

Advanced Page Layout

CSE 190 M (Web Programming), Spring 2008
University of Washington

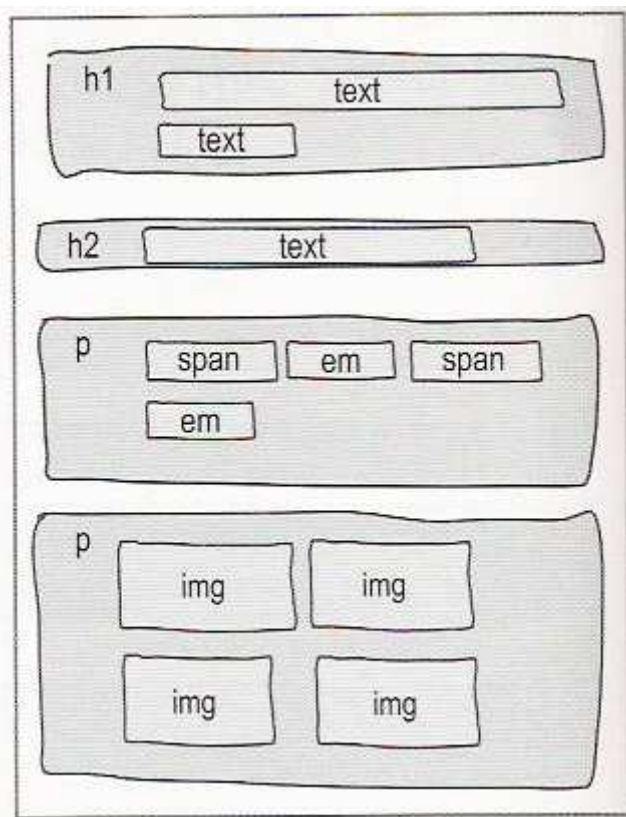
Reading: Chapter 2, sections 2.4 - 2.6

Except where otherwise noted, the contents of this presentation are © Copyright 2008 Marty Stepp and Jessica Miller and are licensed under the Creative Commons Attribution 2.5 License.



Recall: Document flow

Here I am!



The position property (examples)

```
div#rightside {  
  position: fixed;  
  right: 10%;  
  top: 36%;  
}
```

CSS

- `static` : default position
- `relative` : offset from its normal static position
- `absolute` : at a fixed position *within its containing element*
- `fixed` : at a fixed position *within the browser window*
 - `top`, `bottom`, `left`, `right` properties specify positions of box's corners

Absolute positioning

```
#sidebar {
  position: absolute;
  top: 100px;
  right: 200px;
  width: 280px;
}
```

Because sidebar is now absolutely positioned, it is removed from the flow and positioned according to any `top`, `left`, `right`, or `bottom` properties that are specified.

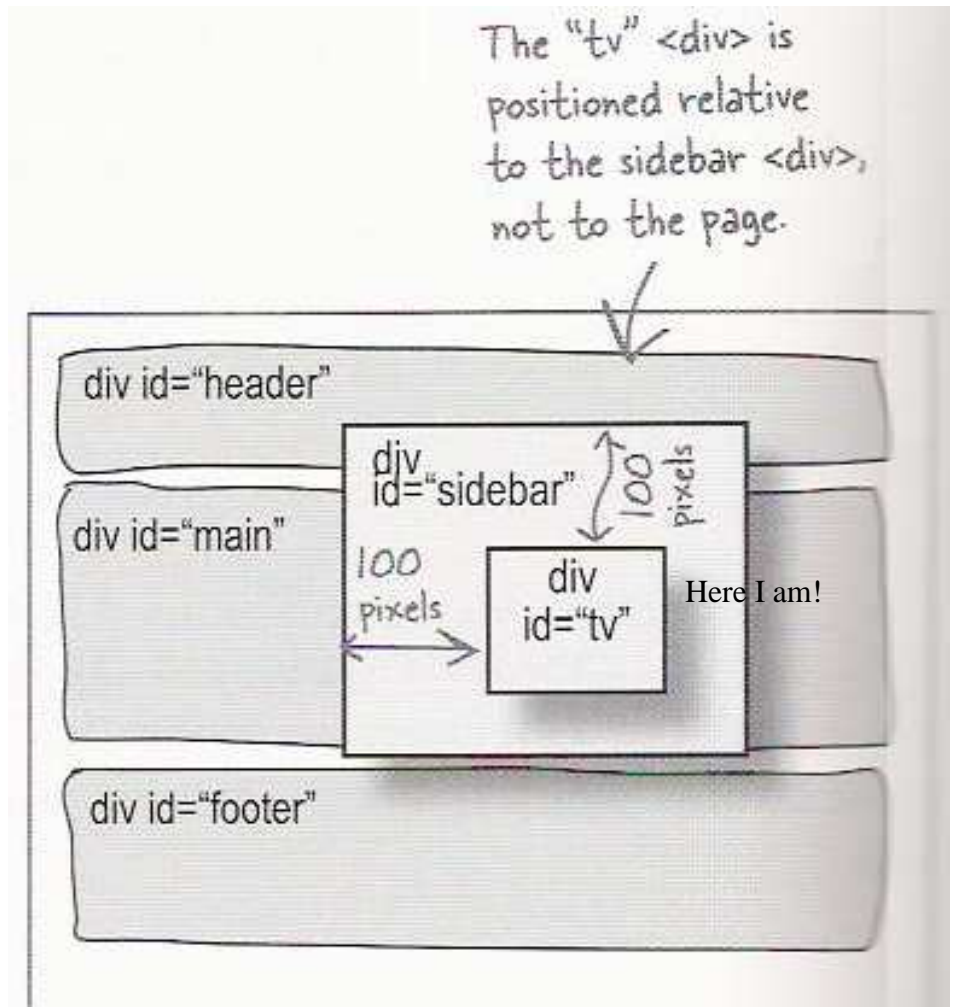
Because the sidebar is out of the flow, the other elements don't even know it is there, and they ignore it totally.

