THE INTERNET

In a Couple of Minutes

WHAT IS IT?

"A global network of networks."

(It's just a bunch of connected computers)

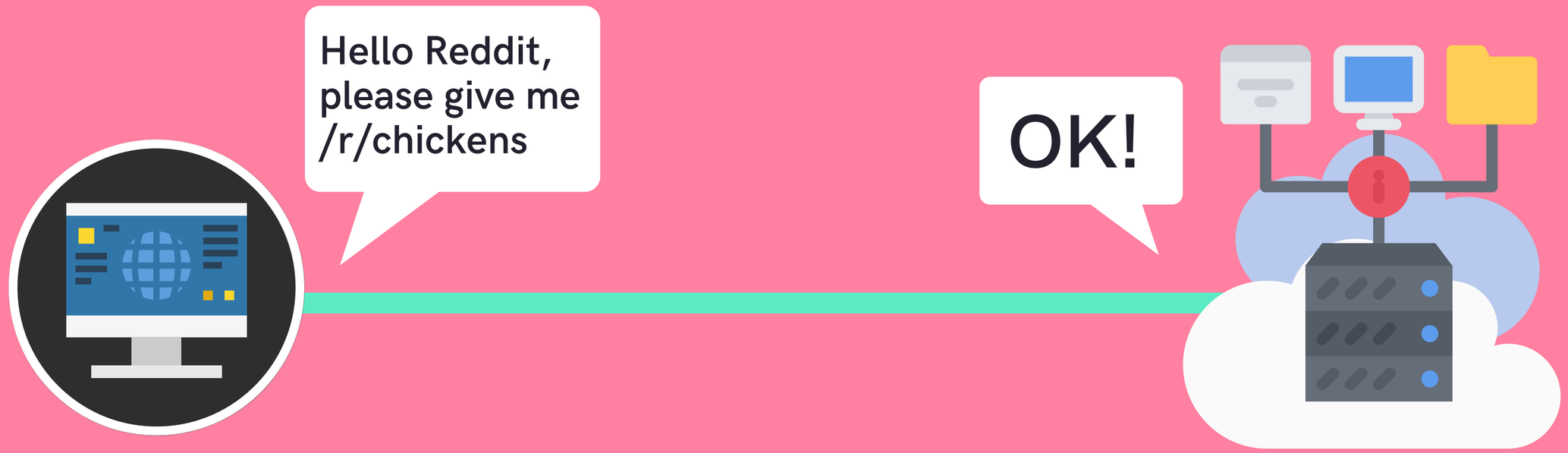
THE INTERNET IS THE INFRASTRUCTURE THAT CARRIES THINGS LIKE:

- EMAIL
- THE WEB
- FILE SHARING
- ONLINE GAMING
- STREAMING SERVICES

WEB SERVER

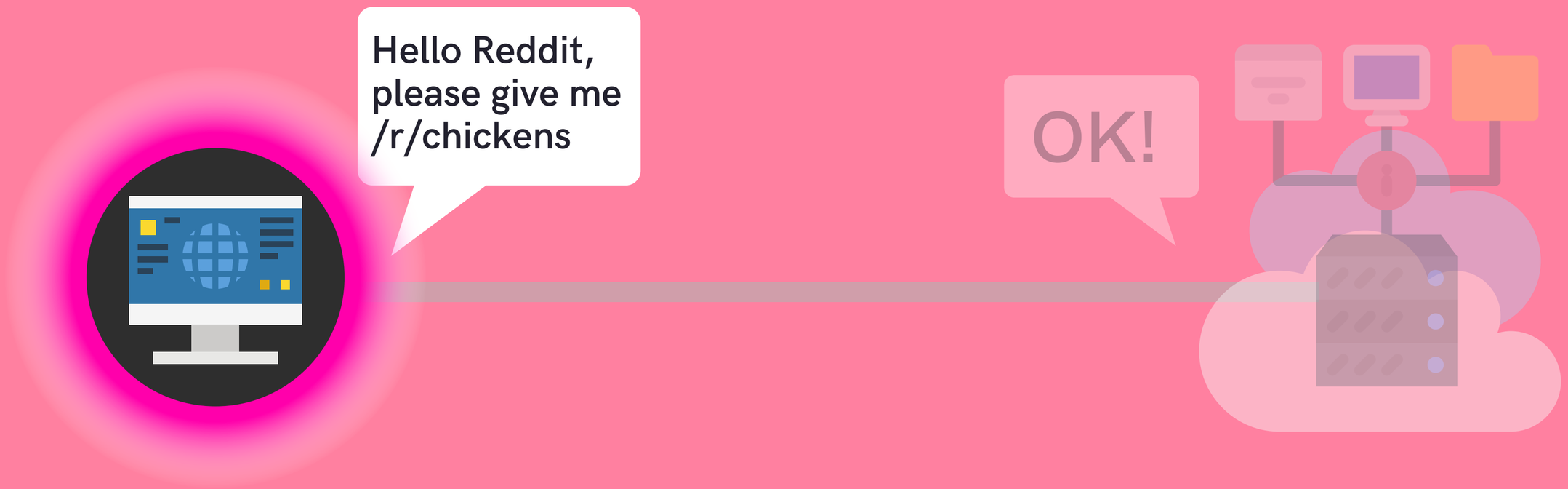
A computer* that can satisfy requests on the Web.

