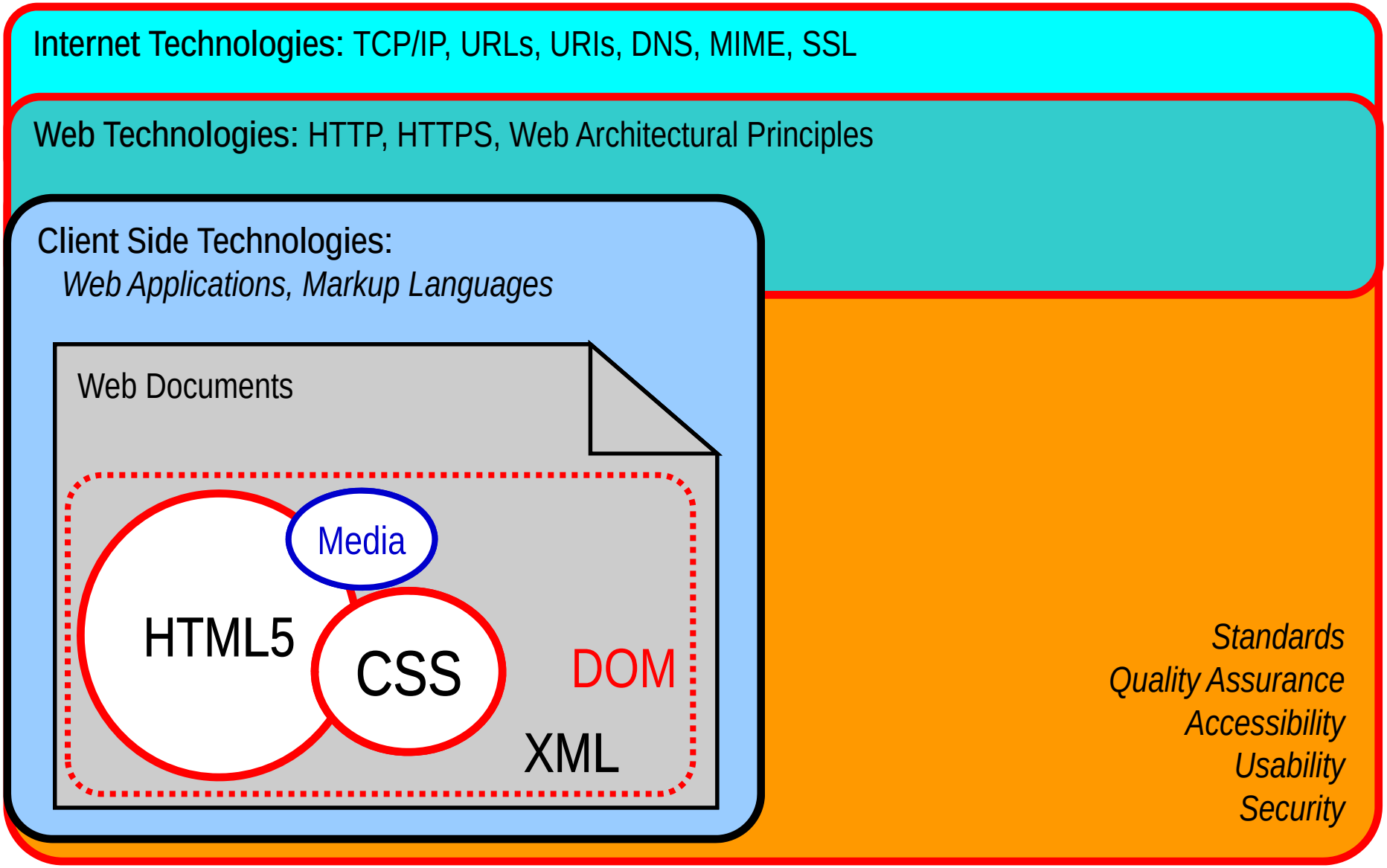
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011
Creating Web Applications

Lecture 4 – Presentation and CSS

Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)

Unit of Study Outline

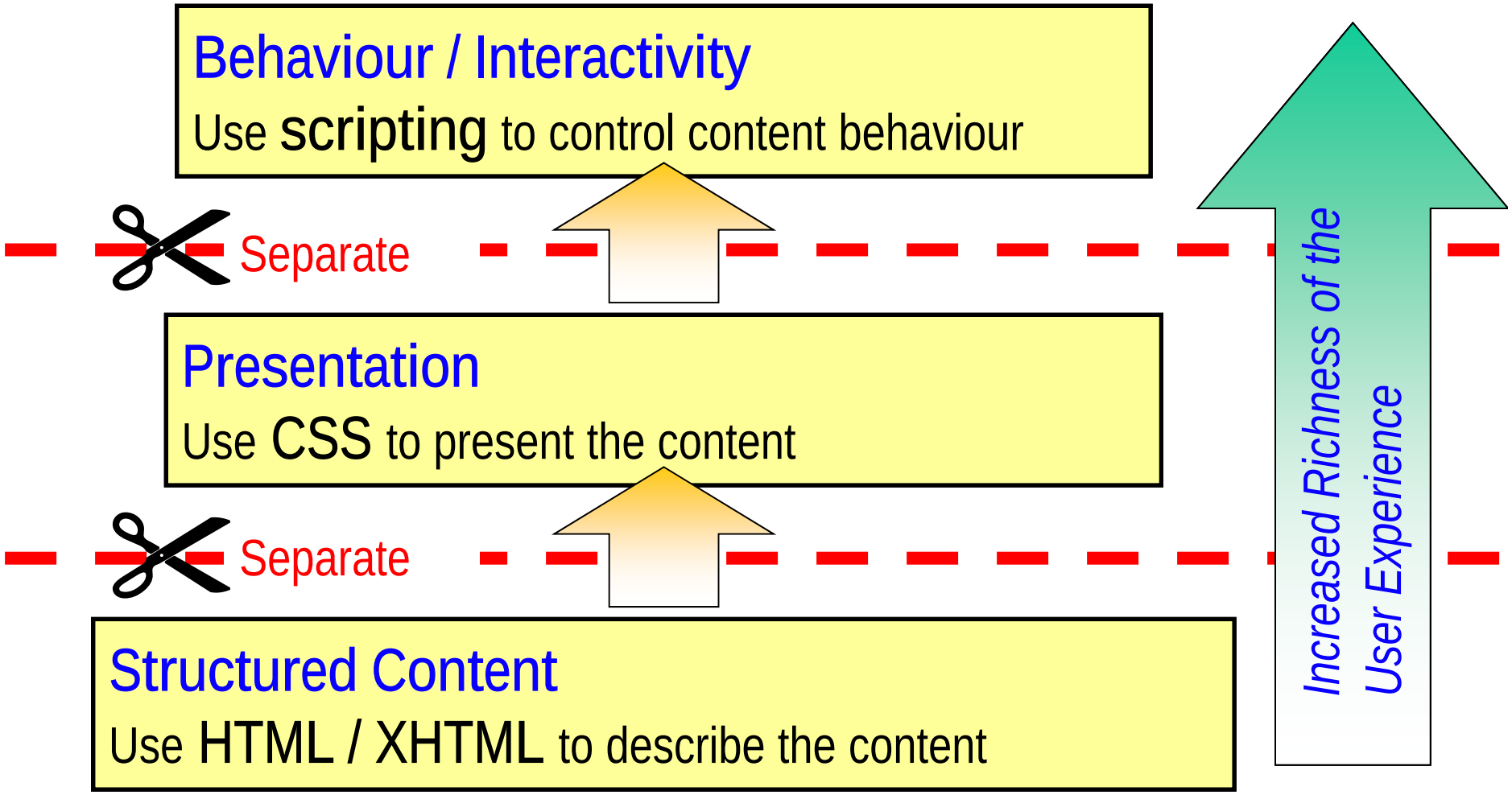


Contents

CSS

- What is CSS?
- Linking CSS to HTML
- CSS Selectors
- CSS Properties
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout
- Alternate Style

Review: Separate content from presentation



Work from the bottom up !

Last week: HTML5 structure

<header>…</header>

<nav>
…

</nav>

<section>…</section>

<section>…</section>

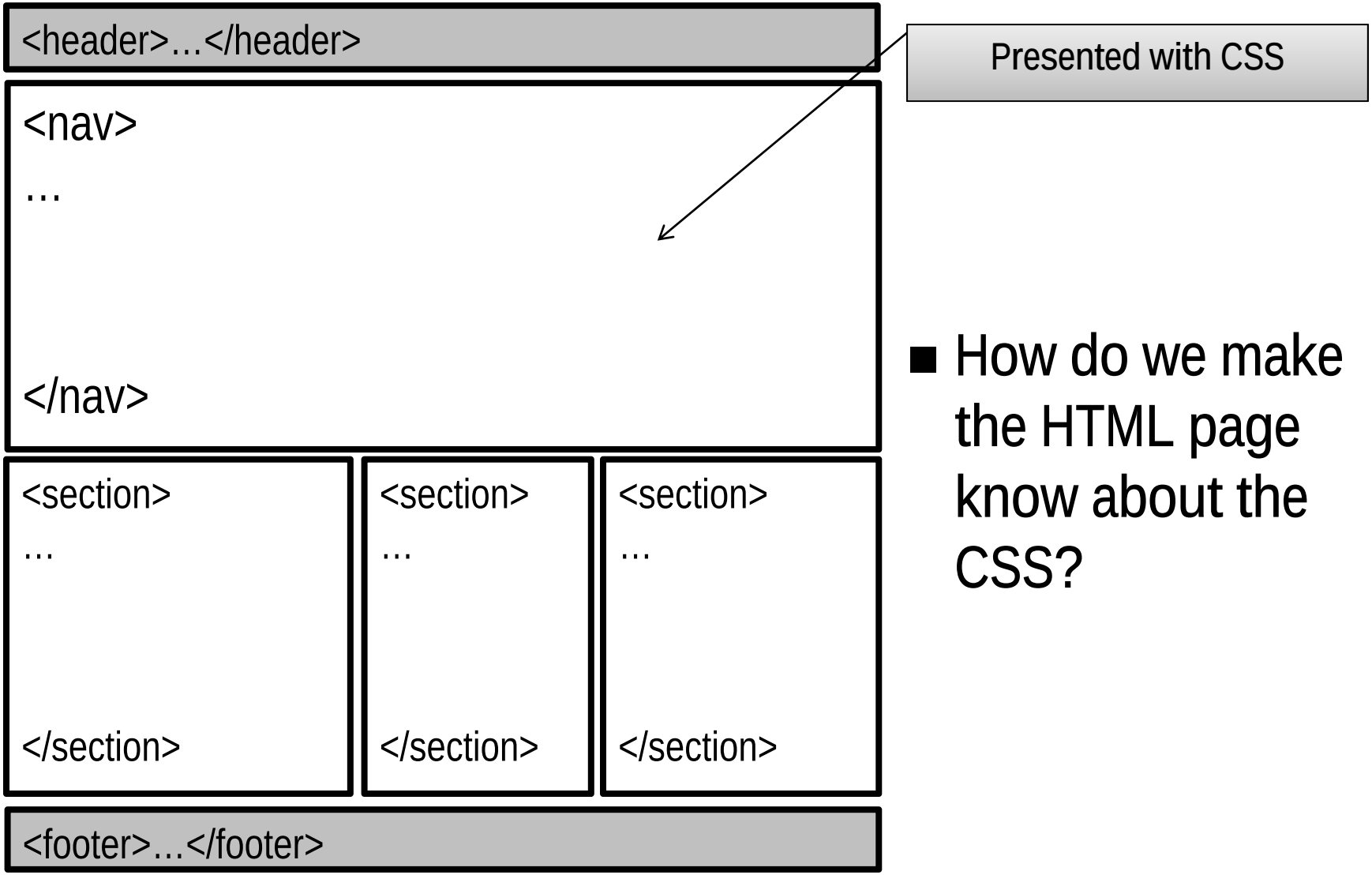
<section>…</section>

<footer>…</footer>

Presented without CSS

- Always remember that HTML is only about **content** and **structure.**
  CSS is used to specify how the HTML will be styled/rendered on the screen,
  (or styled for a printer, voice synthesiser, etc.)

Applying CSS



- How do we make the HTML page know about the CSS?

CSS: Cascading Style Sheets

http://www.csszengarden.com

Shows the power of different styles,
applied to one common HTML page of content

Tutorials

http://www.w3schools.com/css/

- A lot of info here, including 150 interactive examples

http://css.maxdesign.com.au/#downloadable

- Great website and guides

http://css.maxdesign.com.au/index.htm

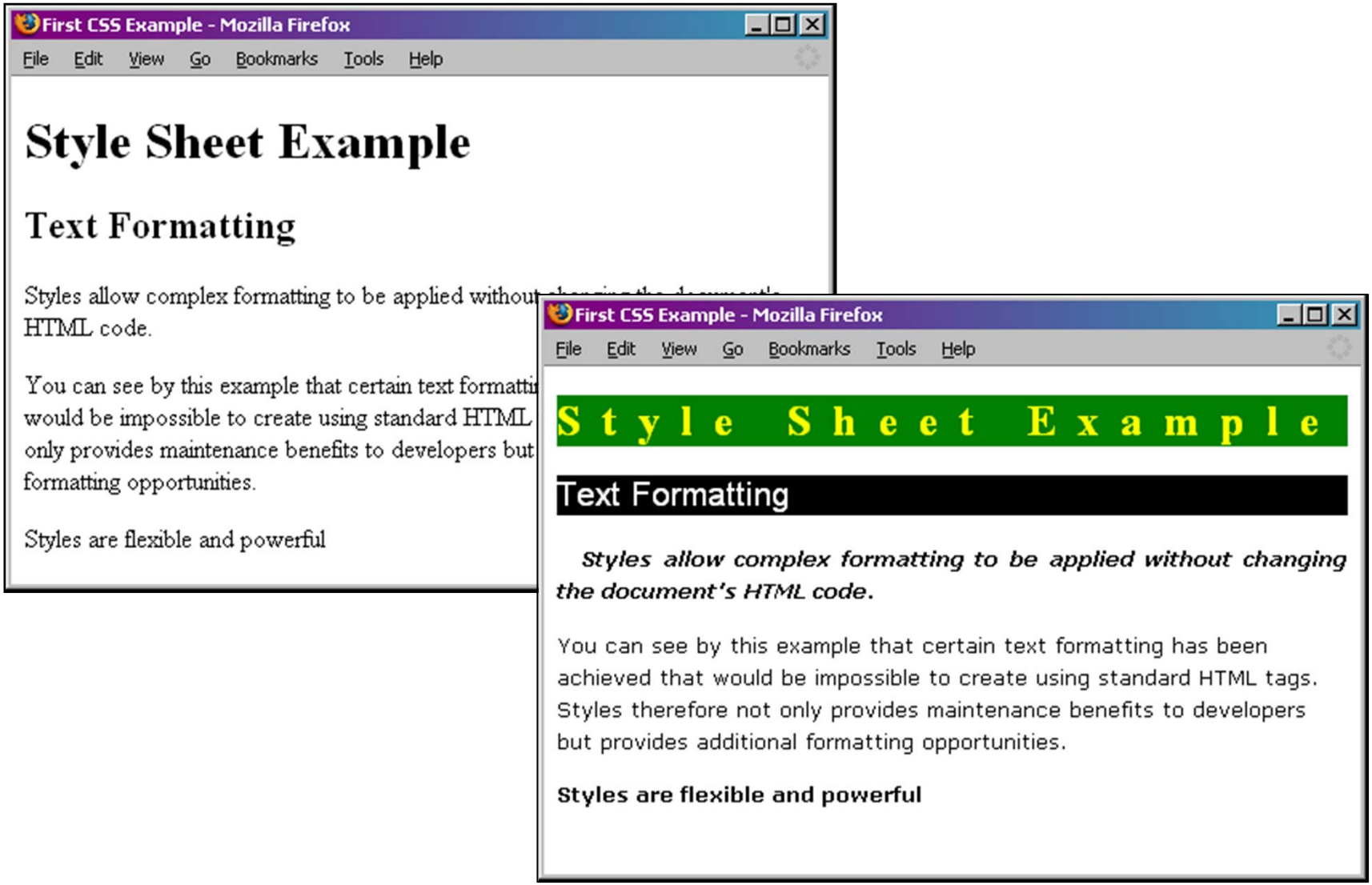
http://www.maxdesign.com.au/articles/css-layouts/

- Older but useful for lists and layout models

Why CSS?

