



# Citi Bike Project with Leaflet and Intro to Projects

Data Boot Camp

Lesson 15.3



# Class Objectives

---

By the end of this lesson, you will be able to:



Complete an in-class group project using Leaflet.js.

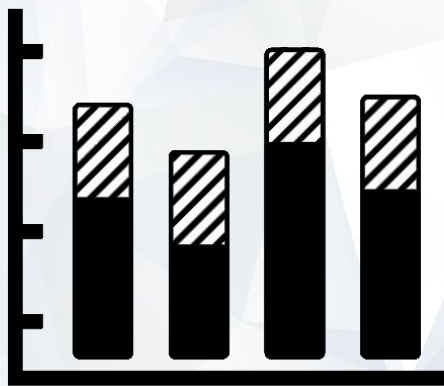


Deploy data visualizations to GitHub Pages.



Draft a project proposal in a team setting.

# Overview Of Your Career Resources



# What are your career goals?

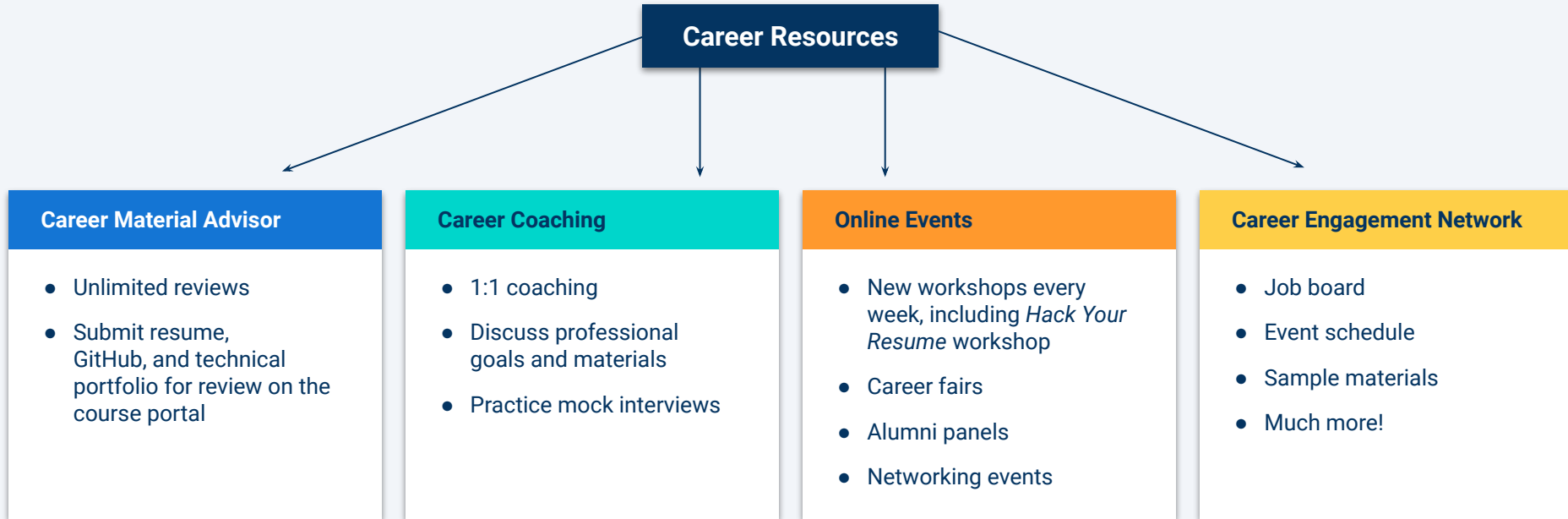
In the chat indicate your post-bootcamp career goals.

- +1 If you want to find a new job.
- +2 If you want a promotion or salary increase.
- +3 If you want to start a business.

**Suggested Time:**  
**1 minute**

# Your Career Resources

---



**After your resume is approved by a Career Material Advisor you will be matched with your Career Coach. Submit your resume via the Career Services tab on the course portal.**

# Working With Your Career Coach

---

Your Career Coach provides you with 1:1 coaching to help you be Employer Competitive in your job search.

Topics include:



Applying and networking



Salary negotiation



Gaining traction to land interviews



Motivation and more!



Conducting mock interviews

# Working With Your Career Coach

---

You have two options:

01

1:1 scheduled bi-monthly recurring coaching calls

02

Reaching out to your Career Coach when needed



## **We recommend scheduled Recurring Calls. Why?**

The data shows that our students who have professional application materials and participate in recurring calls are much more likely to secure the jobs they want.

# Working With Your Career Coach

---

## Next Steps:

01

Visit the Career Engagement Network ([careernetwork.2U.com](https://careernetwork.2U.com)) and explore the resources available to you.

