CIT 281 WEB DEV

281 Project 3

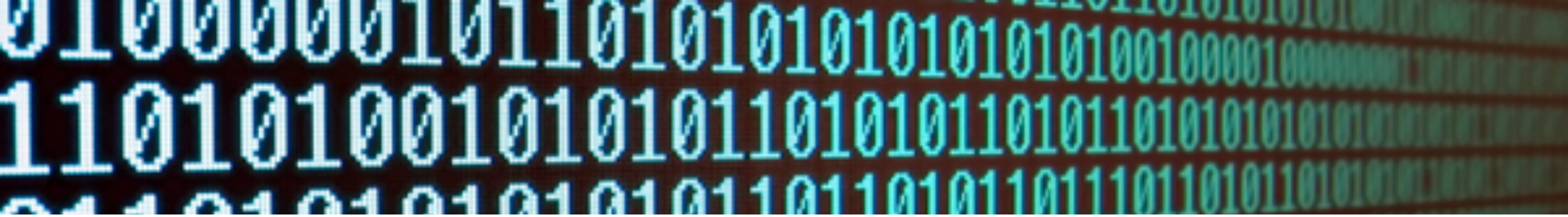
Michael Hennessy, Computer & Information Science, UO

CIT 281 Web Dev

Project 3

Learning Objectives

- A web API or *Application Programming Interface* is an API accessed using the HTTP protocol. Web APIs often deliver data formatted as JSON.
- A web API can be accessed programmatically using JavaScript. The API documentation must first be consulted in order to understand how the API works and what it returns.
- Use jQuery to post AJAX requests asynchronously
- Use the Postman app to test API requests/responses
- Use NodeJS and Express for a testing server
- Use jQuery, JavaScript, and public APIs to fetch JSON data, then use it to refresh part of a web page



Due Dates:

- **23:59 Fri 5/8 (XC)**
- **23:59 Mon 5/11**

GIT WORKFLOW FOR 281

Commits are usually made when you've completed something you are working on-- it's done, and you won't be changing it.

A good time to do a commit is when you are finished with a project requirement.

The Basic Git Workflow

```
$ git status
$ git add .
$ git commit -m ". . ."
$ git push origin master
$ git status
$ git log
```

In Chrome, reload your remote Github repo to verify the push.

Github Tip: never edit files directly on Github. Always make your changes in your local repo, and then push them to Github. This leaves your local and remote repos in the same state.

CREATE YOUR PROJECT-3 GITHUB REPO

- Sign in to Github.

- Paste this Project-3 repo invitation link into Chrome:
`https://classroom.github.com/a/Yd5VOLia`

- Github will create a remote private repo for you, and add it to your other repos on Github.

Important: Notify me right away if your project-3 repo is not available on Github-- I can manually create a repo for you.

- Use the **`git clone`** command to download the remote repo to your computer

- Use Atom's `File > Add Project folder..` command to open your `project-3` repo folder

PROJECT REQUIREMENTS

1. **[20 pts] Your Github User Page.**

   In your week 5 lab, you created a Github User page. Complete the page, if you have not done so already.

2. **[18 pts] Add the Project 3 Learning Objectives to your README.md File.**

   In Atom open README.md. Note that it currently describes the Start-State of your repo.

   a) Open your README.md file from project 2. It currently has two h2 markdown header elements: *281 Project 2 Learning Outcomes*, and 281 *Project 1 Learning Outcomes*.

   - Copy all the Markdown code from your project 2 README.md. and paste it into your project 3 README.md.

   b) At the top, add a third Markdown h2 header element, *281 Project 3 Learning Outcomes*.

   c) Following the new h2 header, add a Markdown unordered list showing the six learning objectives for project 3.

   d) Following the h1 header at the top, add a hyperlink that connects to your Github User Page that your created in your Week 5 lab.

   e) When you are done, use the *Basic Git Workflow* commands to *stage, commit*, and *push* your changes to the *master* branch on your remote repo.

3. **[30 pts] The Dog API**. The Dog API (https://dog.ceo/dog-api/) is an open API that does not require your app to use key authentication, and has a reasonable rate limit. Read the documentation to learn how to retrieve JSON data about dogs by breed (*Airedale, Beagle, Chow, ..., Wolfhound*).

   a) Start a Unix shell (terminal, Gitbash, WSL) and cd to your `~/Documents/repos/281/projects/project-3/` directory.

   b) In Atom, rename the three gallery files as *dogs-gallery.html*, *dogs-gallery.css*, and *dogs-gallery.html*.

   c) Modify *dogs-gallery.html* to connect to the renamed .css and .js files.

