

CSS: INTRODUCTION

Face to face with the Head TA

- Harshani Nagahamulla

harshani.nagahamulla@ucdconnect.ie

Exchange Room – 1

Mondays -> 13:30 – 15:30

Book appointment by Friday, 5:00 pm

Before we leave HTML

- <https://developer.mozilla.org/en-US/docs/Learn/HTML>

WHAT IS CSS?

What is CSS?

You be styling soon

- CSS is a W3C standard for describing the **presentation (or appearance)** of HTML elements.
- With CSS, we can assign
 - font properties,
 - colors,
 - sizes,
 - borders,
 - background images,
 - even the position of elements.

Benefits of CSS

Why using CSS is a better way of describing presentation than HTML

- The degree of formatting control in CSS is significantly better than that provided in HTML.
- Web sites become significantly more maintainable because all formatting can be centralized into one, or a small handful, of CSS files.
- CSS-driven sites are more accessible.
- A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain less markup.
- CSS can be used to adopt a page for different output mediums.

External Style Sheet

Style rules placed within a external text file with the .css extension

```
<html>
<head>
  <link rel="stylesheet" href="css/mystyle.css" />
  <title>HTML with Stylesheet</title>
</head>
```

- This is by far the most common place to locate style rules because it provides the best maintainability.
 - When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.

Internal Stylesheet

```
<html>
  <head>
    <title>My CSS experiment</title>
    <style>
      h1 {
        color: blue;
        background-color: yellow;
        border: 1px solid black;
      }

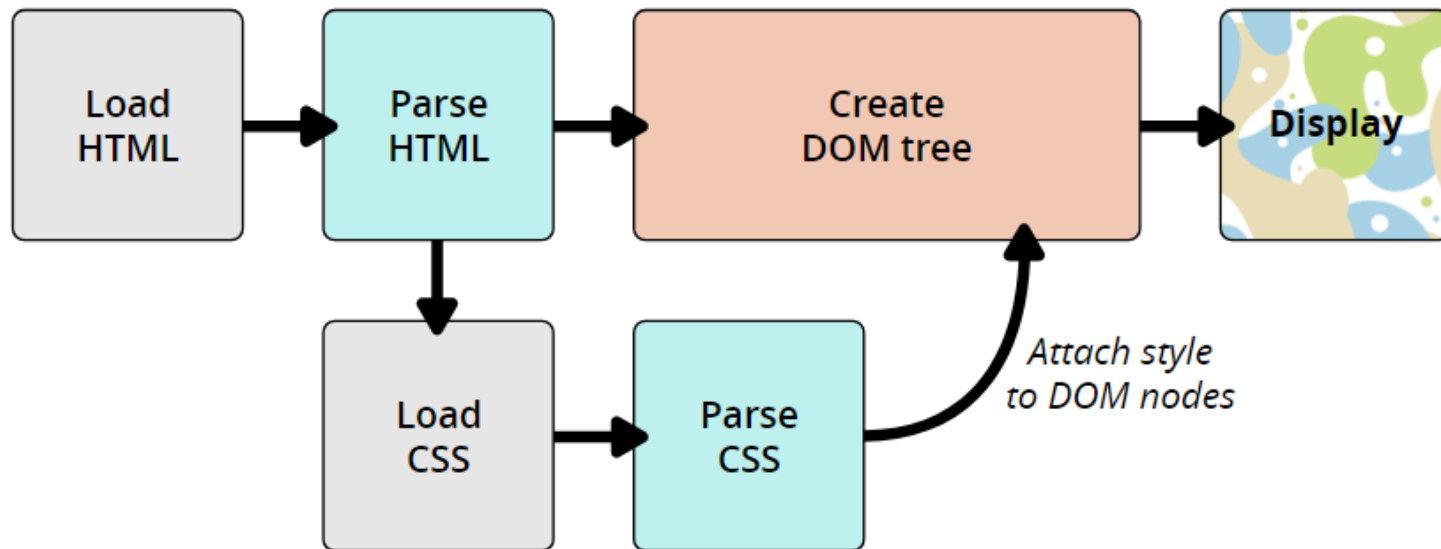
      p {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```


Inline Stylesheet

