

CSS Essentials

Web Pages: Presentation



Evolution of HTML formatting

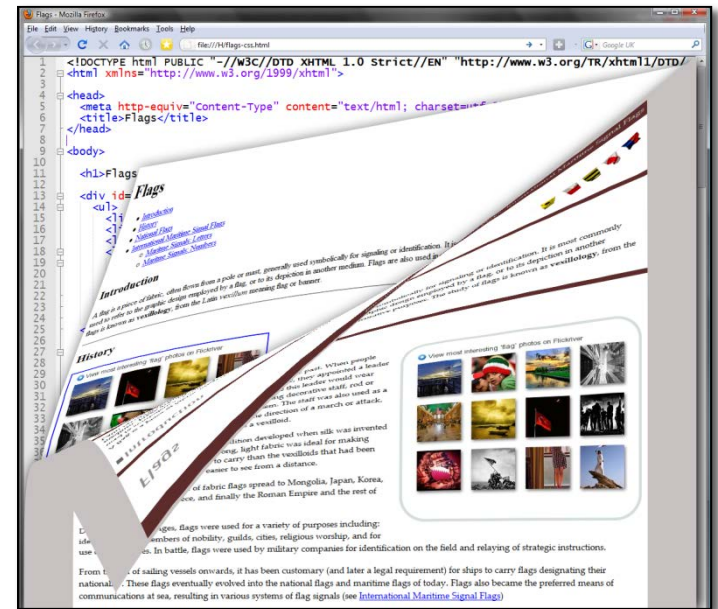
- (X)HTML only for structuring content
 - Specification only contains *guidelines* for visual browsers
- Some tags/attributes added for visual formatting

`Hello` → Hello

- This mixes style and structure
 - Often using proprietary mark up with limitations on what can be applied

The Solution: CSS

- Cascading Style Sheets
 - Separation of style from structure
 - Control – potentially over every item in the page
 - Easier style management
- Strict XHTML & Strict HTML 4.01 both *deprecated* HTML formatting in favour of CSS



Same content... different view

<http://www.csszengarden.com>



<http://www.mezzoblu.com/zengarden/alldesigns/>

Why style?

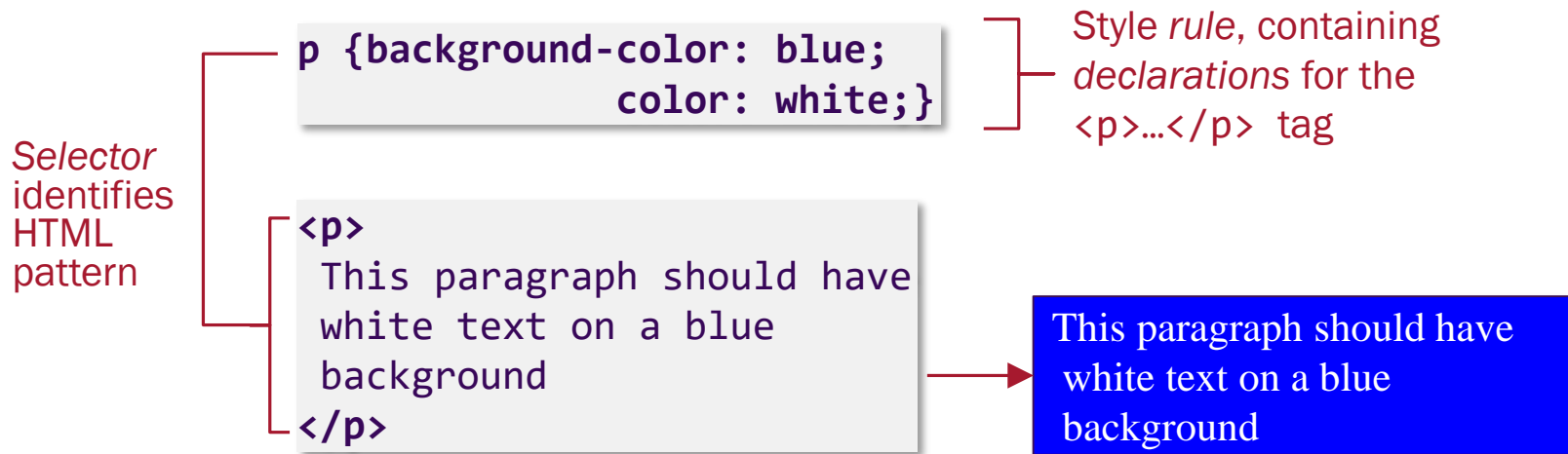
- Plain web pages are dull!
 - Additional meaning and aesthetics enhance (and influence) user experience
- An opportunity for creative expression
- Need to balance signal (information & purpose) with noise (distraction) where...

Absence of style == monotone signal

"Overstyled" == increased noise

CSS style sheets

- Style sheets specify formatting *rules*
- Rules consist of *selectors* and *declarations*



Basic style sheet syntax

Declaration(s) defined inside curly braces as style-property: value;

Selector → `p {background-color: blue;}`

Semi-colon ; separates declarations

`ul {margin-left: 15%; font-weight: bold;}`

Multiple selectors as comma separated list

"Apply declarations to h1 and h2 and h3 and h4"

`h1,h2,h3,h4 {background-color: white;
 color: blue;
 font-style: italic;}`

Internal style sheets

- Rules set out in <style> tags in the <head> section of the page

```
<html>
<head>
  <title>Internal Example</title>
  <style type="text/css">
    h1 {color: green; font-style: italic;}
  </style>
</head>
<body>
  <h1>Heading 1 in green italics</h1>
</body>
</html>
```

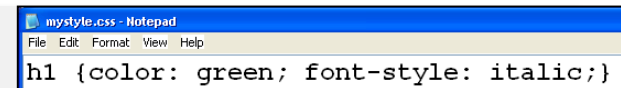


Heading 1 in green italics

External style sheets

- Style sheets are stored in separate files
 - Linked to current document
 - Multiple pages can reuse the same style sheet
 - Multiple style sheets can be linked to a single page

```
<html>
<head>
  <title>CSS example</title>
  <link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
<body>
  <h1>Heading 1 in green italics</h1>
</body>
</html>
```



Heading 1 in green italics

Using @import rules

- Alternative way to include external style sheets


```
<style type="text/css">  
  @import url("styles.css");  
</style>
```

- No difference in effect or behaviour, but can be more convenient
 - Only need one hard-coded <link> in XHTML document
 - Style sheets can be edited/attached/renamed without touching XHTML document

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

```
@import url("default.css");  
@import url("navbar.css");  
@import url("print.css");
```

Single linked style sheet
used to import actual styles
from separate files



Inline styles

- Style can also be added *inline*
 - Uses `style` attribute with CSS rule(s) as value

```
<p style="color:white; background-color: blue;">Hello</p>
```

Hello

- Try and avoid if possible – mixes style and structure back up
- Can be a useful option if needed to overcome a *specificity* issue

CSS selectors

Web Pages: Presentation

More on CSS selectors

- Three basic selector types define patterns to find in the mark-up

Tag – match all instances of the tag e.g. every `<p>...</p>`

Class – match tags containing this `class` attribute

Id – match the unique tag containing this `id` attribute

- Can be combined for more *specific* matches
- Additional syntax and operators allow precise control
- Combine with `<div>` and `` to build a *framework* for display

Classes as selectors

- Used to apply styles to specific sub-sets of HTML tags
 - Tags are grouped using a `class` attribute
 - Tags can be in more than one class

```
<h1 class="special">A heading</h1>
<p>This is a normal paragraph</p>
<p class="special">A different class of paragraph</p>
```

- Define style rule(s) in the style sheet

```
p {text-align: left; color: red;}
.special {text-transform: uppercase;}
p.special {text-align: right; color: green;}
h1.special {text-decoration: underline;}
```

Dot (.) in selector pattern indicates a class e.g.

`p.special`

matches

`<p class="special">`

ID as a selector


- Used to identify *unique* elements in the page
 - Uses an `id` attribute in the tag
 - Each `id` value can only be used *once* in any page (same `id` can be used on multiple pages though)

```
<p>The <span id="oneoff">Important</span> bit of...</p>
```

 `oneoff` now provides a unique id for a single element in this document

- Hash (#) in the CSS selector pattern indicates an `id`

```
#oneoff {font-style: italic; font-weight: bold;}
```

The *Important* bit of...

More selector syntax

Selector	Pattern matched
p	All <p>
.special	<anytag class="special">
p.special	All <p class="special">
#thisBox	The only <anytag id="thisBox">
#thisBox p	All <p> nested <i>anywhere</i> inside the only <anytag id="thisBox">
#thisBox > p	All <p> that are <i>direct children</i> of <anytag id="thisBox">
#thisBox p.special	All <p class="special"> nested <i>anywhere</i> inside the <i>only</i> tag with the id of thisBox
div#thisBox p	All <p> nested <i>anywhere</i> inside the <i>only</i> <div id="thisBox">

<http://www.w3.org/TR/CSS2/selector.html>

Even more selector syntax

Selector	Pattern matched
*	Any element
E F	Any F element that is a descendent of an E element
E > F	Any F element that is a (direct) child of an E element
E + F	Any F element immediately preceded by a sibling E element
E[foo]	Any E element with the attribute foo set
E[foo="value"]	Any E element whose attribute foo is set to "value"
E[lang~="en"]	Any E element whose attribute foo is a list of space separated values, one of them being equal to "value"

<http://www.w3.org/TR/CSS2/selector.html>

Combining selectors

CSS Rules

```
#section1 {color:red; text-align:center;}  
#section2 {color:blue;}  
.caps {text-transform:uppercase;}  
#section2 p {text-decoration:underline;}
```

HTML

```
<div id="section1">  
  <h1>Section one</h1>  
  <p class="caps">A paragraph in  
    section one</p>  
</div>  
<div id="section2">  
  <h1 class="caps">Section two</h1>  
  <p>A paragraph in section two</p>  
</div>
```

#section2 p styles only applied to
<p>...</p> nested inside #section2



Section one

A PARAGRAPH IN SECTION ONE

SECTION TWO

A paragraph in section two

Cascading style sheets

- All available styles for a page are combined as it loads
 - Final appearance for each element is composite of all appropriate rules
- Conflicting property values resolved by simple rules
 1. **Source:** User-specified styles (in the browser) are more specific
 2. **Specificity:** Relative weighting of selector priority
 3. **Order declared:** If specificity value are the same then "last one wins" (means inline styles are always more specific)
- Specificity – a measure of importance
 - The more *specific* the rule is... the greater priority its declarations have
 - Easy to calculate...

