BUILDING AN API-DRIVEN APP

WHAT YOU NEED TO KNOW OR ABOUT TO LEARN

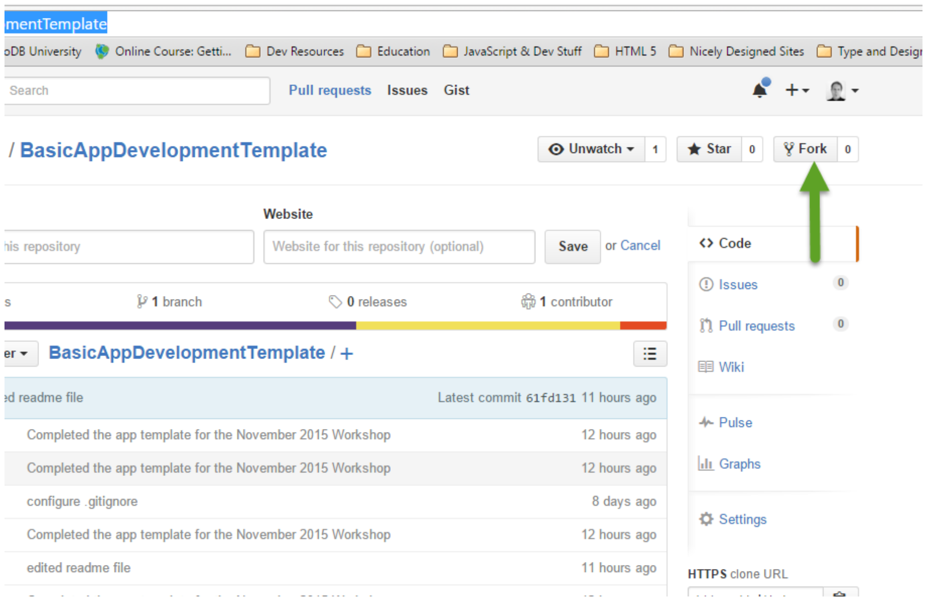
- Difference between JavaScript for web development and JavaScript application development.
- HTML
- CSS & Sass (a style-sheet language)
- JavaScript Programming (Head First JavaScript Programming)
- Git Versioning, a GIthub.com account
- Application Build Tools (e.g., Gulp), this requires Node.js be installed on your laptop
- Asynchronous Processing, AJAX, and API standards

PART 1 OBJECTIVES: CHUNKING OUT THE BIG PARTS

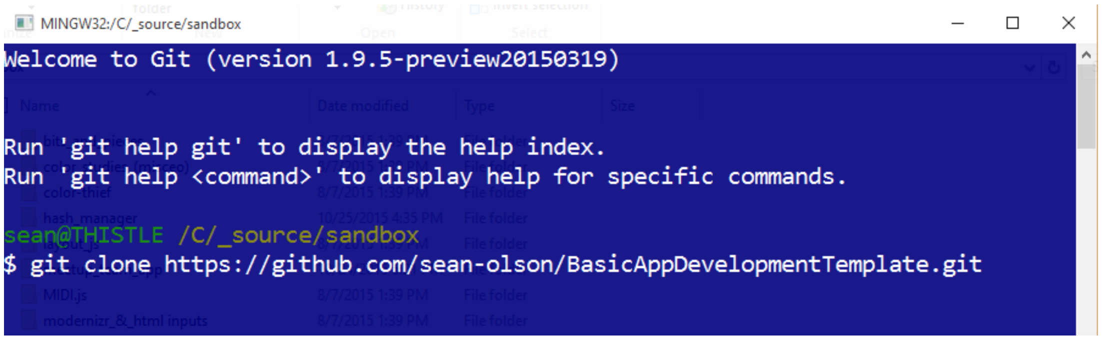
- Getting a working repository
- Setting up a build environment, including Node.js and Gulp
- Work out the logic of the app using the problem solving model we covered in August – scrum style
- Understanding what API's are and how to use asynchronous processing?
- Getting an API key
- Getting to know the Meetup API:
- Issues that are essential but beyond what we're doing: CORS and authentication.
- Constructing a URL to get data
- Using Postman to test
- Wiring up your app to an API, getting data back and displaying it on the page.

