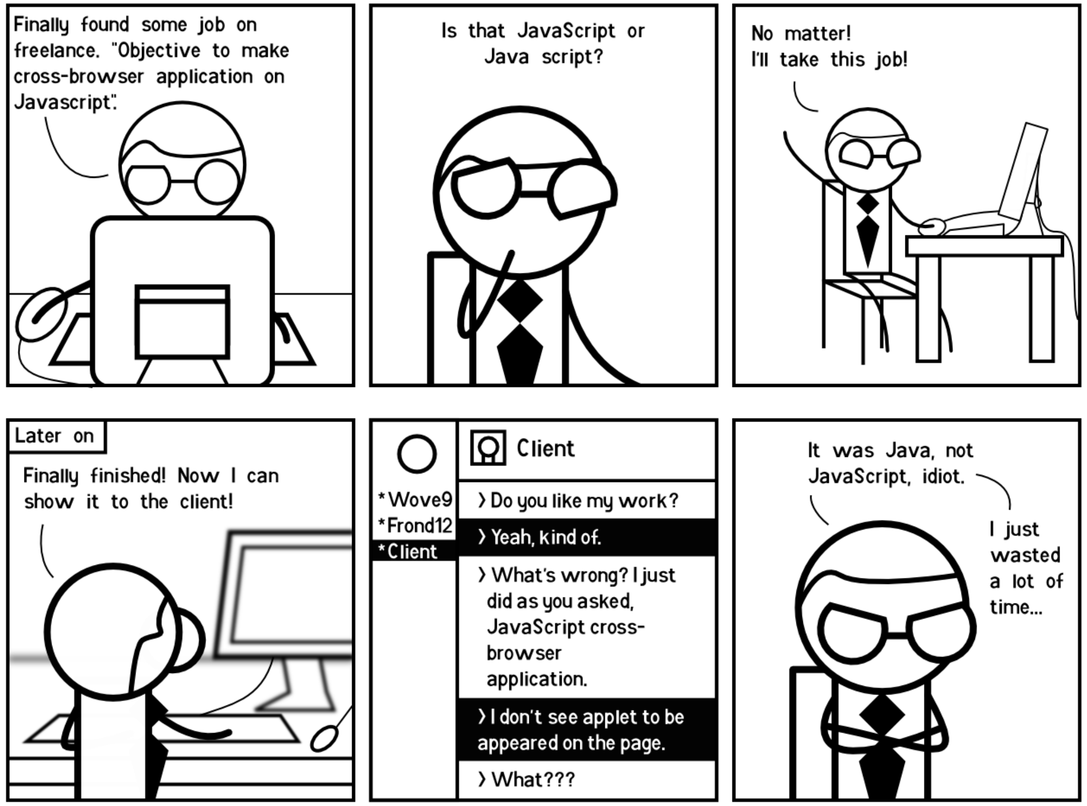
# CSc 337

LECTURE 5: GRID LAYOUT


Finally found some job on freelance. "Objective to make cross-browser application on Javascript".
Is that JavaScript or Java script?
No matter! I'll take this job!
Later on
Finally finished! Now I can show it to the client!
Client
*Wove9
*Frond12
*Client
› Do you like my work?
› Yeah, kind of.
› What's wrong? I just did as you asked, JavaScript cross-browser application.
› I don't see applet to be appeared on the page.
› What???
It was Java, not JavaScript, idiot.
I just wasted a lot of time...

## The position property

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

css

Here I am!

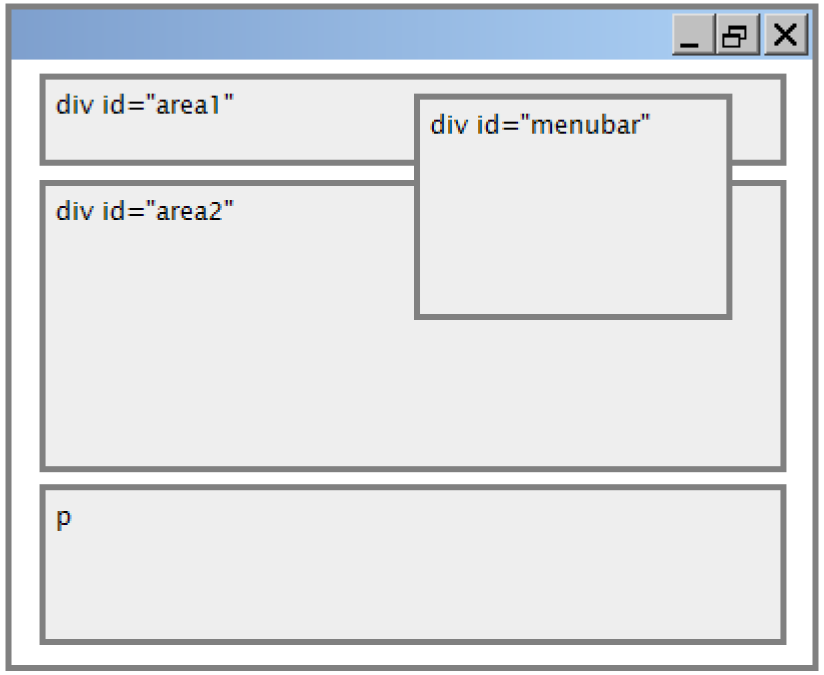
| property | value | description |
|---|---|---|
| position | static | default position |
| | relative | offset from its normal static position |
| | absolute | a fixed position within its containing element |
| | fixed | a fixed position within the browser window |
| top, bottom, left, right | positions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