Specificity calculator

Count the number of ID, class and tag names in each selector

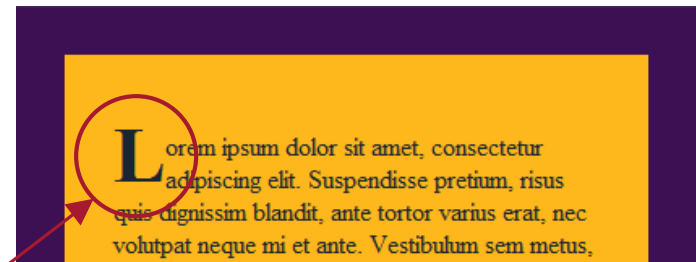
Selector	IDs	Classes	Tags
#thisbox	1	0	0
.special	0	1	0
p	0	0	1

Basic values of:
ID = 100
Class = 10
Tag = 1

Pseudo elements

Selectors for special parts of some elements

```
p.opening:first-letter {  
  font-size: 300%;  
  font-weight: bold;  
  float: left; }
```



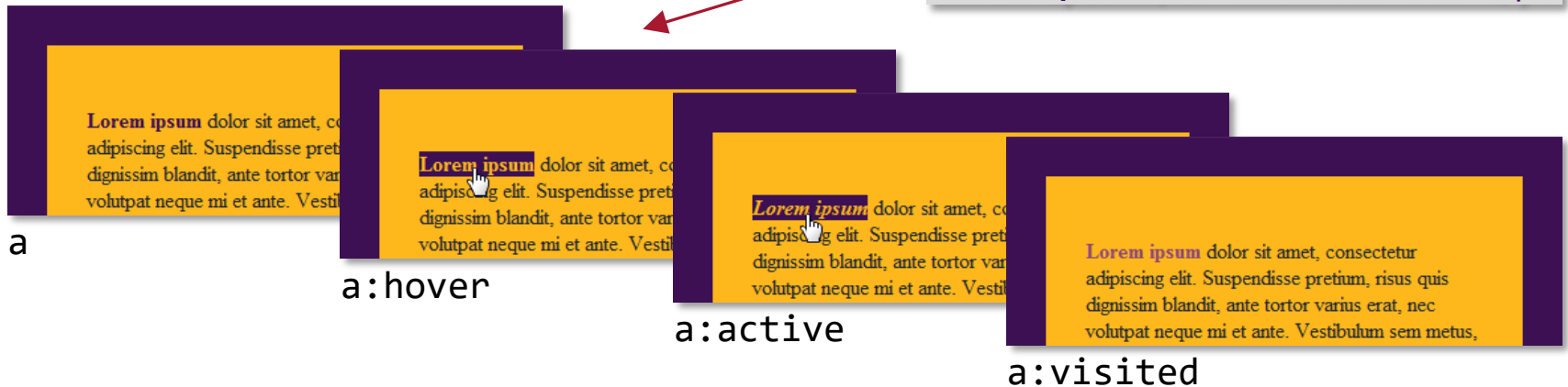
```
<p class="opening">Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Suspendisse pretium,  
risus quis dignissim blandit, ante tortor... etc  
</p>
```

Pseudo classes

Selectors for special status of some elements

```
a {text-decoration: none; font-weight: bold; color: #3c1053;}
a:hover {background-color: #3c1053; color: #fffb81c;}
a:active {font-style: italic;}
a:visited {color: #8c4779;}
```

```
<p>
<a href="http://lipsum.com">
  Lorem ipsum</a> dolor... etc </p>
```



CSS Properties and Values

Web Pages: Presentation

CSS Units

- CSS supports many types of measurement unit
- **Absolute** units calculated independently of other page content and/or browser defaults
 - Useful for precise layout
 - Include Pixels (px), Points (pt), Millimetres (mm)
- **Relative** units calculated proportionally against other page content or a browser default
 - e.g. currently available width, default text size etc.
 - Include Percentages (%), Ems (em), Exes (ex)
 - Also special relative units for text e.g. small, large, x-large ... etc.
- Good design uses a combination of both

CSS colours

- CSS allows rich control over color
 - Any colour can be specified using RGB or Hex (hexadecimal) codes
 - Only 17 *names* are actually valid in CSS2

[illegible]

```
color: red; ✓
```

```
color: magenta; x
```

```
color: rgb(0,32,234); ✓
```

color: #0000ff; ✓



<http://www.w3.org/MarkUp/Guide/Style.html>

<http://colorschemedesigner.com/>

CSS Text properties

- `color: red; color: rgb(127,0,0); color: #a00000; color: #a00`
- `text-align: "left";` (right, center, justify)
- `text-decoration: "underline";` (none, overline, line-through)
- `text-transformation: "uppercase";` (lowercase, capitalize)
- `text-indent: 2em;`
- letter-spacing, line-height, direction, word-spacing

CSS Font properties

- font-family: “Times New Roman”, Times, serif;
- font-style: normal; (italic, oblique)
- font-size: 24px; font-size: 1.5em; font-size:150%;
- font-weight: bold; (normal)
- font-variant: small-caps;

https://www.w3schools.com/cssref/css_websafe_fonts.asp

CSS List properties

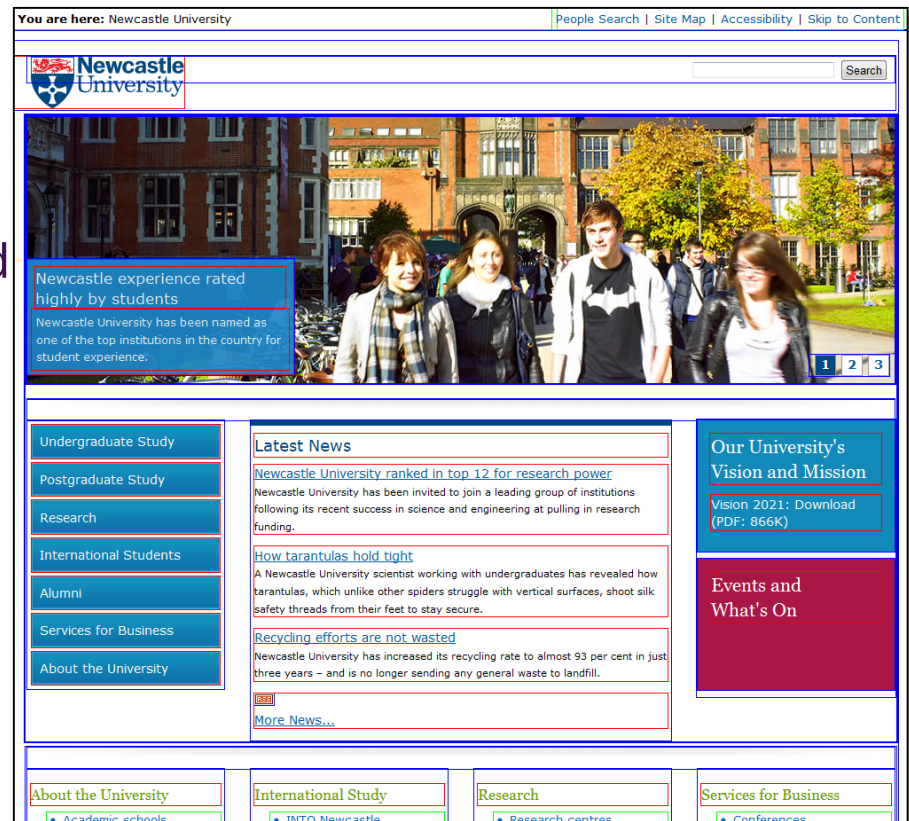
- `list-style-type: circle;` (square, upper-roman, lower-alpha)
- `list-style-image: url('...');`
- `list-style-position: outside;` (inside)

The CSS box model

Web Pages: Presentation

The CSS box model

- Fundamental to CSS layout
 - Every page element represented as a box
 - Box properties not inherited from parent boxes
- Box properties can apply to whole box or individually to any of the 4 sides
 - Shorthand declarations make this easy
 - Depends on property



