

# JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

---

**JAVASCRIPT DEVELOPMENT**

---

# **THE COMMAND LINE**

# LEARNING OBJECTIVES

At the end of this class, you will be able to

- › Use the most common commands to navigate and modify files / directories via the terminal window.
- › Initialize a local Git repository and push/pull changes to a remote Git repository.
- › Run basic JavaScript code on the command line using Node.

# AGENDA

- JS and web technology
- The terminal
- Git and GitHub
- Command line JS

### **Think about last class:**

- We installed software from the command line by typing commands
- We also installed software by downloading an installer, double-clicking it, and following the prompts

# ACTIVITY

---



## EXERCISE

### KEY OBJECTIVE

---

- ▶ Use the most common commands to navigate and modify files / directories via the terminal window.

### TYPE OF EXERCISE

---

- ▶ Turn and Talk

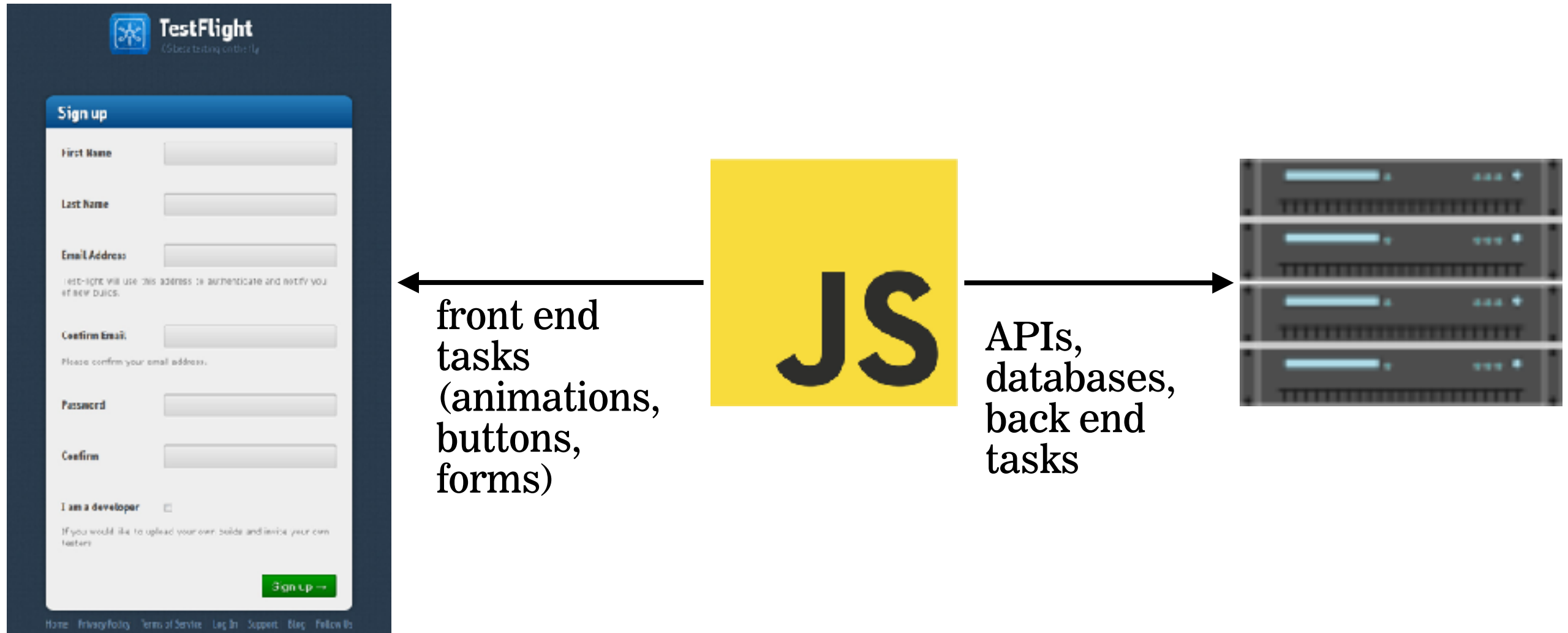
### TIMING

---