*Definitely check out the virtual workshops and events!*

02

Get your resume approved by a Career Material Advisor by submitting it via the Career Services tab in the course portal.

*This will grant you access to your Career Coach!*

03

Schedule a 1:1 meeting with your Career Coach!





# Instructor Demonstration

---

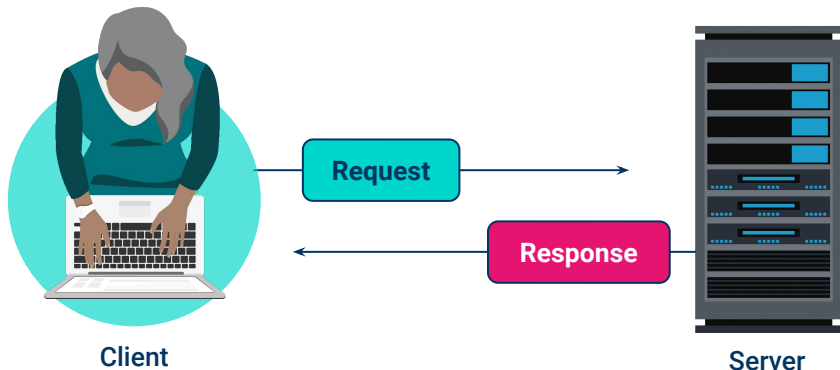
Use the Python HTTP Server

# Use the Python HTTP Server

Here are some things to note as we live-code:

## A server

A **server** is a program or device that performs actions such as processing and sharing data.



## Cross-Origin Resource Sharing

**Cross-Origin Resource Sharing (CORS)** is a mechanism that tells browsers to access selected resources from a web server through information in the HTTP headers in a web application.

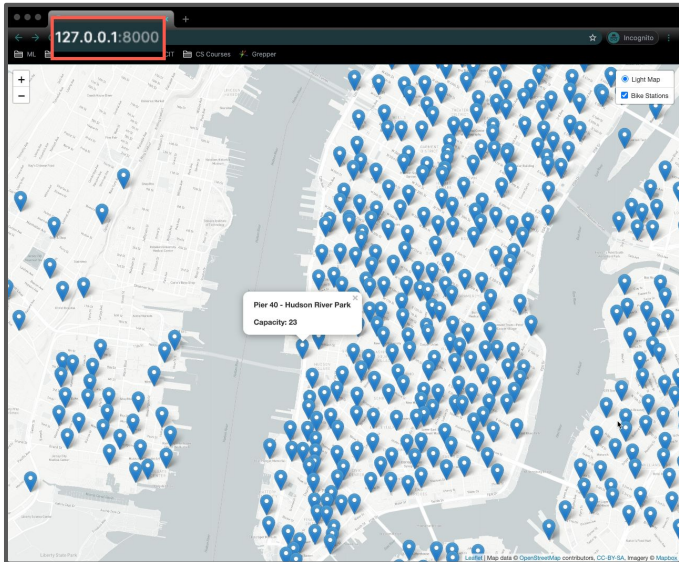
CORS provides a way to allow cross-origin requests.

```
python -m http.server
```

# Create Citi Bike Maps

# Instructor Do: Introduce Citi Bike

## Basic Version



→ Citi Bike API Station Information Endpoint

```
d3.json("https://gbfs.citibikenyc.com/gbfs/en/station_information.json", createMarkers);
```

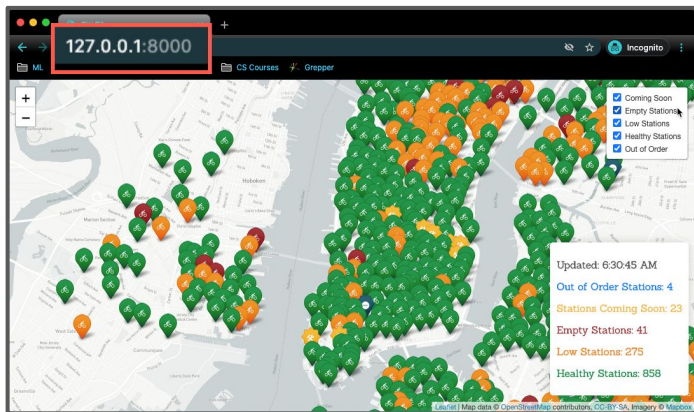
→ One KVP (key value property) of the JSON

```
{
  "stations": [
    {
      "station_type": "classic",
      "lon": -73.99392888,
      "region_id": "71",
      "lat": 40.76727216,
      "rental_url": "http://app.citibikenyc.com/S6Lr/IBV092JufD?station_id=72",
      "name": "W 52 St & 11 Ave",
      "short_name": "6926.01",
      "rental_methods": ["CREDITCARD", "KEY"],
      "electric_bike_surcharge_waiver": false,
      "external_id": "66db237e-baca-11e7-82f6-3863bb44ef7c",
      "eighth_station_services": [],
      "capacity": 55,
      "has_kiosk": true,
      "legacy_id": "72",
      "station_id": "72",
      "eighth_has_key_dispenser": false
    }
  ]
}
```

- Each marker is placed at the latitude and longitude returned by the request.
- When someone clicks a marker, a popup displays the station name and capacity.
- These responses include the name, station, and capacity of each station.