```
<html>
  <head>
    <title>My CSS experiment</title>
  </head>
  <body>
    <h1 style="color: blue;background-color: yellow;border: 1px solid black;">
      Hello World!</h1>
    <p style="color:red;">This is my first CSS example</p>
  </body>
</html>
```

How CSS works

- The browser converts HTML and CSS into the *Document Object Model* (DOM)
- The browser displays the content of the DOM



What is the DOM

```
<p>
  Let's use:
  <span>Cascading</span>
  <span>Style</span>
  <span>Sheets</span>
</p>
```



```
P
├ "Let's use:"
├ SPAN
│   └ "Cascading"
├ SPAN
│   └ "Style"
└ SPAN
    └ "Sheets"
```



Let's use: Cascading Style Sheets

Now apply CSS

```
span {  
  border: 1px solid black;  
  background-color: lime;  
}
```

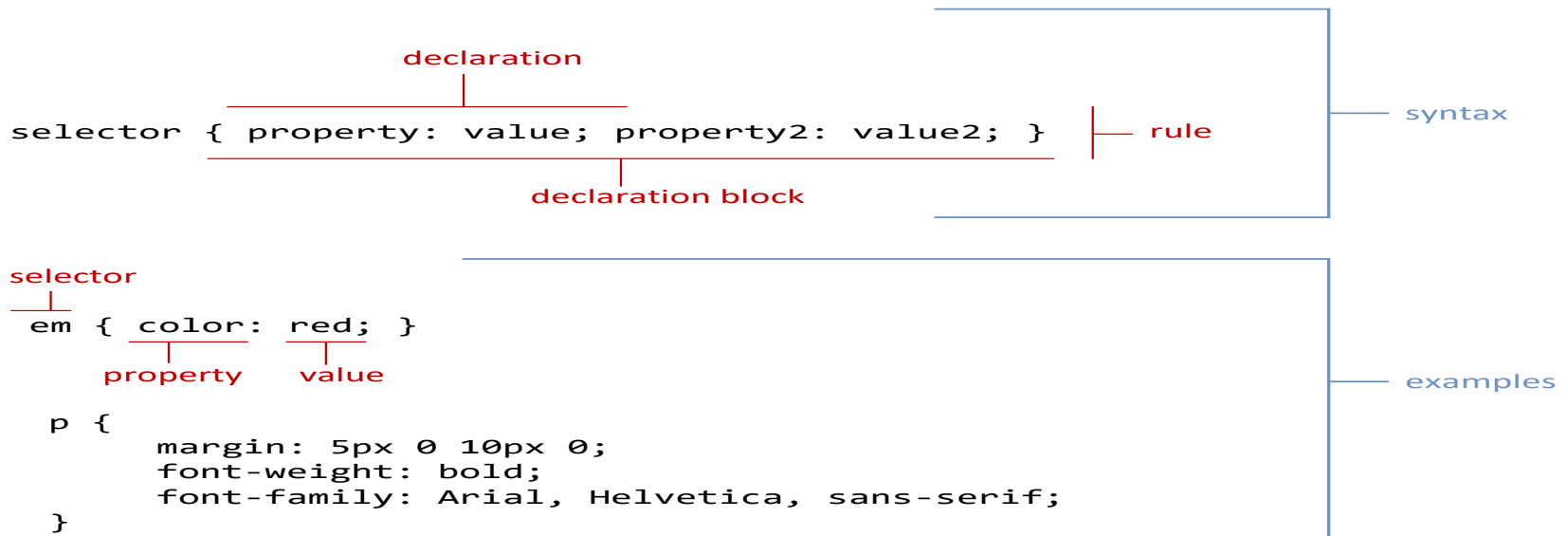


Let's use: **Cascading Style Sheets**

CSS Syntax

Rules, properties, values, declarations

- A CSS document consists of one or more **style rules**.
- A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).

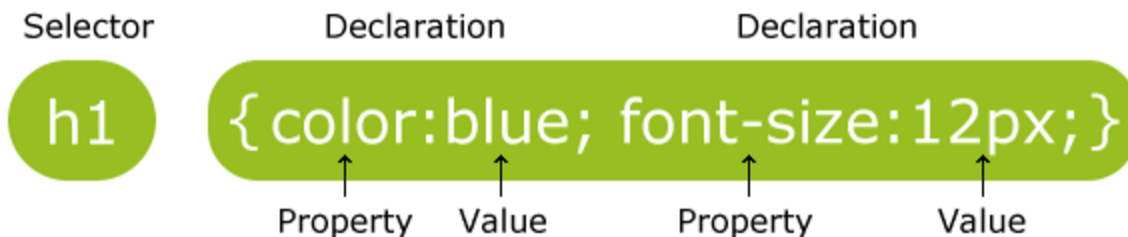


Selectors

Which elements

- Every CSS rule begins with a **selector**.
- The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive the style.



CSS Colours

- CSS colours are most often specified by
 - A common (valid) colour name like “red”, “green”
 - An RGB value like “rgb(255, 125, 42)”
 - A hex value like “#FF7D2A”
- Colour names are not case-sensitive
 - “green” is the same as “Green” or “gREen”
- How do you do “grey”? (hint: same light source for each)

CSS Backgrounds

- Background properties are used to define background effects for elements
- Available properties:
 - `background-color`
 - `background-image`
 - `background-repeat`
 - `background-attachment`
 - `background-position`

Example

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

- You can have different background-color for different elements

