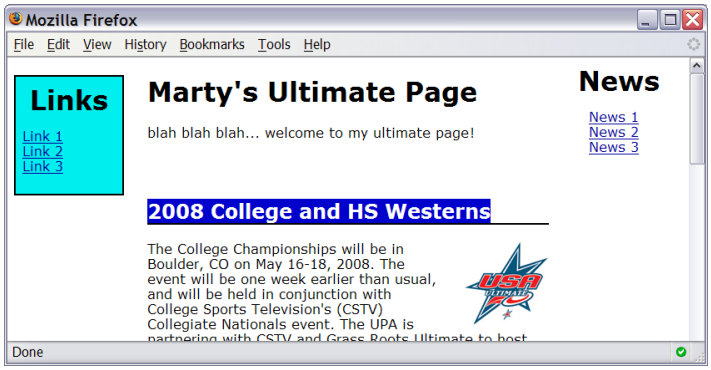
CSE3026: Web Application Development
Page Layout

Scott Uk-Jin Lee

4.1: Styling Page Sections

Motivation for page sections

- want to be able to **style individual elements, groups of elements, sections of text** or of the page
- (later) want to create complex page layouts



The HTML `id` attribute

```
<p>Spatula City!  Spatula City!</p>
<p id="mission">Our mission is to provide the most
spectacular spatulas and splurge on our specials until our
customers <q>esplode</q> with splendor!</p>
```

Spatula City! Spatula City!

Our mission is to provide the most spectacular spatulas and splurge on our specials until our customers “esplode” with splendor!

- allows you to give a unique ID to any element on a page
- each ID must be unique; can only be used once in the page

Linking to sections of a web page

```
<p>Visit <a href=
    "http://www.textpad.com/download/index.html#downloads">
  textpad.com</a> to get the TextPad editor.</p>

<p><a href="#mission">View our Mission Statement</a></p>
```

Visit textpad.com to get the TextPad editor.

View our Mission Statement

- a link target can include an ID at the end, preceded by a #
- browser will load that page and scroll to element with given ID

CSS ID selectors

```
#mission {
  font-style: italic;
  font-family: "Garamond", "Century Gothic", serif;
}
```

Spatula City! Spatula City!

Our mission is to provide the most spectacular spatulas and splurge on our specials until our customers "esplode" with splendor!

- applies style only to the paragraph that has the ID of `mission`
- element can be specified explicitly: `p#mission {`

The HTML class attribute

```
<p class="shout">Spatula City!  Spatula City!</p>
<p class="special">See our spectacular spatula specials!</p>
<p class="special">Today only: satisfaction guaranteed.</p>
```

Spatula City! Spatula City!

See our spectacular spatula specials!

Today only: satisfaction guaranteed.

- classes are a way to group some elements and give a style to only that group ("I don't want ALL paragraphs to be yellow, just these three...")
- unlike an `id`, a `class` can be reused as much as you like on the page

CSS class selectors

```
.special {              /* any element with class="special" */
  background-color: yellow;
  font-weight: bold;
}
p.shout {              /* only p elements with class="special" */
  color: red;
  font-family: cursive;
}
```

Spatula City! Spatula City!

See our spectacular spatula specials!

Today only: satisfaction guaranteed.

- applies rule to any element with class `special`, or a `p` with class `shout`

Multiple classes

```
<h2 class="shout">Spatula City!  Spatula City!</h2>
<p class="special">See our spectacular spatula specials!</p>
<p class="special shout">Satisfaction guaranteed.</p>
<p class="shout">We'll beat any advertised price!</p>
```

Spatula City! Spatula City!

See our spectacular spatula specials!

Satisfaction guaranteed.

We'll beat any advertised price!

- an element can be a member of multiple classes (separated by spaces)

CSS for following examples

```
.special {
  background-color: yellow;
  font-weight: bold;
}
.shout {
  color: red;
  font-family: cursive;
}
```

- for the next several slides, assume that the above CSS rules are defined

Sections of a page:

a section or division of your HTML page (block)

```
<div class="shout">
  <h2>Spatula City!  Spatula City!</h2>
  <p class="special">See our spectacular spatula specials!</p>
  <p>We'll beat any advertised price!</p>
</div>
```

Spatula City! Spatula City!

See our spectacular spatula specials!

We'll beat any advertised price!

