

CMPM 120

---

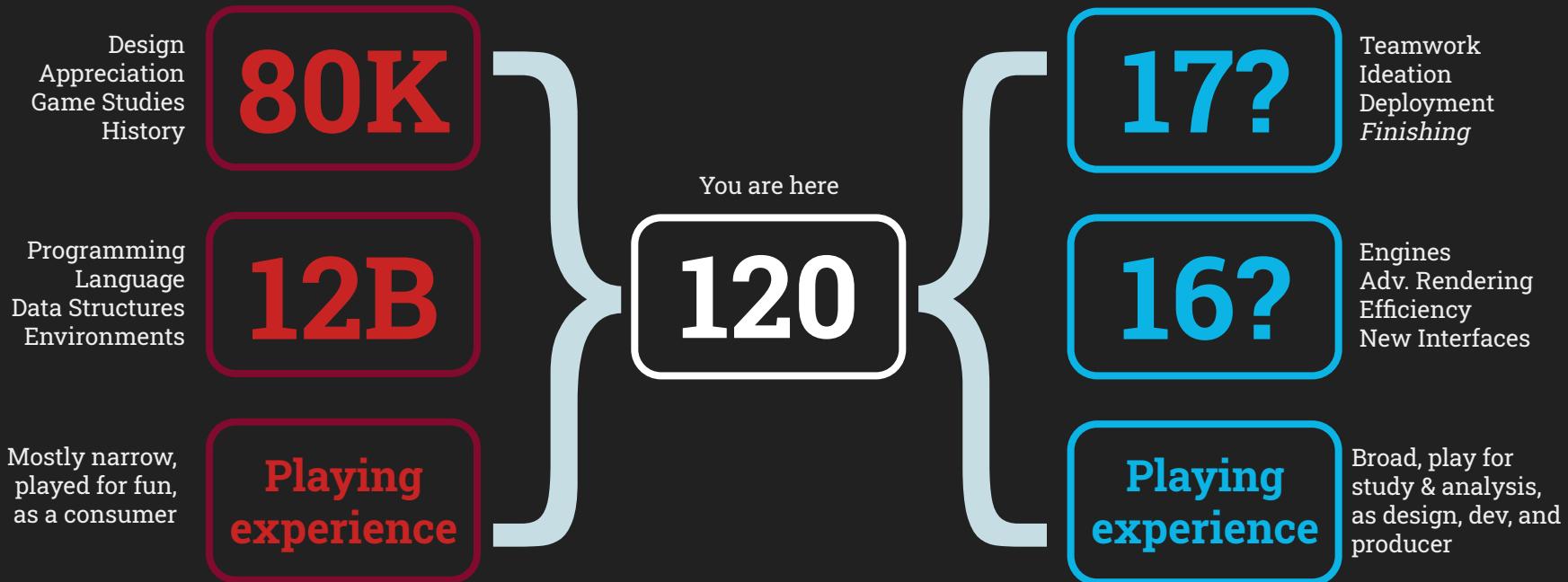
# Game Development Experience

# CMPPM 120

---

Be prepared to ask  
me a question about  
the class

# Where you are



# Why You Are Here

- Learn the **basic principles** of game programming and put them into practice
- Learn how to do the low-level implementation so we can turn **ideas** into **working games**
- Learn how **technology and teamwork** affect game design (PLO 7 & 8)

# The Team

## Isaac Karth

- he/him
- Teaching CMPM 120
- [ikarth@ucsc.edu](mailto:ikarth@ucsc.edu)
- Computational Media PhD candidate (I do procedural generation stuff)

## Dayna Diamond

- Teaching AGPM 120

# Who are you?

---

If you think of yourself as a programmer:  
What is one artwork that you like?

If you think of yourself as an artist:  
What is one programming element that  
you like?

<https://forms.gle/qRAqjxNeVN3mGNDD8>

# Who are you?

---

If you think of yourself as a programmer:

Albrecht Dürer, *Young Hare*



If you think of yourself as an artist:

[list\_item for list\_item in python\_list\_comprehension]

# Who are you?

---

What is one thing about you or your interests that has nothing to do with games?

<https://forms.gle/qRAqjxNeVN3mGNDD8>

# Who are you?

---



# Some Bad News

---

Programming  
Videogames is  
Difficult

Especially in **5 weeks**,  
in Summer,  
and working in a team.

# 5 Weeks is Short

---

We're  
speedrunning

120

```
var Environment = {  
    code: [ "HTML5", "CSS", "JavaScript"] ,  
    framework: "Phaser" ,  
    collaboration: [ "git", "GitHub" ] ,  
    editor: [ "VSCode" ] ,  
    server: "Python"  
}
```



## Desktop and Mobile HTML5 game framework

A fast, free and fun open source framework for Canvas and WebGL powered browser games.

DOWNLOAD & GET STARTED  
Download or Fork via Github



## PHASER FEATURES

### WEBGL & CANVAS

PRELOADER

PHYSICS

SPRITES

GROUPS

ANIMATION

PARTICLES

CAMERA



### INPUT

SOUND

TILEMAPS

DEVICE SCALING

PLUGIN SYSTEM

MOBILE BROWSER

DEVELOPER SUPPORT

WEB FIRST

# Why Phaser?

- **Free!** (vs freemium)
- **Fast!** (computational: running smoothly on almost any device with a modern browser)
- **Instructor choice!** (has worked for CGD+AGPM students in the past)
- **2D!** (finish and deploy your games faster, learn sooner)
- Actively supported and documented for **public use!** (not a local teaching toy)
- Lots of stuff “**for free**” (versus made by you in CMPM 164)
  - Game loop
  - Scene management
  - Physics, etc.

# Some of the Languages I've Used

- ActionScript 2
- ActionScript 3
- BASIC
- Bash
- C
- C#
- C++
- Clojure
- ClojureScript
- Common Lisp
- DOS batch files
- HTML
- Inform 6
- Inform 7
- JavaScript
- Lingo
- Logo
- PHP
- Processing
- Python 2
- Python 3

# Some of the Game Engines/Platforms I've Used

DirectX

P5.js

XNA

Quil

Unity

Flixel

Unreal

Scratch

Ogre

NeoAxis

Phaser

...and probably a bunch of stuff I'm forgetting.

Processing

Flash

# There is no perfect...

---

...no perfect language

...no perfect framework

...no perfect engine

The sooner you learn this, the better.

You should be grateful  
I'm not forcing you to  
learn ClojureScript.

# Why VSCode?

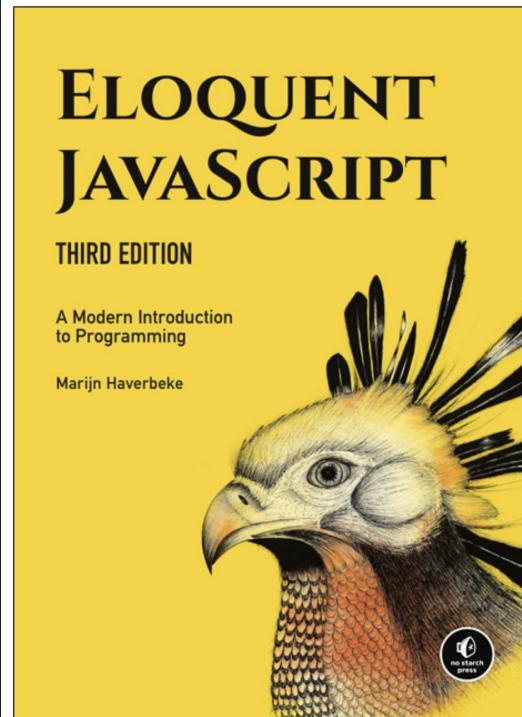
- Consistent experience across platforms (including Chromebooks)!
- Industrial-strength! (we'll use a tiny bit of it)
- Has features specific to the web platform (HTML/CSS/JS)
- Experimental [Live Share](#) feature (real-time collaboration like Google Docs)

# Why git / GitHub?

- Industrial-strength! (we'll use a tiny bit of it)
- Publish playable versions of your games easily!
- This is how we'll grade your projects in this class!
- If you haven't already, sign up for their [student benefits program](#) (free Pro account plus free stuff from partner companies: e.g. domain name registration)

**Note: You don't have to use your real name for GitHub**

# Do we need a textbook? Nah, just



ELOQUENT JAVASCRIPT  
3RD EDITION (2018)

## CONTENTS

- Introduction
- 1. Values, Types, and Operators (Part 1: Language)
- 2. Program Structure
- 3. Functions
- 4. Data Structures: Objects and Arrays
- 5. Higher-order Functions
- 6. The Secret Life of Objects
- 7. Project: A Robot
- 8. Bugs and Errors
- 9. Regular Expressions
- 10. Modules
- 11. Asynchronous Programming
- 12. Project: A Programming Language (Part 2: Browser)
- 13. JavaScript and the Browser
- 14. The Document Object Model
- 15. Handling Events
- 16. Project: A Platform Game
- 17. Drawing on Canvas
- 18. HTTP and Forms
- 19. Project: A Pixel Art Editor
- 20. Node.js (Part 3: Node)
- 21. Project: Skill-Sharing Website

If prefer to learn from video, many students have learned arts-relevant JavaScript skills from the videos on the YouTube channel [The Coding Train](#).

