Web Programming Step by Step

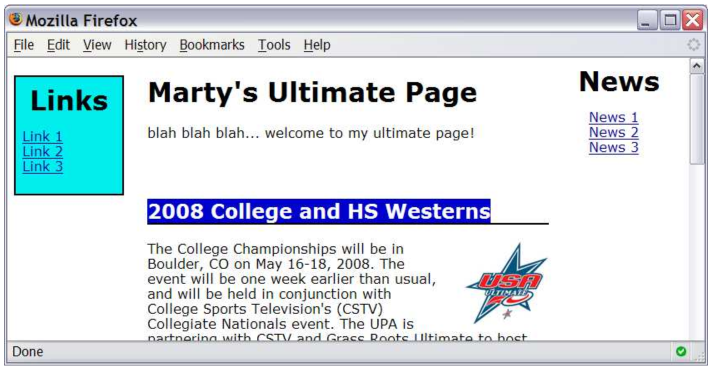
Chapter 4
Page Layout

4.1: Styling Page Sections

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

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



Marty's Ultimate Page

blah blah blah... welcome to my ultimate page!

2008 College and HS Westerns

The College Championships will be in Boulder, CO on May 16-18, 2008. The event will be one week earlier than usual, and will be held in conjunction with College Sports Television's (CSTV) Collegiate Nationals event. The UPA is partnering with CSTV and Grass Roots Ultimate to host

Sections of a page: `<div>` (4.1.1)

a section or division of your HTML page (block)

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

```
Spatula City! Spatula City!

See our spectacular spatula specials!

We'll beat any advertised price!
```
output

- 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: `<span>` (4.1.2)

an inline element used purely as a range for applying styles

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

Spatula City! Spatula City!

See our **spectacular** spatula specials!

We'll beat any advertised price!

output

- 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 (4.1.3)

```css
selector1 selector2 {
   properties
}
```
CSS

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

```css
selector1 > selector2 {
   properties
}
```
CSS

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

Context selector example

```html
<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>
```
HTML

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

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

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

output

More complex example

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

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

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

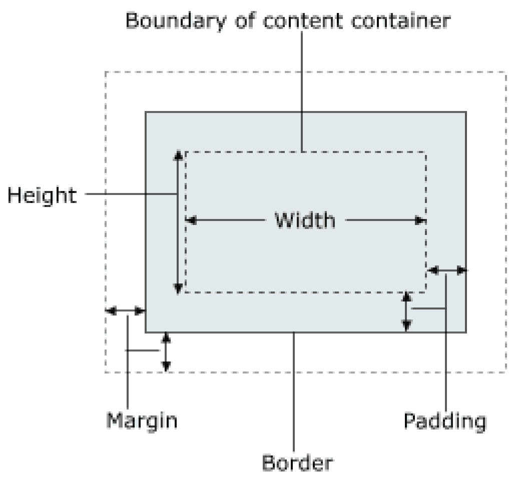
- The **best** prices in town!
- Act **while supplies last!**