## Absolute positioning

```css
#menubar {
      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

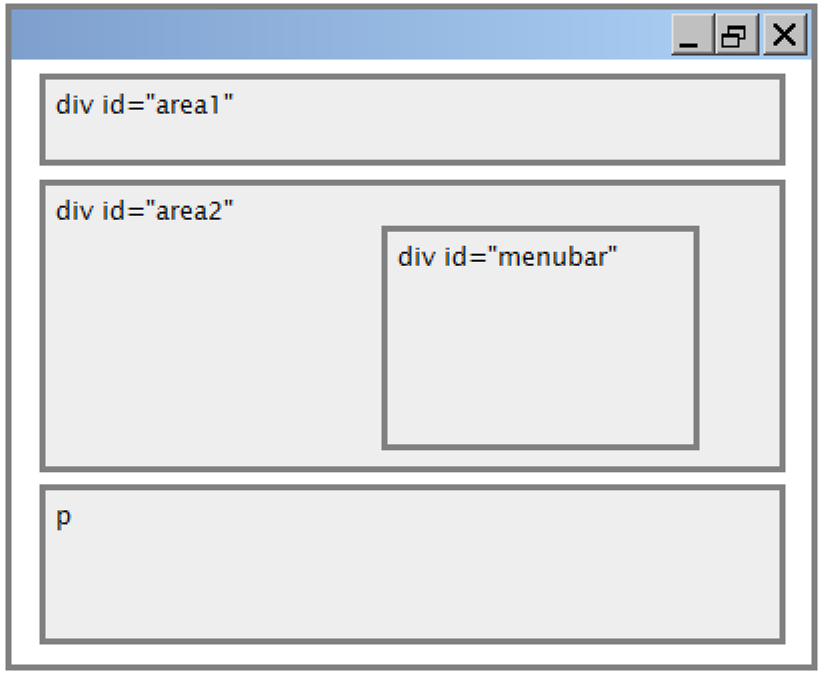


# Relative positioning

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

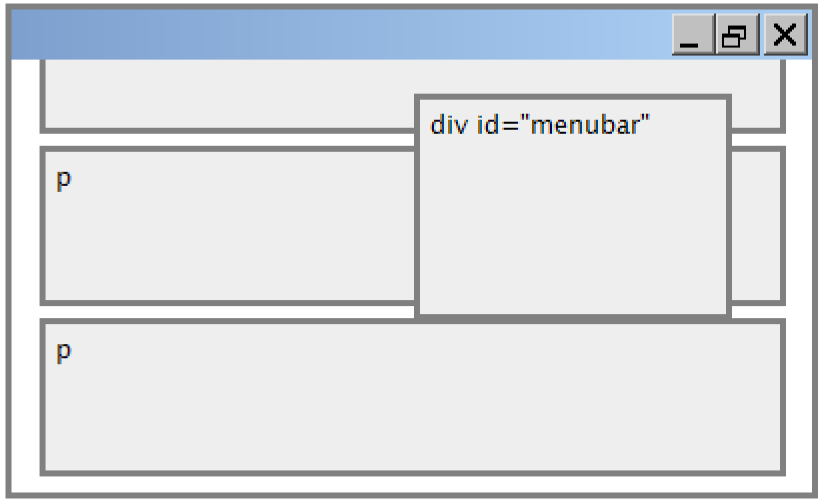
- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page

- 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



## Complex Layouts

**Flexbox** - designed for one-dimensional layouts

**Grid** - designed for two-dimensional layouts

## Grid Layout

Use if you want rows and columns

Works similarly to Flexbox
- outer container `display: grid`
- inner items end up in a grid

## Grid Layout

By default all items are in one column

to change the number of rows and columns specify the grid template in the **container** CSS:

```
        grid-template-rows: width width …
        grid-template-columns: width width …
```

width is the width of the column

write a width as many times as columns you want

## Grid Layout Example

Example:

```
.container {
    display: grid;
    grid-template-rows: 200px 200px 200px;
    grid-template-columns: 200px 200px 200px;
}
```

Creates a grid with three rows and three columns

## `fr` Unit

Calculates percentages of the container for you

Example:

```
grid-template-rows: 2fr 3fr
```

gives you **2 rows**

first takes up **2/5** of vertical space

second takes up **3/5** of vertical space

## Specifying many columns

Tedious to write out widths many times

`repeat` Example:

```
        grid-template-rows: repeat(3, 1fr);
```

Creates 3 rows of equal height

## Template shorthand

Create a grid template in one line:

```
grid-template: rows / columns;
```

Example:

