

**HECK YEA, IT'S  
CCLAB!**

# **TODAY'S PROGRAM**

# TODAY'S PROGRAM

- Admin: Attendance
- Presentation of Assignments
- Terminal + Git
- LocalStorage
- JavaScript Libraries

# ASSIGNMENTS

# **TERMINAL & GIT**

**SHELL YEAH!**

# TERMINAL + GIT

**GIT =**

- a distributed version control system
- a full-fledged repository
- history + version-tracking

# TERMINAL + GIT

## Main Expressions

- `cd, ls, mkdir, rmdir, pwd`
- `git init`
- `git remote add`
- `git pull`



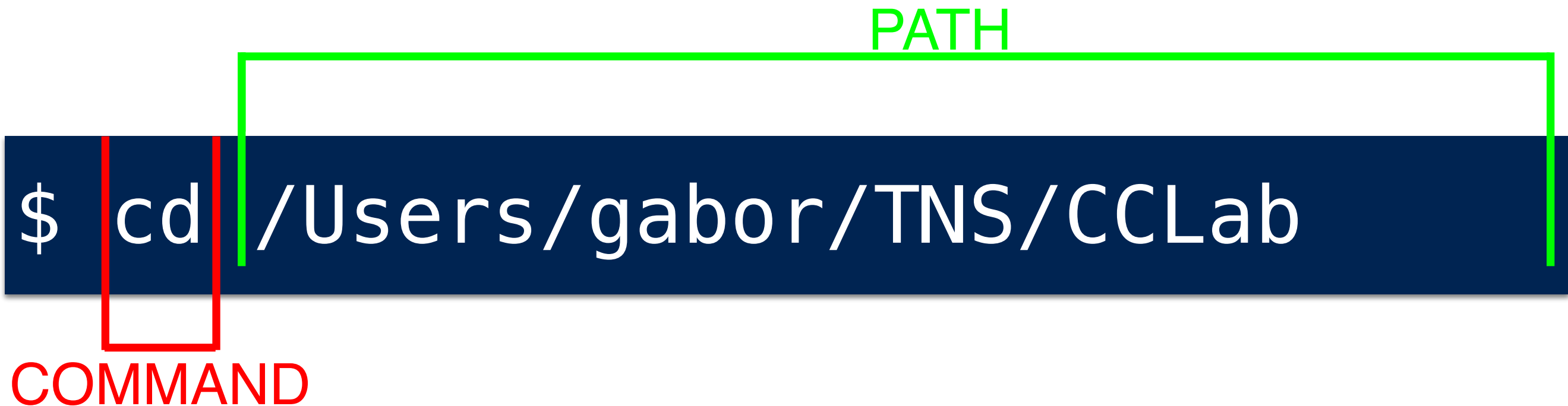
# TERMINAL + GIT

type **cd** and space

# TERMINAL + GIT

drag your CCLab folder  
to the terminal window

# TERMINAL + GIT



A diagram illustrating a terminal command. The command is displayed on a dark blue background. A red bracket underlines the word 'cd', with the label 'COMMAND' in red text below it. A green bracket spans the entire path '/Users/gabor/TNS/CCLab', with the label 'PATH' in green text above it.

