Week 2

CSS Introduction

Daniel Awde - Park Innovation

Section 1 of 10

WHAT IS CSS?

Daniel Awde - Park Innovation

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.

What is CSS?

You be styling soon

CSS is a language in that it has its own syntax rules.

CSS can be added directly to any HTML element (via the style attribute), within the **<head>** element, or, most commonly, in a separate text file that contains only CSS.

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.

CSS Versions

Let's just say there's more than 1

- W3C published the CSS Level 1 Recommendation in 1996.
- A year later, the CSS Level 2 Recommendation (also more succinctly labeled simply as CSS2) was published.
- Even though work began over a decade ago, an updated version of the Level 2 Recommendation, CSS2.1, did not become an official W3C Recommendation until June 2011.
- And to complicate matters even more, all through the last decade (and to the present day as well), during the same time the CSS2.1 standard was being worked on, a different group at the W3C was working on a CSS3 draft.

Browser Adoption

Insert obligatory snide comment about Internet Explorer 6 here

While Microsoft's Internet Explorer was an early champion of CSS, its later versions (especially IE5, IE6, and IE7) for Windows had uneven support for certain parts of CSS2.

In fact, all browsers have left certain parts of the CSS2 Recommendation unimplemented.

CSS has a reputation for being a somewhat frustrating language.

- this reputation is well deserved!

Section 2 of 10

CSS SYNTAX

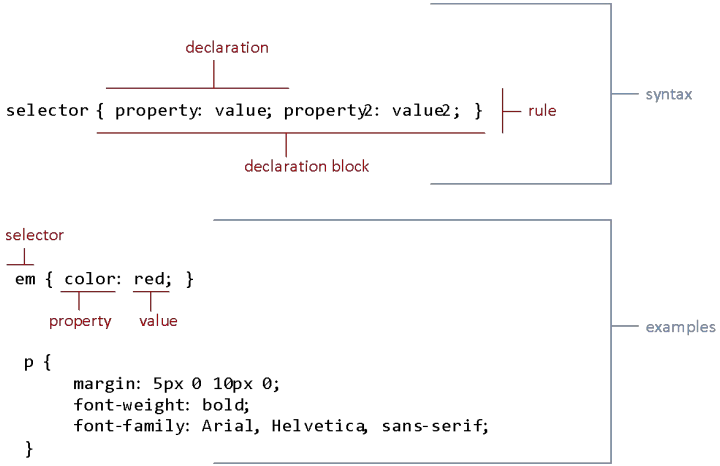
Daniel Awde - Park Innovation

CSS Syntax

Rules, properties, values, declarations

A CSS document consists of one or more **style rules**.

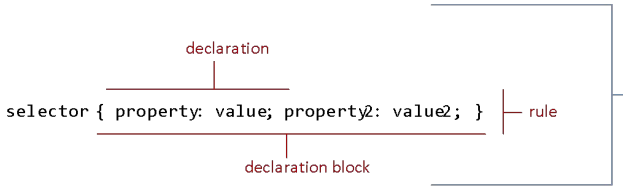
A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).

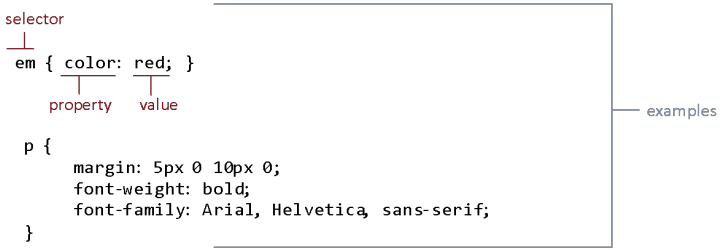


Declaration Blocks

The series of declarations is also called the **declaration block**.

- A declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space
- Each declaration is terminated with a semicolon.





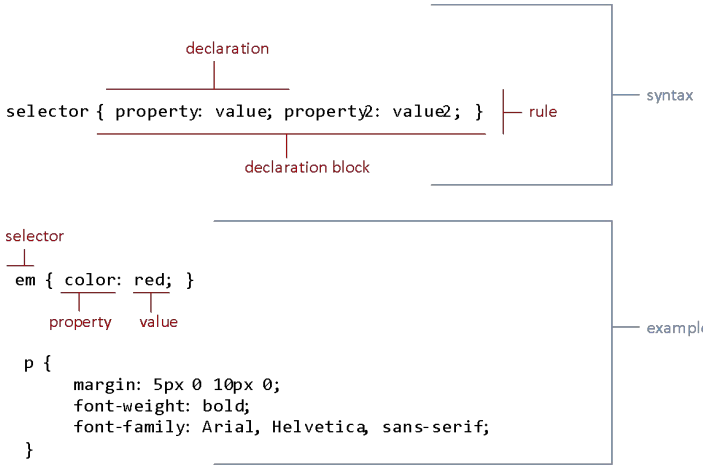
Selectors

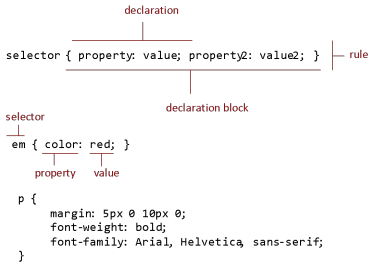
Which elements

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.





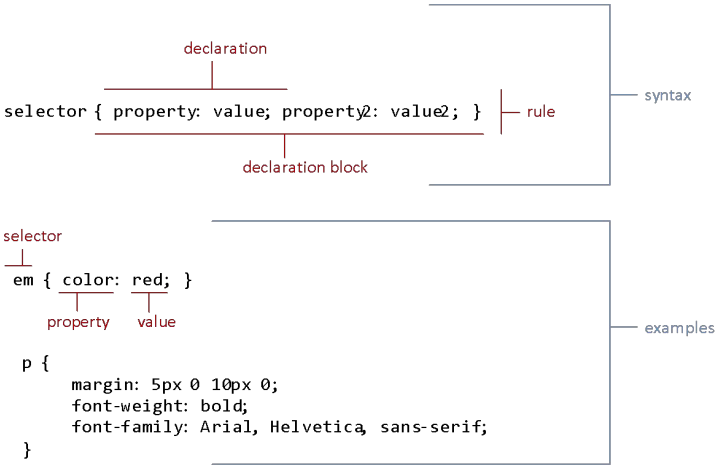
Properties

Which style properties of the selected elements

Each individual CSS declaration must contain a **property**.

These property names are predefined by the CSS standard.

The CSS2.1 Recommendation defines over a hundred different property names.



Properties

Common CSS properties

Property Type	Property
Fonts	font
	font-family
	font-size
	font-style
	font-weight
	@font-face
Text	letter-spacing
	line-height
	text-align
	text-decoration
	text-indent
Color and background	background
	background-color
	background-image
	background-position
	background-repeat
	color
Borders	border
	border-color
	border-width
	border-style
	border-top
	border-top-color
	border-top-width
	etc

Da

Properties

Common CSS properties continued.

Property Type	Property
Spacing	padding
	padding-bottom, padding-left, padding-right, padding-top
	margin
	margin-bottom, margin-left, margin-right, margin-top
Sizing	height
	max-height
	max-width
	min-height
	min-width
	width
Layout	bottom, left, right, top
	clear
	display
	float
	overflow
	position
	visibility
	z-index
Lists	list-style
	list-style-image
	list-style-type

Da

Values

What style value for the properties

Each CSS declaration also contains a **value** for a property.

- The unit of any given value is dependent upon the property.
- Some property values are from a predefined list of keywords.
- Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

Color Values

CSS supports a variety of different ways of describing color

Method	Description	Example
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	color: red; color: hotpink; /* CSS3 only */
RGB	Uses three different numbers between 0 and 255 to describe the Red, Green, and Blue values for the color.	color: rgb(255,0,0); color: rgb(255,105,180);
Hexadecimal	Uses a six-digit hexadecimal number to describe the red, green, and blue value of the color; each of the three RGB values is between 0 and FF (which is 255 in decimal). Notice that the hexadecimal number is preceded by a hash or pound symbol (#).	color: #FF0000; color: #FF69B4;
RGBa	Allows you to add an alpha, or transparency, value. This allows a background color or image to “show through” the color. Transparency is a value between 0.0 (fully transparent) and 1.0 (fully opaque).	color: rgb(255,0,0, 0.5);
HSL	Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well.	color: hsl(0,100%,100%); color: hsl(330,59%,100%);

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. In CSS2 this is a relative measure, while in CSS3 it is absolute (1/96 of an inch).	Relative (CSS2) 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 */

Section 3 of 10

LOCATION OF STYLES

Daniel Awde - Park Innovation

Actually there are three …

Different types of style sheet

Author-created style sheets (what we are learning in this presentation).

User style sheets allow the individual user to tell the browser to display pages using that individual's own custom style sheet. This option is available in a browser usually in its accessibility options area.

The **browser style sheet** defines the default styles the browser uses for each HTML element.

Style Locations

Author Created CSS style rules can be located in three different locations

CSS style rules can be located in three different locations.

- Inline
- Embedded
- External

You can combine all 3!

Inline Styles

Style rules placed within an HTML element via the style attribute

```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 3.1 Internal styles example

An inline style only affects the element it is defined within and will override any other style definitions for the properties used in the inline style.

Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability.

Inline styles can however be handy for quickly testing out a style change.

Embedded Style Sheet

Style rules placed within the <style> element inside the <head> element

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <style>
      h1 { font-size: 24pt; }
      h2 {
       font-size: 18pt;
       font-weight: bold;
      }
   </style>
</head>
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   ...
```

LISTING 3.2 Embedded styles example

While better than inline styles, using embedded styles is also by and large discouraged.

Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

External Style Sheet

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

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

LISTING 3.3 Referencing an external style sheet

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.
- The browser is able to cache the external style sheet which can improve the performance of the site

Section 4 of 10

SELECTORS

Daniel Awde - Park Innovation

Selectors

Things that make your life easier

When defining CSS rules, you will need to first need to use a **selector** to tell the browser which elements will be affected.

CSS selectors allow you to select

- individual elemenets
- multiple HTML elements,
- elements that belong together in some way, or
- elements that are positioned in specific ways in the document hierarchy.

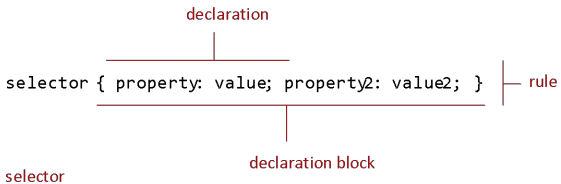
There are a number of different selector types.

Element Selectors

Selects all instances of a given HTML element

Uses the HTML element name.

You can select all elements by using the **universal element selector**, which is the * (asterisk) character.




```
p {
    margin: 5px 0 10px 0;
    font-weight: bold;
    font-family: Arial, Helvetica, sans-serif;
}
```

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.

Reset

```
html, body, div, span, h1, h2, h3, h4, h5, h6, p {
 margin: 0;
 padding: 0;
 border: 0;
 font-size: 100%;
 vertical-align: baseline;
}
```

Grouped selectors are often used as a way to quickly **reset** or remove browser defaults.

The goal of doing so is to reduce browser inconsistencies with things such as margins, line heights, and font sizes.

These reset styles can be placed in their own css file (perhaps called reset.css) and linked to the page **before** any other external styles sheets.

Class Selectors

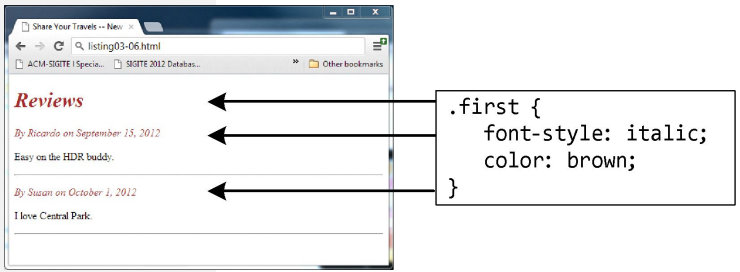
Simultaneously target different HTML elements

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.

