Introduction to the JSON APIs and AJAX Challenges

Similar to how User Interfaces help people use programs, Application Programming Interfaces (APIs) help programs interact with other programs. APIs are tools that computers use to communicate with one another, in part to send and receive data. You can use API functionality in your page once you understand how to make requests and process data from it. Programmers often use AJAX technologies when working with APIs.  
  
The term AJAX originated as an acronym for Asynchronous JavaScript And XML. It refers to a group of technologies that make asynchronous requests to a server to transfer data, then load any returned data into the page. An asynchronous process has a couple key properties. The browser does not stop loading a page to wait for the server's response. Also, the browser inserts updated data into part of the page without having to refresh the entire page.  
  
User experience benefits from asynchronous processes in several ways. Pages load faster since the browser isn't waiting for the server to respond in the middle of a page render. Requests and transfers happen in the background, without interrupting what the user is doing. When the browser receives new data, only the necessary area of the page refreshes. These qualities especially enhance the user experience for single page applications.  
  
The data transferred between the browser and server is often in a format called JavaScript Object Notation (JSON). JSON resembles JavaScript object literal syntax, except that it's transferred as a string. Once received, it can be converted into an object and used in a script.

LESSON 1

You want your code to execute only once your page has finished loading. For that purpose, you can attach a JavaScript event to the document called DOMContentLoaded. Here's the code that does this:

document.addEventListener('DOMContentLoaded',function() {  
  
});

You can implement event handlers that go inside of the DOMContentLoaded function. You can implement an onclick event handler which triggers when the user clicks on the element with id getMessage, by adding the following code:

document.getElementById('getMessage').onclick=function(){};

JSON APIs and Ajax: Get JSON with the JavaScript XMLHttpRequest Method

You can also request data from an external source. This is where APIs come into play.

Remember that APIs - or Application Programming Interfaces - are tools that computers use to communicate with one another. You'll learn how to update HTML with the data we get from APIs using a technology called AJAX.

Most web APIs transfer data in a format called JSON. JSON syntax looks very similar to JavaScript object literal notation. JSON has object properties and their current values, sandwiched between a { and a } . These properties and their values are often referred to as "key-value pairs".

However, JSON transmitted by APIs are sent as bytes, and your application receives it as a string. These can be converted into JavaScript objects, but they are not JavaScript objects by default. The JSON.parse method parses the string and constructs the JavaScript object described by it.

You can request the JSON from freeCodeCamp's Cat Photo API. Here's the code you can put in your click event to do this:

req=new XMLHttpRequest();  
req.open("GET",'/json/cats.json',true);  
req.send();  
req.onload=function(){  
  json=JSON.parse(req.responseText);  
  document.getElementsByClassName('message')[0].innerHTML=JSON.stringify(json);  
};

Here's a review of what each piece is doing.

The JavaScript has a number of properties and methods that are used to transfer data. First, an instance of the XMLHttpRequest object is created and saved in the req variable.

Next, the open method initializes a request - this example is requesting data from an API, therefore is a "GET" request. The second argument for open is the URL of the API you are requesting data from. The third argument is a Boolean value where true makes it an asynchronous request.

The send method sends the request. Finally, the onload event handler parses the returned data and applies the JSON.stringify method to convert the JavaScript object into a string. This string is then inserted as the message text.

## Convert JSON Data to HTML

Now that you're getting data from a JSON API, you can display it in the HTML.

You can use a forEach method to loop through the data since the cat photo objects are held in an array. As you get to each item, you can modify the HTML elements.

First, declare an html variable with var html = "";.

Then, loop through the JSON, adding HTML to the variable that wraps the key names in strong tags, followed by the value. When the loop is finished, you render it.

Here's the code that does this:

json.forEach(function(val) {  
  var keys = Object.keys(val);  
  html += "<div class = 'cat'>";  
  keys.forEach(function(key) {  
    html += "<strong>" + key + "</strong>: " + val[key] + "<br>";  
  });  
  html += "</div><br>";  
});

GET GEOLOCATION DATA FOR USERS GPS COORDINATES(API4.js

Another cool thing you can do is access your user's current location. Every browser has a built in navigator that can give you this information.

The navigator will get the user's current longitude and latitude.

You will see a prompt to allow or block this site from knowing your current location. The challenge can be completed either way, as long as the code is correct.

By selecting allow, you will see the text on the output phone change to your latitude and longitude.

Here's code that does this:

if (navigator.geolocation){  
  navigator.geolocation.getCurrentPosition(function(position) {  
    document.getElementById('data').innerHTML="latitude: "+ position.coords.latitude + "<br>longitude: " + position.coords.longitude;  
  });  
}

First, it checks if the navigator.geolocation object exists. If it does, the getCurrentPosition method on that object is called, which initiates an asynchronous request for the user's position. If the request is successful, the callback function in the method runs. This function accesses the position object's values for latitude and longitude using dot notation and updates the HTML.

## POST DATA WITH THE JAVASCRIPT XMLHttpRequest Method

In the previous examples, you received data from an external resource. You can also send data to an external resource, as long as that resource supports AJAX requests and you know the URL.

JavaScript's XMLHttpRequest method is also used to post data to a server. Here's an example:

req=new XMLHttpRequest();  
req.open("POST",url,true);  
req.setRequestHeader('Content-Type','text/plain');  
req.onreadystatechange=function(){  
  if(req.readyState==4 && req.status==200){  
    document.getElementsByClassName('message')[0].innerHTML=req.responseText;  
  }  
};  
req.send(userName);

You've seen several of these methods before. Here the open Nmethod initializes the request as a "POST" to the given URL of the external resource, and uses the trueBoolean to make it asynchronous.

The setRequestHeader method sets the value of an HTTP request header, which contains information about the sender and the request. It must be called after the open method, but before the send method. The two parameters are the name of the header and the value to set as the body of that header.

Next, the onreadystatechange event listener handles a change in the state of the request. A readyState of 4 means the operation is complete, and a status of 200 means it was a successful request. The document's HTML can be updated.

Finally, the send method sends the request with the username value, which was given by the user in the input field.