- *Separate **content** from **presentation***
- ***Easier to maintain** large projects*
- *Provides **more control** than just HTML*
- *Supports **different user needs***
- *Supports **different presentation alternatives***
- *Supports **device independence***
- *Supports **device specific styles***

First CSS Example

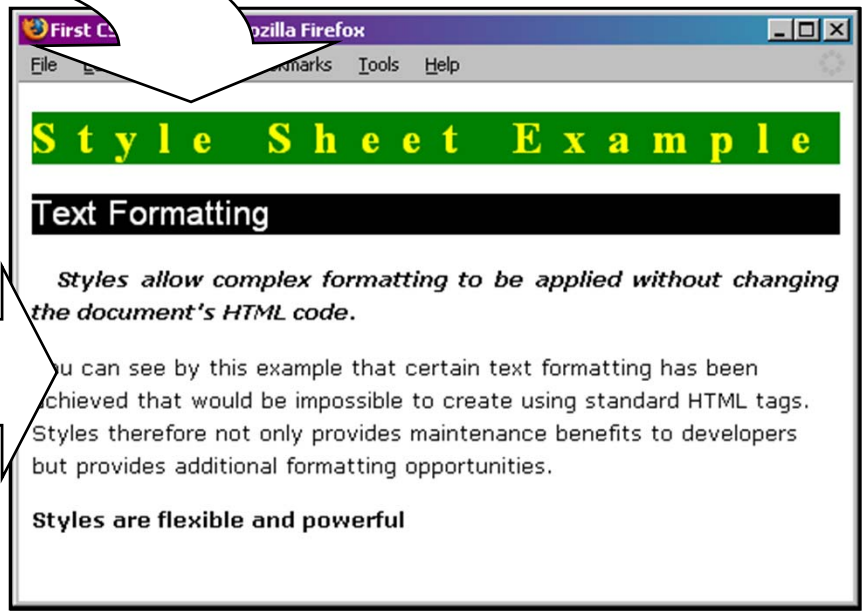


First CSS Example

Remember the simple structure of HTML documents!

```
<!DOCTYPE ...>
<html>
  <head>
    <title>...</title>
    <link rel="stylesheet"
    type="text/css"
    href= "mystyle.css" />
  </head>
  <body>
    <!-- body content here -->
  </body>
</html>
```

One external stylesheet can be linked and can be re-used for many webpages



First CSS Example

HTML Structure

```
...
<h1>Style Sheet Example</h1>
<h2>Text Formatting</h2>
<p class="intro">Styles allow
  complex formatting to be applied
  without changing the document's
  HTML code.</p>
<p class="intro">You can see by
  this example that certain text
  formatting has been achieved that
  would be impossible to create
  using standard HTML tags.  Styles
  therefore not only provides
  maintenance benefits to
  developers but provides
  additional formatting
  opportunities.</p>
<p id="keypt">Styles are flexible
  and powerful</p>
...
```

Here both style rules also "inherit" the **p** *element style*

CSS Presentation

```
h1 { letter-spacing: .5em;
     background-color: green;
     color: yellow;
     font-size: 20pt;
     font-family: serif; }

h2 { color: white;
     background: black;
     font: normal 16pt Arial,
     sans-serif; }

p { line-height: 15pt;
     font-size: 10pt;
     font-family: "Verdana",
     sans-serif; }

.intro { text-align: justify;
     text-indent: 12pt;
     font-style: italic;
     font-weight: bold; }

#keypt { font-weight: bold; }
```

element styles

class style

id style

Style Sheet Basics

- **Style sheets** contain style information as a collection of "**rules**"
- **Rules** start with a **`selector`** and then contain style **`properties`** and **`values`**.

You need to know this terminology, so you can talk with other web developers.

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

CSS Rule

A **selector** identifies the **markup elements** that the style property values will be applied to. *eg. element, class, id*

More about selector "types" later ...

CSS: Quick Start Style Rule Examples

```
h1, h2     { font-family: sans-serif; }
th         { color: #3366CC;}
div, p     { border: 1px solid #FF0000; }
a:hover    { font-weight: bold; }
li         { font-size: 12px; }
a          { text-decoration: underline overline; }
h3         { border-bottom: 2px dashed green; }
p          { text-align: justify; }
p.indent   { text-indent: 20px; }
.upper     { text-transform: uppercase; }
img        { float: right; }
ol         { list-style-type: upper-roman; }
```

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

CSS Rule

Contents

CSS

- What is CSS?
- Linking CSS to HTML
- CSS Selectors
- CSS Properties
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout
- Alternate Style

Style Sheet Basics

- Style sheet information can be stored in either:
  - A separate *external* CSS file, linked with a **`link`** element (in the **`head`** element)

  and / or

  - an *embedded* style sheet within a **`style`** element (in the **`head`** element)

  and / or

  - using *inline* style with a **`style`** attribute within *any* element (as a core attribute)

CSS: Methods of Incorporating CSS

- External

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

- Imported

```
@import “filename.css";
  or
@import url(“filename.css");
```

Link to file

Link to url

Downloads CSS first → slow

CSS: Methods of Incorporating CSS

- Inline

```
<HTML tag style = "…" >…</HTML tag>
<h1        style = "color : blue;" >
    …</h1>
```

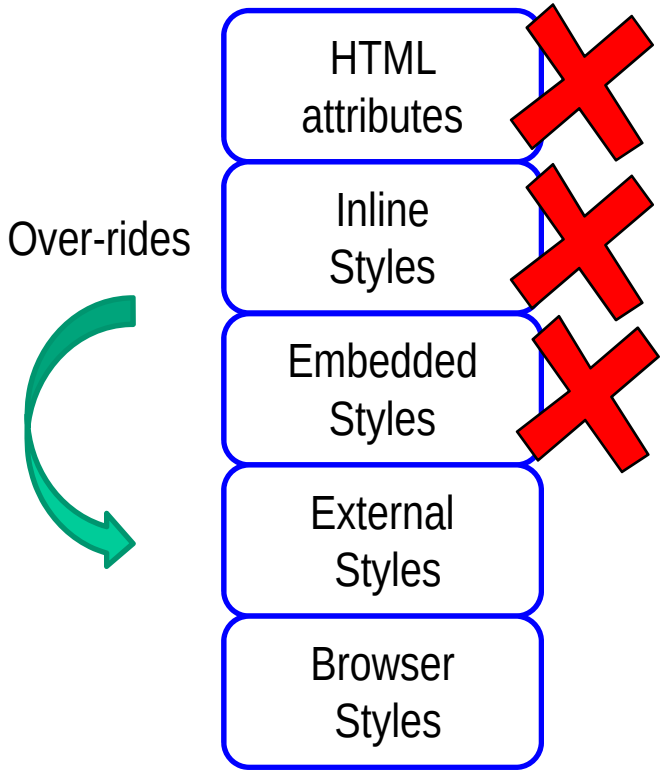
- Embedded

```
<style type= "text/css">
  h1 {color : blue;}
  …
</style>
```

Defined within the <style> element, inside the <head> section of an HTML page:

CSS: Methods of Incorporating CSS

- **Inline** – coded as an attribute
- **Embedded** – defined in the head section (last defined takes precedence)
- **External** – coded in a separate file
- **Imported** – similar to external, but allows a style to import another style



Rules of precedence in applying styles

CSS1, CSS 2.1, and CSS3 http://www.w3.org/TR/CSS/

- CSS1 introduced CSS (now superseded by CSS 2.1)
- CSS2.1 Now largely fully supported by most modern browsers. CSS2.1 was a revision of CCS2.

In this introductory unit we will mainly use CSS2.1

- CSS3 builds on CSS2 *module by module*,
  using the CSS2.1 spec. as its core.
  eg.CSS3 Selectors, CSS3 Colors, CSS3 Media Queries, etc.

Each module is in a different stage of development
(eg. CSS3 Selectors fully developed and supported by most browsers)
CSS3 is being quickly adopted and becoming 'mainstream'.

- CSS4 modules are being developed as new needs arise.

For current CSS status see: http://www.w3.org/Style/CSS/current-work.html

Validating CSS

- W3C CSS validator

http://jigsaw.w3.org/css-validator/

- Integrated validator beta now available

http://www.w3.org/2013/ValidatorSuite/beta/

- Can use Web Developer Extension add-on to validate served or local files.
  (Use 'Tools' / 'Edit tools to set to CSS3)

Writing CSS Comments

- Comments are enclosed in /* … */
- For example

```
/*
  defines the style for all
  articles
*/
article {
  color : blue; /* font color*/
}
```

In your assignments you must have

- Header comments on your CSS
- Line comments on any rules whose application is not obvious
- Comments acknowledging sources of any 3rd party CSS

Contents

CSS

- What is CSS?
- Linking CSS to HTML
- CSS Selectors
- CSS Properties
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout
- Alternate Style

CSS Selectors

- CSS1 introduced the initial set of selectors. Supporting:
  - rules for element types, specific id values, generic classes
  - grouping and contextual selection of rules *(combinators)*
  - some pseudo classes
- CSS2 added several new selector types. Allowing:
  - more **power** and **control** over rule application.
  - element <u>content</u> to control rule application.

- CSS3 provides improved context, including different xmlns
  See overall summary CSS1-CSS3: http://www.w3.org/TR/css3-selectors/#selectors
- CSS4 evolving additional selectors as user interfaces change
  http://dev.w3.org/csswg/selectors-4/

CSS1 Selectors

CSS1 Selectors

Selector	Description	Example
element	Applies the style rule to *all elements* that match the element name. *Also called "tag style"*	`h1 {color: green;}`

#id	Applies the rule *only for the single element* that has this id value.
	eg. <tag **id="info">**
	Also called "id style"
		`#info {`
		`  background-color: red;`
		`}`

.class	Applies the rule to *any elements* that have the matching class value. eg. <<tag **class="note">** *Also called "class style"*	`.note {color: blue;}`
element.class	Applies the rule only to elements with the specified element name that *also* have the matching class value. eg.**<p class="note">** *Also called "tag specific class"*	`p.note {` `  border: 1px solid blue;}`

CSS: Selectors (id)

- Apply to an HTML **id** attribute

```
<p id = "copyright" >…</p>
```

- Select by using the id name prefixed with a hash "#"

```
#copyright { color : red; }
```

- An `id` *must be unique* in a webpage, so this selector will apply style rules to only one element in the page.

CSS: Selectors (class)

- Apply an HTML **class** attribute

```
<p class = "story" >…</p>
```

- Select by using the class name prefixed with a dot "."

```
.story { color : blue; }
```

- The `class` can be added to many elements, so one class style can be applied many times on a page.

- Can be made element specific by adding an element selector before the dot "."

```
p.story { color : blue; }
```

CSS1 Selectors

- CSS1 Selectors - Grouped & Contextual *("combinators")*

Selector	Description	Example
Grouping	Applies the rule to a *group of selectors,* (separated by *commas*)	`h1, h2, p {font: sans-serif;}` `header, nav {border-style : dotted;}`

Contextual	*Also called* Descendant combinator
	Applies the rule to the descendant (contained or 'nested') elements. (separated by *spaces*)
	Could be 3 levels deep.
	Most expensive selector in CSS
		`ul li {`
		`  list-style-type: upper-alpha;`
		`}`

Note: if any one of the selectors is invalid, the whole group may be ignored ☹

Any list item that is a descendant of an unordered list

Selectors: Pseudo-Classes

- pseudo-class selectors apply to elements based on characteristics other than an element name

Selector	Description	Example
:link	Selects all unvisited links	a:link
:visited	Selects all visited links	a:visited
:active	Selects the active link	a:active
:hover	Selects links on mouse over	a:hover
:focus	Selects the form element which has focus	input:focus
:first-child	Selects every <p> elements that is the first child of its parent	p:first-child

- Class selector can also be used as well, for example

```
a.classname:link { … }
```

Selectors - Pseudo Classes

Link Pseudo Classes (CSS1)

The pseudo-class concept was introduced to permit selection based on information that lies outside of the document tree **or** *that cannot be expressed using the other simple selectors.*

Selector	Description	Example
a:link	An unvisited hypertext link	`a:link {color: blue;}`
a:visited	A link that has already been visited	`a:visited { background-color: yellow; }`
a:active	An active link (as it is being 'clicked')	`a:active {color: red;}`

CSS Selectors - Dynamic Pseudo Classes

- CSS2 examples

Selector	Description	Example

`:hover`	Applied when the browser "cursor" is hovering over an element.
	(similar to a "mouseover" event)
		`a:hover {font-weight: bold;}`
		`p:hover {`
		`  border: 1px solid red;`
		`}`

`:active`	Applies while an element is being activated by the user.
	(eg, the time between when a user presses the mouse button and releases it.)
		`#mybutton:active {`
		`  color: red;`
		`}`

`:focus`	Applies when an element receives "focus" – commonly used with form elements like `<input … />`.	`input:focus {`
		`  background-color: white;`
		`}`

CSS Selectors - Pseudo Elements

- Pseudo-elements selects aspects of a document that are not classified by elements

Selector	Description	Example

`:first-line`	The first line of content (text) contained within the selected element (acts as a pseudo element)	`p:first-line {`
		`  **font-weight:** bold;`
		`}`

`:first-letter`	Treats the very first character (letter) of element content as a separate pseudo element and applies the rule.	`p:first-letter {`
		`  **color:** red;`
		`  **font-size:** 150%;`
		`}`

CSS Selectors

■ CSS2 Selectors

Selector	Description	Example

*	Wildcard or universal selector, used to apply a rule to any element, *or*
	contextually, any element within a parent element . ie. as a descendant combinator
		`* { background-color: red; }`
		
