

Lecture Notes 8:

Advanced CSS layout

CSE 190 M (Web Programming), Spring 2007
University of Washington

Reading: Sebesta Ch. 6 sections 6.1 - 6.4



The position property (examples)

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

Here I am!

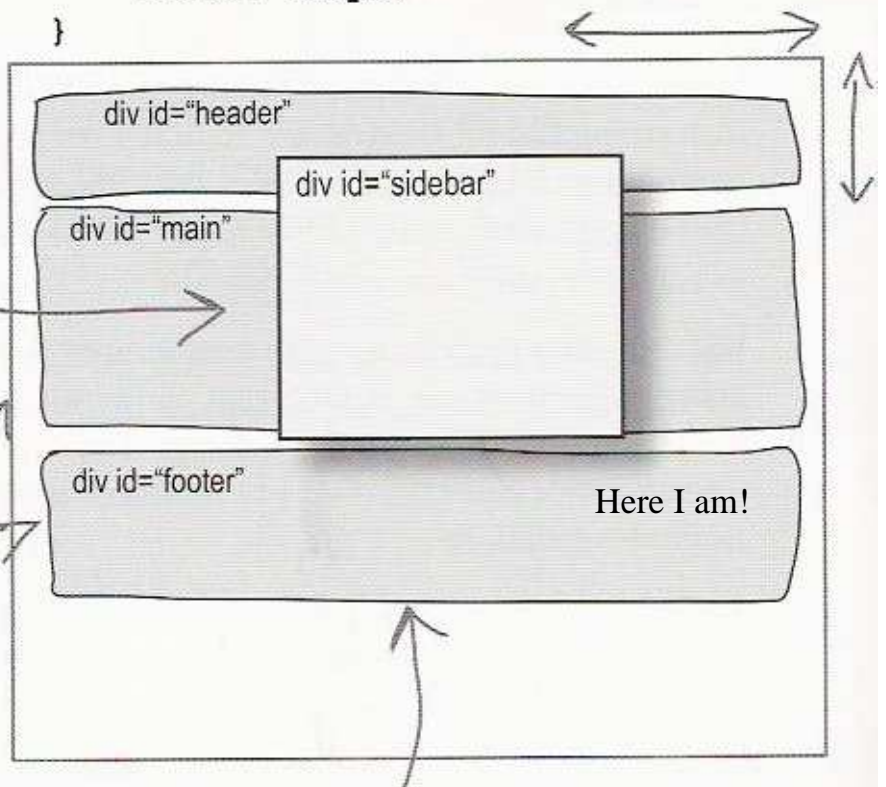
-
- `static` : default position
 - `relative` : offset from its normal static position
 - `absolute` : at a fixed position *within its containing element*
 - `fixed` : at a fixed position *within the browser window*
 - `top`, `bottom`, `left`, `right` properties specify positions of box's corners
-

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.