Class Selectors

```
<head>
   <title>Share Your Travels </title>
         <style>
                  .first {
                           font-style: italic;
                           color: brown;
                  }
         </style>
</head>
<body>
   <h1 class="first">Reviews</h1>
   <div>
      <p class="first">By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p class="first">By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```



Danie

Id Selectors

Target a specific element by its id attribute

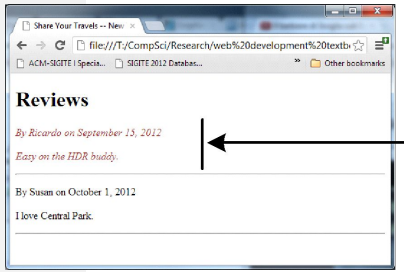
An **id selector** allows you to target a specific element by its id attribute regardless of its type or position.

If an HTML element has been labeled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name.

Note: You should only be using an **id** once per page

Id Selectors

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
          <style>
                    #latestComment {
                              font-style: italic;
                              color: brown;
                    }
          </style>
</head>
<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```




```
#latestComment {
   font-style: italic;
   color: brown;
}
```

Da

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.

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.

Attribute Selectors

```
[title] {
   cursor: help;
   padding-bottom: 3px;
   border-bottom: 2px dotted blue ;
   text-decoration: none;
}
```


```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
          <style>
                    [title] {
                              cursor: help;
                              padding-bottom: 3px;
                              border-bottom: 2px dotted blue;
                              text-decoration: none;
                    }
          </style>
</head>
<body>
   <div>
      <img src="images/flags/CA.png" title="Canada Flag" />
            <h2><a href="countries.php?id=CA" title="see posts from Canada">
          Canada</a>
       </h2>
            <p>Canada is a North American country consisting of … </p>
            <div>
                    <img src="images/square/6114907897.jpg" title="At top of Sulpher Mountain">
                    <img src="images/square/6592317633.jpg" title="Grace Presbyterian Church">
                    <img src="images/square/6592914823.jpg" title="Calgary Downtown">
            </div>
   </div>
</body>
```

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>
```

LISTING 3.8 Styling a link using pseudo-class selectors

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

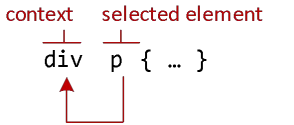
Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element	div p Selects a <p> element that is contained somewhere within a element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct child of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

Descendant Selector

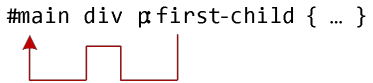
Selects all elements that are contained within another element

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 character used to indicate descendant selection is the space character.

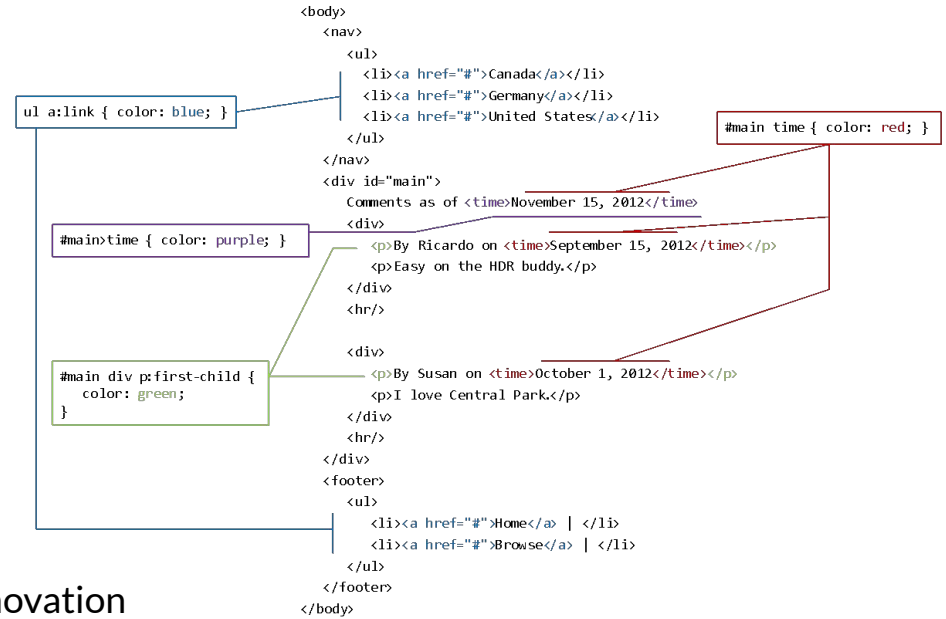


Selects a <p> element somewhere within a element



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

Contextual Selectors in Action



Daniel Awde - Park Innovation

Section 5 of 10

THE CASCADE: HOW STYLES INTERACT

Daniel Awde - Park Innovation

Why Conflict Happens

In CSS that is

Because

- there are three different types of style sheets (author-created, user-defined, and the default browser style sheet),
- author-created stylesheets can define multiple rules for the same HTML element,

CSS has a system to help the browser determine how to display elements when different style rules conflict.

Cascade

How conflicting rules are handled in CSS

The “Cascade” in CSS refers to how conflicting rules are handled.

The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks.

The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.

Cascade Principles

CSS uses the following cascade principles to help it deal with conflicts:

- **inheritance,**
- **specificity,**
- **location**

Inheritance

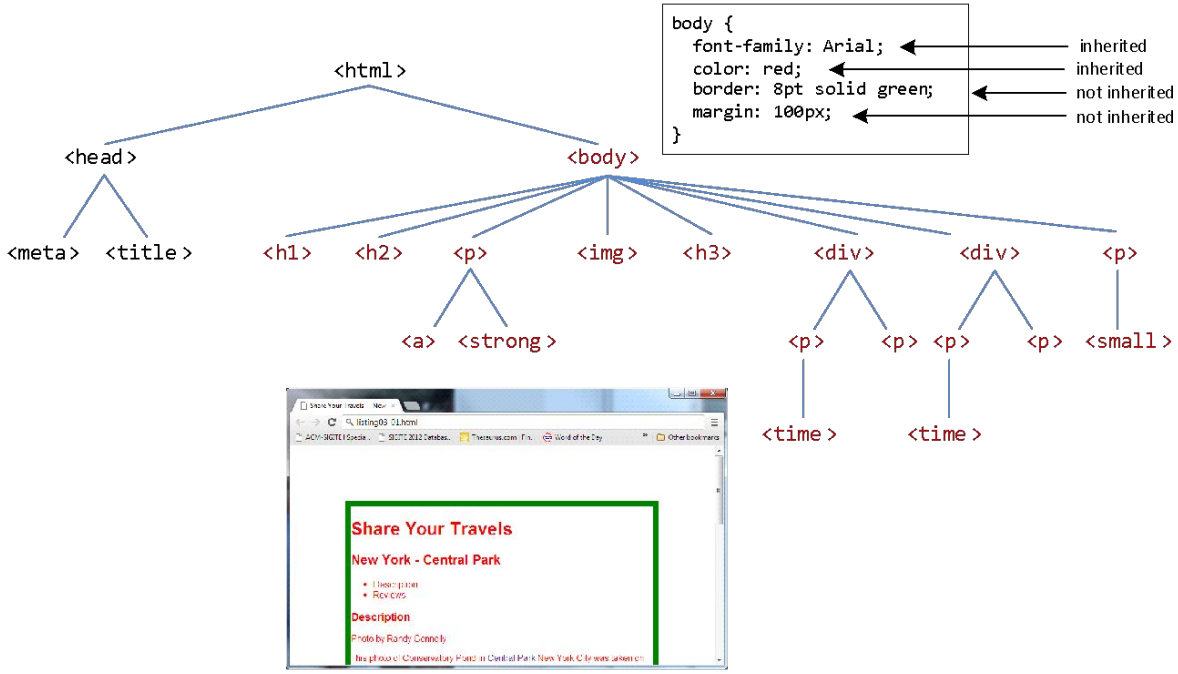
Cascade Principle #1

Many (but not all) CSS properties affect not only themselves but their descendants as well.

Font, color, list, and text properties are inheritable.

Layout, sizing, border, background and spacing properties are not.

Inheritance

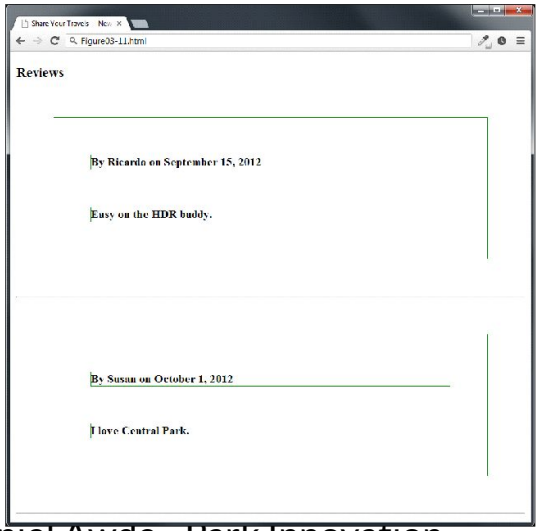


Daniel Awde - Park Innovation

Inheritance

How to force inheritance

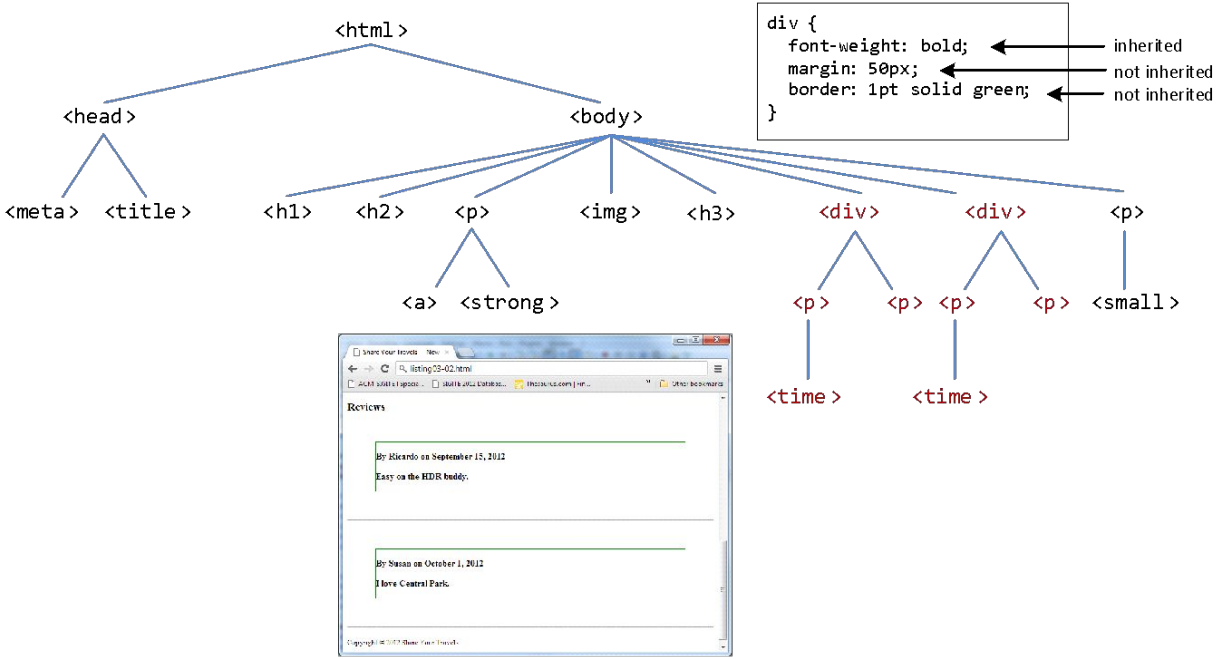
It is possible to tell elements to inherit properties that are normally not inheritable.




```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
p {
  border: inherit;
  margin: inherit;
}
```

```
<h3>Reviews</h3>
<div>
   <p>By Ricardo on <time>September 15, 2012</time></p>
   <p>Easy on the HDR buddy.</p>
</div>
<hr/>

<div>
   <p>By Susan on <time>October 1, 2012</time></p>
   <p>I love Central Park.</p>
</div>
<hr/>
```

Inheritance



Specificity

Cascade Principle #2

Specificity is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element.

The more *specific* the selector, the more it takes precedence (i.e., overrides the previous definition).

Specificity

How it works

The way that specificity works in the browser is that the browser assigns a weight to each style rule.

When several rules apply, the one with the greatest weight takes precedence.

Location

Cascade Principle #3

When inheritance and specificity cannot determine style precedence, the principle of **location** will be used.

The principle of location is that when rules have the same specificity, then the latest are given more weight.

Section 6 of 10

THE BOX MODEL

Daniel Awde - Park Innovation

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.

Daniel Awde - Park Innovation

Background

Box Model Property #1

The background color or image of an element fills an element out to its border (if it has one that is).

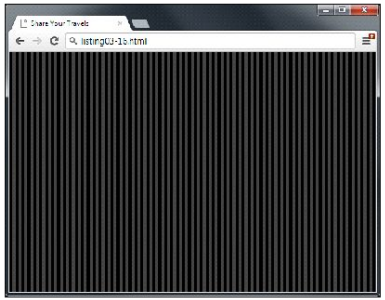
In contemporary web design, it has become extremely common too use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc) rather than using the <img> element.

