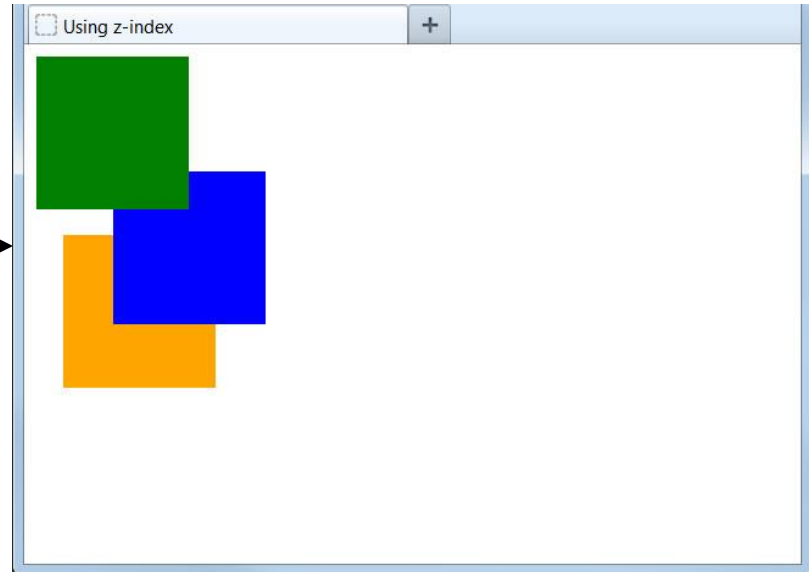
By default, the element last added to the page will appear on top of all others and the first element will appear below all others.

Note that we can set position values to be negative numbers. This can even be used to pull elements partially off the visible page.

CONTROLLING THE OVERLAP

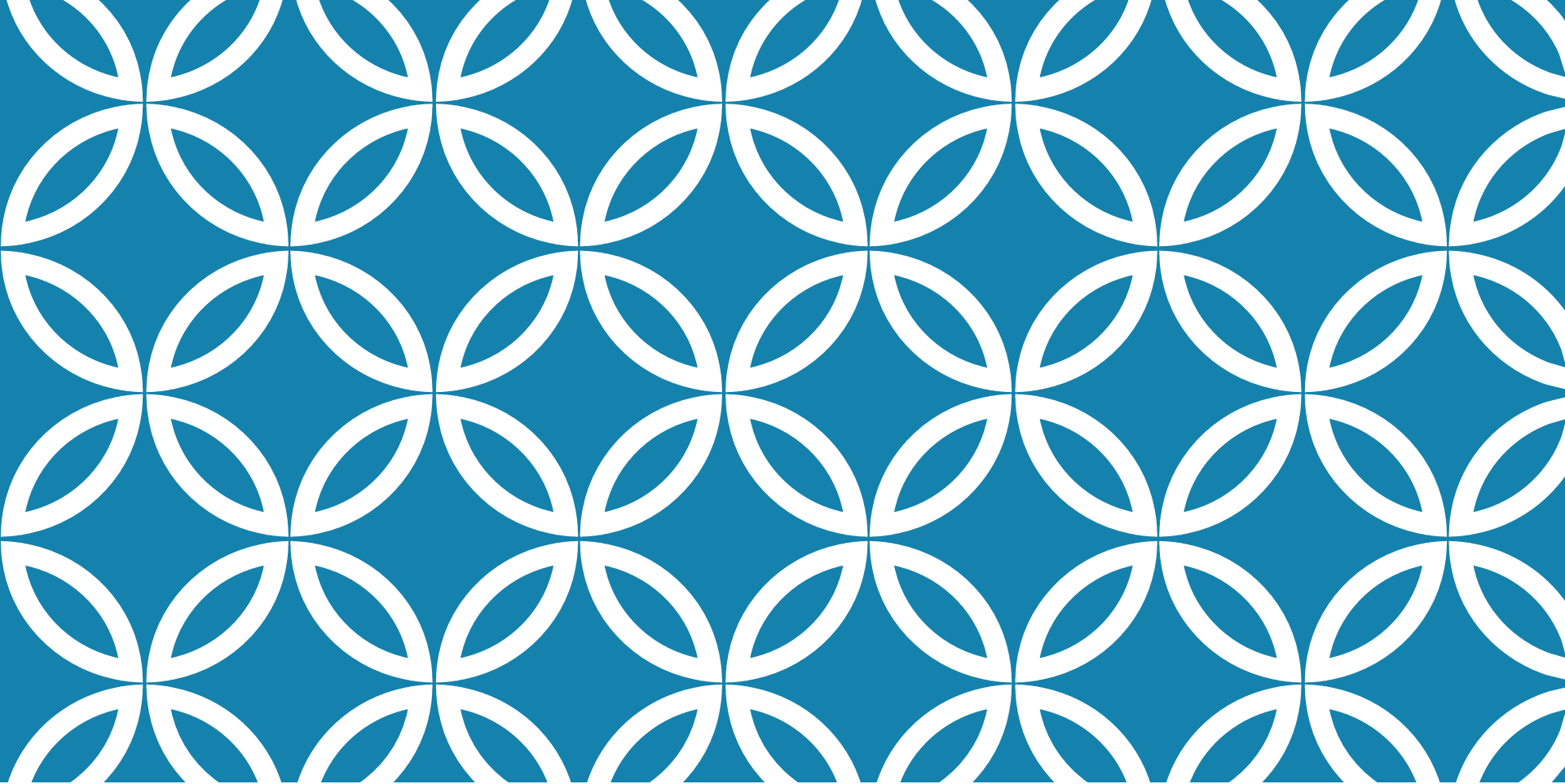
We can override the default precedence by setting the **z-index** property:

```
.box1 {  
  height: 100px;  
  width: 100px;  
  background-color: green;  
  position: relative;  
  z-index: 3;  
}  
.box2 {  
  ...  
  position: relative;  
  top: -25px;  
  left: 50px;  
  z-index: 2;  
}  
.box3 {  
  ...  
  position: absolute;  
  top: 125px;  
  left: 25px;  
  z-index: 1;  
}  
...
```



