

# SWE 432 -Web Application Development

Fall 2022

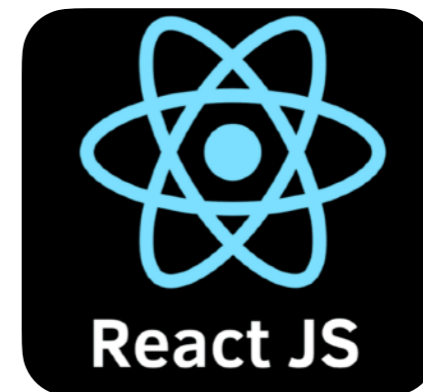


George Mason  
University

---

Dr. Kevin Moran

*Week 10:*  
CSS &  
DOM





# Administrivia

- *HW Assignment 3* - Due Thursday, grades and comments will be posted by Thursday next week.
- *HW Assignment 4* - Out Thursday, Due in three weeks (November 17th)
  - Extra Credit Opportunity!
- *Mid-Semester Course Feedback Survey*: Please fill out the survey!



# Class Overview

- *Part 2:* CSS & DOM
  - Brief Lecture
  - Hands-On Session

# CSS + DOM





# CSS History

- **1994:** Cascading HTML style sheets—a proposal
  - Hakon W Lie proposes CSS
  - Working w/ Tim-Berners Lee at CERN
- **1996:** CSS1 standard, recommended by W3C
  - Defines basic styling elements like font, color, alignment, margin, padding, etc.
- **1998:** CSS2 standard, recommended by W3C
  - Adds positioning schemes, z-index, new font properties
- **2011:** CSS3 standards divided into modules, begin adoption
  - Add more powerful selectors, more powerful attributes

<https://dev.opera.com/articles/css-twenty-years-hakon/>

[https://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets#History](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#History)



# CSS Tutorials and Reference

<https://developer.mozilla.org/en-US/docs/Web/CSS>

MDN web docs  
Technologies ▾ References & Guides ▾ Feedback ▾ Search MDN Sign in

## CSS: Cascading Style Sheets

Web technology for developers > CSS: Cascading Style Sheets English ▾

**Related Topics**  
[CSS](#)  
[CSS Reference](#)

**Cascading Style Sheets (CSS)** is a [stylesheet](#) language used to describe the presentation of a document written in [HTML](#) or [XML](#) (including XML dialects such as [SVG](#), [MathML](#) or [XHTML](#)). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is one of the core languages of the **open Web** and is standardized across Web browsers according to the [W3C specification](#). Developed in levels, CSS1 is now obsolete, CSS2.1 is a recommendation, and [CSS3](#), now split into smaller modules, is progressing on the standardization track.

### CSS Introduction

If you're new to web development, be sure to read our [CSS basics](#) article to learn what CSS is and how to use it.

### CSS Tutorials

Our [CSS learning area](#) contains a wealth of tutorials to take you from beginner level to proficiency, covering all the fundamentals.

### CSS Reference

Our [exhaustive CSS reference](#) for seasoned Web developers describes every property and concept of CSS.

## Tutorials

Our [CSS Learning Area](#) features multiple modules that teach CSS from the ground up — no previous knowledge required.

### Introduction to CSS

This module starts with the basics of how CSS works, including selectors and properties, writing CSS rules, applying CSS to HTML, how to specify length, color, and other units in CSS, cascade and inheritance, box model basics, and debugging CSS.

### Styling text

This module discusses text styling fundamentals, including setting

## Reference

- [CSS reference](#): This exhaustive reference for seasoned Web developers describes every property and concept of CSS.
- CSS key concepts:
  - [The syntax and forms of the language](#)
  - [Specificity, inheritance and the Cascade](#)
  - [CSS units and values](#)
  - [Box model and margin collapse](#)
  - [The containing block](#)
  - [Stacking and block-formatting contexts](#)

# CSS: Cascading Style Sheets

- Language for styling documents

```
p {  
  font-family: Arial;  
}
```

Property Value



“Select all <p> elements”

Selector describes a *set* of HTML elements

“Use Arial font family”

Declaration indicates how selected elements should be styled.