output

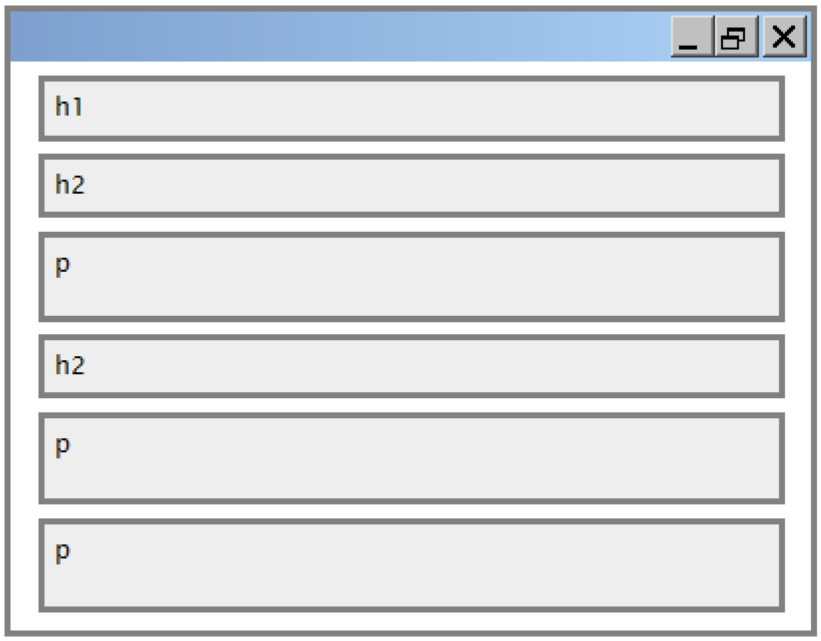
4.2: Introduction to Layout

The CSS Box Model (4.2.1)

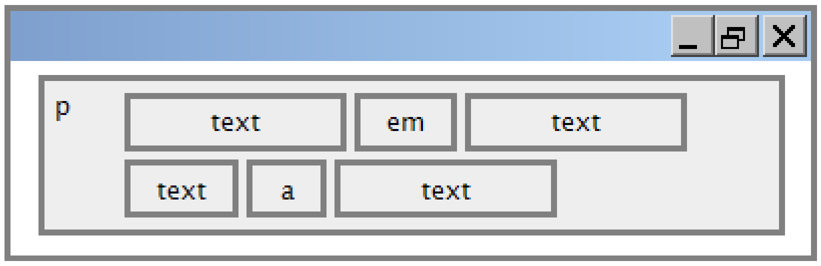
- for layout purposes, every element is composed of:
  - the actual element's **content**
  - a **border** around the element
  - **padding** between the content and the border (*inside*)
  - a **margin** between the border and other content (*outside*)
- width = content width + L/R padding + L/R border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



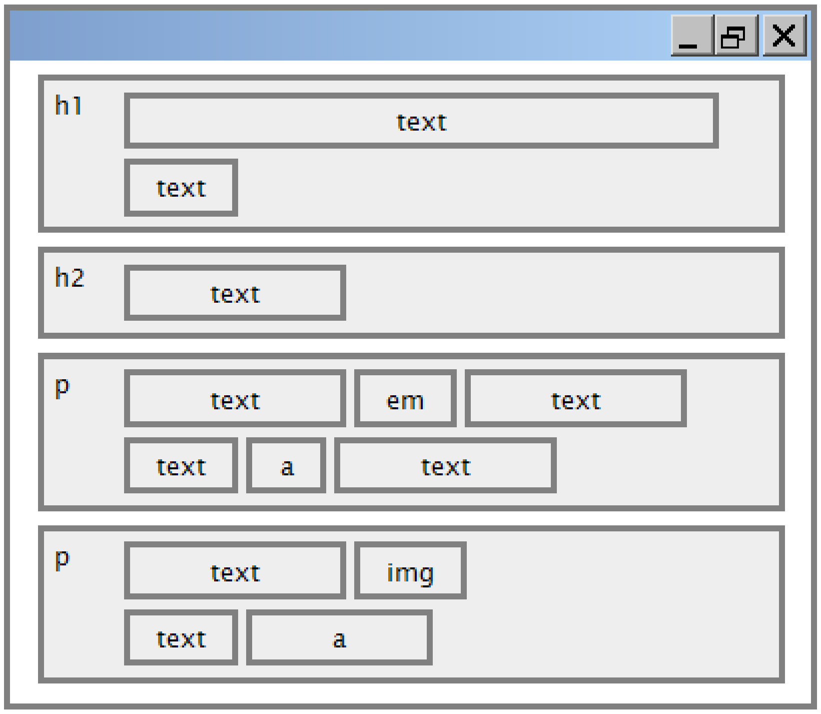
Document flow - block elements



Document flow - inline elements



Document flow - a larger example



CSS properties for borders

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

CSS

This is a heading.

output

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

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

This is a heading.

