

Learning XHTML / CSS

- XHTML and CSS are easy to write
 - Simple syntax
 - Relatively little to memorize
 - Tutorials abound on the web
- <http://www.w3schools.com/>
<http://reference.sitepoint.com/css>
<http://www.csszengarden.com/>
- They are also easy to write **poorly!**
 - The focus here is more on best practice for organizing and using XHTML and CSS



Understanding Roles

- XHTML (extensible hypertext markup language) -
organize content
- CSS (cascading style sheets) -
content presentation

Structure (XHTML)
Presentation (CSS)
Behavior (JS)

MVC analogy -
XHTML (model)
CSS (view)
JS (controller)



Writing good XHTML (valid)

- Don't use deprecated HTML!
 - example: ``, ``, `<i>`
- Case sensitive – tags are lower case
- Attributes in quotes
- Close all tags (`` or `<p> ... </p>`)
- Properly nest tags
 - `<p> ... </p> == BAD!`
- Use a validation tool



Writing good XHTML (semantic)

- Organize content in most logical order

Try to think in terms of a document

- Use tags that fit with/describe the content

- An example would be using an unordered list for navigation instead of a table.

Navigation is truly a “list” of information and not tabular data.

- When semantic XHTML elements aren't enough try XHTML compounds
 - When XHTML compounds aren't enough use meaningful class names

- Understand block level and inline elements

Id vs. Class

- What's the difference?
 - Id specified once per page and has a higher inheritance specificity (important later for *cascade* behaviour).

Use for labeling page sections or special/unique elements

- Class is reusable

Use for semantic categorization

class = 'blue-text' → describes presentation

class = 'external-link' → semantic value



Why semantics matters

- Accessibility
 - Increases readability for people and software
(Search engines, functional testing or screen readers)
- Maintainability
 - Clean (as in less clutter)
 - Self-documenting
 - Easy modification
 - Better collaboration potential



Using CSS

- External Style Sheet

```
<link href="stylesheet" type="text/css" href="location.css" />
```

Preferred method

- Also takes a “media” descriptor (screen, print, handheld, etc)

- Embedded Styles

```
<style type="text/css">  
  body {  
    ...  
  }  
</style>
```

