Web Programming Step by Step

Lecture 5
Floating and Positioning
Reading: 4.3 - 4.5

4.3: Floating Elements

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

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

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

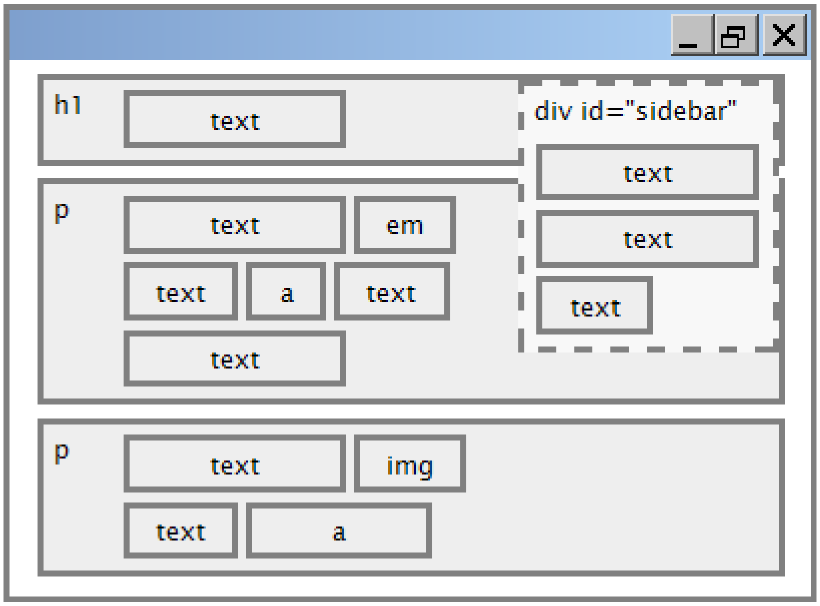
CSS

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)

```
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

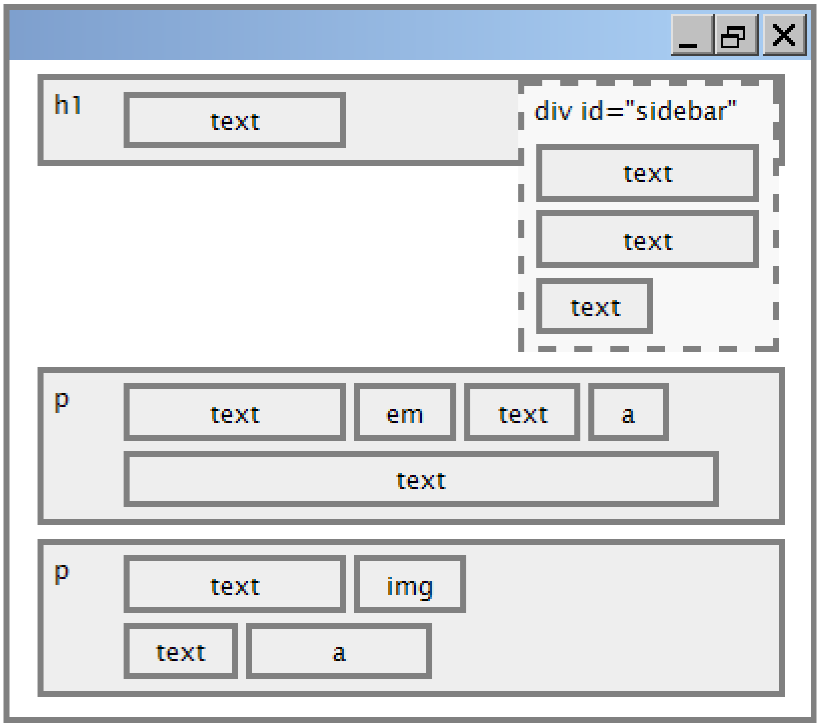
output

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

Clear diagram

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

CSS



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) (4.4.2)

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

CSS

Here I am!

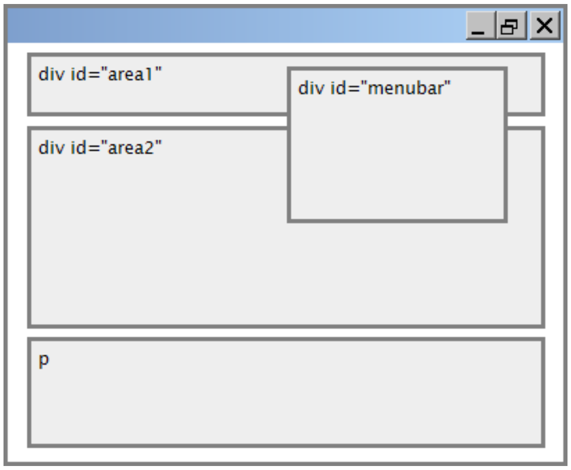
property	value	description
position	static	default position
	relative	offset from its normal static position
	absolute	a fixed position *within its containing element*
	fixed	a fixed position *within the browser window*
top, bottom, left, right	positions of box's corners	