- Separates *visual presentation* (CSS) from document structure (HTML)
  - Enables changes to one or the other.
  - Enables styles to be reused across sets of elements.

# CSS Styling

## Events [\(Details\)](#) [\(Calendar\)](#)

**Oral Defense of Doctoral Dissertation:  
Energy Management in Performance-  
Sensitive Wireless Sensor Networks**

Friday, September 09, 2016, 1:00-  
2:00pm, ENGR 4801  
Maryam Bandari

## News [\(Details\)](#)

dt | 612 x 40

**Prof. Zoran Duric appointed as Deputy Editor of journal Pattern Recognition [\(more\)](#)**

Prof. Zoran Duric has been appointed the Deputy Editor of the Elsevier journal [Pattern Recognition](#) for a three year term starting August 1, 2016.

**Professor Jim Chen appointed as Editor-in-Chief of the journal Computing in Science & Engineering [\(more\)](#)**

Professor Jim Chen's term as Editor in Chief of the journal [Computing in Science & Engineering \(CiSE\)](#) will commence on January 1 2017. Professor Chen has been on the editorial board of CiSE since 1999.

- Invisible box around every element.
- Rules control how sets of boxes and their contents are presented

## Example Styles

### BOXES

Width, height

Borders (color, width, style)

Position in the browser window

### TEXT

Typeface

Size, color

Italics, bold, lowercase





# Using CSS

## External CSS

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="main.css">
  <title>Prof Bell's Webpage</title>
</head>
```

## Internal CSS

```
<!DOCTYPE html>
<html>
<head>
  <title>Prof Bell's Webpage</title>
  <style type="text/css">
    body {
      background-image: url("bluerock.jpg");
      font-family: Comic Sans MS, Comic Sans;
      color: #FFFF00;
    }
  </style>
```

- External CSS enables stylesheets to be reused across multiple files
- Can include CSS files
- Can nest CSS files
  - @import url("file.css") imports a CSS file in a CSS file

# CSS Type Selectors

- What if we wanted more green?

## Prof Kevin Moran



This is Prof Moran's *ACTUAL* homepage from 19991

**Welcome, students!**

[See how to make this page](#)

**Some funny links**

- [Homestar Runner](#)
- [Hamster Dance](#)

**About Prof Moran**

Prof Moran's office is at 4442 Engineering Building. His email address is [kpmoran@gmu.edu](mailto:kpmoran@gmu.edu).

Last updated: September 28th, 1999



# CSS Type Selectors

- What if we wanted more green?

```
h2, h3 {  
  color: LightGreen;  
}
```

“Select all <h2> and <h3> elements”

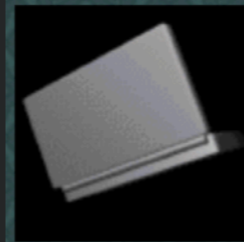
Type selector selects one or more element types.

```
* {  
  color: LightGreen;  
}
```

“Select all elements”

Universal selector selects all elements.

## Prof Kevin Moran



This is Prof Moran's ACTUAL homepage from 1991

### Welcome, students!

[See how to make this page](#)

### Some funny links

- [Homestar Runner](#)
- [Hamster Dance](#)

### About Prof Moran

Prof Moran's office is at 4442 Engineering Building. His email address is [kpmoran@gmu.edu](mailto:kpmoran@gmu.edu).

Last updated: September 28th, 1999

# CSS Class Selectors

```

```

“Label <img> element with imageLarge class”

```
.imageLarge {  
  width: 200px;  
  height: 200px;  
}
```

“Define class imageLarge.”

