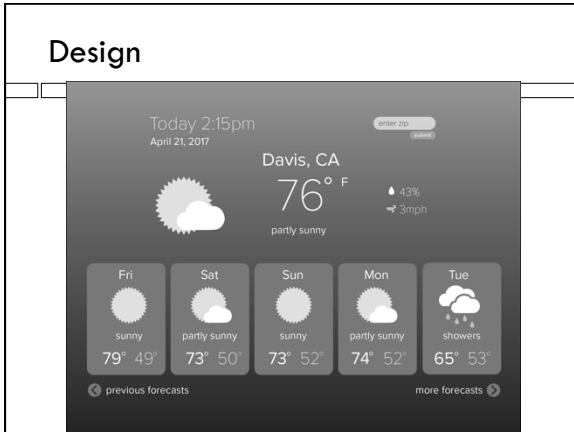


ECS 189H WEB PROGRAMMING

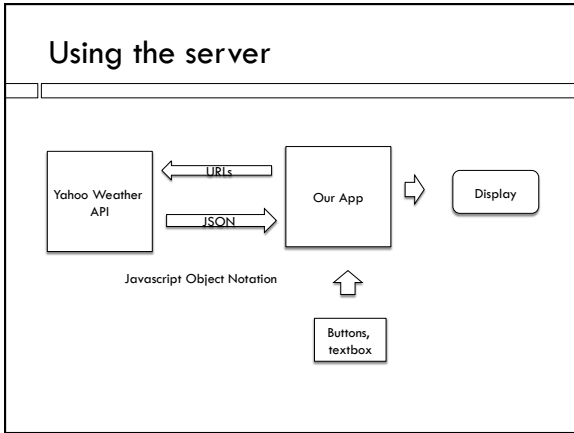
4/21



Grading for Assn 3

- Functionality (getting data, slider window) – 7/10
- Matching design – 3/10

- Get it working, then worry about how it looks.



URL sent

```

<script src="https://query.yahooapis.com/v1/public/
yql?q=select * from weather.forecast where woeid
in (select woeid from geo.places(1) where
text='davis,
ca')&format=json&callback=callbackFunction">
</script>
  
```

- We sent the data request URL, by requesting a script from the server.
- Instead of a local address for the script, we use a (long complex) remote URL. Anybody recognize the syntax?

URL sent

```

<script src "https://query.yahooapis.com/v1/public/
yql?q=select * from weather.forecast where woeid
in (select woeid from geo.places(1) where
text='davis,
ca')&format=json&callback=callbackFunction">
</script>
  
```

- callbackFunction() is a function we have to write. The script the server sends back calls callbackFunction() on the weather data object.

Script returned

```
callbackFunction( {"query":{"count":  
1,"created":"2016-04-21T15:36:48Z","lang":"en-  
US","results":{"channel":{"units":...
```

...goes on and on.

- The returned script calls our callbackFunction on a Javascript object literal. So it's like...

```
function f(y) { return(y.cow); }  
f( {"cow":1} );
```

JSONp

- The name of the protocol – where we ask the server for a JSON string, by pretending to ask for a script, is called JSONp
- It's an unusual technique and not too many servers support it (but Google search and Google maps are two more common examples!)
- More usual is communication from one Web server to another, or so-called AJAX requests from a Web page to its own server.

Why are we doing this?

- Why are we pretending to ask for a script when really we want data?
- In general, a Web page is disabled from getting data from a server other than its own.
- This is a security measure, meant to deter “cross-site scripting” attacks.
- But, people really want Javascript libraries in their browser code.
- So we are allowed to get scripts from sources other than our server!

Using the data

```
callbackFunction({"query":{"count":  
1,"created":"2016-04-21T15:36:48Z","lang":"en-  
US","results":{"channel":{"units":...
```

- To find the weather, callback function needs to parse the object it gets as a parameter and find the part of it containing the current weather and 10-day forecast.
- One way to figure out how to do that would be to check the Yahoo documentation.

When does this happen?

- When does the callback function get called?

When does this happen?

- When does the callback function get called?
- When the page gets loaded, and the browser gets to the bottom of the page and hits the script tag, executes the script retrieved from the URL.
- This is a great for loading the initial Davis weather but how are we going to get the weather from someplace else, when someone enters a zip code?

Even sleazier trick

- Use document methods that modify the DOM to remove the original script, and replace it with a new one.
- The browser executes whenever we modify the DOM, to produce the new display.
- So in this case it will fetch and call the new script!

Adding the text box and button

- ```
<input id="zipbox" type="text"
 placeholder="zipcode or place name">
<button onclick="gotNewPlace()">submit</button>
```
- Easiest to add button that will grab data from text entry box
  - Beware of Websites that tell you to use a <form>, kind of old-school complex tag that can be replaced by a bit of Javascript

## Getting text in Javascript

```
var newPlace =
 document.getElementById("zipbox").value;
```

- It's the "value" property of the text box element.

## Adding box to the HTML

```
var script = document.createElement('script');
script.src = "...";
script.id = "jsonpCall";
document.body.appendChild(script);
```

- Make a new DOM element, the add it as child of the body.
- The "..." is the complicated URL, hopefully including the new location instead of Davis.

## Removing the old script element

```
var oldScript =
document.getElementById("jsonpCall");
if (oldScript != null) {
 document.body.removeChild(oldScript);
}
```

- If there is no old script, we get the value null
- null is a valid value in Javascript, just like true