* "server" is also used to refer to the software running on the computer

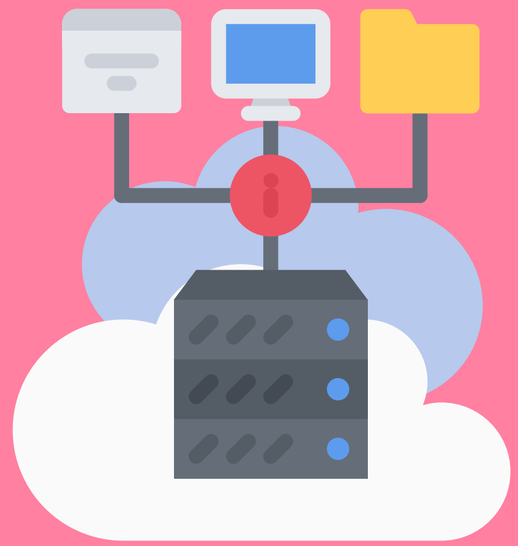


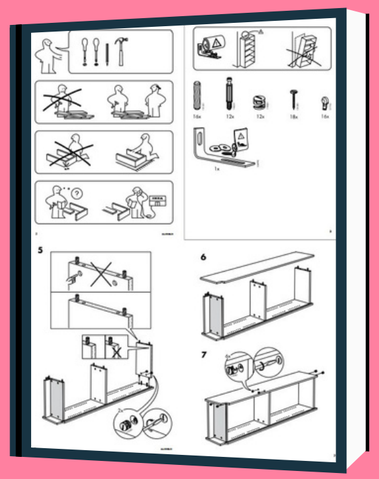
CLIENT

The computer that accesses a server



A SERVER SOMEWHERE





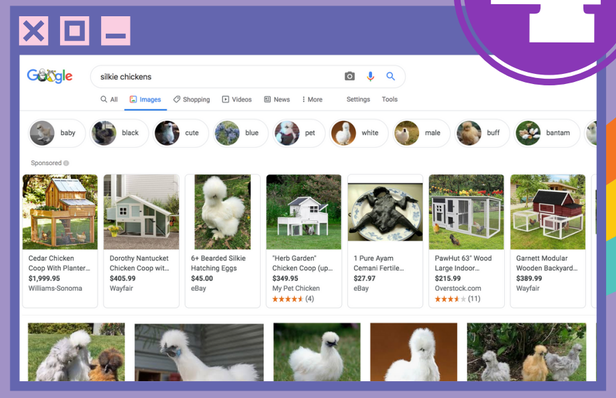
Here are the instructions!

YOUR BROWSER

Look What I Made!