		`div * span {`
		`  background-color: blue;`
		`}`

>	Child combinator
	Match a **directly** enclosed child element
	(eg. **only** body > p *not* body > div > p)
		`body > p {`
		`  font-size: 12pt;`
		`}`

+	Adjacent sibling combinator Match an adjacent sibling element, (eg. first paragraph following a level 2 heading)	`h2 + p { color: blue; }`

[ ]	The attribute selector. *Very powerful!*
	= for an exact match,
	~= for partial matches,
	|= for an item in a space separated list
		`a[href] { color: green;}`
		`a[href~="http://"] { … }`
		`p[lang|="en"] { … }`

CSS2 Selectors

CSS2 Selectors - Pseudo Classes

Selector	Description	Example
`:first-child`	Match the first child contained in an element.	`p:first-child { color: blue; }`
`:lang`	Language dependent style application.	`*:lang(fr) { color: blue; }` `*:lang(en) { color: green; }`

CSS2 Selectors - Pseudo Elements

Selector	Description	Example
`:before`	Place content before an element	`div:before {` `  content: url(header.gif); }`
`:after`	Place content after an element	`div:after {` `  content: url(footer.gif); }`

CSS3 Selectors

- CSS3 has introduced a wide range of powerful selectors

  e.g string selectors, more pseudo-classes, …

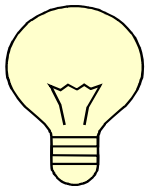
Provides **very** powerful access to objects, eg. third row of a table

- Now widely supported by most browsers

  http://www.w3schools.com/css/css3_intro.asp

Cascading: Hierarchy and Inheritance

- CSS is applied to the *HTML document structure*.
- Some style properties that are applied to a "parent" element will be inherited by its "children" elements.
- Not all style properties are inherited by children ...
  - *Foreground* properties *are inherited* (color, font-weight etc),
  - *Background* and *"box model"* properties *are not inherited*
    (unless you specifically set them to be inherited…)



Because the default background properties of an element are usually "transparent", you will still see the parent background properties

CSS: Hierarchy and Inheritance Example

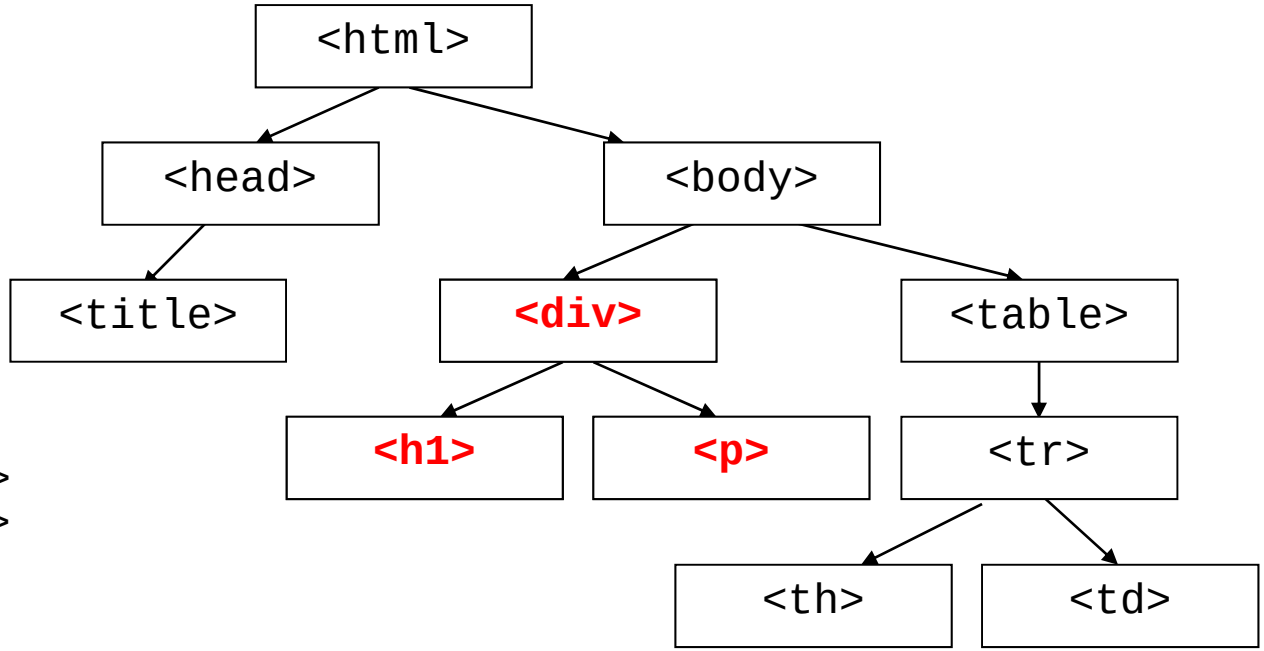
Consider the document hierarchy created in our simple HTML.

- When we apply this style rule to the document:

```
div { color: red; font-weight: bold; }
```

  - The rule will set all **div** elements to be a red foreground colour with bold text.
  - The red bold properties will be inherited by the child **h1** and **p** elements.

```
<html>
<head>
  <title>...</title>
</head>
<body>
  <div>
    <h1>...</h1>
    <p>...</p>
  </div>
  <table>
    <tr> <th>...</th>
         <td>...</td>
    </tr>
  </table>
</body>
</html>
```



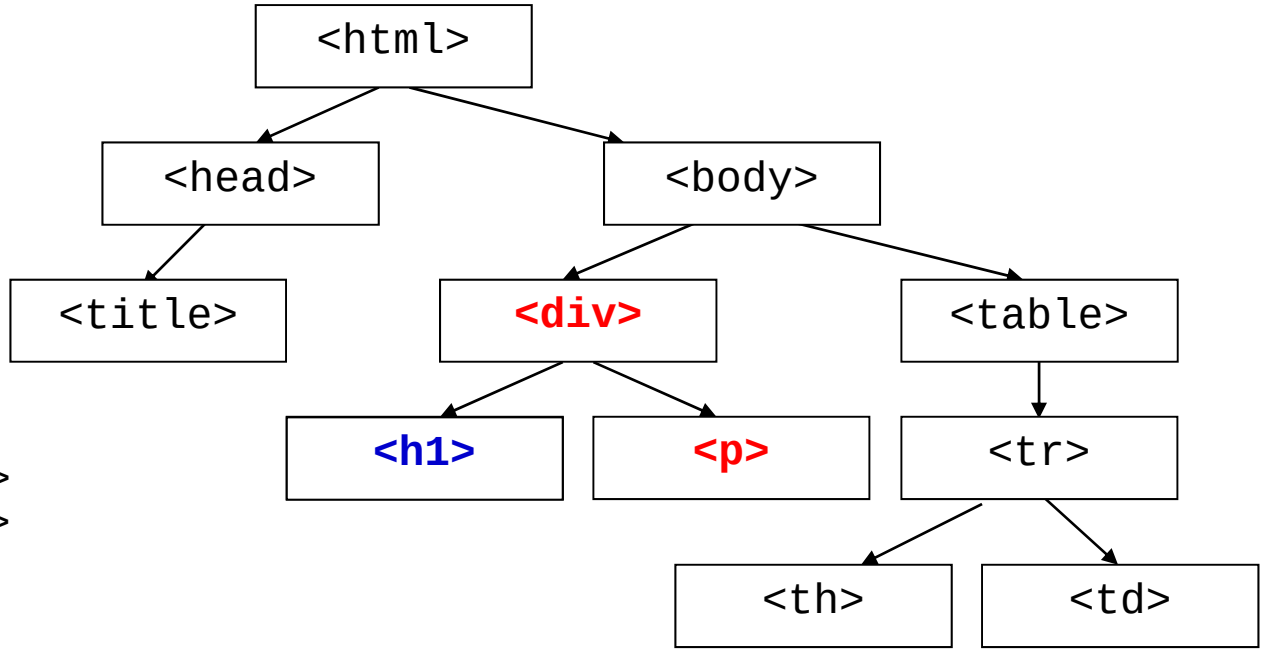
CSS: Hierarchy and Inheritance Example

- If we specify another style rule as well:

```
div { color: red; font-weight: bold; }
h1 { color: blue; }
```

  - This will set all **h1** elements to the foreground colour blue;
  - This new rule will override the existing inherited red colour.

```
<html>
<head>
  <title>...</title>
</head>
<body>
  <div>
    <h1>...</h1>
    <p>...</p>
  </div>
  <table>
    <tr> <th>...</th>
         <td>...</td>
    </tr>
  </table>
</body>
</html>
```



Contents

CSS

CSS: Property Groups

- Animation
- Background
- Border and outline
- Box
- Color
- Content Paged Media
- Dimension
- Flexible Box
- Font
- Generated content
- Grid
- Hyperlink
- Linebox
- List
- Margin
- Marquee
- Multi-column
- Padding
- Paged Media
- Positioning
- Print
- Ruby
- Speech
- Table
- Text
- 2D/3D Transform
- Transition
- User-interface

CSS Properties

- CSS properties define which aspect of the *selected* HTML will be changed or styled
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout

CSS Units - Measurement

- *Relative* is used for styling screen webpages

```
e.g. h1 { letter-spacing: .5em;}
```

Unit	Abbr	Description	Example
EM	em	Height of the current font's default size	p {padding: 2em;}
Percentage	%	Works like em, where 100% is the default font size	p {line-height: 100%;}
Ex	ex	Height of letter *x* in the current font	p {margin: 25ex;}
Pixel	px	Pixel size of screen	p {font-size: 12px;}

http://www.w3.org/Style/Examples/007/units.en.html

CSS Units - Measurement

- *Absolute* is used for styling webpages for print
- Avoid for screen media

Unit	Abbr	Description	Example
centimetre	cm	metric centimetre	p {padding: 1cm;}
inch	in	US inch	p {margin: 1.25in;}
millimetre	mm	metric millimetre	p {word-spacing: 10mm;}
pica	pc	Equal to 12 points	p {font-size: 20pc;}
point	pt	Equal to 72 points in an inch	p {font-size: 24pt;

CSS Properties

- CSS properties define which aspect of the *selected* HTML will be changed or styled
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout

CSS Units - `color`

- We can specify **`color:`** in the following four basic ways:

Format	Description and Examples
name	Colour **names**. There are 16 basic colours (from the Windows VGA palette) Many others are now accepted by popular browsers, but best to use 'hex' colors.
	`h1 {color: red}` `p {color: green}`
#rrggbb *or* **#rgb**	Red, green and blue values in **hexadecimal** format Written in "hex' format in 6 or concise 3 character versions. Colour values between 00 and FF (or 0 and F)
	`hr {color: #FF0000} /* red */` `td {color: #00F} /* blue concise format - saves bandwidth */`
rgb(r,g,b)	rgb (red,green,blue) values in **decimal** with the **`rgb()`** command. Units between 0 and 255
	`.info {color: rgb(255,0,255); } /* purple info class */`
rgb(r%,g%,b%)	rgb (red,green,blue) values in **percentage** units with the **`rgb()`** command. Unit values between 0% and 100%.
	`em {color: rgb(100%,0%,100%); } /* purple emphasised text */`

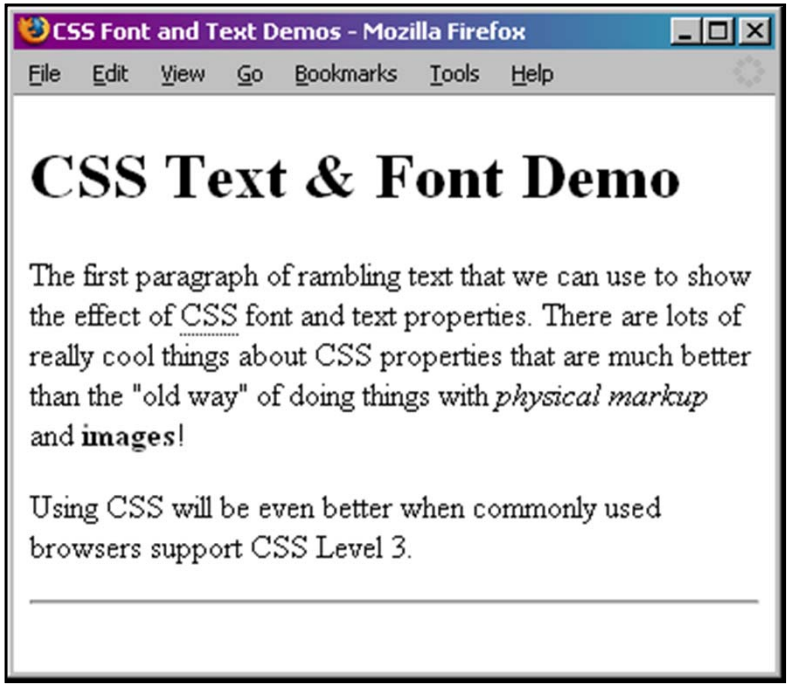
CSS3 Named Colors: http://www.w3.org/TR/css3-color/#svg-color

CSS Properties

- CSS properties define which aspect of the *selected* HTML will be changed or styled
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout

CSS Font and Text Example

```
...
<body>
<h1>CSS Text &amp; Font Demo</h1>
<p class="intro">The first paragraph of rambling text that we can use to show
   the effect of <abbr title="Cascading Style Sheets">CSS</abbr> font and text
   properties. There are lots of really cool things about CSS properties that
   are much better than the &quot; old way &quot; of doing things with
   <em>physical markup</em> and <strong>images</strong>!</p>
<p id="tag">Using CSS will be even better when commonly used browsers support
   CSS Level 3. </p>
<hr />
</body>
</html>
```



CSS Font and Text Example

font-variant: small-caps;

font-size: 150%;

text-decoration: underline;

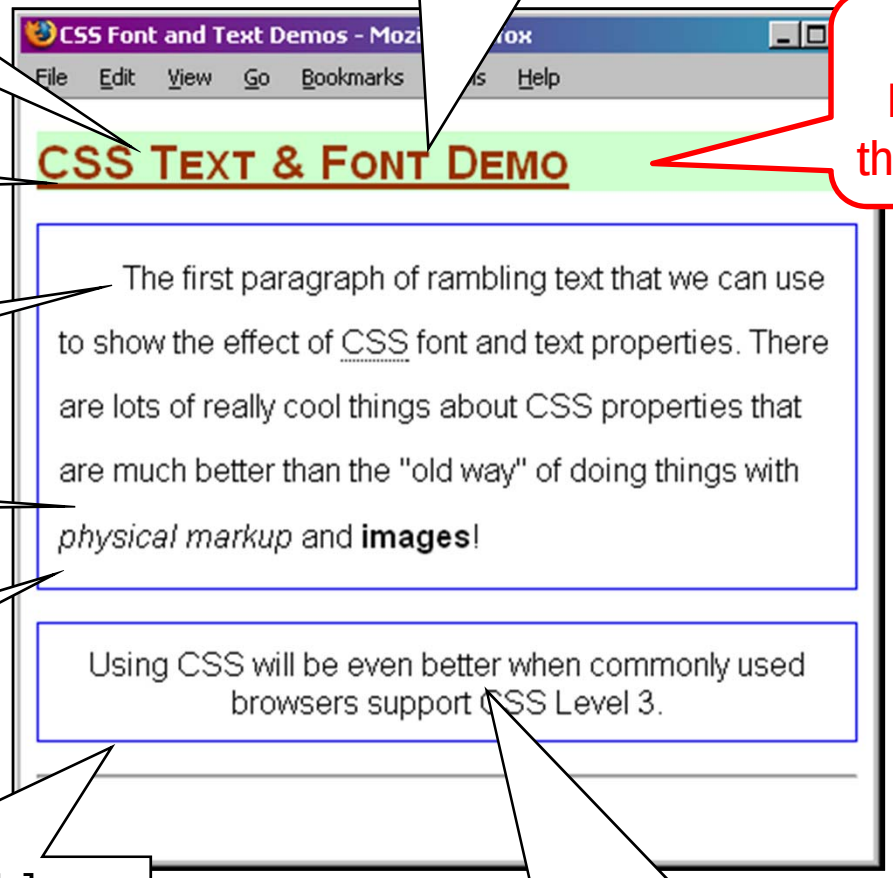
text-indent: 2em;

line-height: 200%;

padding: 10px;

border: 1px solid blue;

text-align: center;



Usability Issue with this example!