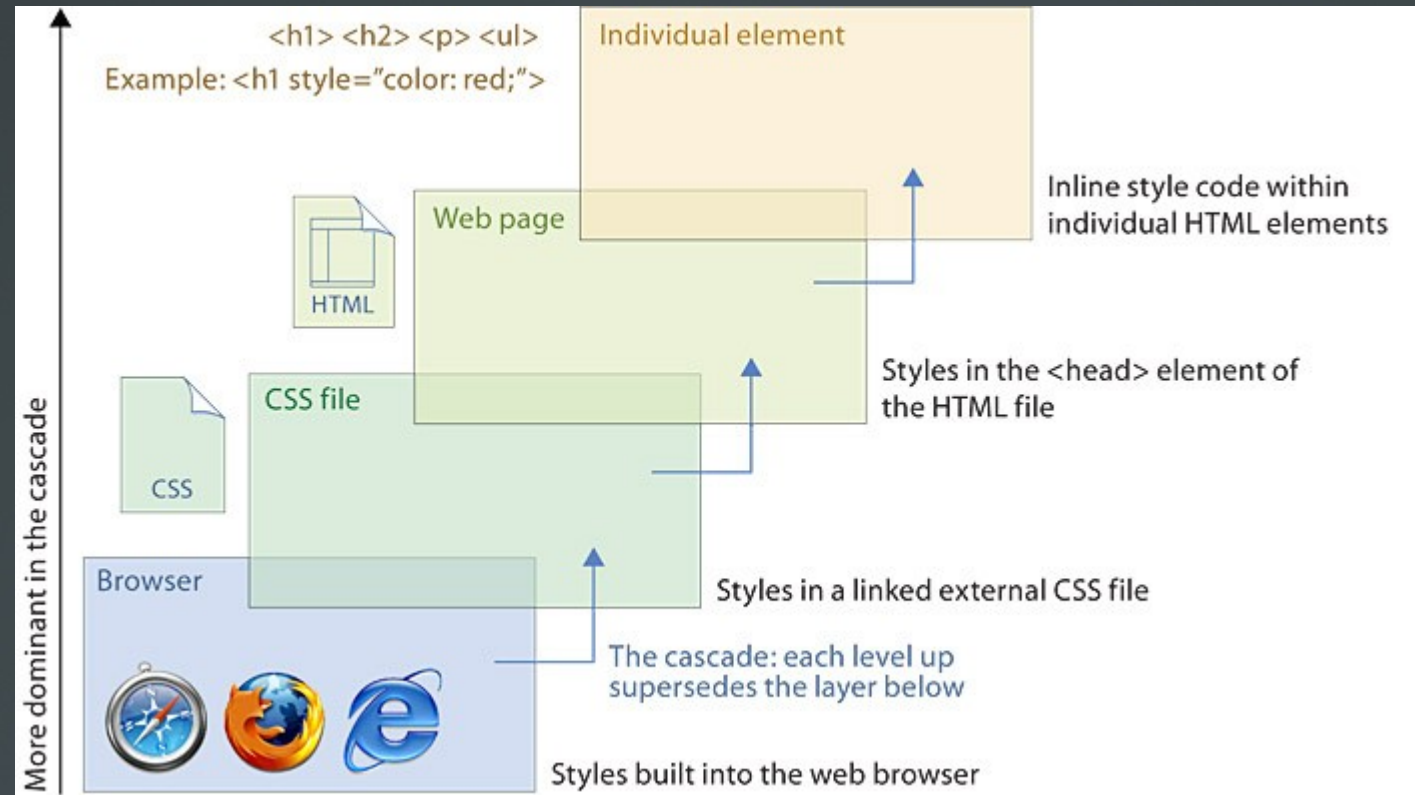
- In-line Styles

```
<p style="font-size: 12px">Lorem ipsum</p>
```



CSS – the Cascade!

- The power of CSS is found in the “cascade” which is the combination of the browser’s default styles, external style sheets, embedded, in-line, and even user-defined styles.
- The cascade sets priorities on the individual styles which effects inheritance.



CSS syntax tips

■ Group selectors

```
h1, h2, h3, h4, h5, h6 {  
  font-family: "Trebuchet  
  MS", sans-serif;  
}
```

```
.menu-item a, .menu-item  
a:selected {  
  text-decoration:none;  
}
```

■ Inheritance

```
body {  
  color: blue;  
}  
p.non-blue {  
  color:red;//inherits blue  
}
```

■ Shorthand

- Consolidate many styles into a single declaration.

```
font-family: veranda, sans-  
serif;
```

```
font-weight: bold;
```

```
font-size: 12px;
```

```
→ font: bold 12px veranda,  
sans-serif;
```

```
padding-top: 5px;
```

```
padding-right: 8px;
```

```
padding-bottom: 5px;
```

```
padding-left: 10px;
```

```
→ padding: 5px 8px 5px 10px;
```

More about CSS Inheritance

- Allows elements to “inherit” styles from parent elements.
- Helpful in reducing the amount of CSS to set styles for child elements.
- Unless a more specific style is set on a child element, the element looks to the parent element for its styles.
- Each style has a numeric specificity value that is given based on its selector.