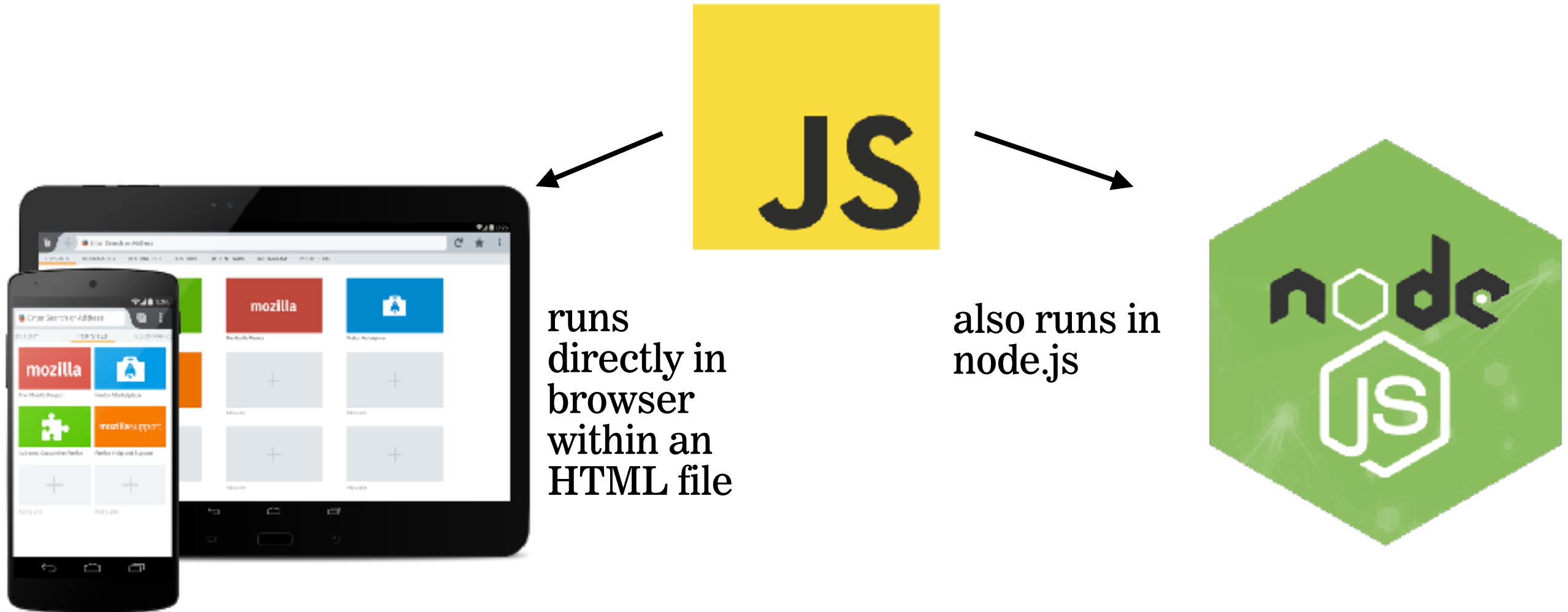
*4 min*

1. List at least 2 advantages to using the command line.
2. List at least 2 disadvantages to using the command line.

## WHAT CAN JAVASCRIPT DO?

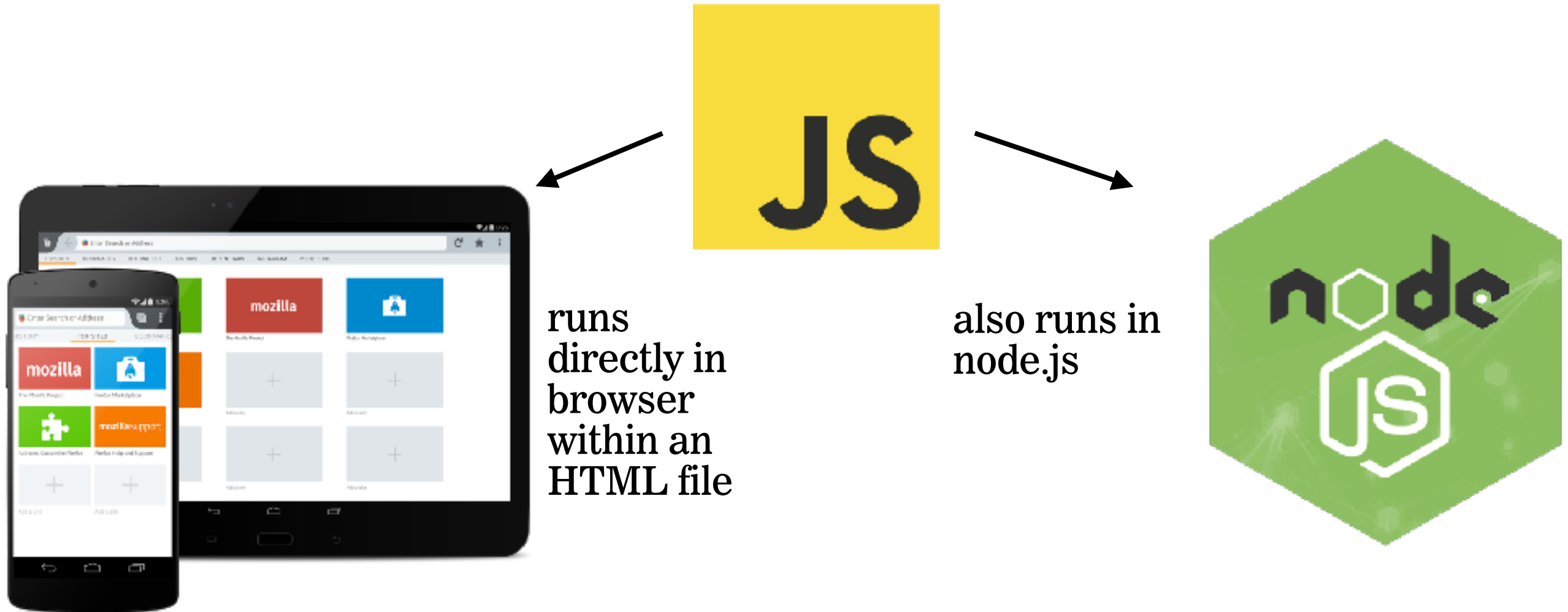


## VERY FEW STEPS TO RUN

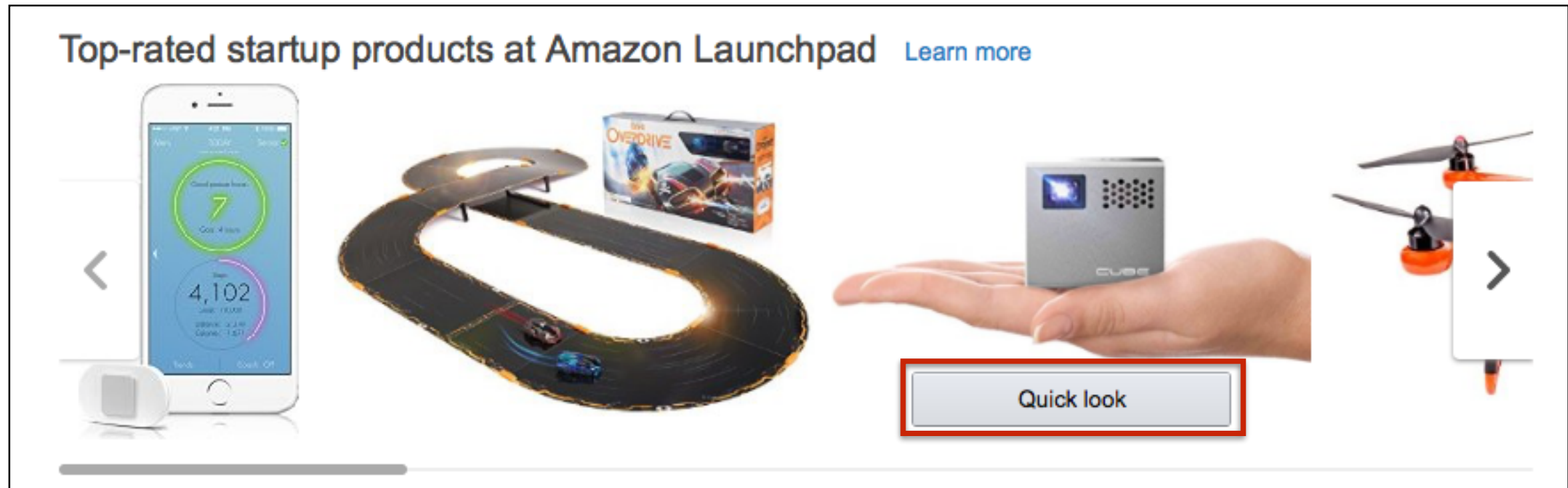




## AND WORKS EVEN WHEN COMPUTERS ARE OFFLINE



# HIGHLY RESPONSIVE INTERFACES



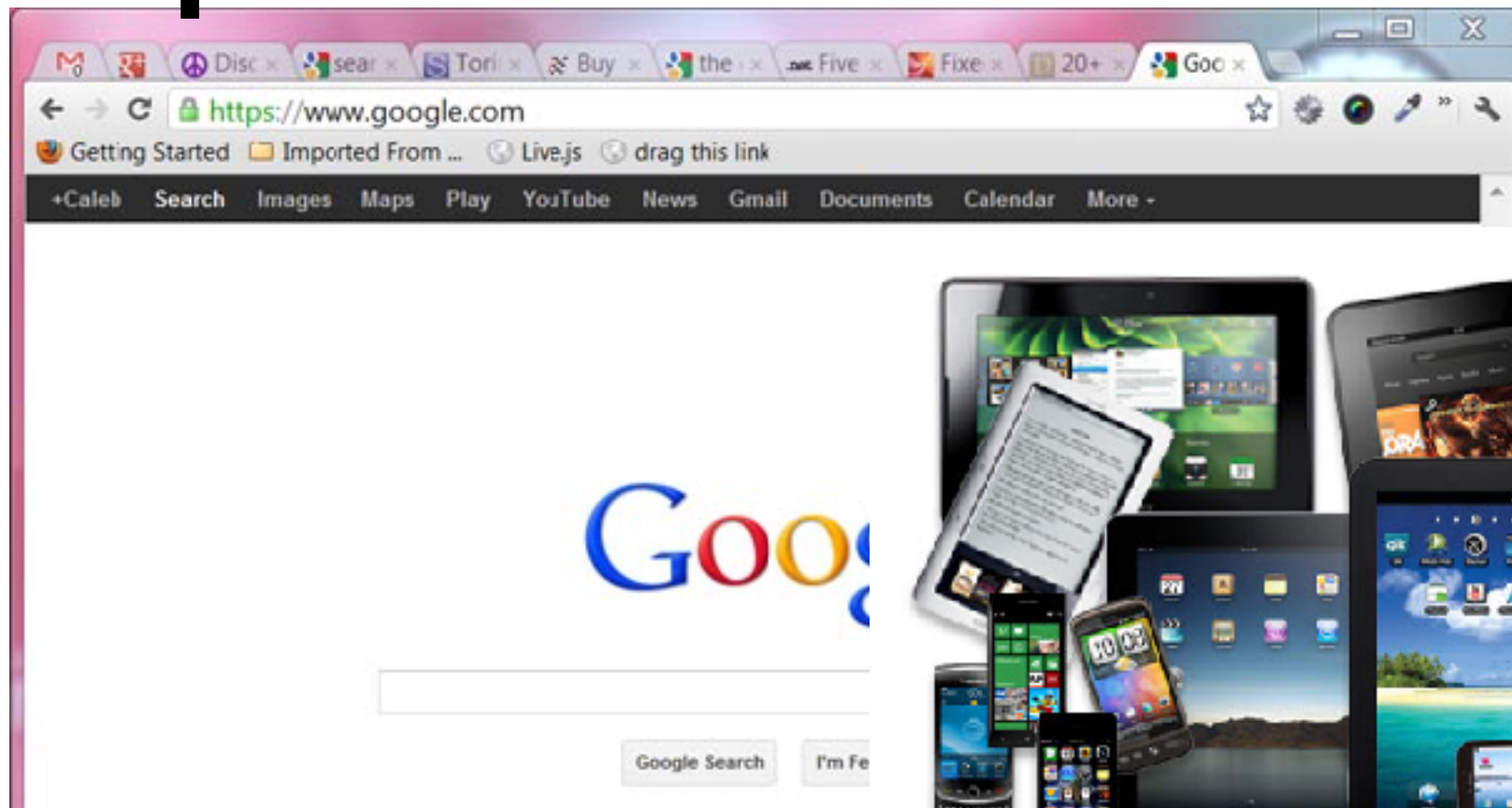
# LOAD ADDITIONAL CONTENT WHEN USER NEEDS IT (AJAX)