CSS Font and Text Example – Family, Colour, Spacing

```
<!DOCTYPE HTML >
<html lang="en">
<head>
<title>CSS Font and Text Demos</title>
<style type="text/css" >
h1, p { font-family: Arial, Helvetica, sans-serif; }

/* shows the "block" in a background color */
h1 { background-color: #CCFFCC; color: #993300; }
/* percentage of the "normal" text size */
h1 { font-size: 150%; }
/* note that the h1 content is NOT in CAPITALS! Cool!*/
h1 { font-variant: small-caps; }
/* not good - confuses users - they think it's a hyperlink! */
h1 { text-decoration: underline; }

p.intro { line-height: 200%; }
/* "em" units will scale nicely with font size! */
p.intro { text-indent: 2em; }
/* note border values. padding between text and border */
p { border: 1px solid blue; padding: 10px; }

/* only effects the #tag element */
#tag { text-align: center;}
</style>
</head>
...
```

grouping selector h1, p

element selector h1

It would be better if these rules were grouped into one rule.

class selector .intro

element selector p

id selector #tag

CSS: Generic & Specific Fonts

- A *specific font* is a font such as “Times New Roman”, “Arial”, or “Garamond”. *Specific fonts are installed on a user’s computer, so availability depends on the user’s machine.*
- A *generic font* refers to the font’s general appearance such as: “serif”, “sans-serif”, “monospace”, “cursive” or “fantasy”.

Five Generic Fonts

	Font Samples		
serif	defg	defg	defg
sans-serif	defg	defg	defg
monospace	defg	defg	defg
cursive	defg	defg	defg
fantasy	defg	defg	DEFG

FF

The lines on letters are called “serifs”. “sans-serif” means “without serifs”.

CSS Font Family

- To specify the font, we use the **font-family** property.
  Example:

```
p {
   font-family: Verdana; }
```

Any font names containing characters such as whitespace, font must be quoted.
eg. "Times New Roman"

- If you specify a "specific font" a user might not have it on their device, so you *may* list alternatives, and *should* include a final "generic font"

```
h1 {
   font-family: Verdana, Arial, Helvetica, sans-serif;
}
```

Validation Warning if no generic font

The preferred specific font

Alternative specific fonts

A "generic font" alternative

- Fonts can also be downloaded using @font

```
@font-face {
   font-family: myfont;
   src : url("http://www.allfont.com/myfont.ttf");}
```

See examples: http://www.w3.org/TR/css3-fonts/#font-resources

See also WOFF / woff-fonts: http://www.w3.org/TR/WOFF/

CSS Font Properties

- **font-size:**

  xx-small, x-small, small, **medium**, large, x-large, xx-large, smaller, larger, [length], [%]

- **font-style:**

  normal, italic, oblique

- **font-weight:**

  normal, bold, bolder, lighter, [100,200, **400**, … , 900]

- **font-variant:**

  normal, small-caps

- **font-stretch:**

  normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded

- **line-height:**

  normal, [number], [length], [%]

= default values

CSS Font Property

- We can write several font-properties in a shorthand, single declaration, format:

  `font: [style] [variant] [weight] size [/line-height] family`

  - **`size`** and **`family`** values *are required*
  - The values in square brackets `[ ]` are optional.
  - The first three values can be specified in any order
  - `/line-height`, if used, *must* come straight after `size`

- Example:

```
p {
    font: italic normal bold 10pt/14pt Helvetica, sans-serif;
}
```

Note: Be aware of ***default*** and ***required*** values for style properties.

Text Alignment

- text-indent: <value>; (first line of paragraph)
  - text-indent : 2em;
  - text-indent : -2em; (for hanging indent)
- text-align : left | right | center | justify;
  - text-align : center;
    - Justify is not supported by all browser
- line-height : normal | <value>;
  - line:-height : 150%; (1.5 spacing assuming font size is normal)
- text-decoration: none, underline, overline, line-through, blink

'underline' is default for 'a' element

- *Also see CSS3 text-decoration:* http://www.w3.org/TR/css-text-decor-3/

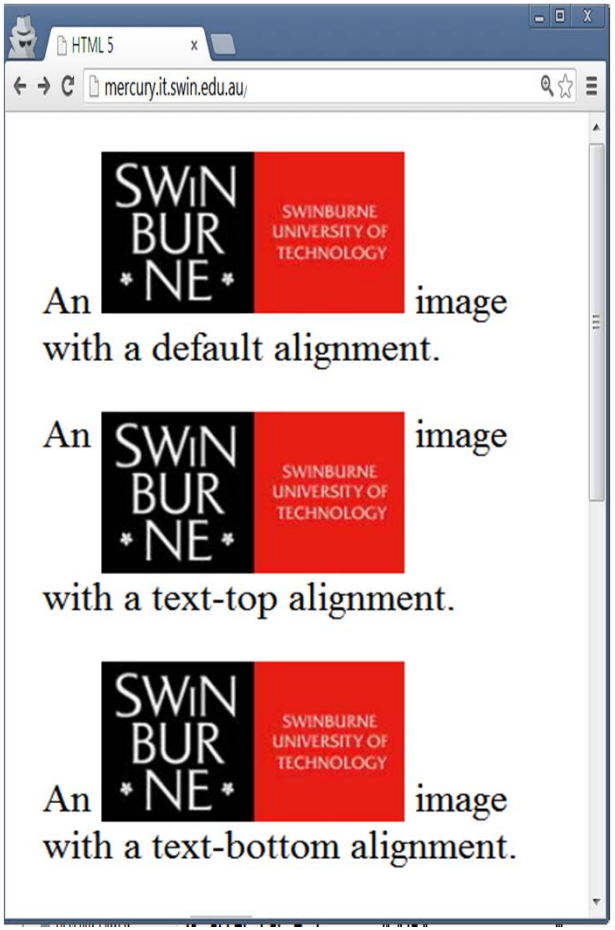
Alignment of Graphics with Text

- vertical-align : baseline | sub | super | top | text-top | middle | bottom | text-bottom | <value>
  - vertical-align : super; (superscript)
  - vertical-align : middle; (used on table cell)
  - vertical-align : text-top; (used on images)

Alignment of Graphics with Text

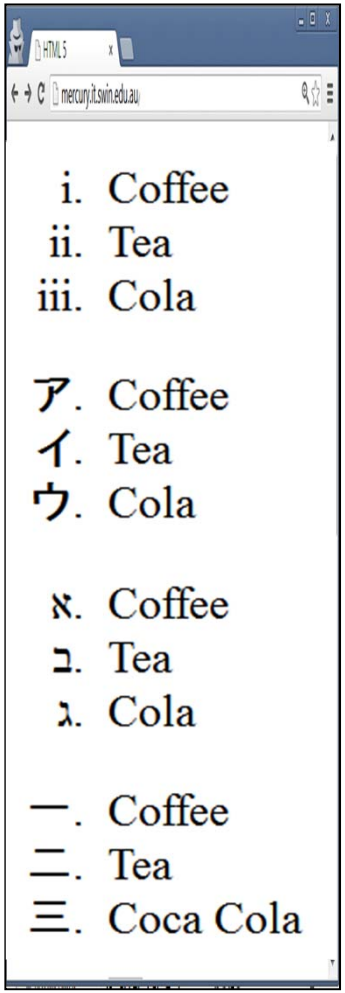
```
<p>An <img src="logo.jpg" alt="SUT" width="120"
height="64" /> image with a default
alignment.</p>

<p>An <img class="top" src=" logo.jpg"alt="SUT"
width="120" height="64" /> image with a text-top
alignment.</p>

<p>An <img class="bottom" src=" logo.jpg "
alt="SUT" width="120" height="64" /> image with
a text-bottom alignment.</p>

</body>
```

```
CSS
  img.top     {
        vertical-align:text-top;}

  img.bottom {
        vertical-align:text-bottom;}
```



Creating List styles with CSS

- list-style-type : none | disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha
  - ul.a {list-style-type:lower-roman;}
  - ul.b {list-style-type:katakana ;}
  - ul.c {list-style-type:hebrew ;}
  - ul.d {list-style-type:cjk-ideographic ;}



Creating List styles with CSS

- list-style-image: none | <url>
  - list-style-image : url("spade.gif");
- list-style-position : inside | outside;
  - list-style-position : inside;
- list-style : [type] [position] [image];
  - list-style : lower-alpha inside;

http://css-tricks.com/almanac/properties/l/list-style/

CSS Properties

- CSS properties define which aspect of the *selected* HTML will be changed or styled
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout

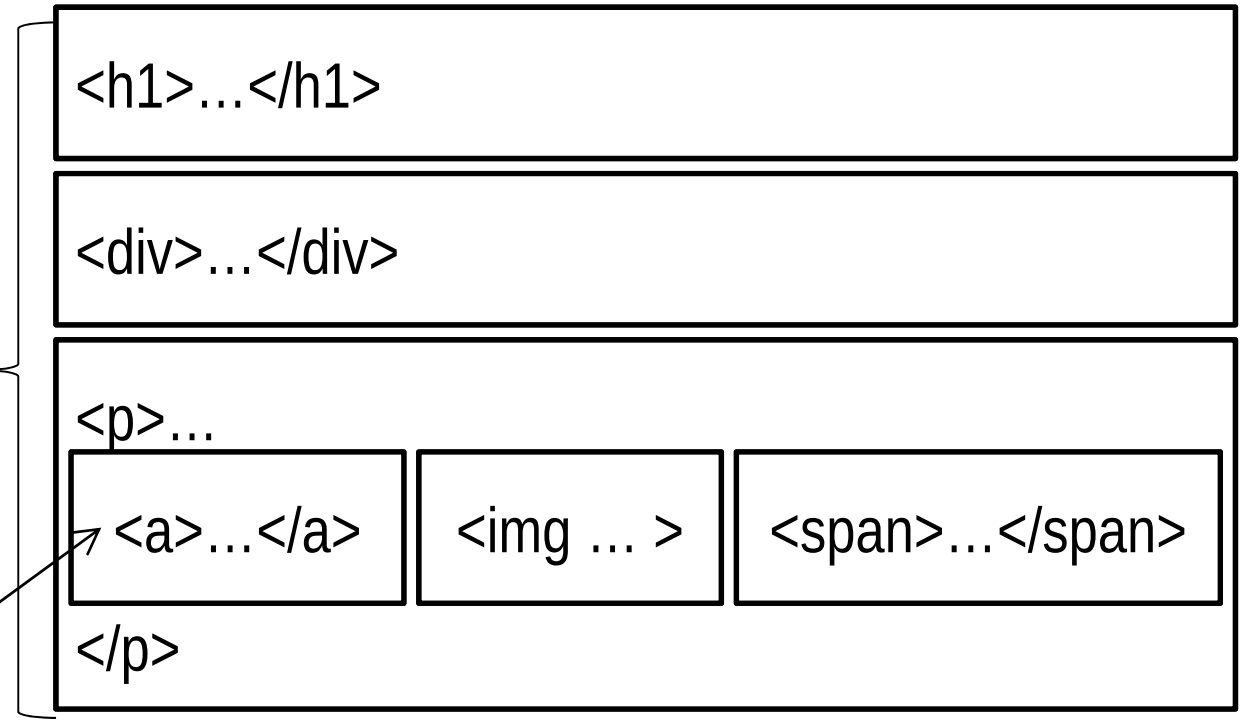
CSS: Visual Format and Box Models

- Visual formatting model describes how the element content boxes should be displayed
  - **Block-level elements** appear as blocks *such as paragraphs*
  - **Inline-level elements** are contained within block-level elements, *such as anchors*
- *Box model* describes the rectangular boxes that contain content on a web page

Model: Visual Formatting

- Arrangement is top to bottom left to right according to how the elements are ordered

```
<h1>…</h1>
<div>…</div>
<p>…
  <a>..</a>
  <img … />
  <span> … </span>
</p>
```