```

```

```
img.large {  
  width: 200px;  
  height: 200px;  
}
```

“Define large class that applies only to <img> elements”

```
.transparent {  
  opacity: .50;  
}
```

“Define transparent class”

Classes enable the creation of sets of elements that can be styled in the same way.

# CSS id Selectors

```
<div id="exampleElem">
  Some text
</div>

#exampleElem {
  font-weight: bold;
}
```

**Some text**

- Advantages

- Control presentation of individual elements

- Disadvantages

- Must write separate rule for *each* element



# Additional Selector Types

Selector	Meaning		Example
<b><i>Descendant selector</i></b>	Matches all descendants of an element	<code>p a { }</code>	Select <code>&lt;a&gt;</code> elements inside <code>&lt;p&gt;</code> elements
<b><i>Child selector</i></b>	Matches a direct child of an element	<code>h1&gt;a { }</code>	Select <code>&lt;a&gt;</code> elements that are directly contained by <code>&lt;h1&gt;</code> elements.
<b><i>First child selector</i></b>	Matches the first child of an element	<code>h1:first-child { }</code>	Select the the elements that are the first child of a <code>&lt;h1&gt;</code> element.
<b><i>Adjacent selector</i></b>	Matches selector	<code>h1+p { }</code>	Selects the first <code>&lt;p&gt;</code> element after any <code>&lt;h1&gt;</code> element
<b><i>Negation selector</i></b>	Selects all elements that are not selected.	<code>body *:not(p)</code>	Select all elements in the body that are not <code>&lt;p&gt;</code> elements.
<b><i>Attribute selector</i></b>	Selects all elements that define a specific attribute.	<code>input[invalid]</code>	Select all <code>&lt;input&gt;</code> elements that have the <code>invalid</code> attribute.
<b><i>Equality attribute selector</i></b>	Select all elements with a specific attribute value	<code>p[class="invisible"]</code>	Select all <code>&lt;p&gt;</code> elements that have the <code>invisible</code> class.



# CSS Selectors

- **Key principles in designing effective styling rules:**
  - Use classes, semantic tags to create sets of elements that share a similar rules
  - Don't repeat yourself (DRY)
    - Rather than create many identical or similar rules, apply single rule to all similar elements
  - Match based on semantic properties, not styling
    - Matching elements based on their pre-existing styling is *fragile*



# Cascading Selectors

- What happens if more than one rule applies?
- Most *specific* rule takes precedence
  - *p b* is more specific than *p*
  - *#maximizeButton* is more specific than *button*
- If otherwise the same, *last* rule wins
- Enables writing generic rules that apply to many elements that are overridden by specific rules applying to a few elements





# CSS Inheritance

- When an element is contained inside another element, some styling properties are inherited
  - e.g., font-family, color
- Some properties are not inherited
  - e.g., background-color, border
- Can force many properties to inherit value from parent using the inherit value
  - e.g., padding: inherit;

# Pseudo Classes

```
.invisible {  
  display: none;  
}
```

```
input:invalid {  
  border: 2px solid red;  
}
```

```
input:invalid + div {  
  display: block;  
}
```

```
input:focus + div {  
  display: none;  
}
```

```
<label>  
  Email: <input type="email" />  
  <div class="invisible">Please enter a valid email.</div>  
</label>
```

Email:

“Select elements with the invalid attribute.”

“Select elements that have focus.”

Classes that are automatically attached to elements based on their attributes.



# Examples of Pseudo Classes

- :active - elements activated by user. For mouse clicks, occurs between mouse down and mouse up.
- :checked - radio, checkbox, option elements that are checked by user
- :disabled - elements that can't receive focus
- :empty - elements with no children
- :focus - element that currently has the focus
- :hover - elements that are currently hovered over by mouse
- :invalid - elements that are currently invalid
- :link - link element that has not yet been visited
- :visited - link element that has been visited

# Color

- Can set text color (`color`) and background color (`background-color`)
- Several ways to describe color
  - six digit hex code (e.g., `#ee3e80`)
  - color names: 147 predefined names
  - `rgb(red, green, blue)`: amount of red, green, and blue
  - `hsla(hue, saturation, lightness, alpha)`: alternative scheme for describing colors
- Can set opacity (`opacity`) from 0.0 to 1.0

```
body {  
    color: Red;  
    background-color: rgb(200, 200, 200); }  
h1 {  
    background-color: DarkCyan; }  
h2 {  
    color: #ee3e80; }  
p {  
    color: hsla(0, 100%, 100%, 0.5); }  
div.overlay {  
    opacity: 0.5; }
```

# Typefaces



**im**

Serif



**im**

Sans-Serif



**im**

Monospace



**im**

Cursive

font-family: Georgia, Times, serif;

“Use Georgia if available, otherwise Times, otherwise any serif font”.

font-family enables the typeface to be specified. The typeface must be installed. Lists of fonts enable a browser to select an alternative.

# Styling text