Advanced CSS Selectors

- Descendant Selector

```
body h1 { }
```

```
#navigation p { }
```

- Adjacent Sibling Selectors

```
p.intro + span { }
```

- Child Selectors

```
div ol > p { }
```

- Pseudo-Class Selectors

```
a:active { }
```

```
#nav:hover { }
```

- Attribute Selectors

```
div[href="http://home.org"]
```

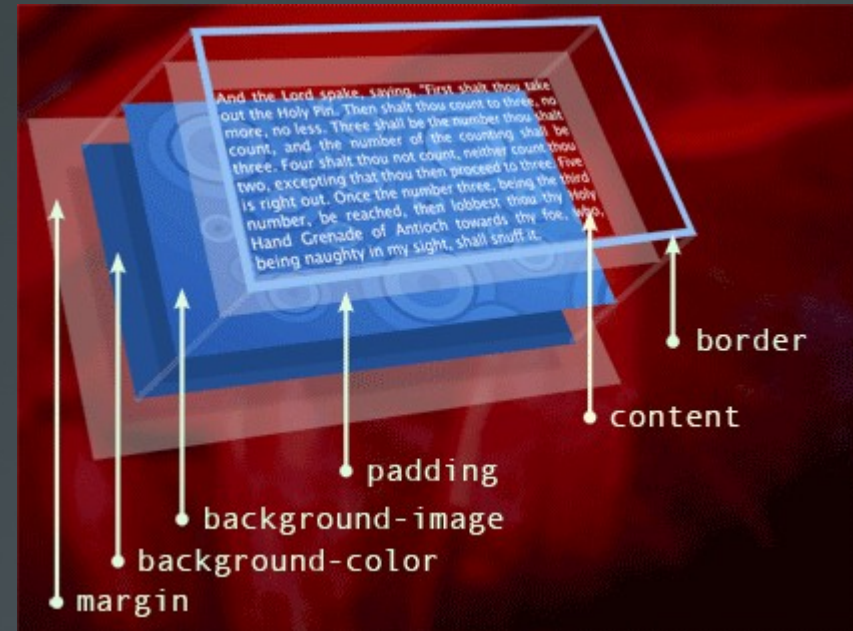
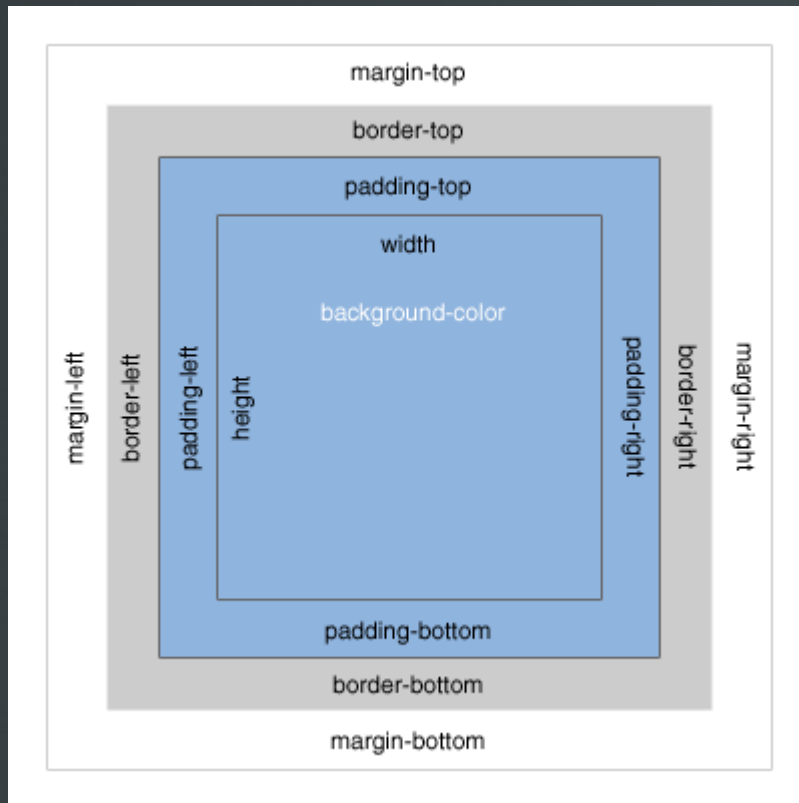
- Universal Selector

```
* { }
```



The Box Model

Every element in the DOM (Document Object Model) has a conceptual “box” for presentation.