Block-level elements

Inline-level elementsl

Model: Visual Formatting – Display

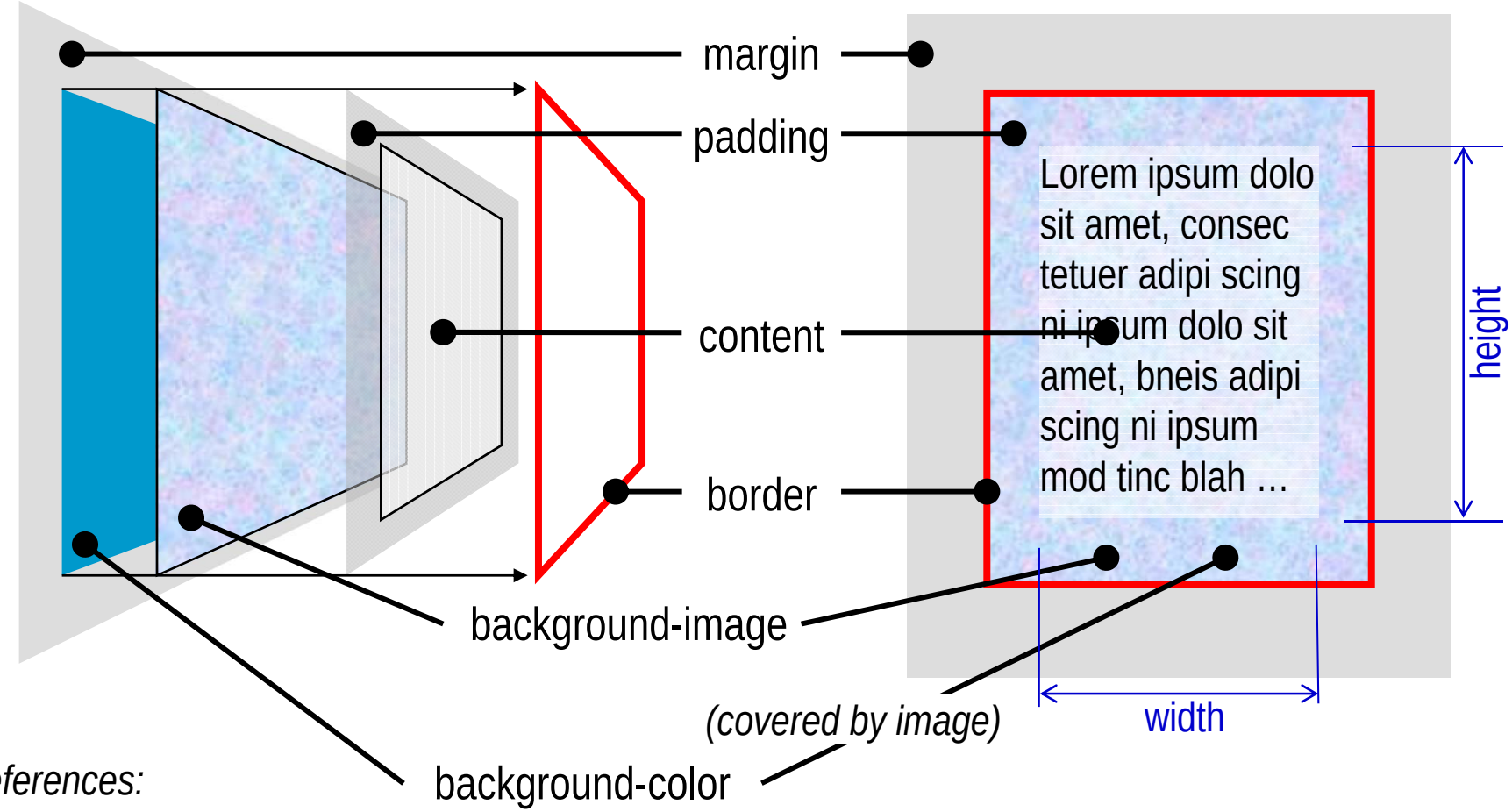
- display : inline | block | list-item | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none
  - display: block
    used to change an inline element to a block level element,
  - display:inline
    used to change a block level element to an inline element
  - display: table values
    used to create table-like displays using CSS
    (HTML tables are only for tabular data)
  - display: none value hides the element from display

http://www.w3schools.com/cssref/tryit.asp?filename=trycss_display_inline

http://css-tricks.com/almanac/properties/d/display/

The CSS Box Model

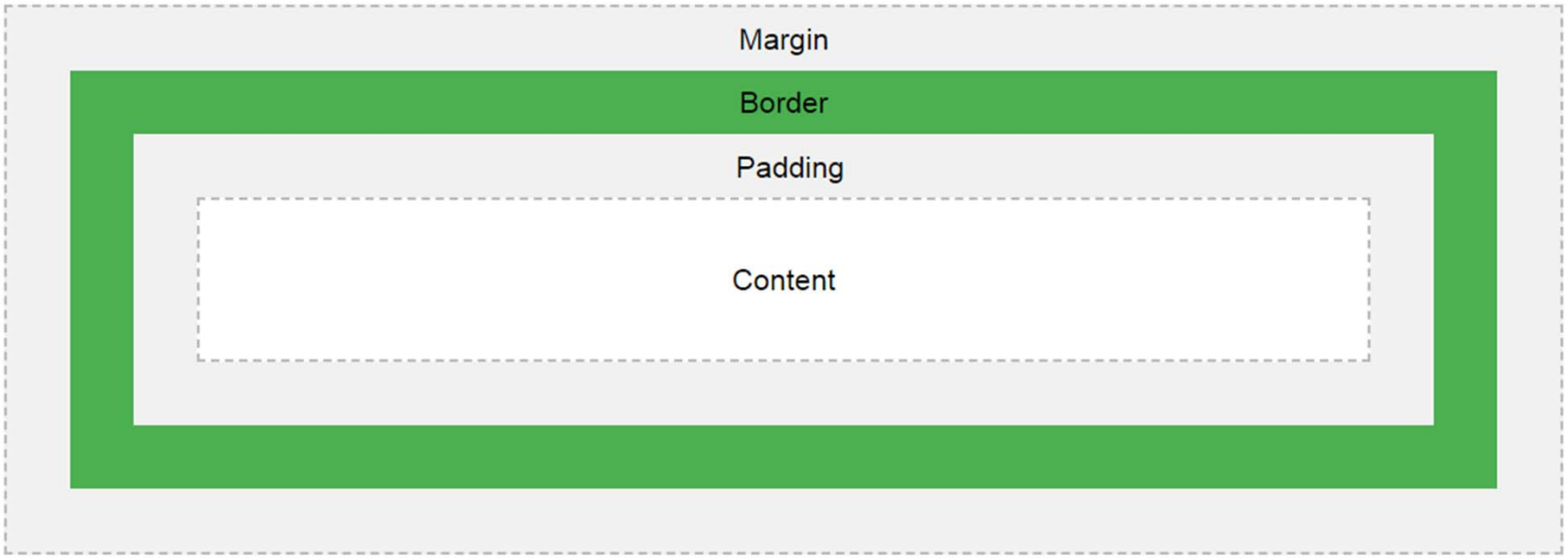
- Below is a representation of the CSS box model.



- *References:*
- CSS2.1 Box Model http://www.w3.org/TR/CSS2/box.html
- CSS Backgrounds and Borders Module Level 3 http://www.w3.org/TR/css3-background/

The CSS Box Model

- Laid out according to the visual formatting model.



Example

```
div {
  width: 300px;
  border: 25px solid green;
  padding: 25px;
  margin: 25px;
}
```

CSS Background

- CSS box model background:
  - It is *behind* the content (text, image etc)
  - It extends to the border, so it *includes* the "padding".
  - It does *not* extend past the border (where the "margin" is).
- **`background-color:`** `[colour-rgb], [colour-hex], [colour-name],` **`transparent`**
  - The default background color `transparent` allows the parent element (content / background etc) to show through as the background.
- **`background-image:`** `[url()], none`

  Example:

```
body { background-image: url(tiles.gif);}
```

  - If we use a `background-image`, it will be presented over the top of the `background-color`.

CSS Background

- **`background-position:`**
  `top, bottom, left, right, center,`
  `[x-% y-%], [x-pos y-pos]`

  Note: Percentages will position an image based on center of the image, however constants (eg. 30px) use the top left corner of the image.

  Example: 50% horizontal (center), 30px vertical (down)

  ```
  background-position: 50% 30px;
  ```

- **`background-repeat:`**
  `repeat, repeat-x, repeat-y, no-repeat`

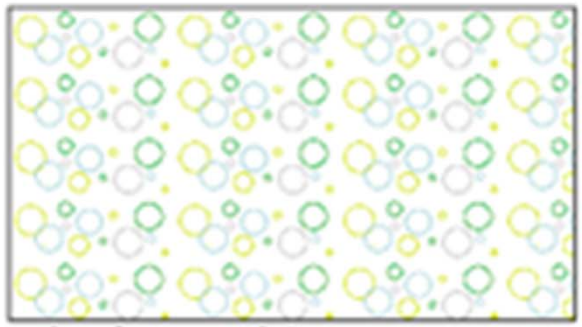
  Example: Repeats the image along the x (horizontal) axis

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

CSS Background Image Repeat example

- *repeat position, background image, fixed position…*

background image



background-image: repeat

background-image: repeat-x

background-image: repeat-y

background-image: no-repeat

CSS Background

- **background-attachment:**
  `scroll, fixed`

  Example: The background image will stay in the same window location regardless of the browser window scroll.

```
div {
  background-image: url(flowers.gif);
  background-attachment: fixed;
}
```

- **background:** *(grouped multiple property short-form)*

```
background-color background-image background-repeat*
  background-attachment* background-position*
```

Example:

```
body {
  background:
  black url(tile.gif) no-repeat top left;
}
```

*Note: be aware of **default** and **minimum** values.*

CSS Border

- Border surrounds the elements padded content
  - Borders are separated from other elements by the margins.
  - **`border:`** *(grouped short form, applied to all borders)*

    `border-width border-style border-color`

    Example:

    ```
    p { border: 1px dashed #000; }
    ```

    *Note: be aware of **default** and **minimum** values.*
  - **`border-style:`** (= `border-`*`[all]`*`-style`)

    `none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`
  - **`border-color:`** (= `border-`*`[all]`*`-color`)

    `[colour-rgb()], [colour-hex], [colour-name]`

CSS Border

- We can also specify border grouped properties for individual sides.
  - **`border-`*`[top,right,bottom,left]`*`:`**

    (grouped short form, three properties at once for a side)

    `border-width border-style border-color`

    Example:

    ```
    h1 {border-bottom: 1px double green; }
    ```

  - **`border-`*`[top,right,bottom,left]`*`-width:`**

    `thin, medium, thick, [length]`

    trouble

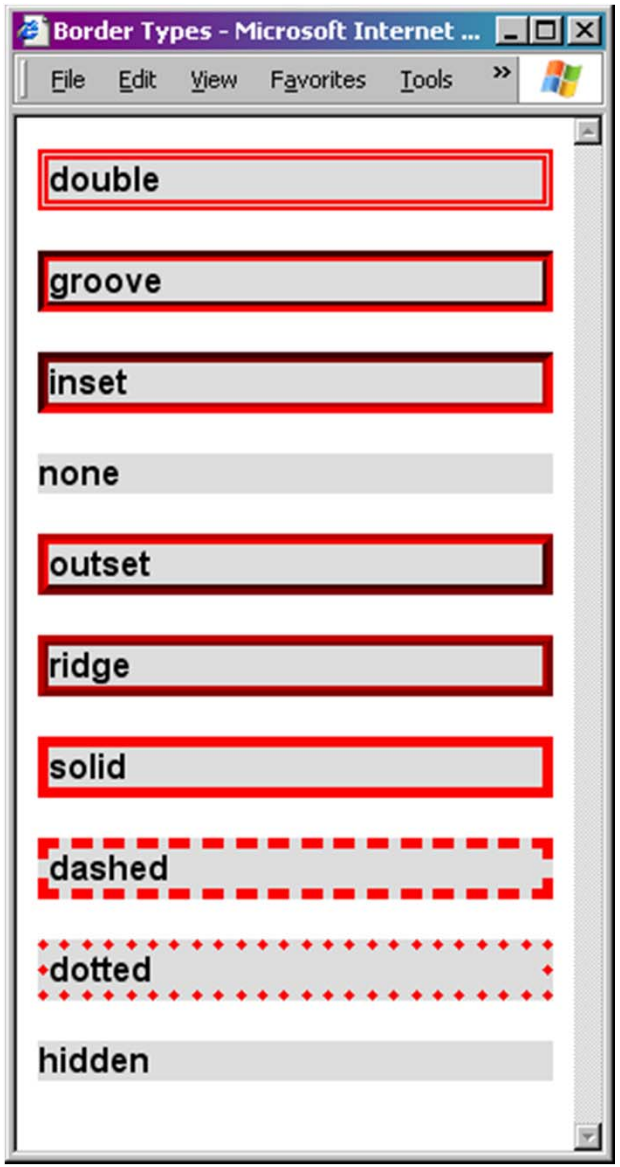
  - **`border-`*`[top,right,bottom,left]`*`-style:`**

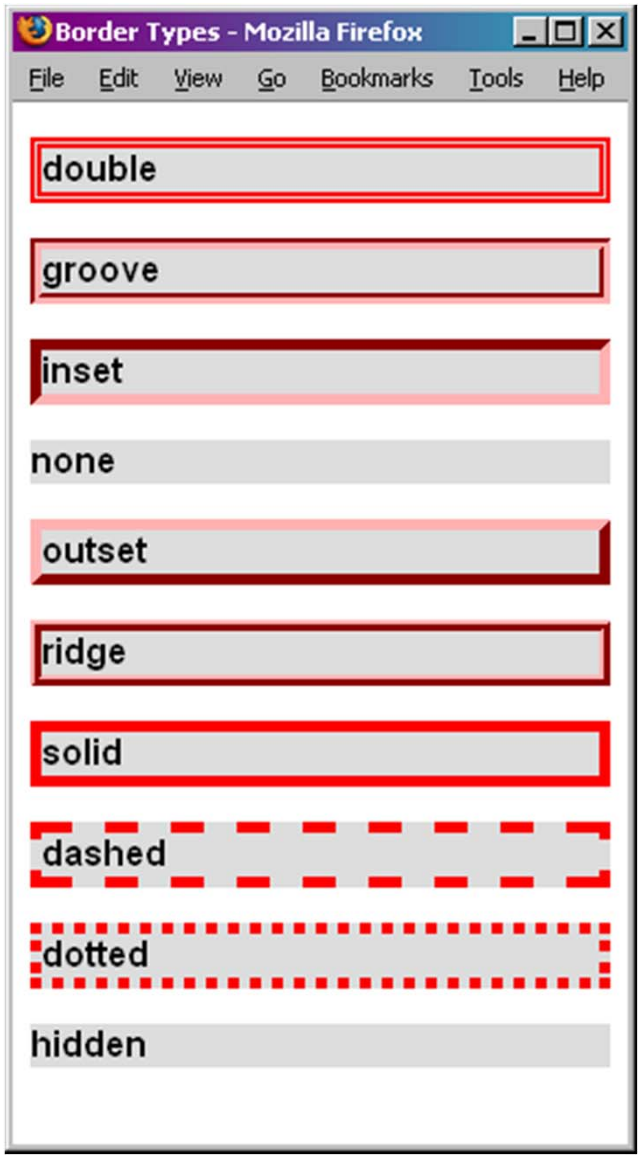
    `none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`

  - **`border-`*`[top,right,bottom,left]`*`-color:`**

    `[colour-rgb()], [colour-hex], [colour-name]`

CSS Border Examples





CSS Box Dimensions

- The **`width`** and **`height`** properties can be used to specify the dimensions of block (or "replaced") elements.
  - *They are not "valid" properties for inline elements*
  - If content requires more space than the **`width`** and **`height`** you have specified, the display behaviour is specified by the **`overflow`** property.
  - **`width:`**
    `auto, [length], [%]`
  - **`height:`**
    `auto, [length], [%]`
  - **`max-width, min-width:`**
    `none, [length], [%]`
  - **`max-height, min-height:`**
    `none, [length], [%]`

Note: **`width:`** and **`height:`** (and respective min/max properties) apply to the width and height of the *content box* of the element.
The **`padding:`** and **`border:`** of the element are *outside the specified width and height.*

CSS Margin

- Margin allows us to separate elements.
  - Margins do not act as a "fixed buffer" between elements but ensure a *minimum* separation. The margins of adjacent elements *overlap* and the *biggest margin* is the gap that is displayed.
- There is a grouped "short form" for the margin properties :
  - **`margin:`** (set each margin, clock-wise order from top)