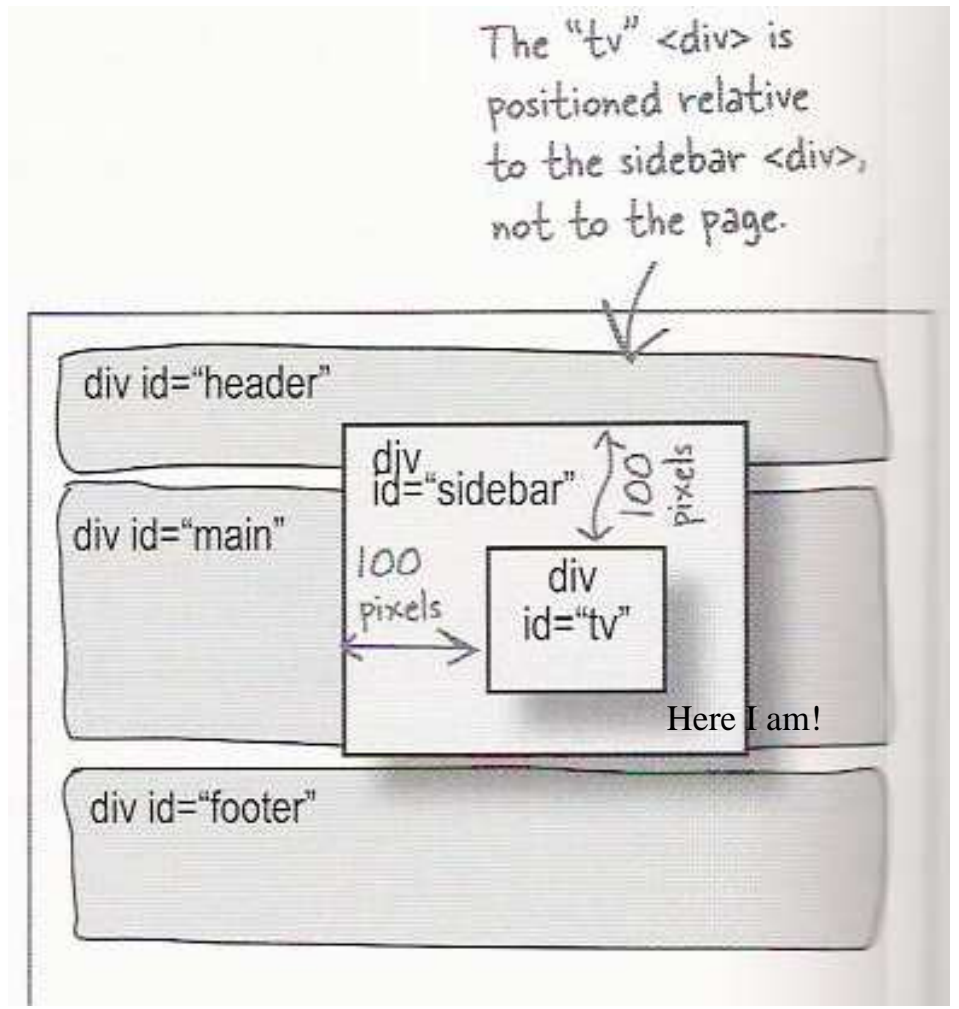


Elements that are in the flow don't even wrap their inline content around an absolutely positioned element. They are totally oblivious to it being on the page.

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

Absolute positioning details

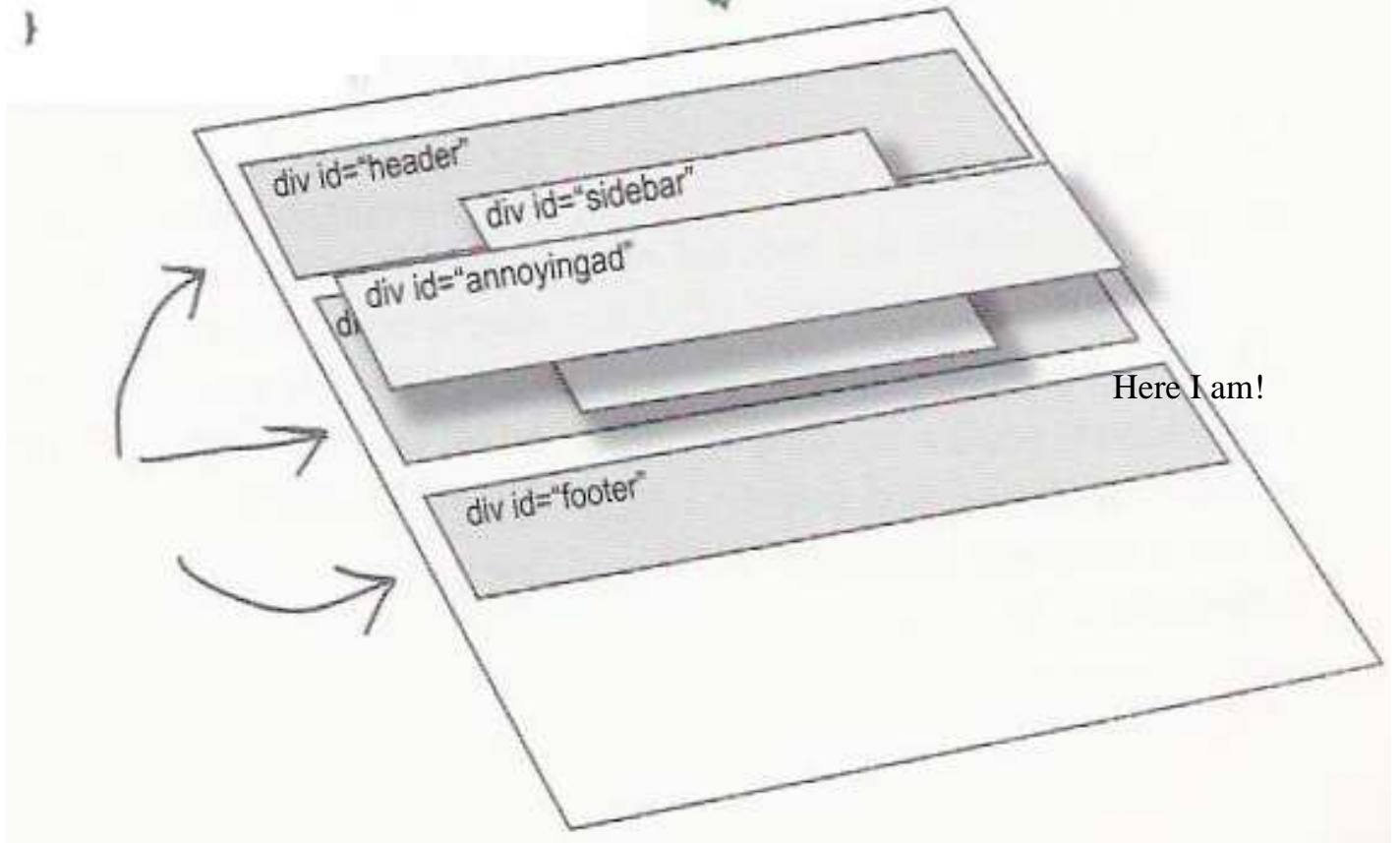
- positioned relative to the block element containing them