BACK END

FRONT END

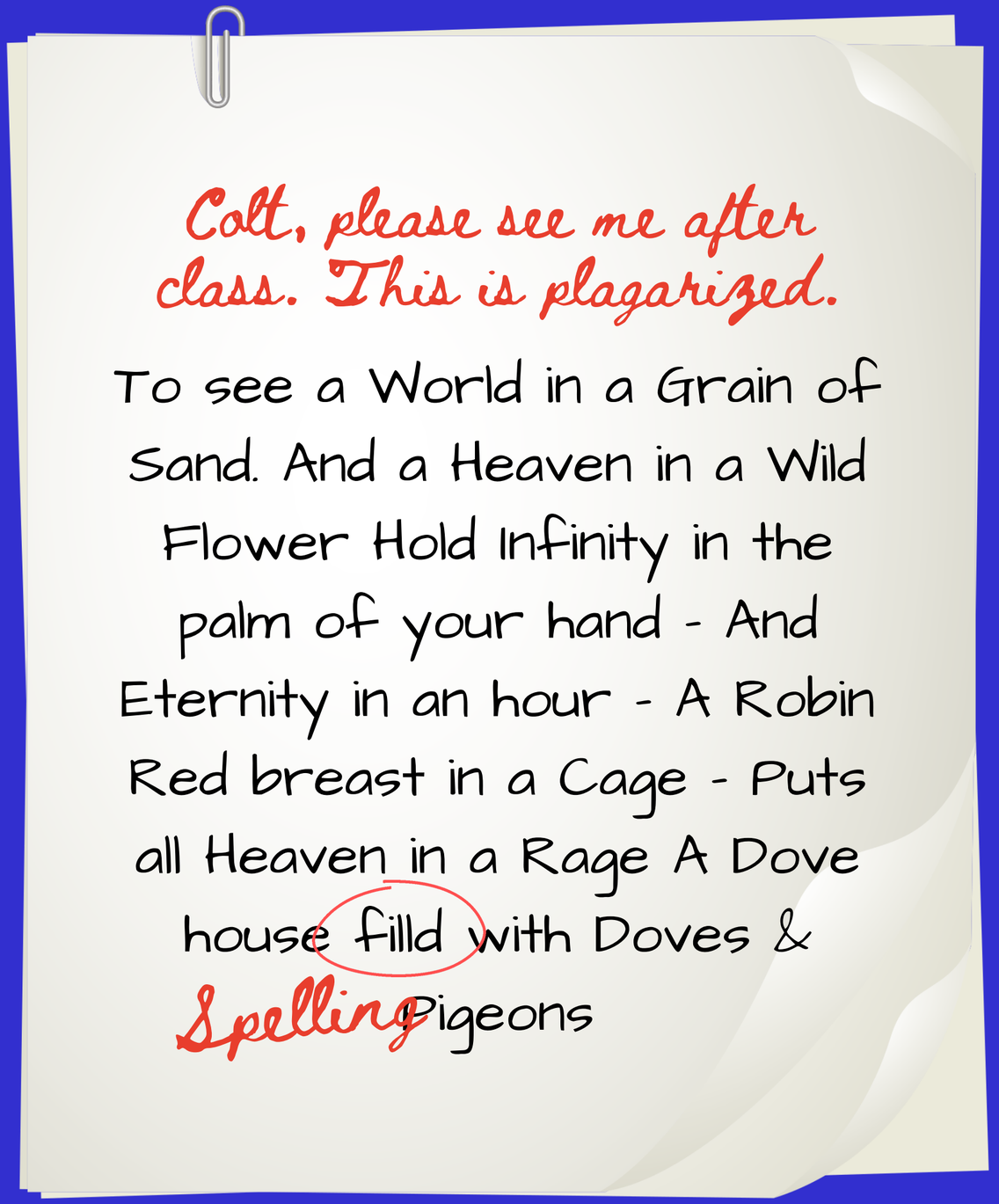


FRONT END

BACK END

HTML

IS A MARKUP LANGUAGE



HTML ELEMENTS

To write HTML, we pick from a set of standard Elements that all browsers recognize

Common Elements include:

- *<p> element* - represents a paragraph of text
- *<h1> element* - represents the main header on a page
- *<img> element* - embeds an image
- *<form> element* - represents a form

HTML TAGS

We create elements by writing *tags*.
Most (but not all) elements consist of an opening and closing tag.