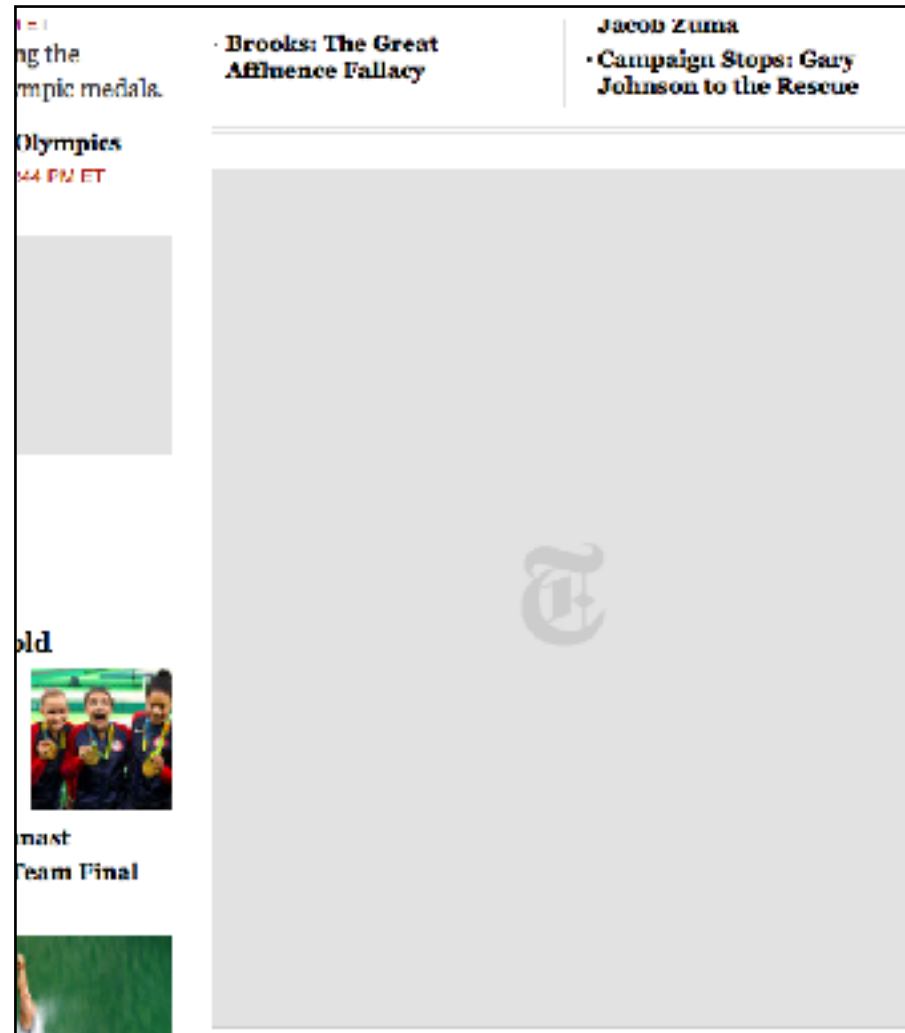
# **WHAT ELSE CAN JAVASCRIPT DO?**

- Determine your browser functional limitations and react accordingly (progressive enhancement)
- Power website backends and physical devices (node.js)

# DRAWBACK: The environment in which JavaScript operates is unknown



# DRAWBACK: JavaScript can be disabled



# Node.js

# Node.js

- A definition (from Wikipedia):
  - In software development, Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications.
- Enables JavaScript on the server (the backend)
- Written in C, C++, and JS (so, not a JS framework)
- Interprets JS using Chrome's V8 engine
- Module driven; see Node Package Manager (npm)
- All about non-blocking, asynchronous input/output



# **Node.js**

- We will not be using Node.js as a web server (backend) - see Firebase
- We will be taking advantage of Node's command line interface
- Allows us to run JavaScript from our terminal applications
- More at the end of class...

# JavaScript Frameworks and Libraries

# A Library

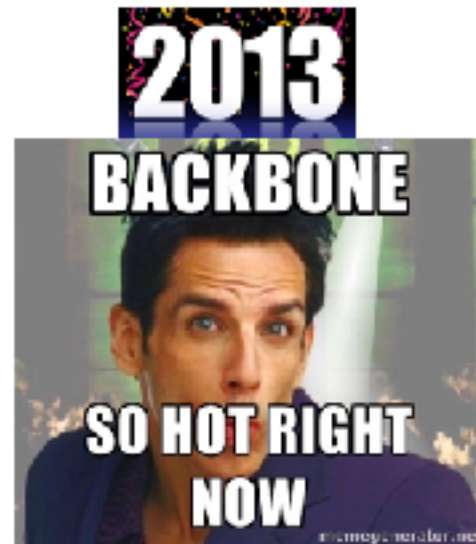
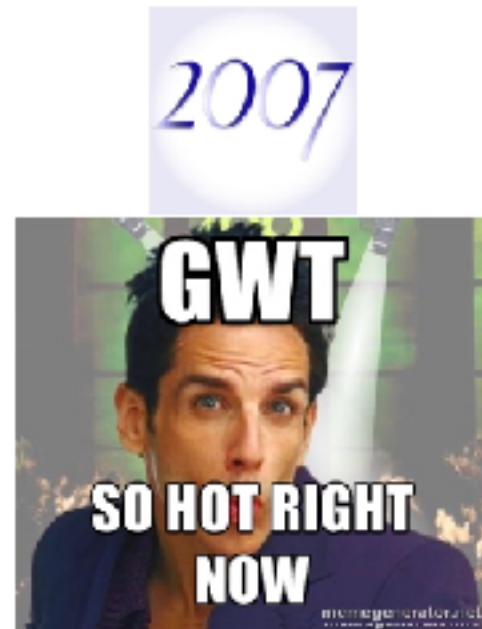
- Set of predefined functions that your code calls
- Each call performs work and returns a result (and control) to your code
- Specific, well-defined operations
- Example: jQuery

# A Framework

- Opinionated architecture for building software
- Control-flow exists, you fill in with your code
- Calls your code; is always in control
- Examples: Angular and Ember

# Libraries vs Frameworks

- The primary difference (source):
  - You call library
  - Framework calls you
- Please Note:
  - JSD focuses on the foundations of JavaScript as a programming language
  - We will be using the jQuery library
  - Opportunity towards class end for a framework intro



# The Terminal

# **INTRODUCTION TO THE TERMINAL**

- Terminal allows you to interact with your computer faster
- Terminal === Command Line === Console



# UNIX

- Family of operating systems, including all Linux systems and OS X/macOS

# **SHELL**

- A generic name for the primary program that runs inside a terminal

# **BASH**

- Bourne-again shell: a specific shell program

# ANATOMY OF THE TERMINAL

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ █
```

# ANATOMY OF THE TERMINAL

**Host (computer) name**

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ █
```

# ANATOMY OF THE TERMINAL

## Working directory (current folder)

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ █
```

# ANATOMY OF THE TERMINAL

## Username

A screenshot of a terminal window with a black background. The prompt text 'Sashas-MacBook-Pro:JS-SF-6 sasha\$' is displayed in a light blue font. The username 'sasha' is enclosed in a red rectangular box. A small grey cursor block is visible to the right of the dollar sign.

