## Announcements

- **Midterm**: Mon May 1
  - In class, bring any books or papers you want
  - Mix of short answer and short programs
  - Open books, open notes
  - No computers, phones

- Starting Wednesday, using a Web server

## Programming problems

- Fill in some functions that are part of a short Javascript program that uses a JSONp API to get information and put it into the DOM.
  - Understand JSONp
  - Might need to use objects defined in the code
  - Need to know how to get and modify DOM elements

## Other topics

- buttons, textboxes, images, paragraphs
- HTML head elements
- RGB colors, hex
- default CSS layout
- flexbox
- media queries
- getting and using DOM elements in Javascript
- Javascript types, conversions, equality
- objects (not prototypes)
- variable scope, inadvertent globals
- string methods and properties

## HTML

- In the head: link in CSS, title, <meta> commands
- The body contains everything you see when you first load the page
- The browser executes the HTML by constructing the Document Object Model (DOM), which controls what appears on the page; think of HTML file as initializing the DOM
- Later changes to the DOM may change the Web page

## CSS

- CSS properties control how elements are displayed:
  - background-color: #99f99;
  - color: #000000;
- Browser uses CSS to construct display as it builds or changes the DOM
- Font stack, from most desired to least
  - font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
- Some properties are inherited by child elements, others are not
**Color**

- Very specific colors can be specified by giving red, green and blue components.
- In CSS:
  
  ```
  color: #ff8020;
  ```

  means all the red, about half the green, and a little blue. An intense reddish orange.

**Default layout**

- Inline elements follow one another like words in text.
- Block elements start and end on a new line, like paragraphs.
- Default block size shrinks to fit content.
- Width and height can be fixed.
  - If so, usually specify in pixels.
  - Note: size of a pixel will vary appropriately with device, does not have to match size of device pixels.
- Can be percentage of screen, also with max and min, to be more responsive.

**Box model**

- All elements can be surrounded by boxes to keep them from butting up against each other.
- Size of element includes size of content and size of padding, border and margin.

**Flexbox**

- Setting display: flex; makes a box and its children flexboxes.
- flex-direction: indicates main layout direction.
- justify-content controls distribution in main direction.
- align-items controls distribution in cross direction.
- align-self controls an item’s own distribution in the cross direction in its container.
- flex-grow and flex-shrink claim an items’ own share of any extra or missing space, which is distributed among sibling items in container.

**Example 1**

```css
main {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  flex-grow: 1; /* fill horizontal space */ /* because it's container is a row box */
}

.box {
  display: flex;
  flex-direction: column;
  justify-content: center;
  flex-grow: 1;
}
```

**Example 2**

```css
main {
  display: flex;
  flex-direction: column;
  align-items: center;
  flex-grow: 1;
}

.box {
  display: flex;
  flex-direction: column;
  justify-content: center;
  flex-grow: 1;
}

#special {
  align-self: end;
  flex-grow: 2;
}
Position property

- To control the position of an element within its block explicitly with Javascript, set `position: relative;` or `position: absolute;`
- After that, setting the `.style.left` (or `.style.right` or `style.top` or `style.bottom`) in Javascript places the element.
- With absolute, just give x or y coordinate, in px, where you want it.
- With relative, give x or y offset from where layout algorithms have already placed it.

Media queries

- Media queries allow you to make fundamental changes to the CSS, typically based on the width of the viewport (window on a computer and screen for phone or tablet).

```javascript
@media (max-width: 480px) {
    #menuIcon {
        display: block;
    }
    nav {
        display: none;
    }
}
```

Javascript

- A variable defined in a function without the `var` keyword become global.
- A variable can contain a function:
  ```javascript
  var f = function() {...} same as function f() {...}
  ```
- Assignment statements and comparisons do automatic conversion:
  ```javascript
  6+"cars" === "6cars" // true!
  3.0 == "3" // true!
  3.0 === "3" // false!
  ```

Object constructor example

```javascript
function CarObj() {
    this.car = "Toyota";
    this.tell = function() {console.log(this.car, mileage);};
    var mileage = 52;
}
myCar =new CarObj;
myCar.car = "Tesla";
myCar.tell();
console.log(myCar.mileage) // causes an error
```

Javascript

- Objects
  - When defined as literals or with assignment to methods and properties.
    - Everything is public
  - When defined using constructor functions
    - Allows private data and methods, defined using `var`
    - Public data and methods are defined, and referred to, using "this"

JSON

- The JSON data format is a Javascript literal
  ```javascript
  dataString = '{"car": "Toyota"}; // the string is JSON
  yourCar = JSON.parse(dataString);
  console.log(yourCar.car); // yourCar is an object
  anotherString = JSON.stringify(yourCar); // the string is JSON
  dataString == anotherString; // true!
  ```
- JSON cannot include methods, only data
Objects can be complex

```javascript
movieData = {"total": 2, "movies": [
  { "id": "770672122", "title": "Toy Story 3", "year": 2010, "mpaa_rating": "G", "runtime": 103,
    "critics_consensus": "Deftly blending comedy, adventure, and honest emotion, Toy Story 3 is a rare second sequel that really works.", "release_dates": {
      "theater": "2010-06-18",
      "dvd": "2010-11-02"
    }, ...
  rating = movieData.movies[0].mpaa_rating;
```