# This course is specific to 2021.

- Back in 2011:
  - C# / Microsoft XNA
  - PHP / ActionScript / Adobe Flash
- Today in 2021:
  - HTML/CSS/JavaScript focus
  - Phaser framework
- Someday in 2031:

The details change quickly:

- How do I draw a colored rectangle?
- How do I run code when someone presses a button?
- How do I made an end-of-level score summary screen?
- How does my game make noise?
- How do I share my game with a friend?

But the **concepts** and **development process** are quite stable.

Your game's scope:  
small



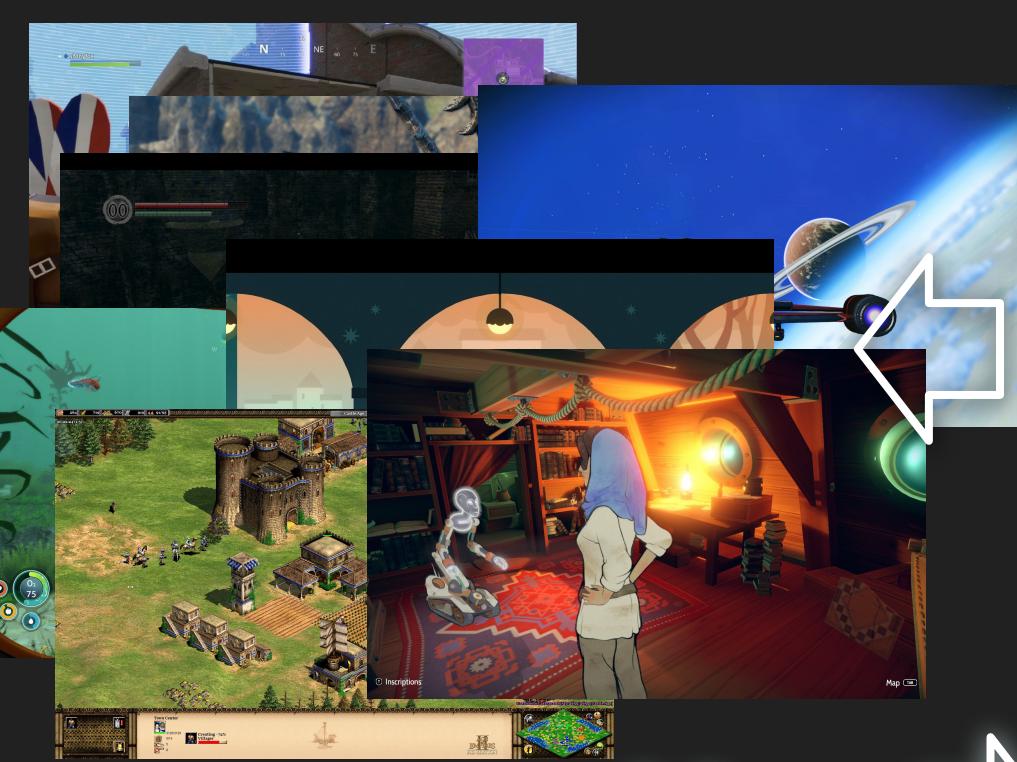
Your game's scope:  
tiny



# Finish your game.

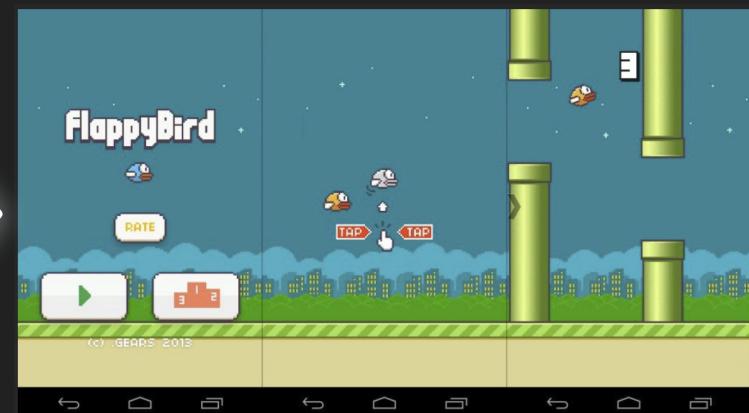


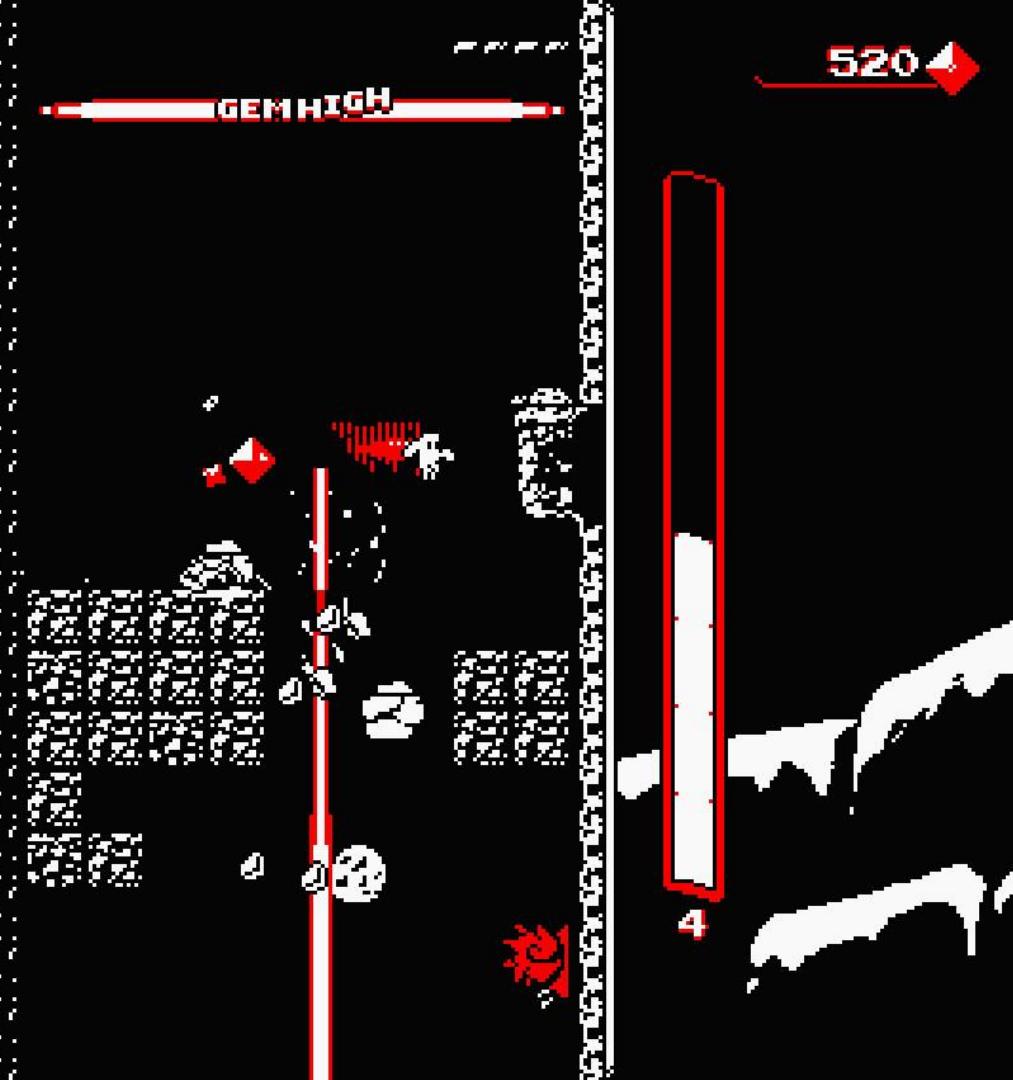
<https://makegames.tumblr.com/post/1136623767/finishing-a-game>



No

Yes





Ambitiously small

Mechanic-centric

Well-structured

Expressive

Idea-driven

Achievable

2d :)

# Think arcade games or **Flash games** from **2009**



# Some Good News

---

Every one of you  
can program a  
videogame

(And we have proof)