Border

- The outline of a box, made up of three sub-properties

Width – a unit of measurement e.g.

```
border-width: 20px;
```

Style – value from a preset list e.g.

```
border-style: solid|dashed|...
```

Color – a valid colour value e.g.

```
border-color: blue;
```

- Simple shorthand to set quickly

```
p {border: 20px solid green;}
```

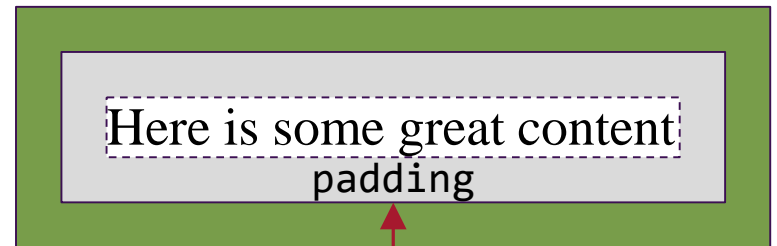
```
<p>Here is some great content</p>
```



Padding

- The space between the *outside* edge of the content and the *inside* edge of its box
 - i.e. excluding any borders
- Creates space *inside* boxes
 - Leave empty (negative space)
 - Use background images to fill

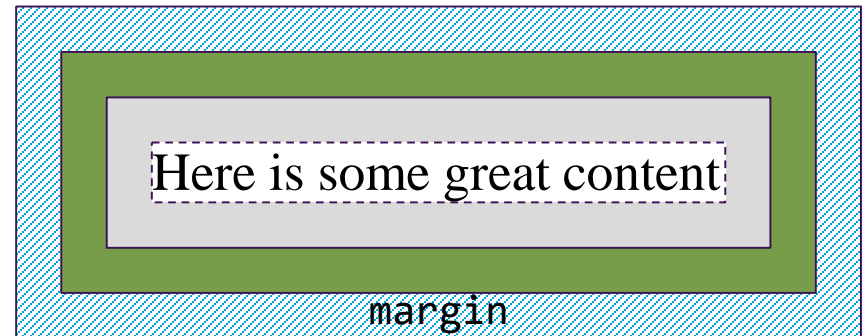
```
p {border: 20px solid green;  
  padding: 20px;  
}
```



`<p>Here is some great content</p>`

Margin

- The space around the outside edge of a box – including borders
- Creates space *between* boxes
 - Useful for spacing content



```
p {border: 20px solid green;  
  padding: 20px;  
  margin: 20px;  
}
```

`<p>Here is some great content</p>`

Shorthand declarations

- Useful for setting box properties quickly and/or precisely
 - Use wherever possible to reduce the amount of code needed to style a page

Declaration	Result
<code>border-width: 20px;</code>	Width 20px on all 4 sides
<code>border-left-width: 20px;</code>	Width 20px on left-hand side only
<code>margin: 20px 40px;</code>	Margin 20px top/bottom, 40px left/right
<code>padding: 20px 40px 10px 30px;</code>	Padding set for top/right/bottom/left
<code>border: 20px solid blue;</code>	20px, solid, blue border on all 4 sides

Width and Height

- By default, browser uses the max width available for each box
- Box property `width` used to impose a defined value for *content* width



```
#box1 {width: 200px;}
```

Total width (i.e. as drawn) = width (+ 2x padding + 2x border)

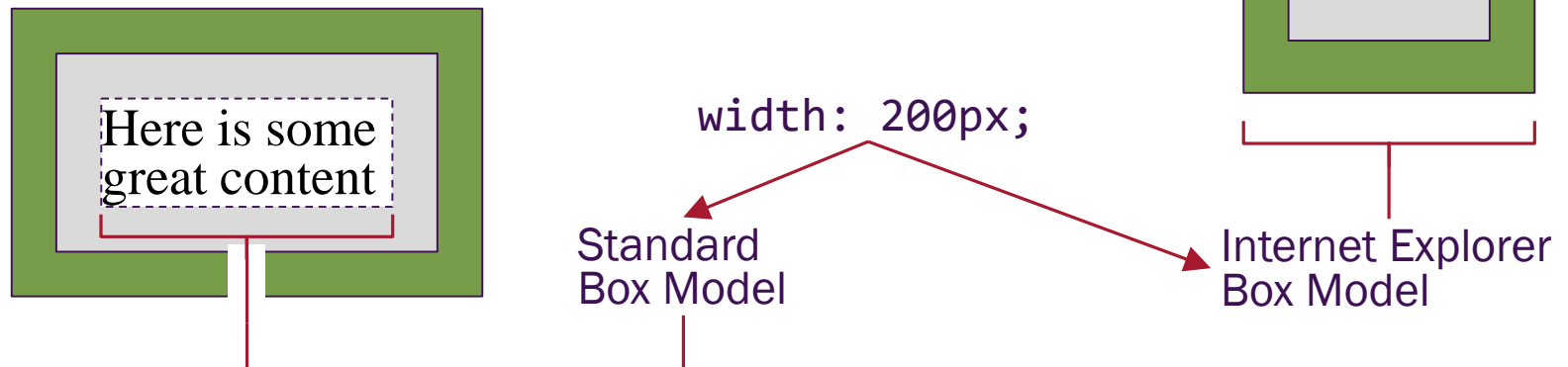
- Modern browsers also support more flexible min/max

```
#box1 {min-width: 200px; max-width: 800px;}
```

- Height can be set in the same way

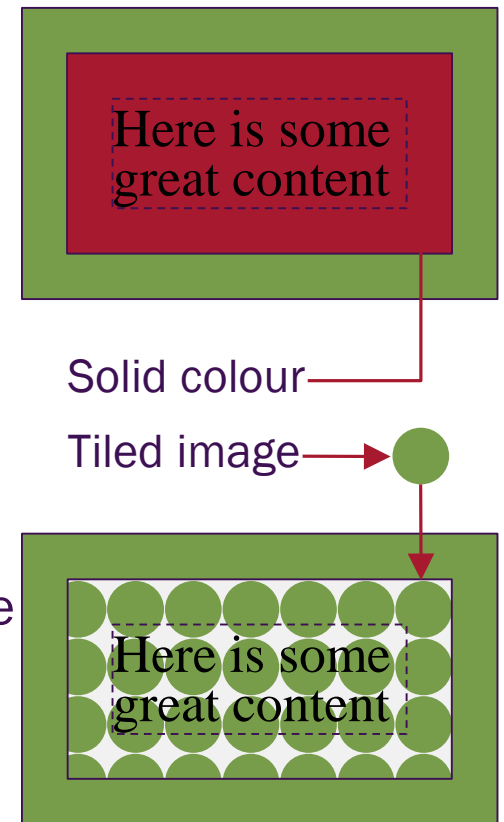
The Internet Explorer box model

- IE Box Model uses CSS width (and height) for *total* box dimensions
 - Common cause of cross-browser quirks in layouts
- IE6+ now uses standard model if DTD present in file
 - Fix is simply to write standards compliant code!



Background

- Background of all the visible space *inside* any borders i.e. (content + padding)
- Made up of several sub-properties
 - Color** – a valid colour value (or transparent)
`background-color: red;`
 - Image** – Use an image file as box background
`background-image: url(/images/bg1.png);`
 - Position** – Control placement for initial image tile
`background-position: top right;`
 - Repeat** – Control direction of image tiling
`background-repeat: repeat-x|y|no-repeat;`



Using background images

- Form the basis of most modern web designs
- Easy to work with once you have grasped the basic box model (and got the graphics!)
- Two important things to remember...
 1. CSS2 only allows one image per box (CSS3 allows multiple images 😊)
 2. Images delivered by CSS are a decorative part of the design **not** the content
 - Image *content* e.g. photos, diagrams etc. should be part of the XHTML (via the `` tag)



Placing a background image

```
background-image: url(bg1.png);  
background-position: bottom right;  
background-repeat: no-repeat;
```

