Cascading Style Sheets

Agenda

- Introduction
  - Syntax
  - Inclusion & Inheritance
- Positioning & Visibility
  - Element flow
  - Positioning
  - Floating
- Selectors
  - Grouping & Nesting selectors
  - Relational selectors (adjacent, child, sibling, etc)
  - Pseudo-class selectors
- CSS Techniques
  - Mountain top corner boxes
  - Tabbed navigation
  - Drop shadows
  - Replacing text with images
  - CSS3 border radius & gradients
  - Clipping & Masking
- Transformations & Animations
  - 2D & 3D Transforms
  - Transitions
  - Keyframe animation
- Text & Fonts
  - Text effects
  - Web fonts

Agenda

- Object Oriented CSS
- Responsive UI
- LESS CSS
- Image Manipulation
- Performance & Optimization

CSS – Intro

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Any document has 2 aspects – information & style
  - HTML's role is to define information / content
  - Styles define **how to display** HTML elements
- CSS is a W3C recommendation
  - Current version is 2.1 (released in 1998)
  - CSS3 is yet to be released officially, though most browsers already implement most of its recommendations

A Style Sheet

- The aim of CSS is to give the developer control on how a page is displayed by the browser

- A **style sheet** is a set of **rules** that controls the formatting of HTML elements
  - The appearance of an HTML page can be changed by changing the style sheet associated with it
  - There is no need to make detailed changes within the HTML to change it looks

The Cascading Style Sheet

- Some of the advantages of using style sheets are:
  - Separation of style and content
  - Different styling can be provided for different users / devices

- A website may use a common style sheet
- A set of web pages, in turn, may use a further refined style sheet
- A web page, again, may have its own style sheet that refines the information in the previous style sheet
- Thus style sheets **cascade**

CSS Syntax

- A CSS rule has 2 parts:
  1. A **selector** – defines which HTML elements are controlled by the rule
  2. A **declaration** – says what the required effect is

- Example:

  h1 {color: blue;}
  - h1 is the selector
  - all that follows between {…} are the declarations

  - Selectors are the target elements in HTML & declarations are the styles that is applied on the "selected" elements

CSS Declarations

```
selector {property1:value1; property2:value2;}
```

- CSS declarations are a collection of properties
  - Each property is made up of the property name & property value
  - The name & value is separated by a colon
  - The property pairs are separated by a semi-colon
  - All the declarations are enclosed within { and }
  - Example:

```
{
margin:0px;
padding:0px;
font-family:Arial;
}
```

Grouping Elements & Styles

- The elements of a document can be grouped by the HTML **class** attribute
- For example:

```
<h1 class="firstsection"> Section One Heading </h1>
<h1 class="secondsection"> Section Two heading </h1>
<h1 class="thirdsection">Section Three Heading </h1>
```

- The style then can be defined in CSS by the className:

```
.firstsection {color: red}
.secondsection {color: green}
.thirdsection {color: blue}
```

Classes

- In CSS, a class name
  - Is identified by a period (**.**)
  - starts with an alphabet followed by alphabets, numbers, or hyphens
- A class can be attributed to more than one HTML element in the document:

```
.introText { font-weight: bold; color: red; }
<h1 class="introText">Welcome</h1>
<p class="introText">The para content…</p>
```

Classes…

- An HTML element can have more than one class attributed to it

```
.mine {color:red}
.his {color:green}
.code {font-family:Courier}
.ref {font-style:italic}

<p class="mine code"> x=y </p>
<p class="his ref"> [16] </p>
<p class="mine ref"> [18] </p>
<p class="his code"> y=3 </p>
```

Identifiers

- **id** attributes in HTML give another level of naming
  - In addition, **id** attributes are unique and are associated with a single element
  - Any element in HTML cannot have more than one id attributed to it

- In CSS, id names
  - Are identified by a **#** sign
  - It consist of an initial letter followed by alphabets, numbers, hyphens

- Example:

```
#wrapper { padding:12px; border:1px solid #ccc; }
<div id="wrapper">content goes here...</div>
```

Linking Stylesheets to HTML

CSS Inclusion

- There are 3 ways of including a style sheet:
  - External style sheet
  - Internal style sheet
  - Inline style

- **External style sheet**
  - All style definitions are stored in an external file with extension .css
  - The external file is included in the HTML file by using **<link>** tag
  - `<link href="style.css" rel="stylesheet" type="text/css" />`

CSS Inclusion…

- **Internal style sheet**
  - All style definitions are included within the **<style>** tag, in the HTML document itself, usually defined within the <head> section

```
<style type="text/css">
h1 {color:blue; font-size:12px;}
</style>
```

- **Inline style sheet**
  - Adds styles to specific HTML elements using the **style** attribute
  - `<p style="color:green; text-align:justify;">Turpis dapibus egestas nisi et phasellus</p>`

The Target Media

- The <link> and <style> tags have an optional **media** attribute, which specifies those media to which the style sheet should be applied

- Possible value that the media attribute can have are:
  - **print** : for output to a printer
  - **screen** : for presentation on computer screens
  - **braille** : for presentation on Braille tactile feedback devices
  - **handheld** : for small monochrome screens (phone, PDA)
  - **tv** : for low resolution devices with limited scrolling
  - **tty** : for limited devices with fixed width characters
  - **all** : for all output devices (the default)

CSS Media

- Example:

```
<link href="style.css" type="text/css" media="screen">
<link href="another.css" type="text/css"
media="screen, print">
```

OR

```
<style type="text/css" media="screen">
body { background:url(grass.gif) red; color: black }
p em { background: yellow; color: black}
.note { margin-right: 1in; margin-left: 1in }
</style>
```

Multiple Stylesheets

- You can have more than 1 style sheet in the HTML
  - If the multiple stylesheets have common selectors & properties, then the CSS that loads last, overrides the styles of the previously loaded CSS
  - Only the common properties are overridden, while others are combined
  - For example,
    - styleA.css has p **{font-size: 12px; color:green;}**
    - styleB.css has p **{font-weight:bold; color:blue;}**
    - If styleA.css is declared earlier than styleB.css in the HTML, then the final application will be
      p **{font-size:12px; font-weight:bold; color:blue;}**

Importing a Style Sheet

- **@import** directive is used to import a style sheet
  - This statement may appear in a .css file or inside the style element
- Example:

```
@import url(/css/system.css);
@import url(../css/local.css);
p {background: yellow; color: black }
```