Higher z-index values will always appear on top of lower ones, so we can precisely control how elements stack on the page.

For the z-index to work, an element must be positioned as relative, absolute, or static.



CSS BOX MODEL



What is the Box Model?

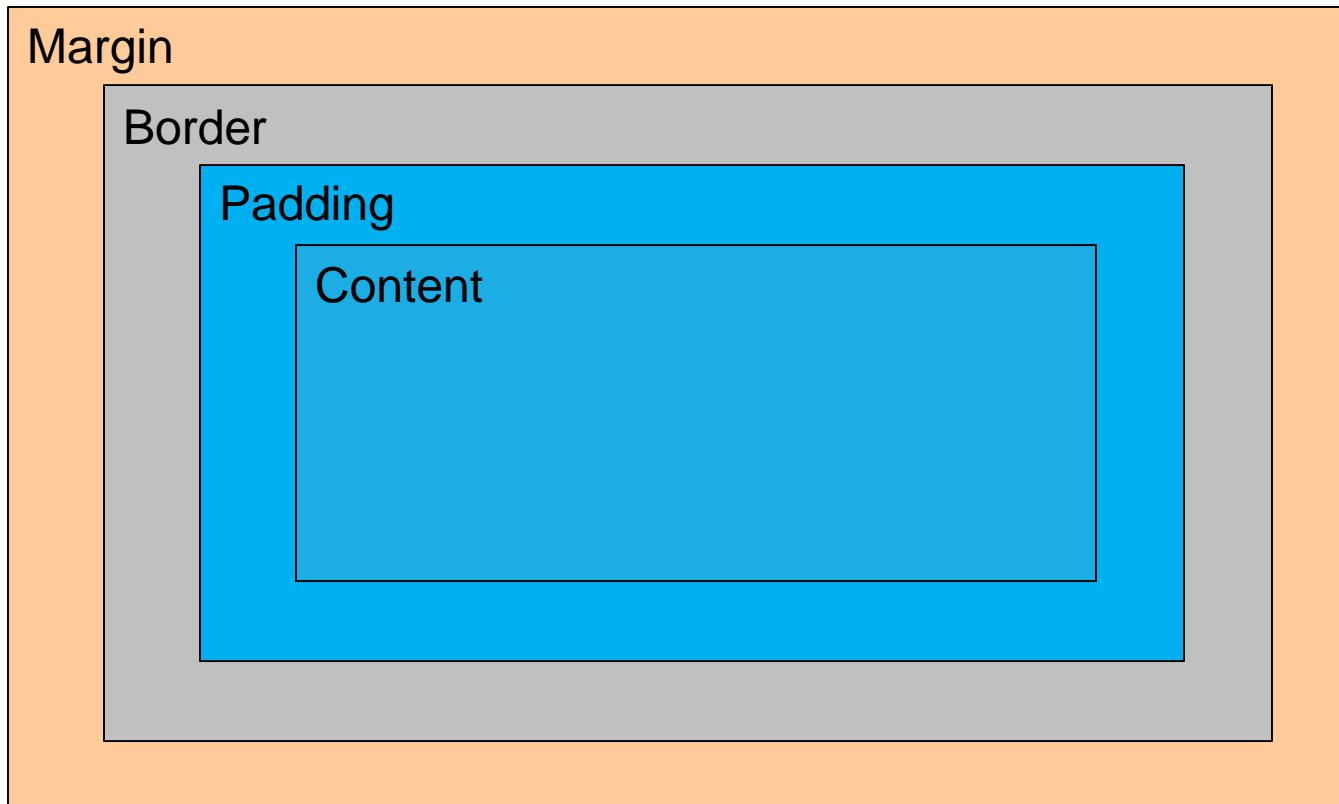
The box model is a tool we use to understand how our content will be displayed on a web page.

- Each HTML element appearing on our page takes up a "box" or "container" of space.
- Each box size is affected not only by content but also by padding, borders, and margins.
- By knowing how to calculate the dimensions of each box, we can accurately predict how elements will lay out on the screen.
- As we build a new page, we can arrange these boxes on the screen, creating a balanced layout with white space around the content.

The importance of the box model concept cannot be overemphasized. It would be difficult and frustrating to create a website without understanding this concept.