# Instructor Do: Introduce Citi Bike

## Advanced Version



→ Citi Bike API Station Information + Status Endpoint

```
d3.json("https://gbfs.citibikenyc.com/gbfs/en/station_information.json", function(infoRes) {
d3.json("https://gbfs.citibikenyc.com/gbfs/en/station_status.json", function(statusRes) {
  var updatedAt = infoRes.last_updated;
  var stationStatus = statusRes.data.stations;
  var stationInfo = infoRes.data.stations;
  var stationCount = {
    COMING_SOON: 0,
    EMPTY: 0,
    LOW: 0,
    NORMAL: 0,
    OUT_OF_ORDER: 0
  };
};
```

- This version groups markers into layers according to station status.
- When someone clicks a marker, a popup displays the station name, capacity, and bikes available.
- These responses include the name, station, and capacity of each station.



## Groups Do: Create Citi Bike Maps

In this activity, you and your group will work with the Citi API to build a map of all the Citi Bike stations and their statuses.

**Suggested Time:**  
30 minutes



# Instructions: Groups Do: Create Citi Bike Maps

---

- **Basic Version**

1. Use the [Citi Bike station information endpoint](#) to get information about the station names and locations. Take a moment to study the data that the endpoint sends back in your browser. Note the following:
  - Each object in the `stations` array has `station_id`, `name`, `capacity`, `lat`, and `lon` properties.
  - The [logic.js](#) file contains coordinates that you can use to position a Leaflet map over New York City.
2. Create a function named `createMap` that takes `bikeStations` as an argument. This function will create both the tile layer and an overlay with the pins for each station.
3. Create a second function named `createMarkers` that will take `response` as an argument.
  - Using the response from a future D3 call, loop through the stations, and create a marker to represent each station.
  - Give each marker a popup to display the name and capacity of its station.
4. In the `createMarkers` function, pass the resulting bike markers to the `createmap` function as a `layerGroup`.
5. Using D3, retrieve JSON data from the [Citi Bike station information endpoint](#), and call the `createMarkers` function.

# Instructions: Groups Do: Create Citi Bike Maps

---

- **Advanced Version**

1. Write code to perform a second API call to the [Citi Bike station status endpoint](#). Take a few moments to study the data that the endpoint returns. In particular, notice `station_id`, `num_bikes_available`, `is_installed`, and `is_renting`.
2. Using the data returned by the second API call, add the following functionality:
  - In the popup for each marker, display the number of available bikes.
  - Add a layer control, and split the markers into the following layer groups:
    - i. **Coming Soon:** This applies if a station isn't yet installed.
    - ii. **Empty Stations:** This applies if a station has no available bikes.
    - iii. **Out of Order:** This applies if a station is installed but not renting.
    - iv. **Low Stations:** This applies if a station has less than five available bikes.
    - v. **Healthy Stations:** This applies if a marker doesn't fall into any of the previous layer groups.
3. Use a Leaflet plugin to create different types of markers to represent the layers. The following step shows an example map that uses [Leaflet.ExtraMarkers](#). However, feel free to use another plugin if you prefer.
4. Add a legend to your map to explain the different markers.
5. When you complete the app, deploy it to GitHub Pages.



# Instructions

## Groups Do: Create Citi Bike Maps

---

- **Hints**

- Make sure that you run `python -m http.server` in the folder that contains your files. Because you'll do all the work on the front end of your app, you won't need to restart the router after making changes.
- Here are some helpful links:
  - [Leaflet map example](#)
  - [Citi Bike station information API endPoint](#)
  - [Leaflet popup documentation](#)
  - [Citi Bike station status API endPoint](#)
  - [Leaflet layer groups documentation](#)
  - [Leaflet.ExtraMarkers](#)
  - [Leaflet legend documentation](#)