  - All @import statements **must** occur at the start of the style sheet
  - All styles defined after the import will override the styles in imported files (for the same selectors and properties)
  - The order in which style sheets are imported is important as it determines how cascading takes place

Cascading Order & Inheritance

- Style inheritance is implicit
  - Styles of the container elements are applied to the inner elements
  - For example, if body is selected and assigned color:red, then all text elements in elements within the body are also applied the same color, unless overridden

- Inline styles override internal & external style sheets

Cascading Order…

- The priority of a particular style rule can be enhanced by the **!important** attribute

- Example:

```
<style type="text/css">
p { color:green !important; }
</style>
```

  - Usually applied in internal styles
  - In which case, it overrides the inline styles

Browser Specific CSS

- Most browsers provide additional style properties that work only in those browsers
  - Usually due to lack of mandate in the W3C recommendations
  - Are usually non-official properties

- Browser property names:
  - Mozilla Firefox property names start with –moz-{property}
  - Chrome / Safari property names start with –webkit-{property}
- Example:

```
p { -webkit-margin-before:0px; }
p { -moz-margin-start:0px; }
```

Element Flow

HTML Structure

- HTML body elements are of two types:
- **Block-level**
  - Terminates the previous element & effectively starts its own new line
  - Any element added at the end of block-level elements are started in new lines
  - Example: <h1> or <div> or <p>

- **Inline**
  - Do not terminate the previous element and form part of the previous element
  - Example: <em> or <a>

- Inline elements when embedded inside a block-level element, it usually inherit the style of the parent block-level element

HTML Structure Example

```
<body>

<div>Lectus parturient in rhoncus! <a
   href="http://google.com">Et ac augue?</a> In
   urna ridiculus mauris eros. Turpis magnis mid, vel,
   rhoncus ut habitasse, duis, a ridiculus.</div>

<p>Turpis dapibus egestas nisi et phasellus. Dignissim
   tincidunt penatibus, <em>integer porttitor sit
   et</em>, hac dolor? Mid odio, ridiculus mauris!</p>

</body>
```

Display

- **display** property describes how an element is displayed

- Syntax:
  display: block | inline | list-item | run-in | inline-block | none
  - Default display property is set on elements based on their type – block-level or inline
  - Display property can be used to change that behaviour

- Example:

```
span { display:block; } /* makes span a block element */
li { display:inline; } /* makes bullets align horizontally */
```

Display...

- Display property's behaviour:
  - **list-item** : same as block except a list-item marker is added
  - **run-in** : can be inline or block depending on the context
  - **inline-block** : Acts as a block-level element but flows like an inline box
  - **none** : no display at all, hides the element from display

- Example:

```
/* without inline-block, span would not get the width / height */
span {width:150px; height:100px; display:inline-block;}

<span>span 1</span><span>span 2</span><span>span 3</span>
```

  - Setting width or height on inline elements have no effect, inline-block enforces that, while keeping its flow inline

Visibility

- **visibility** property specifies if an element should be visible or hidden

- Syntax:
  visibility: hidden

- Example:

```
p { visibility: hidden; }
<p>Etiam est dictumst placerat ultrices magna?
   Parturient aenean mus? Augue dignissim.</p>
```

display:none Vs visibility:hidden

```
<style type="text/css">
.vis-hid {visibility:hidden;}
.disp-non {display:none;}
</style>
<ul>
  <li>item 1</li>
  <li class="vis-hid">item 2</li>
  <li>item 3</li>
  <li class="disp-non">item 4</li>
  <li>item 5</li>
</ul>
```

- display:none and visibility:hidden both result in element not being visible
- What is the difference between display:none & visibility:hidden?

Box Properties

Float

- **float** property is used to wrap other elements around an element
  - Elements float horizontally, hence they can be floated left or right, not up or down
  - The elements after the floating element will flow around it
  - The elements before the floating element will not be affected
- Syntax:
  float: left | right | none
  - If left is specified, the element floats to the left and the text wraps around to the right and vice versa
- Example:

```
img { float:right; }
```

Clear

- **clear** property can be used by elements to ensure that a previous floating element is not adjacent to one of its sides
  - *clear* turns off floating
- Syntax:
  clear: none | left | right | both

- Example:
  - If an image is set to float:right, then the block-level element following it should have its clear set to right and vice versa

Float & Clear Example

- CSS:

```
div {border:1px dotted #ff3399;}
img {float:right;}
p {clear:right;}
```

- Markup:

```
<div>
<img src="bgtile.jpg">
Nisi aliquam sociis. Aliquam odio turpis, magnis. Habitasse.
    Pulvinar scelerisque ac placerat et, augue ut pulvinar magna
    tortor! Magna integer augue elit aliquet elementum, rhoncus?
<p>Augue placerat augue ac sagittis sit quis sed diam duis
    scelerisque dictumst pellentesque, et augue, dolor lundium
    amet. Platea? Turpis dictumst. Elementum tincidunt!</p>
</div>
```

Exercise: Adjacent *divs* with float

- Float is commonly used to place divs adjacent to each other, left to right
- Create 3 divs with classes sidebar-first, content & sidebar-second and make them place left to right
- Use & modify these classes:

```
div { border:1px solid #ccc; }
.sidebar-first { width:23%; }
.content { width:50%; }
.sidebar-second { width:23%; }
.clearfix { clear:both; }
```

Positioning

- **position** property moves the element out of the normal flow
- Syntax:
  position: static | relative | absolute | fixed
  - **static** : block-level element is laid out according to normal flow
  - **relative** : positioned relative to its normal position, can overlap other elements
  - **absolute** : positions relative to the first ancestor element that has a position other than static
  - fixed : offset is fixed relative to the browser window

- Setting offset: use *left*, *right*, *top* and *bottom* properties to define offset

position:fixed Example

- Example:

```
/* fixes <img class="sticky" src=".."> to top right
even when scrolled */
img { position:fixed; top:3px; right:3px; }
```

position:absolute Example

```
<style type="text/css">
.container { height:400px; border:1px dotted #ccc;
    position:relative; }

.colorbox { width:25px; height:25px; background-
    color:#FF0066; position:absolute; top:0; }
</style>

<div class="container">
<p>Dapibus integer elementum sit sociis tincidunt! Velit cum
    porta tempor pulvinar pellentesque pulvinar in.</p>
<div class="colorbox"><!-- --></div>
</div>
```