output

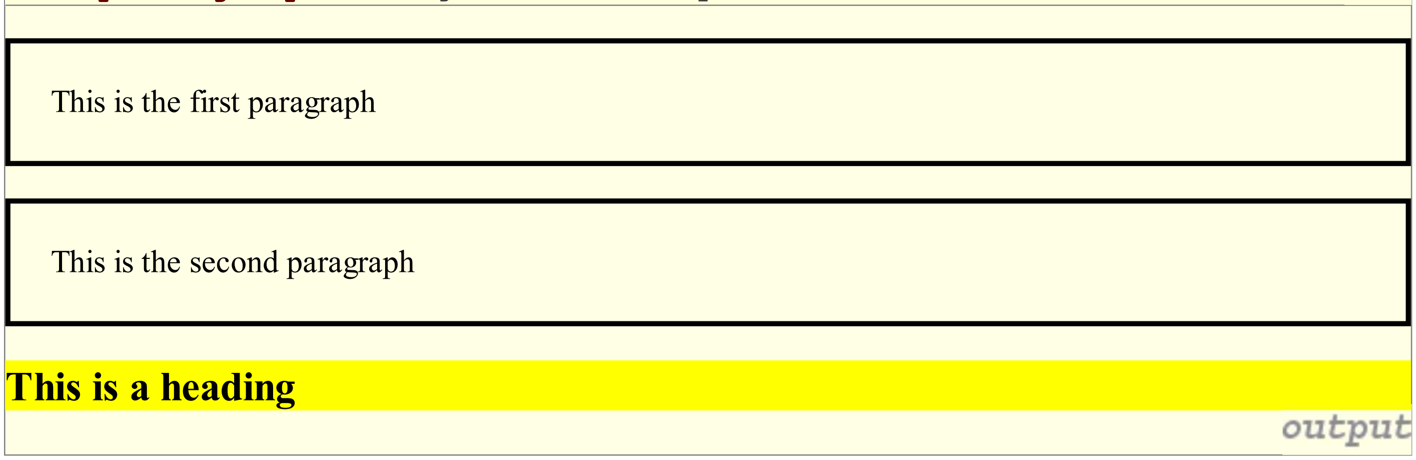
- each side's border properties can be set individually
- if you omit some properties, they receive default values (e.g. `border-bottom-width` above)