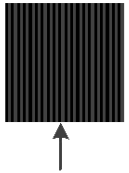
B

Property	Description
background	A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short-hand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. Possible values are: fixed, scroll.
background-color	Sets the background color of the element.
background-image	Specifies the background image (which is generally a jpeg, gif, or png file) for the element. Note that the URL is relative to the CSS file and not the HTML. CSS3 introduced the ability to specify multiple background images.
background-position	Specifies where on the element the background image will be placed. Some possible values include: bottom, center, left, and right. You can also supply a pixel or percentage numeric position value as well. When supplying a numeric value, you must supply a horizontal/vertical pair; this value indicates its distance from the top left corner of the element.
background-repeat	Determines whether the background image will be repeated. This is a common technique for creating a tiled background (it is in fact the default behavior). Possible values are: repeat, repeat-x, repeat-y, and no-repeat.
background-size	New to CSS3, this property lets you modify the size of the background image.

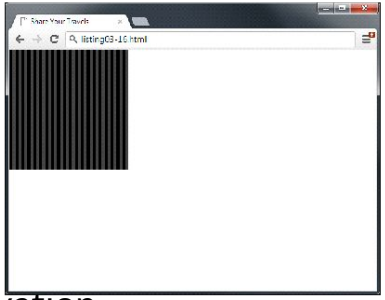
Daniel A

Background Repeat

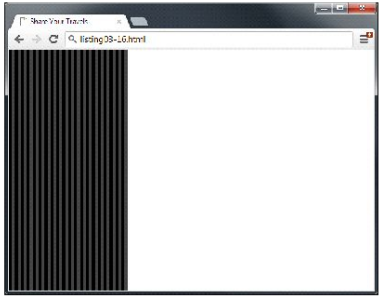




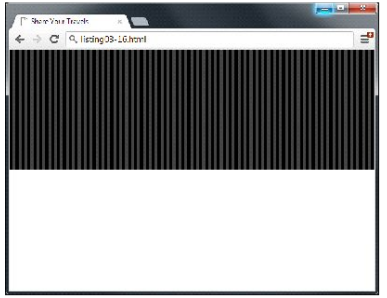
```
background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;
```



```
background-repeat: no-repeat;
```

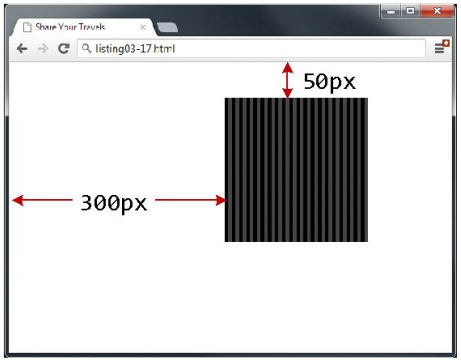


```
background-repeat: repeat-y;
```



```
background-repeat: repeat-x;
```

Background Position




```
body {
        background: white url(../images/backgrounds/body-background-tile.gif) no-repeat;
        background-position: 300px 50px;
}
```

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	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: **border-style border-width border-color**
border-style	Specifies the line type of the border. Possible values are: solid, dotted, dashed, double, groove, ridge, inset, and outset.
border-width	The width of the border in a unit (but not percents). A variety of keywords (thin, medium, etc) are also supported.
border-color	The color of the border in a color unit.
border-radius	The radius of a rounded corner.
border-image	The URL of an image to use as a border.

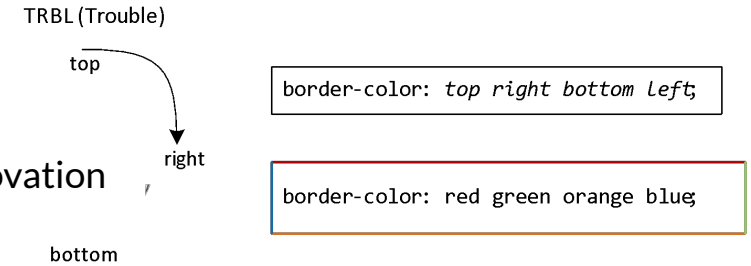
Shortcut notation

TRBL

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

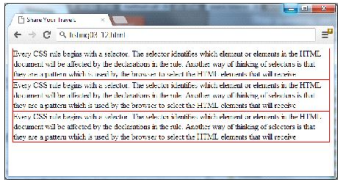
```
border-top-color: red;          /* sets just the top side */
border-right-color: green;      /* sets just the right side */
border-bottom-color: yellow;    /* sets just the bottom side */
border-left-color: blue;        /* sets just the left side */

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.**

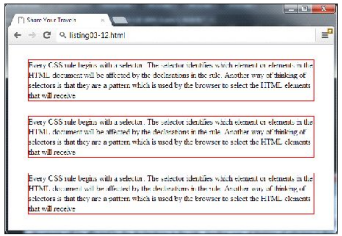


Margins and Padding

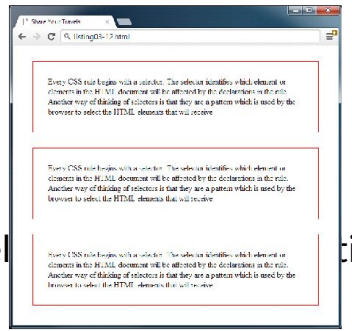
Box Model Properties #3 and #4



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



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

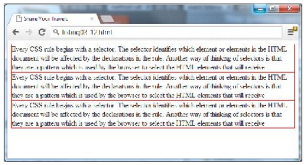


```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 30px;
}
```

Daniel [illegible]tion

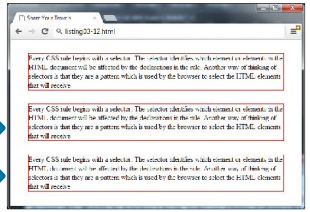
Margins

Why they will cause you trouble.



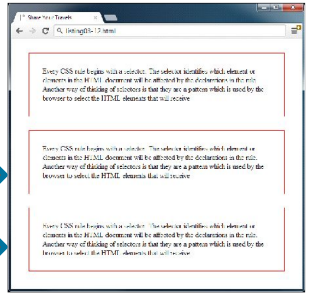
```
p {
    border: solid 1pt red;
    margin: 0;
    padding: 0;
}
```

Did you notice that the space between paragraphs one and two and between two and three is the same as the space before paragraph one and after paragraph three?



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

This is due to the fact that adjoining vertical margins collapse.



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

Width and Height

Box Model Properties #5 and #6

The width and height properties specify the size of the element's content area.

Perhaps the only rival for collapsing margins in troubling our students, box dimensions have a number of potential issues.

Width and Height

Potential Problem #1

Only block-level elements and non-text inline elements such as images have a **width** and **height** that you can specify.

By default the width of and height of elements is the actual size of the content.

For text,

- this is determined by the font size and font face;

For images,

- the width and height of the actual image in pixels determines the element box's dimensions.

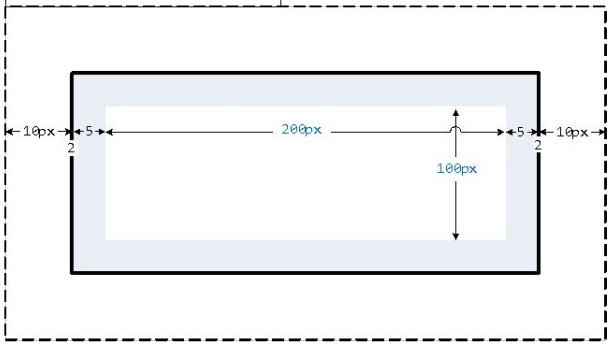
Width and Height

Potential Problem #2

Since the width and the height refer to the size of the content area, by default, the total size of an element is equal to not only its content area, but also to the sum of its padding, borders, and margins.

```
div {
  box-sizing: content-box;
  width: 200px;
  height: 100px;
  padding: 5px;
  margin: 10px;
  border: solid 2pt black;
}
```

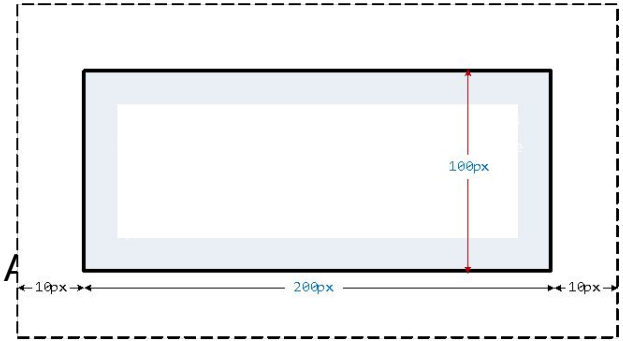
True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px



Default

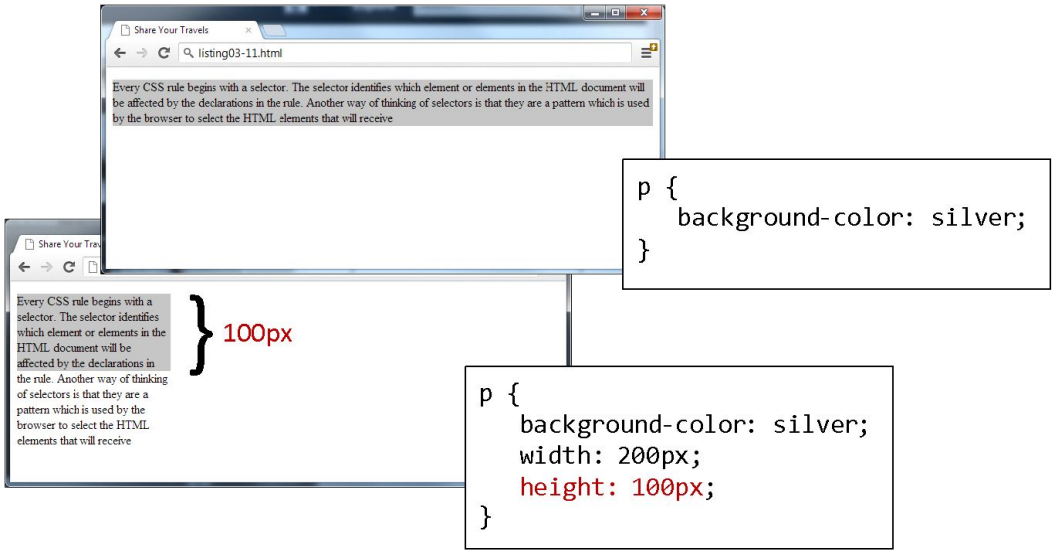
```
div {
  ...
  box-sizing: border-box;
}
```

True element width = 10 + 200 + 10 = 220 px
True element height = 10 + 100 + 10 = 120 px



Daniel A

Width and Height



Daniel Awde - Park Innovation

Overflow Property

Daniel Awde - Park Innovation

Sizing Elements

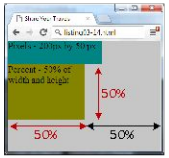
Time to embrace ems and percentages

While the previous examples used pixels for its measurement, many contemporary designers prefer to use percentages or em units for widths and heights.

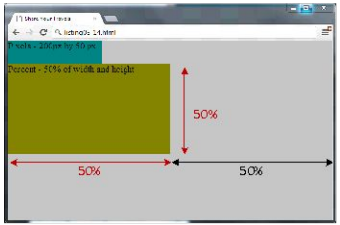
- When you use percentages, the size is relative to the size of the parent element.
- When you use ems, the size of the box is relative to the size of the text within it.

The rationale behind using these relative measures is to make one's design scalable to the size of the browser or device that is viewing it.

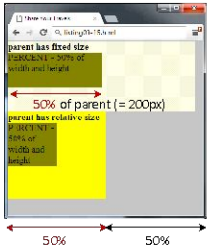
```
<style>
  html,body {
      margin:0;
      width:100%;
      height:100%;
      background: silver;
  }
  .pixels {
      width:200px;
      height:50px;
      background: teal;
  }
  .percent {
      width:50%;
      height:50%;
      background: olive;
  }
```



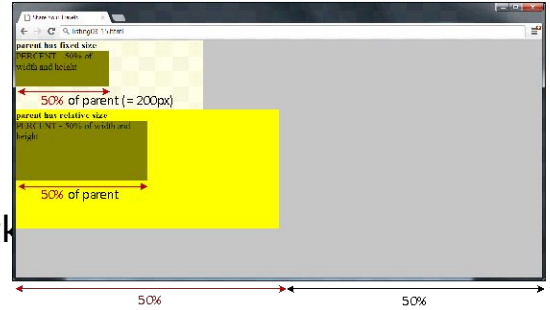