FORKING AND BRANCHING THE REPO

1. Verify that everyone has a git hub account and fork the repository: https://github.com/RiversideJSW/building_api_driven_app

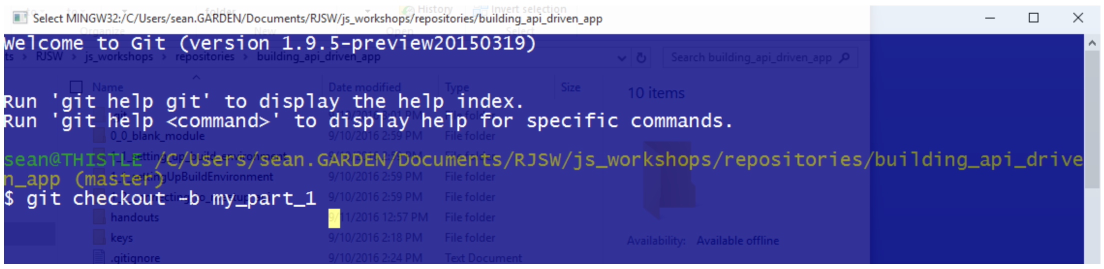


2. Clone the repository to the laptop. Once you have the repo cloned to your laptop you will have all the handouts for this project



```
MINGW32:/C/_source/sandbox
Welcome to Git (version 1.9.5-preview20150319)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

sean@THISTLE /C/_source/sandbox
$ git clone https://github.com/sean-olson/BasicAppDevelopmentTemplate.git
```

3. Create a working branch *my_part_1*

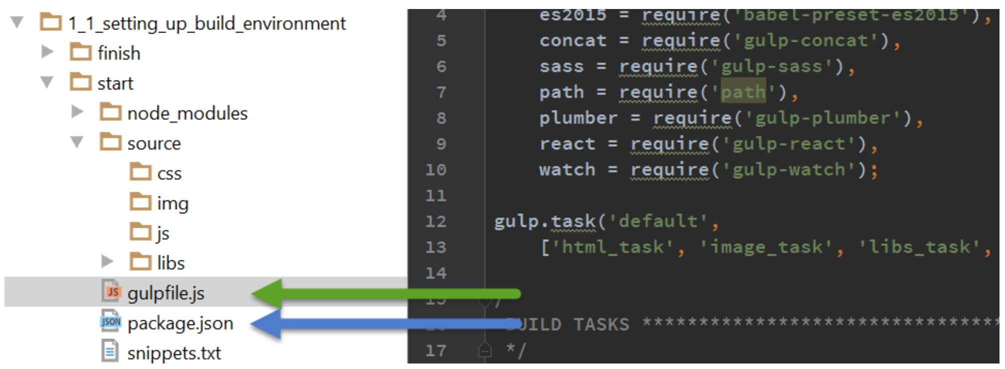


```
Select MINGW32:/C/Users/sean.GARDEN/Documents/RJSW/js_workshops/repositories/building_api_driven_app
Welcome to Git (version 1.9.5-preview20150319)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

sean@THISTLE /C/Users/sean.GARDEN/Documents/RJSW/js_workshops/repositories/building_api_drive
n_app (master)
$ git checkout -b my_part_1
```

As you develop your application you will want to periodically update your git repository and "push" it to GitHub.com. You do this by using a simple set of Git commands. You will find a cheat sheet in the handouts folder in your project. Look for the sub-folder labeled *2_git_notes*.

SETTING UP YOUR BUILD ENVIRONMENT

For this project we will be using Gulp.js to run our build tasks such as generating our CSS files, combining JavaScript files, running a development server on your local machine, etc. The configuration for those operations is contained in your *Gulp file*, named gulpfile.js. Your gulpfile.js resides at the root level of your application's directory. In addition to a configuration, Gulp also requires a number of modules to function. Those modules are listed in a second file in the root directory titled package.json.