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

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


This is the first paragraph

This is the second paragraph

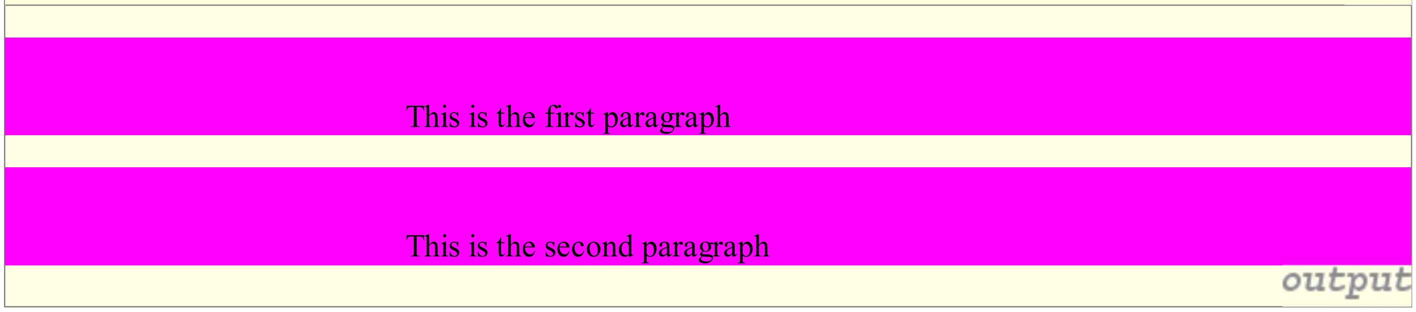
This is a heading

output

Padding example 2

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

CSS



This is the first paragraph

This is the second paragraph

output

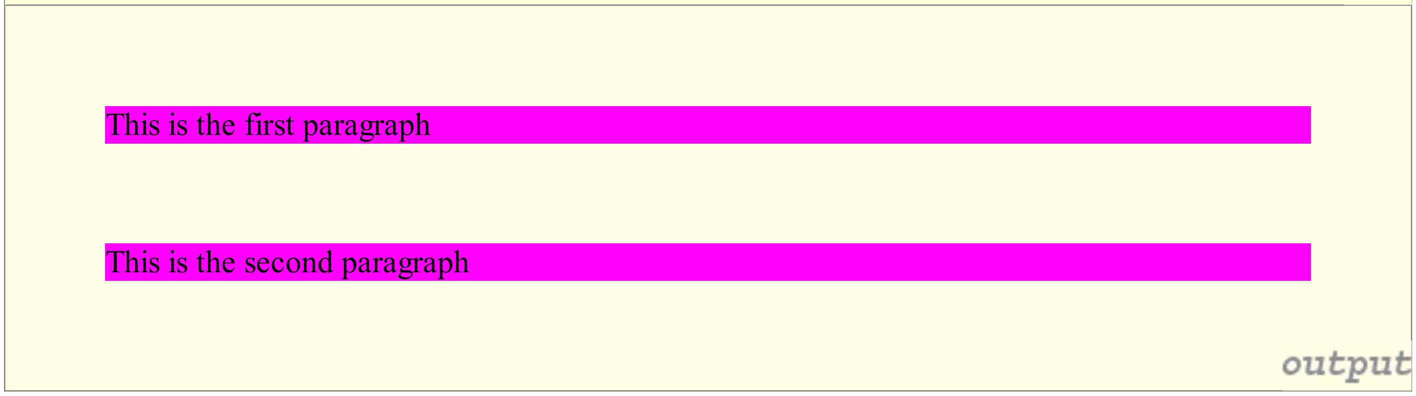
- each side's padding can be set individually
- notice that padding shares the background color of the element

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

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



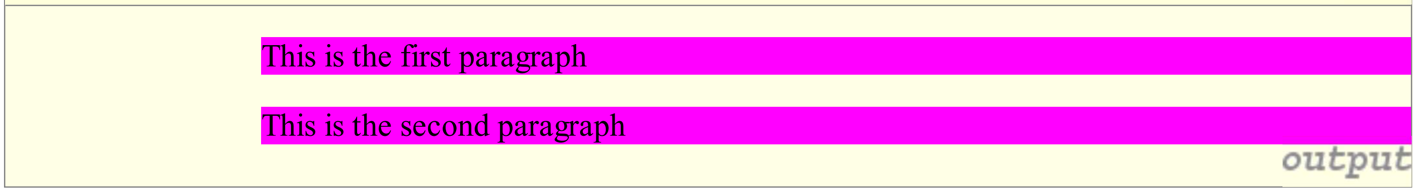
```
This is the first paragraph

This is the second paragraph
```
output

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

Margin example 2

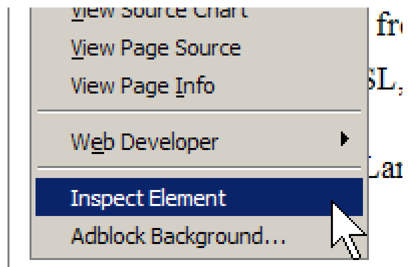
```css
p {
  margin-left: 8em;
  background-color: fuchsia;
}
```
CSS