Box Components

Each element on the page has the following components:



The easiest way to understand these components is to use one of the most versatile tools available to us as web designers: the `<div>` element.

Introducing the `<div>` Element

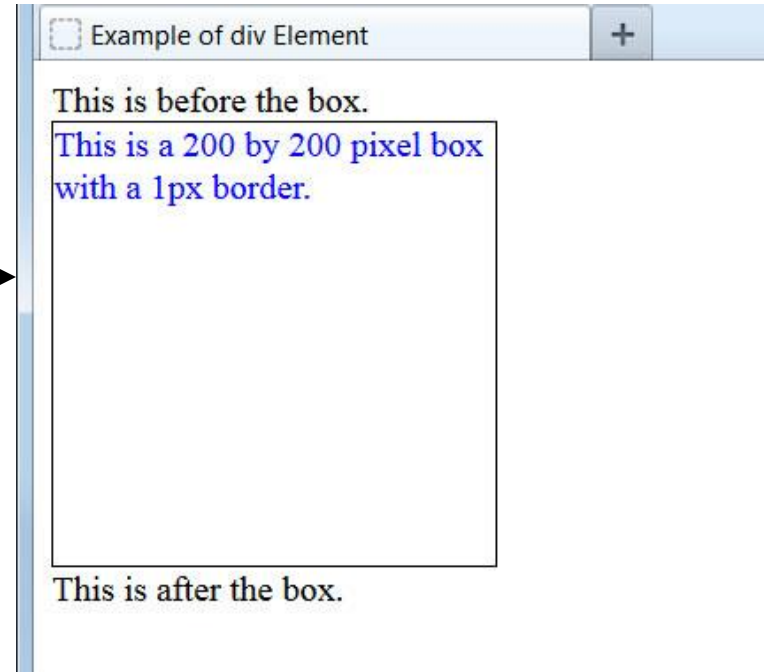
The `<div>` ("division") element groups other elements on the screen.

- By setting the **width** and **height** attributes via CSS, we can reserve a precise amount of space on our page for specific content.
- The actual content is nested and contained within the opening `<div>` and closing `</div>` tags.
- When we apply CSS styling directly to the `<div>` element, all the elements contained within that `<div>` will inherit that style.
- By using multiple `<div>` elements as building blocks, we can design an entire web page layout.

Example <div> Element

Let's create a box on the screen and add a border around it:

```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    border: 1px solid black;
    color: blue;
  }
</style>
...
This is before the box.
  <div class="box200">
This is a 200 by 200 pixel box with
  a 1px border.
  </div>
This is after the box.
...
```



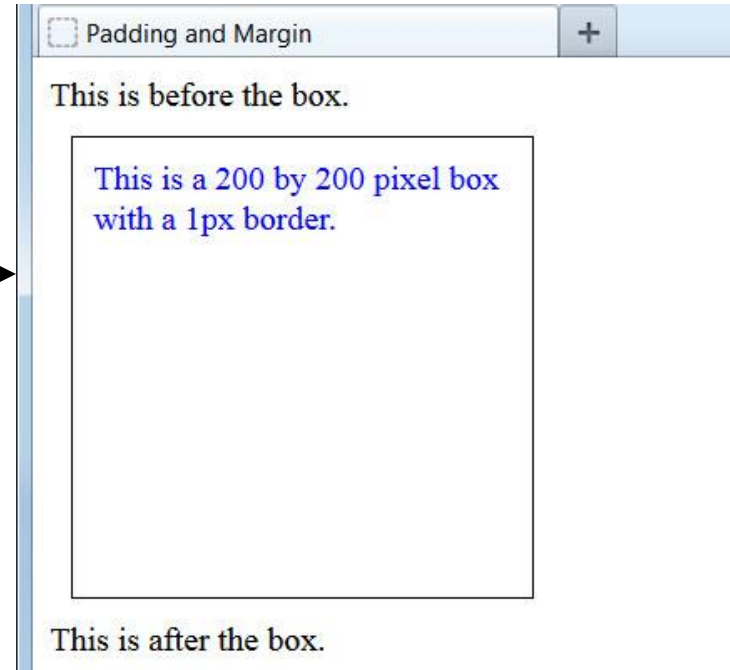
With the border set, we can see the exact space taken up on the page.

Notice that there is almost no space separating the text from the box border. Elements such as paragraphs, headers, and lists automatically insert padding and margins, but plain text does not do so.