Answer this

- If two or more elements are set with position:absolute within a container, what would happen?
- Will they float around each other?
- Will they overlap each other?
- What will be the order – which would get placed first, which next?

z-index

- When elements are positioned outside the normal flow, they can overlap other elements
- **z-index** property specifies the stack order of an element
  - Which element should be placed in front of, or behind, the others
  - An element can have a positive or negative stack order
  - An element with greater stack order is always in front of an element with a lower stack order
  - If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top
- Example:

```css
.colorbox {position:absolute; top:0; z-index:-10;}
.colorbox2 {position:absolute; top:0; z-index:555;}
```

Selectors

Selectors

- In CSS, you can select elements by:
  - Element name
  - ID
  - Class name

- Element Selector
  - Select the elements by their names
  - All elements within the document are selected

```
li { display:inline; } /* selects all li */
a { text-decoration:none; } /* selects all anchors */
```

Id & Class Selectors

- Id Selector
  - Elements can be selected by the id value associated with them

```
#block1 { width:120px; } /* selects element with id="block1" */
```

- Class Selector
  - Elements can be selected by the class name associated with them

```
.error { color:red; } /* selects all elements with class="error" */
```

Universal Selector

- To match any element, it is possible to replace the element name in the selector by the symbol *****

- Example:

```
/* selects all HTML elements in the document */
* { padding: 0; margin: 0; }
```

Composite Selector

- Composite selectors allow selection of elements of a specific type with a specific id or class name or both
  - The element name is followed by a class name or id
  - Notice that there is no space between the element name & class / id

- Example:

```
h1.title { font-size:160%; }              /* selects <h1 class="title"> */

div#intro { font-weight:bold; }   /* selects <div id="intro"> */

p#first.highlight { font-weight:bold; } /* What is selected here?
   */
```

Contextual / Descendant Selector

- Contextual selectors consist of two or more simple selectors following each other
  - The selectors are separated by a space
  - The first selector becomes the ancestor and the selectors following it becomes their descendant, and so on
  - The selectors can be of any type – element, class, id or composite
  - There is no limitation on the ancestory

- Example:

```
ul li { display:inline; } /* selects all li under all ul */

div#intro p { color:#c40000; } /* selects all <p> under <div
   id="intro"> */
```

Contextual Selector…

- Example:

```
/* What is selected here? */
div#header ul.menu a { text-decoration:none; }

/* What is selected now? */
.left p.more { text-align:right; }

/* And now? */
ul * li { font-weight: bold; }
```

Child Selector

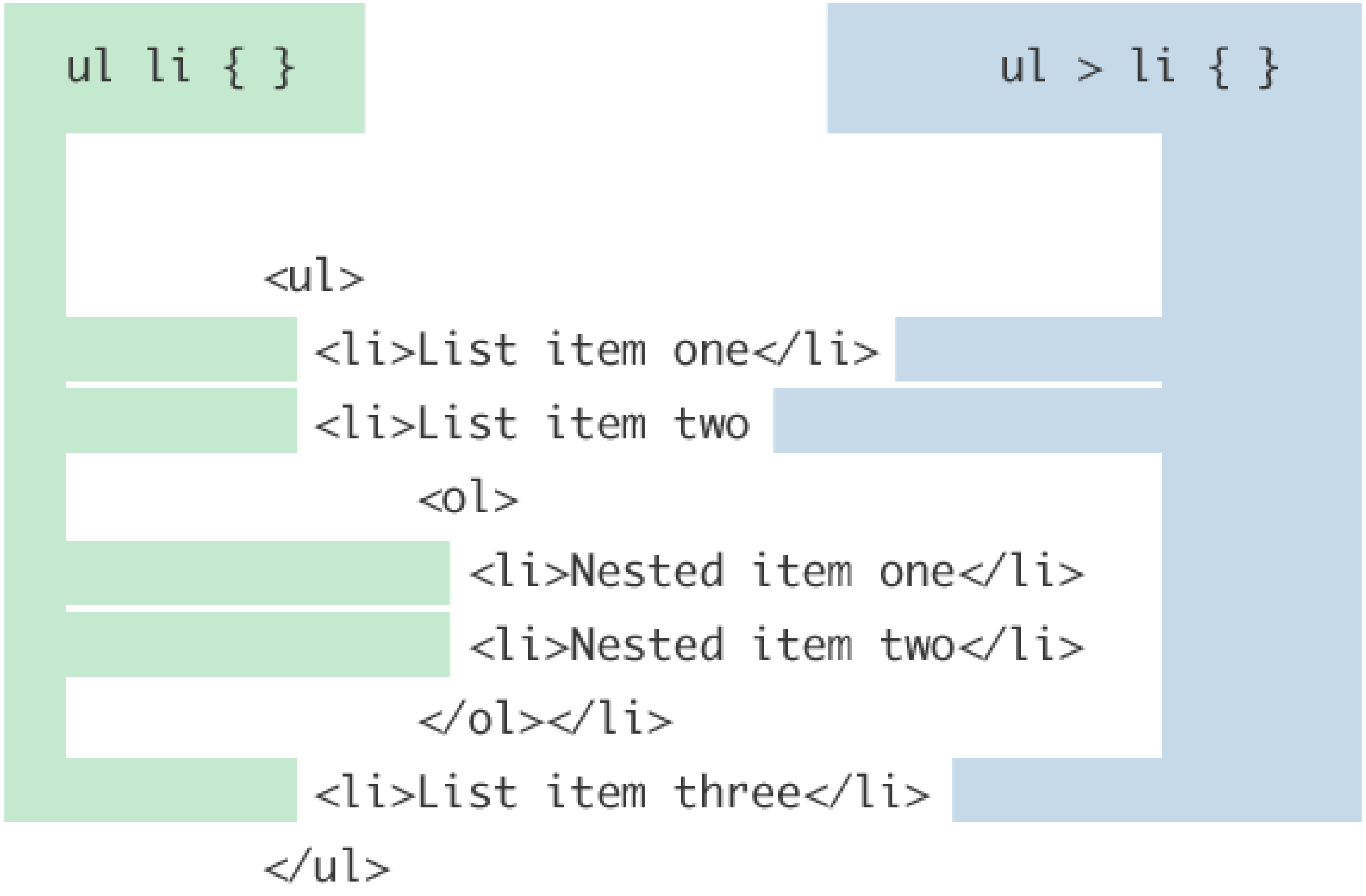
- Child selectors consist of two or more selectors where the second selector is the immediate child of the first
  - Contrary to the contextual selector where an ancestor – descendant relationship is created

- Use > between selectors to define parent:child relationship

- Example:

```
h1 > em { color: green }
p > em { color: red }
div > p > em { background: blue }
```

Child Selector



```
<ul>
  <li>List item one</li>
  <li>List item two
      <ol>
        <li>Nested item one</li>
        <li>Nested item two</li>
      </ol></li>
  <li>List item three</li>
</ul>
```

Ref: http://css-tricks.com/child-and-sibling-selectors/

Parent Selector?