The box consists of margin, padding, border, content (width, height), and offset (top, left)

The Box Model (cont.)

- The primary concept for page layout

Tables have a place but this isn't it

- Blocks vs. Inline

Block and inline are properties of HTML elements – slightly different meaning for CSS

Understand the CSS display property and how it manipulates but does not change the HTML

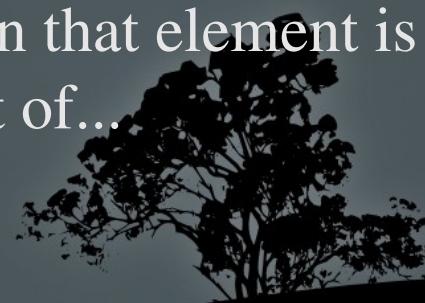
Floating or absolutely positioning can transform an element in unexpected ways



CSS Positioning

- **Relative**
 - control position relative to normal position in document flow and element remains in document flow
- **Absolute**
 - Removes the element from the normal document flow
 - Position element absolutely (relative to containing block.)
containing block should be **explicitly** positioned
- **Float**
 - Behaves alot like absolute positioning in that element is removed from document flow ... sort of...
 - Clear is the magic companion of float

(Fixed and static useful but rarely used)



Staying organized & CSS

- CSS can get out of hand fast

One giant css file with no clear order to it is obviously not ideal

There is no standard for organizing

- Tips & Tricks

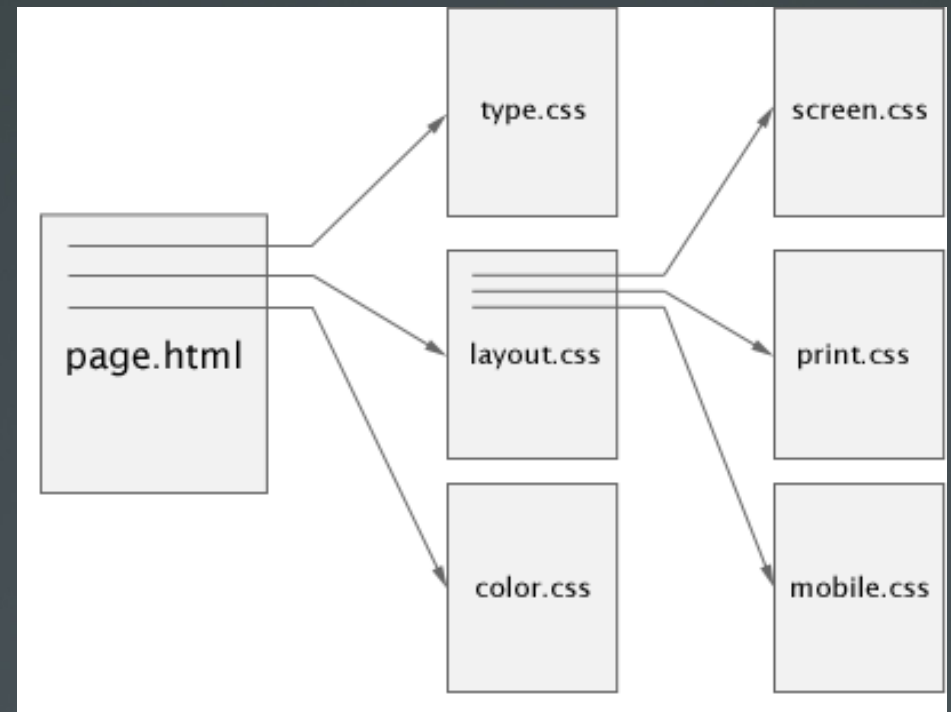
use good naming conventions

use comment flags to break up file

```
/* =drop-down-menu
```

consider modular approach

do something rather than nothing...



```
/* main layout */  
@import url("layout.css");
```



Resources

- <http://www.w3schools.com/>
- <http://reference.sitepoint.com/css>
- <http://www.csszengarden.com/>