Lazy River: an endless drunker

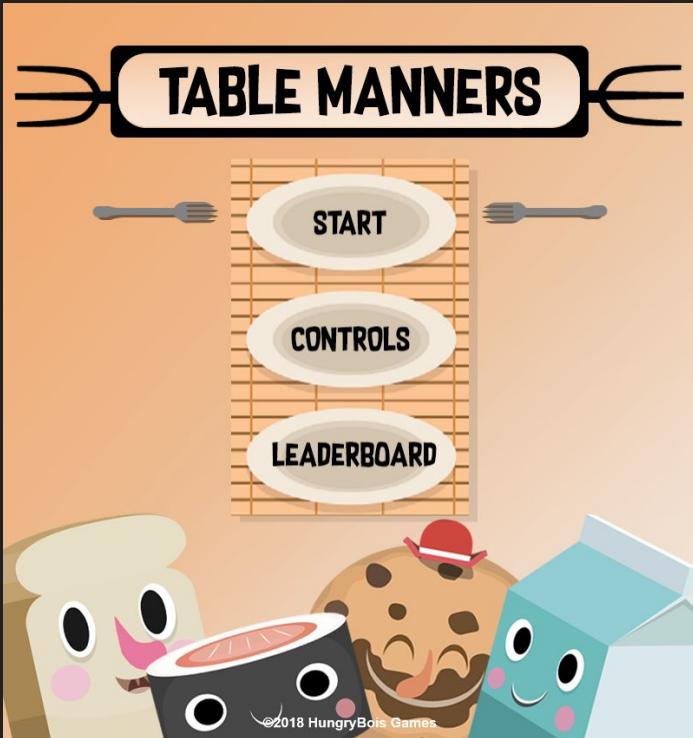


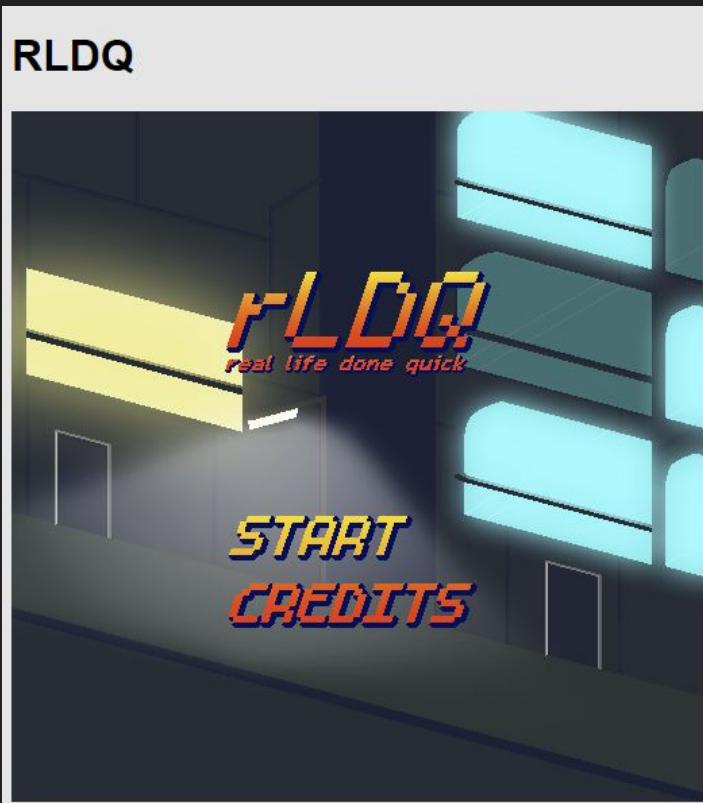
Table Manners: a game about stealing sushi



**Gravobot:** a 2D puzzle platformer with a gravity orb mechanic



Tiny Steps: a storybook game wherein you play as the CUTEST mouse



rLDQ: a minigame collection of mundane daily tasks

## The Light



The Light: a survival typing game (!?)

# Some More Good News

---

Any style or genre  
you choose is fine  
with us\*

\*As long as you do so with  
creativity, thoughtfulness,  
and professionalism

# Simple 2D games



[Celeste](#) (2015; for PICO-8), a 4-day project by experience developers; [later reimplemented](#) and published for many different platforms.

In this class, your team games can\*  
be in any style or genre that you  
choose.

\*as long as you make your game with  
creativity, thoughtfulness, and empathy

*Make something you'd be proud to invite  
someone to play one year later.*



The Runs: a very mature game



StumpJumper: a commercial UCSC game

**SPECIALIZED**

BIKES EQUIPMENT INSIDE SPECIALIZED

Stumpjumper: The Video Game

You live to rip, but Big Corporation wants to harsh your vibe and plaster condos over the hometown trails. No way! Lucky for you, there's a mysterious crew in your corner, bent on helping you win the big race for the future of the trails. You ready? Time to show these two-wheeled turkeys who really runs the show out here in Stumpjumper Country.

\*For the optimal game experience on iOS, you'll need an iPhone 6 or newer running iOS11 or newer. For iOS12 users, please be sure to disable the browser tabs feature in Safari. And if you're experiencing speaker sound issues, make sure your phone isn't set to vibrate. Oh yeah, and have fun.

PLAY THE GAME

Team of six, 10-minute game, 6 months, \$25K Budget

# Course Structure

---

# Canvas Resources

Syllabus: <https://canvas.ucsc.edu/courses/44176>

# Schedule Overview\*

- 6/22      Introduction
- 6/24      Programming Our First Phaser Game
- 6/29      Version Control & Scenes
- 7/1        Input and Movement
- 7/6        Physics and Debugging
- 7/8        State Machines & Cameras
- 7/13      JSON, Tilemaps, Map Editors
- 7/15      Tweens & Particles
- 7/20      Special Topics
- 7/22      Final Presentations

\*This will inevitably change a bit

# Schedule Overview\*

6/22	Introduction			
6/24	Programming Our First Phaser Game	Rocket Patrol Tutorial Due	6/26	
6/29	Version Control & Scenes	Rocket Patrol Mods Due	6/29	
7/1	Input and Movement			
7/6	Physics and Debugging	Endless Runner Due	7/6	
7/8	State Machines & Cameras			
7/13	JSON, Tilemaps, Map Editors	Final Game: First Build	7/13	
7/15	Tweens & Particles			
7/20	Special Topics			
7/22	Final Presentations	Final Game Due	7/22	

\*This will inevitably change a bit

# Policies & Expectations

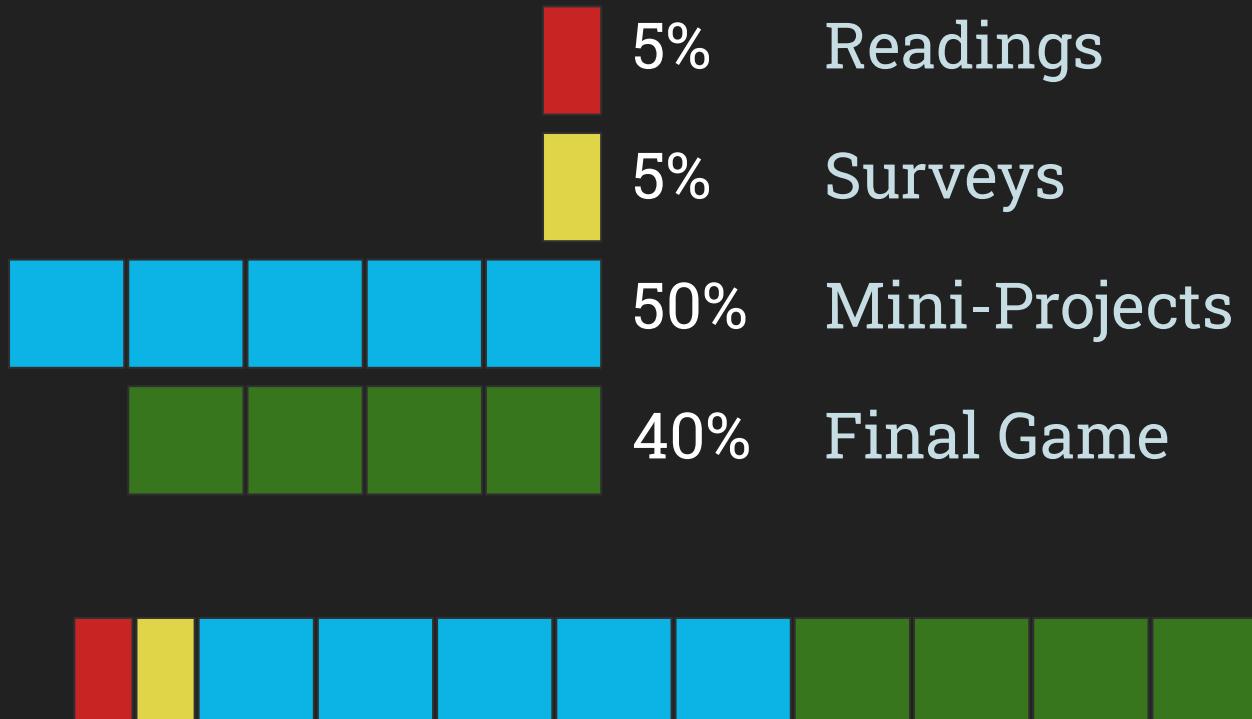
---

# There are no labs in the summer

However, I **will** have office hours:

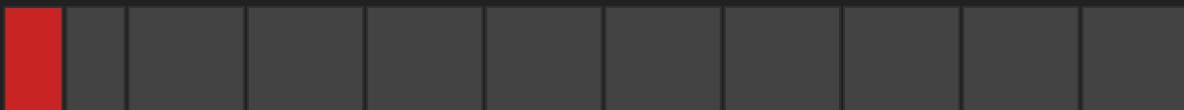
- At a time that is partially determined by your answers in the First Day Survey
- This week, I will be available on Friday, 1pm - 3pm.