Adding Padding and Margin

Let's create some space between elements by adding both padding and margin:

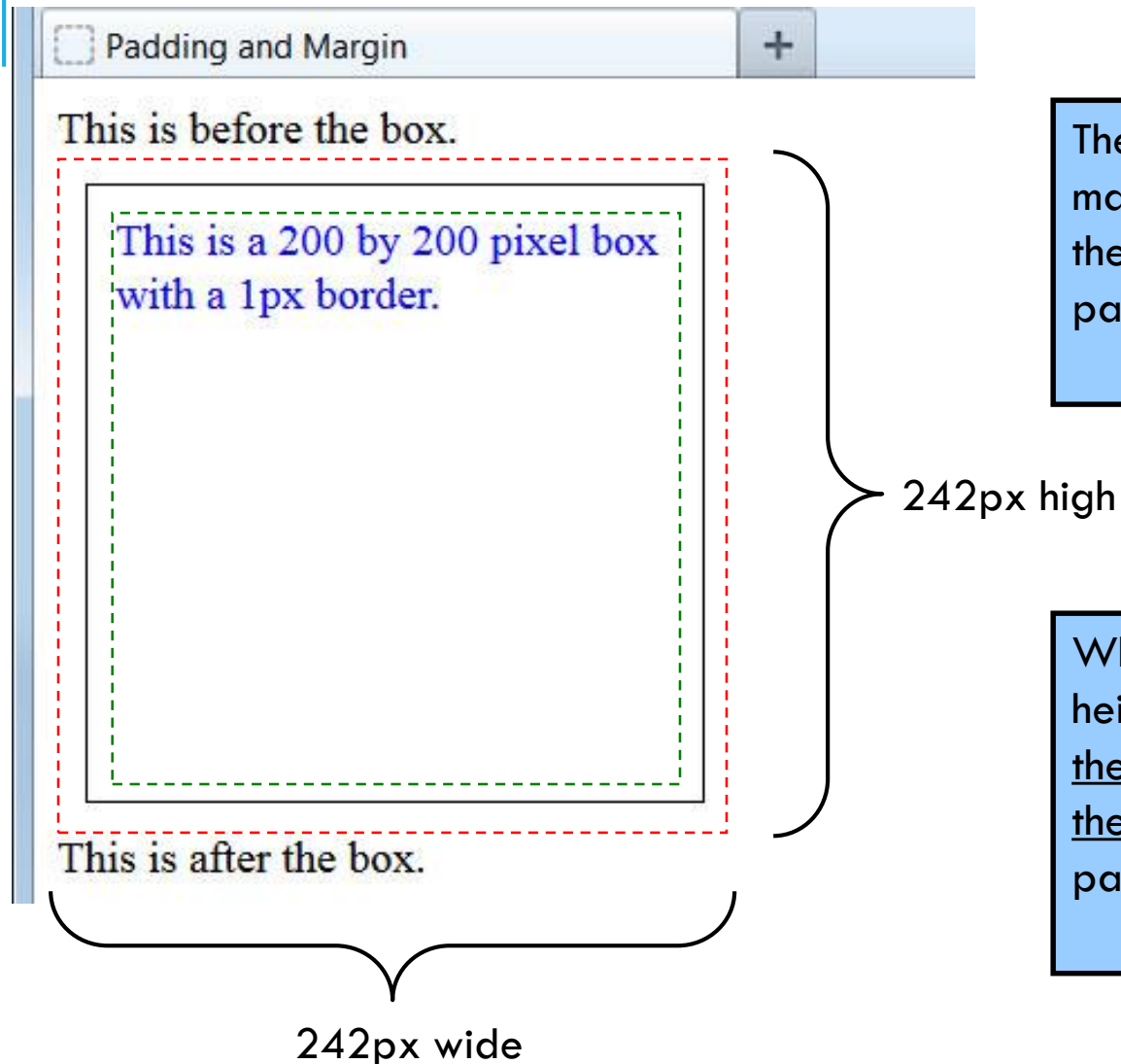
```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    border: 1px solid black;
    color: blue;
    padding: 10px;
    margin: 10px;
  }
</style>
...
```



The 10 pixel padding adds buffer space, on all four sides, between the content and border. The 10 pixel margin adds buffer space, on all four sides, between the border and surrounding elements.

Let's examine this a bit closer up to see exactly what is happening.

Padding and Margin Illustrated



The dotted red line shows the margin's outer boundary and the dotted green line shows the padding's inner boundary.

When we define the width and height of a `<div>` element, these dimensions apply only to the actual content, not to the padding, border, or margin.

Calculating Total Dimensions

When we are planning our page, we have to calculate exactly how much screen space a `<div>` or any other element will use. The formula is:

- Total element width = defined width + left padding + right padding + left border + right border + left margin + right margin.
- Total element height = defined height + top padding + bottom padding + top border + bottom border + top margin + bottom margin.

In our previous example:

- Total `<div>` width = 200px + 10px + 10px + 1px + 1px + 10px + 10px = 242px.
- Total `<div>` height = 200px + 10px + 10px + 1px + 1px + 10px + 10px = 242px.

Padding, borders, and margins do not have to be the same on all four sides, as they are in this example. Let's see how to set these individually.

Setting Individual Margins

We can set the specific margin sizes by using these four properties:

margin-top:

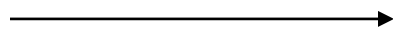
margin-right:

margin-bottom:

margin-left:

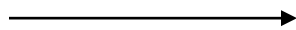
In practice, few web designers use these properties, preferring a **shorthand** method that condenses the declaration to a single line:

margin: 10px;



Sets all four margins to be 10px.

margin: 10px 5px;



Sets the top and bottom margins as 10px and the left and right margins as 5px.

margin: 20px 10px 5px 15px;



Sets the top margin as 20px, the right margin as 10px, the bottom margin as 5px, and the left margin as 15px.