```
h2 {  
  text-transform: uppercase;  
  text-decoration: underline;  
  letter-spacing: 0.2em;  
  text-align: center;  
  line-height: 2em;  
  vertical-align: middle;  
  text-shadow: 1px 1px 0 #666666;  
}
```

**THIS TEXT IS IMPORTANT**

- text-transform: uppercase, lowercase, capitalize
- text-decoration: none, underline, overline, line-through, blink
- letter-spacing: space between letters (kerning)
- text-align: left, right, center, justify
- line-height: total of font height and empty space between lines
- vertical-align: top, middle, bottom, ...
- text-shadow: [x offset][y offset][blur offset][color]

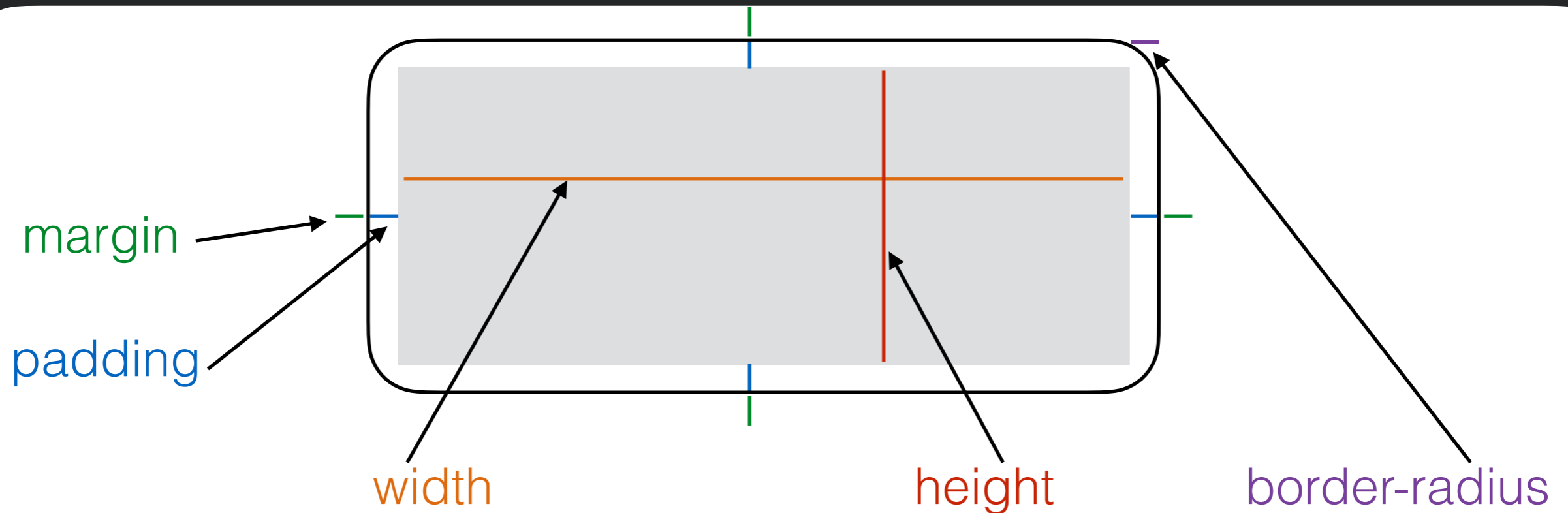
```
<a class="movableItem">Walt Whitman</a>
```

Walt hitman

```
a.movableItem {  
  cursor: move;  
}
```

- Can change the default cursor with cursor attribute
  - auto, crosshair, pointer, move, text, wait, help, url("cursor.gif")
- Should *only* do this if action being taken clearly matches cursor type

# CSS "Box" Model



- Boxes, by default, are sized *just* large enough to fit their contents.
- Can specify sizes using `px` or `%`
  - `%` values are relative to the container dimensions
- `margin: 10px 5px 10px 5px;` (clockwise order - [top] [right] [bottom] [left])
- `border: 3px dotted #0088dd;` ([width] [style] [color])
  - style may be: solid, dotted, dashed, double, groove, ridge, inset, outset, hidden / none



# Centering Content