Elements that are in the flow don't even wrap their inline content around an absolutely positioned element. They are totally oblivious to it being on the page.

- 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

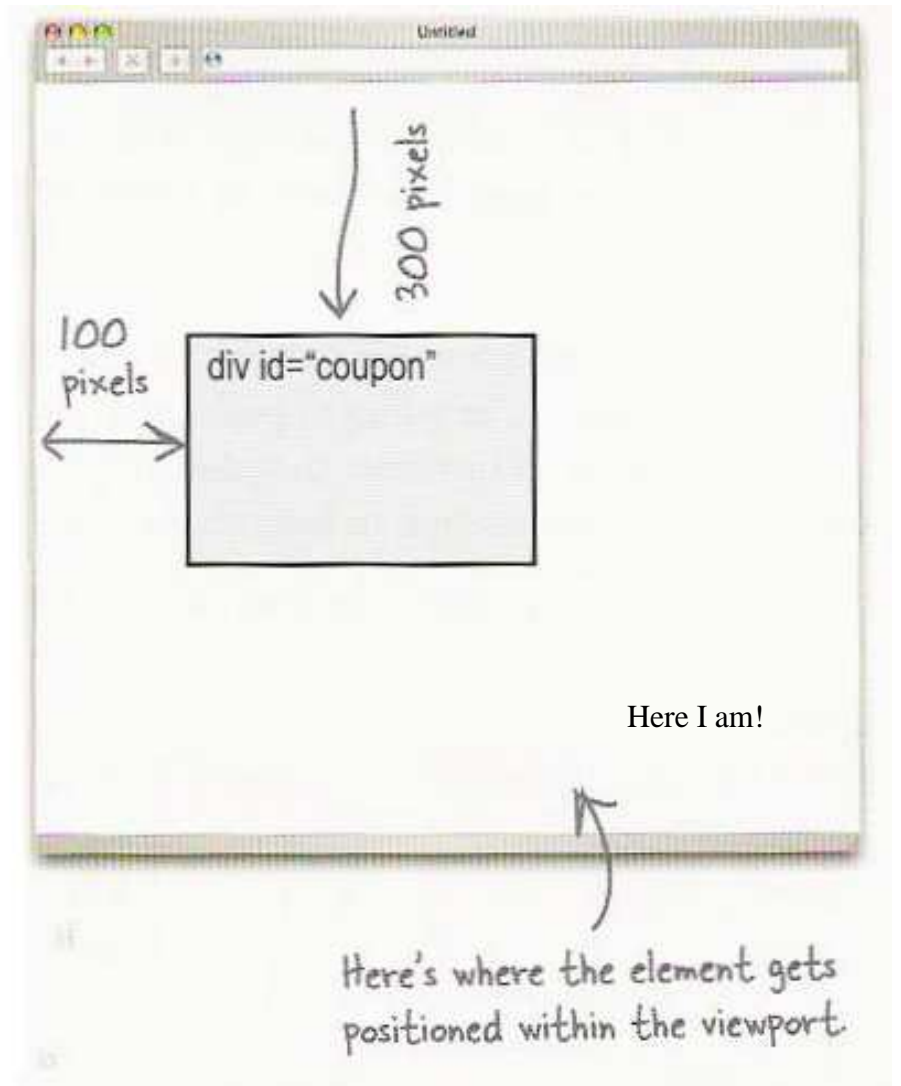
Absolute positioning details

- positioned relative to the block element containing them
- to position many elements absolutely but close to their normal default position, wrap the absolute elements in a relative element