```
h1 {  
    background-color: green;  
}
```

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

Background-Image

- Specifies an image to be used as background of an element

```
body {  
    background-image: url("gradient_bg.png");  
}
```

- By default, CSS will repeat the image to fill up the background

```
body {  
    background-image: url("gradient_bg.png");  
    background-repeat: repeat-x;  
}
```

Background-Image

- Position

```
background-position: right top;
```

- Fixed position

```
background-attachment: fixed;
```

- Background-shorthand!

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

Borders

Box Model Property #2

- Borders provide a way to visually separate elements.
- You can put borders around all four sides of an element, or just one, two, or three of the sides.

Borders

Property	Description
border	<p>A combined short-hand property that allows you to set the style, width, and color of a border in one property. The order is important and must be:</p> <p>border-style border-width border-color</p>
border-style	<p>Specifies the line type of the border. Possible values are: solid, dotted, dashed, double, groove, ridge, inset, and outset.</p>
border-width	<p>The width of the border in a unit (but not percents). A variety of keywords (thin, medium, etc) are also supported.</p>
border-color	<p>The color of the border in a color unit.</p>
border-radius	<p>The radius of a rounded corner.</p>
border-image	<p>The URL of an image to use as a border.</p>

Borders

- The `border` property allows you to set
 - Style – defines what kinds of border to display
 - Width – specifies width of four borders
 - Color – set the colour of the borders

I have borders on all sides.

I have a red bottom border

I have rounded borders.

I have a blue left border.

border-style

`dotted` - Defines a dotted border

`dashed` - Defines a dashed border

`solid` - Defines a solid border

`double` - Defines a double border

`groove` - Defines a 3D grooved border. The effect depends on the border-color value

`ridge` - Defines a 3D ridged border. The effect depends on the border-color value

`inset` - Defines a 3D inset border. The effect depends on the border-color value

`outset` - Defines a 3D outset border. The effect depends on the border-color value

`none` - Defines no border

`hidden` - Defines a hidden border

- Can have upto four values (top right bottom left)

```
p.mix {border-style: dotted dashed solid double;}
```

More about borders

- `border-width`
 - Specifies the width of the borders – top right bottom left
- `border-color`
 - Sets the colours of the four borders – top right bottom left
- Individual sides can also be specified

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```


More about borders - II

- If a border-style has four values

border-style: dotted solid double dashed;

- top border is dotted
- right border is solid
- bottom border is double
- left border is dashed

- If a border-style has three values

border-style: dotted solid double;

- top border is dotted
- right and left borders are solid
- bottom border is double

- What happens if a border style has only two values?

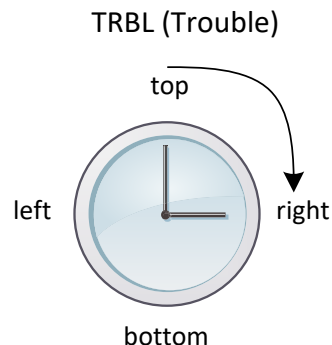
Shortcut notation

TRBL

- With border, margin, and padding properties, there are long-form and shortcut methods to set the 4 sides

```
border-top-color: red;           /* sets just the top side */  
border-right-color: green;       /* sets just the right side */  
border-bottom-color: yellow;     /* sets just the bottom side */  
border-left-color: blue;        /* sets just the left side */  
  
border-color: red;              /* sets all four sides to red */  
  
border-color: red green orange blue; /* sets all four sides differently */
```

When using this multiple values shortcut, they are applied in clockwise order starting at the top.
Thus the order is: **top right bottom left**.