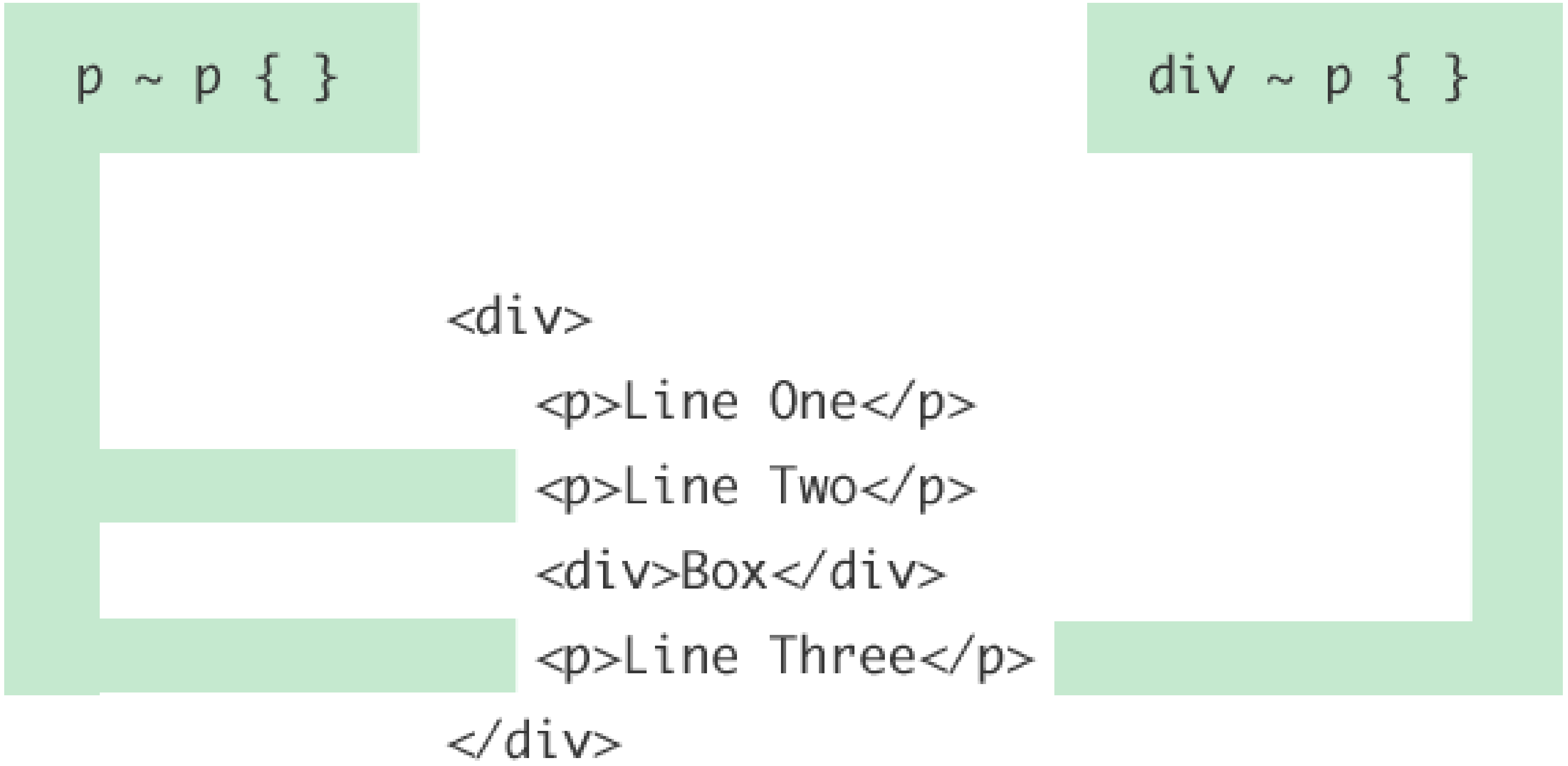
- What would be the syntax for a parent selector?

Sibling Selector

- Sibling selectors consist of two selectors where both selectors have the same parent and the second selector occurs after the first
  - The first selector is not selected
- Use ~ between selectors to define sibling relationship
- Example:

```
p.two ~ p { border:1px dashed #666; }
```

Sibling Selector



```
p ~ p { }
```

```
div ~ p { }
```

```
<div>
   <p>Line One</p>
   <p>Line Two</p>
   <div>Box</div>
   <p>Line Three</p>
</div>
```