# Grading



## Readings (5%)

- Designed to introduce you to the concepts we'll be talking about
- Quizzes are there to provide extrinsic feedback
  - ◆ You can retake the quizzes as many times as you need to, only the last time counts
  - ◆ The quizzes are for assessment, not evaluation



## Surveys (5%)

- Gives me feedback on what you need for the class
- Walks you through essential tasks to set up your programming environment
- You get the grade for answering the questions
- There are no right answers



# Assessment vs. Grades

# Mini-Programs (50%)

Three games:

- Follow the Rocket Patrol Tutorial
- Modify Rocket Patrol
- Make an Endless Runner (team project)

Due dates are mostly on Tuesday before AGPM 120, but check the time and dates on Canvas because with such a tight time span I couldn't make them all consistent.



# Final Game (40%)

- Make a game
- Team project (3 - 4 team members)



# Collaboration

- I won't run cheat-checking code scanners.
- I want you to read **and edit** each others code.
- Don't present someone else's work as your own.
- Imagine you were in a foreign language class.
  - Would you keep quiet in front of other students to make sure they weren't stealing your 中文? No! You'd practice together and help one another fix their mistakes. You'd develop taste for effective communication by seeing examples of less-than-perfect communication.

Easy way to credit sources and collaborators:

```
/* developed in collaboration  
with ____@ucsc.edu */
```

```
/* based on example from  
https://\_\_\_\_ */
```

# Attendance

- Attend when *you can* for maximum personal benefit and to provide an rich learning environment for others. (You are part of the show for others!)
- Catch up with Zoom recordings later when you can't.
- Zoom usage:
  - Use Zoom chat to ask questions. Okay for student's to help one another, but try not to be too distracting.
  - Consider turning on your camera. Seeing your face helps me and others see agreement and confusion.

# My Attendance



# First-day Survey (first assignment)

Let's find it on Canvas.

**We're excited you're here!**

---

**Ask me a question**

# Break

---

Your first reading assignment

---

# "Learning Web Design"

There's a quiz.

# Modern Web Development

---

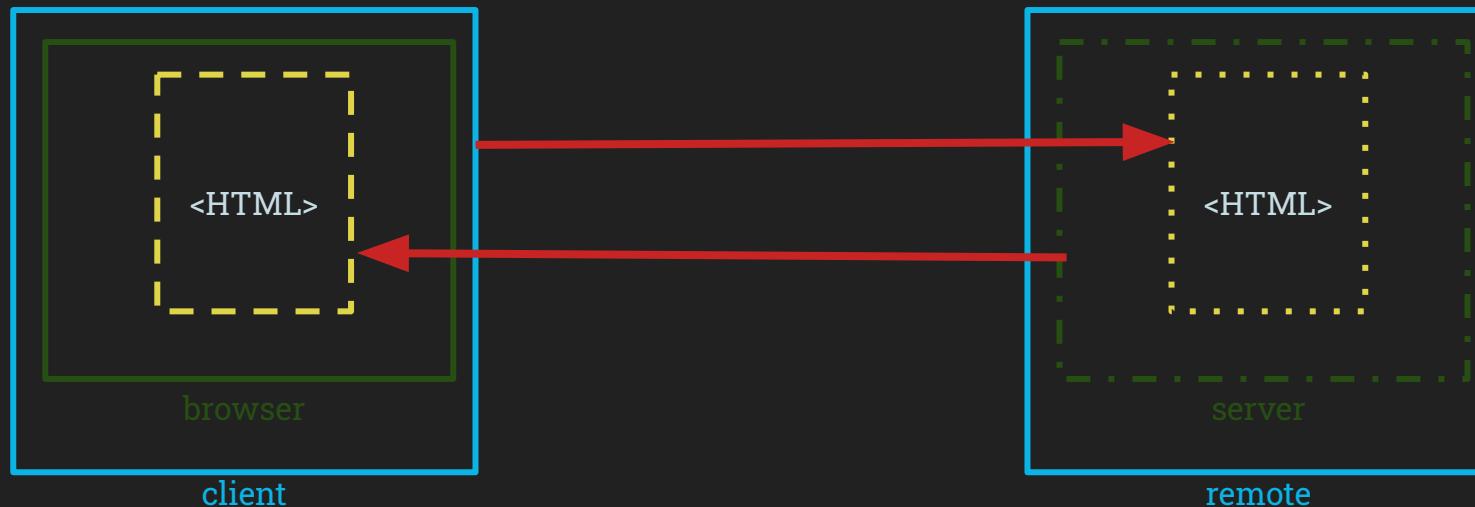
# **HTML + CSS + JS**

**HTML:** semantic layer  
(how a page is described)

**CSS:** presentation layer  
(how a page looks)

**JavaScript:** interaction layer  
(how a page behaves)

# A basic model of how the web works



# Learning Outcome

---

Be able to explain how HTML, CSS, and Javascript get turned into a web page that you can view.

# Jigsaw

---

- 1. Divide into jigsaw groups**
- 2. Each of you, pick one of the topics and read it.**
- 3. Divide into expert groups, explain topic to your expert group**
- 4. Return to jigsaw group**
- 5. Explain the topic to your jigsaw group**
- 6. Assessment Quiz**

# Jigsaw

---

Research:

[https://docs.google.com/forms/d/e/1FAIpQLSfnFFmfKy\\_PHK\\_gDlpkBN\\_8CWhCSIoTj\\_K\\_QVtoX9cGsTwhQ/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfnFFmfKy_PHK_gDlpkBN_8CWhCSIoTj_K_QVtoX9cGsTwhQ/viewform?usp=sf_link)

Read the material assigned to your number and be prepared to teach others about it.

If you already are an expert on your assigned topic, great! Here's your opportunity to teach others about it.

Even if you think you know everything about it, try to look for a detail in the reading that you think would be surprising to someone who is new to the topic.

# Jigsaw

---

Assessment:

[https://docs.google.com/forms/d/e/1FAIpQLSf2VaXMqfltTla3pT9O1jECQVyJHe1UhJL8fAnesixBIURpLA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSf2VaXMqfltTla3pT9O1jECQVyJHe1UhJL8fAnesixBIURpLA/viewform?usp=sf_link)

This is purely to give me (and you) a sense of what aspects you understand.

# Break

---

# Learning Outcome

---

Be able to implement **HTML**,  
**CSS**, and **Javascript** in a web  
page.

---

# Let's make a web page

# Let's make a simple page with HTML + CSS

HTML elements:

- <html>
- <head>
- <body>
- <h1>
- <p>
- <ul> and <li>
- <strong> and <em>
- <a>
- <img>

See also

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

CSS rules:

Selectors:

- [by tag name]
- [by class]

...

Declarations:

- Color
- Font-family

...

See also

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)

# JavaScript Essentials

---

Are you looking at these slides without access to the lecture video?

Try <http://ex-artist.com/CMPM120/Resources/JavaScript%20ES6%20Essentials.html>

# JavaScript Overview

- First developed in 1995 at Netscape (for Navigator 2.0)
- Not really related to Java
- Actually a scripting language (domain-specific for web environment)
- Relies on host for input/output (e.g., browser)
- Multi-paradigm (e.g., procedural, functional, OOP, etc.)
- Dynamic (i.e., executes at runtime)
- Loosely typed
- Standardized as ECMAScript
- Historically maligned/praised for its flexibility

# Activity plan

## Environment setup

- Install [VSCode](#).
- Create folder for today's demos
- Open folder in VSCode
- Create index.html in that folder
- Populate with basic html template
- Show inline documentation on mouseover
- Customize page title; refresh.
- Create src/main.js;  
`console.log('#deface');`

## JavaScript demos:

- Babbling in the console window
- Printing messages with `console.log`
- Variables with `let/const`
- Template strings with `label: \${var}`
- Conditionals with `if/else`
- Loops with `for/while`
- Arrays
  - Literals with [...]
  - Indexing with [] and `.length`
  - `for-of` loops / `a.forEach`
- Objects
  - Literals with (...) (key: value)
- Functions
  - `funciton(){}`
  - `=>`
  - As values
- DOM interaction (`addEventListener` / `getElementById`)



# Reading 1

---

# First Day Survey

## Reading 2

---

How is your  
programming  
environment set up?

# Reading 1

---

## Web Dev Basics

# Assignment

---

Follow the  
Rocket Patrol  
Tutorial

# Assignment: Rocket Patrol Tutorial

Complete the in-class [Rocket Patrol tutorial](#).

**Turn in two things to Canvas:**

- a link to your public GitHub repository
- a link to your published, playable game (using GitHub Pages)

*Please be sure to test your game in Chrome—that's the browser we use to grade.*

Remember our contract of trust in this class. You could clone my finished game and turn it in as your own, but you won't learn anything that way, and it'd be an odd waste of your time. Choose the TRUE path: type in your own work 

# Next Class:

---

# Watch me make a game

# Bonus Slides

---

**What**

Job title, keywords, or company

**Where**

City, state, or zip code

front end developer

Santa Cruz, CA

Find jobs

[Advanced Job Search](#)

front end developer jobs in Santa Cruz, CA

## My recent searches

front end developer - Soquel, CA

[» clear searches](#)

## Sort by:

relevance - date

## Distance:

within 50 miles

## Salary Estimate

\$100,200+ (861)

\$115,000+ (673)

\$120,000+ (582)

\$130,000+ (356)

\$140,000+ (172)

## Job Type

Full-time (962)

Contract (87)

Internship (24)

Part-time (13)

Temporary (9)

Commission (3)

## Location

Sunnyvale, CA (161)

San Jose, CA (153)

Santa Clara, CA (116)

Palo Alto, CA (98)

Mountain View, CA (96)

[+ more »](#)

## Company

Walmart eCommerce (35)

Apple (34)

Walmart (20)

New! Join Indeed Prime - Get offers from great tech companies

Page 1 of 1,045 jobs

**Software Engineer, Front End**

Google 4.5/5 3,009 reviews

Mountain View, CA

Experience with front end technologies and/or front end frameworks. As a Front End Software Engineer at Google, you will specialize in building responsive and...