```
<body>
   <div class="pixels">
     Pixels - 200px by 50 px
   </div>
   <div class="percent">
     Percent - 50% of width and height
   </div>
</body>
```




```
  .parentFixed {
      width:400px;
      height:150px;
      background: beige;
  }
  .parentRelative {
      width:50%;
      height:50%;
      background: yellow;
  }
</style>
```




```
<body>
<div class="parentFixed">
   <strong>parent has fixed size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
<div class="parentRelative">
   <strong>parent has relative size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
</body>
```



Section 7 of 10

TEXT STYLING

Daniel Awde - Park Innovation

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

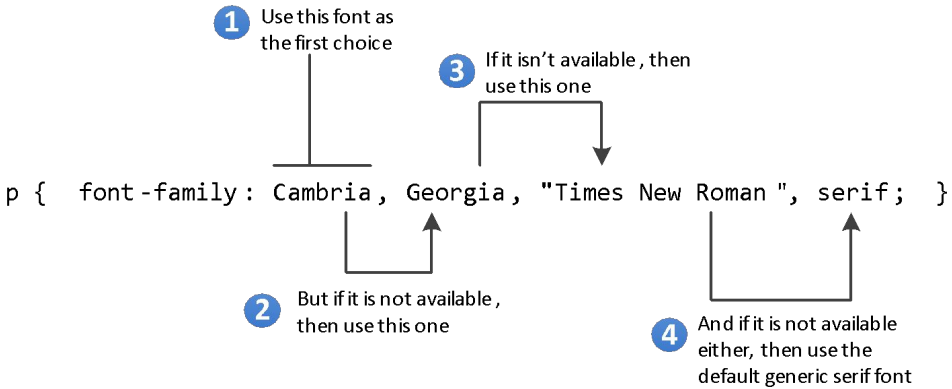
A few issues here

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 in not on the user's computer.

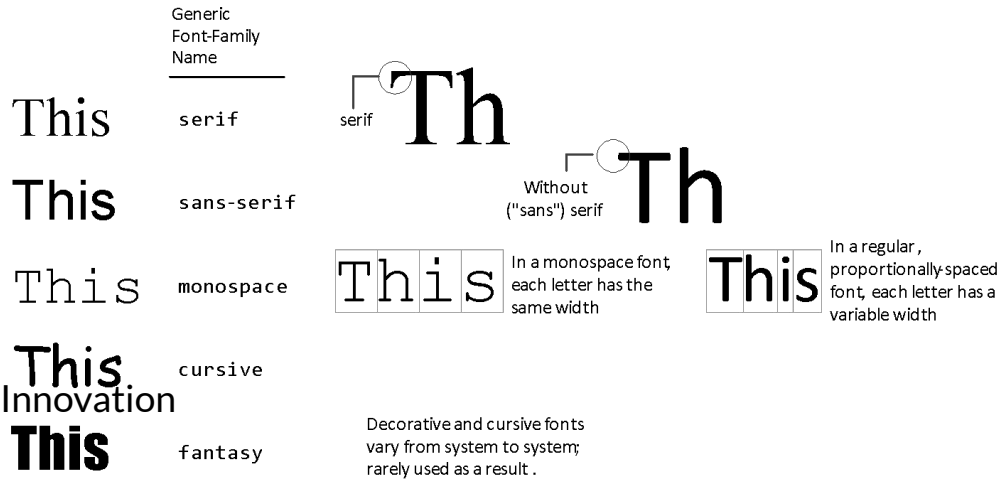
Specifying the Font-Family



Generic Font-Family

The font-family property supports five different generic families.

The browser supports a typeface from each family.



@font-face

The future is now

Over the past few years, the most recent browser versions have begun to support the **@font-face** selector in CSS.

This selector allows you to use a font on your site even if it is not installed on the end user's computer.

Due to the on-going popularity of open source font sites such as Google Web Fonts (http://www.google.com/webfonts) and Font Squirrel (http://www.fontsquirrel.com/), @font-face seems to have gained a critical mass of widespread usage.

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).

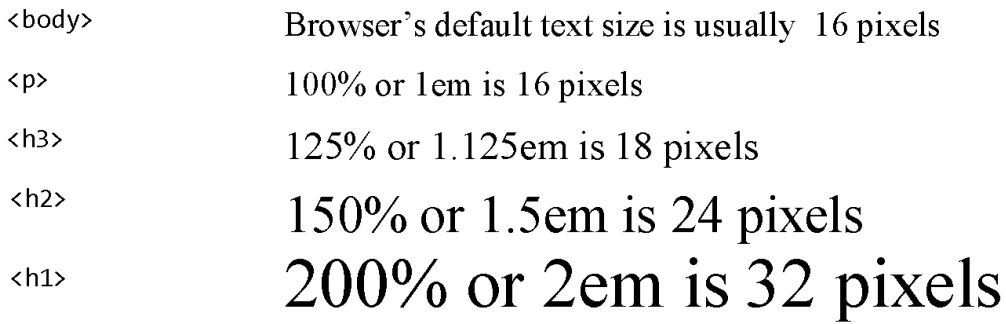
One of the principles of the web is that the user should be able to change the size of the text if he or she so wishes to do so.

Using percentages or em units ensures that this user action will work.

How to use ems and percents

When used to specify a font size, both em units and percentages are relative to the parent's font size.

How to use ems and percents




```
/* 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>
```

How to use ems and percents

It might seem easy … but it's not …

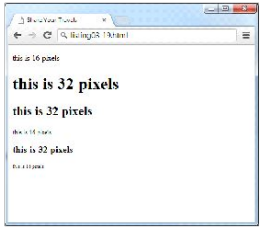
While this looks pretty easy to master, things unfortunately can quickly become quite complicated.

Remember that percents and em units are relative to their parents, so if the parent font size changes, this affects all of its contents.

ems and percents

```
<body>
  <p>this is 16 pixels</p>
  <h1>this is 32 pixels</h1>
  <article>
     <h1>this is 32 pixels</h1>
     <p>this is 16 pixels</p>
     <div>
        <h1>this is 32 pixels</h1>
        <p>this is 16 pixels</p>
     </div>
  </article>
</body>
```


```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }      /* 1 x 16 = 16px */
h1   { font-size: 2em; }      /* 2 x 16 = 32px */
```




```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }
h1   { font-size: 2em; }

article { font-size: 75% }   /* h1 = 2 * 16 * 0.75 = 24px
                                p = 1 * 16 * 0.75 = 12px  */

div  { font-size: 75% }      /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                                p = 1 * 16 * 0.75 * 0.75 = 9px   */
```

Daniel Awde - Park Innovation

The rem unit

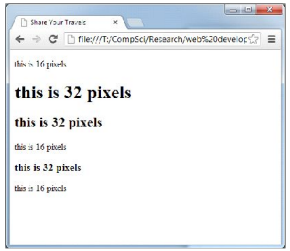
Solution to font sizing hassles?

CSS3 now supports a new relative measure, the **rem** (for root em unit).

This unit is always relative to the size of the root element (i.e., the <html> element).

However, since Internet Explorer prior to version 9 do not support the rem units, you need to provide some type of fallback for those browsers.

The rem unit




```
/* using 16px scale */

body { font-size: 100%; }
p {
       font-size: 16px;  /* for older browsers: won't scale properly though */
       font-size: 1rem;  /* for new browsers: scales and simple too */
}
h1   { font-size: 2em; }

article { font-size: 75% }  /* h1 = 2 * 16 * 0.75 = 24px
                               p = 1 * 16 = 16px  */

div  { font-size: 75% }      /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                               p = 1 * 16 = 16px   */
```

Section 8 of 10

Bootstrap

Daniel Awde - Park Innovation

What is bootstrap?

- Bootstrap is the most popular CSS Framework for developing mobile-first websites.
- Bootstrap is a free open-source CSS Framework
- Bootstrap 4 is the newest version of Bootstrap
- Bootstrap 4 is supported by all major browsers except Internet Explorer 9.

Responsive web design

- Responsive Web design is the approach that suggests that design and development should respond to the user's behavior and environment based on screen size, platform and orientation.

Where to get bootstrap 4

- There is two ways to get Bootstrap 4.
  - First is using the Bootstrap 4 Content Delivery Network (CDN)
  - Second you could visit the getbootstrap.com website to download Bootstrap 4 to host it on your personal computer.

Bootstrap 4 – HTML5 Doctype

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
  </head>
</html>
```

- Bootstrap 4 uses HTML elements and CSS properties that require the HTML5 doctype.
- Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set.

Bootstrap 4 – Mobile First

- Bootstrap 4 is designed to be responsive to mobile devices. Mobile first
- To ensure proper rendering and zoom, add <meta> tag inside the head element.
- The width=device-width part sets the width of the page to the screen-width of the device
- The initial-scale=1 sets initial zoom level when browser loaded.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Bootstrap 4 - CDN

- If you do not want to download and host Bootstrap 4, you must include the CDN (Content Delivery Network)
- Include these line inside the <head> tags

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Popper JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">
</script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

Bootstrap 4 - Progress

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>

</body>
</html>
```

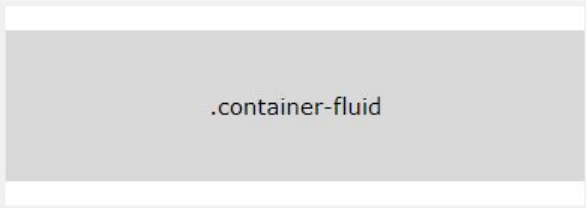
Bootstrap 4 – Containers

.CONTAINER

- Provides a responsive fixed width container.

.CONTAINER-FLUID

- Provides a full width container, spanning the entire width of the viewport



Bootstrap syntax

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This part is inside a .container class.</p>
  <p>The .container class provides a responsive fixed width container.</p>
</div>

</body>
</html>
```

Bootstrap 4 – Container Formatting

- pt-3 = sets top padding
- p-3 = sets padding (all directions)
- my-3 = sets margins
- border = plain container border
- bg-color = sets background color & color
- text-color = sets text color

Daniel Awde - Park Innovation

Bootstrap 4 - Colors

Text Color

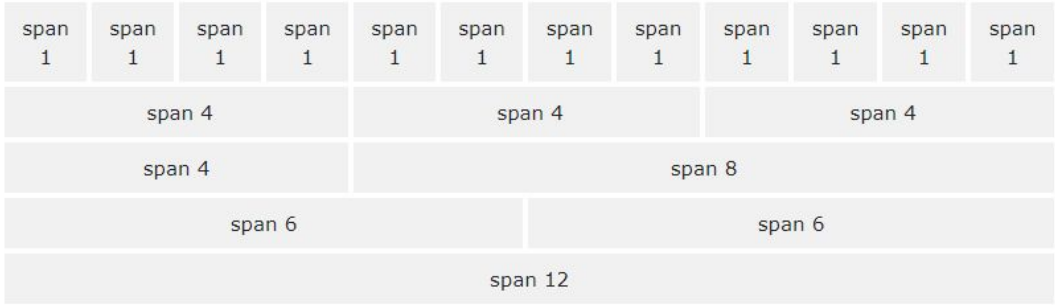
- .text-primary
- .text-muted
- .text-success
- .text-warning
- .text-info
- .text-danger
- .text-secondary
- .text-white
- .text-dark
- .text-body (default color)
- .text-light

Background Color

- .bg-primary
- .bg-success
- .bg-info
- .bg-warning
- .bg-danger
- .bg-secondary
- .bg-dark
- .bg-light

Bootstrap 4 Grids

- Built with flexbox and allows up to 12 columns across the page.
- If you do not want to use all 12 columns individually, you can group the columns together to create wider columns.
- The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.
- Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).



```
<!-- Control the column width, and how they should appear on different devices -->
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>

<!-- Or let Bootstrap automatically handle the layout -->
<div class="row">
  <div class="col"></div>
  <div class="col"></div>
  <div class="col"></div>
</div>
```

Grid Format

Bootstrap 4 - Jumbotron

- Jumbotrons are used to bring extra attention to special content or information.
- Use a element with a .jumbotron to create a jumbotron
- If you want rounded corners place a .jumbotron inside of the .container

```
<div class="jumbotron">
  <h1>Bootstrap Tutorial</h1>
  <p>Bootstrap is the most popular HTML, CSS...</p>
</div>
```

Bootstrap 4 – Tables

- .table class adds basic styling to a table
- .table-striped class adds zebra-stripes to a table
- .table-bordered class adds borders to a table and cells.
- .table-hover class adds a hover effect on table rows
- .table-dark class adds a black background to the table
- .table-borderless class removes borders from the table

Bootstrap 4 – Icons & Images

- https://fontawesome.bootstrapcheatsheets.com/# - icons for bootstrap
- .rounded – adds rounded corners to an image.
- .rounded-circle – shapes the image to a circle.
- .img-thumbnail – shapes the image to a thumbnail (bordered).
- .float-right – float image to the right.
- .float-left – float image to the left.
- .image-fluid – add this class to create a responsive photo.