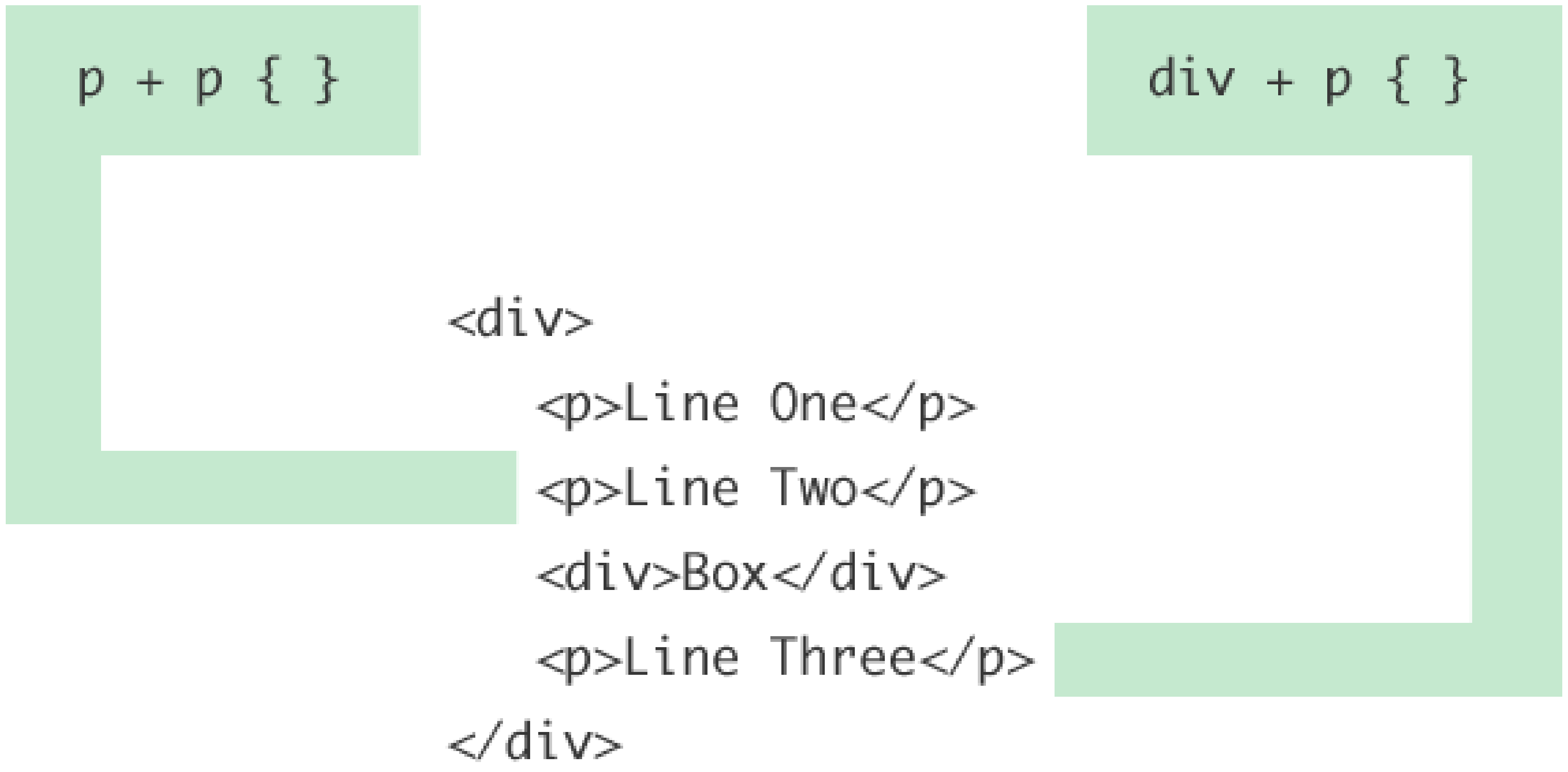
Ref: http://css-tricks.com/child-and-sibling-selectors/

Adjacent Sibling Selector

- Adjacent sibling selectors allows you to select an element that is directly after another specific element
- Use + between selectors to define adjacent sibling relationship
- Example:

```
p + p { border:1px dashed #666; }
```

Adjacent Sibling Selector



```
<div>
   <p>Line One</p>
   <p>Line Two</p>
   <div>Box</div>
   <p>Line Three</p>
</div>
```

Ref: http://css-tricks.com/child-and-sibling-selectors/

Grouping The Selectors

- If the same rule applies to a set of elements, they can be grouped together
  - The selectors are separated by commas

- Example:

```
/* all <h1>, <p> & elements with class="feedBlock" */
h1, p, .feedBlock { color:#800080; line-height:18px; }
```

Attribute Selectors

- Attribute selectors allow selection based on an element’s attribute name or specific attribute value

- Syntax:
  - **[*attribute*]** : Selects all elements with the attribute name
  - **[*attribute=value*]** : Selects all elements with the specific attribute name and specific value

- Examples:

```
/* any element with attribute "title" */
[title] {font-weight:bold;}
```

Attribute Selectors…

- Examples:

```
/* any element with attribute alt and value logo */
[alt="logo"] { border-color:red; }

/* element <a target="_blank"> */
a[target="_blank"] { color:blue; }

/* element with attribute type with value text AND attribute name
   with value dob */
[type="text"][name="dob"] { color:green; }
```

CSS3 Attribute Selectors

- **[*attribute^=value*]**
  - a[src^="https"] → Selects every a element whose src attribute value begins with "https"

- **[*attribute$=value*]**
  - a[src$=".pdf"] → Selects every a element whose src attribute value ends with ".pdf"

- **[*attribute*=value*]**
  - a[src*="microsoft"] → Selects every a element whose src attribute value contains the substring "microsoft"

Pseudo-classes

- Pseudo-classes are classes that are automatically added to certain elements by the browser

- Pseudo-classes have the form:
  selector**:pseudo-class** { property: value }

Anchor Pseudo-class

- Pseudo-classes are assigned to an anchor element to allow links to be displayed differently depending on whether they are visited or active links

- Anchors are assigned one of these pseudo-classes at any time:
  - a:link – a link that has not been visited before by the browser
  - a:active – a link that is being followed at the moment by the browser
  - a:visited – a link that has already been visited by the browser before
  - a:hover – a link that is being hovered over by the user

Anchor Pseudo-class…

- By default, most browsers set:

```
a:link { color:blue; text-decoration:underline; }

a:active { color:red; text-decoration:underline; }

a:visited { color:purple; text-decoration:underline; }
```

  - However, you can override it and set your own styles

Dynamic Pseudo-classes

- Browsers sometimes change the rendering in response to user actions
  - For example, when the user selects the textfield

- The dynamic pseudo-classes:
  - :hover = applied when the user selects an element, but does not activate it
  - :active = applied while an element is being activated by the user (between the times the user presses the mouse button and releases it)
  - :focus = applied when an element has the focus

Dynamic Pseudo-class Example

- Example:

```
a:active { background-color:red; }
a:hover, textarea:hover { background-color:green; }
textarea:focus { background-color:pink; }
…
<textarea name="textarea" rows="8"></textarea>
<a href="example">Get help here.</a>
```

