Section 5

Week of March 4

Outline

A few notes

JavaScript

AJAX

A few notes:

Project 2 was released Monday

Due in three weeks: with suggested milestones

JavaScript

- JS is a language, similar to python, that is used to run code on the **client-side**
- It is added to an html page between the <script> </script> tags
- We use ES6 -- a stardard version of JS

Keywords: client, server

- Variables in JavaScript do <u>not</u> require a type specifier, and do <u>not</u> need to be declared in advance. But there is a special keyword for introducing them.

- Variables in JavaScript do <u>not</u> require a type specifier, and do <u>not</u> need to be declared in advance. But there is a special keyword for introducing them.

```
let x = 44;

var y = 44;

const z = 44;
```

- Conditionals are the same as C, and curly braces are used to delimit the blocks again.

- Conditionals are the same as C, and curly braces are used to delimit the blocks again.

```
if
else if
else
switch
?:
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

- Loops are the same as C, and curly braces are used to delimit the blocks again.

```
while
do-while
for
```

- Functions are introduced with the `function` keyword (basically equivalent to Python’s `def`).

- JavaScript functions can be *anonymous*--you don’t have to give them a name!
    - We’ll revisit this idea shortly.
    - By the way, Python technically has this ability too!

- Declaring arrays (again called arrays in JavaScript) looks really similar to a Python list, and can contain mixed types as before.

- Declaring arrays (again called arrays in JavaScript) looks really similar to a Python list.

```
let nums = [1, 2, 3, 4, 5];
```

- As was the case with Python, JavaScript has the ability to behave as an object-oriented programming language, with properties contained within the object, and methods that apply only to objects that define those methods.

- JavaScript objects look a lot like Python dictionaries:

- As was the case with Python, JavaScript has the ability to behave as an object-oriented programming language, with properties contained within the object, and methods that apply only to objects that define those methods.

- JavaScript objects look a lot like Python dictionaries:

```
let herbie = { year: 1963, model: "Beetle"};
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

```
while
do-while
for
for ... in
```

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                 sound: "honk.mp3"
              };
```

- How do we iterate across all of the keys of an object?

```
   let herbie = {  year: 1963,
                   model: "Beetle",
                    sound: "honk.mp3"
                 };
```

```
for (let prop in herbie)
{
    console.log(herbie[prop]);
}
```

- How do we iterate across all of the keys of an object?

```
let herbie = { year: 1963,
               model: "Beetle",
                sound: "honk.mp3"
             };
```

```
for (let prop in herbie)
{
    console.log(herbie[prop]);
}
```

```
1963
Beetle
honk.mp3
```

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                 sound: "honk.mp3"
              };


for (let prop in herbie)
{
   console.log(prop);
}
```

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                 sound: "honk.mp3"
              };
```

```
for (let prop in herbie)
{
   console.log(prop);
}
```

```
year
model
sound
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

```
while
do-while
for
for ... in
for ... of
```

- How do we iterate across all of the elements of an array?)

```
let wkArray = ["Mon", "Tue", "Wed"];
```

- How do we iterate across all of the elements of an array?)

```
    let wkArray = ["Mon", "Tue", "Wed"];
for (let day of wkArray)
{
   console.log(day);
}
```

- How do we iterate across all of the elements of an array?)

```
    let wkArray = ["Mon", "Tue", "Wed"];
for (let day of wkArray)
{
   console.log(day);
}
```

```
Mon
Tue
Wed
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (day + 1) + " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (day + 1) + " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (parseInt(day) + 1) +
            " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

- As with Python, there are still underlying data types, we just don’t often have to worry about them. But here’s the tradeoff of losing the precise control we had in C!

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- This one will give us a good way to introduce anonymous functions.

- An **event** in HTML/JavaScript is a response to user interaction with the web page (e.g. user clicked a button, a page has finished loading…)

- JavaScript supports **event handlers**, which are functions (usually called **callbacks**) that respond to HTML events.

- We can write a generic event handler here, which will automatically create an *event object*, that will tell us which of the two buttons was clicked.

```html
<html>
  <head>
    <title>Event Handlers</title>
  </head>
  <body>
    <button onclick="">Button 1</button>
    <button onclick="">Button 2</button>
  </body>
</html>
```

Ajax

Ajax

- Asynchronous JavaScript and XML
- Ajax is a technique that allows us to dynamically update a page without necessarily needing user intervention.

  Example: infinite scroll on facebook or instagram

- Central to our ability to asynchronously update our pages is to make use of a special JavaScript object called an **`XMLHttpRequest`**.

```
let xhttp = new XMLHttpRequest();
```

- Central to our ability to asynchronously update our pages is to make use of a special JavaScript object called an **`XMLHttpRequest`**.

- After we have this object, we need to define its **`onload`** behavior, which basically is an anonymous function that gets called when the asynchronous request has completed (and so defines what is expected to change on your site!)

```
function ajax_request(argument)
{
   const aj = new XMLHTTPRequest();
   aj.open("GET", <the URL>, true);

   aj.onload = function() {

   };

   aj.send();
}
```

Any remaining questions?