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

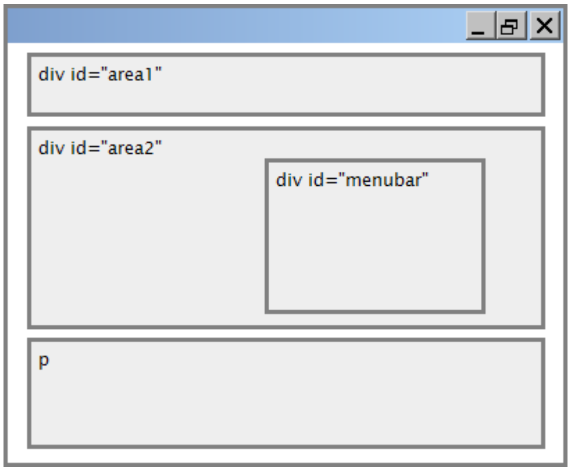


Relative positioning

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

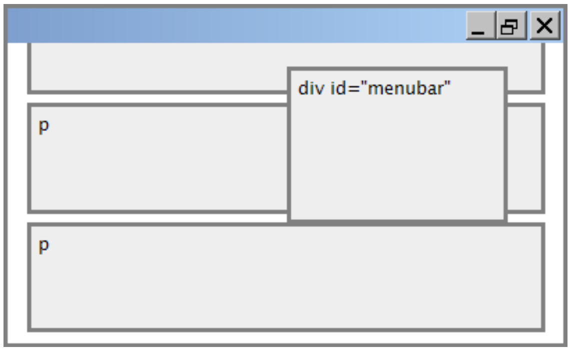
CSS

- 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

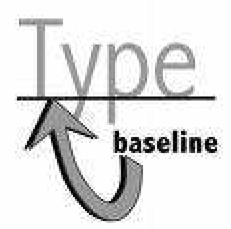
Details about inline boxes

- 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

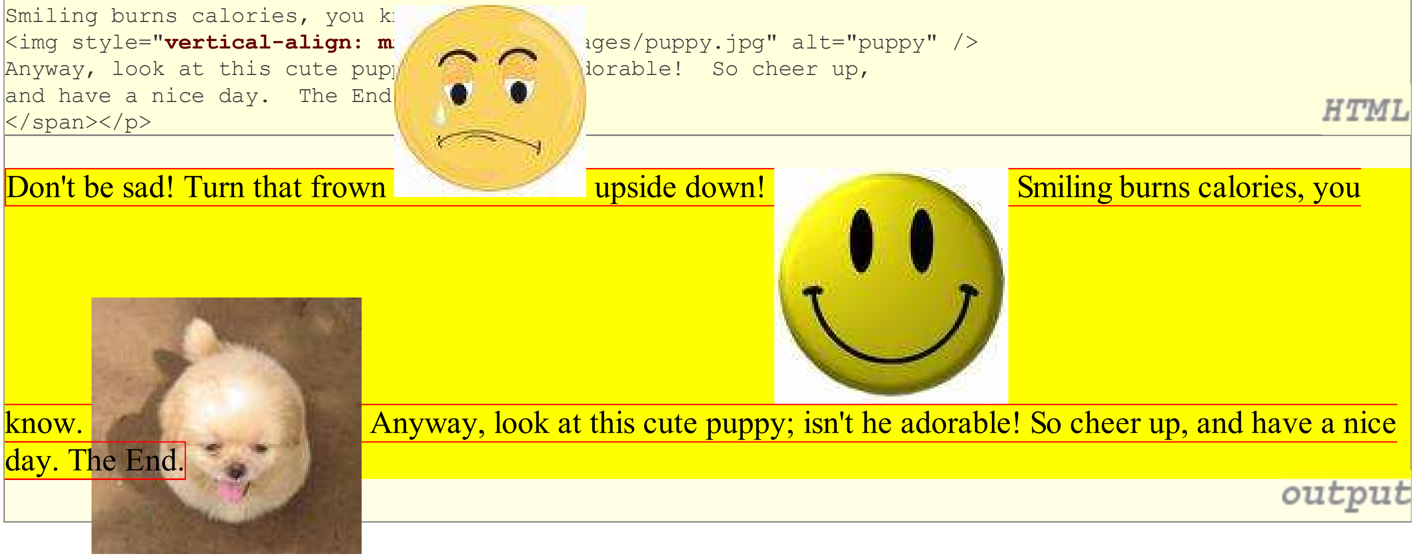
- can be `top`, `middle`, `bottom`, `baseline` (default), `sub`, `super`, `text-top`, `text-bottom`, or a length value or `%`
  - `baseline` means aligned with bottom of non-hanging letters



`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

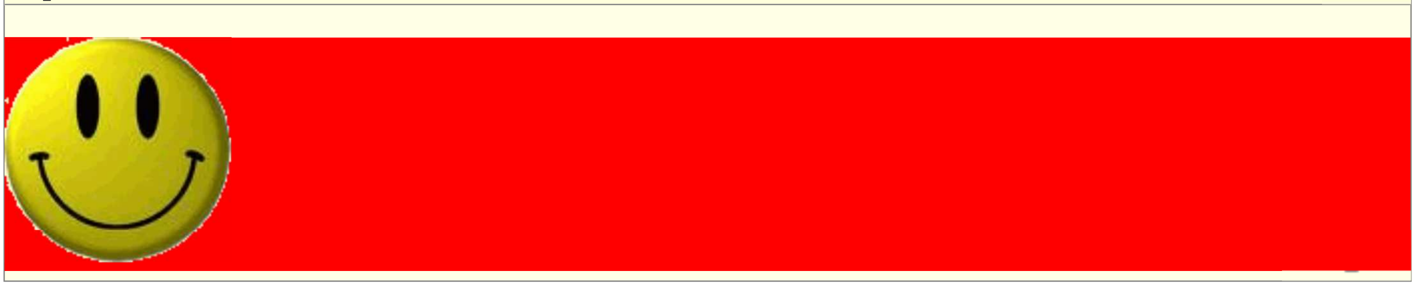


Don't be sad! Turn that frown upside down! Smiling burns calories, you know. Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.

output

Common bug: space under image

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



- 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`)