The Language Pseudo-class

- The **:lang()** pseudo class selector matches elements based on the context of their given language attribute in HTML
- Example:

```
:lang(en) q { quotes: '“' '”'; }
:lang(fr) q { quotes: '«' '»'; }
:lang(de) q { quotes: '»' '«'; }
…
<!DOCTYPE html>
<html lang="fr">
<body>
<p><q>Et non eros quis pulvinar phasellus, tortor eu
    ac!</q></p>
</body>
</html>
```

More Pseudo-classes

- **::first-letter**
  - styles the first letter in an element
  - `p::first-letter { font-size:3em; color:#c80000; }`

- **::first-line**
  - styles the first line in an element
  - Not the first sentence, only the first rendered line (test is by resizing the browser)
  - ::first-letter overrides ::first-line
  - `p::first-line { font-size:2em; color:green; }`

- **:first-child**
  - target the first element immediately inside another element
  - `li:first-child { font-size:1.3em; color:#c80000; }`

CSS3 Pseudo-class Selectors

- **:last-child**
  - p:last-child → Selects every p element that is the last child of its parent
- **:only-child**
  - p:only-child → Selects every p element that is the only child of its parent
- **:nth-child(*n*)**
  - p:nth-child(2) → Selects every p element that is the second child of its parent
- **:nth-last-child(*n*)**
  - p:nth-last-child(2) → Selects every p element that is the second child of its parent, counting from the last child

CSS3 Pseudo-class Selectors…

- **:empty**
  - p:empty → Selects every p element that has no children (including text nodes)
- **:enabled**
  - input:enabled → Selects every enabled input element
- **:disabled**
  - input:disabled → Selects every disabled input element
- **:checked**
  - input:checked → Selects every checked input element
- **:not(*selector*)**
  - :not(p) → Selects every element that is not a p element
- **::selection**
  - ::selection → Selects the portion of an element that is selected by a user

Generated Content with CSS

- **::before** & **::after** are pseudo elements that allows to insert content onto a page from CSS
  - ::before inserts content before any other content in the selected element
  - ::after inserts content after all the content in the selected element

- Example:

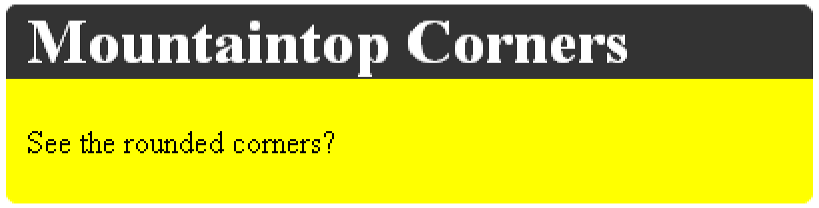
```
ul::before { content: "list start"; color:#FF6600; }
ul::after { content: "list end"; color:#003399; }
```

  - The value for content can be a string or an image url
  - The added content does not get added to the DOM
  - HTML in the content string is not parsed as html

CSS Techniques

Mountaintop Corner Boxes

- Mountaintop corner boxes is a technique to create rounded corners



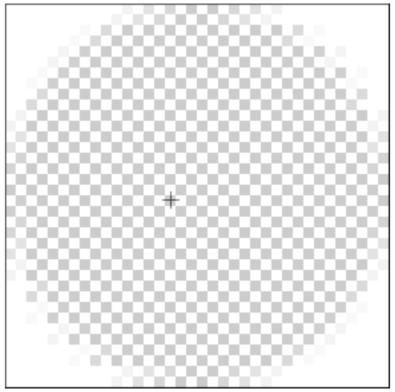
Mountaintop Corners

See the rounded corners?

- The idea is to cover the corners of the box with a convex / concave image of the same color as the background leaving the other portion transparent

Mountaintop Corner Boxes…

- Let us consider making the corners of a *div* rounded with images
- The 4 corners images are put together as a single image



- Inside the div, we will create 4 empty *span*s with classes assigned to it
- Using the classes set on *div* & *span*s, we use *position*, *background* and *background-position* properties to make the *div* get rounded corners

Mountaintop Corner Boxes…

- The HTML:

```
<div id="box1" class="box">
    <span class="top-left bg"><!-- --></span>
    <span class="bottom-right bg"><!-- --></span>
    <h1>Mountaintop Corners</h1>
    <div class="content">See the rounded corners?</div>
</div>
```

  – The spans are kept empty as they will be used only to style the corners

Mountaintop Corner Boxes…

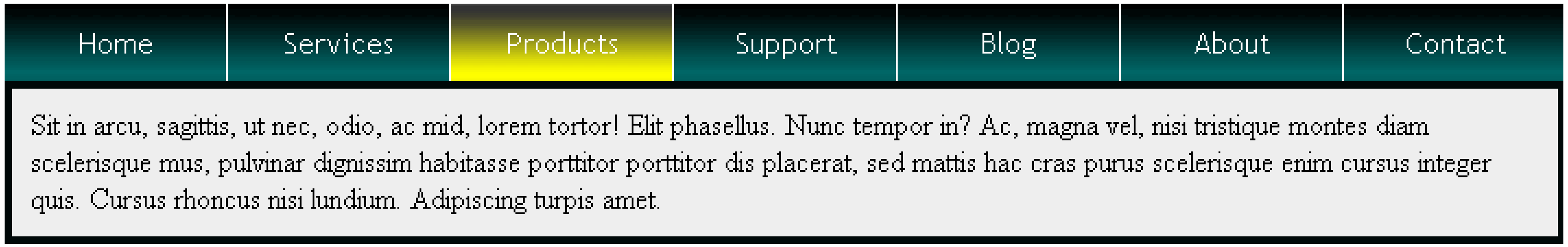
- The CSS:

```
.box { position:relative; margin:10px; }
.box .bg { position:absolute; height:9px; width:9px;
   background:transparent url(hole.png) no-repeat; }

.box .top-left { top:-1px; left:-1px; background-position:top
   left; }
.box .bottom-right { bottom:-1px; right:-1px; background-
   position:bottom right; }

#box1 { background:#ff0; }
#box1 h1 { background:#333; color:#fff; text-indent:10px;
   }
```

Tabbed Navigation

Tabbed Navigation

- Tabbed navigation is a popular technique to create attractive menu that uses images and has hover effects too



Sit in arcu, sagittis, ut nec, odio, ac mid, lorem tortor! Elit phasellus. Nunc tempor in? Ac, magna vel, nisi tristique montes diam scelerisque mus, pulvinar dignissim habitasse porttitor porttitor dis placerat, sed mattis hac cras purus scelerisque enim cursus integer quis. Cursus rhoncus nisi lundium. Adipiscing turpis amet.