    `margin-top margin-right margin-bottom margin-left`

    Example:

    ```
    p {margin: 4px 10px 4px 10px; }
    ```
- Individual margins can be set if needed:
  - **`margin-`***`[top,right,bottom,left]`***`:`**
    `auto,` *`[length], [%]`*

    Example:

    ```
    li { margin-top: 4em; }
    ```

CSS Margin

- The effect of multiple margin values:
  - Single margin value, applied to all sides:

```
p { margin: 4px; }
```

  - Two margin values:

```
p { margin: 10em auto; }
```

    - first value (10em) sets the top and bottom margins
    - second value (auto) applied to the left and right margins
  - Four margin values in clock-wise order (top, right, bottom, left):

```
p { margin: 4px 10px 6px 10px; }
```

trouble

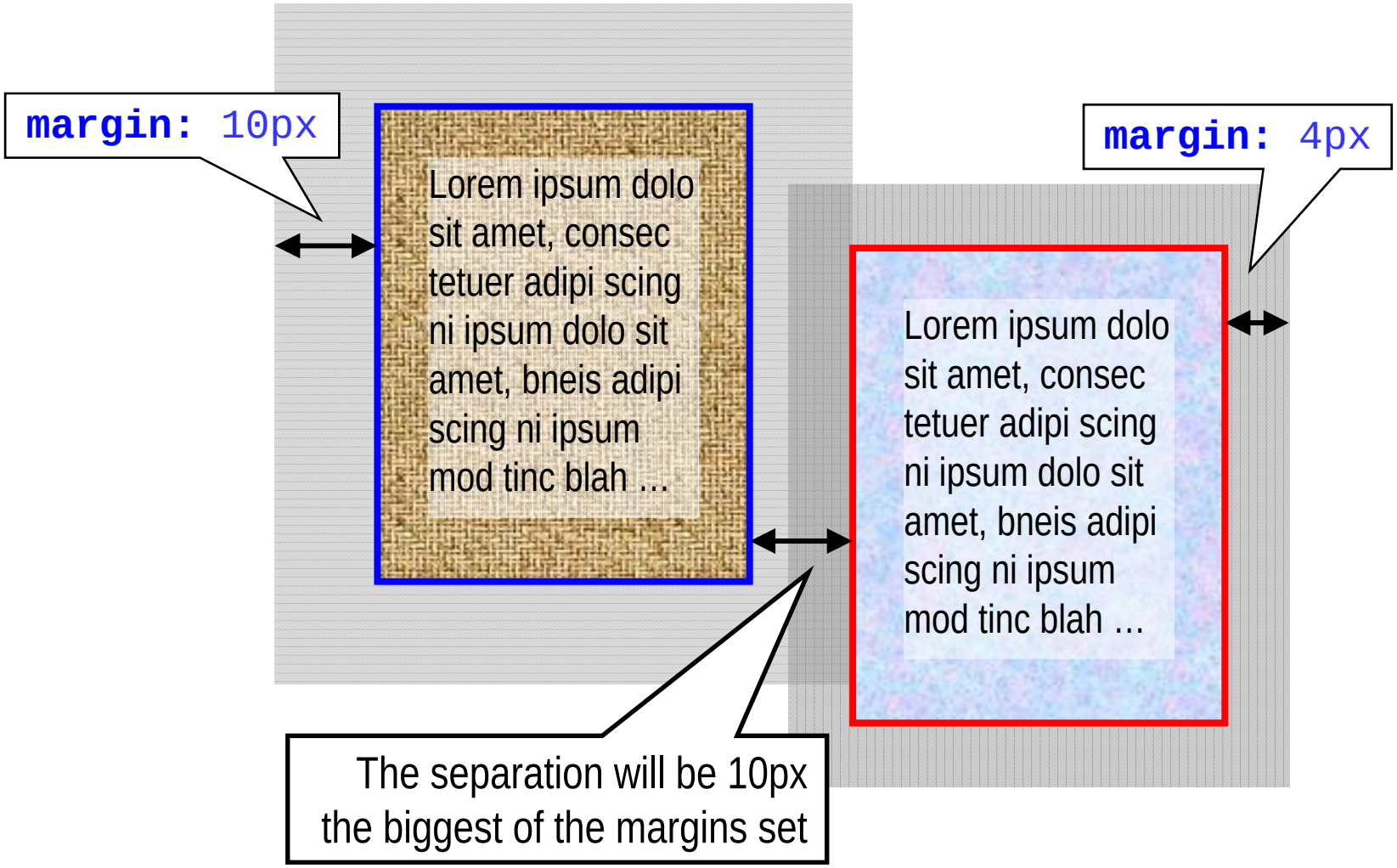
- We can use the "**auto**" margin value to centre an element:

```
table { margin-left: auto; margin-right: auto; }
```

CSS Margin Example

- *Margin is a minimum **separation distance** between elements.*

http://reference.sitepoint.com/css/collapsingmargins



CSS Padding

- Padding is placed between the border and the content.

  (This stops text from being squashed next to the border!)

  - **`padding:`** (grouped short form, clock-wise from the top)

    ```
    padding-top padding-right
    padding-bottom padding-left
    ```

- Like margin, we can also use 1,2 or 4 values:

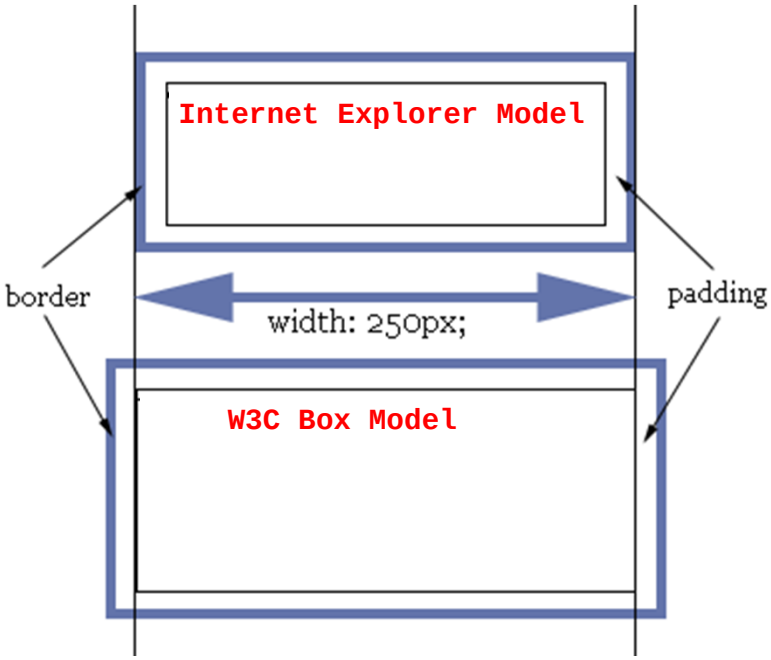
`padding:` `4px;`	4px applied to all sides
`padding:` `6px 4px:`	6px top and bottom, 4px left and right

- We can specify padding for individual sides if we need

  - **`padding-`***`[top,right,bottom,left]`***`:`** *`[length], [%]`*

CSS3 - Box Width (and Height)

- *In the W3C CSS2.1 specification, the box width is the **width of the content** - the padding and border is outside*



http://www.quirksmode.org/css/box.html

Note: Internet Explorer incorrectly treated the width as outside the border ☹

CSS3 – proposes

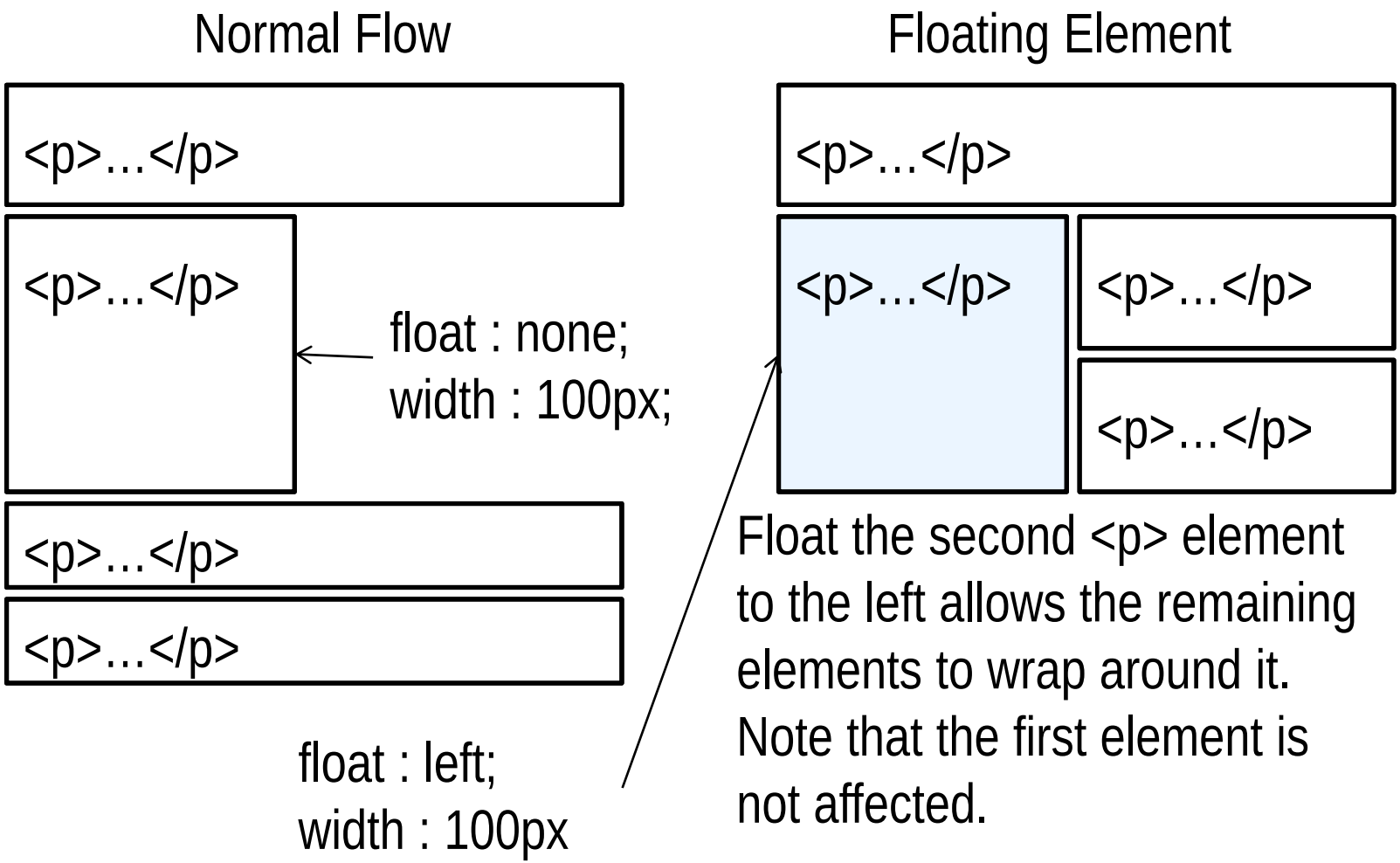
box-sizing: border-box ;
box-sizing: content-box;

Page Layout: Flow

- **Normal** is the default browser display of elements, this is one after the other
  - Block-level – vertically from top to bottom
  - Inline-level – horizontally from left to right
- **Float** is taking an element out of the normal flow
  - Non-floating elements remain in the normal flow
  - Originally intended to allow text to wrap around images, currently used for page layout

Page Layout: Flow

- float : none | left | right;



CSS Element Layout

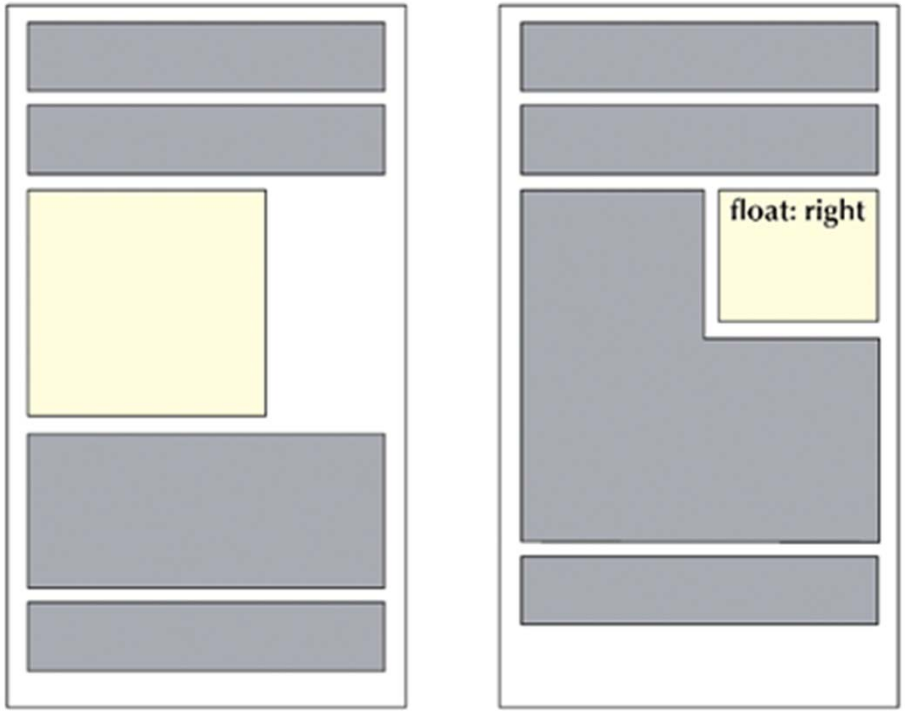
- **`float:`**
  `left, right, none`
  - Set an element to **`float`** against the parent border. Other block positions are unaffected, but block contents (eg. text) will flow around the floated element.
- **`clear:`**
  `left, right, both, none`
  - The **`clear`** property lets you position elements “clear” from other “floated” elements.