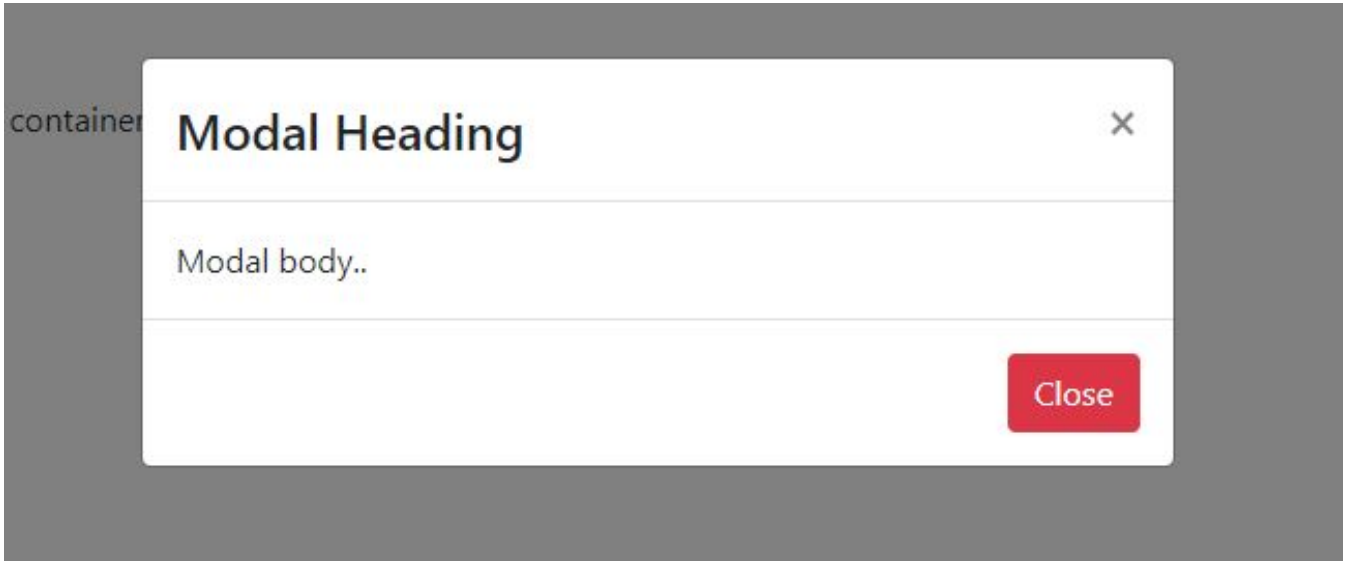
Bootstrap 4 – Navbar

- With Bootstrap. A navigation bar can extend or collapse, depending on screen size.
- A standard navigation bar is created with the .navbar class, followed by a responsive collapsing class.
  - .navbar-expand-xl
  - .navbar-expand-lg
  - .navbar-expand-md
  - .navbar-expand-sm
- Determines when a .navbar should stack vertically based on screen size.
- To add links inside the navbar, use a <ul> element with class="navbar-nav". Then add <li> elements with a .nav-item class followed by an <a> element with a .nav-link class.

Bootstrap 4 – Navbar

- You may want to hide the navigation links, especially on small screens, and replace them with a button. By pressing the button you will reveal the links.
- To create a collapsible navigation bar, use a button with class="navbar-toggler", data-toggle="collapse" and data-target="#thetarget". Then wrap the navbar contents inside a div element with class="collapse navbar-collapse", followed by an id that matches the data-target of the button: "thetarget".