  – Navigation (menu) items are a form of list
  – The list items are first made to flow horizontally
  – The list items are then styled with background images to show various effects

Tabbed Navigation…

- The HTML:

```
<div id="wrapper">
 <ul id="mainMenu">
   <li class="first"><a href="home.html">Home</a></li>
   <li><a href="products.html">Products</a></li>
   <li><a href="support.html">Support</a></li>
   <li><a href="about.html">About</a></li>
   <li><a href="contact.html">Contact</a></li>
 </ul>
 <div id="content"><p>Sit in arcu, sagittis, ut nec, odio, ac
   mid, lorem tortor! Elit phasellus. Nunc tempor in? Ac,
   magna vel, nisi tristique montes.</p></div>
</div>
```

Tabbed Navigation…

- The CSS:

```
#mainMenu {
   width:846px;
    padding:0;
}
#mainMenu li {
   display:block;
   float:left;
   width:120px;
   margin-left:1px;
}
#mainMenu li:first-child, #mainMenu li.first {
   margin-left:0px;
}
```

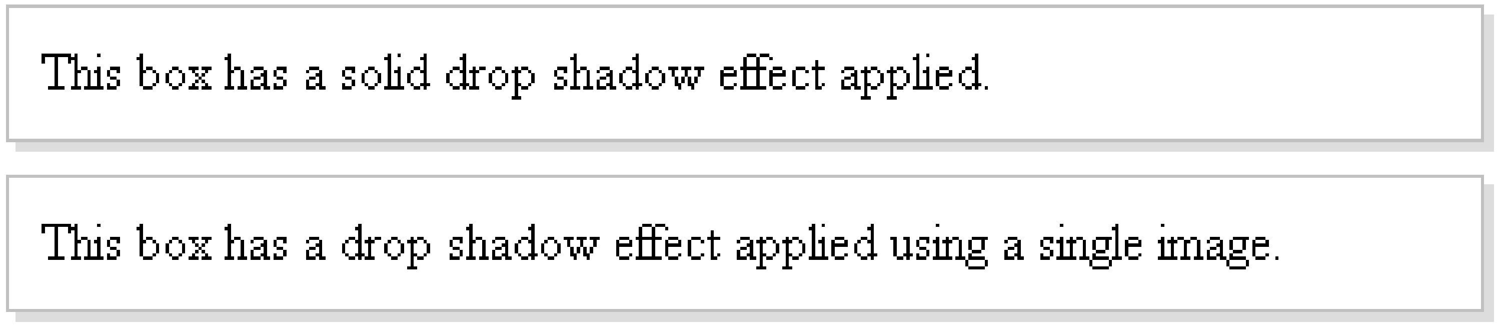
Tabbed Navigation...

```
#mainMenu a {
   display:block;
   padding:10px;
   background:transparent url(link.gif) repeat-x;
   color:#fff;
   text-align:center;
}
#mainMenu a:hover {
   background:transparent url(hover.gif) repeat-x;
   color:#333;
}
#mainMenu a:active {
   background:transparent url(active.gif) repeat-x;
   color:#fff;
}
```

Drop Shadows

Drop Shadows

- A drop shadow effect can be created by nesting one div inside of another div
- Using relative positioning, the nested div, which will contain the content, can be pushed from the bottom right, thus allowing the outer div to show through
- The outer div is then given a background color or image to create the drop shadow effect


This box has a solid drop shadow effect applied.

This box has a drop shadow effect applied using a single image.

Drop Shadows...

- The HTML:

```
<div class="box shadow1">
    <div class="content">
        <p>This box has a solid drop shadow effect applied.</p>
    </div>
</div>
<div class="box shadow2">
    <div class="content">
        <p>This box has a drop shadow effect applied using a single image.</p>
    </div>
</div>
```

Drop Shadows…

- The CSS:

```
.box { width: 95%; margin: 10px; }
.box .content {
   position: relative;
   background-color: #fff;
   border: 1px solid #c0c0c0;
   bottom: 3px;
   right: 3px;
}
.shadow1 { background: #ddd; }
.shadow2 { background:url(shadow-bg.gif) bottom right; }
```

CSS Image Replacement

CSS Image Replacement

- CSS image replacement is a technique of replacing a text element with an image
- There are several techniques to achieve image replacement:

- Technique #1:
  - HTML:

```
<h1 class="technique-one">Hello CSS</h1>
```

  - CSS:

```
h1.technique-one {
    width:2327px; height:158px;
    background:url(image.png) top right no-repeat;
    margin:0 0 0 -2000px;
}
```

CSS Image Replacement…

- Technique #2:
  - HTML:

```
<h1 class="technique-two">Hello CSS</h1>
```

  - CSS:

```
h1.technique-two {
    width:327px; height:158px;
    background:url(image.png) top left no-repeat;
    text-indent:-9999px;
}
```

CSS Image Replacement…

- Technique #3:
  - HTML:

```html
<h1 class="technique-three">Hello CSS</h1>
```

  - CSS:

```css
h1.technique-three {
    width:327px;
    padding:158px 0 0 0;
    height:0;
    background:url(image.png) top left no-repeat;
    overflow:hidden;
}
```

CSS Sprites

CSS Sprites

- A CSS sprite is a single image file that contains several graphics

- By showing different parts of the sprite in different locations, it appears that there are several different images, but they are all contained in a single file, which translates to a single download

CSS Sprite Example

- For tabbed navigation, we used 3 different background images
- Instead of having three different images, we can combine these into a single image

CSS Sprite Example…

- Now we can use the background-position property to show only portions of this single image

```
#mainMenu a {
   …
   background:transparent url(sprite.gif) 0px 0px repeat-x;
}
#mainMenu a:hover {
   background-position: 0 -148px;
   color:#333;
}
#mainMenu a:active {
   background-position: 0 -74px;
   color:#fff;
}
```

CSS3 Borders

CSS3 Borders

- With CSS3, you can create rounded borders, add shadow to boxes, and use an image as a border - without using a design program, like Photoshop

- The new CSS3 border properties introduced are:
  - border-radius
  - box-shadow
  - border-image