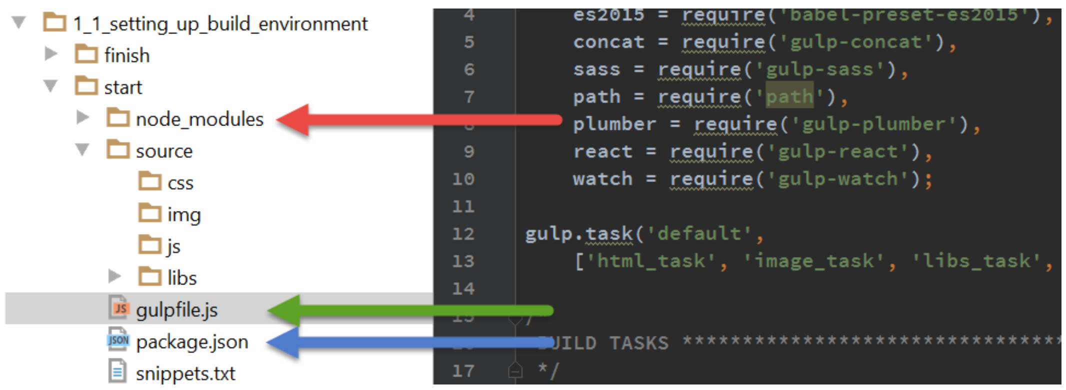
```
1_1_setting_up_build_environment
    finish
    start
        node_modules
        source
            css
            img
            js
            libs
        gulpfile.js
        package.json
        snippets.txt
```

```
    es2015 = require('babel-preset-es2015'),
    concat = require('gulp-concat'),
    sass = require('gulp-sass'),
    path = require('path'),
    plumber = require('gulp-plumber'),
    react = require('gulp-react'),
    watch = require('gulp-watch');

gulp.task('default',
    ['html_task', 'image_task', 'libs_task',

*/
 UILD TASKS ********************************
*/
```

To install the modules, open a command/console in the root folder and run the command to install the modules:

```
> npm install
```

This will create a node_modules folder in the root of your application that contains the modules requires for Gulp.js to function.



```
1_1_setting_up_build_environment
    finish
    start
        node_modules
        source
            css
            img
            js
            libs
        gulpfile.js
        package.json
        snippets.txt
```

```
    es2015 = require('babel-preset-es2015'),
    concat = require('gulp-concat'),
    sass = require('gulp-sass'),
    path = require('path'),
    plumber = require('gulp-plumber'),
    react = require('gulp-react'),
    watch = require('gulp-watch');

gulp.task('default',
    ['html_task', 'image_task', 'libs_task',

*/
 UILD TASKS ********************************
*/
```

From the same command/console window, now type the command

```
> gulp
```

Once you have gulp running, set up the basic files for you application. Use the snippets file to paste in the necessary code. Here are the files to create in your start/source directory:

- /index.html
- /js/app.js
- /css/app.scss

- If necessary you can restart Gulp by stopping it in the command window/console by pressing Ctrl+C. Then type the command gulp *gulp,*and it should fire back up.
- Open a browser and navigate to http://localhost:8080.