Inserts a single instance of bg1.png

Create space for the image using padding – set to at least the image width

```
padding-right: 200px;
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla tincidunt rhoncus mauris, at tempor neque iaculis a. Curabitur dignissim sapien quis urna luctus placerat. Donec blandit interdum elit, non convallis tortor porttitor sagittis. Sed luctus sagittis nibh, sed scelerisque nulla rhoncus nec. In suscipit commodo felis vel adipiscing. Suspendisse ante est, vestibulum in sollicitudin et, imperdiet sed arcu. Curabitur quis quam tortor. Nam ultrices congue lorem in vehicula. In varius ultrices lacus, quis euismod sapien ultrices vel.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla tincidunt rhoncus mauris, at tempor neque iaculis a. Curabitur dignissim sapien quis urna luctus placerat. Donec blandit interdum elit, non convallis tortor porttitor sagittis. Sed luctus sagittis nibh, sed scelerisque nulla rhoncus nec. In suscipit commodo felis vel adipiscing. Suspendisse ante est, vestibulum in sollicitudin et, imperdiet sed arcu. Curabitur quis quam tortor. Nam ultrices congue lorem in vehicula. In varius ultrices lacus, quis euismod sapien ultrices vel.

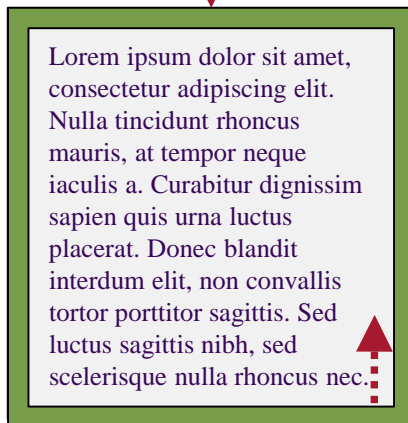
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla tincidunt rhoncus mauris, at tempor neque iaculis a. Curabitur dignissim sapien quis urna luctus placerat. Donec blandit interdum elit, non convallis tortor porttitor sagittis. Sed luctus sagittis nibh, sed scelerisque nulla rhoncus nec. In suscipit commodo felis vel adipiscing. Suspendisse ante est, vestibulum in sollicitudin et, imperdiet sed arcu. Curabitur quis quam tortor. Nam ultrices congue lorem in vehicula. In varius ultrices lacus, quis euismod sapien ultrices vel



Content > Box Size = Overflow

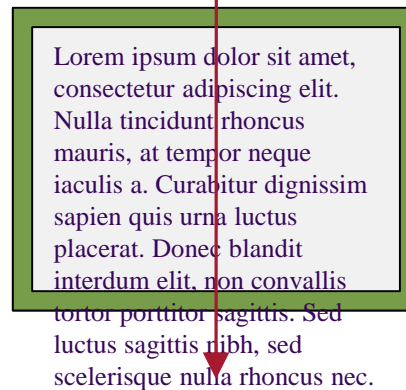
- Use box's overflow property to control

Shrinking a box
creates an *overflow*



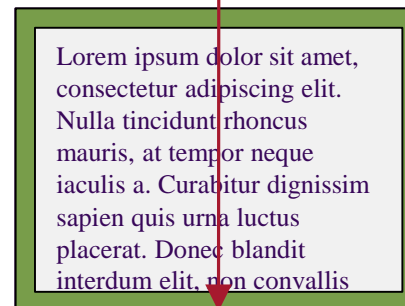
Allow overflow to spill
out of box (default):

`overflow:visible;`



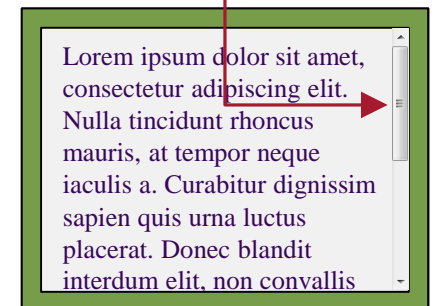
Overflow can be
hidden (clipped) with:

`overflow:hidden;`



Add scrollbar to see
clipped content with:

`overflow:scroll;`
or
`overflow:auto;`



Display and Visibility

- CSS `display` property controls the *display type* of an element
 - Can be used to override HTML default or even remove items from the styled page flow

```
display: block;  
display: inline;  
display: none;
```

- CSS `visibility` does exactly what it says on the tin!
 - Show/hide elements whilst leaving them in the page flow

```
visibility: hidden;  
visibility: visible;  
visibility: collapse; (tables only)
```

Display and Visibility

```
p {display: inline;}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse pretium, risus quis dignissim blandit, ante tortor varius erat, nec volutpat neque mi et ante. Vestibulum sem metus, volutpat in convallis et, fringilla nec quam. Integer gravida, ligula eu fermentum lacinia, dui arcu lacinia diam, interdum scelerisque enim dui sit amet dui. Nulla ornare, purus sit amet accumsan ullamcorper, erat lectus venenatis eros, ultricies scelerisque dui purus nec diam. **Lorem ipsum** dolor sit amet, consectetur adipiscing elit. Nam volutpat egestas nunc in fringilla. Cras condimentum pulvinar blandit. Aenean hendrerit rhoncus dui, at dignissim risus euismod ac. Donec eu mollis nunc. Donec non neque felis, ut gravida odio. Mauris a odio leo, blandit adipiscing leo. Donec vel pellentesque odio.

```
p.opening {  
  display: none;}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam volutpat egestas nunc in fringilla. Cras condimentum pulvinar blandit. Aenean hendrerit rhoncus dui, at dignissim risus euismod ac. Donec eu mollis nunc. Donec non neque felis, ut gravida odio. Mauris a odio leo, blandit adipiscing leo. Donec vel pellentesque odio.

```
p.opening {  
  visibility: hidden;}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam volutpat egestas nunc in fringilla. Cras condimentum pulvinar blandit. Aenean hendrerit rhoncus dui, at dignissim risus euismod ac. Donec eu mollis nunc. Donec non neque felis, ut gravida odio. Mauris a odio leo, blandit adipiscing leo. Donec vel pellentesque odio.

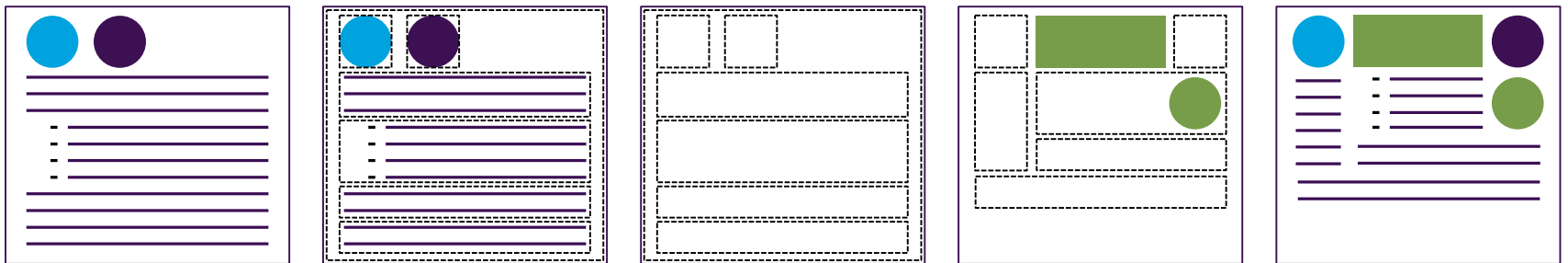
Positioning possibilities

- Once content can be viewed as "just boxes", CSS positioning and layout simply involves moving, placing or rearranging them

Structured content provides the basis for box model framework

Consider page as just the boxes

Manipulate boxes (position, background, padding etc.) to achieve final layout



- Specific box properties control how positioning takes place
 - position, float, top, left, etc...
 - These are considered in more detail in *CSS Positioning*

CSS Positioning

Web Pages: Presentation



First... clean (X)HTML

- Good CSS layout relies on good (X)HTML
- Positioning is easier to manage in a well structured document
 - Important to know which elements are contained within which others
 - Good use of <div>, , class and id to create additional framework
- Choosing HTML or XHTML doesn't matter
- Choosing *strict* HTML or XHTML *is* important
 - Avoids temptation to use deprecated tags/attributes
 - Encourages the use of CSS instead

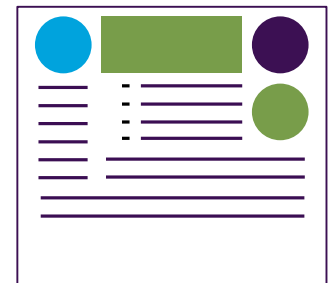
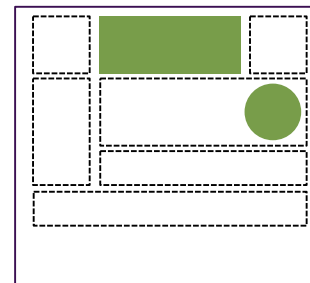
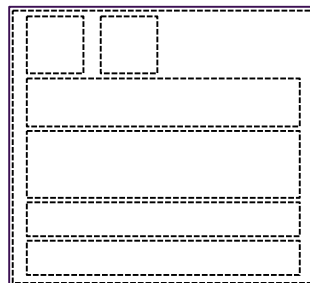
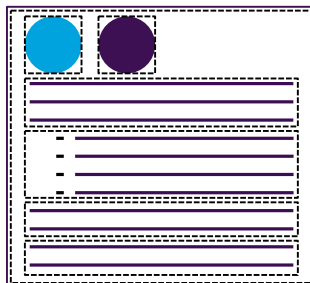
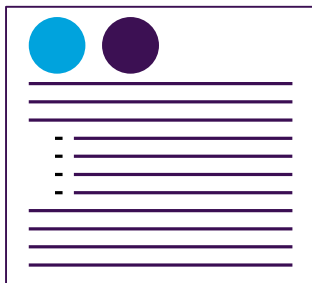
Second... the CSS box model

- The first step to mastering CSS positioning and layout is to understand the CSS Box Model
 - Easier with well structured XHTML
- Once content is viewed as "just boxes", CSS positioning and layout simply involves moving, placing or rearranging them

Structured content provides the basis for box model framework

Consider page as just the boxes

Manipulate boxes (position, background, padding etc.) to achieve final layout

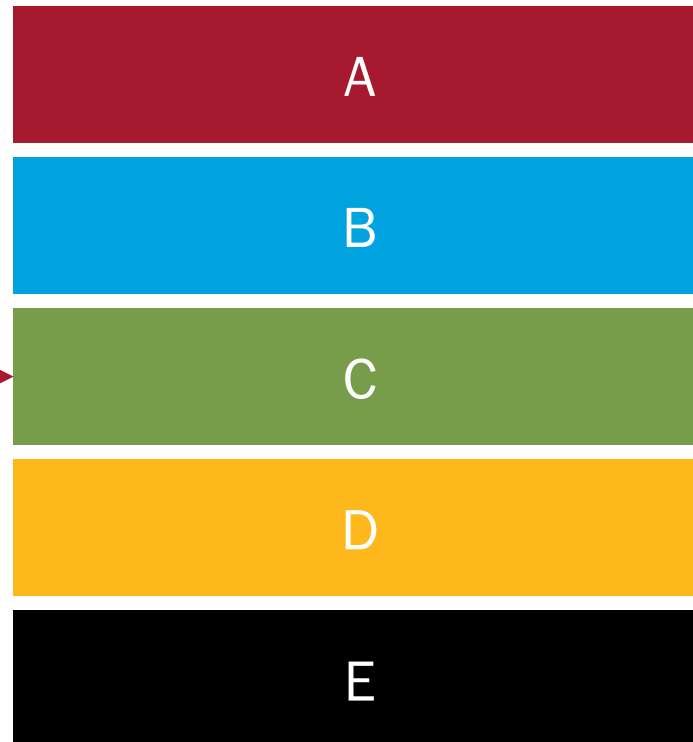


Next... check natural page flow

- The *natural page flow* of a document is the *source order* display of the XHTML element boxes it contains
- The final rendered position of each block of content determines the starting point for the following ones
- With no CSS applied this may not look pretty – but should make logical sense to help non-visual browsers
 - e.g. screen reading text-to speech browsers