The z-index property

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

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.

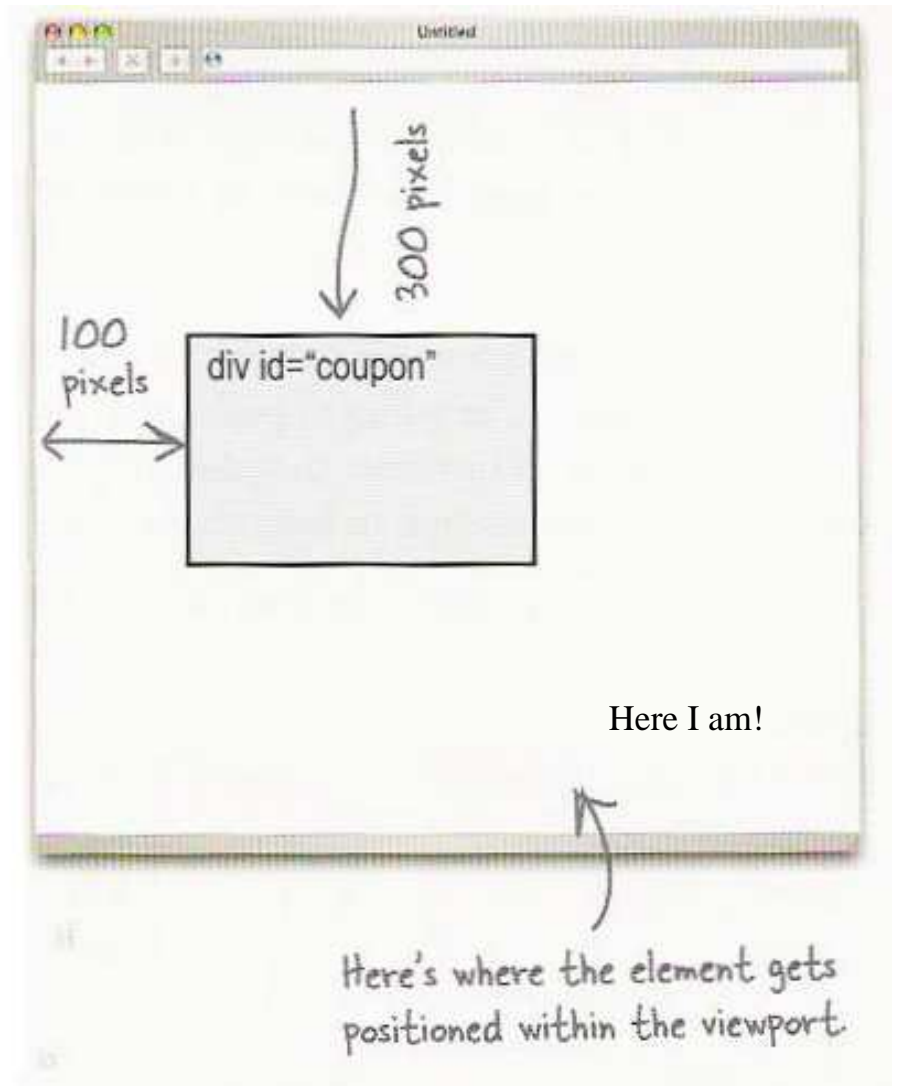


Here I am!