- a tag used to indicate a logical section or area of a page
- has no appearance by default, but you can apply styles to it

Inline sections:

an inline element used purely as a range for applying styles

```
<h2>Spatula City!  Spatula City!</h2>
<p>See our <span class="special">spectacular</span> spatula specials!</p>
<p>We'll beat <span class="shout">any advertised price</span>!</p>
```

Spatula City! Spatula City!

See our **spectacular** spatula specials!

We'll beat **any advertised price**!

- has no onscreen appearance, but you can apply a style or ID to it, which will be applied to the text inside the span

CSS context selectors

```
selector1 selector2 {
  properties
}
```

- applies the given properties to *selector2* only if it is inside a *selector1* on the page

```
selector1 > selector2 {
  properties
}
```

- applies the given properties to *selector2* only if it is *directly* inside a *selector1* on the page (*selector2* tag is immediately inside *selector1* with no tags in between)

Context selector example

```
<p>Shop at <strong>Hardwick's Hardware</strong>...</p>
<ul>
  <li>The <strong>best</strong> prices in town!</li>
  <li>Act while supplies last!</li>
</ul>
```

```
li strong { text-decoration: underline; }
```

Shop at **Hardwick's Hardware**...

- The **best** prices in town!
- Act while supplies last!

More complex example

```
<div id="ad">
    <p>Shop at <strong>Hardwick's Hardware</strong>...</p>
    <ul>
        <li class="important">The <strong>best</strong> prices!</li>
        <li>Act <strong>while supplies last!</strong></li>
    </ul>
</div>
```

```
#ad li.important strong { text-decoration: underline; }
```

Shop at **Hardwick's Hardware**...

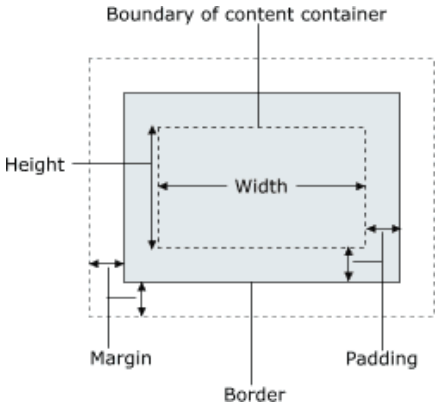
- The **best** prices!
- Act **while supplies last!**

4.2: Introduction to Layout

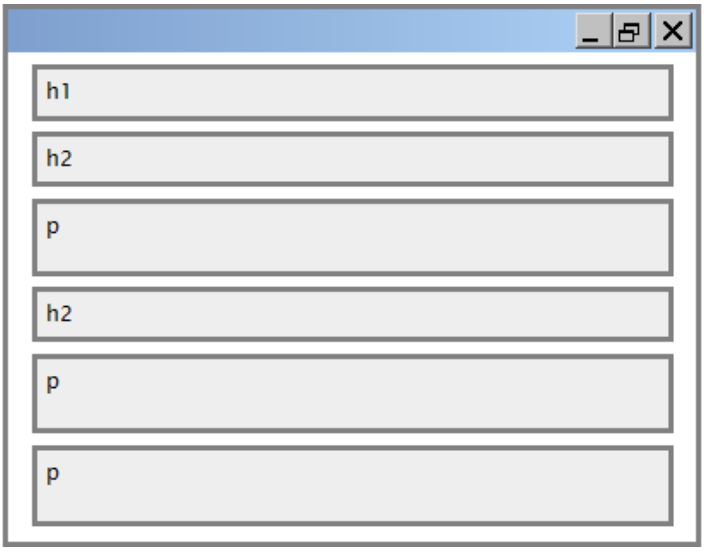
- 4.1: Styling Page Sections
- **4.2: Introduction to Layout**
- 4.3: Floating Elements
- 4.4: Sizing and Positioning