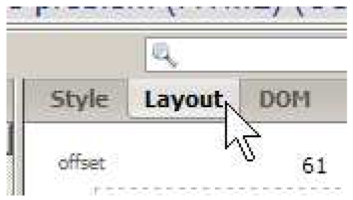


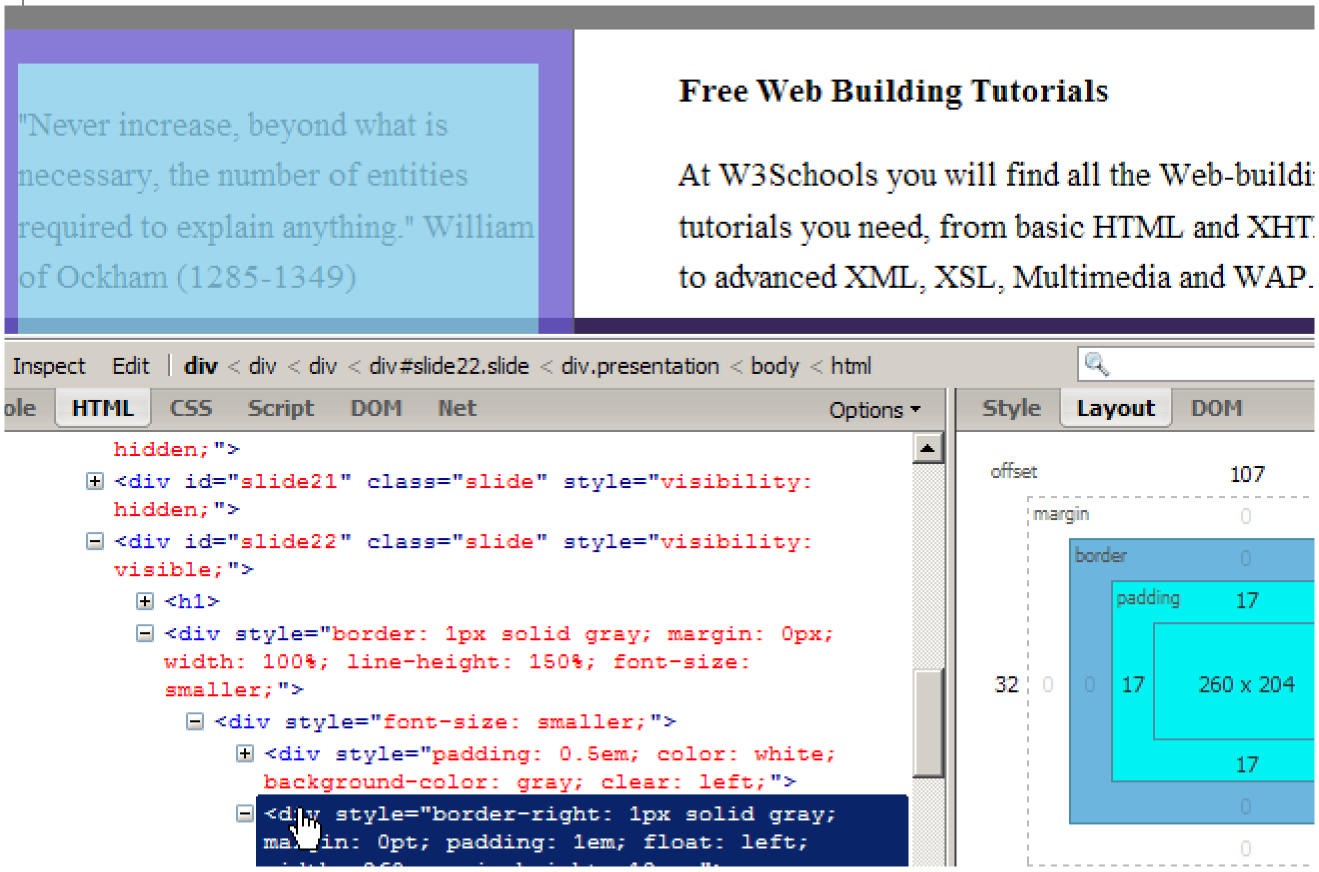
```
This is the first paragraph
This is the second paragraph
```
output

- each side's margin can be set individually

Firefox Firebug add-on (4.2.2)







Free Web Building Tutorials

At W3Schools you will find all the Web-buildi
tutorials you need, from basic HTML and XHT
to advanced XML, XSL, Multimedia and WAP.

"Never increase, beyond what is necessary, the number of entities required to explain anything." William of Ockham (1285-1349)

```
hidden;">
⊞ <div id="slide21" class="slide" style="visibility:
  hidden;">
⊟ <div id="slide22" class="slide" style="visibility:
  visible;">
    ⊞ <h1>
    ⊟ <div style="border: 1px solid gray; margin: 0px;
      width: 100%; line-height: 150%; font-size:
      smaller;">
        ⊟ <div style="font-size: smaller;">
            ⊞ <div style="padding: 0.5em; color: white;
              background-color: gray; clear: left;">
            ⊟ <div style="border-right: 1px solid gray;
              margin: 0pt; padding: 1em; float: left;
```

4.3: Floating Elements

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

CSS properties for dimensions (4.3, 4.4.1)

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

This paragraph uses the first style above.