Opening Tag

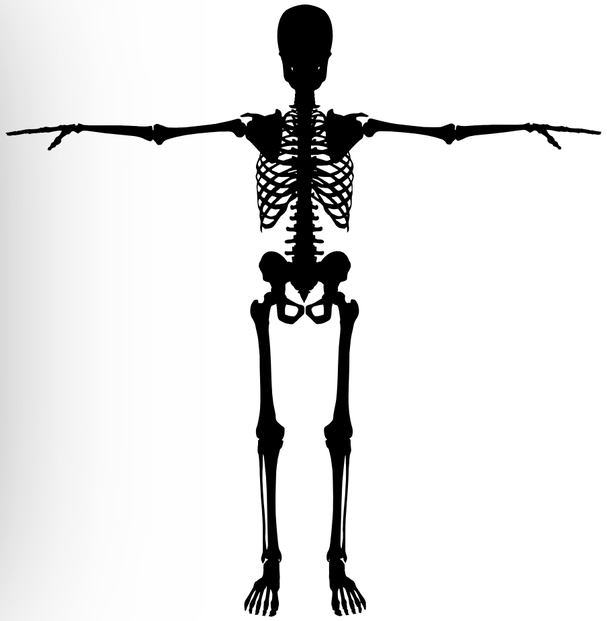
```
<p>I am a paragraph</p>
```

Closing Tag

HTML SKELETON

We write our HTML in a standard "skeleton"

```html
<!DOCTYPE html>
<html>
<head>
  <title>My First Page</title>
</head>
<body>
  <!-- Content Goes Here -->
</body>
</html>
```



HTML

What is HTML5?

LIVING STANDARD

The HTML standard is a document that describes how HTML should work.

ROLE OF BROWSERS

The standard describes the rules of HTML, but browsers actually have to do the work and implement HTML according to those rules.

HTML5?

HTML5 is the latest evolution of the standard that defines HTML. It includes new elements & features for browsers to implement,


{ }
CSS

CSS

WHAT IS IT?

CSS is a language for describing how documents are presented visually - how they are arranged and styled.

WHAT DOES IT STAND FOR?

CSS stands for Cascading Style Sheets. We'll cover the "cascading" part in a bit; don't worry about it for now!

THERE'S A LOT!

CSS is very easy to get the hang of, but it can be intimidating because of how many properties we can manipulate.

CSS RULES

(almost) everything you do in CSS follows this basic pattern:

```
selector {
  property: value;
}
```

RESPONSIVE DESIGN

what is it & why you should care

THE PROBLEM

As mobile devices and tablets became widely available, developers had a problem...how do we create websites that look good on all screen sizes?

ONE APPROACH

Early on, it was common to create separate stylesheets for different devices, or even completely different websites for each size.

ENTER RESPONSIVE

These days, we typically create ONE website and stylesheet that is able to respond to different device sizes and features.

MEDIA QUERIES

Media queries allow us to modify our styles depending on particular parameters like screen width or device type.

Bootstrap

"THE WORLD'S MOST POPULAR CSS FRAMEWORK"

What is it?

Bootstrap helps us quickly build responsive websites.

Components

Bootstrap gives us access to a bunch of pre-built components that we can incorporate into our own websites.

Grid System

Bootstrap also comes with a grid system, which helps us construct our own custom, responsive layouts.

Web Developer Bootcamp

JavaScript Basics

VALUES & VARIABLES

LEARN JS ON ITS OWN. NO HTML/CSS.