The CSS Box Model

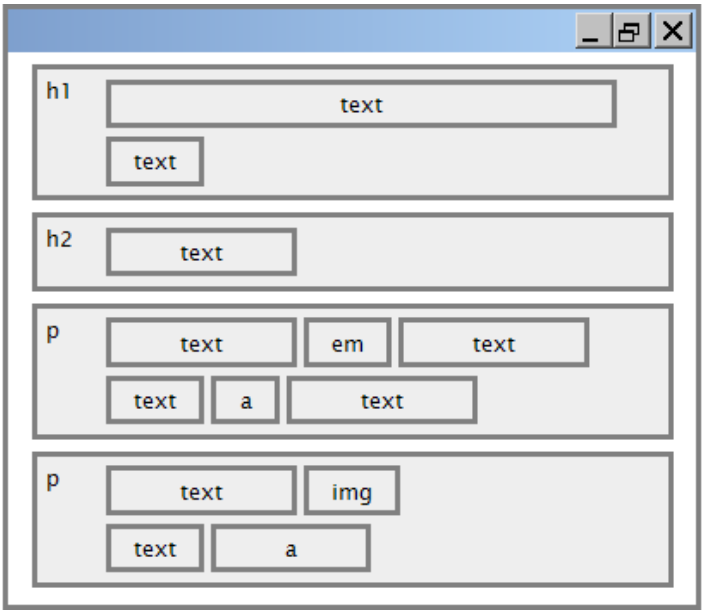
- for layout purposes, every element is composed of:
  - the actual element's **content**
  - a **border** around the element
  - **padding** between the content and the border (*inside*)
  - a **margin** between the border and other content (*outside*)
- width = content width + L/R padding + L/R border + L/R margin
  height = content height + T/B padding + T/B border + T/B margin
  - IE6 doesn't do this right



Document flow - block elements



Document flow - block and inline elements



CSS properties for borders

```
h2 { border: 5px solid red; }
```

This is a heading.

property	description
border	thickness/style/size of border on all 4 sides

- **thickness** (specified in px, pt, em, or thin, medium, thick)
- **style** (none, hidden, dotted, dashed, double, groove, inset, outset, ridge, solid)
- **color** (specified as seen previously for text and background colors)

More border properties

property	description
border-color, border-width, border-style	specific properties of border on all 4 sides
border-bottom, border-left, border-right, border-top	all properties of border on a particular side
border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, border-top-color, border-top-style, border-top-width	properties of border on a particular side
Complete list of border properties	

Border example 2

```
h2 {
  border-left: thick dotted #CC0088;
  border-bottom-color: rgb(0, 128, 128);
  border-bottom-style: double;
}
```

This is a heading.

- each side's border properties can be set individually
- if you omit some properties, they receive default values (e.g. border-bottom-width above)

Rounded corners with border-radius

```
p {
  border: 3px solid blue;
  border-radius: 12px;
  padding: 0.5em;
}
```

This is a paragraph.

This is another paragraph.
It spans multiple lines.

- each side's border radius can be set individually, separated by spaces

CSS properties for padding

property	description
padding	padding on all 4 sides
padding-bottom	padding on bottom side only
padding-left	padding on left side only
padding-right	padding on right side only
padding-top	padding on top side only
Complete list of padding properties	

Padding example 1

```
p { padding: 20px; border: 3px solid black; }
h2 { padding: 0px; background-color: yellow; }
```

This is the first paragraph

This is the second paragraph

This is a heading

Padding example 2

```
p {
  padding-left: 200px; padding-top: 30px;
  background-color: fuchsia;
}
```

This is the first paragraph

This is the second paragraph

- each side's padding can be set individually
- notice that padding shares the background color of the element

CSS properties for margins

property	description
margin	margin on all 4 sides
margin-bottom	margin on bottom side only
margin-left	margin on left side only
margin-right	margin on right side only
margin-top	margin on top side only
Complete list of margin properties	

Margin example 1

```
p {
  margin: 50px;
  background-color: fuchsia;
}
```

This is the first paragraph

This is the second paragraph

- notice that margins are always transparent
  (they don't contain the element's background color, etc.)

Margin example 2

```
p {
  margin-left: 8em;
  background-color: fuchsia;
}
```

This is the first paragraph

This is the second paragraph

- each side's margin can be set individually

CSS properties for dimensions

```
p { width: 350px; background-color: yellow; }
h2 { width: 50%; background-color: aqua; }
```

This paragraph uses the first style above.