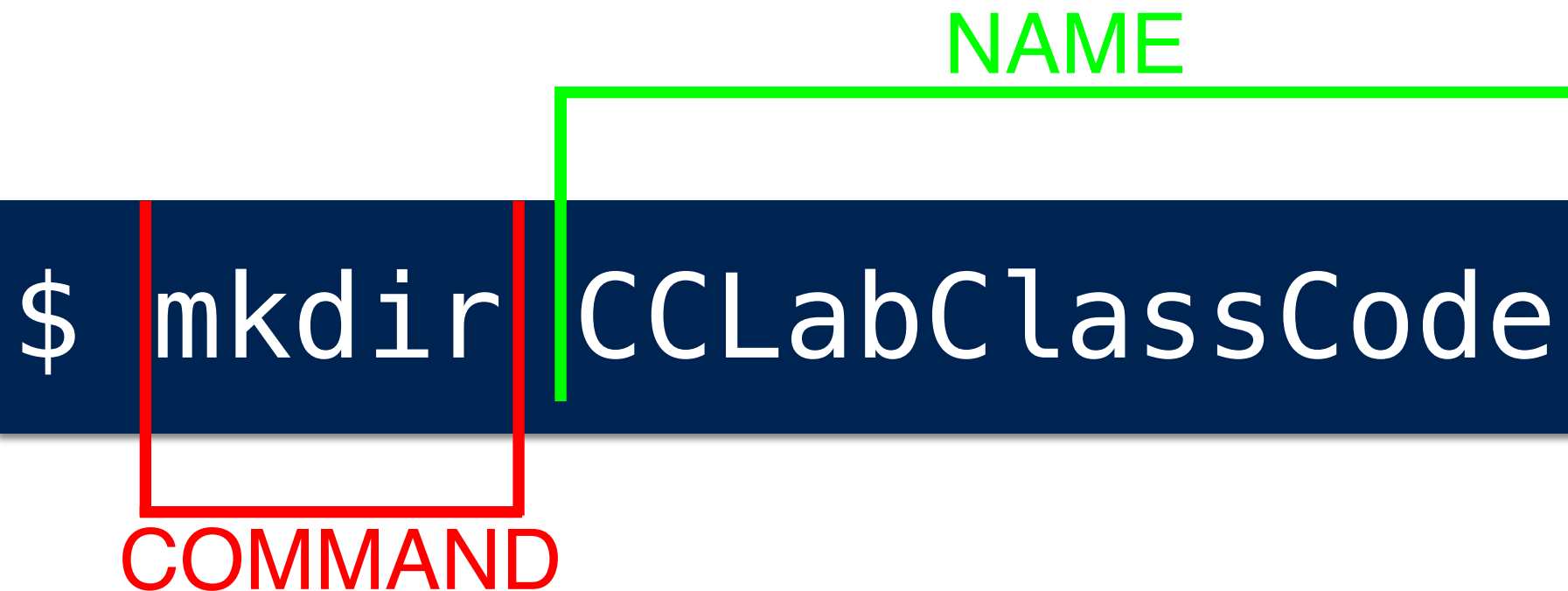
```
$ cd /Users/gabor/TNS/CCLab
```

COMMAND

PATH

cd means “change directory”