```
Sashas-MacBook-Pro:JS-SF-6 sasha$
```

# ANATOMY OF THE TERMINAL

## Bash prompt

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ █
```



# ANATOMY OF THE TERMINAL

## Command (program)

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ ls
```

# ANATOMY OF THE TERMINAL

## Argument (input)

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ ls 00-installfest
```

# ANATOMY OF THE TERMINAL

## Option

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ ls -a 00-installfest
```

# ANATOMY OF THE TERMINAL

## Output

```
Sashas-MacBook-Pro:JS-SF-6 sasha$ ls -a 00-installfest
```

```
.          .DS_Store      index.html    slides.md  
..         img            install.md
```

```
Sashas-MacBook-Pro:JS-SF-6 sasha$
```

# **(UNIX) COMMAND LINE BASICS**

---

## THE COMMAND LINE

---



# **Command line codealong**

## **For Mac**

Open the Terminal app (Applications > Utilities > Terminal)

## **For Windows**

Open the Git BASH application

# **Configure Visual Studio Code so you can call it from the command line**

## **For Mac**

<https://code.visualstudio.com/docs/setup/osx>

## **For Windows**

(no configuration required)



# LAB — COMMAND LINE

---



## EXERCISE

### KEY OBJECTIVE

---

- Use the most common commands to navigate and modify files / directories via the terminal window.

### TYPE OF EXERCISE

---

- Individual/Pairs

### TIMING

---

*10 min*

Follow the instructions posted on the class resources repo to navigate and modify files and directories using the command line.

# EXERCISE — COMMAND LINE

---



## EXERCISE

### KEY OBJECTIVE

---

- Use the most common commands to navigate and modify files / directories via the terminal window.

### TYPE OF EXERCISE

---

- Whole class brainstorm

### TIMING

---

*2 min*

1. Name a command line command and explain what it does. Let's hear from everyone at least once!

# Introduction to Git/GitHub

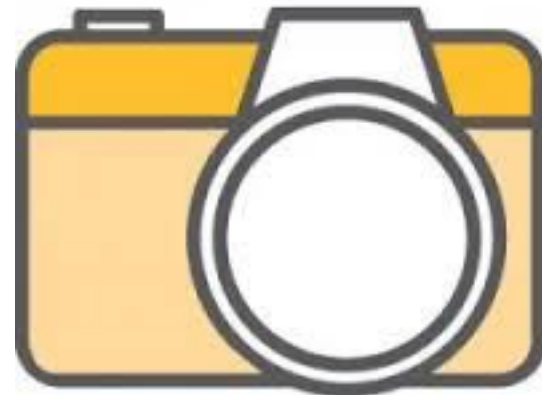
---

## WHAT IS GIT?

---

### **GIT**

- ▶ A **version control** program that saves the state of your project's files and folders