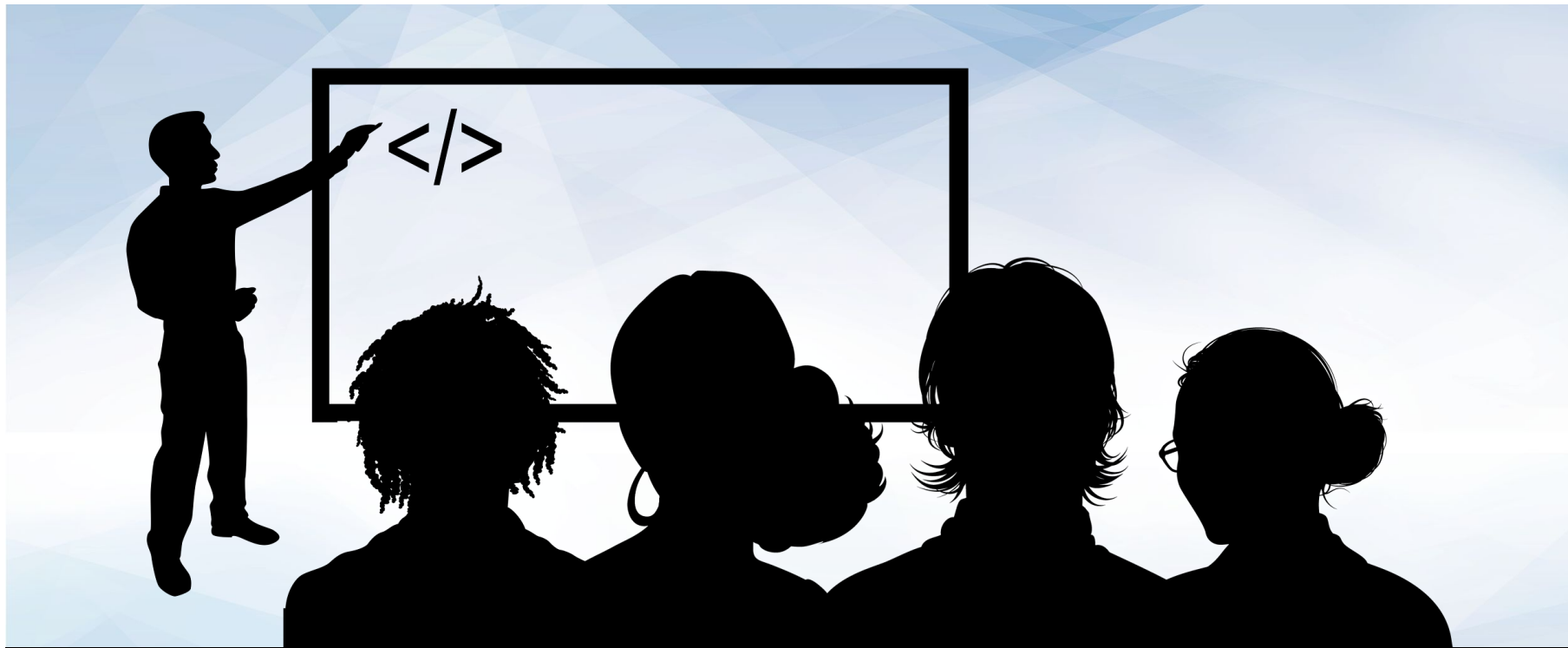


**Let's Review**



A close-up photograph of a white computer keyboard. The central focus is a large, white, rectangular key with rounded corners. On this key, there is a dark blue icon of a coffee cup with three wavy lines above it representing steam. Below the icon, the word "Break" is printed in a dark blue, serif font. The key is slightly raised from the keyboard's surface. Surrounding this key are other white keys: to the left is a key with double quotation marks, above is a key with a single apostrophe, and to the right is a key with a single dash/underscore. The keyboard has a light-colored, textured surface.