An h2 heading

output

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;
}
```
CSS

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

output

- 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;`

The CSS `float` property (reference) (4.3.1)

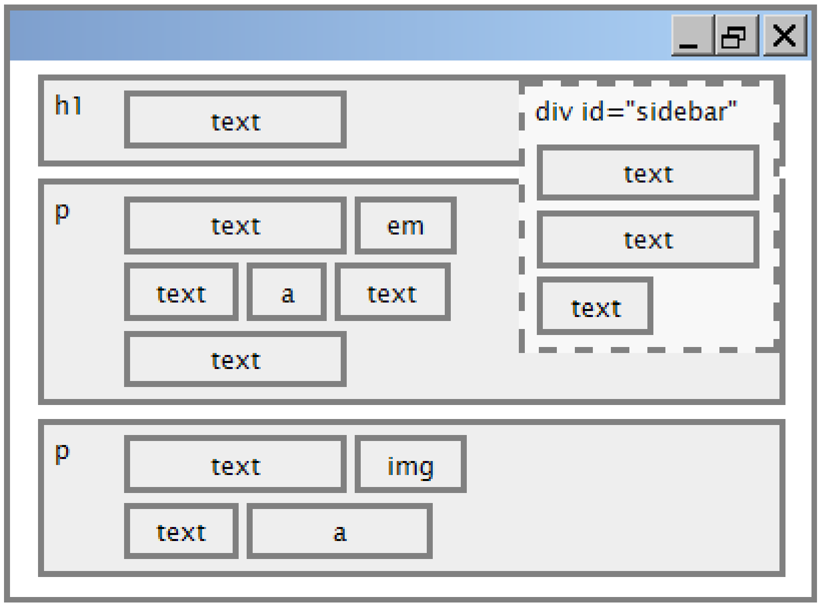
```
img.headericon {
  float: right;    width: 130px;
}
```
CSS

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 ...

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

- removed from normal document flow; underlying text wraps around as necessary

Floating elements diagram



Common `float` bug: missing width

I am not floating, no width

I am floating right, no width

I am not floating, 45% width

I am floating right, 45% width

- often floating block elements must have a `width` property value
  - if no `width` is specified, the floating element may occupy 100% of the page width, so no content can wrap around it

The `clear` property (4.3.2)

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

CSS

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with references to 1980s and 1990s pop culture, notably video games, classic television and popular music.

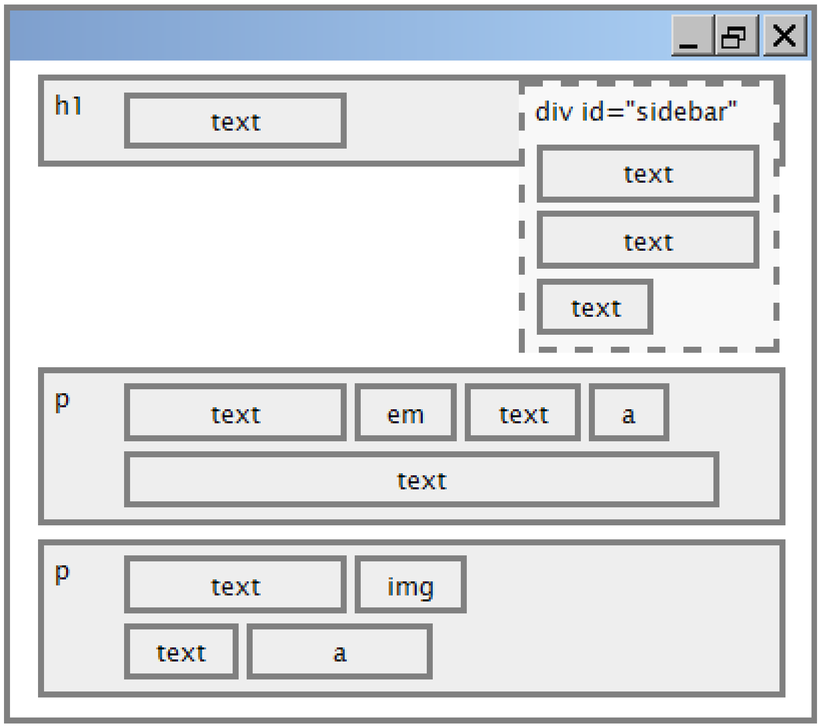
My Homestar Runner Fan Site

output

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

Clear diagram

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