- ▶ Basically, it takes a "snapshot" of what all your files look like at a moment and stores a reference to that "snapshot"



---

## WHAT IS GITHUB?

---

### GITHUB IS A WEB APP/PLATFORM THAT

- **Platform** that makes it easy to manage git repositories.
- Similar to Dropbox or Google Drive, but for code.
- Stores a history of files and the changes that happen within each changed document.
- Hosts files on the cloud so you can share the finished product with other people.
- **Git** - the technology that Github is based on top of - was designed to allow for multiple engineers to work on the same project.

**GitHub**



---

## WHY USE GITHUB?

---



### HISTORY

- ▶ Since GitHub stores a history of the code, it allows developers to go back in time if something breaks.



### COLLABORATION

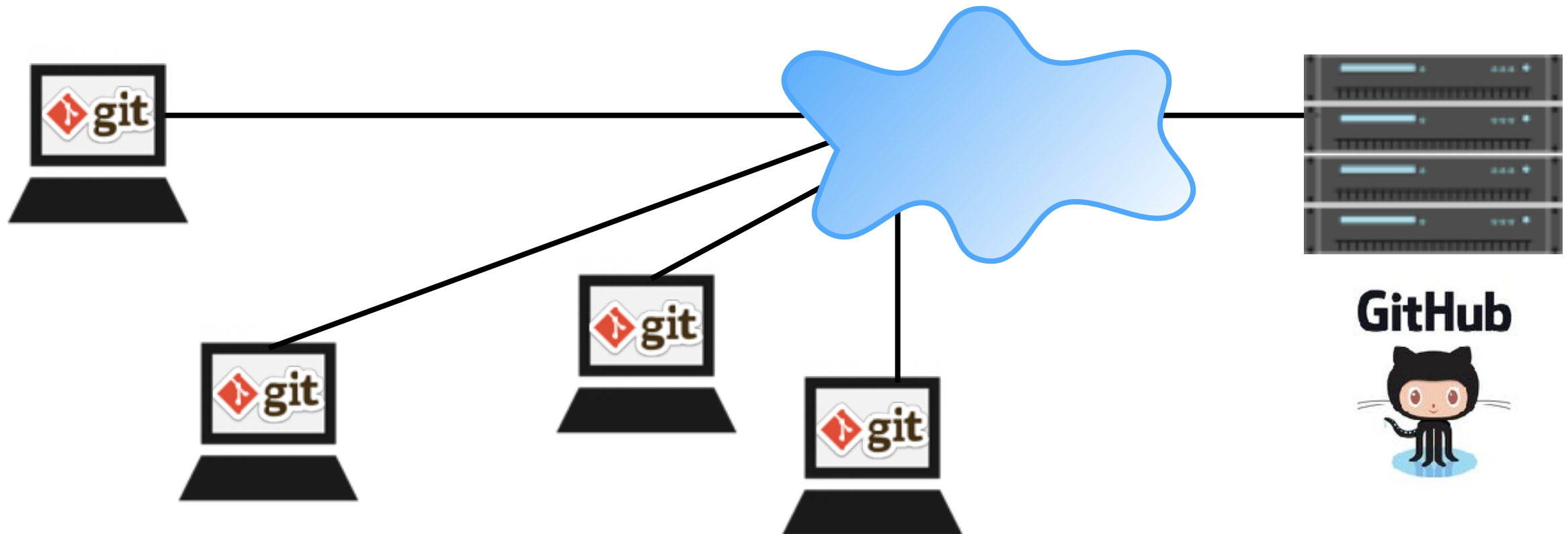
- ▶ Allows multiple developers to work on the same project. Much like Google Drive lets multiple people collaborate on the same document, GitHub allows this for code.
- ▶ You can see who worked on what.



### FEEDBACK

- ▶ GitHub allows for feedback to be given on the code which, hopefully, increases code quality.

- **Git** is version control software
- **GitHub** is a website and platform for utilizing Git in a collaborative way



# **Git/GitHub Vocabulary**

- **Repository**
- **Clone**
- **Commit**
- **Push**



# What is a repository (repo)?

- Basic element of GitHub
- Contains all of a project's files (all the code)
- One or more users can contribute to a single repository
- Repositories are either public or private
- By the end of class today, you will create your own repo



# clone

- Git command that copies/clones a **remote** repo to your machine
- This copy/clone is called a **local** repo
- Changes to the **local** repo will not affect the **remote**

# **Commit**

- Git command that creates a snapshot of changes to a repo
- Think of it as "saving" your changes
- Contains a message describing the changes made

# Push

- Git command that sends your commits (saved changes) to a **remote** repository
- Allows other developers to see your changes and copy (“pull”) them to their own local repos