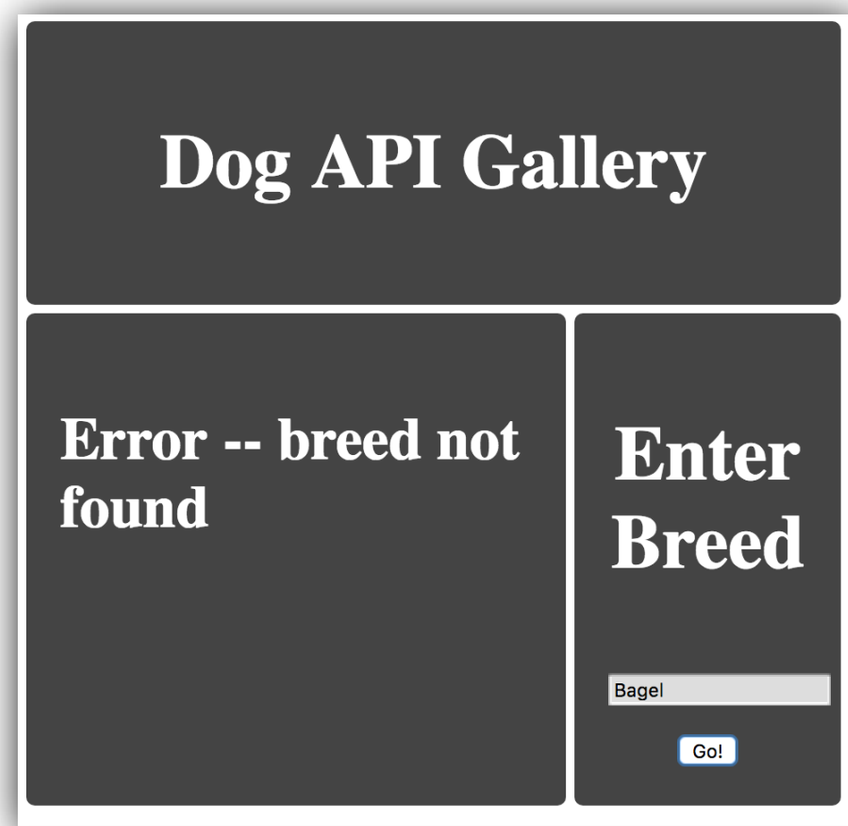
   d) Add this fail handler to the top of the `.js` file:

```
let failHandler = () => {
   console.log("Fail -- unknown breed");
   $(".photos").empty().html("<h3>Error -- breed
not found<h3>");
};
```

   Attach it to the end of the `getJSON` method as shown, below. **Notice** that the dot `(.)` immediately follows the right parenthesis, with no space between the parenthesis and the dot:

```
$.getJSON(requestURL, function(response) {. . .})
.fail(failHandler);
```

   When the user enters an invalid breed, your app will flag it like this:

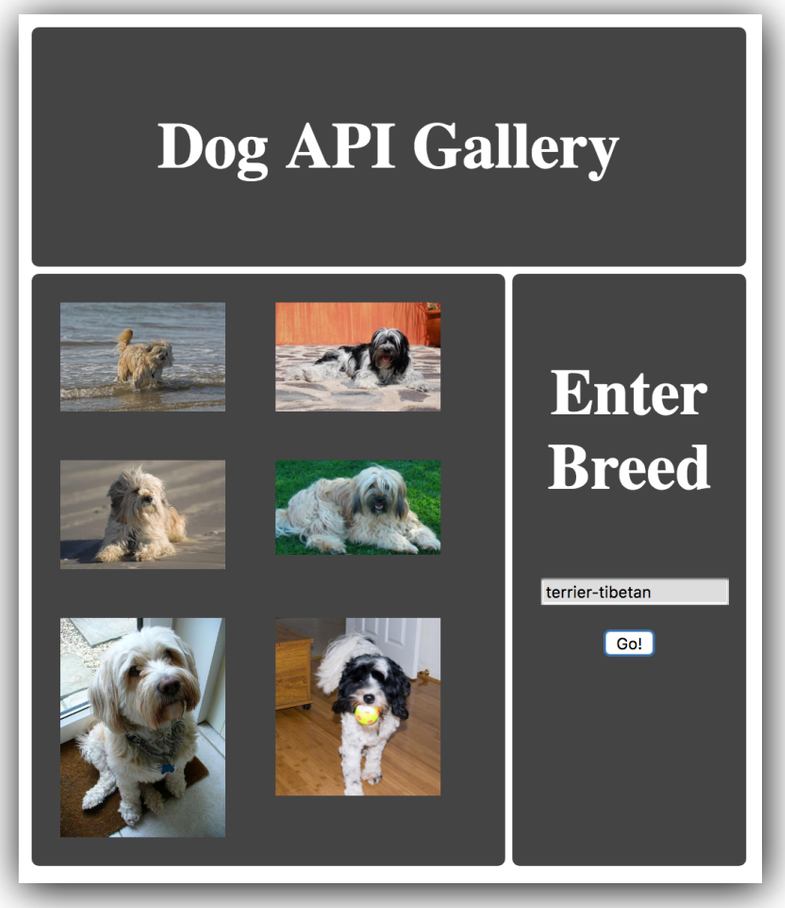


   e) Next, you will modify *dogs-gallery.js* to read the dog breed entered by the user, fetch images of that breed from the dog.ceo server, and display six images in the gallery.