4.4: Sizing and Positioning

The `position` property (examples) (4.4.2)

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

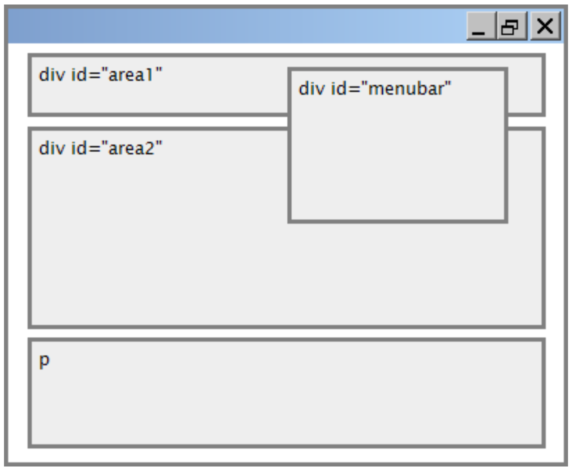
Here I am!

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

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

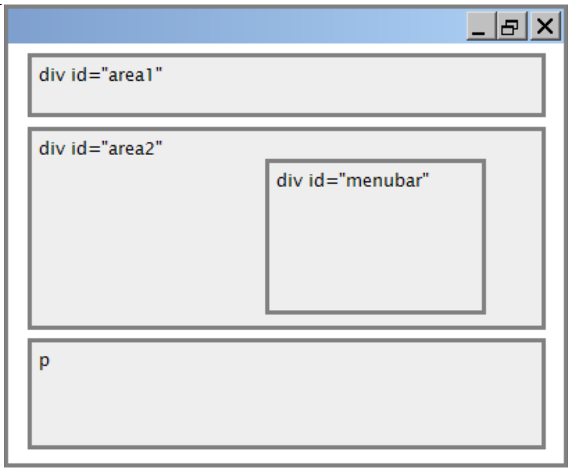
- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses `absolute` or `relative` positioning)
- actual position determined by `top`, `bottom`, `left`, `right` values
- should often specify a `width` property as well



Relative positioning

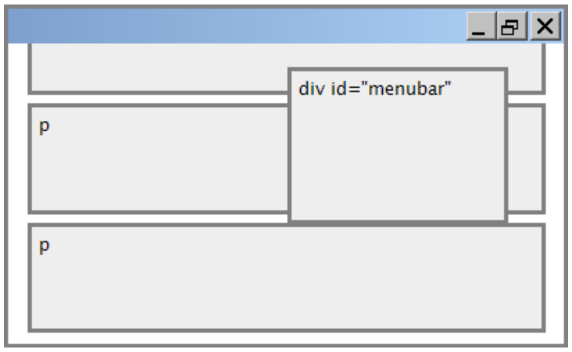
```css
#area2 { position: relative; }
```
CSS

- 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

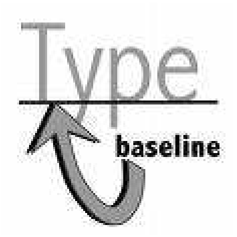
Details about inline boxes

- 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 `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

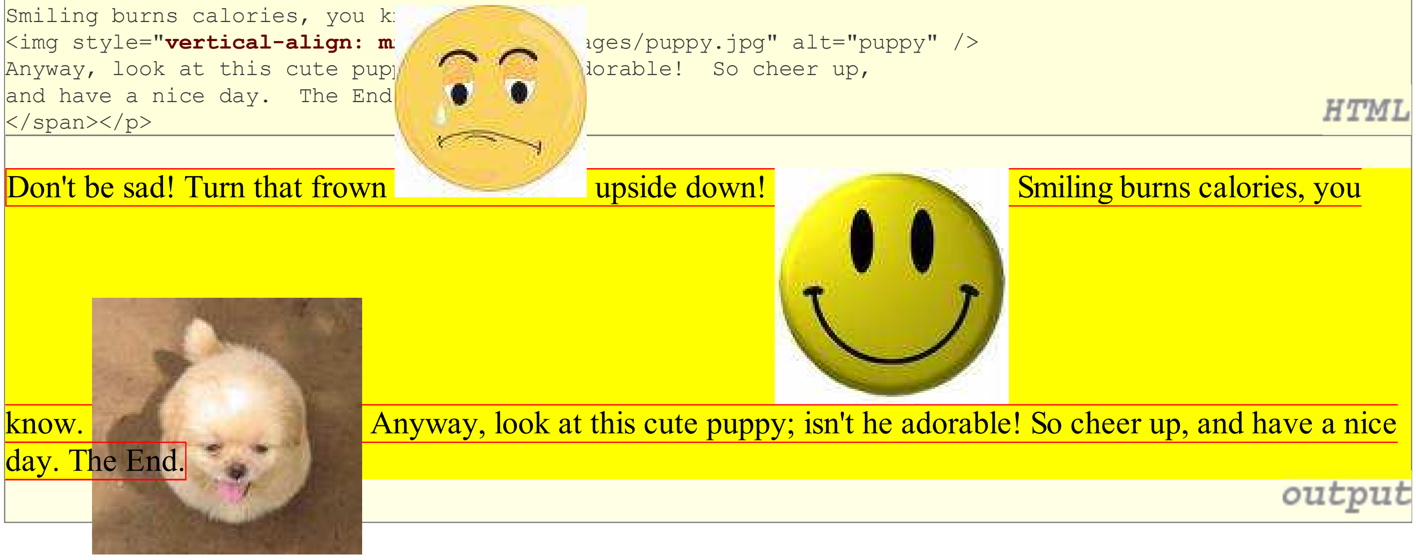
- can be `top`, `middle`, `bottom`, `baseline` (default), `sub`, `super`, `text-top`, `text-bottom`, or a length value or `%`
  - `baseline` means aligned with bottom of non-hanging letters



`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 k
<img style="vertical-align: m                ages/puppy.jpg" alt="puppy" />
Anyway, look at this cute pupp               dorable!  So cheer up,
and have a nice day.  The End
</span></p>
```

