Trigger Click Events with jQuery

In this section, we'll learn how to get data from APIs. APIs - or Application Programming Interfaces - are tools that computers use to communicate with one another.

We'll also learn how to update HTML with the data we get from these APIs using a technology called Ajax.

First, let's review what the $(document).ready() function does. This function runs such that all of the code inside of it executes only once our page has finished loading.

So Let's start by implementing a click event handler inside of our $(document).ready() function by adding this code:

$("#getMessage").on("click", function(){  
  
});

After completing this, proceed to the next challenge where we will make our "Get Message" button change the text of the element with the class message.

Change Text with Click Events

When our click event happens, we can use jQuery to update an HTML element.

Let's make it so that when a user clicks the "Get Message" button, we change the text of the element with the class message to say "Here is the message".

We can do this by adding the following code within our click event:

$(".message").html("Here is the message");

Get JSON with the jQuery getJSON 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.

Most web APIs transfer data in a format called JSON. JSON stands for JavaScript Object Notation.

You've already been using JSON whenever you create a JavaScript object. JSON is nothing more than object properties and their current values, sandwiched between a { and a }.

These properties and their values are often referred to as "key-value pairs".

Let's get the JSON from freeCodeCamp's Cat Photo API.

Here's the code you can put in your click event to do this:

$.getJSON("/json/cats.json", function(json) {  
 $(".message").html(JSON.stringify(json));  
});

Once you've added this, click the "Get Message" button. Your Ajax function will replace the "The message will go here" text with the raw JSON output from the Free Code Camp Cat Photo API.

#### Convert JSON Data to HTML

Now that we're getting data from a JSON API, let's display it in our HTML.

We can use the .forEach() method to loop through our data and modify our HTML elements.

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

Then, let's loop through our JSON, adding more HTML to that variable. When the loop is finished, we'll 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>";  
});

#### Render Images from Data Sources

We've seen from the last two lessons that each object in our JSON array contains an imageLink key with a value that is the URL of a cat's image.

When we're looping through these objects, let's use this imageLinkproperty to display this image in an img element.

Here's the code that does this:

html += "<img src = '" + val.imageLink + "' " + "alt='" + val.altText + "'>";

#### Prefilter JSON

If we don't want to render every cat photo we get from our freeCodeCamp Cat Photo JSON API, we can pre-filter the json before we loop through it.

Let's filter out the cat whose "id" key has a value of 1.

Here's the code to do this:

json = json.filter(function(val) {  
 return (val.id !== 1);  
});

#### Get Geolocation Data

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

The navigator will get our 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 some code that does this:

if (navigator.geolocation) {  
 navigator.geolocation.getCurrentPosition(function(position) {  
 $("#data").html("latitude: " + position.coords.latitude + "<br>longitude: " + position.coords.longitude);  
 });  
}

Don't forget to use double quotes around selectors as mentioned [**here.**](https://www.freecodecamp.org/challenges/target-html-elements-with-selectors-using-jquery)