# TERMINAL + GIT

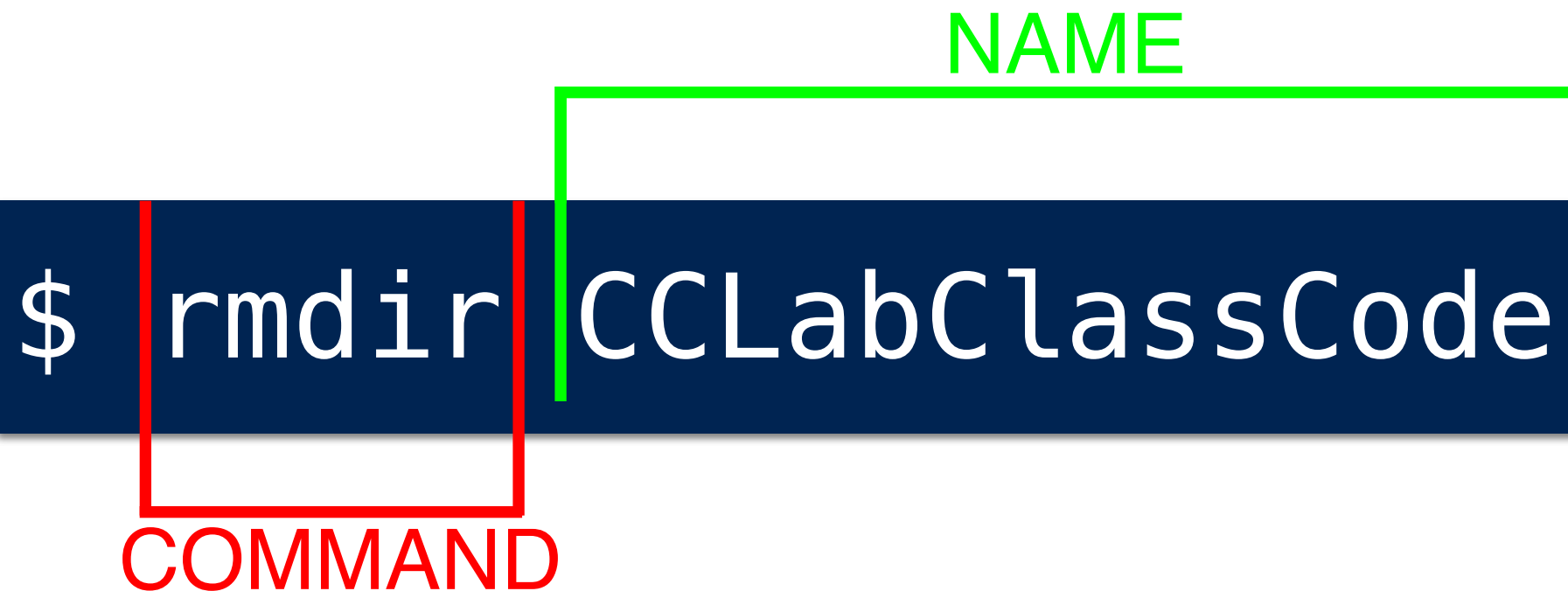


A diagram illustrating the components of a terminal command. The command `$ mkdir CCLabClassCode` is displayed on a dark blue background. A red bracket below the word `mkdir` is labeled **COMMAND** in red text. A green bracket above the text `CCLabClassCode` is labeled **NAME** in green text.

```
$ mkdir CCLabClassCode
```

`mkdir` means “make directory”

# TERMINAL + GIT

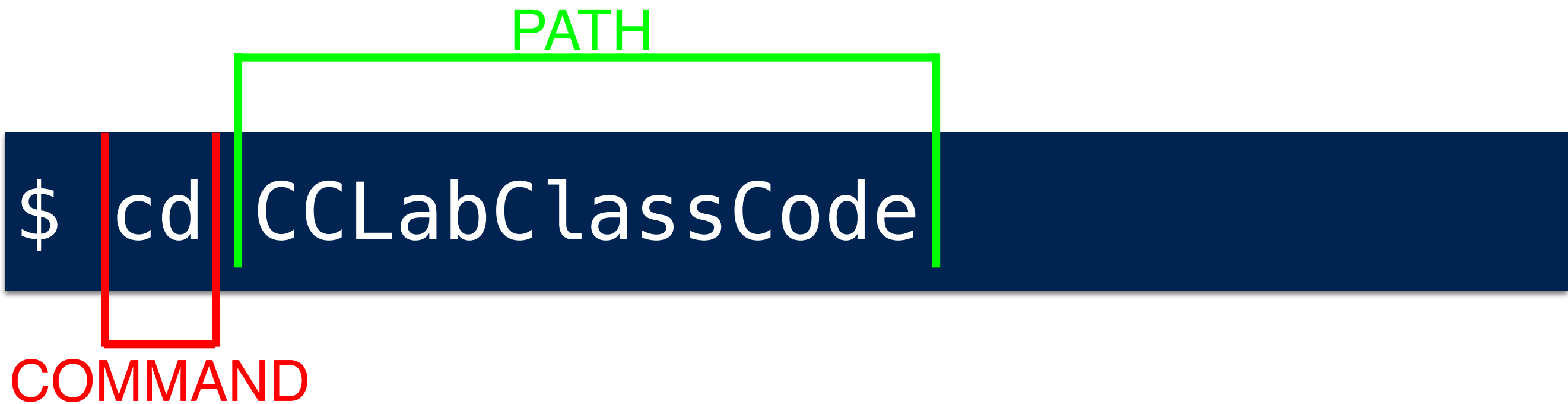


A diagram illustrating the components of a terminal command. The command is displayed on a dark blue background. A red bracket underlines the word 'rmdir', which is labeled 'COMMAND' in red text below it. A green bracket underlines the path 'CCLabClassCode', which is labeled 'NAME' in green text above it.

```
$ rmdir CCLabClassCode
```