Setting Individual Padding

We can set specific padding sizes by using these four properties:

padding-top:

padding-right:

padding-bottom:

padding-left:

Again, these are rarely used. Instead, the same shorthand method is employed:

padding: 5px; →

Sets padding as 5px on all four sides.

padding: 25px 5px; →

Sets top and bottom padding as 25px and left and right padding as 5px.

padding: 50px 10px 15px 5px; →

Sets top padding as 50px, right padding as 10px, bottom padding as 15px, and the left padding as 5px.

Setting Individual Borders

Customizing borders is more involved. Since they are visible on the page, we have to specify style, thickness, and color. Accordingly, there are more properties available:

<code>border-top-width:</code>	<code>border-top-style:</code>	<code>border-top-color:</code>
<code>border-right-width:</code>	<code>border-right-style:</code>	<code>border-right-color:</code>
<code>border-bottom-width:</code>	<code>border-bottom-style:</code>	<code>border-bottom-color:</code>
<code>border-left-width:</code>	<code>border-left-style:</code>	<code>border-left-color:</code>

To make things easier, we can use the shorthand method as before:

```
border-width: 10px;  
border-style: solid dashed;  
border-color: blue red orange gray;
```

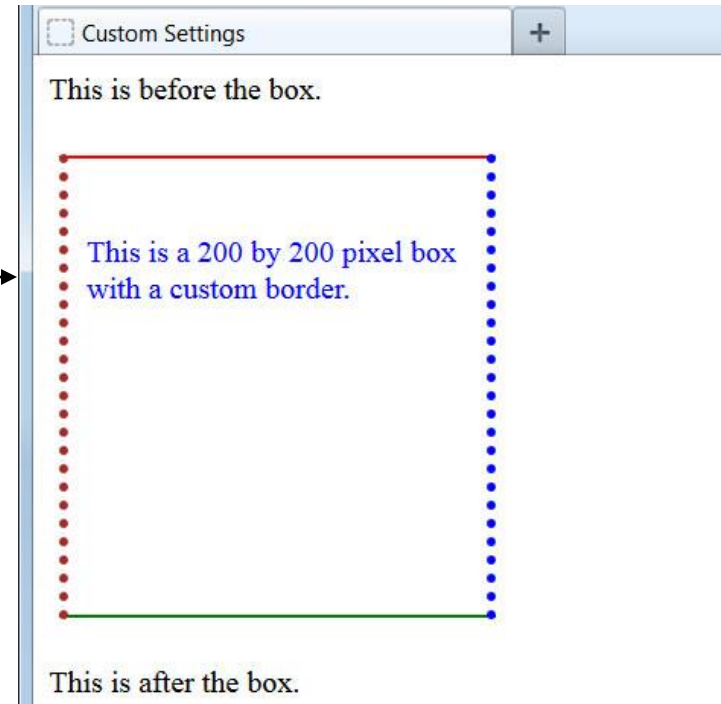
If the three border properties will be identical on all four sides, we can use a single-line border shorthand, as we did in an earlier lesson:

```
border: 5px solid blue;
```