```
.centered {  
  width: 300px;  
  margin: 10px auto 10px auto;  
  border: 2px solid #0088dd;  
}
```

This box is centered in its container.

- How do you center an element inside a container?
- Step 1: Must first ensure that element is *narrower* than container.
  - By default, element will expand to fill entire container.
  - So must usually explicitly set width for element.
- Step 2: Use *auto* value for left and right to create equal gaps

# Visibility and layout

- Can force elements to be inline or block element.
  - display: inline
  - display: block
- Can cause element to not be laid out or take up any space
  - display: none
  - *Very* useful for content that is dynamically added and removed.
- Can cause boxes to be invisible, but still take up space
  - visibility: hidden;

```
<ul>
  <li>Home</li>
  <li>Products</li>
  <li class="coming-soon">Services</li>
  <li>About</li>
  <li>Contact</li>
</ul>
```

```
li {
  display: inline;
  margin-right: 10px; }
li.coming-soon {
  display: none; }
```

Home Products About Contact

```
li {
  display: inline;
  margin-right: 10px; }
li.coming-soon {
  visibility: hidden; }
```

Home Products About Contact



# Positioning Schemes

## Normal flow (default)

**Lorem Ipsum**

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

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Block level elements appear on a new line. Even if there is space, boxes will not appear next to each other.

## Relative positioning

**Lorem Ipsum**

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

Ut enim ad minim veniam, quis nostrud exercitation u nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
p.example {
  position: relative;
  top: 10px;
  left: 100px;
}
```

Element shifted from normal flow. Position of other elements is *not* affected.

## Absolute positioning

eiusmod tempor incididunt ut labore et dolore magna **Lorem Ipsum**

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
h3 {
  position: absolute;
  background-color: LightGray;
  left: 350px;
  width: 250px;
}
```

Element taken out of normal flow and does not affect position of other elements. Moves as user scrolls.

## Fixed positioning

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

**Lorem Ipsum**

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
h3 {
  position: fixed;
  background-color: LightGray;
  left: 40px;
  width: 250px;
}
```

Element taken out of normal flow and does not affect position of other elements. Fixed in window position as user scrolls.

## Floating elements

**Lorem Ipsum**

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
h3 {
  float: left;
  background-color: LightGray;
  left: 40px;
  width: 250px;
}
```

Element taken out of normal flow and position to far left or right of container. Element becomes block element that others flow around.

# Stacking elements

```
h3 {  
  position: absolute;  
  background: LightGray;  
  opacity: 0.6;  
  z-index: 10;  
}
```

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do  
~~>Lorem ipsum~~ incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris  
nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum  
dolore eu fugiat nulla pariatur.

- Elements taken out of normal flow may be stacked on top of each other
- Can set order with z-index property
  - Higher numbers appear in front
- Can set opacity of element, making occluded elements partially visible

# Transform - examples

```
.box {  
  width: 100px;  
  height: 100px;  
  color: White;  
  text-align: center;  
  background-color: #0000FF;  
}
```



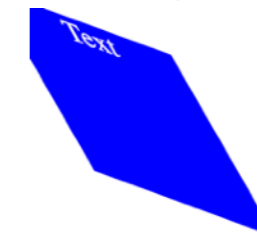
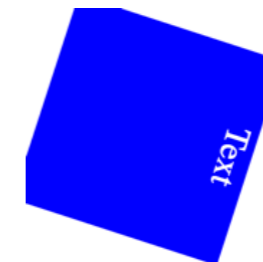
```
.transform1 {  
  transform: translate(12px, 50%);  
}
```

```
.transform2 {  
  transform: scale(2, 0.5);  
}
```

```
.transform3 {  
  transform: rotate(0.3turn);  
}
```

```
.transform4 {  
  transform: skew(30deg, 20deg);  
}
```

```
<div class="box">Text</div>
```



- Can modify coordinate space of element to rotate, skew, distort

# Transitions