[save job](#)**Front End Engineer**

Amazon.com Services, Inc. 4.5/5 34,428 reviews

Cupertino, CA

You will design, build, and operate web services and web front end at scale, interface with front-end and back-end teams, and deliver the plumbing in-between....

[save job](#)**Front-End JavaScript Developer (Contract)**

LG Electronics 4.5/5 2,069 reviews

Santa Clara, CA

Front-End JavaScript Developer, Advanced Platform Lab of LG ARC (America R&D Center) in Santa Clara, CA, is looking for passionate and talented JavaScript &...

3 days ago [save job](#)**Front End Developer**

BlockRules

Fremont, CA

As a Front End Developer, you will work on development and implementation of features such as smart contracts, integration of crypto-exchanges, and registration...

[Easy apply](#)[save job](#)**Sr Frontend Engineer (Remote)**

Noom Inc. 4.5/5 8 reviews

United States

**\$100,000 - \$150,000 a year**

At Noom, we use scientifically proven methods to help our users create healthier

**Software Engineer, Front End**

Google 4.5/5 3,009 reviews - Mountain View, CA

**Apply On Company Site**

In school or graduated within last 9 months? We encourage you to apply to openings on the Student Jobs site

Our engineering teams bring the best of Google to our users around the world with high-impact, innovative work. Learn about career opportunities in Bangalore and Tokyo today.

Note: By applying to this position your application is automatically submitted to the following locations: Mountain View, CA, USA; San Francisco, CA, USA; San Bruno, CA, USA; Sunnyvale, CA, USA  
Minimum qualifications:

- BA/BS degree in Computer Science or related technical field or equivalent practical experience.
- 1 year of work experience.
- Experience in JavaScript, and one or more programming languages including but not limited to: Java, C/C++, Python or Go.
- Experience with front end technologies and/or front end frameworks.

## Preferred qualifications:

- Experience developing user-facing software.
- Experience with the latest and greatest web standards, including HTML5 and CSS3.
- Knowledge of web libraries and frameworks such as AngularJS, Polymer, Closure or Backbone.
- Strong sense of web design and attuned to the fundamentals of user experience.
- Familiarity with the whole web stack, including protocols and web server optimization techniques.
- An understanding of the principles of accessibility and can build products that are accessible to users with disabilities.

**About the job**

Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. We're looking for engineers who bring fresh ideas from all areas, including information retrieval, distributed computing, large-scale system design, networking and data storage, security, artificial intelligence, natural language processing, UI design and mobile; the list goes on and is growing every day. As a software engineer, you will work on a specific project critical to Google's needs with opportunities to switch teams and projects as you and our fast-paced business grow and evolve. We need our engineers to be versatile, display leadership qualities and be enthusiastic to take on new problems across the full-stack as we continue to push technology forward.

Do you want to help Google build next-generation web applications like Inbox, Gmail, Google Search, Google Maps and more? As a Front End Software Engineer at Google, you will specialize in building responsive and elegant web

**Front End Developer**

Apple ★★★★☆ 5,034 reviews - Santa Clara Valley, CA

**Apply On Company Site**

Apple is seeking a Front-End Developer to drive user experience innovations for apple.com. This developer will not only be responsible for defining the architectural strategy for front-end technologies, including **HTML5, CSS3 & JavaScript**, but for evangelizing that technology across the team and Apple as a whole.

### Key Qualifications

- Comfortable with source version control software and package managers (SVN, Git, NPM)
- Well-versed in fundamental visual and interactive design discipline
- Strive to use web standards to build solutions using semantic markup, templates (Handlebars) and SASS
- Understanding of all major browsers and the special considerations required for all various quirks
- Competent JavaScript programmer who doesn't need to rely on libraries to accomplish innovative interactions
- Aware of the interplay between JavaScript and HTML & CSS, and can dynamically create, modify, and style elements on a page with ease
- Experience with WebGL is a plus.

### Description

Lead development efforts on large scale web-based projects, ensuring robust and lasting solutions are implemented Awareness of Apple's mobile platform with the ability to build solution that take advantage of the latest iOS features while remaining performant on the latest iOS devices Maintain existing JavaScript libraries: making sure they support the engineering and creative needs of apple.com Mentor team members: Educate on software development best practices and new technologies, especially HTML5 & CSS3 Innovate: Build things that people will blog and Twitter about

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/How\\_the\\_Web\\_works](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works)

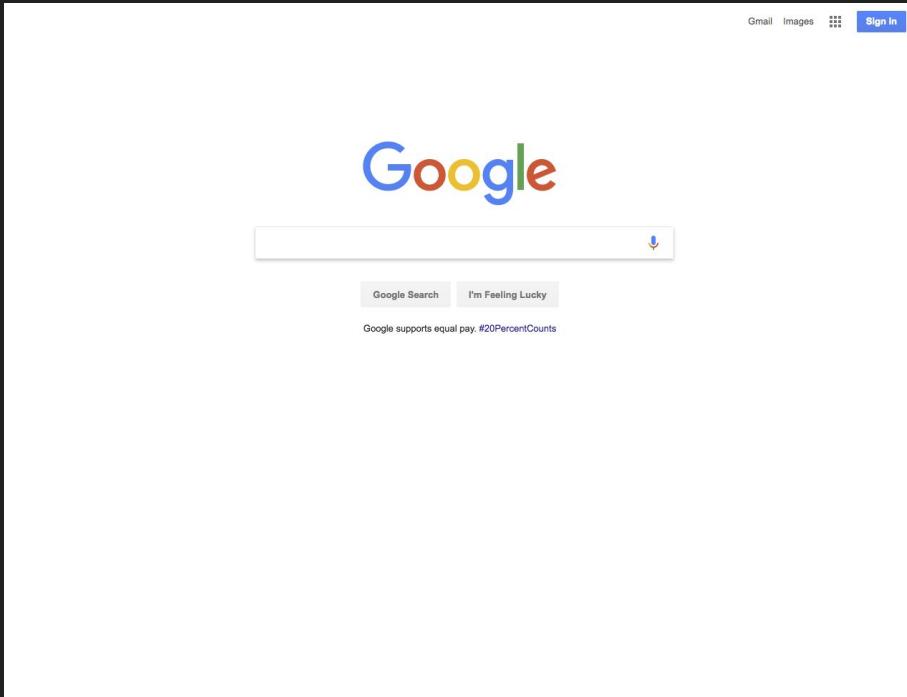
## Before we start...

---

# What the heck is the Internet?

The screenshot shows the Facebook sign-up page. At the top, there's a blue header bar with the word "facebook" in white. To the right of the logo are fields for "Email or Phone" and "Password", a "Log In" button, and a "Forgot account?" link. Below the header, a large banner on the left says "Connect with friends and the world around you on Facebook." It features three icons: a photo album icon with the text "See photos and updates from friends in News Feed.", a star icon with the text "Share what's new in your life on your Timeline.", and a magnifying glass icon with the text "Find more of what you're looking for with Facebook Search.". On the right side, the "Sign Up" heading is displayed in large, bold letters. Below it, the text "It's free and always will be." is shown. There are four input fields: "First name" and "Last name" (both with placeholder text), "Mobile number or email" (with placeholder text), and "New password" (with placeholder text). Below these fields is a "Birthday" section with dropdown menus for "Month", "Day", and "Year". A link "Why do I need to provide my birthday?" is next to the year dropdown. Underneath the birthday fields are two radio buttons: "Female" and "Male". At the bottom of the sign-up form is a small text block about account terms and policies, followed by a large green "Create Account" button. Below the button, a smaller link says "Create a Page for a celebrity, band or business." At the very bottom of the page is a navigation bar with links for English (US), Español, Français (France), 中文(简体), 中文(繁体), Português (Brasil), Italiano, 한국어, Deutsch, हिन्दी, 日本語, and a plus sign for more languages. The navigation bar also includes links for Sign Up, Log In, Messenger, Facebook Lite, Mobile, Find Friends, People, Pages, Places, Games, Locations, Celebrities, Marketplace, Groups, Moments, Instagram, About, Create Ad, Create Page, Developers, Careers, Privacy, Cookies, Ad Choices, Terms, and Help.

On the Internet, but not actually \*the\* Internet...