Example of Customized Settings

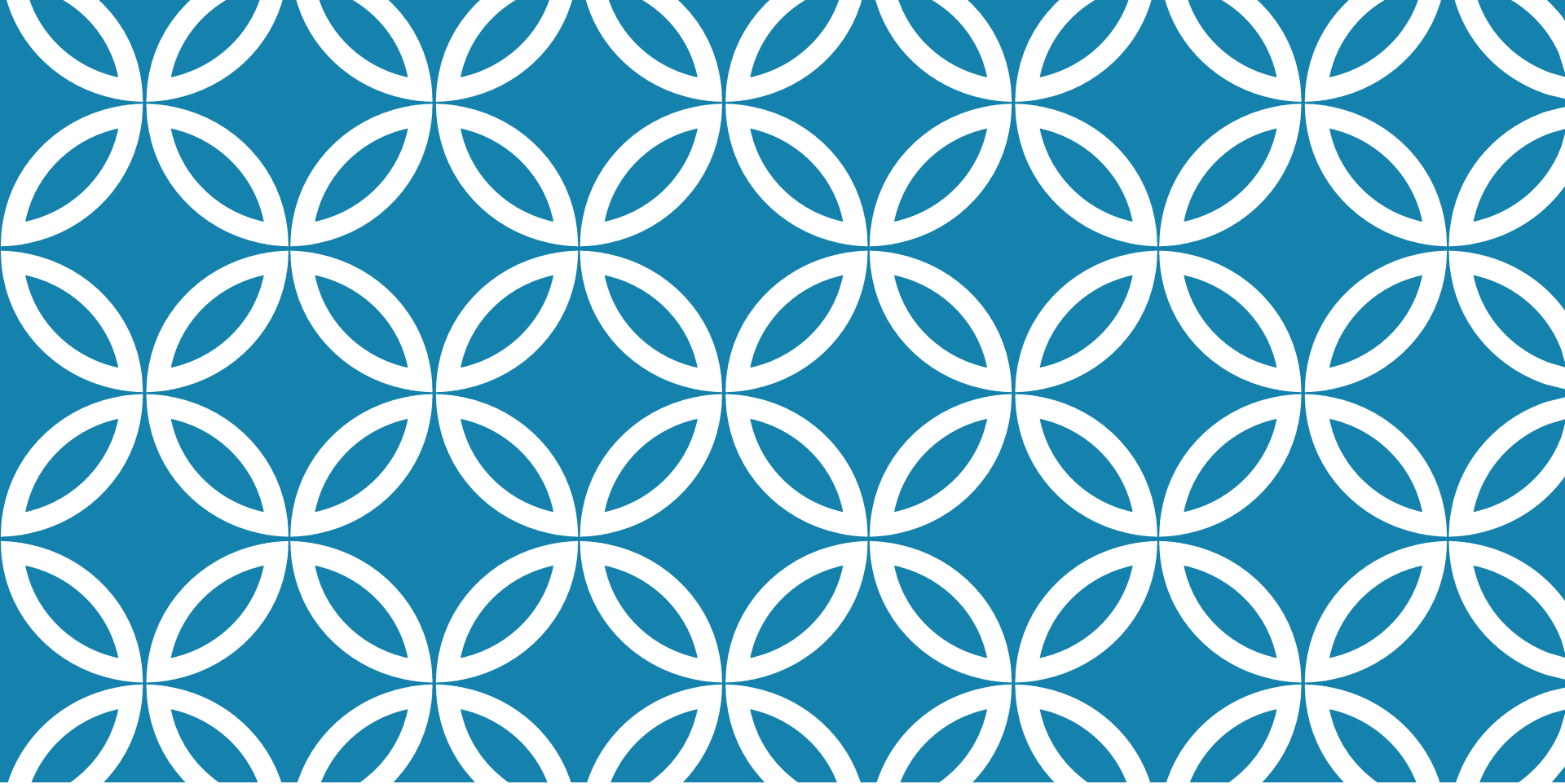
Here we have set custom padding, borders, and margins:

```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    color: blue;
    padding: 40px 10px 0px 10px;
    margin: 25px 5px;
    border-width: 2px 5px;
    border-style: solid dotted;
    border-color: red blue green maroon;
  }
</style>
```



A helpful way to remember the order of shorthand settings is that it starts at noon and goes clockwise around: top, right, bottom, left.





CSS TABLE STYLING



Using CSS to Style Tables

Up to this point, as we haven't applied any CSS styling to our tables, our example tables have not been too pleasing to the eye. CSS offers us several properties to customize how our tables appear on the page:

Style	Description
width	Width of element.
background-color	Background color of element.
color	Color of text in element.
text-align	Horizontal text alignment of element.
border	Border thickness and style of element.
padding	Padding (white space around content) of element.

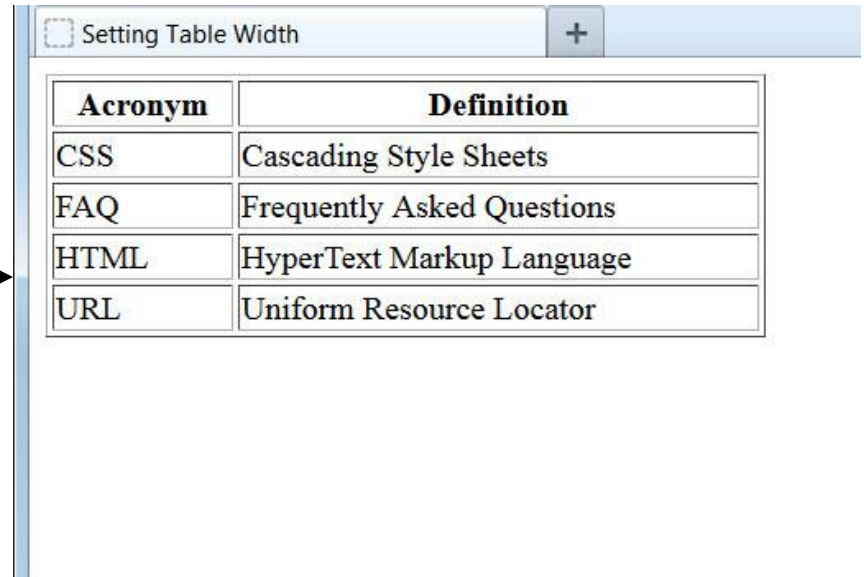
We're already familiar with the **color** and **text-align** properties, but let's see examples of the other styles in action.

Though we're using these properties to style table elements, all of these properties may be used with many other XHTML elements, as we'll see in future lessons.