An h2 heading

property	description
width, height	how wide or tall to make this element (block elements only)
max-width, max-height, min-width, min-height	max/min size of this element in given dimension

Centering a block element: auto margins

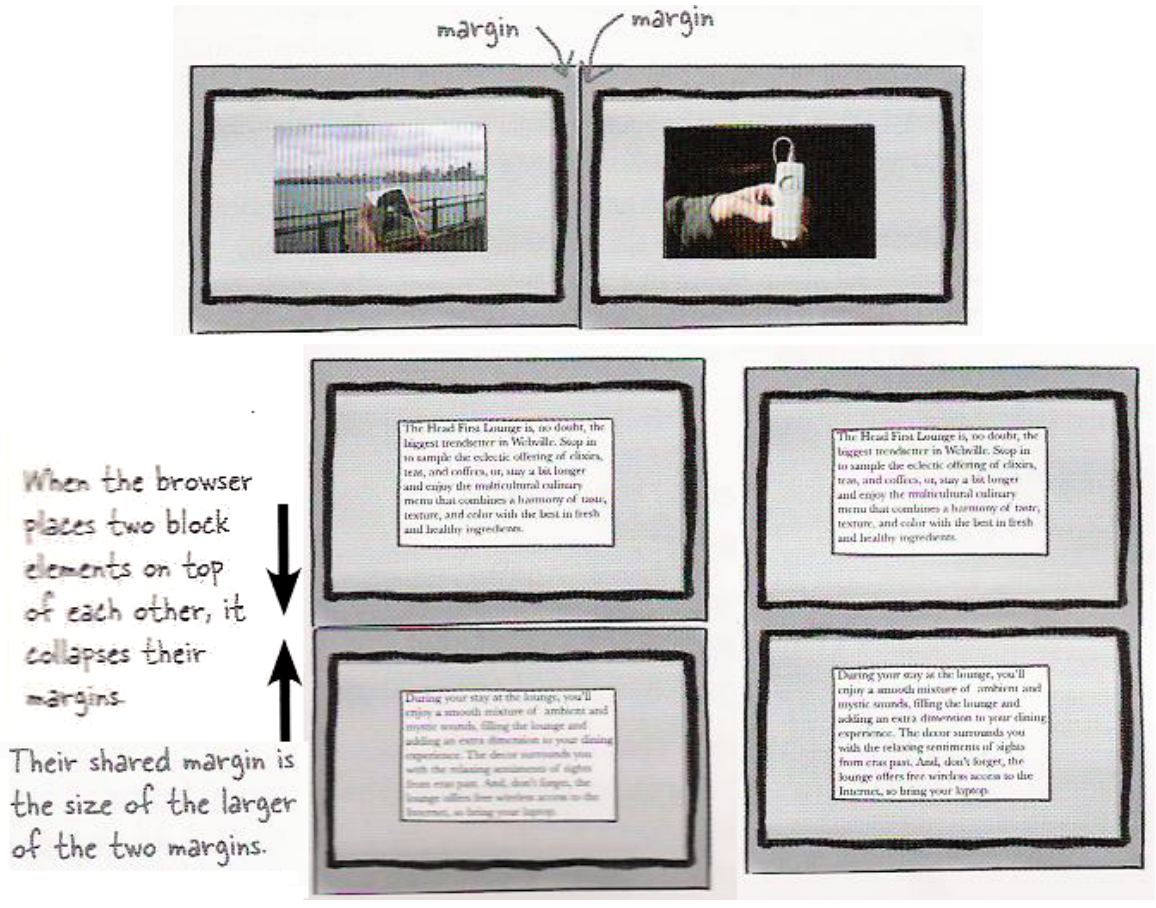
```
p {
  margin-left: auto;
  margin-right: auto;
  width: 750px;
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- works best if `width` is set (otherwise, may occupy entire width of page)
- to center inline elements within a block element, use `text-align: center;`

Top/bottom margin collapse

- when two block elements appear on top of each other, their margins are collapsed
- their shared margin is the larger of the two individual margins

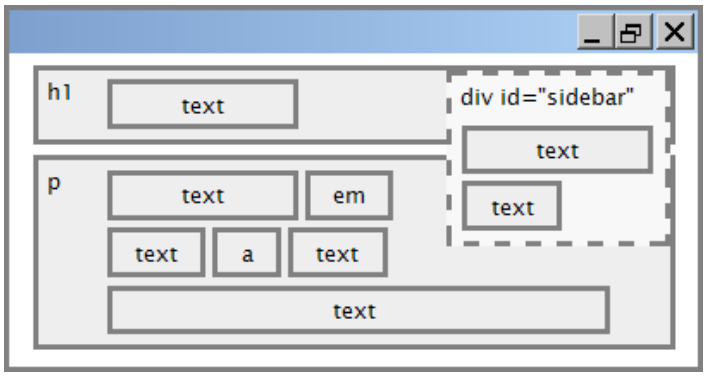


4.3: Floating Elements

The CSS `float` property (reference)

property	description
`float`	side to hover on; can be `left`, `right`, or `none` (default)

- a *floating* element is removed from normal document flow
- underlying text wraps around it as necessary



Float example