Break



# Instructor Demonstration

## Deploy a Project to GitHub Pages

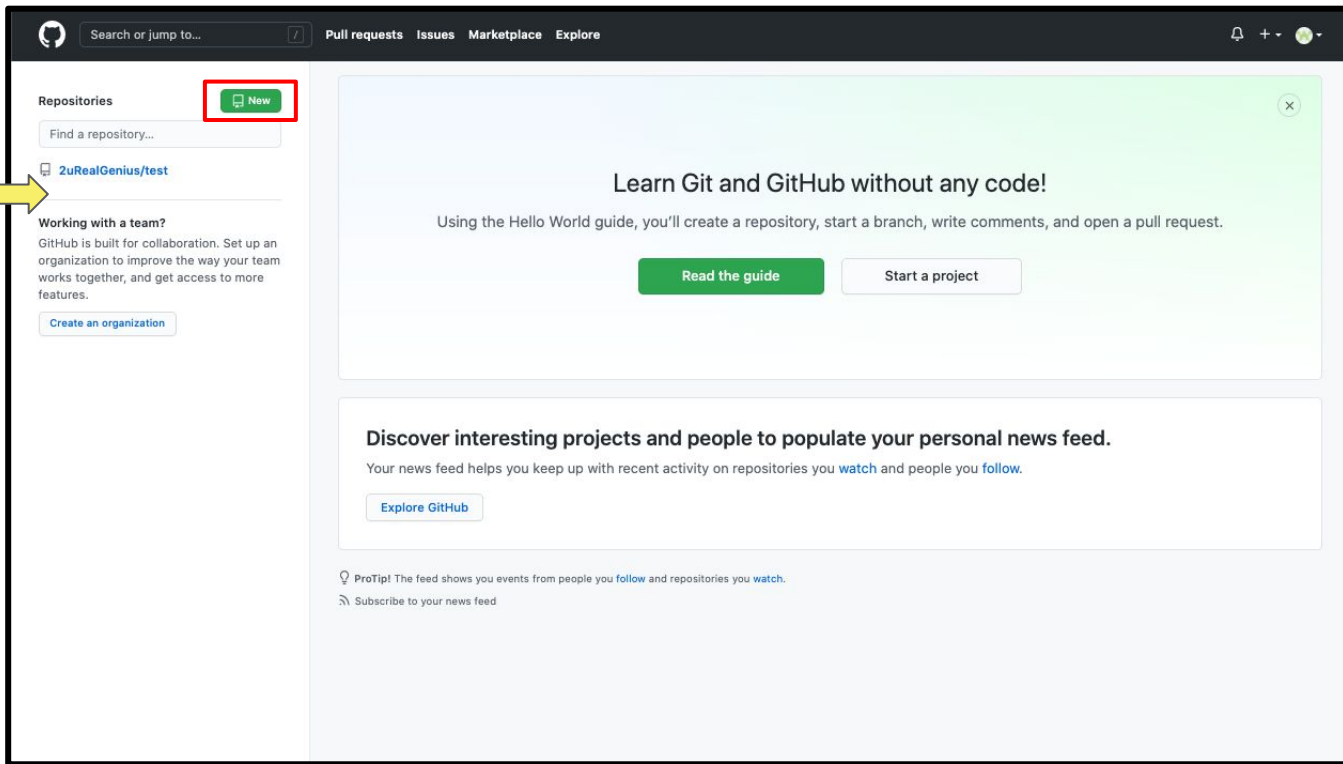
# Instructor Do: Deploy a Project to GitHub Pages