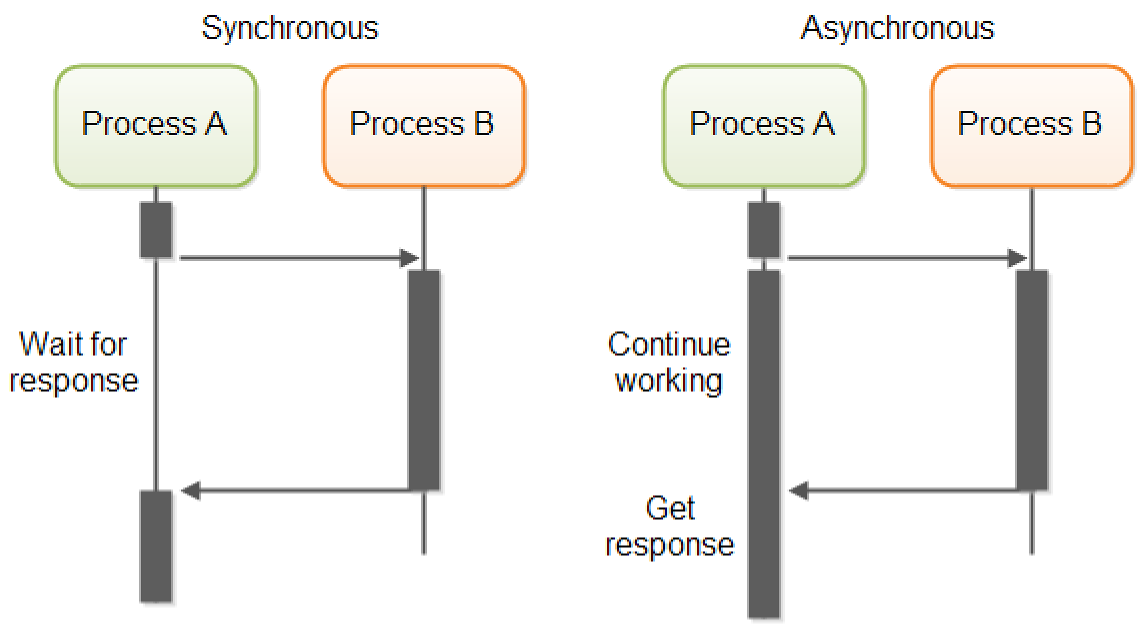
You now have a working development environment, and you're ready to begin development. as soon as you know what you're going to build.

ANALYZE THE PROBLEM AND DEVELOP A MODEL

This process was covered in the August workshop, and the handouts are in the folder titled 3_development_process. Working from the file, *development_process.pdf*, we're going to analyze and work through the problem.

ASYNCHRONOUS PROCESSING – WHAT IS IT?

Asynchronous processing refers to processes that do not depend on each other's outcome, and can therefore occur on different threads simultaneously. The opposite is synchronous. Synchronous processes wait for one to complete before the next begins.



USING CALLBACKS TO DEAL WITH ASYNCHRONOUS PROCESSES

A **callback function**, also known as a higher-order **function**, is a **function** that is passed to another **function** (let's call this other **function** “otherFunction”) as a parameter, and the **callback function** is called (or executed) inside the otherFunction.

GETTING A MEETUP.COM API KEY

Follow these steps to get an API key for Meetup.com

- Log into your account at Meetup.com
- Navigate to: https://secure.meetup.com/meetup_api/key/
- Copy the key from the screen, paste into a text file on your computer.
- In the top level of the project folder, create a directory named "keys" – folders with this name are omitted from being committed to repo.

SECURING YOUR KEY

You never want to put a key value into you code, which means you have to find a way to enter it into your app. In most cases, you would ask a user to enter it one time and save it using a cookie or some other mechanism like local storage.

Let's begin putting together a simple page that will allow the user to enter a key and save it.

1. Close any command windows / consoles from the previous exercise.
2. Open up your project folder, *1_2_connecting_to_meetup_api/start.*
3. Open a command window/console in this folder and run **`npm install`** to load all the modules needed to run gulp.
4. When npm has finished loading all the modules, enter the **`gulp`** command.
5. When gulp has started, open a browser and navigate to http://localhost:8080.
6. Open the app.js file and copy the snippet into the MU module.
7. Open your index.html file and copy in the markup from the snippets file.
8. Refresh the browser, enter your key, and click save.
9. Close and reopen the browser to test.

TESTING YOUR KEY ON THE MEETUP.COM API

Using Postman test your api access using this method call for categories:

https://api.meetup.com/2/categories?key=YOUR_KEY_HERE&order=shortname&desc=false