# How will we use GitHub in JSD7?



JS-SF-7-resources

- contains slides and start files
- you will pull changes at the start of each class

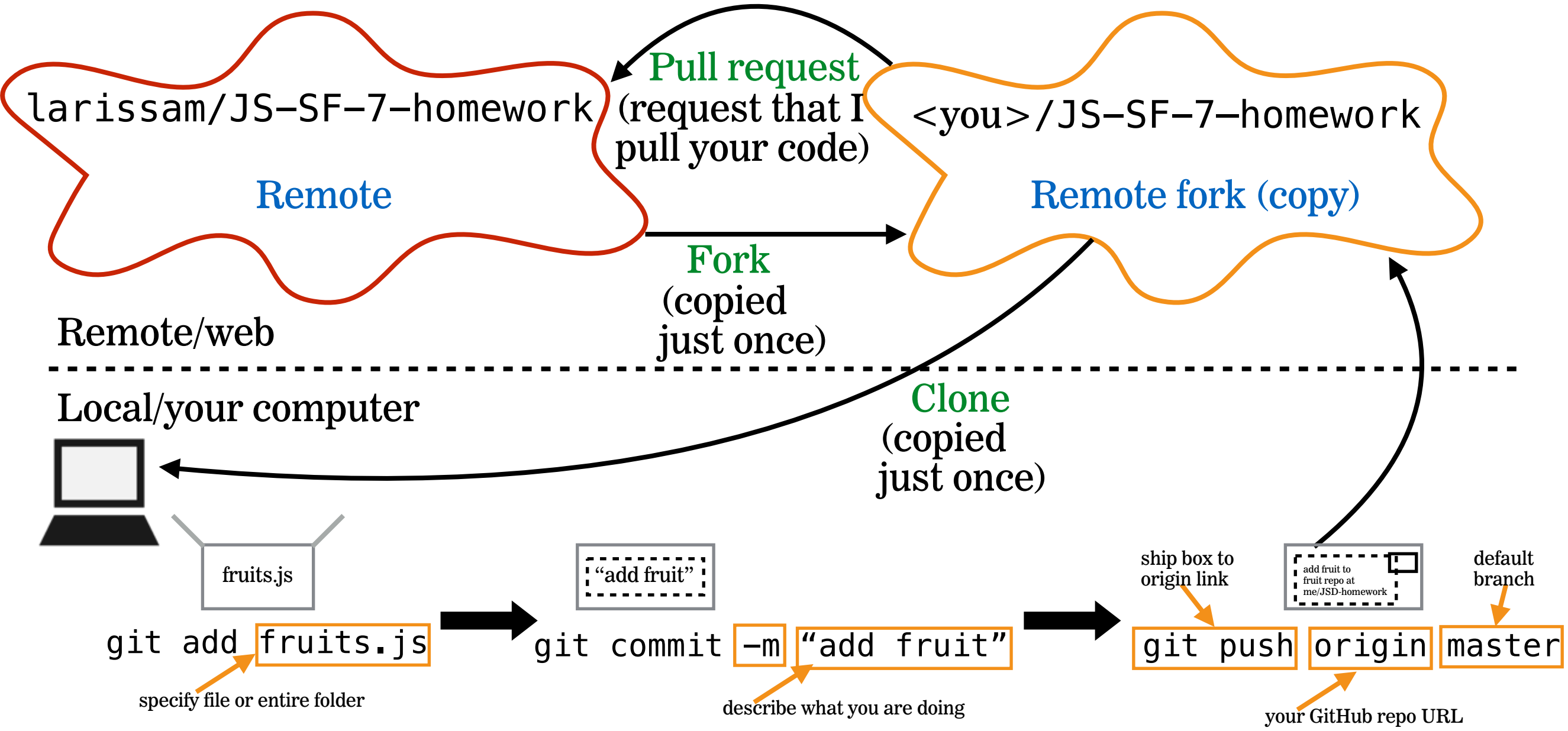


JS-SF-7-homework

- currently empty
- you will push your completed homework and receive feedback here



You will create your own additional repos for the 3 projects during the course of this class.



# **GIT COMMANDS**

---

## **GIT AND GITHUB**

---





# EXERCISE — GIT/GITHUB

---



## EXERCISE

### KEY OBJECTIVE

---

- Initialize a local Git repository and push/pull changes to a remote Git repository.

### TYPE OF EXERCISE

---

- Individual/Pairs

### TIMING

---

*3 min*

1. What command do you use to initialize a local Git repository? (Hint: Check the cheat sheet.) What does initializing do?
2. What command do you use to push changes to a remote Git repository? What does pushing do?
3. What command do you use to pull changes from a remote Git repository? What does pulling do?
4. BONUS: Draw a diagram illustrating all 3 commands

# Intro to Node.js and command line JS

### THE TERMINAL IS THE SWISS ARMY KNIFE OF PROGRAMMING



# **How is Node different from JS in the browser?**

- No browser-specific functionality
- Same JS engine as Chrome

# What is Node good for?

- Creating a backend server for a web application
- Running a script to do data analysis
- File management
- Making command line programs

# **Ways to run Node**

- Interactive command line
- Run a file

# Executing JavaScript code

# Let's write some JavaScript!





# Variables

- Containers that allow us to store values
- Let us tell our program to remember values for us to use later on
- The action of saving a value to a variable is called **assignment**

# **Variable declaration**

- Statement saying that we wish to create a variable

# **Variable assignment**

- Specifying the value we wish to assign to a variable

# **Variable assignment and declaration**

- ▶ We can do both in a single statement

# **console.log**

- Logging to the console is how we print things out for our own inspection

# **Inspecting variables**