Fixed positioning

```
#coupon {  
  position: fixed;  
  top: 300px;  
  left: 100px;  
}
```

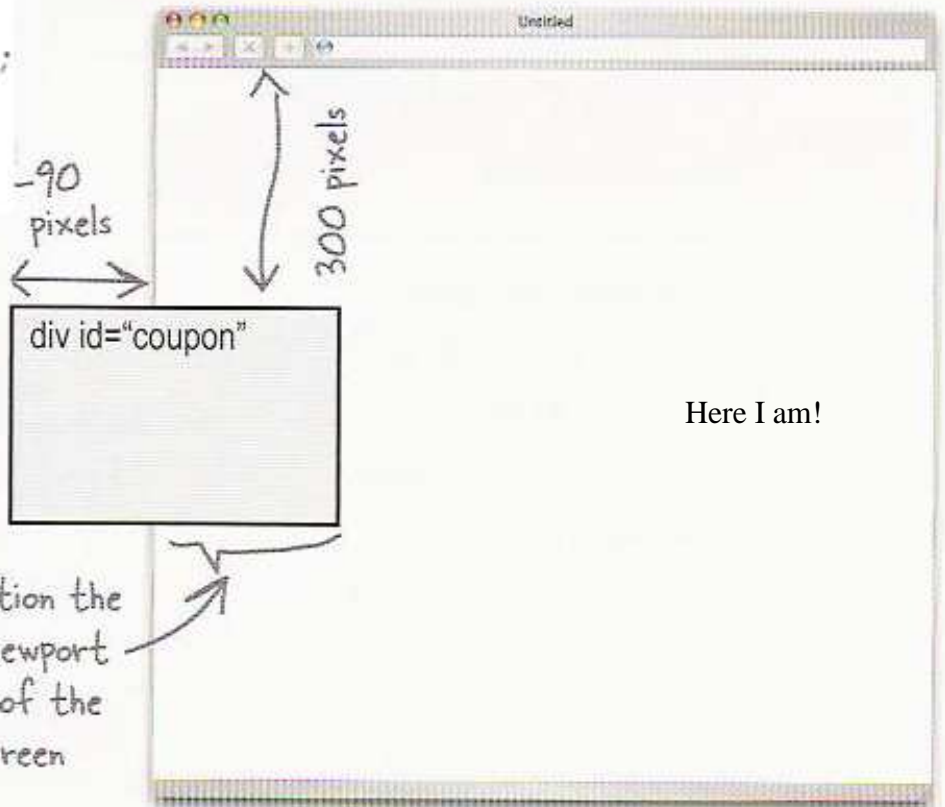


- removed from normal flow (like floating ones)
- positioned relative to the browser window

Negative corners

By specifying `-90 pixels`, we're telling the browser to position the image `90 pixels` to the left of the edge of the viewport.

```
#coupon {  
  position: fixed;  
  top: 300px;  
  left: -90px;  
}
```



The browser will gladly position the image to the left of the viewport for you, and only the part of the image that is still on the screen will be viewable.

- `left`, `right`, `top`, or `bottom` value can be negative to create an element that sits outside the visible browser window

Position vs. float vs. alignment

When trying to correctly position and lay out an element, use the following order:

1. if possible, solve the problem by *aligning* the element's content
 - horizontal alignment: `text-align`
 - set this on a block element, and it aligns the inline text content within a block element (but not the block element itself)
 - vertical alignment: `vertical-align`
 - set this on an inline element, and it aligns it vertically within the line it is on inside its containing block element
2. if alignment won't work, try *floating* the element
3. if floating won't work, try *positioning* the element
 - absolute/fixed positioning should be seen as a last resort and should not be abused

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

- 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
 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.
</span></p>
```

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

Common bug: space under image