```
.box {  
  width: 100px;  
  height: 100px;  
  background-color: #0000FF;  
  transition: width 2s, height 2s, background-color 2s, transform 2s;  
}
```



```
.box:hover {  
  background-color: #FFCCCC;  
  width: 200px;  
  height: 200px;  
  transform: rotate(180deg);  
}
```

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

- transition: [property time], ..., [property time]
  - When new class is applied, specifies the time it will take for each property to change
  - Can use *all* to select all changed properties



# Transition: Example

<https://jsfiddle.net/vs2qo9r1/>

```
1  .parent {
2    width: 250px;
3    height: 125px;
4  }
5
6  .box {
7    width: 100px;
8    height: 100px;
9    background-color: red;
0    font-size: 20px;
1    left: 0px;
2    top: 0px;
3    position: absolute;
4    -webkit-transition-property: width height background-color font-size left top color;
5    -webkit-transition-duration: 2s;
6    -webkit-transition-delay: 1s;
7    -webkit-transition-timing-function: linear;
8    transition-property: width height background-color font-size left top color;
9    transition-duration: 2s;
0    transition-delay: 1s;
1    transition-timing-function: linear;
2  }
3
4  .box1{
5    width: 50px;
6    height: 50px;
7    background-color: blue;
8    color: yellow;
9    font-size: 18px;
0    left: 150px;
1    top: 25px;
2    position: absolute;
3    -webkit-transition-property: width height background-color font-size left top color;
4    -webkit-transition-duration: 2s;
5    -webkit-transition-delay: 1s;
6    -webkit-transition-timing-function: linear;
7    transition-property: width height background-color font-size left top color;
8    transition-duration: 2s;
9    transition-delay: 1s;
0    transition-timing-function: linear;
1  }
2
```

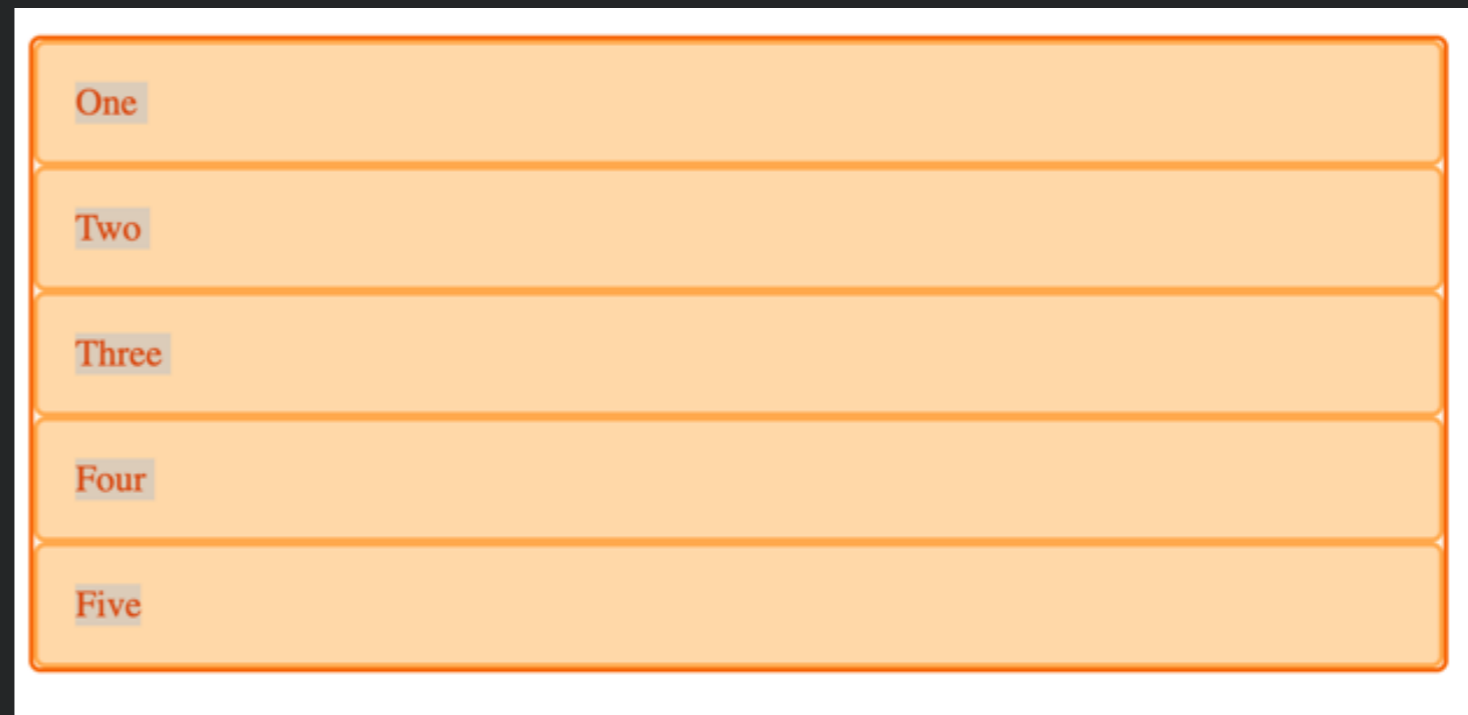


# Grid layout

- Create using display: grid or display: inline-grid