```
console.log(y)
```

# **When do you use `console.log`?**

- When you are developing a program and need help figuring out what's going on (aka debugging)
- When you want to print things to the command line

---

## THE COMMAND LINE

---





# **Exit the Node console**

CTRL + c twice

# EXERCISE — GIT/GITHUB

---



## EXERCISE

### **KEY OBJECTIVE**

---

- Run basic JavaScript code on the command line using Node.

### **TYPE OF EXERCISE**

---

- Turn and talk

### **TIMING**

---

*3 min*

1. What is Node?
2. What did we use it for today?
3. BONUS: How else can it be used?

# **LEARNING OBJECTIVES – REVIEW**

- Use the most common commands to navigate and modify files / directories via the terminal window.
- Initialize a local Git repository and push/pull changes to a remote Git repository.
- Run basic JavaScript code on the command line using Node.

## **Next class preview: Data Types**

- Describe the concept of a "data type" and how it relates to variables.
- Declare, assign to, and manipulate data stored in a variable.
- Create arrays and access values in them.
- Iterate over and manipulate values in an array.

# Exit Tickets!

# Scheduling

- Snack rotation
- Happy Hour (GA buys the first round!)

## **See JS-SF-7 repo for**

- Pre-reading (optional)
- Additional resources on today's topics

# **Q&A**