Natural page flow

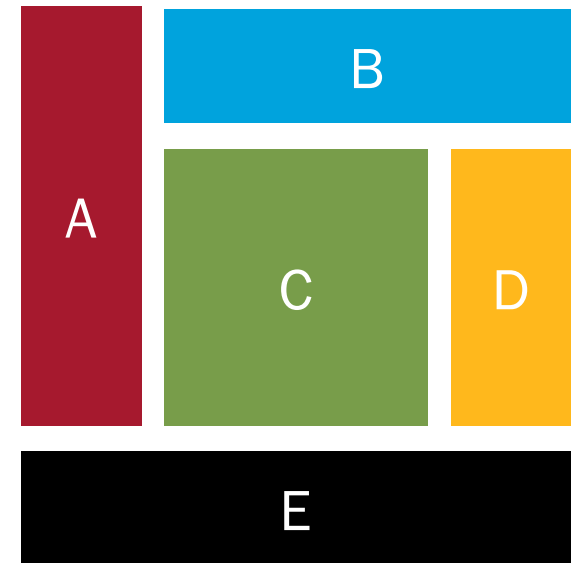
```
<html>
<head>etc.</head>
<body>
<div id="A">1</div>
<div id="B">2</div>
<div id="C">3</div>
<div id="D">4</div>
<div id="E">5</div>
</body>
</html>
```



Finally...apply CSS and enjoy

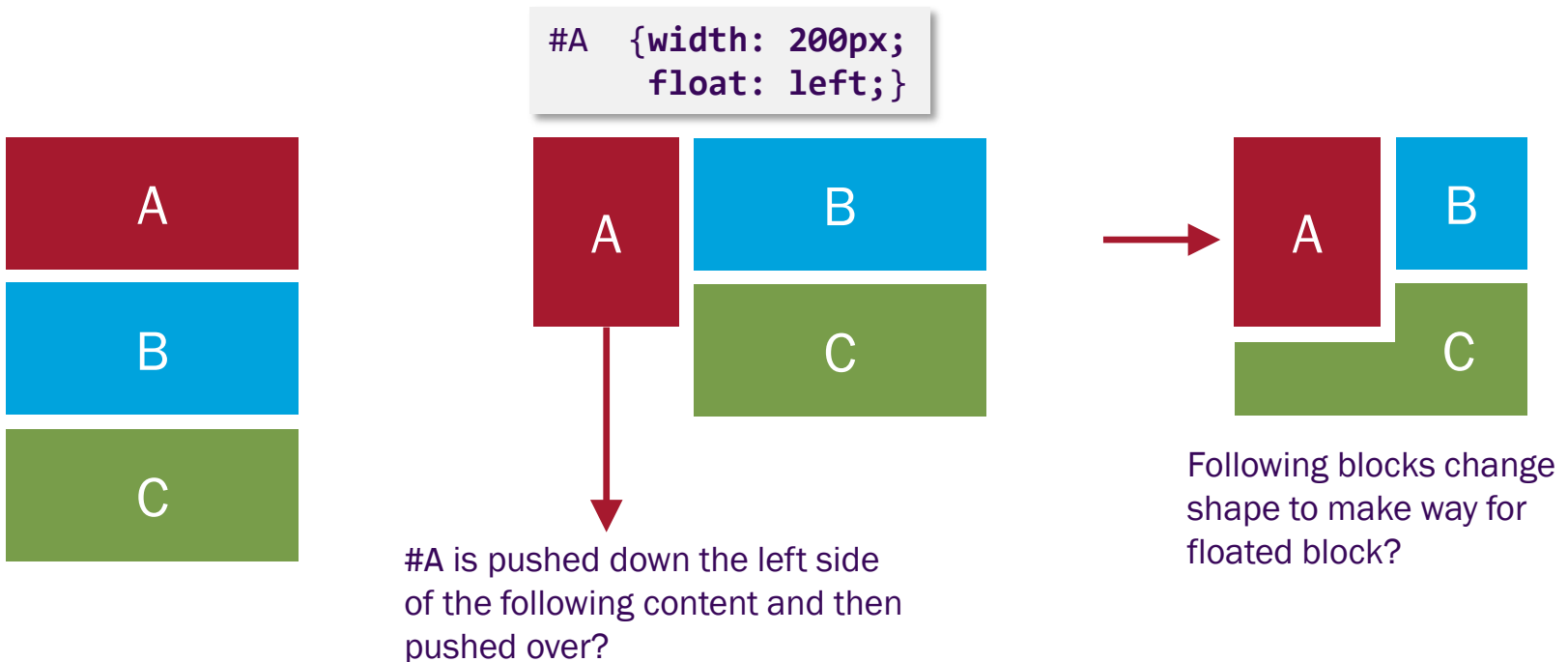


- Browser parses (reads) the XHTML and the CSS styles which apply to each box *before* drawing the page
- The final appearance is determined by combination of CSS properties and (if not explicitly set in CSS) browser defaults
- CSS can be used to re-position content...
 - within the page flow i.e. offset from natural location
 - contrary to the page flow i.e. removed from natural flow and placed elsewhere

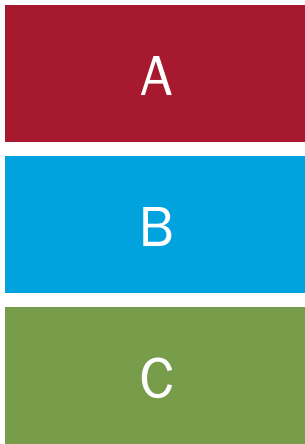


Floats

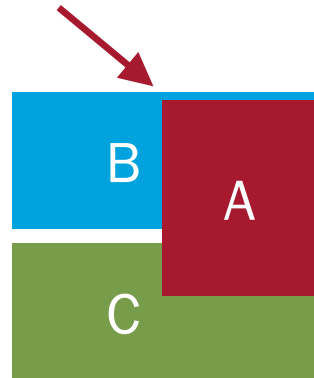
- Potentially very neat way to position content, **but** avoid this important misconception...



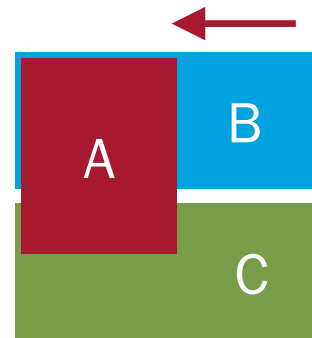
Floats: What really happens



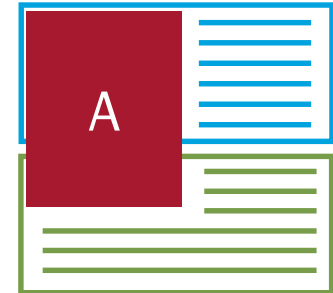
```
#A {width: 200px;  
    float: left;}
```



#A positioned over the following content as far over as possible *in the opposite direction to the float*



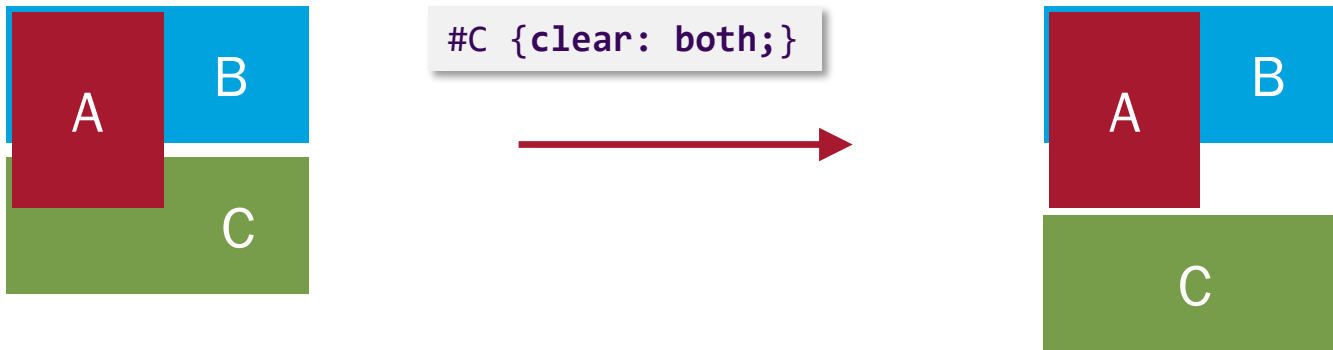
#A is then pushed back across as far as possible *in the direction of the float*.