USE JS TO MANIPULATE HTML/CSS

Primitive Types

THE BASIC BUILDING BLOCKS

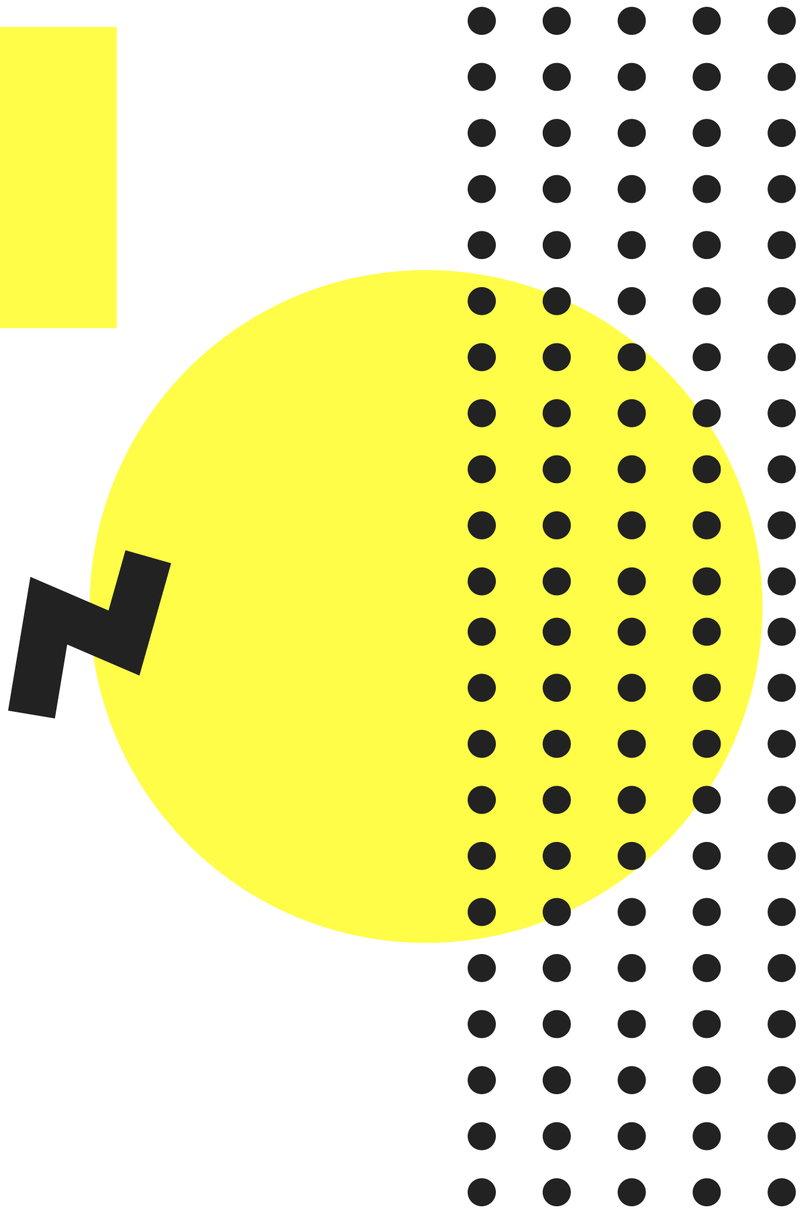
Number

String

Boolean

Null

Undefined

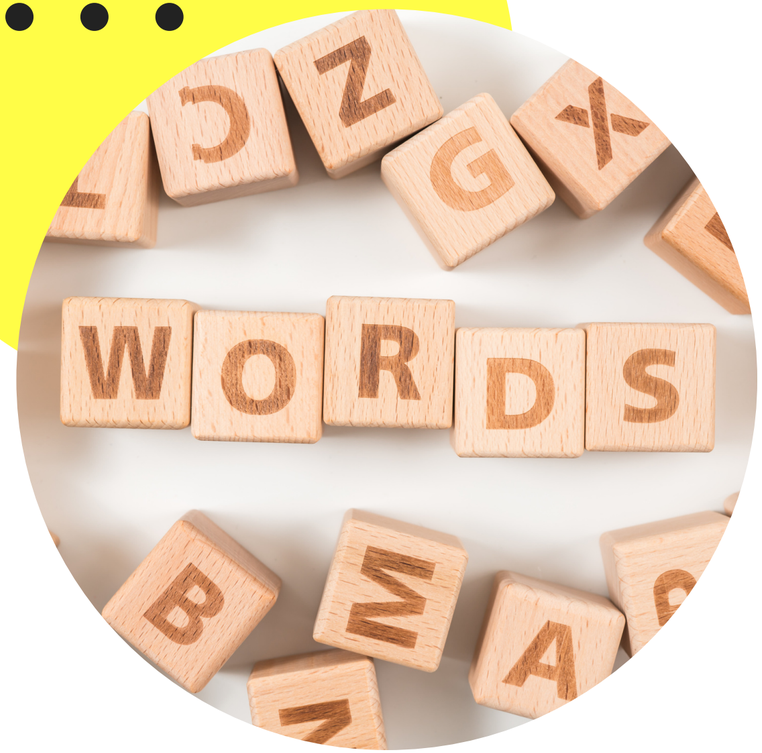


* Technically there are two others: Symbol and BigInt

NOT A NUMBER

NaN

NaN is a numeric value that represents something that is...not a number