`rmdir` means “remove directory”

# TERMINAL + GIT



A diagram illustrating a terminal command. The command is displayed on a dark blue background. A red bracket under the word 'cd' is labeled 'COMMAND'. A green bracket over the path 'CCLabClassCode' is labeled 'PATH'.

```
$ cd CCLabClassCode
```

COMMAND

PATH

# TERMINAL + GIT

```
$ pwd
```

pwd means “print working directory”

# TERMINAL + GIT

you should see something like...

```
$ /Users/gabor/TNS/CCLab/  
CCLabClassCode
```



# TERMINAL + GIT

```
git --version
```

# TERMINAL + GIT

```
$ git version 1.8.2.1
```

# TERMINAL + GIT

```
$ git init
```

# TERMINAL + GIT

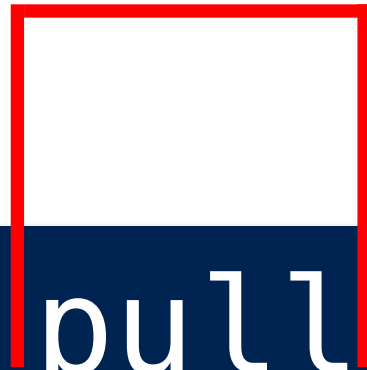
COMMAND

REPOSITORY NICKNAME

```
$ git remote add classcode  
https://github.com/gaborparsons/  
CCLAB-Fall2015
```

# TERMINAL + GIT

COMMAND



```
$ git pull https://github.com/  
gaborparsons/CCLAB-Fall2015  
master
```

BRANCH



# TERMINAL + GIT

check the folder

# TERMINAL + GIT



**LOCAL  
STORAGE**



# LOCAL STORAGE

With local storage, web applications can store data locally within the user's browser.

# LOCAL STORAGE

With local storage, web applications can store data locally within the user's browser.

# LOCAL STORAGE

- Replaces cookies
- More secure
- Large amounts of data
- Info never transformed to server

# LOCAL STORAGE

## Main Expressions

- `localStorage.setItem()`
- `localStorage.getItem()`
- JSON data format (attribute/value)
- `JSON.parse()`
- `JSON.stringify()`

**LOCAL  
STORAGE  
IN 11 LINES**

# LOCAL STORAGE

after `var taskArray = [ ];`

```
//update local storage
var updateLocalStorage = function(){

    //stringify the task array and save it as 'taskList' in local storage
    var taskListString = JSON.stringify(taskArray);
    localStorage.setItem('taskList', taskListString);

};
```

# LOCAL STORAGE

displayTasks( )

```
//check if there is a taskList in localStorage  
var taskListString = localStorage.getItem('taskList');  
if(taskListString){  
    taskArray = JSON.parse(taskListString);  
};
```

# LOCAL STORAGE

saveTask( )

```
//add the value to the taskArray  
taskArray.push(newTask);
```

```
//update local storage  
updateLocalStorage();
```




```
//update your task view
```



# LOCAL STORAGE

deleteTask( )

```
//go to the taskNumber position of taskArray and remove one object  
taskArray.splice(taskNumber, 1);  
  
//fire update local storage  
updateLocalStorage();  
  
//update tasks to show new array
```



# LOCAL STORAGE

`init( )`

`add displayTasks( )`

# JAVASCRIPT LIBRARIES

# LIBRARIES

A JavaScript library is a bunch of pre-written JavaScript files that make it easier to do cool stuff.

# LIBRARIES

Libraries contain objects + functions that you can reference the same way you would reference objects + functions that you wrote.