The following blocks *do not* change shape! Content in them is allowed to wrap around the float

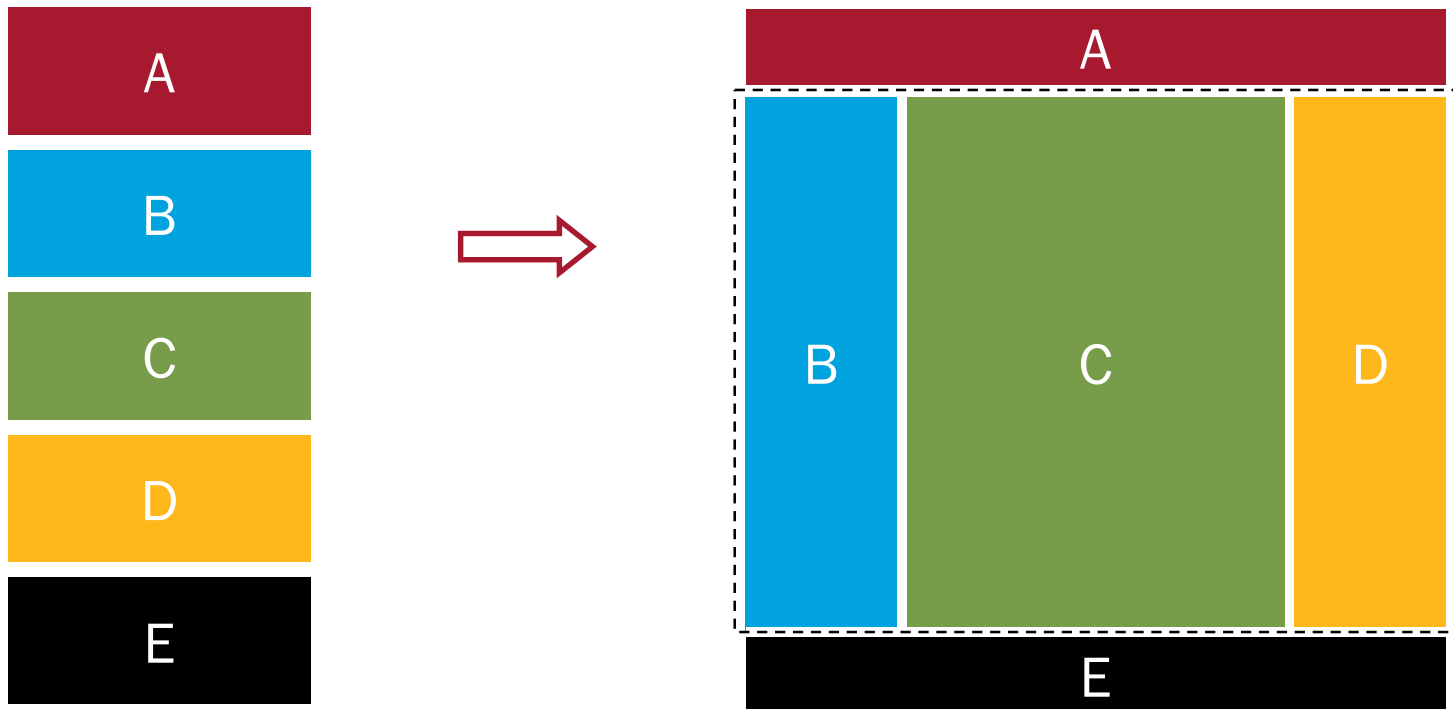
Clearing floats

- The CSS `clear` property can be used to stop the float covering later content.
- Values `left`, `right` or `both` determine which side of the box should be "clear" of any floated elements



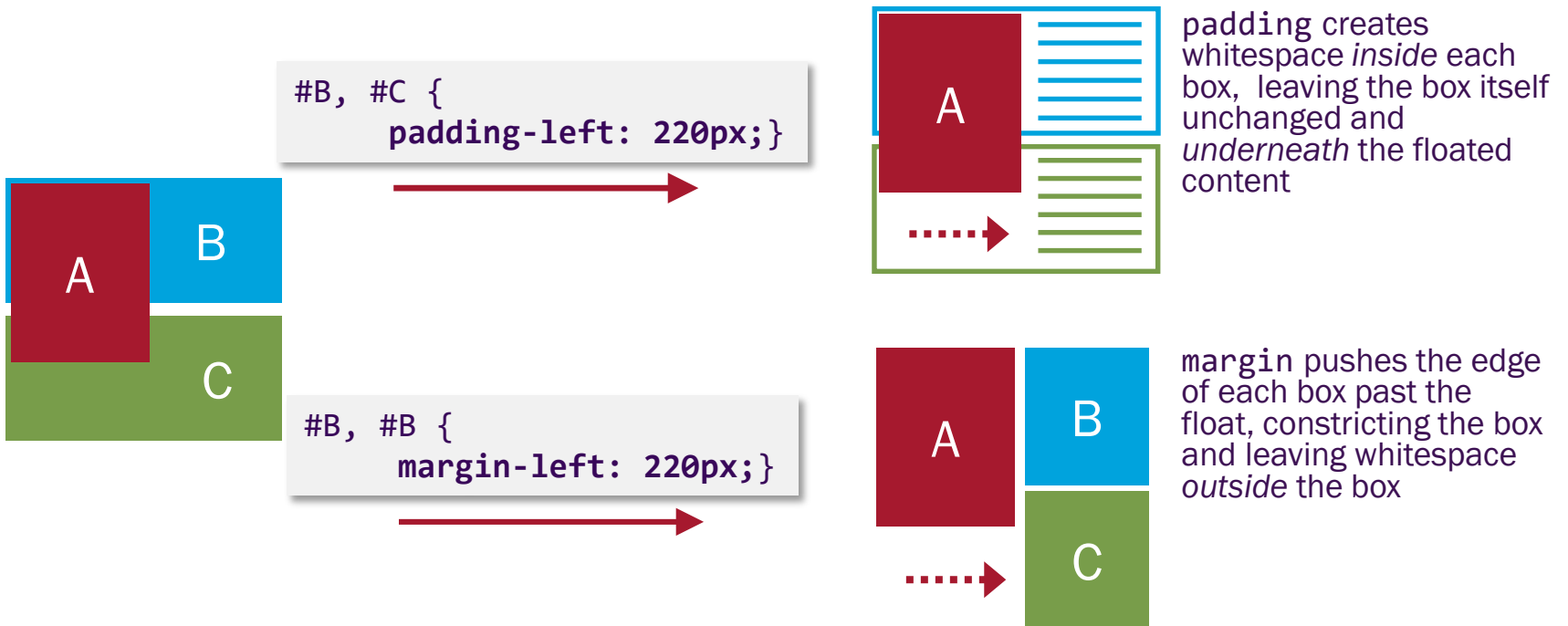
Columns

- CSS columns are (relatively) easy with floats



Floats and columns

- The non-floated boxes can be turned into columns using margin or padding



CSS position property

Four values explicitly re-position boxes using CSS

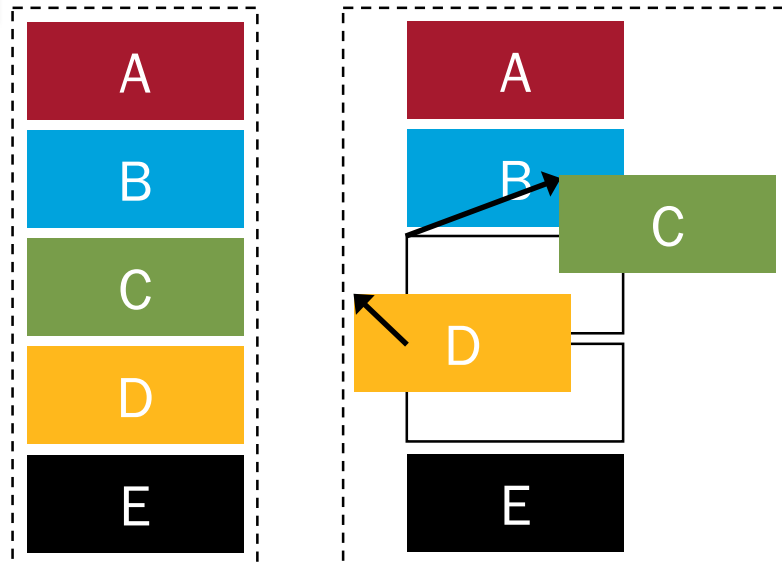
Value	Description
static	Default non-positioned value. Not normally set unless to specify an override of other positioning
relative	Leaves element in the page flow, but allows it to be displayed in an offset position
absolute	Removes element from page flow and allows it to be positioned anywhere
fixed	Removed element from page flow and fixes it to the browser viewport. Rest of page can now scroll behind it

Relative positioning

- Box is initially positioned according to natural flow
 - Offset is specified using "*from the...*" properties

```
#thisBox {position: relative;  
          top: 25px;  
          left: 100px; }
```

- Following XHTML retains original position

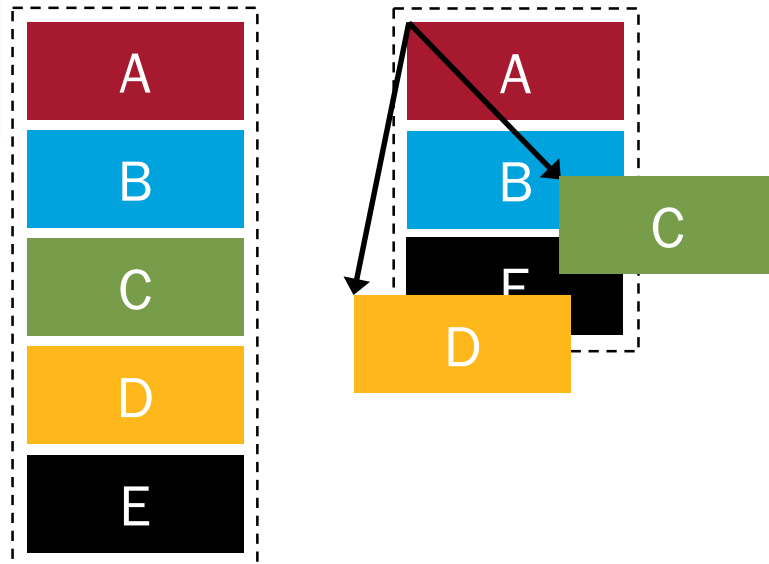


Absolute positioning

- Box *removed* from natural flow
 - New position is specified using "*from the...*" properties
 - Measured from *nearest positioned parent* container (default is <body>)

```
#thisBox {position: absolute;  
          top: 25px;  
          left: 100px; }
```

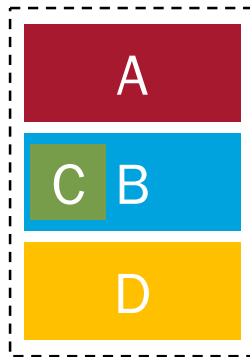
- Following XHTML behaves as if positioned block never existed!