HTML

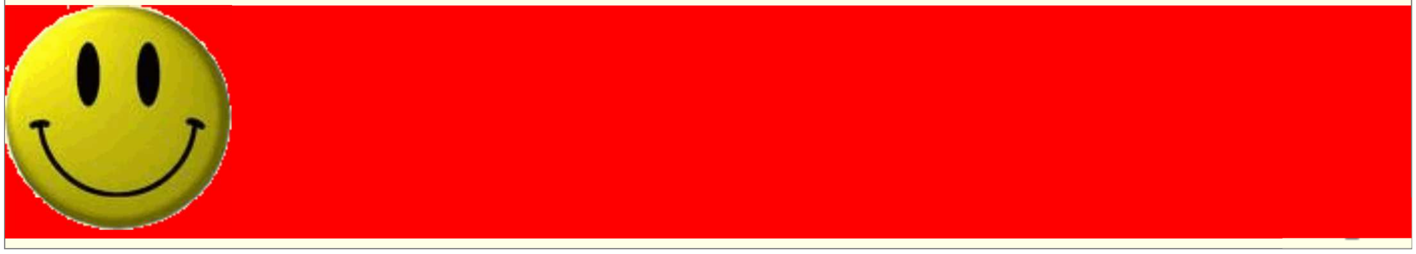


Don't be sad! Turn that frown upside down! Smiling burns calories, you know. Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.

output

Common bug: space under image

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



- 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`)

The `z-index` property (4.4.3)

property	description
`z-index`	sets which absolute positioned element will appear on top of another that occupies the same space

- higher `z-index` elements appears on top of lower ones
- can be `auto` (default) or a number

The `display` property (4.4.4)

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

This is a heading **This is another heading**

output

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>
```
HTML

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

Item 1 Item 2 Item 3

output

- 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;
}
```

CSS

output

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