## 1. GitHub

Navigate to <http://github.com>,

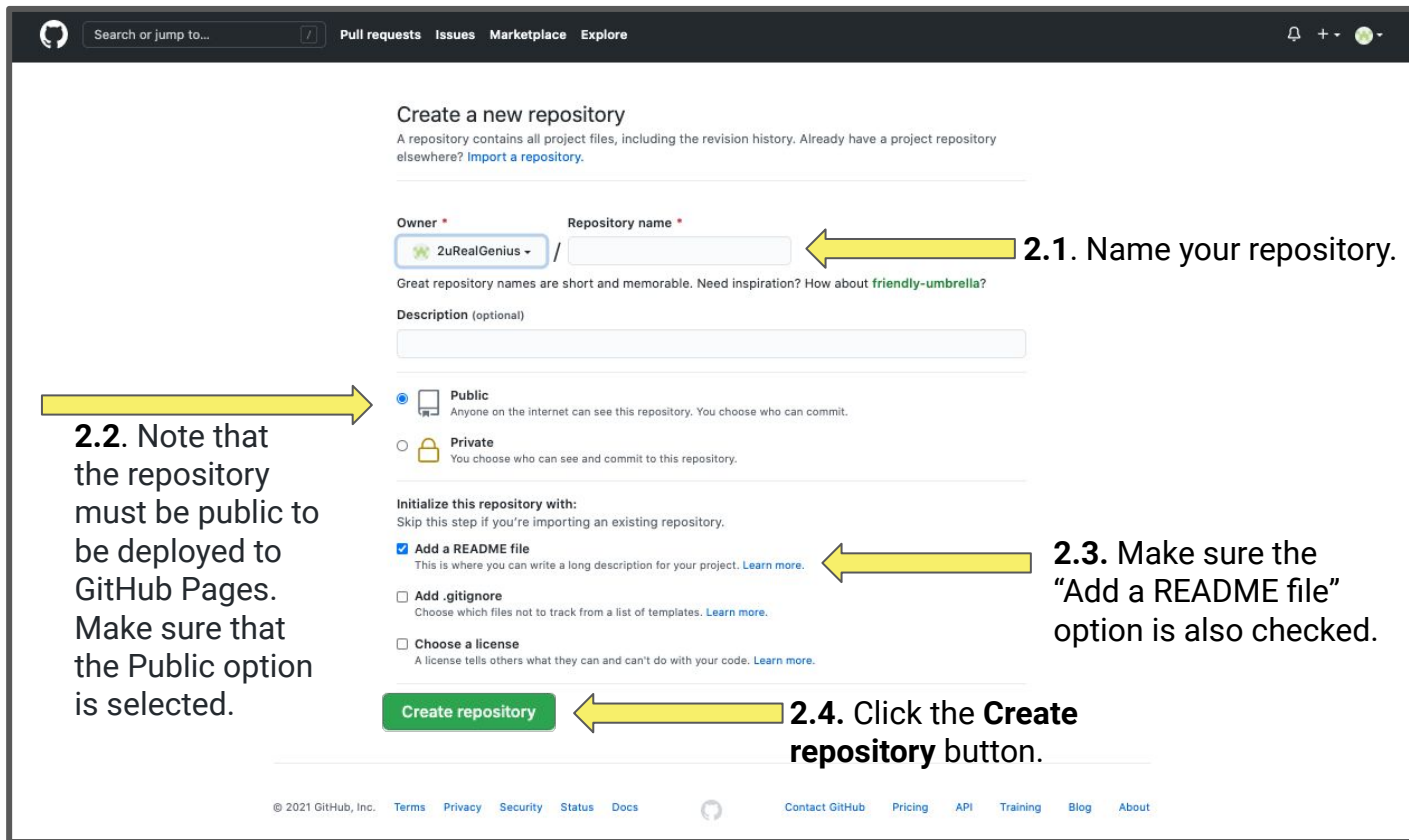
then create a new repository

by clicking



# Instructor Do: Deploy a Project to GitHub Pages

## 2. GitHub



The screenshot shows the GitHub 'Create a new repository' page. The page has a dark header with the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The main content area is white and contains the following sections:

- Create a new repository**: A heading followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).'
- Owner**: A dropdown menu showing '2uRealGenius'.
- Repository name**: An empty text input field.
- Description (optional)**: An empty text input field.
- Visibility**: Two radio buttons: 'Public' (selected) and 'Private'.
- Initialize this repository with:**: A section with three checkboxes: 'Add a README file' (checked), 'Add .gitignore', and 'Choose a license'.
- Create repository**: A green button at the bottom.

Four yellow arrows point to specific elements on the page, each with a numbered annotation:

- 2.1. Name your repository.**: Points to the 'Repository name' input field.
- 2.2. Note that the repository must be public to be deployed to GitHub Pages. Make sure that the Public option is selected.**: Points to the 'Public' radio button.
- 2.3. Make sure the "Add a README file" option is also checked.**: Points to the 'Add a README file' checkbox.
- 2.4. Click the Create repository button.**: Points to the 'Create repository' button.

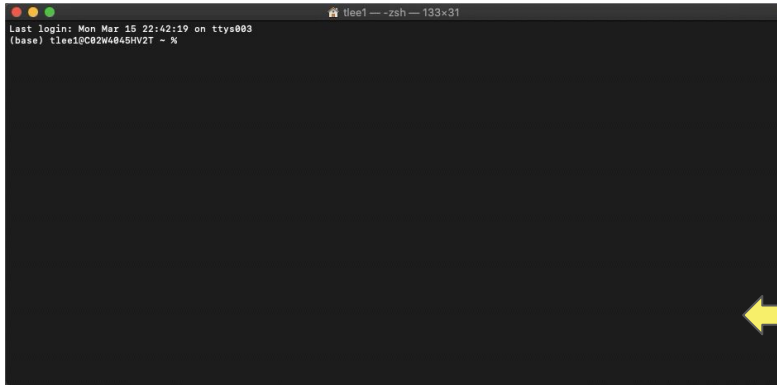
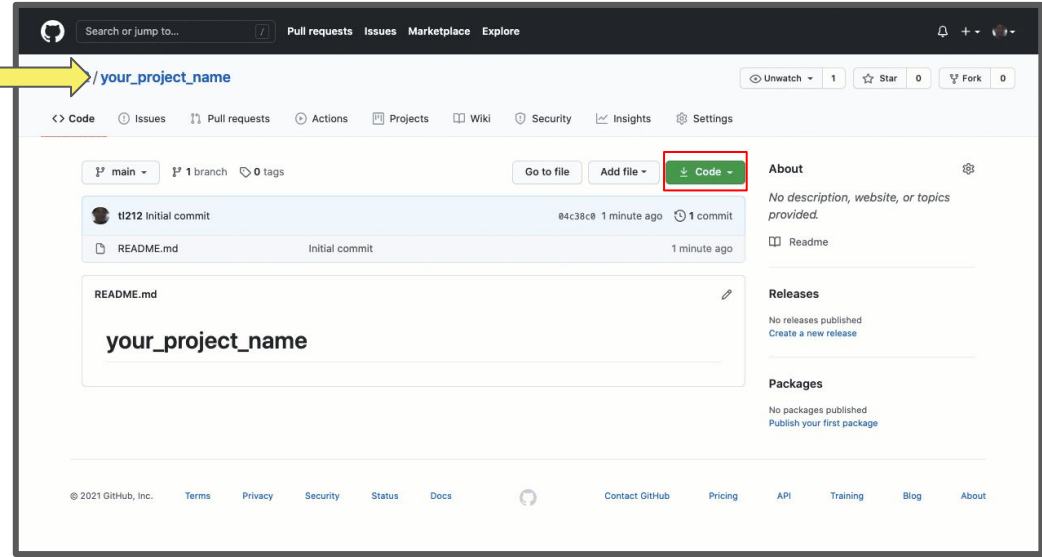
At the bottom of the page, there is a footer with copyright information and links for Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