# LIBRARIES

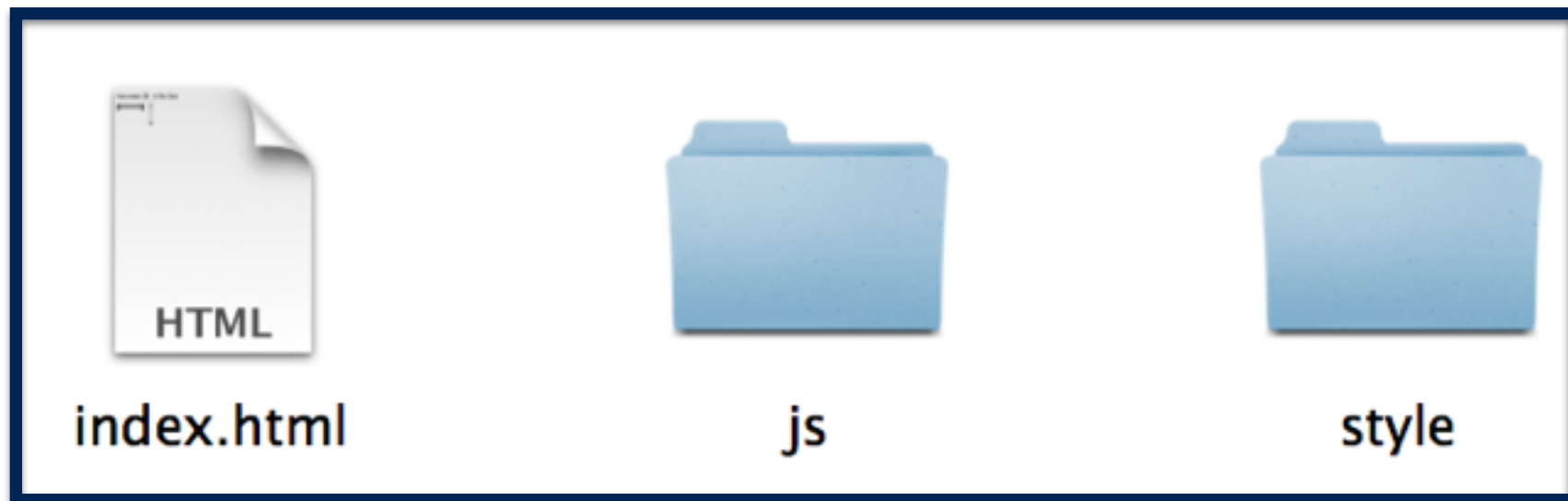
Libraries contain **objects + functions** that you can **reference** the same way you would reference objects + functions that you wrote.

# LIBRARIES

Libraries contain objects + functions that  
you can reference **ALMOST** the same way  
you  
would reference objects + functions  
that you wrote.

# LIBRARIES

## ***YOUR ROOT FOLDER***





# LIBRARIES

## ***YOUR STYLE FOLDER***



style.css

# LIBRARIES

## ***YOUR SCRIPTS FOLDER***



library\_name



main.js

# LIBRARIES

## YOUR HTML FILE




```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>My Awesome Web App</title>
  <link rel="stylesheet" type="text/css" href="style/style.css">
  <script src="js/library_name/main_library_file.js"></script>
</head>
<body>

  <div id="myAwesomeWebApp">

    <!-- Your Site Content -->

  </div>

  <script src="js/main.js"></script>
</body>
</html>
```



**LET'S TRY IT!**

# LIBRARIES

*jQuery*

*<https://jquery.org/>*

# LIBRARIES

## jQuery Expressions

- jQuery handler: `$()`
- `$(document).ready`
- `$('element')`
- `$('#id')`
- `$('.class')`
- `val()`, `empty()`, `click()`

# LIBRARIES

Many libraries can be found at  
**[jsdb.io](https://jsdb.io)**

# ASSIGNMENT



- 1. GET YOUR CODE UP AND RUNNING**
- 2. GO TO JSDB.IO + CHOOSE A LIBRARY**
- 3. MAKE A NEW APP THAT USES  
JQUERY + YOUR NEW LIBRARY**