Bootstrap 4 - Modal



Daniel Awde - Park Innovation

Bootstrap 4 – Modal

- The Modal component is a dialog box/popup window that is displayed on top of the current page.

```
<div class="container">
  <h2>Fading Modal</h2>
  <!-- Button to Open the Modal -->
  <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
    Open modal
  </button>
  <!-- The Modal -->
  <div class="modal fade" id="myModal">
    <div class="modal-dialog">
      <div class="modal-content">
        <!-- Modal Header -->
        <div class="modal-header">
          <h4 class="modal-title">Modal Header</h4>
          <button type="button" class="close" data-dismiss="modal">&times;</button>
        </div>
        <!-- Modal body -->
        <div class="modal-body">
          Modal body
        </div>
        <!-- Modal footer -->
        <div class="modal-footer">
          <button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Bootstrap Example

https://getbootstrap.com/docs/4.0/examples/album/

Daniel Awde - Park Innovation

Section 9 of 10

JavaScript

Daniel Awde - Park Innovation

WHAT IS JAVASCRIPT

Daniel Awde - Park Innovation

What is JavaScript

- JavaScript runs right inside the browser
- JavaScript is dynamically typed
- JavaScript is object oriented in that almost everything in the language is an object
    - the objects in JavaScript are prototype-based rather than class-based, which means that while JavaScript shares some syntactic features of PHP, Java or C#, it is also quite different from those languages

What isn't JavaScript

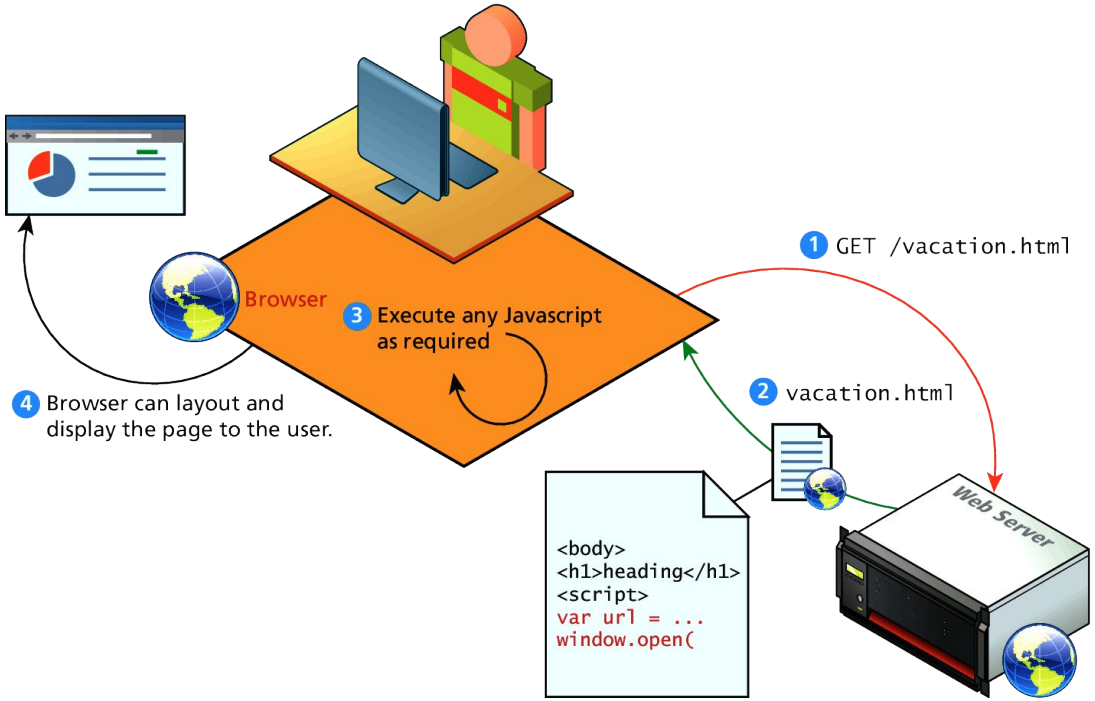
It's not Java

Although it contains the word *Java*, JavaScript and Java are vastly different programming languages with different uses. Java is a full-fledged compiled, object-oriented language, popular for its ability to run on any platform with a JVM installed.

Conversely, JavaScript is one of the world's most popular languages, with fewer of the object-oriented features of Java, and runs directly inside the browser, without the need for the JVM.

Client-Side Scripting

Let the client compute



Daniel Awde - Park Innovation

Client-Side Scripting

It's good

There are many **advantages** of client-side scripting:

- Processing can be offloaded from the server to client machines, thereby reducing the load on the server.
- The browser can respond more rapidly to user events than a request to a remote server ever could, which improves the user experience.
- JavaScript can interact with the downloaded HTML in a way that the server cannot, creating a user experience more like desktop software than simple HTML ever could.

Client-Side Scripting

There are challenges

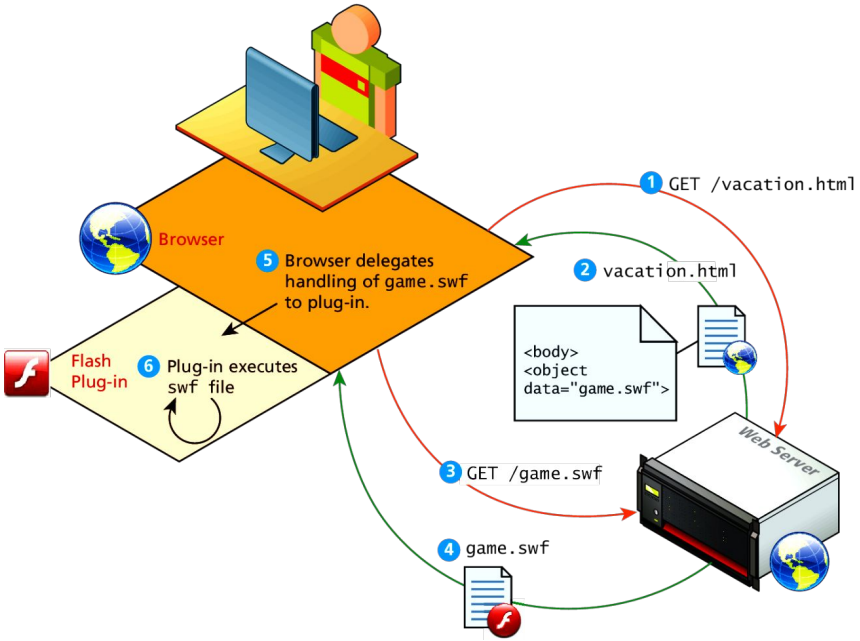
The disadvantages of client-side scripting are mostly related to how programmers use JavaScript in their applications.

- There is no guarantee that the client has JavaScript enabled
- The idiosyncrasies between various browsers and operating systems make it difficult to test for all potential client configurations. What works in one browser, may generate an error in another.
- JavaScript-heavy web applications can be complicated to debug and maintain.

Client-Side Flash

JavaScript is not the only type of client-side scripting.

- Browser Plug-ins
    - Flash

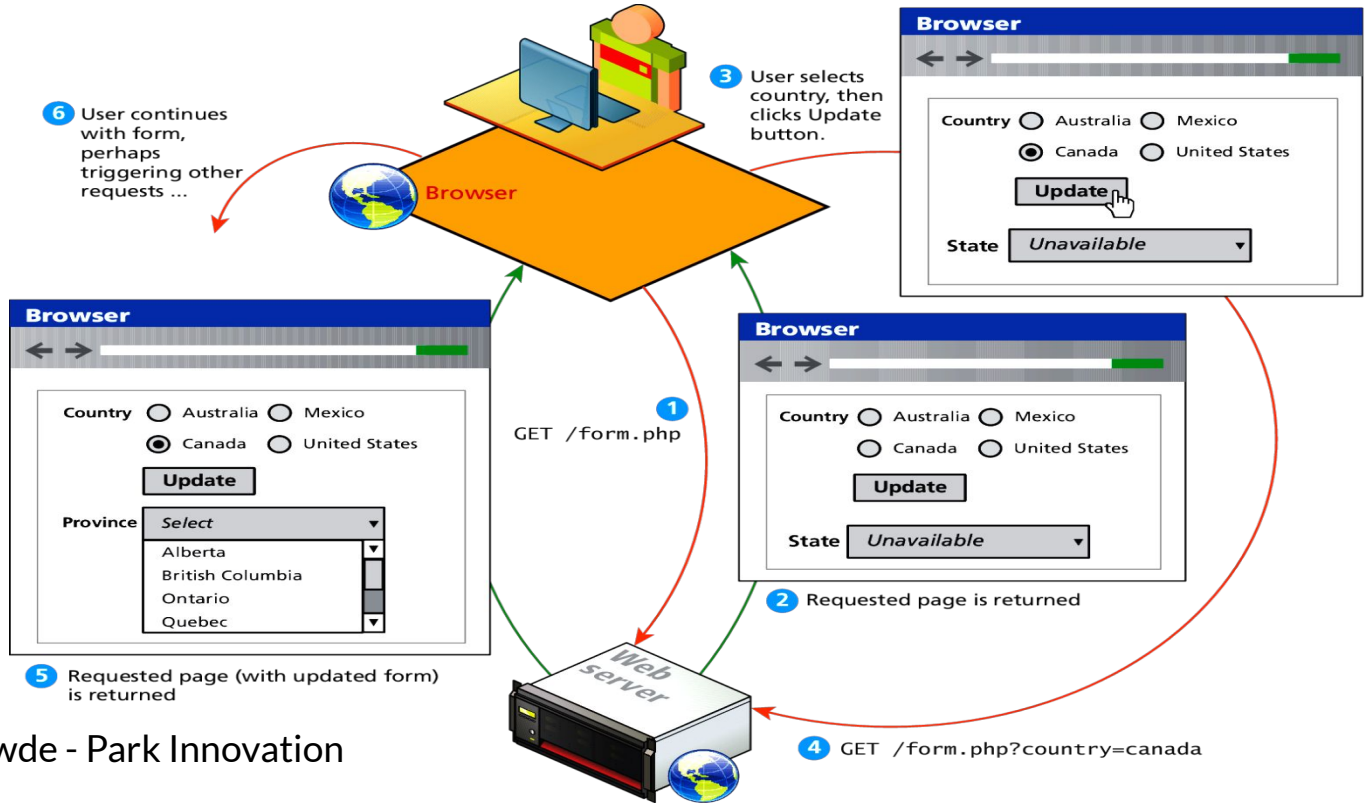


JavaScript History

- JavaScript was introduced by Netscape in their Navigator browser back in 1996.
- JavaScript is in fact an implementation of a standardized scripting language called **ECMAScript**
- JavaScript was only slightly useful, and quite often, very annoying tomany users

HTTP request-reponse loop

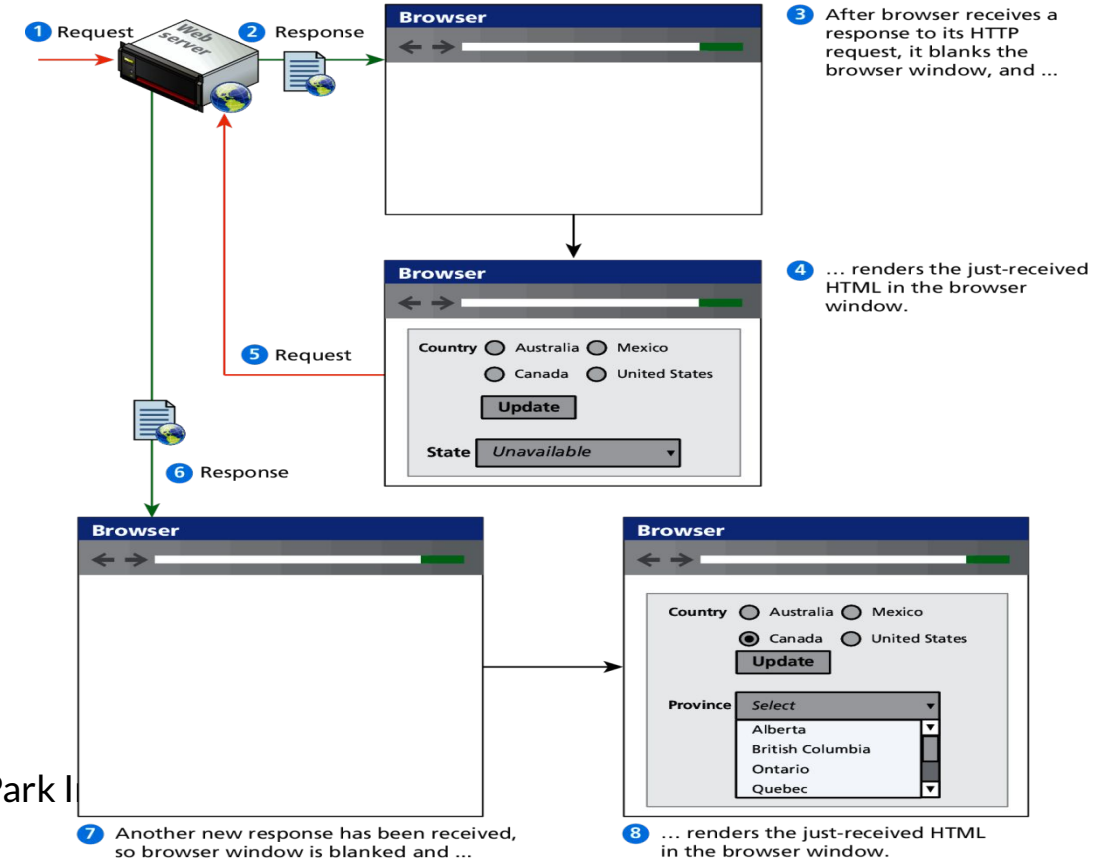
Without JavaScript



Daniel Awde - Park Innovation

HTTP request-reponse loop

Detail



JavaScript in Modern Times

AJAX

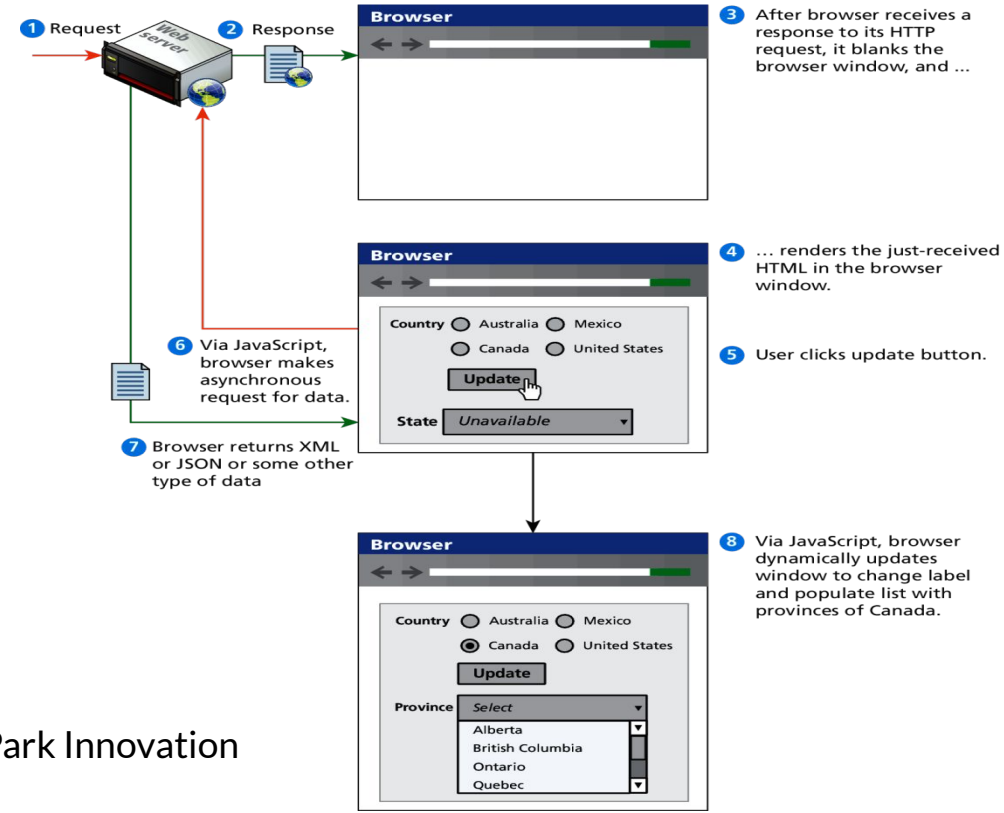
JavaScript became a much more important part of web development in the mid 2000s with **AJAX**.

AJAX is both an acronym as well as a general term.

- As an acronym it means **A**synchronous **J**avaScript **A**nd **X**ML.
- The most important feature of AJAX sites is the asynchronous data requests.

Asynchronous data requests

The better AJAX way



Daniel Awde - Park Innovation

JAVASCRIPT DESIGN PRINCIPLES

Daniel Awde - Park Innovation

Layers

They help organize

When designing software to solve a problem, it is often helpful to abstract the solution a little bit to help build a cognitive model in your mind that you can then implement.

Perhaps the most common way of articulating such a cognitive model is via the term **layer**.

In object-oriented programming, a software **layer** is a way of conceptually grouping programming classes that have similar functionality and dependencies.

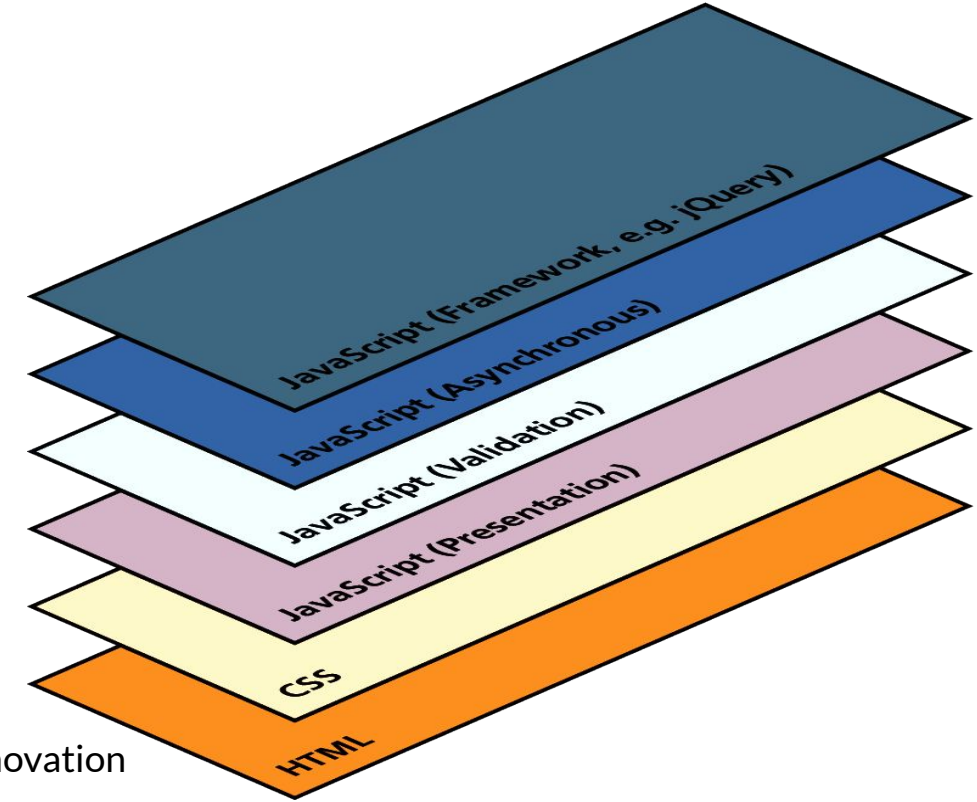
Layers

Common Layers

- **Presentation layer.** Classes focused on the user interface.
- **Business layer.** Classes that model real-world entities, such as customers, products, and sales.
- **Data layer.** Classes that handle the interaction with the data sources.

Layers

Just a conceptual idea



Daniel Awde - Park Innovation

Users Without Javascript

They do exist

- **Web crawler**. A web crawler is a client running on behalf of a search engine to download your site, so that it can eventually be featured in their search results.
- **Browser plug-in**. A browser plug-in is a piece of software that works within the browser, that might interfere with JavaScript.
- **Text-based client**. Some clients are using a text-based browser.
- **Visually disabled client**. A visually disabled client will use special web browsing software to read the contents of a web page out loud to them.

The <noscript> tag

Mechanism to speak to those with JavaScript

Any text between the opening and closing tags will only be displayed to users without the ability to load JavaScript.

It is often used to prompt users to enable JavaScript, but can also be used to show additional text to search engines.

Requiring JavaScript (or Flash) for the basic operation of your site will cause problems eventually and should be avoided.

This approach of adding functional replacements for those without JavaScript is also referred to as **fail-safe design**, which is a phrase with a meaning beyond web development.

WHERE DOES JAVASCRIPT GO?

Daniel Awde - Park Innovation

Where does JavaScript go?

JavaScript can be linked to an HTML page in a number of ways.

- Inline
- Embedded
- External

Inline JavaScript

Mash it in

Inline JavaScript refers to the practice of including JavaScript code directly within certain HTML attributes

Inline JavaScript is a real maintenance nightmare

```
<a href="JavaScript:OpenWindow();"more info</a>
<input type="button" onclick="alert('Are you sure?');" />
```

LISTING 6.1 Inline JavaScript example

Embedded JavaScript

Better

Embedded JavaScript refers to the practice of placing JavaScript code within a <script> element

```
<script type="text/javascript">
/* A JavaScript Comment */
alert ("Hello World!");
</script>
```

LISTING 6.2 Embedded JavaScript example

External JavaScript

Better

JavaScript supports this separation by allowing links to an external file that contains the JavaScript.

By convention, JavaScript external files have the extension .js.