```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

```
.wrapper {  
  display: grid;  
}
```





# Grid tracks

- Define rows and columns on grid with the *grid-template-columns* and *grid-template-rows* properties.
- Define grid tracks.
- A grid track is the space between any two lines on the grid.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

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



# Liquid layouts

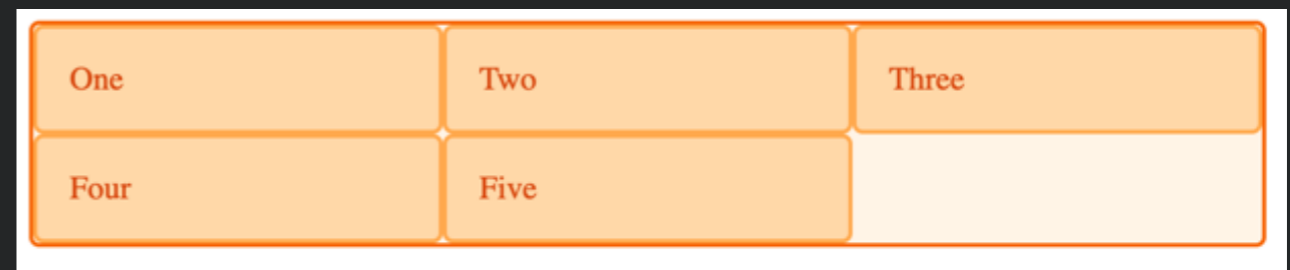
- `fr` represents a fraction of available space for grid container.
- Can mix absolute and flexible, where flexible occupies any remaining space after flexible is subtracted

```

<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}

```



```

.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}

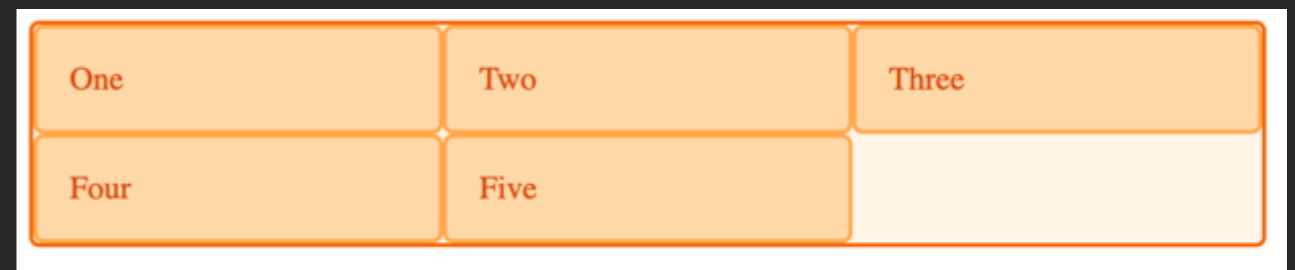
```

# Liquid layouts

- `fr` represents a fraction of available space for grid container.
- Can mix absolute and flexible, where flexible occupies any remaining space after flexible is subtracted