CSS3 Box Shadow

- In CSS3, the box-shadow property is used to add shadow to boxes:

```
div {
box-shadow: 10px 10px 5px #888888;
}
```

- Syntax – **box-shadow: *h-shadow v-shadow blur/spread color* inset;**
  - *h-shadow* – Required. The position of the horizontal shadow. Negative values are allowed
  - *v-shadow* – Required. The position of the vertical shadow. Negative values are allowed
  - *blur/spread* – Optional. The blur/spread distance
  - *color* – Optional. The color of the shadow
  - inset – Optional. Changes the shadow from an outer shadow (outset) to an inner shadow

CSS3 Rounded Corners

- Adding rounded corners in CSS2 was tricky
  - We had to use different images for each corner
- In CSS3, the border-radius property is used to create rounded corners:

```
div {
    border:2px solid;
    border-radius:25px 25px 25px 25px;
}
```

  - the border-radius can take upto 4 values, each in order for top-left, top-right, bottom-right, bottom-left
  - if only a single value is given, then the same is applies to all corners
  - the units can be in pixels or %

CSS3 border-radius

- Each corner can be specifically targeted:

```
div {
    border-top-left-radius: 1px;
    border-top-right-radius: 2px;
    border-bottom-right-radius: 3px;
    border-bottom-left-radius: 4px;
}
```

- The rounding does not have to be perfectly circular, it can be elliptical
- This is done using a slash ("/") between two values:

```
div {
  border-radius: 30px/10px; /* horizontal radius / vertical radius */
}
```

Border Radius & Drop Shadow

- The HTML:

```
<div class="box">test</div>
```

- The CSS:

```
.box {
    position: relative;
    width: 400px; height: 300px;
    background-color: #fff;
    box-shadow: 0 1px 5px rgba(0,0,0,0.25), 0 0 50px
    rgba(0,0,0,0.1) inset;
    border-radius: 1%   1%   1%   1% / 1%   1%   1%   1%;
}
```

Border Radius & Drop Shadow…

```
.box:before {
    position: absolute;
    width: 80%; height: 40%;
    left: 10%; top: 0%;
    border-radius: 50%;
    z-index: -1;
    content: "";
    box-shadow: 0 -7px 16px rgba(0,0,0,0.4);
}
.box:after {
    position: absolute;
    width: 80%; height: 40%;
    left: 10%; bottom: 0%;
    border-radius: 50%;
    z-index: -1;
    content: "";
    box-shadow: 0 7px 16px rgba(0,0,0,0.4);
}
```

CSS3 Gradients

CSS3 Gradients

- CSS3 Gradients create color gradients without the use of images
- Gradients are of two types – linear and radial
- Gradients are created with the background-image property

- Linear gradient syntax:
- **background-image**: linear-gradient(startPosition, color1 stop, color2 stop, … colorN stop);
  - startPosition is the point where the gradient starts. It can have the values top, right, bottom, left or a value in deg (45deg)
  - After the start position, list of colors and stops are given
  - The colors can have transparency using rgba()
  - The stops are provided as %. If ommitted, browser will calculate accordingly

CSS3 Gradients

- Radial gradient syntax:
- **background-image**: radial-gradient(position, shape & size, color1, color2);
  - Position defines the gradient position and can have values top, right, bottom, left or center. It can also be specified in % as x & y values
  - The second argument takes two values – shape and gradient size
  - The shape can have values ellipse or circle
  - Size can have the values:
    - closest-side = The gradient’s shape meets the side of the box closest to its center. A synonym *contain* can also be used
    - closest-corner = The gradient’s shape is sized so it exactly meets the closest corner of the box from its center
    - farthest-side
    - farthest-corner = a synonym *cover* can also be used

CSS3 Linear Gradient Example

- Example1:

```
background-image: -webkit-linear-gradient(left,
   #FF5A00 20%, #FFAE00 20%, #F00);
background-image: linear-gradient(left,
   #FF5A00 20%, #FFAE00 20%, #F00);
```

- Example 2:

```
background-image: -moz-linear-gradient(45deg,
   #FF5A00, #FFAE00, #FF5A00);
background-image: linear-gradient(45deg,
   #FF5A00, #FFAE00, #FF5A00);
```

CSS3 Radial Gradient Example

- Example1:

  background-image: -webkit-radial-gradient(center, ellipse cover, #FFEDA3, #FFC800);

  background-image: radial-gradient(center, ellipse cover, #FFEDA3, #FFC800);

- Example2:

  background-image: -moz-radial-gradient(25% 25%, circle cover, #FF5A00, #FFC800);

  background-image: radial-gradient(25% 25%, circle cover, #FF5A00, #FFC800);

CSS Clipping & Masking

Clipping & Masking

- Masking in image editing is a method of ‘hiding’ a portion of an object based on another object

- We can use the CSS3 background-clip property to achieve the same effect

background-clip Property

- **background-clip** property specifies the painting area of the background

- Syntax: **background-clip**: border-box | padding-box | content-box | inherit;
  - border-box: The background is clipped to the border box
  - padding-box: The background is clipped to the padding box
  - content-box: The background is clipped to the content box

- Webkit / Chrome adds *text* as another value for background-property, making it possible to use text for the mask

Clipping / Masking Example

- The HTML:
```
<h1>CSS3</h1>
```

- The CSS:
```
h1 {
    font: 15em Georgia;
    background-image:url(seasurf.jpg);
    -webkit-background-clip: text;
    color: rgba(0,0,0,0); /* make the text color transparent */
}
```

  - Add the background-image (can be a gradient too)
  - Set the background-clip to text
  - Make the text color transparent

Appendix

CSS Reset

- The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on
- Reset styles quite often appear in CSS frameworks, and the original "meyerweb reset" is one of the foremost used CSS reset (http://meyerweb.com/eric/tools/css/reset/)
- To use CSS Reset, put the CSS code in a file and then @import it into your stylesheet, just above everything else
- Meyerweb CSS Reset:

Meyerweb CSS Reset

```css
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
	margin: 0;
	padding: 0;
	border: 0;
	font-size: 100%;
	font: inherit;
	vertical-align: baseline;
}
```

Meyerweb CSS Reset

```css
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
	display: block;
}
body {
	line-height: 1;
}
ol, ul {
	list-style: none;
}
blockquote, q {
	quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
	content: '';
	content: none;
}
table {
	border-collapse: collapse;
	border-spacing: 0;
}
```