```
<img src="images/borat.jpg" alt="Borat" class="headericon" />
Borat Sagdiyev (born July 30, 1972) is a ...
```

```
img.headericon {
  float: left;
}
```

Borat Sagdiyev (born July 30, 1972) is a fictional Kazakhstani journalist played by British-Jewish comedian Sacha Baron Cohen. He is the main character portrayed in the controversial and successful film Borat: Cultural Learnings of America for Make Benefit Glorious Nation of Kazakhstan ...

Float vs. alignment

none 1 before

none 2 before

left #1 left #2 none 1 after right #2 right #1

none 2 after

- using Firebug, toggle the above `div`s from being aligned to floated...

Common floating content and width

I am not floating, no width set

I am floating right, no width set

I am floating right, no width set, but my text is very long so this paragraph doesn't really seem like it's floating at all, darn

I am not floating, 45% width

I am floating right, 45% width

- often floating elements should have a `width` property value
  - if no `width` is specified, other content may be unable to wrap around the floating element

The clear property

```
p { background-color: fuchsia; }
h2 { clear: right; background-color: yellow; }
```

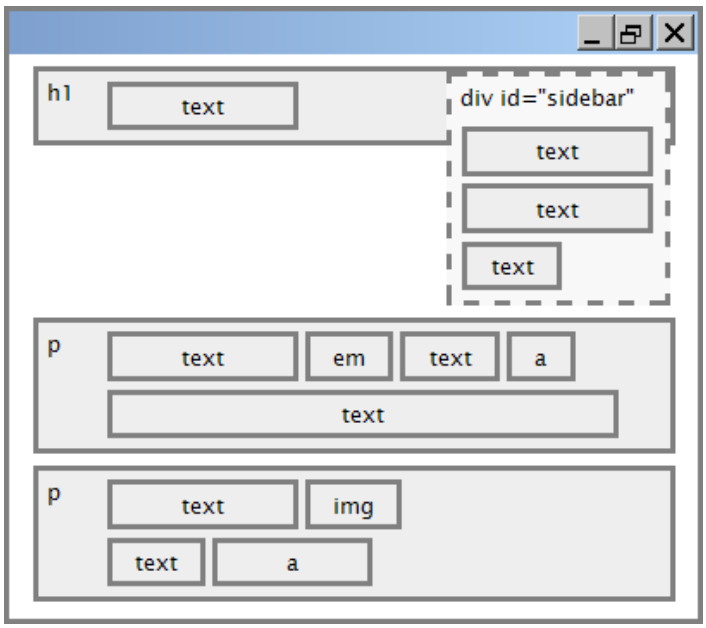
Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ...

My Homestar Runner Fan Site

property	description
`clear`	disallows floating elements from overlapping this element; can be `left`, `right`, `both`, or `none` (default)

Clear diagram

```
div#sidebar { float: right; }
p { clear: right; }
```

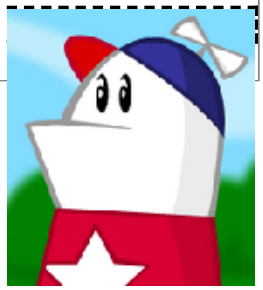


Common error: container too short (4.3.3)

```
<p><img src="images/homestar_runner.png" alt="homestar runner" />
  Homestar Runner is a Flash animated Internet cartoon.
  It mixes surreal humour with ....</p>
```

```
p { border: 2px dashed black; }
img { float: right; }
```

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....



- We want the p containing the image to extend downward so that its border encloses the entire image

The overflow property

```
p { border: 2px dashed black; overflow: hidden; }
```

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....

property	description
overflow	specifies what to do if an element's content is too large; can be auto, visible, hidden, or scroll

Multi-column layouts

```
<div>
  <p>the first paragraph</p>
  <p>the second paragraph</p>
  <p>the third paragraph</p>
  Some other text that is important
</div>
```