```
grid-template: repeat(3, 1fr) / repeat(3, 1fr);
```

## Alignment vs. float vs flexbox vs grid 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 and you want a one-dimensional layout try *flexbox*
3. if alignment won't work and you want a two-dimensional layout try grid
4. if flexbox and grid won't work, try *floating*
5. if floating won't work, try *positioning* the element
   - absolute/fixed positioning are a last resort and should not be overused

## The display property

```css
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 very sparingly**, because it can radically alter the page layout

# Displaying block elements as inline

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

HTML

```css
#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)

## 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

## Exercise - Boxes

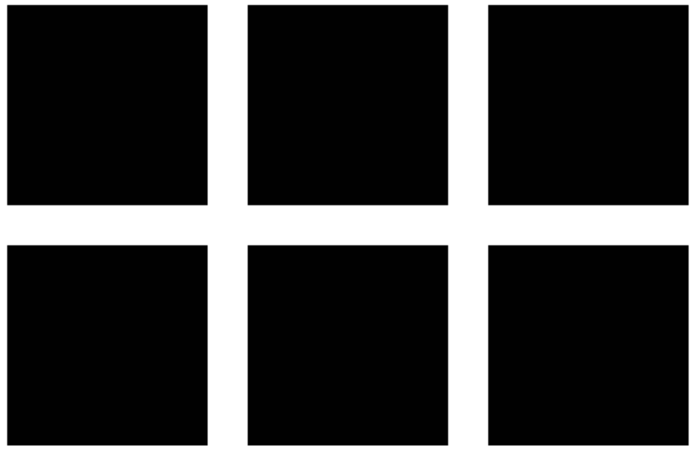
Generate the appearance on the next slide, starting from this HTML and CSS code.

```
<div id="outer">
  <div class="box"></div>
  <div class="box"></div>
  <div class="box"></div>
  <div class="box"></div>
  <div class="box"></div>
  <div class="box"></div>
</div>
```

```
#outer {
  border: 2px dashed black;
  padding: 10px;
}

.box {
  width: 100px;
  height: 100px;
  background-color: black;
  margin: 10px;
}
```

## Exercise - Boxes

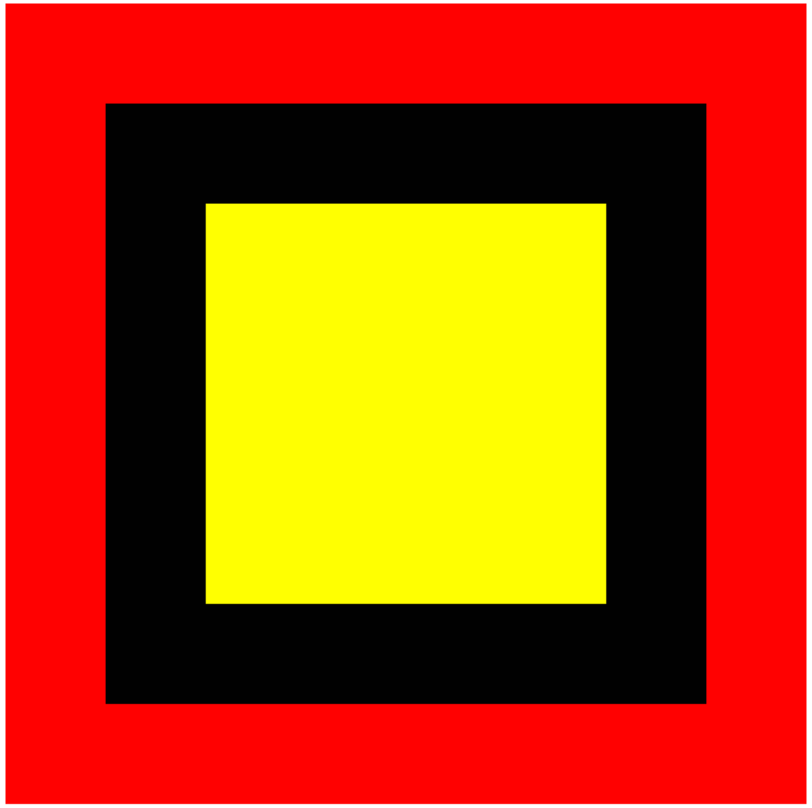


## Exercise - nested boxes

Given the code below, write boxes.css to make the appearance on the next slide.

```
<!DOCTYPE html>
<html>
  <head>
    <link href="boxes.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
  <div id="outer-box">
    <div id="inner-box"></div>
  </div>
  </body>
</html>
```

## Exercise - nested boxes



The outer border of the box is red, the inner border of the box is black, and the inner background color of the box is yellow.

Both the outer and inner borders have a width of 50 pixels. The yellow portion of the box has a width and height of 200 pixels. The overall box has a width and height of 400 pixels.

## Combination layouts

Most pages use many different layouts

Combine and **nest** as needed.

Example:

https://www.cs.arizona.edu/

## Layout Practice

Games to practice your CSS layout skills:

- FlexBox: https://flexboxfroggy.com/
- Grid: https://cssgridgarden.com/