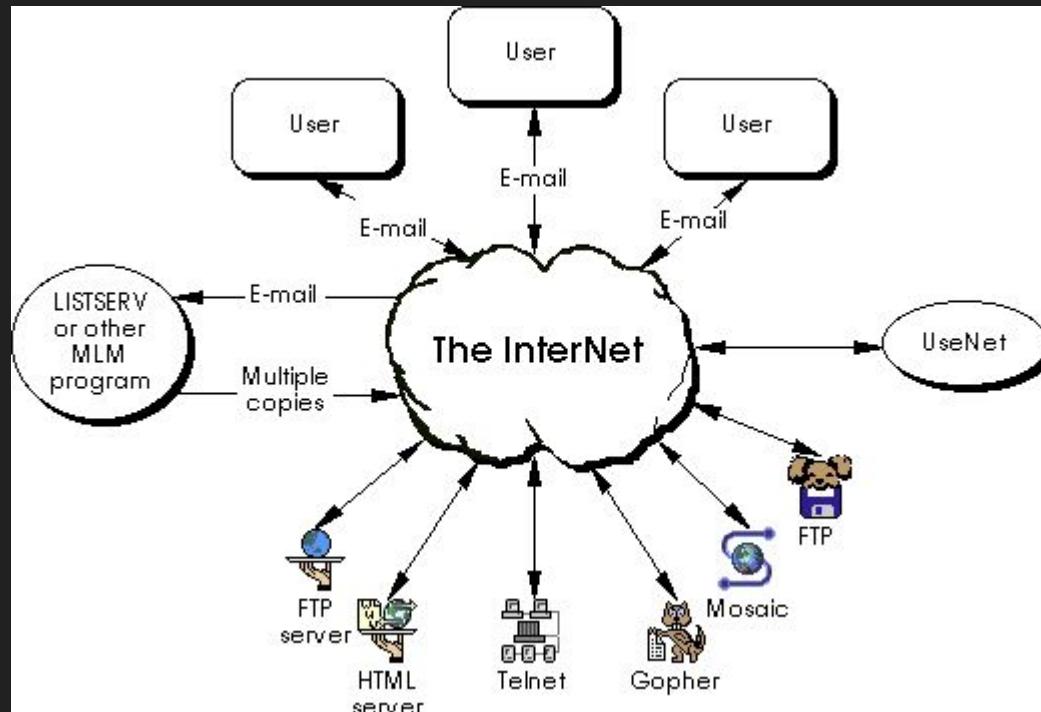
Also not the Internet...



Still not the Internet...



This *is* The Internet (but not the Internet we're talking about)



The **Internet** is a diverse network of connected computers that use a variety of standardized protocols to send and receive information

The World Wide Web is just one (of many) ways to send and receive information via the Internet.

It uses HTTP as its communication protocol, HTML as a language to describe and structure information, browsers to interpret HTML, and hypertext to link documents together

What's this stuff?

The browser window displays the w3schools.com website. The address bar shows "Secure https://www.w3schools.com". The main content area features a large "HTML" heading with the subtext "The language for building web pages". Below this are two buttons: "LEARN HTML" and "HTML REFERENCE". To the right, there is an "HTML Example" code block and a "Try it Yourself" button. Further down, there is a "CSS Example" code block and another "Try it Yourself" button. The left sidebar lists various tutorial categories: HTML and CSS, JavaScript, Server Side, Web Building, and XML Tutorials, each with its own list of sub-topics.

**HTML**  
The language for building web pages

**LEARN HTML** **HTML REFERENCE**

**HTML Example:**

```
<!DOCTYPE html>
<html>
<title>HTML Tutorial</title>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

**Try it Yourself »**

**CSS**  
The language for styling

**LEARN CSS** **CSS REF**

**CSS Example:**

```
body {
    background-color: lightblue;
}
h1 {
    color: white;
    text-align: center;
}
p {
    font-family: verdana;
    font-size: 20px;
}
```

**Try it Yourself »**

**JavaScript**

**JavaScript Example:**

```
<script>
```

# Uniform Resource Locator (URL)

User Agent  
(browser)



TUTORIALS ▾

REFERENCES ▾

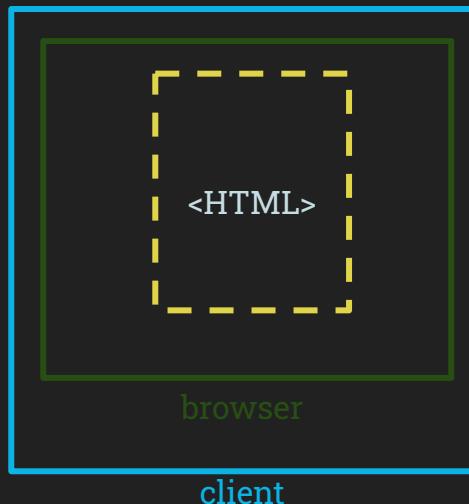
EXAMPLES

protocol://**subdomain**.domain.tld:**port-number**/**path**?**parameters**

# A basic model of how the web works



# Client



"Client-side" or "front end" applications run on our local machine

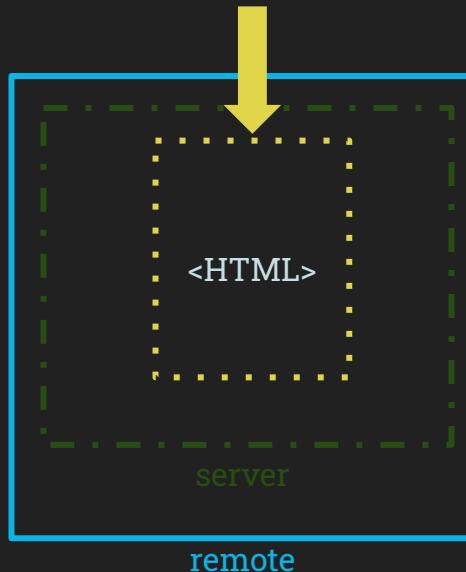
- Limited by the resources of the local machine

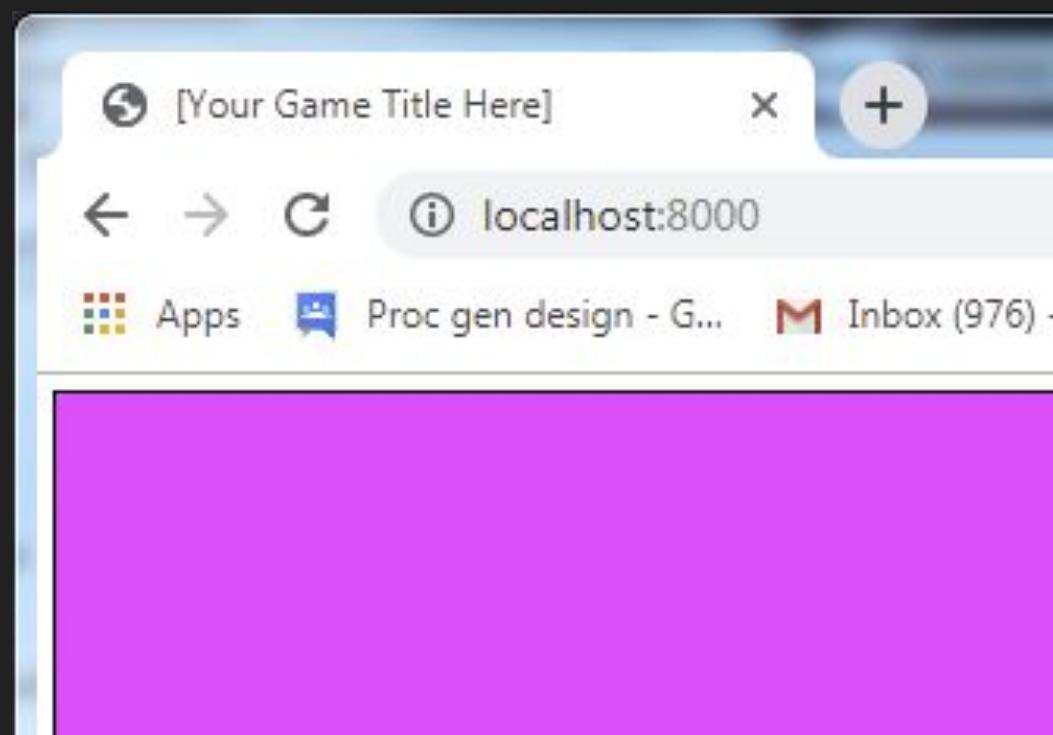
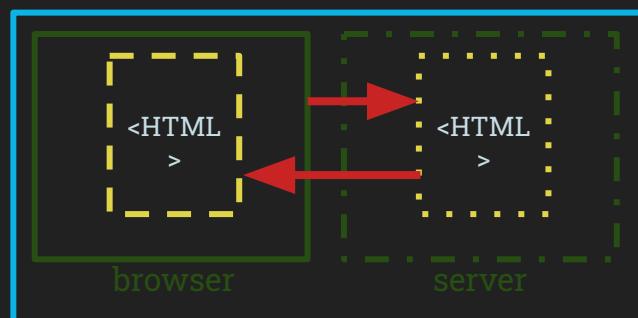
# Server

"server-side" or "back end" applications run on a remote machine

- Limited by the resources of the **remote** machine
- Are often **virtual machines**
  - ◆ Several can share a host machine
  - ◆ Or can be running on a cluster of host servers

May not even be a file:  
the **html** might be generated dynamically





You can serve websites locally

Python 2:

**python -m SimpleHTTPServer**

Python 3:

**python -m http.server**

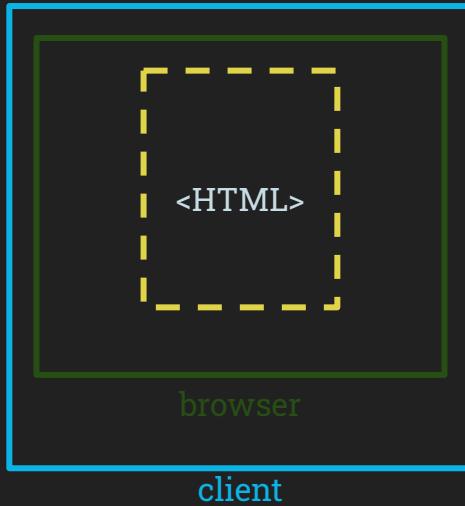
If you don't have Python on your machine, install it.

<https://www.python.org/downloads/>

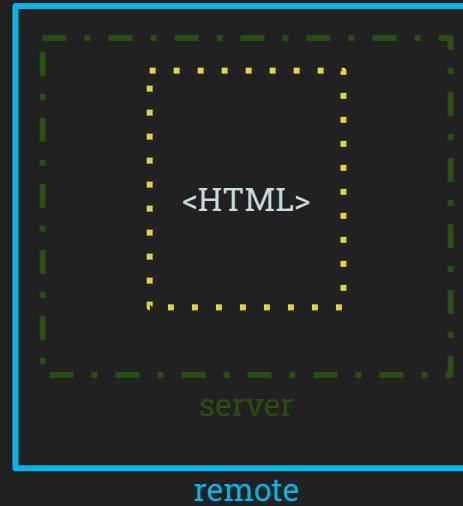
<https://www.anaconda.com/distribution/>

The easiest way to run a local server is with Python

# Technologies



HTML  
CSS  
JavaScript



PHP  
Python  
Ruby  
Clojure  
node.js

**HTML = ???**

**HTML = Hypertext Markup Language**

Nathan Altice  
Santa Cruz, CA

Dear Mom,

I had a fun time at summer camp today. I stole a kid's swimsuit and set it on fire. He didn't mind because we are friends.

When you have the time, please send me a birthday cake in the mail. It isn't my birthday, but I like cakes with my name on them. Please do not send cake in a poster tube like last time.

OK, that's all for now. Please don't touch the things in my room.

Love,  
Nathan

