API Tutorial

Wireframes

Sketch out what you want your version of the class webpage to look like on paper. This is your low-fidelity wireframe. Here is link to the class page for reference: http://www.bvandorn.com/idesn-3535/

Start Coding

Create a new blank webpage! We all know how to do this. Make sure you bring in your normalize.css stylesheet!

jQuery

Bring in jQuery!

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"
integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" crossorigin="anonymous"></script>
```

Class Schedule API

We are going to build our own version of the course website. That's why I didn't put any work at all into designing it :|

Let's get some of the basic structure setup. I'm going to start with some headers, a link to the syllabus, and an empty table.

```
<h2>Syllabus</h2>
<a href="http://bit.ly/2gEbCrd" target="_blank">Syllabus</a>

<h2>Course Schedule</h2>
<table id="class-schedule" border="1">

</table>
```

Use the API

This is one of the most basic forms of an API. It is simply a hosted JSON file that we are going to load into our site. To do this, we are going to use something called AJAX (Asynchronous Javascript And XML). It's a high level abstraction on all the code that is needed to actually make this stuff happen - which you don't need to know.

First we wait for the document to be ready:

```
<script>
  $(document).ready(function() {
    // Do stuff here
  });
</script>
```

Then, let's get the basic ajax script setup:

```
// Do stuff here

$.ajax({
  url: "http://www.bvandorn.com/idesn-3535/schedule.json",
  data: {},
  success: function(response) {
    // This function gets called when we get our file!
  }
})
```

Let's log the response to the console so we can see what is returned when we load up our page.

```
success: function(response) {
  // This function gets called when we get our file!
  console.log(response);
}
```

Open your webpage and take a look at the console, you should see something that looks like this:

Notice how we have a list of every class with a bunch of other information in each class.

Processing The Data

Awesome, now let's do something with that data.
`response.classes` is an array, so we can loop through each class in a for loop. There are many ways to loop through arrays, this is just one of many.

```
// Loop through every class
for (let i = 0; i < response.classes.length; i++){
  const currentClass = response.classes[i];

}
```

Test Your Code: Go ahead and use a console.log to spit every class out to the console and check that your loop is working correctly.

Now we need to do a few things. We are going to create some elements in javascript using `document.createElement`. Again, there are many different ways to create elements, this is not the only way. We then set the innerHTML to contain our class number by using the current index in the loop. The date is a property on our currentClass variable. Then we append our classRow to our table with jQuery.

```
// Loop through every class
for (let i = 0; i < response.classes.length; i++){
  const currentClass = response.classes[i];

  // Create a <tr> element via javascript for our class row.
  const classRow = document.createElement('tr');

  // Setup the Class # and Date for our first column.
  const dateCol = document.createElement('td');
  dateCol.innerHTML = "Class " + i + "<br />" + currentClass.date;
  classRow.append(dateCol);

  $("#class-schedule").append(classRow);

}
```

Go ahead and refresh your page. You should see a table with a list of all classes with dates. Make sure you understand what each line of this code is doing to create the table.

The Agenda

Let's get the agenda next. This is going to call for another loop because `currentClass.agenda` is an array. We can use a forEach loop for this one.

```
// setup the agenda column
const agendaCol = document.createElement('td');
// our agenda is an unordered list so we can create a <ul>
const agendaList = document.createElement('ul');
// We can use forEach as another way to loop through an array
currentClass.agenda.forEach(function(agendaItem){
  // Create list items, set the text as our agendaItem
  const agendaListItem = document.createElement('li');
  agendaListItem.innerText = agendaItem;
  // Append our <li> to our <ul>
  agendaList.append(agendaListItem);
});
// append our <ul> into our column.
agendaCol.append(agendaList);
// add the new column to the class row.
classRow.append(agendaCol);
```

Refresh and bask in the glory of the agenda column!

Homework, Homework, Homework

Try It Yourself: Homework is also an array of strings, just like the agenda. Go ahead and add a homework column.

Class Notes

The last column to add is the class notes. This is also an array, but it is an array of objects - a little more complicated than the array of strings, but not by much. We're going to use string concatenation and innerHTML to make this one use less lines of code.

NOTE: Not every class has notes, so we need to check that there is a list of links using an if statement before trying to loop through it.

```
const notesCol = document.createElement('td');
const notesList = document.createElement('ul');
if (currentClass.links){ // not every class has notes
  currentClass.links.forEach(function(link){
    notesList.innerHTML += `<li><a href="${link.link}">${link.name}</a></li>`;
  });
}
notesCol.append(notesList);
classRow.append(notesCol);
```

Refresh your page and you should have a list of links!

Headers

We are going to take a step back and add a row of headers to our table as a last step. Technically tables should have a <thead> and <tbody> so we'll add those as well.

```
<h2>Course Schedule</h2>
<table id="class-schedule" border="1">
  <thead>
    <tr><th>Class</th><th>Agenda</th><th>Homework</th><th>Class Notes</th></tr>
  </thead>
  <tbody></tbody>
</table>
```

Make sure you go back to your javascript to append your rows into the <tbody>

```
$("#class-schedule > tbody").append(classRow);
```

Styling

Style time! Choose some fonts, remove the bullets, add some color, make this page look good! Try to match what you sketched at the beginning of class. One hint, to remove the massive ugly borders on the table, the CSS property for the table is `border-collapse: collapse;`