```
border-color: top right bottom left;
```

```
border-color: red green orange blue;
```

Units of Measurement

There are multiple ways of specifying a unit of measurement in CSS

- Some of these are **relative units**, in that they are based on the value of something else, such as the size of a parent element.
- Others are **absolute units**, in that they have a real-world size.
 - Unless you are defining a style sheet for printing, it is recommended to avoid using absolute units.
 - Pixels are perhaps the one popular exception (though as we shall see later there are also good reasons for avoiding the pixel unit).

Relative Units

Unit	Description	Type
px	Pixel. 1/96 of an inch.	Absolute (CSS3)
em	Equal to the computed value of the font-size property of the element on which it is used. When used for font sizes, the em unit is in relation to the font size of the parent.	Relative
%	A measure that is always relative to another value. The precise meaning of % varies depending upon which property it is being used.	Relative
ex	A rarely used relative measure that expresses size in relation to the x-height of an element's font.	Relative
ch	Another rarely used relative measure; this one expresses size in relation to the width of the zero ("0") character of an element's font.	Relative (CSS3 only)
rem	Stands for root em, which is the font size of the root element. Unlike em , which may be different for each element, the rem is constant throughout the document.	Relative (CSS3 only)
vw, vh	Stands for viewport width and viewport height. Both are percentage values (between 0 and 100) of the viewport (browser window). This allows an item to change size when the viewport is resized.	Relative (CSS3 only)

Absolute Units

Unit	Description	Type
in	Inches	Absolute
cm	Centimeters	Absolute
mm	Millimeters	Absolute
pt	Points (equal to 1/72 of an inch)	Absolute
pc	Pica (equal to 1/6 of an inch)	Absolute

Comments in CSS

- It is often helpful to add comments to your style sheets. Comments take the form:

/* comment goes here */

Grouped Selectors

Selecting multiple things

```
/* commas allow you to group selectors */
p, div, aside {
    margin: 0;
    padding: 0;
}
/* the above single grouped selector is equivalent to the following: */
p {
    margin: 0;
    padding: 0;
}
div {
    margin: 0;
    padding: 0;
}
aside {
    margin: 0;
    padding: 0;
}
```

LISTING 3.4 Sample grouped selector

- You can select a group of elements by separating the different element names with commas.
- This is a sensible way to reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule.

Selectors

Which are the selected elements

- Selectors can be used to find HTML elements by

- Element name

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

- Class

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

- Id

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

- Group

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


ID versus Class Selectors

How to decide

- ID selectors should only be used when referencing a single HTML element since an id attribute can only be assigned to a single HTML element.
- Class selectors should be used when (potentially) referencing several related elements.

Class Selectors

Simultaneously target different HTML elements

- A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.
- If a series of HTML element have been labeled with ***the same class attribute value***, then you can target them for styling by using a class selector, which takes the form: period (.) followed by the class name.

Attribute Selectors

Selecting via presence of element attribute or by the value of an attribute

- An **attribute selector** provides a way to select HTML elements by either the presence of an element attribute or by the value of an attribute.
- This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them.
- Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.

Pseudo Selectors

Select something that does not exist explicitly as an element

- A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object.
- A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.
- The most common use of this type of selectors is for targeting link states.

Pseudo Selectors