```
<head>
  <script type="text/JavaScript" src="greeting.js">
  </script>
</head>
```

LISTING 6.3 External JavaScript example

Daniel Awde - Park Innovation

Daniel Awde - Park Innovation

SYNTAX

Daniel Awde - Park Innovation

JavaScript Syntax

We will briefly cover the fundamental syntax for the most common programming constructs including

- **variables**,
- **assignment**,
- **conditionals**,
- **loops**, and
- **arrays**

before moving on to advanced topics such as **events** and **classes**.

JavaScript's Reputation

Precedes it?

JavaScript's reputation for being quirky not only stems from its strange way of implementing object-oriented principles but also from some odd syntactic *gotchas:*

- Everything is type sensitive, including function, class, and variable names.
- The scope of variables in blocks is not supported. This means variables declared inside a loop may be accessible outside of the loop, counter to what one would expect.
- There is a === operator, which tests not only for equality but type equivalence.
- **Null** and **undefined** are two distinctly different states for a variable.
- Semicolons are not required, but are permitted (and encouraged).
- There is no integer type, only number, which means floating-point rounding errors are prevalent even with values intended to be integers.

Variables

var

Variables in JavaScript are **dynamically typed**, meaning a variable can be an integer, and then later a string, then later an object, if so desired.

This simplifies variable declarations, so that we do not require the familiar type fields like *int*, *char*, and *String*. Instead we use **var**

Assignment can happen at declaration-time by appending the value to the declaration, or at run time with a simple right-to-left assignment

Variables

Assignment

```
var x;        <-- a variable x is defined

var y = 0;    <-- y is defined and initialized to 0

y = 4;        <-- y is assigned the value of 4
```

```
/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

Condition — Value if true — Value if false

Comparison Operators

True or not True

Operator	Description	Matches (x=9)
==	Equals	(x==9) is true (x=="9") is true
===	Exactly equals, including type	(x==="9") is false (x===9) is true
< , >	Less than, Greater Than	(x<5) is false
<= , >=	Less than or equal, greater than or equal	(x<=9) is true
!=	Not equal	(4!=x) is true
!==	Not equal in either value or type	(x!=="9") is true (x!==9) is false

Logical Operators

The Boolean operators and, or, and not and their truth tables are listed in Table 6.2. Syntactically they are represented with && (and), || (or), and ! (not).

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

AND Truth Table

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

OR Truth Table

A	! A
T	F
F	T

NOT Truth Table

TABLE 6.2 AND, OR, and NOT Truth Tables

Conditionals

If, else if, …, else

JavaScript's syntax is almost identical to that of PHP, Java, or C when it comes to conditional structures such as if and if else statements. In this syntax the condition to test is contained within ( ) } blocks.

```
var hourOfDay;   // var to hold hour of day, set it later...
var greeting;   // var to hold the greeting message.
if (hourOfDay > 4 && hourOfDay < 12){
   // if statement with condition
   greeting =  "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 20){
   // optional else if
   greeting =  "Good Afternoon";
}
else{ // optional else branch
   greeting = "Good Evening";
}
```

LISTING 6.4 Conditional statement setting a variable based on the hour of the day

Loops

Round and round we go

Like conditionals, loops use the ( ) and { } blocks to define the condition and the body of the loop.

You will encounter the **while** and **for** loops

While loops normally initialize a **loop control variable** before the loop, use it in the condition, and modify it within the loop.

```
var i=0;  // initialise the Loop Control Variable

while(i < 10){ //test the loop control variable

        i++;  //increment the loop control variable

}
```

For Loops

Counted loops

A **for loop** combines the common components of a loop: initialization, condition, and post-loop operation into one statement.

This statement begins with the **for** keyword and has the components placed between **( )** brackets, semicolon (;) separated as shown

```
for (var i = 0; i < 10; i++){
  //do something with i
}
```

Functions

Functions are the building block for modular code in JavaScript, and are even used to build **pseudo-classes**, which you will learn about later.

They are defined by using the reserved word **function** and then the function name and (optional) parameters.

Since JavaScript is dynamically typed, functions do not require a return type, nor do the parameters require type.

Functions

Example

Therefore a function to raise x to the yth power might be defined as:

```
function power(x,y){
        var pow=1;
        for (var i=0;i<y;i++){
                pow = pow*x;
        }
        return pow;
}
```

And called as

```
power(2,10);
```

Alert

Not really used anymore, console instead

The alert() function makes the browser show a pop-up to the user, with whatever is passed being the message displayed. The following JavaScript code displays a simple hello world message in a pop-up:

```
alert ( "Good Morning" );
```

Using alerts can get tedious fast. When using debugger tools in your browser you can write output to a log with:

```
console.log("Put Messages Here");
```

And then use the debugger to access those logs.

Errors using try and catch

When the browser's JavaScript engine encounters an error, it will *throw* an **exception**. These exceptions interrupt the regular, sequential execution of the program and can stop the JavaScript engine altogether. However, you can optionally catch these errors preventing disruption

```
try {
  nonexistantfunction("hello");
}
catch(err) {
  alert("An exception was caught:" + err);
}
```

LISTING 6.5 Try-catch statement

Throw your own

Exceptions that is.

Although try-catch can be used exclusively to catch built-in JavaScript errors, it can also be used by your programs, to throw your own messages. The throw keyword stops normal sequential execution, just like the built-in exceptions

```
try {
   var x = -1;
   if (x<0)
      throw "smallerthan0Error";
}
catch(err){
   alert (err + "was thrown");
}
```

LISTING 6.6 Throwing a user-defined exception

Tips

With Exceptions

Try-catch and throw statements should be used for *abnormal* or *exceptional* cases in your program.

Throwing an exception disrupts the sequential execution of a program. When the exception is thrown all subsequent code is not executed until the catch statement is reached.

This reinforces why try-catch is for exceptional cases.

JAVASCRIPT EVENTS

Daniel Awde - Park Innovation

JavaScript Events

A JavaScript **event** is an action that can be detected by JavaScript.

We say then that an event is *triggered* and then it can be *caught* by JavaScript functions, which then do something in response.

JavaScript Events

A brave new world

In the original JavaScript world, events could be specified right in the HTML markup with *hooks* to the JavaScript code (and still can).

As more powerful frameworks were developed, and website design and best practices were refined, this original mechanism was supplanted by the **listener** approach.

JavaScript Events

Two approaches

Old, Inline technique

```
...
 <script type="text/javascript"  src="inline.js"></script>
...

<form name='mainForm' 'onsubmit="validate(this);">
      <input name="name" type="text" onhover="hover(this);" onfocus="focus(this);">
      <input name="email" type="text" onhover="hover(this);" onfocus="focus(this);">
      <input type="submit" onclick="validate(this);">
...
```

New, Layered Listener technique

```
...
 <script type="text/javascript"  src="listener.js"></script>
...

<form name='mainForm'>
      <input name="name" type="text">
      <input name="email" type="text">
      <input type="submit">
...
```

Inline Event Handler Approach

For example, if you wanted an alert to pop-up when clicking a you might program:

<div id="example1" **onclick**="alert('hello')">Click for pop-up

The problem with this type of programming is that the HTML markup and the corresponding JavaScript logic are woven together. It does not make use of layers; that is, it does not separate content from behavior.

Listener Approach

Two ways to set up listeners

```
var greetingBox = document.getElementById('example1');
greetingBox.onclick = alert('Good Morning');
```

LISTING 6.10 The "old" style of registering a listener.

```
var greetingBox = document.getElementById('example1');
greetingBox.addEventListener('click', alert('Good Morning'));
greetingBox.addEventListener('mouseOut', alert('Goodbye'));

// IE 8
greetingBox.attachEvent('click', alert('Good Morning'));
```

LISTING 6.11 The "new" DOM2 approach to registering listeners.

Listener Approach

Using functions

What if we wanted to do something more elaborate when an event is triggered? In such a case, the behavior would have to be encapsulated within a function, as shown in Listing 6.12

```
function displayTheDate() {
   var d = new Date();
   alert ("You clicked this on "+ d.toString());
}
var element = document.getElementById('example1');
element.onclick = displayTheDate;

// or using the other approach
element.addEventListener('click',displayTheDate);
```

LISTING 6.12 Listening to an event with a function

Listener Approach

Anonymous functions

An alternative to that shown in Listing 6.12 is to use an anonymous function (that is, one without a name), as shown in Listing 6.13.

```
var element = document.getElementById('example1');
element.onclick = function() {
   var d = new Date();
   alert ("You clicked this on " + d.toString());
};
```

LISTING 6.13 Listening to an event with an anonymous function

Event Object

No matter which type of event we encounter, they are all **DOM event objects** and the event handlers associated with them can access and manipulate them. Typically we see the events passed to the function handler as a parameter named *e*.

```
function someHandler(e) {
    // e is the event that triggered this handler.
}
```

Event Object

Several Options

- **Bubbles**. If an event's bubbles property is set to true then there must be an event handler in place to handle the event or it will bubble up to its parent and trigger an event handler there.
- **Cancelable**. The Cancelable property is also a Boolean value that indicates whether or not the event can be cancelled.
- **preventDefault**. A cancelable default action for an event can be stopped using the preventDefault() method in the next slide

Event Object

Prevent the default behaviour

```
function submitButtonClicked(e) {
  if(e.cancelable){
    e. preventDefault();
  }
}
```

LISTING 6.14 A sample event handler function that prevents the default event

Event Types

There are several classes of event, with several types of event within each class specified by the W3C:

- mouse events
- keyboard events
- form events
- frame events

Mouse events

Event	Description
onclick	The mouse was clicked on an element
ondblclick	The mouse was double clicked on an element
onmousedown	The mouse was pressed down over an element
onmouseup	The mouse was released over an element
onmouseover	The mouse was moved (not clicked) over an element
onmouseout	The mouse was moved off of an element
onmousemove	The mouse was moved while over an element

Keyboard events

Event	Description
onkeydown	The user is pressing a key (this happens first)
onkeypress	The user presses a key (this happens after onkeydown)
onkeyup	The user releases a key that was down (this happens last)

Keyboard events

Example

<input type="text" id="keyExample">

The input box above, for example, could be listened to and each key pressed echoed back to the user as an alert as shown in Listing 6.15.

```
document.getElementById("keyExample").onkeydown = function
myFunction(e){
   var keyPressed=e.keyCode;      //get the raw key code
   var character=String.fromCharCode(keyPressed); //convert to string
   alert("Key " + character + " was pressed");
}
```

LISTING 6.15 Listener that hears and alerts keypresses

Form Events

Event	Description
onblur	A form element has lost focus (that is, control has moved to a different element, perhaps due to a click or Tab key press.
onchange	Some <input>, <textarea> or <select> field had their value change. This could mean the user typed something, or selected a new choice.
onfocus	Complementing the onblur event, this is triggered when an element gets focus (the user clicks in the field or tabs to it)
onreset	HTML forms have the ability to be reset. This event is triggered when that happens.
onselect	When the users selects some text. This is often used to try and prevent copy/paste.
onsubmit	When the form is submitted this event is triggered. We can do some pre-validation when the user submits the form in JavaScript before sending the data on to the server.

Form Events

Example

```
document.getElementById("loginForm").onsubmit = function(e){
   var pass = document.getElementById("pw").value;
   if(pass==""){
      alert ("enter a password");
      e.preventDefault();
   }
}
```

LISTING 6.16 Catching the onsubmit event and validating a password to not be blank

Frame Events

Frame events are the events related to the browser frame that contains your web page.

The most important event is the **onload** event, which tells us an object is loaded and therefore ready to work with. If the code attempts to set up a listener on this not-yet-loaded , then an error will be triggered.

```
window.onload= function(){

        //all JavaScript initialization here.

}
```

Frame Events

Table of frame events

Event	Description
onabort	An object was stopped from loading
onerror	An object or image did not properly load
onload	When a document or object has been loaded
onresize	The document view was resized
onscroll	The document view was scrolled
onunload	The document has unloaded

FORMS

Daniel Awde - Park Innovation

Validating Forms

You mean pre-validating right?