Positioned parents

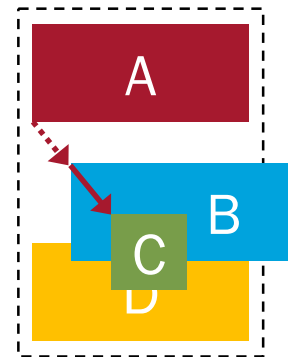
- An element acting as an origin point for absolute positioning must itself *have position*
 - It does not have to have *moved* though 😊
- If no containers *with position* found, browser uses <body>

```
<body>
  <div id="A"></div>
  <div id="B">
    <div id="C"></div>
  </div>
  <div id="D"></div>
</body>
```



```
#B {position: relative;
    top: 50px;
    left: 100px;}

#C {position: absolute;
    top: 20px;
    left: 50px;}
```



Fixed positioning

- Box *removed* from natural flow and fixed to the browser *viewport*
 - New position is specified relative to the viewport... following content scrolls underneath

```
#thisBox {position: fixed;  
top: 0px;  
left:0px; }
```



Stacking order

- Only works on content *with position*
 - Uses z-index property



```
#ThisBox {position:absolute;  
top: 25px;  
left: 100px;  
z-index: 1; }
```

Precision layouts

You can choose to work with or against the browser defaults

- For ultimate precision some designers use a "reset style sheet"
 - Loaded first, typically resets/strips out all default style

JISCnetskills

JISC Netskills, University of Newcastle, Newcastle upon Tyne, NE1 7RU
Tel: +44(0)191 222 5000, Email: enquiries@netskills.ac.uk

[JISC Netskills Training On-Site Delivery Workshop Portfolio](#)

JISC Netskills Training

JISC Netskills is a training and staff development service with 14 dedicated and across the UK. JISC Netskills is partly funded by the Joint Information Systems

Services are provided to a wide range of clients including all UK universities and

```
body,div,ul,h1,h2,h3,p {  
    margin: 0;  
    padding: 0;  
    border: 0;  
    font-size: 100%;}
```



JISCnetskills

JISC Netskills, University of Newcastle, Newcastle upon Tyne, NE1 7RU
Tel: +44(0)191 222 5000, Email: enquiries@netskills.ac.uk

[JISC Netskills Training On-Site Delivery Workshop Portfolio](#)

JISC Netskills Training

JISC Netskills is a training and staff development service with 14 dedicated and across the UK. JISC Netskills is partly funded by the Joint Information Systems O Services are provided to a wide range of clients including all UK universities and and non-government organisations.

Public Workshop Programme

We run a regular programme of training events at a range of venues throughout the They normally include:

- Presentations, demonstrations and discussion
- Extensive practical hands-on exercises
- Opportunities for questions and discussion

Media specific styles

Web Pages: Presentation

CSS: Media types

- CSS spec identifies a range of media types to which specific styles can be applied

Media	Intended device
all	Apply to all outputs (default)
screen	Computer screens
print	Printed media (and Print Preview)
tv	TV like devices (low res, limited scrolling)
projection	Data projectors
handheld	Small devices (small screens, ltd bandwidth)
braille	Braille tactile feedback devices
embossed	Braille printers

CSS: Using media specific styles

- Specify via media attribute in <link />

```
<link rel="stylesheet" type="text/css" href="printer.css" media="print" />
```

- Styles only added to page when media "invoked"
 - Effect is cumulative (i.e. specificity/inheritance important!)
- Use with other media types for better control

```
<link rel="stylesheet" type="text/css" href="core.css" media="all" />  
<link rel="stylesheet" type="text/css" href="layout.css" media="screen" />  
<link rel="stylesheet" type="text/css" href="printer.css" media="print" />
```


CSS: Using media specific styles

- Specify inside style sheets using @media rules

```
@media print {  
  body {background-color: #ffffff; color:#000000;}  
  a {text-decoration: none; font-weight: normal; color: #000000;}  
  #navbar {display: none;}  
  h2 {page-break-before:always;}  
}
```

CSS: Browser compatibility and Validation

Web Pages: Presentation

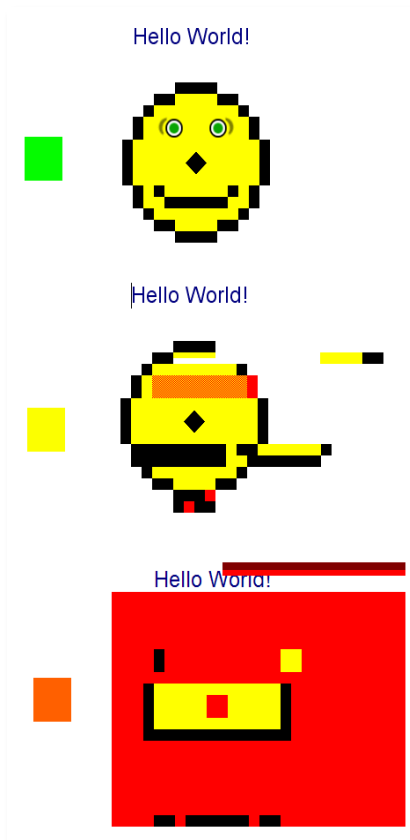
CSS: Browser support

- Often seen as a big issue – mostly historical
 - Worst (mainstream) offenders Netscape 4 (ignore nowadays) & IE 4-6
- IE still seen by many as the "big problem" but...
 - IE 4 -5.5 can safely be ignored
 - IE7 & IE 8 much better
 - Main issue is slow (corporate) migration from IE 6
- Standards compliance is important for less reliance on "hacks"
- All browsers have some "quirks" – is perfection possible?

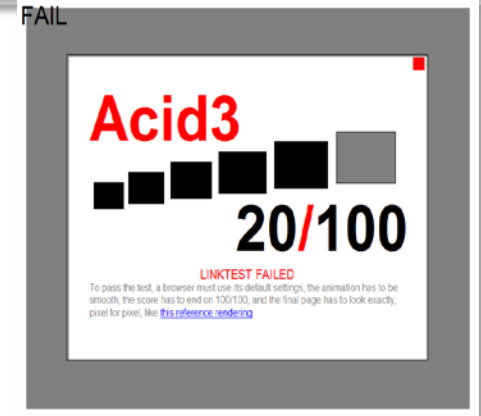
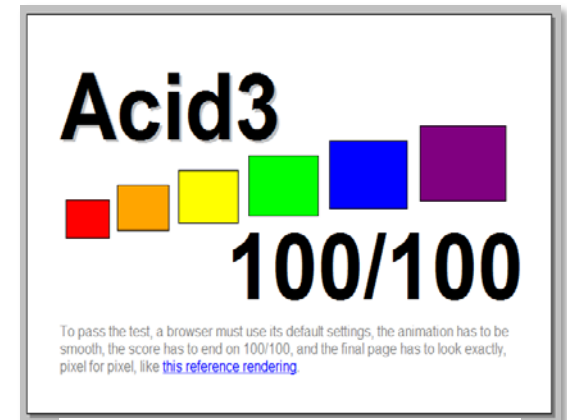
CSS: The ACID test

<http://acid2.acidtests.org/>

<http://acid3.acidtests.org/>



<http://www.webstandards.org/>



CSS: Browser support

- Tools to help
 - !Doctype switching
 - Compatibility tables
 - Clean code and good design practice!

Selector	IE 5.5	IE 6	IE 7	IE 8 as IE 7	IE 8 as IE 8	FF 2	FF 3.0	FF 3.1b	Saf 3.0 Win	Saf 3.1 Win	Saf 4.0b Win	Chrome 1	Chrome 2	Opera 9.62	Opera 10a	Konqueror 3.5.7
:before and :after To generate content before and after an element		no		yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes		yes
:hover and :active An element in a hovered (mouseover) or active (mousedown) state.		minimal	incomplete	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes		almost
<ul style="list-style-type: none">• IE 5/6 supports both only on links. IE 7 supports <code>:hover</code>, but not <code>:active</code>, on all elements.• Konqueror removes the <code>:active</code> styles almost immediately.																

<http://www.quirksmode.org/css/contents.html>

- er... test for yourself?