```
<head>
  <title>Share Your Travels</title>
  <style>
    a:link {
      text-decoration: underline;
      color: blue;
    }
    a:visited {
      text-decoration: underline;
      color: purple;
    }
    a:hover {
      text-decoration: none;
      font-weight: bold;
    }
    a:active {
      background-color: yellow;
    }
  </style>
</head>
<body>
  <p>Links are an important part of any web page. To learn more about
    links visit the <a href="#">W3C</a> website.</p>
  <nav>
    <ul>
      <li><a href="#">Canada</a></li>
      <li><a href="#">Germany</a></li>
      <li><a href="#">United States</a></li>
    </ul>
  </nav>
</body>
```

Contextual Selectors

Select elements based on their ancestors, descendants, or siblings

- A **contextual selector** (in CSS3 also called **combinators**) allows you to select elements based on their ancestors, descendants, or siblings.
- That is, it selects elements based on their context or their relation to other elements in the document tree.

Contextual Selectors

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element	<code>div p</code> Selects a <code><p></code> element that is contained somewhere within a <code><div></code> element. That is, the <code><p></code> can be any descendant, not just a child.
Child	A specified element that is a direct child of the specified element	<code>div>h2</code> Selects an <code><h2></code> element that is a child of a <code><div></code> element.
Adjacent Sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element.	<code>h3+p</code> Selects the first <code><p></code> after any <code><h3></code> .
General Sibling	A specified element that shares the same parent as the specified element.	<code>h3~p</code> Selects all the <code><p></code> elements that share the same parent as the <code><h3></code> .

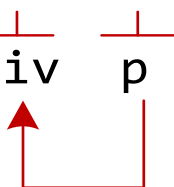
Descendant Selector

Selects all elements that are contained within another element

- While some of these contextual selectors are used relatively infrequently, almost all web authors find themselves using descendant selectors.
- A **descendant selector** matches all elements that are contained within another element. The space character used to indicate descendant selection.

context selected element

div p { ... }



Selects a <p> element
somewhere
within a <div> element

#main div p:first-child { ... }



Selects the first <p> element
somewhere within a <div> element
that is somewhere within an element
with an id="main"

Text Properties

Two basic types

- CSS provides two types of properties that affect text.
- **font properties** that affect the font and its appearance.
- **paragraph properties** that affect the text in a similar way no matter which font is being used.

Font-Family

A few issues here

- A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different.
- However, just because a given font is available on the web developer's computer, it does not mean that that same font will be available for all users who view the site.
- For this reason, it is conventional to supply a so-called **web font stack**, that is, a series of alternate fonts to use in case the original font choice is not on the user's computer.

Specifying the Font-Family

1 Use this font as
the first choice

3 If it isn't available, then
use this one

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

2 But if it is not available,
then use this one

4 And if it is not available
either, then use the
default generic serif font

Font Sizes

Mo control, mo problems

- The issue of font sizes is unfortunately somewhat tricky.
- In a print-based program such as a word processor, specifying a font size in points is unproblematic.
- However, absolute units such as points and inches do not translate very well to pixel-based devices.
- Somewhat surprisingly, pixels are also a problematic unit.
- Newer mobile devices in recent years have been increasing pixel densities so that a given CSS pixel does not correlate to a single device pixel.

Font Sizes

Welcome ems and percents again

- If we wish to create web layouts that work well on different devices, we should learn to use relative units such as **em** units or **percentages** for our font sizes (and indeed for other sizes in CSS as well).
- One of the principles of the web is that the user should be able to change the size of the text if he or she so wishes to do so.
- Using percentages or em units ensures that this user action will work.

How to use ems and percents

<code><body></code>	Browser's default text size is usually 16 pixels
<code><p></code>	100% or 1em is 16 pixels
<code><h3></code>	125% or 1.125em is 18 pixels
<code><h2></code>	150% or 1.5em is 24 pixels
<code><h1></code>	200% or 2em is 32 pixels

/ using 16px scale */*

```
body { font-size: 100%; }  
h3 { font-size: 1.125em; } /* 1.25 x 16 = 18 */  
h2 { font-size: 1.5em; } /* 1.5 x 16 = 24 */  
h1 { font-size: 2em; } /* 2 x 16 = 32 */
```

`<body>`