```
<address>
<name>
Nathan Altice
</name>
<location>
Santa Cruz, CA
</location>
</address>

<salutation>
Dear Mom,
</salutation>

<body>
I had a fun time at summer camp today. I stole a kid's swimsuit and set it on
fire. He didn't mind because we are friends.

When you have the time, please send me a <important>birthday cake</important>
in the mail. It isn't my birthday, but I like cakes with my name on them.
Please do not send cake in a poster tube like last time.

OK, that's all for now. Please don't touch the things in my room.
</body>

<closing>
Love,
</closing>
<signature>
Nathan
</signature>
```

# Marking up text gives it **structure** and **meaning**



language



semantic

**HTML uses  
standardized **tags** to  
markup text**

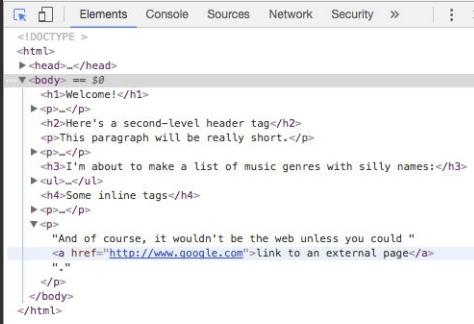
Tags provide semantic  
meaning to content

```
<!DOCTYPE>  
<html>  
</html>
```

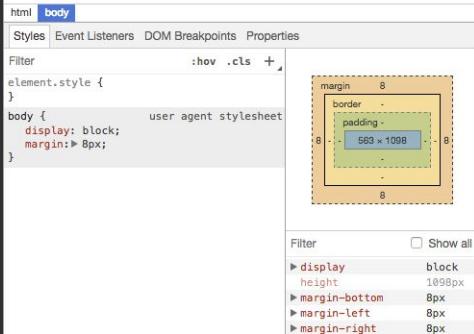
# All major browsers have developer tools

But every browser renders  
webpages slightly differently.

*THE BROWSER WARS! A PERIOD OF  
CIVIL WAR. REBEL OPEN-SOURCE STARTUPS,  
STRIKING FROM A HIDDEN BASE, HAVE WON  
THEIR FIRST VICTORY AGAINST THE EVIL  
NON-STANDARDS-COMPLIANT INTERNET  
EXPLORER...*



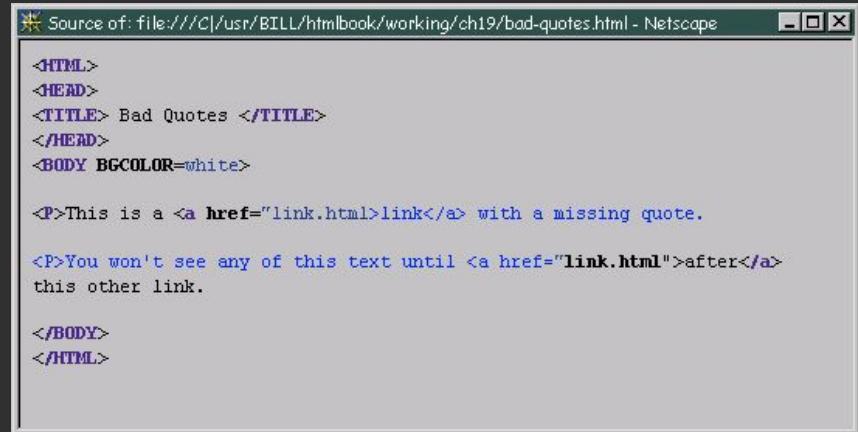
A screenshot of a browser's developer tools showing the DOM tree. The tree starts with the root element <html>, followed by <head> and <body>. The body contains an <h1> element with the text "Welcome!", an <h2> element with "Here's a second-level header tag", a short paragraph, an <h3> element with "I'm about to make a list of music genres with silly names:", a <ul> element containing three <li> items, another short paragraph, and a final <p> element with a link to "http://www.google.com".



A screenshot of a browser's developer tools showing the element inspector for the body element. It shows the current styles applied to the body, including a user agent stylesheet rule for body { display: block; margin: 0px; } and a specific rule for body { border: 1px solid black; padding: 8px; margin: 0px; }. On the right, there is a visual representation of the element's bounding box with a green center, a yellow border, and a black outline, with dimensions of 563 x 1098px indicated.

# View Source

The 90s web designer's  
best friend



A screenshot of the Netscape browser window titled "Source of: file:///C:/usr/BILL/htmlbook/working/ch19/bad-quotes.html - Netscape". The window displays the raw HTML source code of a page. The code includes a title, a white background, and two paragraphs of text. The second paragraph contains a link that is described as missing its quote.

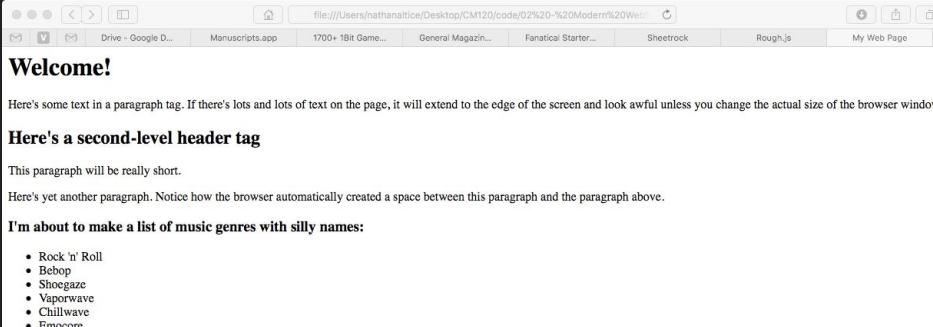
```
<HTML>
<HEAD>
<TITLE> Bad Quotes </TITLE>
</HEAD>
<BODY BGCOLOR=white>

<P>This is a <a href="link.html">link</a> with a missing quote.

<P>You won't see any of this text until <a href="link.html">after</a>
this other link.