- 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";`

Fixed positioning

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

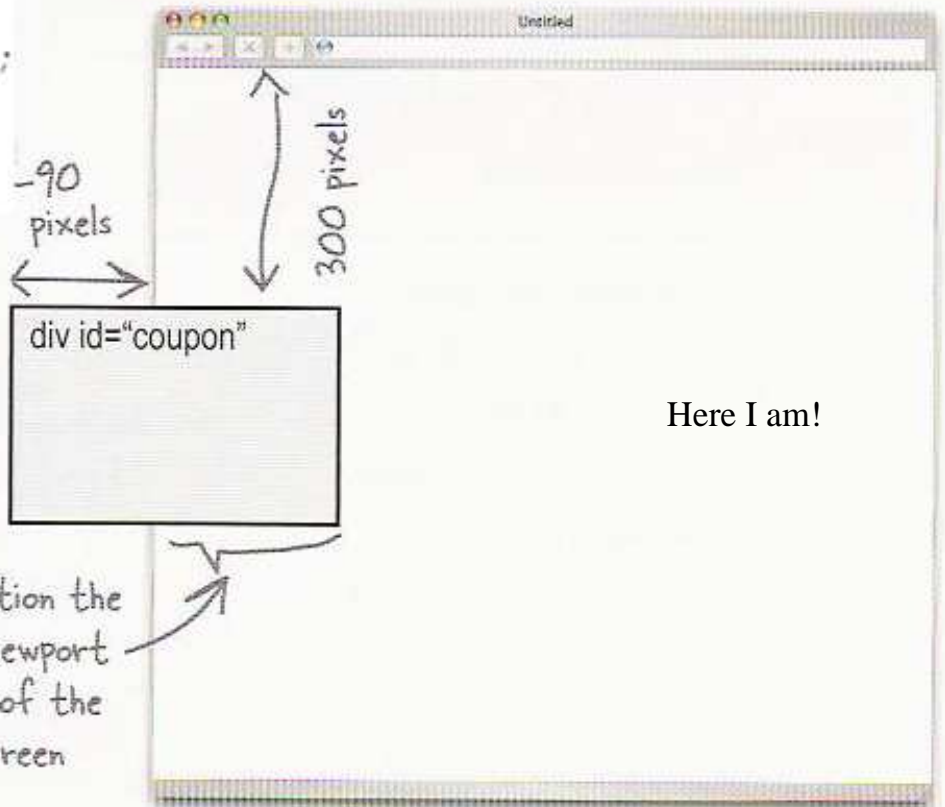


- removed from normal flow (like floating ones)
- positioned relative to the browser window

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

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`,

Type

text-top, text-bottom, or a length value or %
◦ baseline means aligned with bottom of non-hanging letters

- in DOM:
object.style.verticalAlign = "value";



vertical-align example


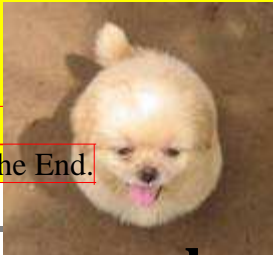
```
<p style="background-color: yellow;">
<span style="vertical-align: top; border: 1px solid red;">
Don't be sad! Turn that frown
 upside down!

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. Here I am!



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">

</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)

The display property

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

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
-

Practice problem: Smiley color

- Modify this example page ([HTML](#), [CSS](#)) that displays this [smiley face image](#) to have the following layout and behavior:
 - The smiley face is centered within the overall browser window and is half as tall and wide as the page area.
 - Underneath the smiley face are three checkboxes: Red, Green, and Blue. These controls are centered horizontally within the page and placed vertically at the very top of the page. Each checkbox appears on its own line.
 - When checked, each box adds that color to the background color behind the smiley face. When unchecked, that color is removed from the smiley's background color. Here I am!