# Instructor Do: Deploy a Project to GitHub Pages

## 3. GitHub

You will now be directed to your repository page.

Click on  to copy the URL of your repository.



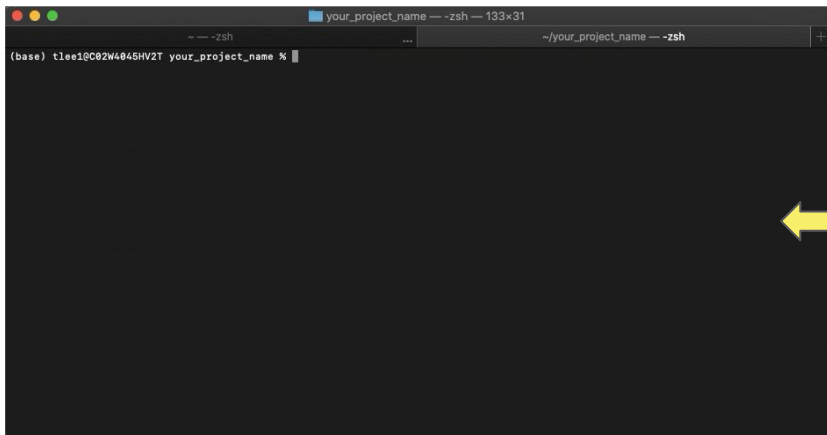
Next, open the command line and type:

```
git clone <url>
```

# Instructor Do: Deploy a Project to GitHub Pages

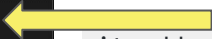
## 4. GitHub

Now that we have the repository in GitHub and cloned to your local machine, copy and paste the HTML, JavaScript, and JSON files from the `Solved` directory to your local repository.



```
your_project_name — zsh — 133x31
~ — zsh
(base) tlee1@C02W4046HV2T your_project_name %
```

Once you have pasted the files to your local repository, open CLI to push the changes by typing:



```
git add .
git commit -m 'your commit msg'
git push origin main
```



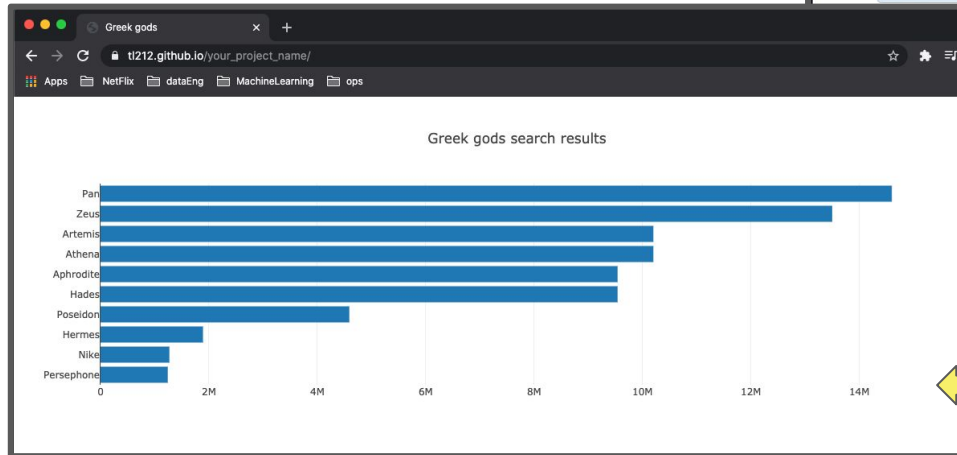
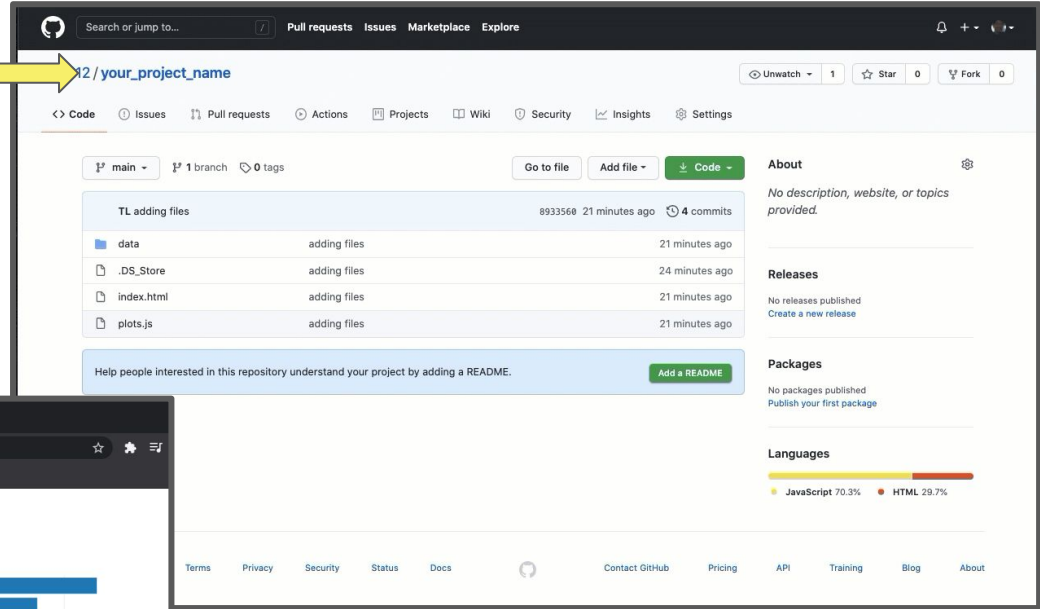
```
▼ data
  {} data.json
  <> index.html
  JS plots.js
```