```
<p>this will be about 16 pixels</p>  
<h1>this will be about 32 pixels</h1>  
<h2>this will be about 24 pixels</h2>  
<h3>this will be about 18 pixels</h3>  
<p>this will be about 16 pixels</p>  
</body>
```

Layout

- `display` is CSS's most important property for controlling layout
- Default for most elements is usually `block` or `inline`
- `<div>` is the standard block-level element
 - A block-level element starts on a new line and stretches to the left and right as far as it can
- `` is the standard inline element
- `none` Setting `display` to `none` will render the page as if the element did not exist

Layout – width, margin, max-width

- Setting the `width` prevents a block element from stretching to the left and right
- Setting the margin to `auto` will horizontally centre the element within the container
- Setting `max-width` will make the browser handle small windows better

Layout - position

The position property can take four values:

- `static` – No special positioning is applied
- `relative` – Like static, unless you add extra properties
- `fixed` - Positioned relative to viewport, using `top`, `right`, `bottom`, `left`
- `absolute` – Positioned relative to nearest positioned element

Learn more about layouts

Excellent Tutorial available on the web

`http://learnlayout.com/`

Animations

```
@keyframes rotate {  
  0% {  
    transform: rotate(0deg) ;  
  }  
  
  100% {  
    transform: rotate(360deg) ;  
  }  
}  
  
p {  
  color: red;  
  width: 300px;  
  font-size: 40px;  
  transform-origin: center;  
}  
  
p:hover {  
  animation-name: rotate;  
  animation-duration: 0.6s;  
  animation-timing-function: linear;  
  animation-iteration-count: 5;  
}
```

Things to notice

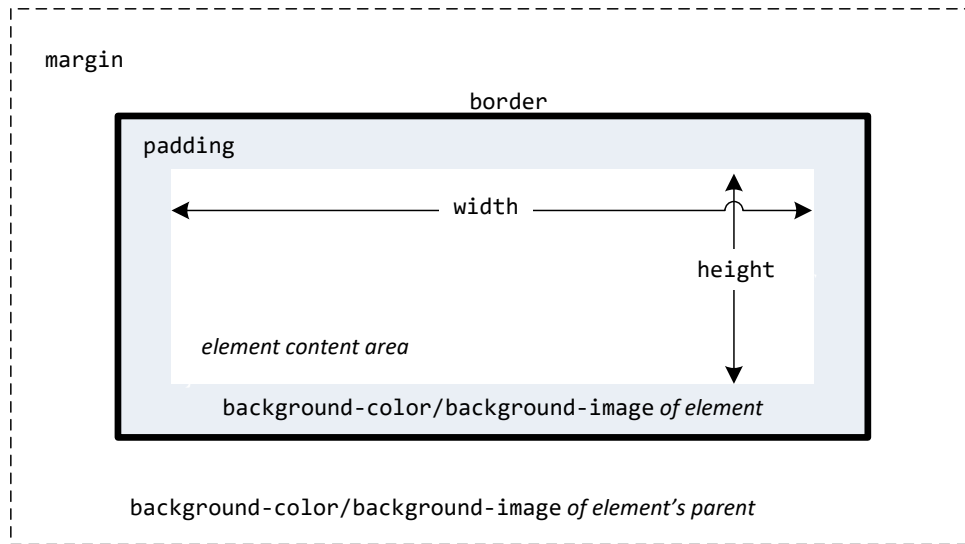
- `animation-iteration-count`: how many times does the animation run?
- CSS has functions!
 - `rotate`
 - `rgb`
 - `url`

The Box Model

Time to think inside the box

- In CSS, all HTML elements exist within an **element box**.
- It is absolutely essential that you familiarize yourself with the terminology and relationship of the CSS properties within the element box.

The Box Model



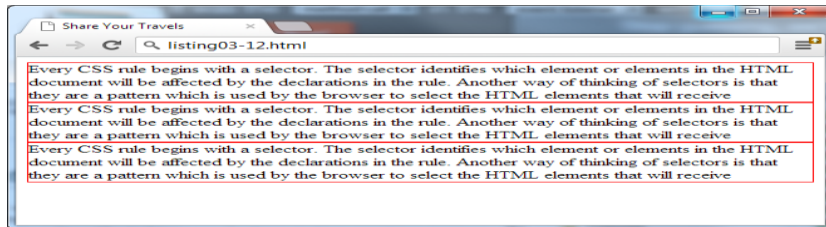
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Margins

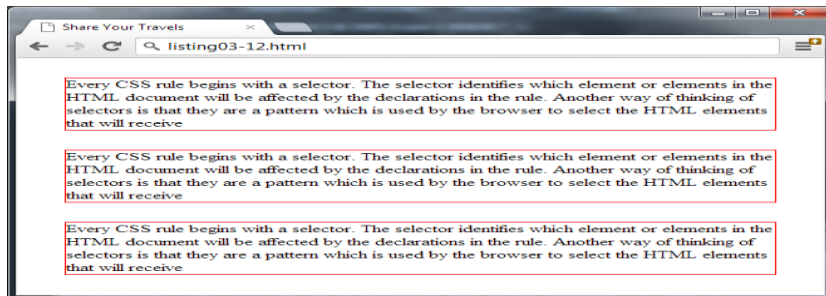
- Generates space around the elements
 - `margin-top`
 - `margin-right`
 - `margin-bottom`
 - `margin-left`
- All margins can have four values
 - `auto`
 - length (in px, pt, cm, etc.)
 - % - percentage of width of containing element
 - `inherit`

Margins and Padding

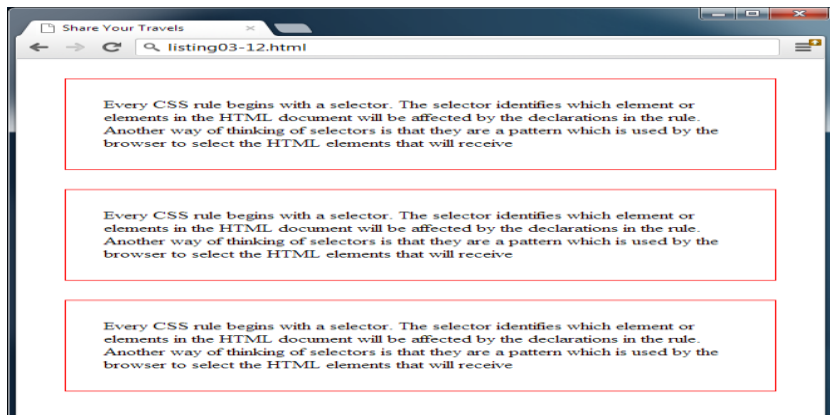
- Box Model Properties #3 and #4