First: modify the test code to extract a single image from the response. The Postman app is useful for examining the structure of the JSON response returned by the dog api.

Second: modify the *forEach* method to create six image elements and add each to the photos box, just like you did for the Flickr gallery app on project 2.

When complete, your web app should look similar to this:



f) Each time the user clicks the button, clear the photos box, so that only six photos are displayed at one time.

g) When you are done, use the *Basic Git Workflow* commands to *stage, commit*, and *push* your changes to the *master* branch on your remote repo.

4. **[32 pts] Coding Practice: BitBatBotOrNot.**

a) In Atom, create a new file named *functions.js* in your *client/javascripts* directory. Paste this starter code into functions.js.

b) Write a function named *bitBatBotOrNot* that accepts a number and returns either "Bit", "Bat", "Bot", or "Not", depending on the number's factors:

- If the number has 3 as a factor, return 'Bit'.
- If the number has 5 as a factor, return 'Bat'.
- If the number has 7 as a factor, return 'Bot'.
- If the number does not have 3, 5, or 7 as a factor, return 'Not'.

Examples:

9 has factors: 1, 3, 9:

```
bitBatBotOrNot(9) -> 'Bit'
```

10 has factors: 1, 2, 5, 10:
`bitBatBotOrNot(10) -> 'Bat'`

28 has factors 1, 2, 4, **7**, 14, 28:
`bitBatBotOrNot(28) -> 'Bot'`

30 has factors 1, 2, **3, 5**, 6, 10, 15, 30:
`bitBatBotOrNot(30) -> 'BitBat'`

105 has factors: 1, 3, 5, 7, 105:
`bitBatBotOrNot(105) -> 'BitBatBot'`

34 has factors: 1, 2, 17, 34:
`bitBatBotOrNot(34) -> 'Not'`

c) Write an arrow function named *findAllbitBatBotOrNots1* that accepts an array of numbers, and returns a new array with each number replaced by a string consisting of the number followed by its *bitBatBotOrNot* value.

- Use the Array.map method, and no loops at all

Example:

```
findAllbitBatBotOrNots1([ 9, 10, 28, 30, 34,
105]) ->
["9: Bit", "10: Bat", "28: Bot", "30: BitBat",
"34: Not", "105: BitBatBot"]
```

d) Write an arrow function named *findAllbitBatBotOrNots2, that does the same thing as findAllbitBatBotOrNots1.*

- Use a *for* loop, and the Array.push method

e) Write an arrow function named *findAllbitBatBotOrNots3,* that does the same thing as *findAllbitBatBotOrNots1.*

- Use a *for..of* loop, and Array.push

f) Write an arrow function named *findAllbitBatBotOrNots4,* that does the same thing as *findAllbitBatBotOrNots1.*

- Use the *Array.forEach* method, and Array.push

g) When you are done, use the *Basic Git Workflow* commands to *stage, commit*, and *push* your changes to the *master* branch on your remote repo.

5. **[+5pts XC] Optional Extra Credit: findAllbitBatBotOrNotsXC**.

   In *functions.js*, write an arrow function named *findAllbitBatBotOrNotsXC,* that does the same thing as *findAllbitBatBotOrNots1.*

   - Use a *for..in* loop, and *Array.push*

6. **[+5 XC] Optional eXtra Credit: Tags with Spaces**.

   The Dog API has endpoints by Breed and Sub-Breed.

   Allow the user to enter sub-breeds using spaces, as shown in these examples:

   Tag entered by user: `english bulldog`
   URL:
   https://dog.ceo/api/breed/bulldog/english/images/random/6

   Tag entered by user: `golden retriever`
   URL:
   https://dog.ceo/api/breed/retriever/golden/images/random/6

   Tag entered by user: `russell terrier`
   URL:
   https://dog.ceo/api/breed/terrier/russell/images/random/6

To indicate that you have completed this XC exercise, add Spaces are OK to box c in the .html file:

Meeting the Deadline

How to Handle the Deadline

- Start working on your project early. Do not delay.
- Turn in what you have by the deadline-- partial credit is better than none.

[Instructors] are a Superstitious Sect, Great Keepers of Set Times and Places.

-- from *Poor Richard's Almanac*