Writing code to prevalidate forms on the client side will reduce the number of incorrect submissions, thereby reducing server load.

There are a number of common validation activities including email validation, number validation, and data validation.

Daniel Awde - Park Innovation

Validating Forms

Empty field

```
document.getElementById("loginForm").onsubmit = function(e){
   var fieldValue=document.getElementByID("username").value;
   if(fieldValue==null || fieldValue== ""){
      // the field was empty. Stop form submission
      e.preventDefault();
      // Now tell the user something went wrong
      alert("you must enter a username");
   }
}
```

LISTING 6.18 A simple validation script to check for empty fields

Validating Forms

Empty field

If you want to ensure a checkbox is ticked, use code like that below.

```
var inputField=document.getElementByID("license");
if (inputField.type=="checkbox"){
        if (inputField.checked)
                //Now we know the box is checked
}
```

Validating Forms

Number Validation

```
function isNumeric(n) {
    return !isNaN(parseFloat(n)) && isFinite(n);
}
```

LISTING 6.19 A function to test for a numeric value

Submitting Forms

Submitting a form using JavaScript requires having a node variable for the form element. Once the variable, say, formExample is acquired, one can simply call the submit() method:

```
var formExample = document.getElementById("loginForm");

formExample.submit();
```

This is often done in conjunction with calling **preventDefault()** on the onsubmit event.

Variables

```
var name = expression;                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                       JS
```

- types are not specified, but JS does have types ("loosely typed")
  - `Number, Boolean, String, Array, Object, Function, Null, Undefined`
  - can find out a variable's type by calling `typeof`

let

- The let declaration declares a block-scoped local variable, optionally initializing it to a value.

```
let x = 1;
if (x === 1)
{  let x = 2;  console.log(x);
 // expected output: 2}
console.log(x);// expected output: 1
```

Syntax

```
let name1 [= value1] [, name2 [= value2]] [, ..., nameN [= valueN];
```

- let allows you to declare variables that are limited to the scope of a block statement, or expression on which it is used, unlike the var keyword, which declares a variable globally, or locally to an entire function regardless of block scope. The other difference between var and let is that the latter is initialized to a value only when a parser evaluates it

let vs var

- Unlike var, let begins declarations, not statements. That means you cannot use a lone let declaration as the body of a block (which makes sense, since there's no way to access the variable).

- if (true) let a = 1; // SyntaxError: Lexical declaration cannot appear in a single-statement context

Section 10 of 10

Dev tools

Daniel Awde - Park Innovation

Developer Tools

Help is on the way

Developer tools in current browsers make it significantly easier to examine and troubleshot 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.

Chrome DevTools

- Built-in Debugging tool in Google Chrome
- Why do you need to know this?
  - To debug your code
  - To set Javascript breakpoint
  - To find any kind of layout issues

Accessing DevTools

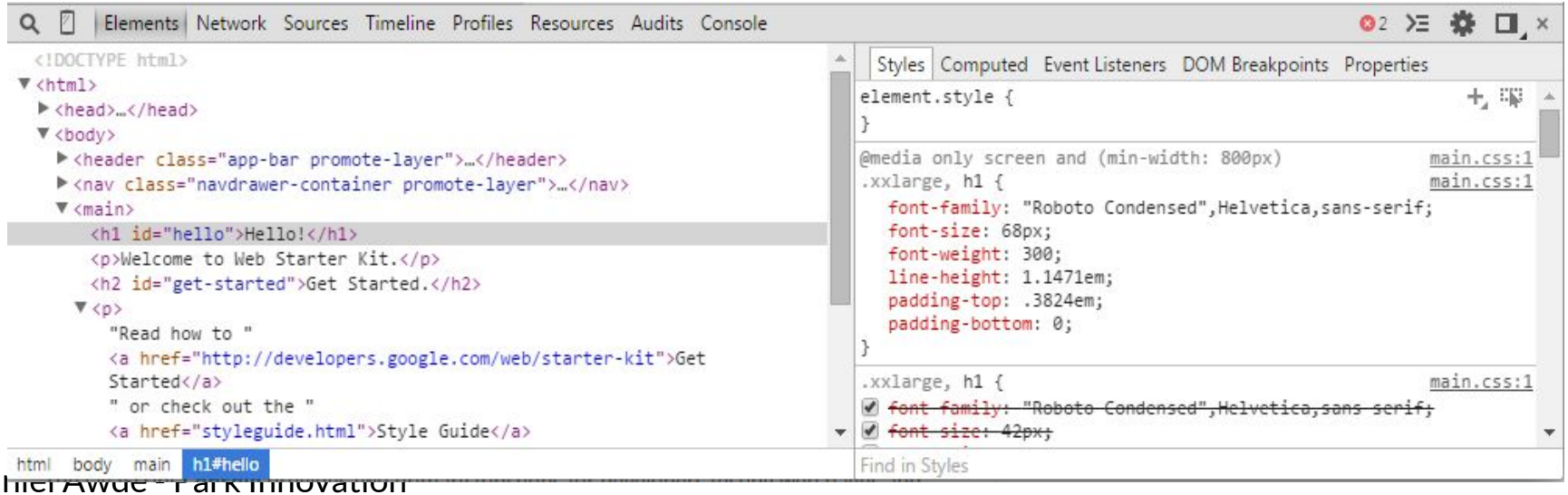
- Go to Chrome Menu. Select Tools -> Developer Tools
- Right – click on any page element and select inspect element.

Shortcuts:

- Ctrl+Shift+I (Cmd + Opt + I on Mac) to open DevTools
- Ctrl+Shift+J (Cmd + Opt + J on Mac) to open Console
- Ctrl+Shift+C (Cmd + Shift + C on Mac) to open Devtools in Inspect element mode

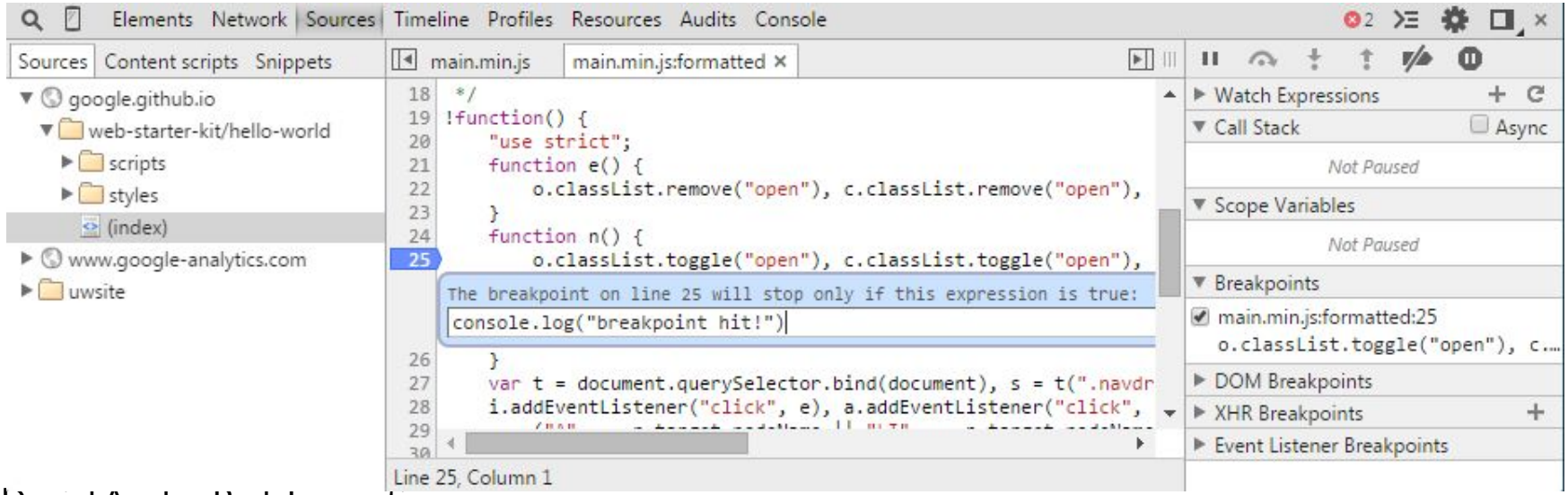
DevTools Window - ELEMENT

Elements Panel: Inspect and Live-edit the HTML and CSS of the webpage.

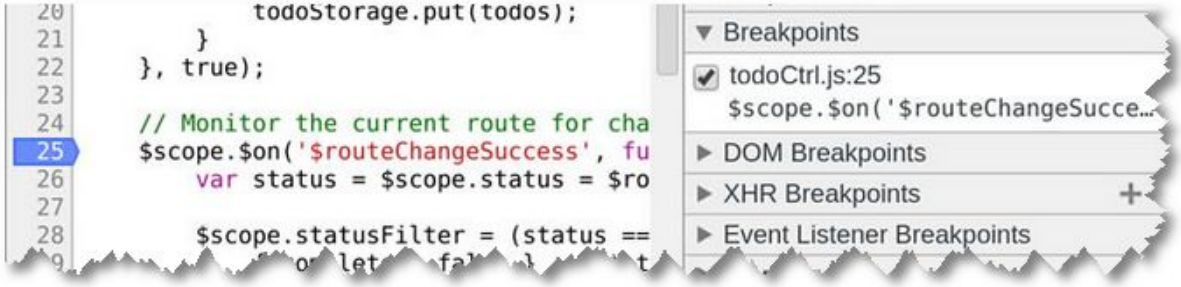


DevTools Window - SOURCES

Sources Panel: Used to debug your JavaScript code using breakpoints.



Setting up a breakpoint



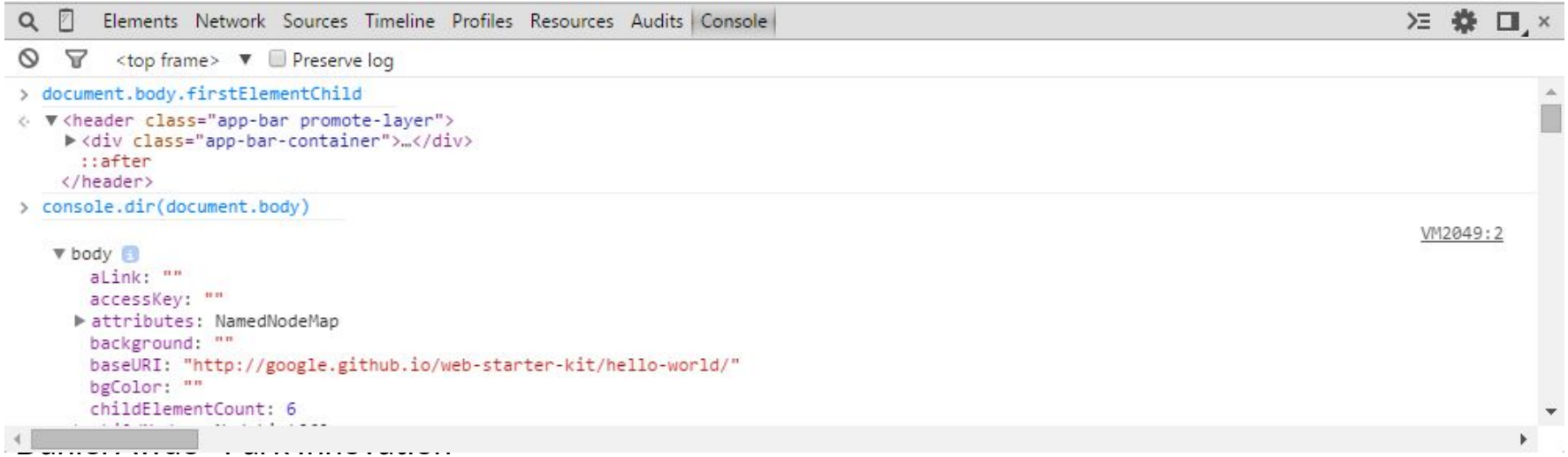
Breakpoint on a line of code

Breakpoint on JavaScript Event Listener

Daniel Awde - Park Innovation

DevTools Window - CONSOLE

Console Panel: Used to log diagnostic information in the development process and provide shell prompt which can be used to interact with the document and DevTools.



DevTools Window - CONSOLE

Simulating Mobile Devices

Working with Element Panel

Daniel Awde - Park Innovation

JavaScript Playgrounds

Code playgrounds are online code editors that are available publicly. They are more like services that allow you to create, edit, share, fork a snippet of code, and much more.

-Using JS Fiddle

-Using Code Pen

-Using JS BIN

-Using Plunker

-Using LiveWeave

-Using Dabblet

Daniel Awde - Park Innovation