    Example: Make sure that the next “intro” paragraph is clear, both left and right, from any floated images:

```
p.intro {
  clear: both;
}
```

CSS Element Layout Example

- Float example, clear example



See also CSS Page Layout notes. eg.'float' div blocks into columns

http://css.maxdesign.com.au/floatutorial/

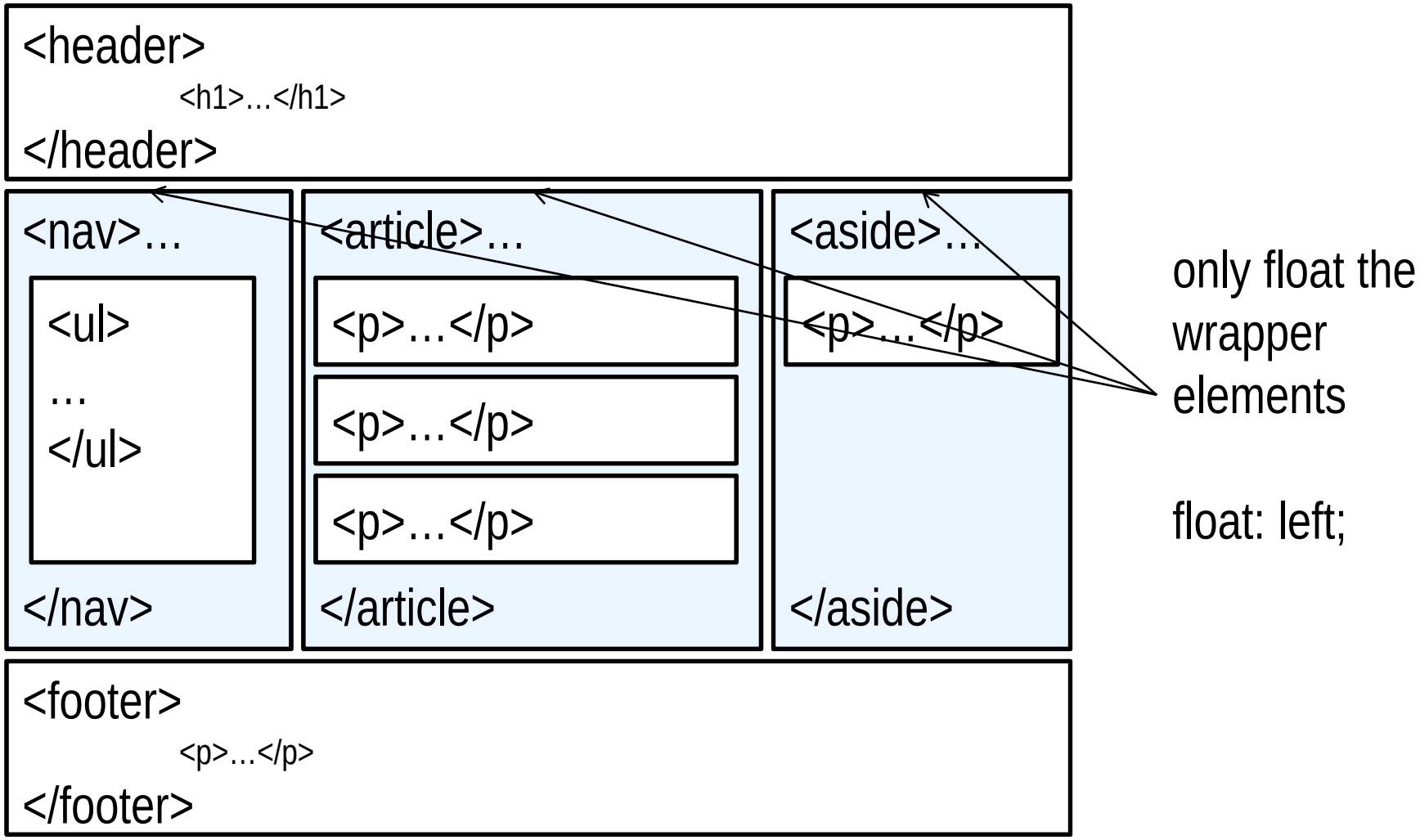
CSS Properties

- CSS properties define which aspect of the *selected* HTML will be changed or styled
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout

Page Layout: Wrapper Elements

- Wrapper elements are used to hold page pieces together
- Wrapper elements can be . With HTML5, these are <header>, <nav>, <article>, <main>, <section>, <aside> and <footer>
- If elements inside the wrapper element are floated,
  - margins are used to set the gutters between elements
  - the height of the wrapper is based on the maximum height of a non-floating element

Page Layout: Wrapper Elements

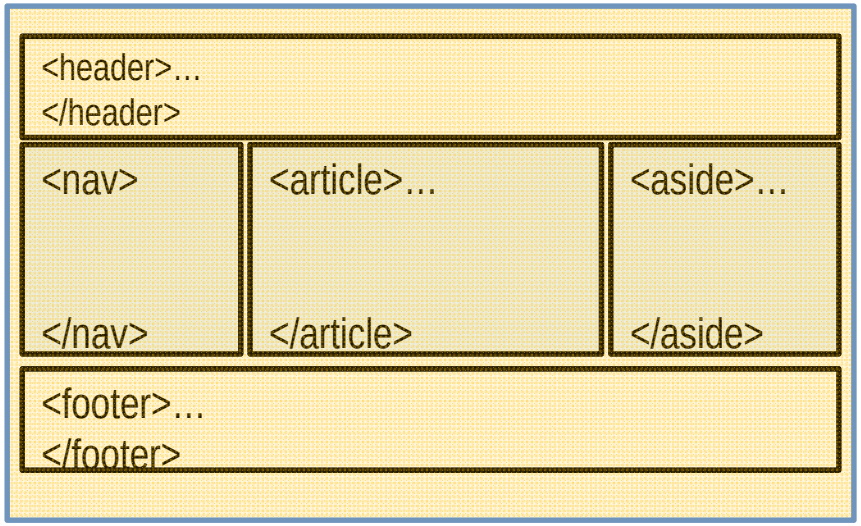


Page Layout: Design

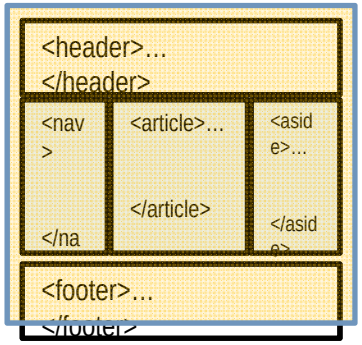
- **Fixed layout:** defines exact size of every element in **absolute** units such as pixels.
  - Gives precise control over appearance
  - Does not adapt to the size of the browser window
- **Fluid (Flexible/Liquid) layout:** one or more elements are set with **relative** units.
  - Layout adapts to the size of the browser window.
  - Typically related to **width** rather than height
  - Page content “flows” into free areas of the browser window

Page Layout: Design - Fluid

```
<header >…
 </header>
<nav >…
 </nav>
<article>…
 </article>
<aside>…
 </aside>
<footer>…
 </footer>
```



Adapts to the size of the browser window




```
header {width:100%;}
nav {width:25%;  float:left;}
article {width:50%; float:left;}
aside {width:20%; float:left;}
footer {width:100%; clear:both;}
```

CSS Positioning

- In CSS 2.1, a box may be laid out according to three positioning schemes: *Reference: http://www.w3.org/TR/CSS21/visuren.html*
  - Normal flow

    Includes block formatting of block-level boxes, inline formatting of inline-level boxes, and relative positioning of block-level and inline-level boxes.
  - Floats

    In the float model, a box is first laid out according to the normal flow, then taken out of the flow and shifted to the left or right as far as possible. Content may flow along the side of a float.
  - Absolute positioning

    In the absolute positioning model, *a box is removed from the normal flow entirely* (it has no impact on later siblings) and assigned a position with respect to a containing block.

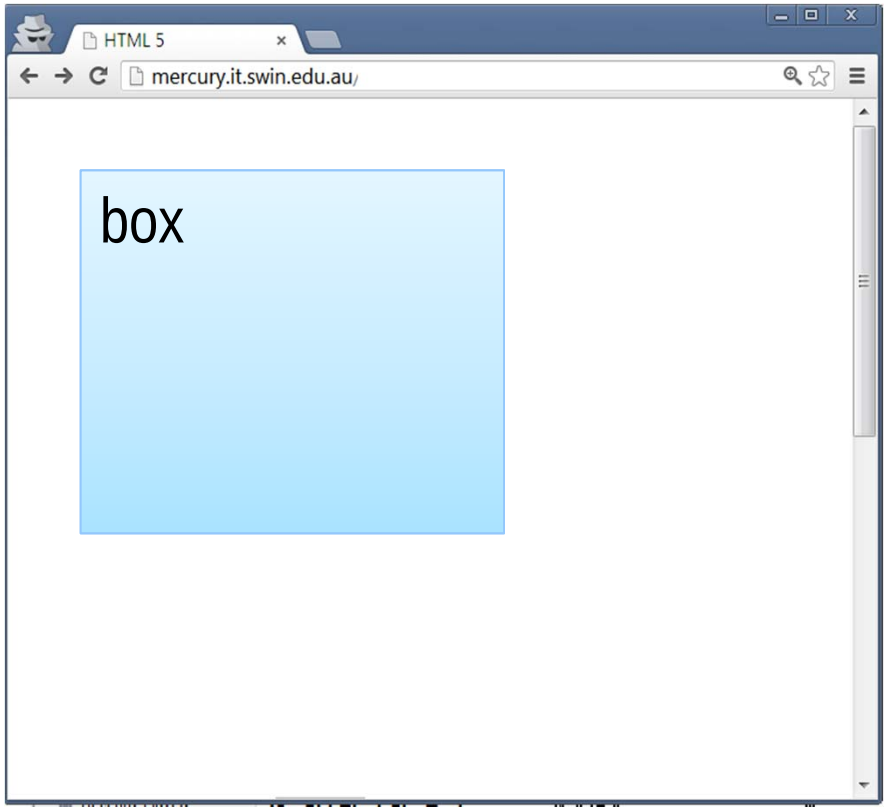
Page Layout: Position, Top and Left

- position: **static** | absolute | fixed | relative;
  - static is the default positioning of the elements as they appear in the document flow
  - relative positions the element relative to its normal position, (offsetting from static)
  - absolute positions the element relative to its first positioned ancestor element
  - fixed positions the element relative to the view port or browser window
- Used with top and left property
- Avoid `position:` unless really needed

Page Layout: Position, Top and Left

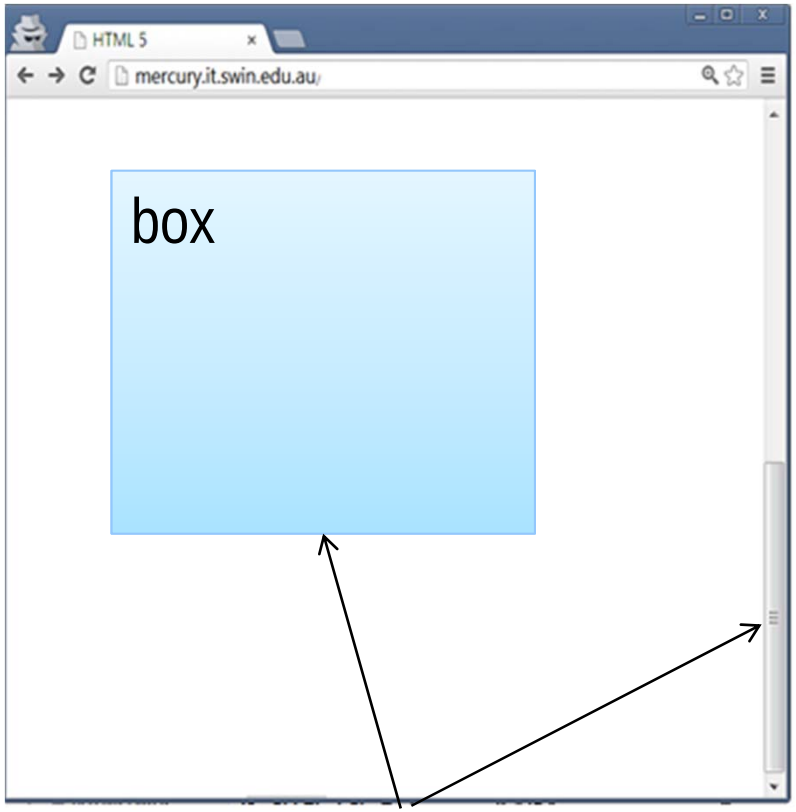
- top: auto | <value>;
- left: auto | <value>;

```
<div style="
width:100px;height:100px;
border:1px solid #black;
background-color:skyblue;

position:absolute;
top:100px;left:100px;">
box</div>
```