```
p { float: right; width: 20%; margin: 0.5em;
    border: 2px solid black; }
div { border: 3px dotted green; overflow: hidden; }
```

Some other text that is important

the third paragraph

the second paragraph

the first paragraph

4.4: Sizing and Positioning

- 4.1: Styling Page Sections
- 4.2: Introduction to Layout
- 4.3: Floating Elements
- **4.4: Sizing and Positioning**

The position property (examples)

```
div#ad {
  position: fixed;
  right: 10%;
  top: 45%;
}
```

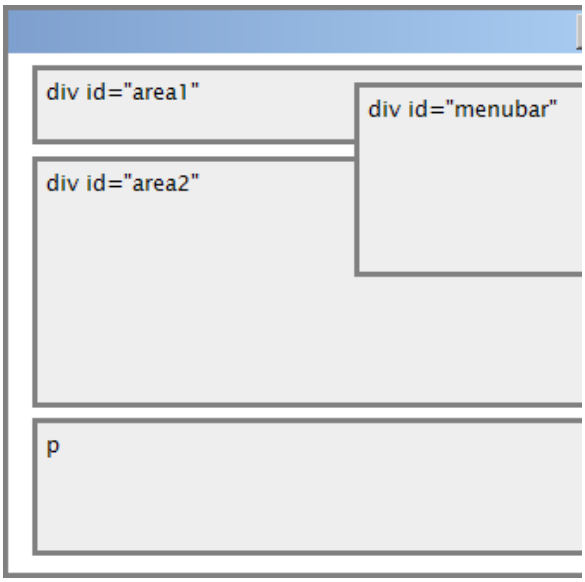
Here I am!

property	value	description
position	static	default position
	relative	offset from its normal static position
	absolute	a fixed position *within its containing element*
	fixed	a fixed position *within the browser window*
top, bottom, left, right	positions of box's corners	

Absolute positioning

```
#menubar {
  position: absolute;
  left: 400px;
  top: 50px;
}
```

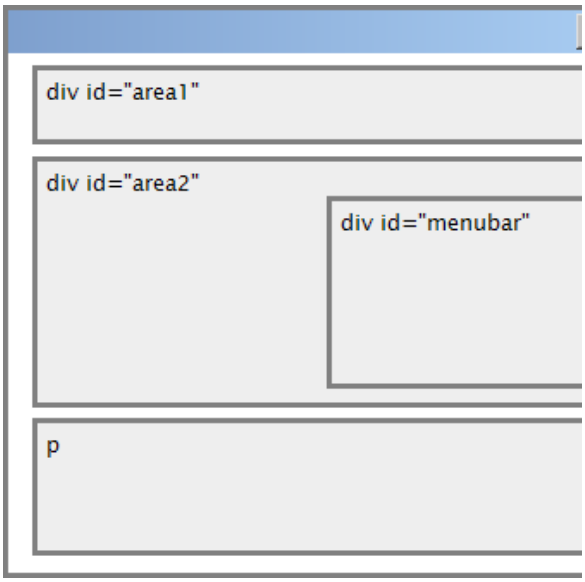
- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses `absolute` or `relative` positioning)
- actual position determined by `top`, `bottom`, `left`, `right` values
- should often specify a `width` property as well



Relative positioning

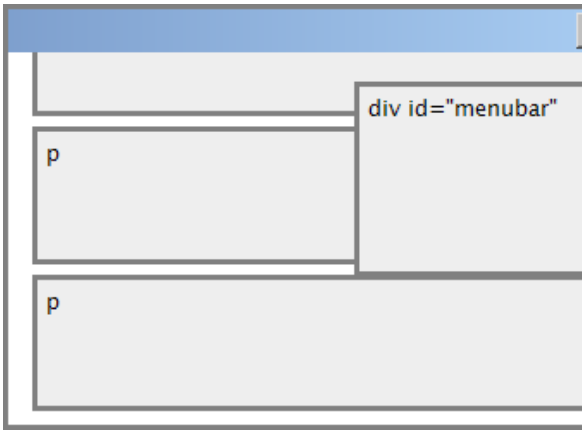
```
#area2 { position: relative; }
```

- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page
- to instead cause the absolute element to position itself relative to some other element's corner, wrap the `absolute` element in an element whose `position` is `relative`



Fixed positioning

- removed from normal flow (like floating ones)
- positioned relative to the browser window
  - even when the user scrolls the window, element will remain in the same place



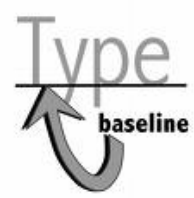
Alignment vs. float vs. position

1. if possible, lay out an element by *aligning* its content
   - horizontal alignment: `text-align`
     - set this on a block element; it aligns the content within it (not the block element itself)
   - vertical alignment: `vertical-align`
     - set this on an inline element, and it aligns it vertically within its containing element
2. if alignment won't work, try *floating* the element
3. if floating won't work, try *positioning* the element
   - absolute/fixed positioning are a last resort and should not be overused

The vertical-align property

property	description
vertical-align	specifies where an inline element should be aligned vertically, with respect to other content on the same line within its block element's box

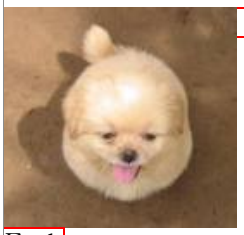
- can be top, middle, bottom, baseline (default), sub, super, text-top, text-bottom, or a length value or %
  - baseline means aligned with bottom of non-hanging letters



vertical-align example