```
p {  
  border: solid 1pt red;  
  margin: 0;  
  padding: 0;  
}
```



```
p {  
  border: solid 1pt red;  
  margin: 30px;  
  padding: 0;  
}
```



```
p {  
  border: solid 1pt red;  
  margin: 30px;  
  padding: 30px;  
}
```

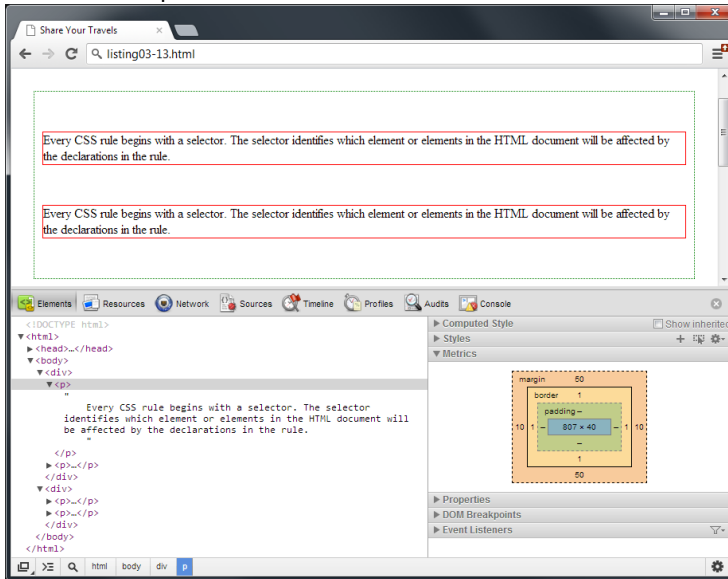
Developer Tools

Help is on the way

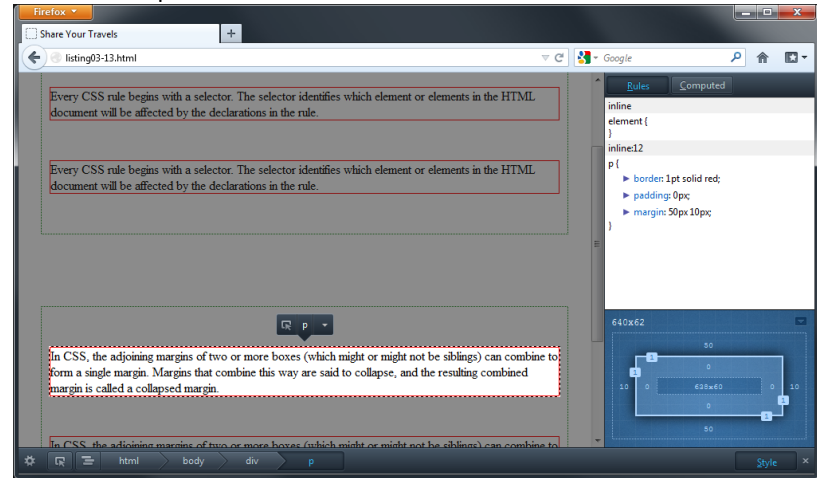
- Developer tools in current browsers make it significantly easier to examine and troubleshoot CSS than was the case a decade ago.
- You can use the various browsers' CSS inspection tools to examine, for instance, the box values for a selected element.

Developer Tools

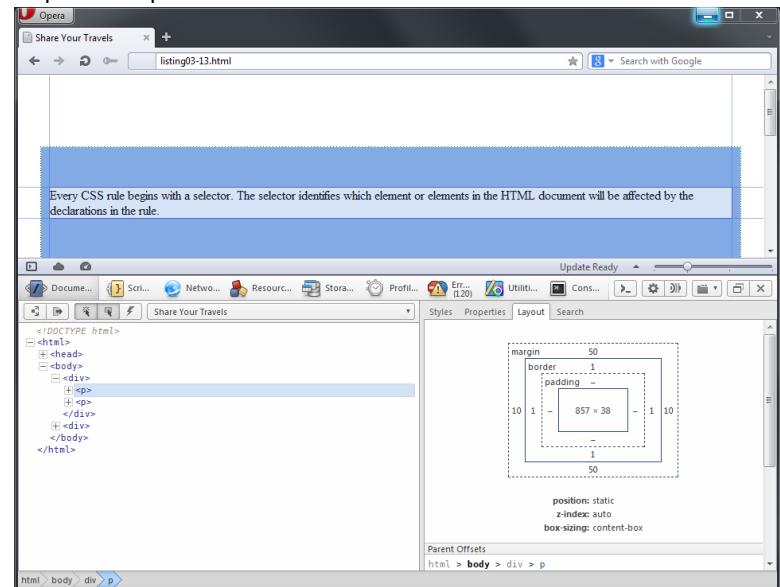
Chrome – Inspect Element



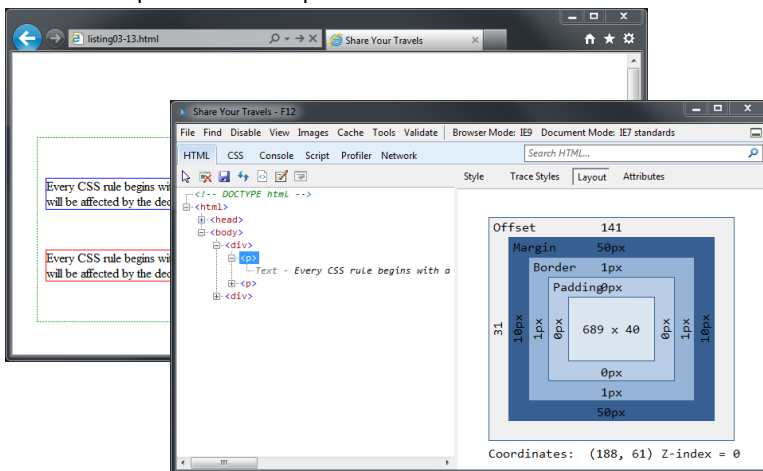
Firefox – Inspect



Opera – Inspect Element



Internet Explorer – Developer Tools



Good Tutorial

<https://htmldog.com/guides/css/beginner/applyingcss/>

That's all, folks!

- Questions?