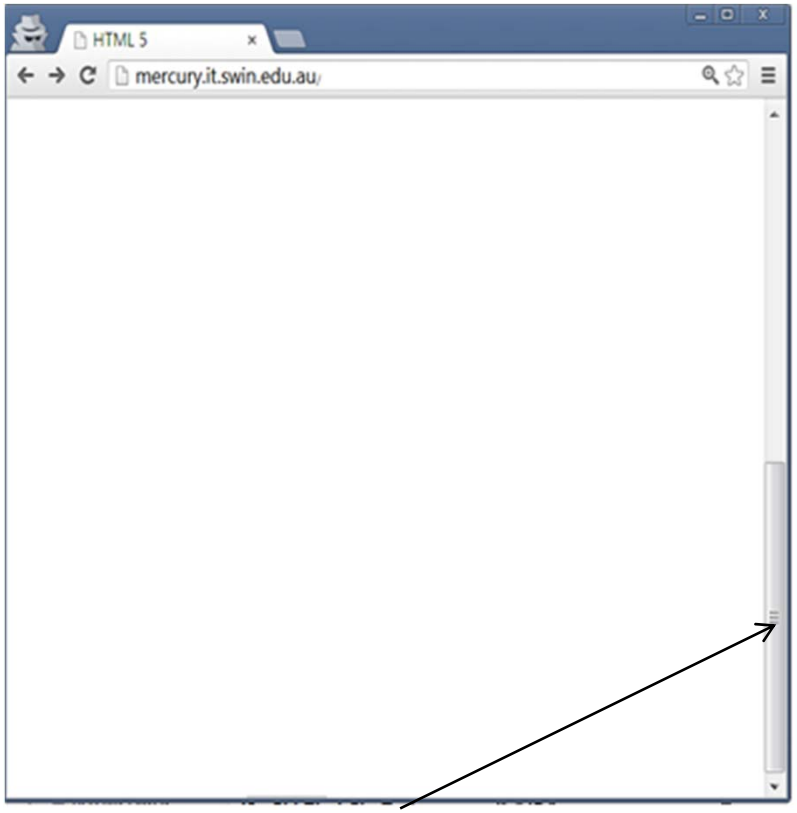
Page Layout: Position, Top and Left

- fixed



Relative to the window, stays on screen
Even if user scrolls down

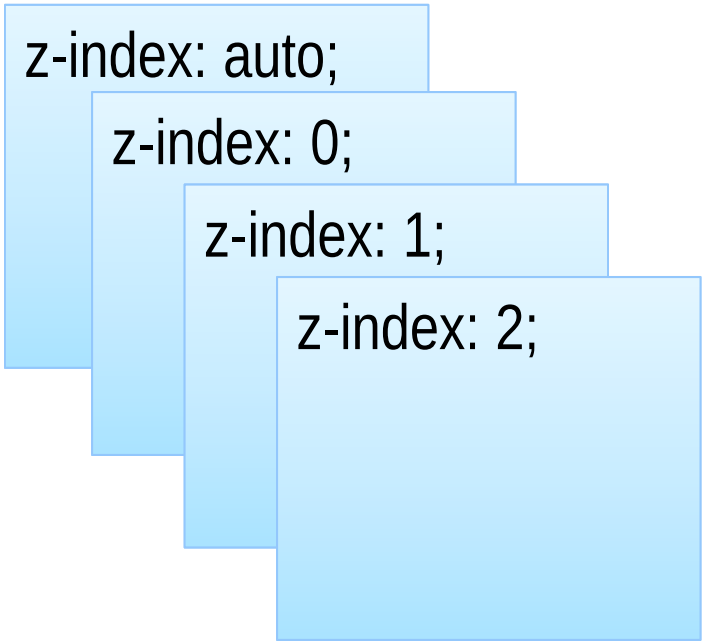
- absolute

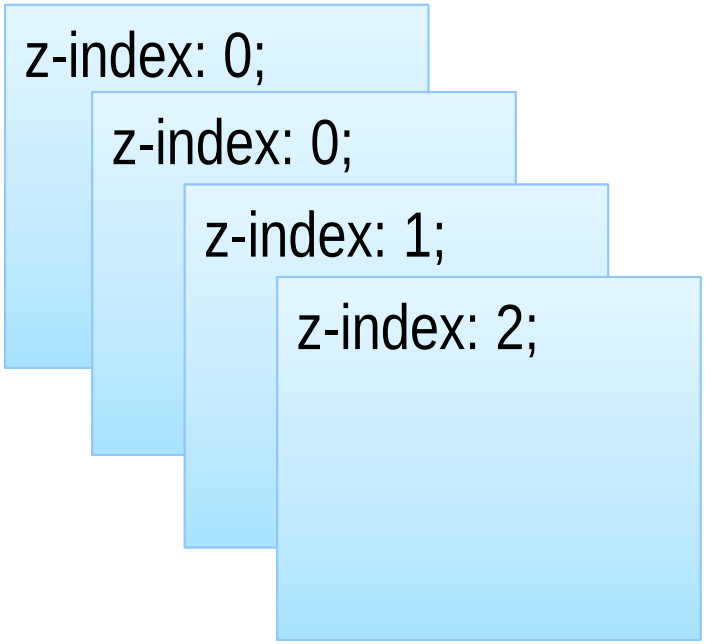


Relative to the page, scrolls with the webpage

Page Layout: z-index

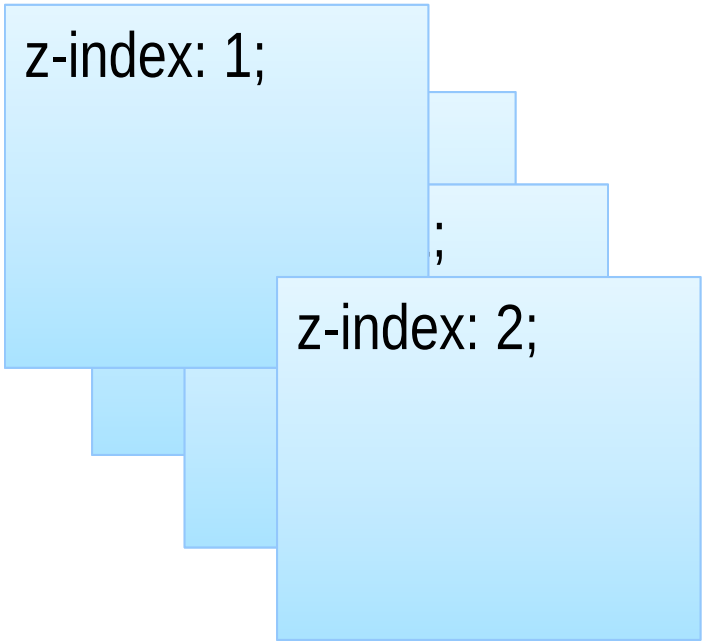
- z-index : **auto** | <number>;
  - Modifies the stacking order of the elements

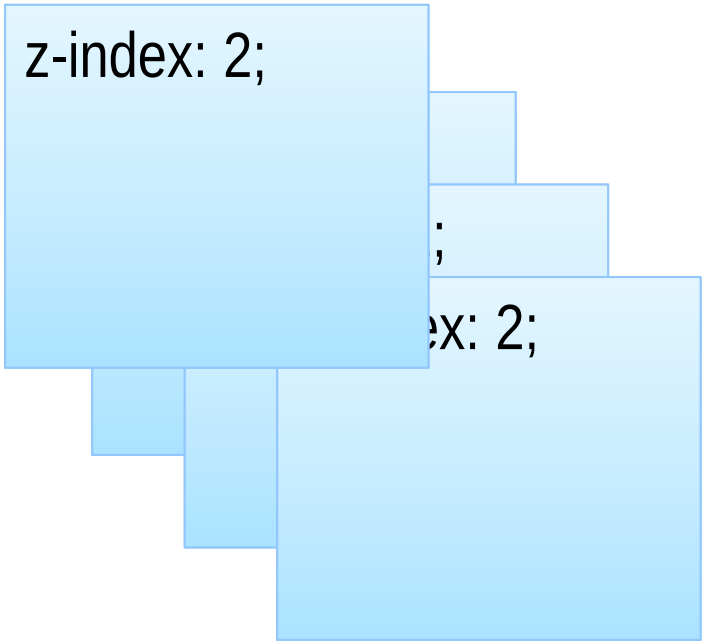




Page Layout: z-index

- Stacking order of elements with the same z-level value is based on the order in the HTML text





Page Layout: z-index (sample code)

```
<div style="
width:100px;height:100px;
border:1px solid #black;
background-color:skyblue;

position:absolute;
top:0px;left:0px;
z-index:1;">z-index auto</div>

<div style="
width:100px;height:100px;
border:1px solid #black;
background-color:lightblue;
opacity:0.5;
position:absolute;
top:20px;left:20px;
z-index:0;">z-index 0</div>
```

```
<div style="
width:100px;height:100px;
border:1px solid #black;
background-color:lightblue;
opacity:0.5;
position:absolute;
top:40px;left:40px;
z-index:1;">z-index 1</div>

<div style="
width:100px;height:100px;
border:1px solid #black;
background-color:lightblue;
opacity:0.5;
position:absolute;
top:60px;left:60px;
z-index:2;">z-index 2</div>
```

Contents

CSS

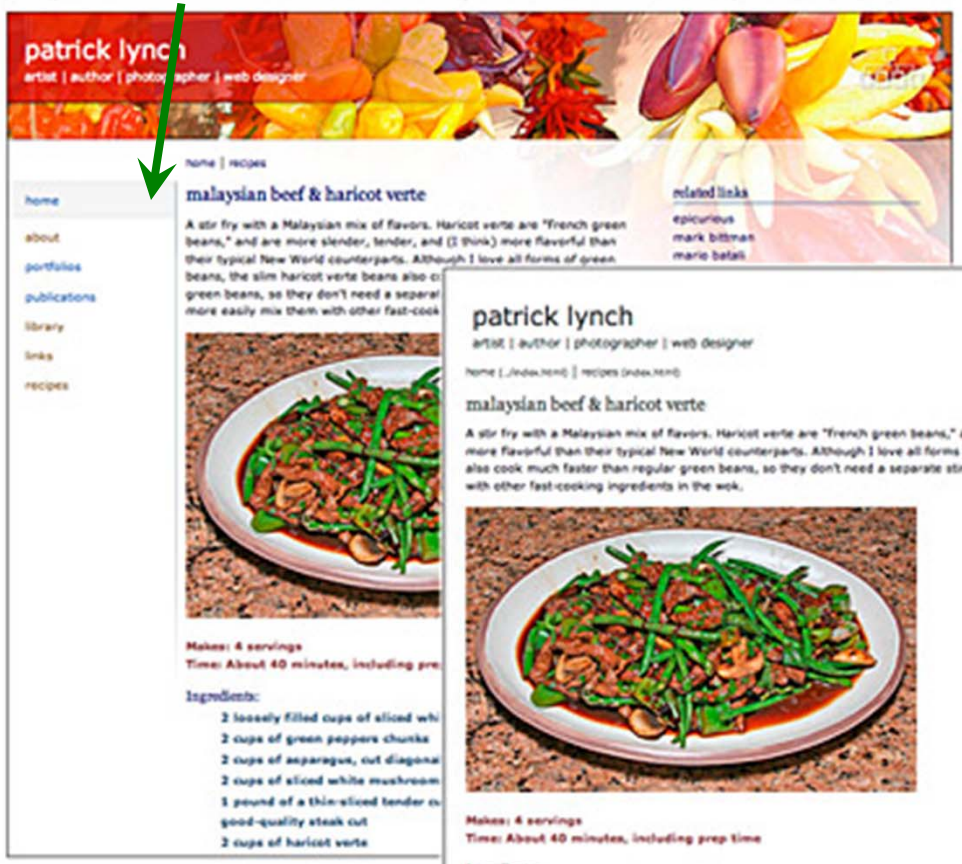
- What is CSS?
- Linking CSS to HTML
- CSS Selectors
- CSS Properties
  - Measurement
  - Colour
  - Typography
  - Box model
  - Page Layout
- Alternate Style

Designing for different devices

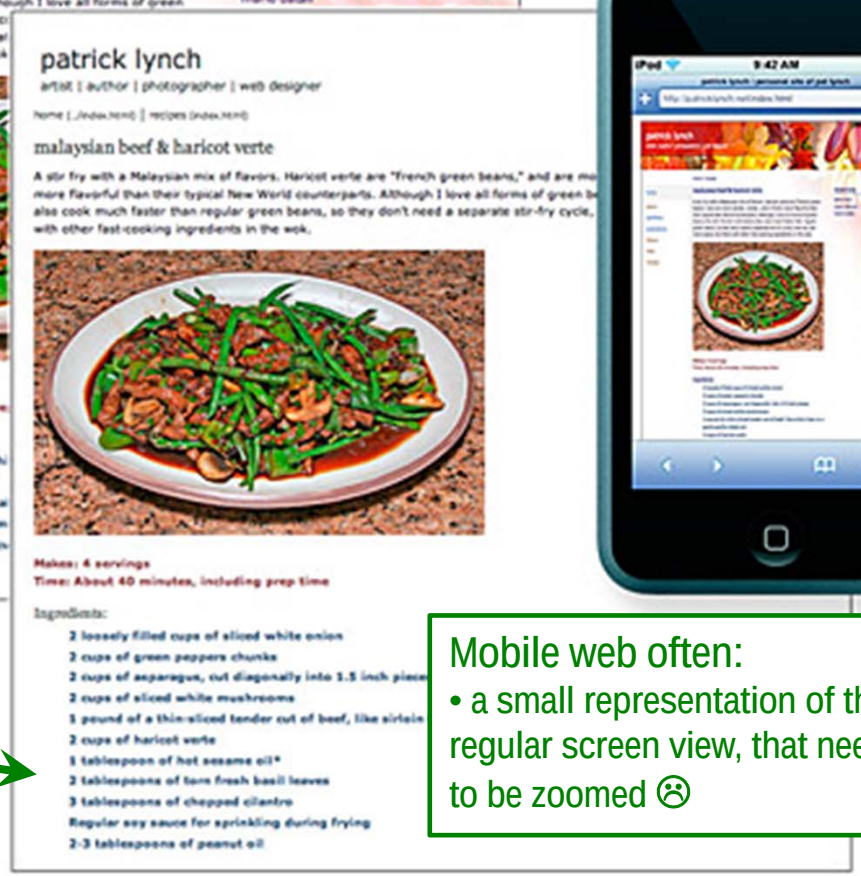
- Designing for mobile becoming increasingly important - “mobile first”
- “Responsive design” is also very important, eg. user changes orientation of a mobile device, changes screen resolution, window size.
  - Web Dev Toolbar | Resize | View Responsive Layouts
- Discussed in Mobile Apps development subjects…

CSS: Alternate Stylesheets

Page rendered with a screen style sheet



Page as printed with a print style sheet



Best Practice: use a different stylesheet, and set initial viewport ☺

Mobile web often:
- a small representation of the regular screen view, that needs to be zoomed ☹

or
- specifically styled menu pages that link to stripped-down pages. ☹

CSS: Alternate Stylesheets

- The idea of CSS is to be able to have *different* style sheets for *different* users and *different* devices.
- An easy way to offer this is by providing "alternate" style sheet links:

```
<link rel="stylesheet"
      href="normal.css" title="normal" />

<link rel="alternate stylesheet"
      href="bigfont.css" title="bigfont" />

<link rel="alternate stylesheet"
      href="aqua.css" title="aqua" />
```

- *The user can select* one of the "alternate stylesheet" options.
  (<=IE7 no, >=IE8 yes, Firefox yes, Chrome yes)

CSS: @media

- The **@media** selector is used *within a single style sheet*, to define style rules for multiple media types.

```
@media screen {
        body {
                font-family: sans-serif; font-size: 18pt;
        }
}

@media print {
        body {
                font-family: serif; font-size: 9pt;
        }
}

@media screen, print {
        body {
                line-height: 150%;
        }
}
```

http://www.w3schools.com/css/tryit.asp?filename=trycss_media

CSS: Responsive Layouts

Waves of new ideas and new techniques have evolved recently, particularly to address touch screens,
smart phones and tablets.

- ☐ *Fluid Layouts* *John Allsopp*
- ☐ *Progressive Enhancement*
- ☐ *Graceful Degradation*
- ☐ *Mobile First* *Luke Wroblewski* http://www.lukew.com/
- ☐ *Responsive design* *Ethan Marcotte* http://unstoppablerobotninja.com/
  - *- eg. Boston Globe, **break points**.*
  - *- Can now view responsive layouts in 'Web Developer' Add-on*
- ☐ *Pixels are flexible* *- now we can zoom*
- ☐ *Form Design* *- mobile strategy changes*
- ☐ *Gamification* *- acquire / engage / retain users*

As soon as these have been applied, users expect them !

Media Type and Queries

- CSS3 introduces Media Queries which is an expansion on the concept of media types in CSS2
  - Media type specify the different style rules
  - Media Queries creates more precise rules
- Both are for different types of destination media, such as screen~~, handheld~~, projection, tv, print, embossed, braille, speech, tty, and all.
-

```
<link href="mobile_device.css
      rel="stylesheet“
      type="text/css"
      media="screen and (max-width:480px)" >
```

design breakpoint

CSS: using the HTML `meta viewport`

- The HTML **`meta viewport`** is widely used to determine the initial "scale" that a web page will be presented in a browser.

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

- This is then used with the *media attribute* with *design breakpoints* to trigger the use of different stylesheets, in response to changes in "window size", "device orientation", "scale", and thus provide *"responsive web design"*

```
<link rel="stylesheet" media="(max-width:600px)"
href="small.css" type="text/css" />
<link rel="stylesheet" media="(min-width:601px)"
href="large.css" type="text/css" />
```

See also http://www.w3.org/TR/mwabp/#bp-viewport
https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/
http://css-tricks.com/snippets/html/responsive-meta-tag/
https://developer.mozilla.org/en/docs/Mozilla/Mobile/Viewport_meta_tag

Extra Notes

- Character sets
- Colour
- Digital images
- Image maps
- Multi-media
- Vector Graphics

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011
Creating Web Applications

What's Next?

- Client-side Scripting
- JavaScript

Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)