```
<p style="background-color: yellow;">
<span style="vertical-align: top; border: 1px solid red;">
Don't be sad!  Turn that frown
<img src="images/sad.jpg" alt="sad" /> upside down!
<img style="vertical-align: bottom" src="images/smiley.jpg" alt="smile" />
Smiling burns calories, you know.
<img style="vertical-align: middle" src="images/puppy.jpg" alt="puppy" />
Anyway, look at this cute puppy; isn't he adorable!  So cheer up,
and have a nice day.  The End.
</span></p>
```

Don't be sad! Turn that frown  upside down!  Smiling burns calories, you know.

 Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.

Common bug: space under image

```
<p style="background-color: red; padding: 0px; margin: 0px">
<img src="images/smiley.png" alt="smile" />
</p>
```

- red space under the image, despite `padding` and `margin` of 0
- this is because the image is vertically aligned to the baseline of the paragraph (not the same as the bottom)
- setting `vertical-align` to `bottom` fixes the problem (so does setting `line-height` to `0px`)

Details about inline boxes

- size properties (`width`, `height`, `min-width`, etc.) are ignored for inline boxes
- `margin-top` and `margin-bottom` are ignored, but `margin-left` and `margin-right` are not
- the containing block box's `text-align` property controls horizontal position of inline boxes within it
  - text-align does not align block boxes within the page
- each inline box's `vertical-align` property aligns it vertically within its block box

The display property

```
h2 { display: inline; background-color: yellow; }
```

This is a heading This is another heading

property	description
display	sets the type of CSS box model an element is displayed with

- values: none, inline, block, run-in, compact, ...
- use sparingly, because it can radically alter the page layout

Displaying block elements as inline

```
<ul id="topmenu">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

```
#topmenu li {
  display: inline;
  border: 2px solid gray;
  margin-right: 1em;
}
```

Item 1 Item 2 Item 3

- lists and other block elements can be displayed inline
  - flow left-to-right on same line
  - width is determined by content (block elements are 100% of page width)

The visibility property

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

property	description
visibility	sets whether an element should be shown onscreen; can be visible (default) or hidden

- hidden elements will still take up space onscreen, but will not be shown
  - to make it not take up any space, set display to none instead
- can be used to show/hide dynamic HTML content on the page in response to events

The opacity property

```
body     { background-image: url("images/marty-mcfly.jpg"); background-repeat: repeat; }
p        { background-color: yellow; margin: 0; padding: 0.25em; }
p.mcfly1 { opacity: 0.75; }
p.mcfly2 { opacity: 0.50; }
p.mcfly3 { opacity: 0.25; }
```

Marty McFly in 1985

Marty McFly in 1955 fading away, stage 1

Marty McFly in 1955 fading away, stage 2

Marty McFly in 1955 fading away, stage 3

property	description
opacity	how not-transparent the element is; value ranges from 1.0 (opaque) to 0.0 (transparent)

HTML5 and layout: The old way

- web pages often have to give semantic meaning to content through `class` and `id` attributes, rather than through the tag elements themselves

HTML5 semantically meaningful tags