Setting Table Width

We can set the overall width of a table by applying a class to the `<table>` element:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th>Acronym</th>
    <th>Definition</th>
  </tr>
  <tr>
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
  </tr>
  ...
</table>
```



Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

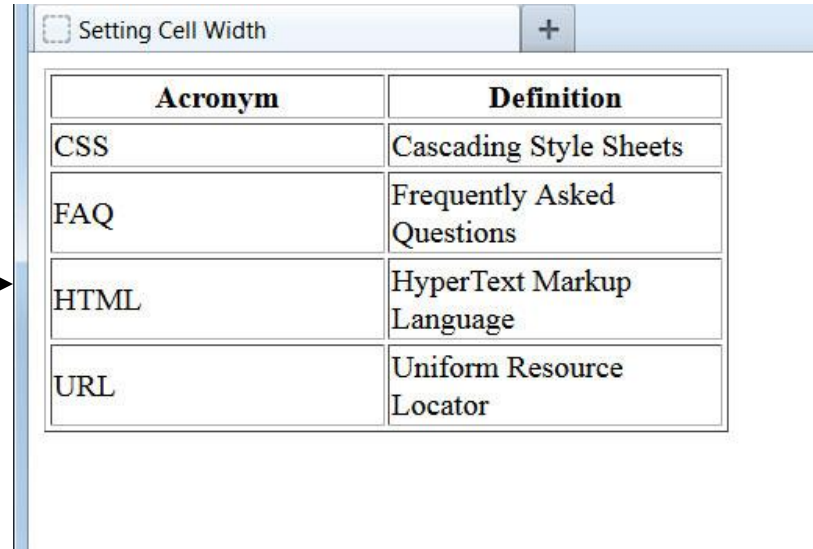
The table is now 350 pixels wide, creating some extra space in the data cells.

The width of the columns is automatically established by the browser to accommodate the cell contents, but we don't have to accept this format.

Setting Column Width in Pixels

We can set the width of columns by setting the widths of the first row of data cells:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
  .col {
    width: 175px;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th class="col">Acronym</th>
    <th class="col">Definition</th>
  </tr>
  <tr>
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
  </tr>
  ...
</table>
```



Setting Cell Width +

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

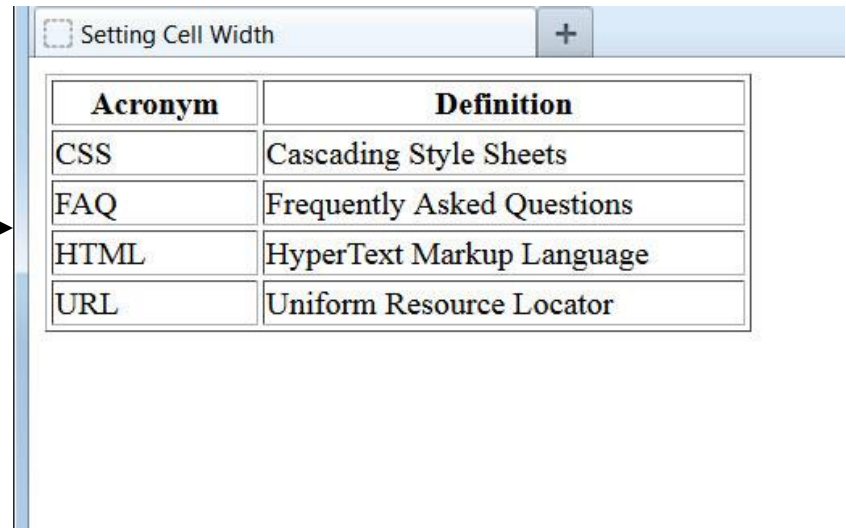
We've now set each of the columns to be 175 pixels in width, or half of the total available table width.

Notice that due to the enforced width, some cell contents had to wrap to an additional line.

Setting Column Width in Percent

Rather than pixels, we can also set column widths to be a percent of the total available width:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
  .col1 {
    width: 30%;
  }
  .col2 {
    width: 70%;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th class="col1">Acronym</th>
    <th class="col2">Definition</th>
  </tr>
  ...
</table>
```



Setting Cell Width

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Setting column widths by percent instead of pixels has the advantage of flexibility. Should we ever alter the width of the table itself, the columns will automatically adjust.

Borders

CSS provides us plenty of flexibility when generating borders around elements. There are multiple properties that affect the border that displays on the page:

Style	Description
border-style	Type of border: none, solid, dashed, dotted, double, groove, ridge, inset, outset.
border-color	Color of border.
border-width	Width of border, measured in pixels. Also available: thin, medium, and thick.
border-collapse	collapse: borders display as single border. separate: borders are detached (default).

We'll examine each of the first three, but first let's take a look at the **border-collapse** property.

The border-collapse Property

By invoking the **border-collapse** property, we can force a table to collapse its borders into a single line between cells:

```
<style type="text/css">
.glossary {
  width: 350px;
}
.collapse {
  border-collapse: collapse;
}
...
</style>
...
<table class="glossary collapse"
  border="1">
  <tr>
    <th class="col1">Acronym</th>
    <th class="col2">Definition</th>
  </tr>
  ...
</table>
```

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

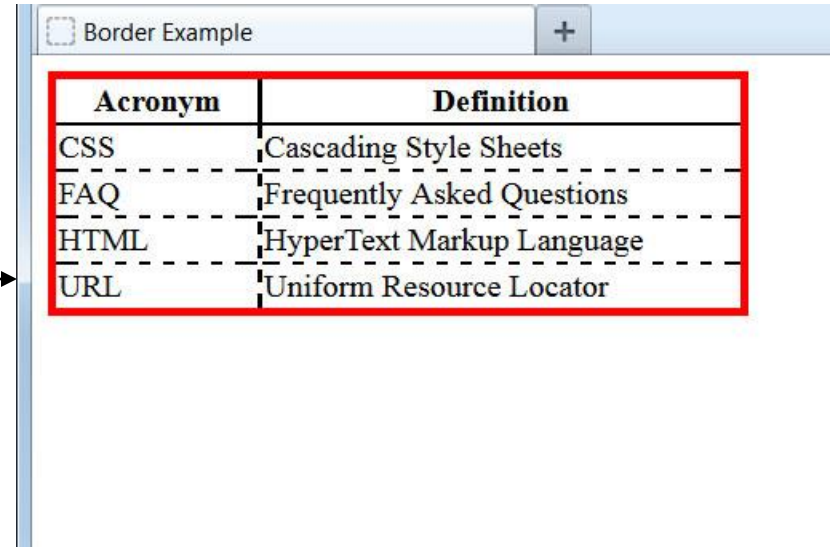
Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Here we created an identical table but added a class that included the **border-collapse** property.

Customizing Borders

By using the other three properties, we can create custom borders for our table:

```
<style type="text/css">
.glossary {
  width: 350px;
  border-width: 4px;
  border-style: solid;
  border-color: red;
}
th {
  border-width: 2px;
  border-style: solid;
}
td {
  border-width: 2px;
  border-style: dashed;
}
...
```



Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Notice that we can apply different styles to the table, table headers, and table cell elements.

For demonstrative purposes, we applied CSS styles directly to the `<th>` and `<td>` elements, instead of using separate classes. In the real world, this would likely be a bad idea, as any other tables on our page would be affected too.

Using CSS Border Shorthand

When specifying multiple border properties, we can use the CSS **border shorthand** to reduce the statement to a single line:

```
<style type="text/css">
.glossary {
  width: 350px;
  border-width: 4px;
  border-style: solid;
  border-color: red;
}
th {
  border-width: 2px;
  border-style: solid;
}
td {
  border-width: 2px;
  border-style: dashed;
}
...
```



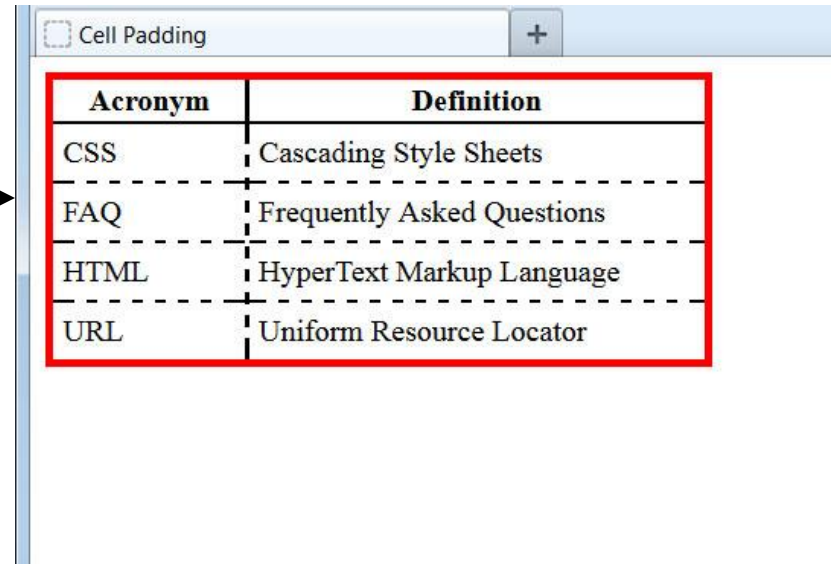
```
<style type="text/css">
.glossary {
  width: 350px;
  border: 4px solid red;
}
th {
  border: 2px solid;
}
td {
  border: 2px dashed;
}
...
```

By convention, the properties are ordered as **border: border-width border-style border-color;** (The color portion may be omitted.)

Adding Cell Padding

By setting the **padding** property, we can make sure there is at least that much white space around our cell contents. This keeps the actual cell contents from displaying too closely to the borders:

```
<style type="text/css">
.glossary {
  width: 350px;
  border: 4px solid red;
}
th {
  border: 2px solid;
}
td {
  border: 2px dashed;
  padding: 5px;
}
...
```



Cell Padding

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Different border and padding settings can be set for the top, bottom, left, and right sides of elements. We will learn how to do this in an upcoming lesson.

Setting a Background Color

By setting the **background-color** property, we can change our table's background away from the default:

```
<style type="text/css">
.glossary {
  width: 350px;
  border: 4px solid red;
  background-color: gray;
}
th {
  border: 2px solid;
}
td {
  border: 2px dashed;
  padding: 5px;
}
...
```

Background Color

+

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Let's use this property now on the data cells to make our table a bit more readable.

Using Background Color on Rows

```
<style type="text/css">
...
.odd {
    background-color: lime;
}
.even {
    background-color: aqua;
}
...
</style>
...
<tr class="odd">
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
</tr>
<tr class="even">
    <td>FAQ</td>
    <td>Frequently Asked Questions</td>
</tr>
<tr class="odd">
...

```

☐ Alternating Row Colors +

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

This is a handy way to make our table rows show in alternating background colors. This technique makes reading wide tables far easier for viewers.