</BODY>
</HTML>
```

# In Safari: Preferences > Advanced > Show Develop Menu...



The screenshot shows a web page with the following content:

Welcome!

Here's some text in a paragraph tag. If there's lots and lots of text on the page, it will extend to the edge of the screen and look awful unless you change the actual size of the browser window.

**Here's a second-level header tag**

This paragraph will be really short.

Here's yet another paragraph. Notice how the browser automatically created a space between this paragraph and the paragraph above.

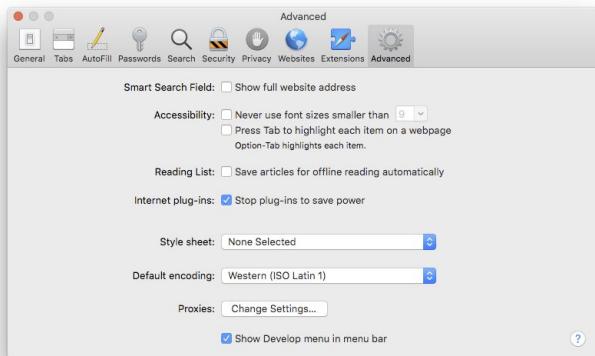
I'm about to make a list of music genres with silly names:

- Rock 'n' Roll
- Bebop
- Shoegaze
- Vaporwave
- Chillwave
- Emocore
- Dubstep

Some inline tags

I can also nest tags inside of tags. So this paragraph can have *italicized text* or **bold text** or even preformatted text for code examples.

And of course, it wouldn't be the web unless you could [link to an external page](#).



The screenshot shows the "Advanced" tab of the Safari Preferences window. The following settings are visible:

- Smart Search Field:  Show full website address
- Accessibility:
  - Never use font sizes smaller than
  - Press Tab to highlight each item on a webpage
  - Option-Tab highlights each item.
- Reading List:  Save articles for offline reading automatically
- Internet plug-ins:  Stop plug-ins to save power
- Style sheet:
- Default encoding:
- Proxies:
- Show Develop menu in menu bar

# Developer Tools Keyboard Shortcuts

Command+Option+I

OsX

F12 or Control+Shift+I

Windows

**CSS = ???**

# CSS = Cascading Style Sheets

# CSS: Let me sum up

- CSS separates presentation from structure
- CSS is a separate language with its own syntax
- CSS statements are called **rules**
- Rules contain a **selector** and a **declaration**
- Style rules "cascade" downward
- CSS definitions may live in a **<style>** tag (usually bad) or be linked externally (much better)

```
body {  
    font-family: "Arial";  
    font-size: 14px;  
    background-color: #facade;  
}
```

```
h1 {  
    border: 1px dotted red;  
}
```

```
h2 {  
    font-variant: small-caps;  
}
```

```
.green {  
    color: green;  
}
```

```
/* h1 {  
    font-family: serif;  
    font-size: 5em;  
} */
```

**But what about...**

---

**What is HTML5?**

# HTML5



HTML5 is the latest evolution of the standard that defines [HTML](#). The term represents two different concepts:

- It is a new version of the *language* HTML, with new elements, attributes, and behaviors,
- and a larger set of **technologies** that allows more diverse and powerful Web sites and applications. This set is sometimes called *HTML5 & friends* and often shortened to just *HTML5*.

Designed to be usable by all Open Web developers, this reference page links to numerous resources about HTML5 technologies, classified into several groups based on their function.

- *Semantics*: allowing you to describe more precisely what your content is.
- *Connectivity*: allowing you to communicate with the server in new and innovative ways.
- *Offline and storage*: allowing webpages to store data on the client-side locally and operate offline more efficiently.
- *Multimedia*: making video and audio first-class citizens in the Open Web.
- *2D/3D graphics and effects*: allowing a much more diverse range of presentation options.
- *Performance and integration*: providing greater speed optimization and better usage of computer hardware.
- *Device access*: allowing for the usage of various input and output devices.
- *Styling*: letting authors write more sophisticated themes.



`<canvas>The part we care about</canvas>`

An HTML element that allows us to draw graphics using scripting (i.e. JavaScript).



plugin



native

**HTML5 provides a**

---

**target container  
for our games**



The screenshot shows a Mac OS X desktop environment. In the foreground, there is a terminal window with a dark background and light-colored text. The terminal window has three tabs at the top: "02-02-firstpage.html", "02-03-important.html", and "02-03-important.html". The third tab is currently active. The terminal displays the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Page Name</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7         /* CSS goes here */
8     </style>
9     <link rel="stylesheet" type="text/css" href="css/styles.css">
10    <script type="text/javascript" src="js/game.js"></script>
11 </head>
12 <body>
13
14    <!-- content goes here -->
15
16    <script type="text/javascript">
17        // code goes here
18    </script>
19 </body>
20 </html>
```

In the background, a file browser window titled "02-03-important.html" is visible. It shows a list of files and folders, including "index.html", "css", "js", and "img". The "img" folder contains several image files like "background.jpg", "button.png", "bullet.png", "bullet2.png", "bullet3.png", "bullet4.png", "bullet5.png", "bullet6.png", "bullet7.png", "bullet8.png", "bullet9.png", and "bullet10.png". The "js" folder contains "game.js". The "css" folder contains "styles.css". The "index.html" file is the active tab in the browser.

# HTML, CSS, and JS combined

# The Document Object Model

## Introduction to the DOM



### DOM Reference

- Introduction to the DOM
- Events and the DOM
- Examples

### IN THIS ARTICLE

This section provides a brief conceptual introduction to the **DOM**: what it is, how it provides structure for **HTML** and **XML** documents, how you can access it, and how this API presents the reference information and examples.

### What is the DOM?

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of the document and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods. Essentially, it connects web pages to scripts or programming languages.

A Web page is a document. This document can be either displayed in the browser window, or as the HTML source. But it is the same document in both cases. The Document Object Model (DOM) provides another way to represent, store and manipulate that same document. The DOM is a fully object-oriented representation of the web page, and it can be modified with a scripting language such as JavaScript.

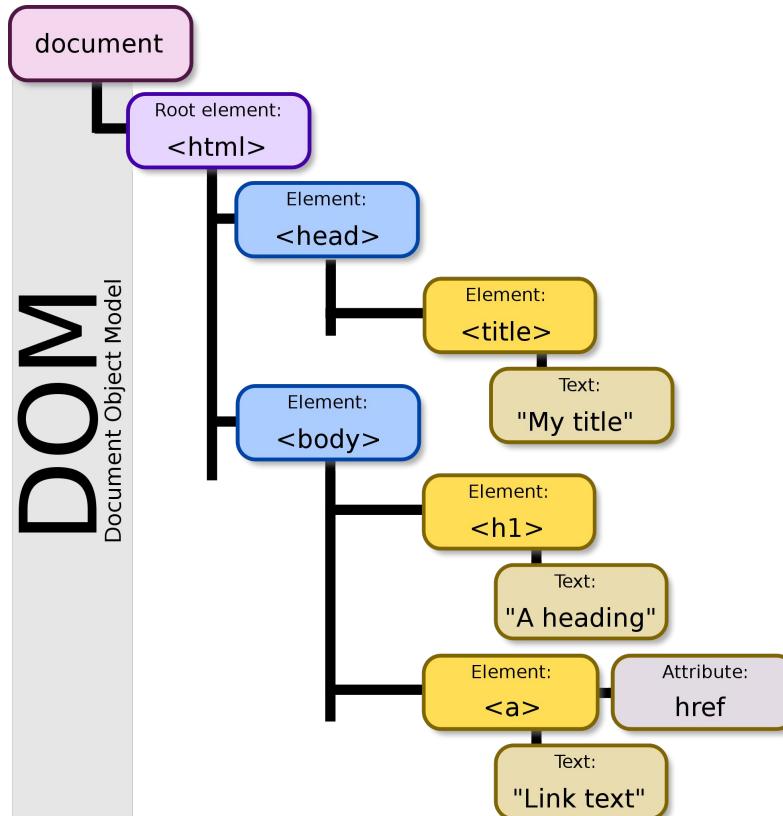
The [W3C DOM](#) and [WHATWG DOM](#) standards form the basis of the DOM implemented in most modern browsers. Many browsers offer extensions beyond the standard, so care must be exercised when using them on the web where documents may be accessed by various browsers with different DOMs.

For example, the standard DOM specifies that the `getElementsByName` method in the code below must return a list of all the `<P>` elements in the document:

```
1 var paragraphs = document.getElementsByTagName("P");
2 // paragraphs[0] is the first <p> element
3 // paragraphs[1] is the second <p> element, etc.
4 alert(paragraphs[0].nodeName);
```

“A Web page is a document. This document can be either displayed in the browser window, or as the HTML source. But it is the same document in both cases. The **Document Object Model (DOM)** provides another way to represent, store and manipulate that same document. The DOM is a fully **object-oriented representation of the web page**, and it can be modified with a scripting language such as JavaScript.”

MDN



<https://commons.wikimedia.org/wiki/File:DOM-model.svg>

**JS = JavaScript**

# JavaScript Overview

- First developed in 1995 at Netscape (for Navigator 2.0)
- Not really related to Java
- Actually a scripting language (domain-specific for web environment)
- Relies on host for input/output (e.g., browser)
- Multi-paradigm (e.g., procedural, functional, OOP, etc.)
- Dynamic (i.e., executes at runtime)
- Loosely typed
- Standardized as ECMAScript
- Historically maligned/praised for its flexibility

# JavaScript Types

number

string

Boolean

Object

Function

Array

Date

RegExp

null

undefined

# JavaScript Types

## Some examples

```
// number
var year = 2019;
var course_number = 120;

// string
var name = "Isaac Karth";

// Boolean
var ownsCar = false;

//Object (Function)
var addNumbers = function(a, b) {
    return a + b;
}

//Object (Array)
var favGames = ['Thief: The Dark Project', 'SimCity 2000',
    'Heaven's Vault', 'Crusader Kings 2', 'Pathologic 2']

//Object (Date)
var today = new Date(2019, 6, 25);

//Object (RegExp)
var re = new RegExp('\\w');

var the_abyss = null;

undefined // it's complicated
```