# Instructor Do: Deploy a Project to GitHub Pages

## 5. GitHub

Navigate back to your GitHub repository page. Under Settings, go to **GitHub Pages**, and then in the **Select source list**, select **main branch** and click **Save**.



The project should now be deployed to GitHub Pages, as in the following.



## Activity: Deploy the Citi Bike Project

In this activity, you will deploy a Plotly project with a local data file to GitHub Pages.

**Suggested Time:**  
20 minutes



## Instructions:

# Activity: Deploy the Citi Bike Project

---

1. Note that you've been given a Plotly visualization project with `index.html`, `plot.js`, and `data.json`.
2. Deploy the project to GitHub Pages.

- **Hints:**

- Consult [GitHub Pages](#) for reference. Be sure to select the Project Site and Start from Scratch options for instructions.



**Let's Review**



# Instructor Demonstration

## Introduce Project 3

# Project Requirements

# Project Description

---

01

Your task is to **tell a story** with data visualizations.

02

Focus on providing users an **interactive way** to explore data themselves.

03

Prepare a **10-minute presentation** that lays out your theme, coding approach, data wrangling techniques, and final visualization.

04

You may choose a project of any theme, but we encourage you to **think broadly**.

05

You will have **plenty of time in class** to work with your group, but expect to put in **hours outside of class** as well.

# Specific Requirements

---

1. Your visualization must include a Python Flask-powered API, HTML/CSS, JavaScript, and at least one database (SQL, MongoDB, SQLite, etc.).
  2. Your project should fall into one of the below four tracks:
    - A combination of web scraping and Leaflet or Plotly
    - A dashboard page with multiple charts that update from the same data
    - A “thick” server that performs multiple manipulations on data in a database prior to visualization (**must be approved**)
  3. Your project should include at least one JS library that we did not cover.
  4. Your project must be powered by a dataset with at least 100 records.
  5. Your project must include some level of user-driven interaction, such as menus, dropdowns, and textboxes.
  6. If possible, your final visualization should include at least three views.
-



# Schedule

# Project Schedule

---

## Day 1 (Next Class)

**Start brainstorming topics with your group and researching potential datasets. Your focus be:**

- Selecting a topic
- Finding a dataset
- Finding inspiration
- “Sketching” your ideal visuals
- Creating a 1-page proposal

## Day 2

**You will need to create a one-page proposal that includes:**

- A brief articulation of your chosen topic and rationale
- A link to your dataset(s) and a screenshot of the metadata, if it exists.
- Three or four screenshots of relevant, “inspiring” visualizations that show your creative ideas
- A sketch of the final design
- A link to the primary GitHub repository where you’ll be housing your work

## Day 3

**Project Work**

# Final Thoughts

---

01

Project week is a great time to tie up loose ends, both with your group and on your own.

02

If there are topics you'd like to review, send me and the TAs a message. We're happy to do (recorded) extra review sessions for small groups during these weeks.

03

Good luck and have fun!



Questions?