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```



```
.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```

# Positioning items

- Can explicitly place elements inside grid into grid areas

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
  <div class="box4">Four</div>
  <div class="box5">Five</div>
</div>
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```



- Can set gaps between columns and rows

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  column-gap: 10px;
  row-gap: 1em;
}
```

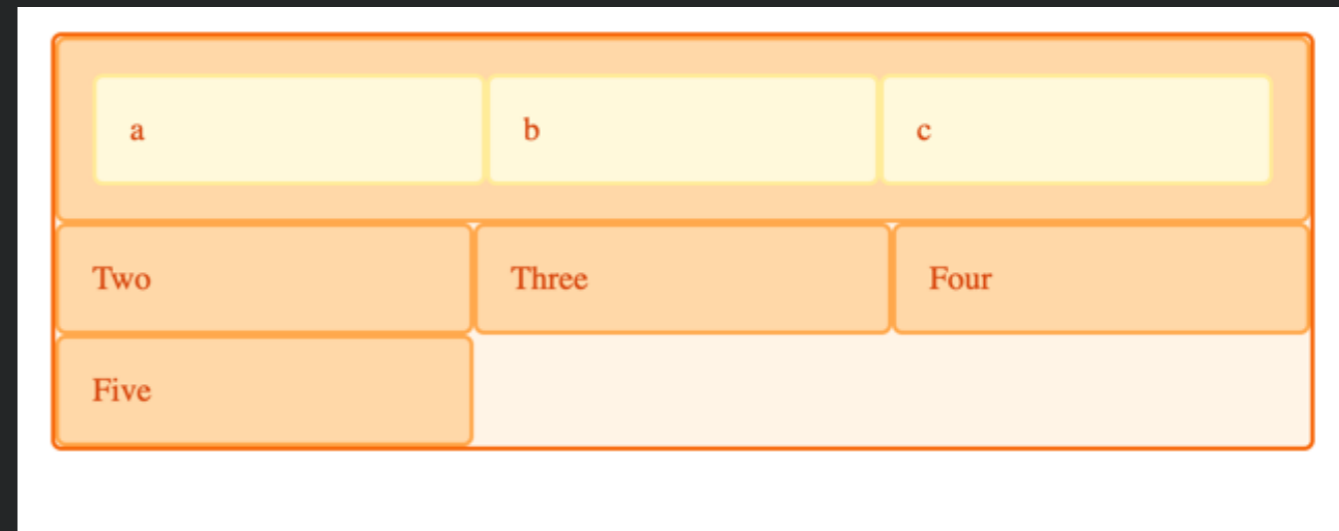


# Nesting

- Can nest grids, which behave just like top-level

```
<div class="wrapper">
  <div class="box box1">
    <div class="nested">a</div>
    <div class="nested">b</div>
    <div class="nested">c</div>
  </div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```





# Designing for mobile devices

- Different devices have different aspect ratios.
  - Important to test for different device sizes.
  - May sometimes build alternative layouts for different device sizes.
- Using specialized controls important.
  - Enables mobile browsers to use custom device-specific widgets that may be much easier to use.

<small>Mon 4 Nov</small>	<small>12</small>	<small>56</small>
Tue 5 Nov	13	57
Wed 6 Nov	14	58
Thu 7 Nov	15	59
<b>Today</b>	<b>16</b>	<b>00</b>
Sat 9 Nov	17	01
Sun 10 Nov	18	02
Mon 11 Nov	19	03
<small>Tue 12 Nov</small>	<small>20</small>	<small>04</small>



# CSS Best Practices

- When possible, use CSS to declaratively describe behavior rather than code
  - Easier to read, can be optimized more effectively by browser
- Don't repeat yourself (DRY)
  - Rather than duplicating rules, create selectors to style all related elements with single rule
- CSS should be readable
  - Use organization, indentation, meaningful identifiers, etc.





# CSS Exercise

- <https://replit.com/@kmoran/swe-432-react-example#src/App.jsx>
  - Center a component inside it's container
  - Use a display grid to create layout with multiple rows and columns
  - Override one of the Bootstrap selectors



# Acknowledgements

- Slides Adapted from Dr. Thomas LaToza's SWE-432 course