```
<p style="background-color: red; padding: 0px; margin: 0px">
```

HTML

```

</p>
```

HTML



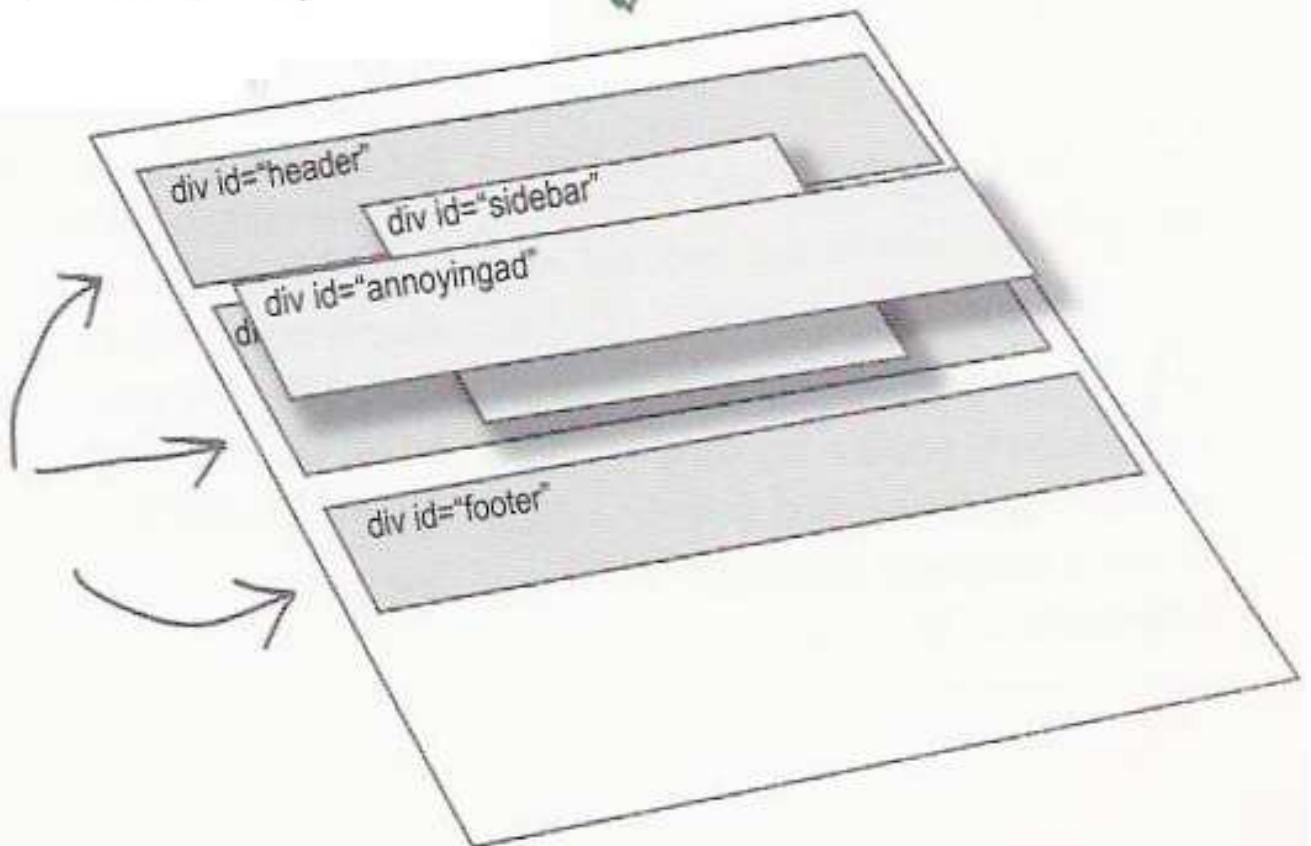
- red space under the image, despite padding and margin of 0
- this is because the image is vertically aligned to the baseline of the paragraph (not the same as the bottom)
- setting vertical-align to bottom fixes the problem (so does setting line-height to 0px)

The z-index property

```
#annoyingad {
  position: absolute;
  top: 150px;
  left: 100px;
  width: 400px;
  z-index: 1;
}
```

Here I am!

The sidebar and annoyingad <div>s are layered on the page, with the annoyingad having a greater z-index than the sidebar, so it's on top.



- sets which absolute positioned element will appear on top of another that occupies the same space
- higher z-index wins
- can be auto (default) or a number
- can be adjusted in DOM:
`object.style.zIndex = "value";`

The display property

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

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

- sets the type of CSS box model an element is displayed with
- can be none, inline, block, run-in, compact, ...
- use sparingly, because it can radically alter the page layout

Here I am!

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

- 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
```


- sets whether an element should be shown onscreen
 - the element 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 `visible` (default) or `hidden`
- can be used to show/hide dynamic HTML content on the page in response to events

Here I am!