Dealing with IE

- Policies
 - Decide not to cater for IE differences?
 - Hacks – CSS tricks to hide/show specific rules to IE?
 - Conditional comments – regular HTML comments with IE specific syntax (not just CSS!)

```
<!--[if IE 6]>  
<style type="text/css">  
  p {property: value for IE6 only;}  
</style>  
<![endif]-->
```

<http://www.quirksmode.org/css/condcom.html>

CSS: Validation

- CSS is error tolerant
 - Process all complete, valid rules, ignore "incorrect" rules
- W3C spec for CSS 1 & CSS 2.1 currently widely adopted (CSS3 creeping out)
- Validation service exists – use it!

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

Sorry! We found the following errors (1)

URI : <file://localhost/TextArea>

3 #national

Property background-position doesn't exist : bottom right

Valid CSS information

```
#national {  
  background-image : url("3-flags75.png");  
  background-repeat : no-repeat;  
}
```

CSS3 you can use now

...BUT NOT IN ASSIGNMENT 1 !!!

Fonts

- Use the `@font-face` rule to define a font to be used in a document

```
@font-face {  
    font-family: 'BLOKKNeue-Regular';  
    src: url('BLOKKNeue-Regular.eot');  
    src: url('BLOKKNeue-Regular.eot?#iefix') format('embedded-opentype'),  
         url('BLOKKNeue-Regular.woff') format('woff'),  
         url('BLOKKNeue-Regular.svg#BLOKKNeue-Regular') format('svg');  
    font-weight: normal;  
    font-style: normal;  
}
```

<http://www.css3files.com/font/>

- Apply as normal

```
body { font-family: 'BLOKKNeue-Regular'; }
```

Text shadows

- If you really want to...

```
h1 {text-shadow: 1px 3px 5px gray; }
```

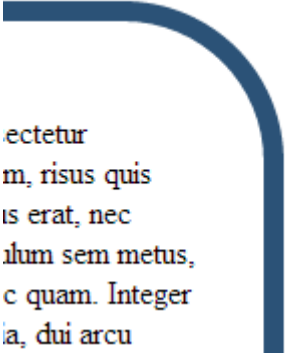
Lorem ipsum dolor

<http://www.css3files.com/shadow/>

Box model

- Rounded corners with border-radius

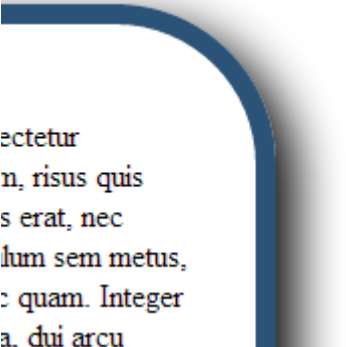
```
#thisbox {border-radius: 10px;}
```



ectetur
m, risus quis
is erat, nec
lum sem metus,
c quam. Integer
a, dui arcu

- Shadows with box-shadow

```
#thisbox {  
  box-shadow: 10px 10px 40px 0px #000000;  
}
```

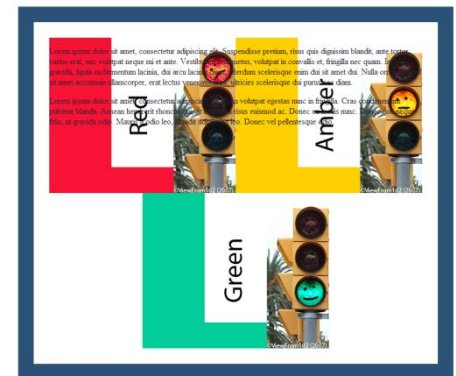


ectetur
n, risus quis
s erat, nec
lum sem metus,
c quam. Integer
a, dui arcu

Background enhancements

- Multiple background images

```
background-image: url('red-box.png'),  
                  url('amber-box.png'),  
                  url('green-box.png');  
background-position: 30px 30px,  
                    390px 30px,  
                    210px 330px;  
background-repeat: no-repeat;
```

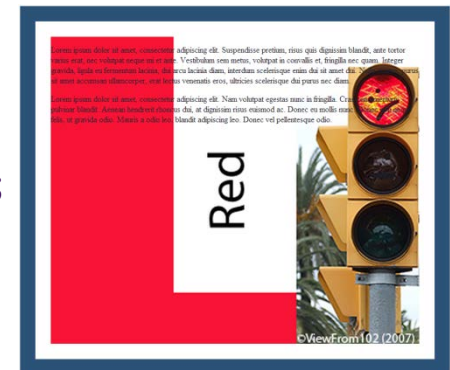


- Background origin
(includes/excludes the border)

```
background-origin: padding-box|content-box|border-box;
```

- Background sizing

```
background-size: 50% 50%;
```



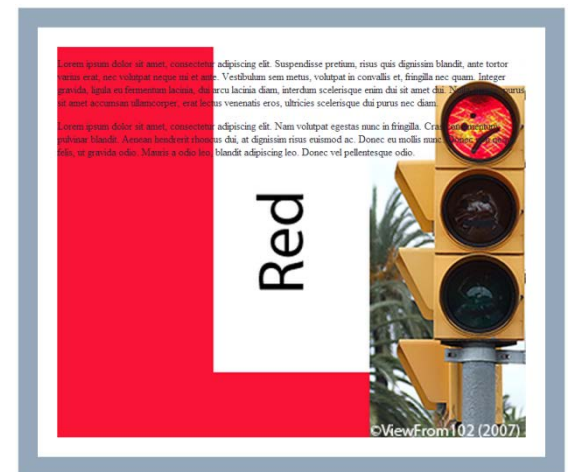
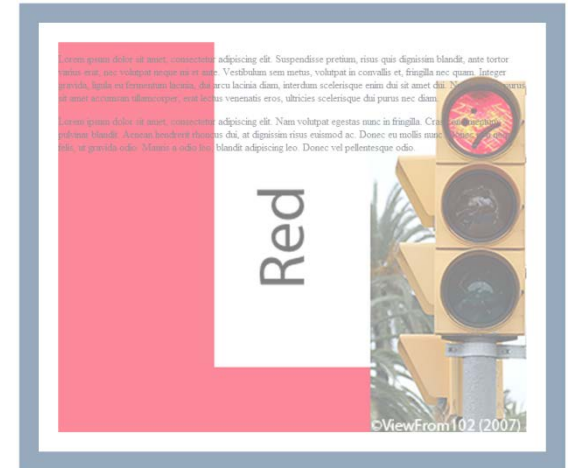
Opacity

- For whole elements use opacity

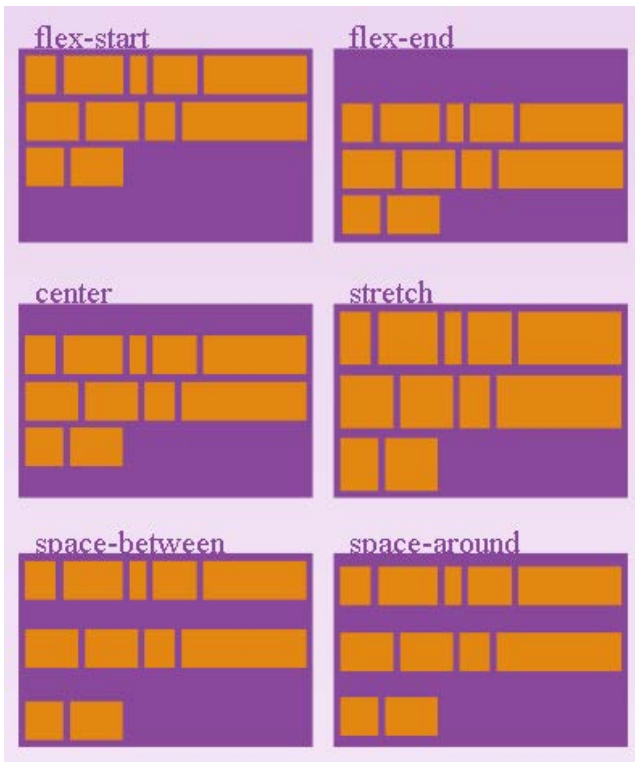
```
#thisbox {opacity: 0.5;}
```

- For just the colour use the new rgba specification






```
#thisbox {border-color: rgba(43,82,119,0.5);}
```



Positioning with Flexbox



- flex-container (parent) controls layout of flex-items (children)
- wrapping (e.g. multiple columns)
- alignment, justification
- direction of flow (ltr, ttb, ...)
- order of items

				
29.0	11.0	22.0	10	48

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Generated content

- Dynamically inserting content based on CSS patterns e.g.
 - text, images, counters etc.
- Uses better psuedo-element support in modern browsers

```
h1:before {content: "Banana: "}
```

+

```
<h1> A heading</h1>
```

= **Banana: A Heading**

http://www.westciv.com/style_master/academy/css_tutorial/advanced/generated_content.html

Media queries

- Cornerstone of "responsive design"
- Allow browser to adapt presentation based on browser dimensions

```
@media screen and (max-width: 980px) {  
    /*CSS rules for viewports smaller than 980px*/  
}  
@media screen and (max-width: 650px) {  
    /*CSS rules for viewports smaller than 650px*/  
}  
@media screen and (max-width: 480px) {  
    /*CSS rules for viewports smaller than 480px*/  
}
```

<http://webdesignerwall.com/tutorials/responsive-design-with-css3-media-queries>

References & look-up tables

<http://caniuse.com/>

<http://www.css3files.com/>

<http://html5please.com/>