Strings

"STRINGS OF CHARACTERS"

Strings are another primitive type in JavaScript. They represent **text**, and must be wrapped in quotes.

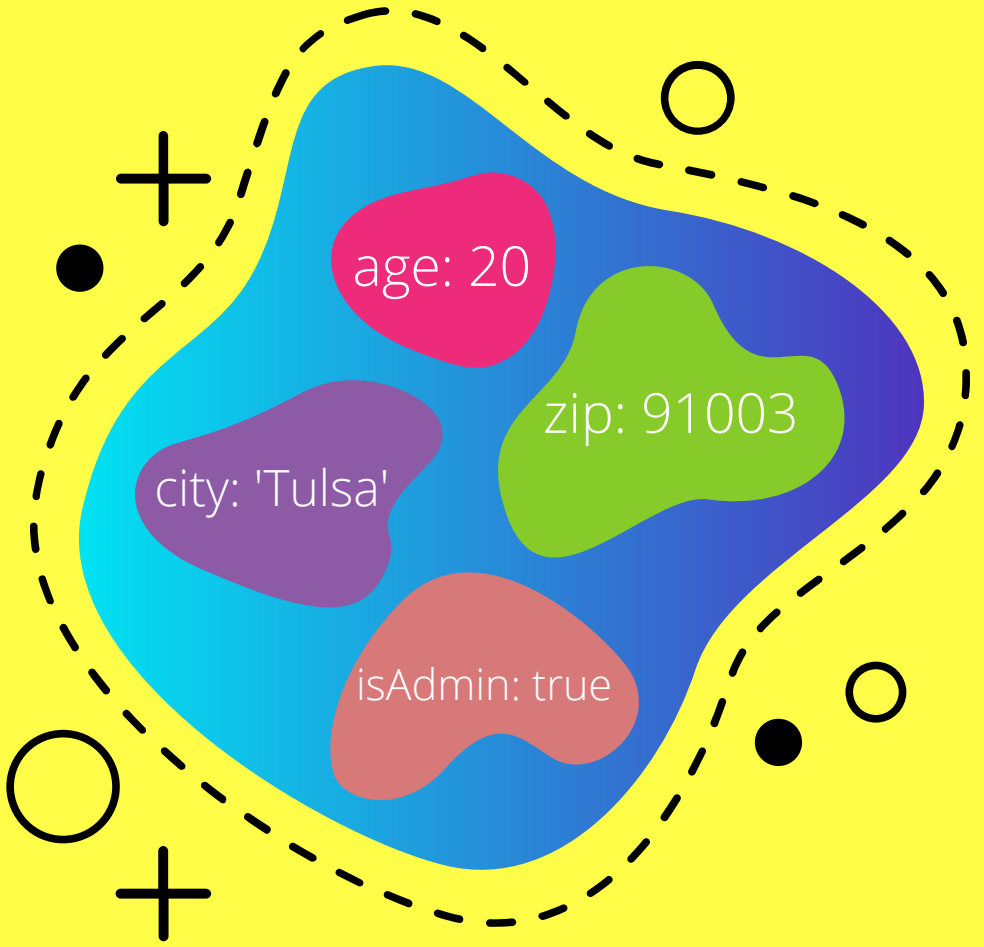
ARRAYS

Ordered collections of values.

- List of comments on IG post
- Collection of levels in a game
- Songs in a playlist

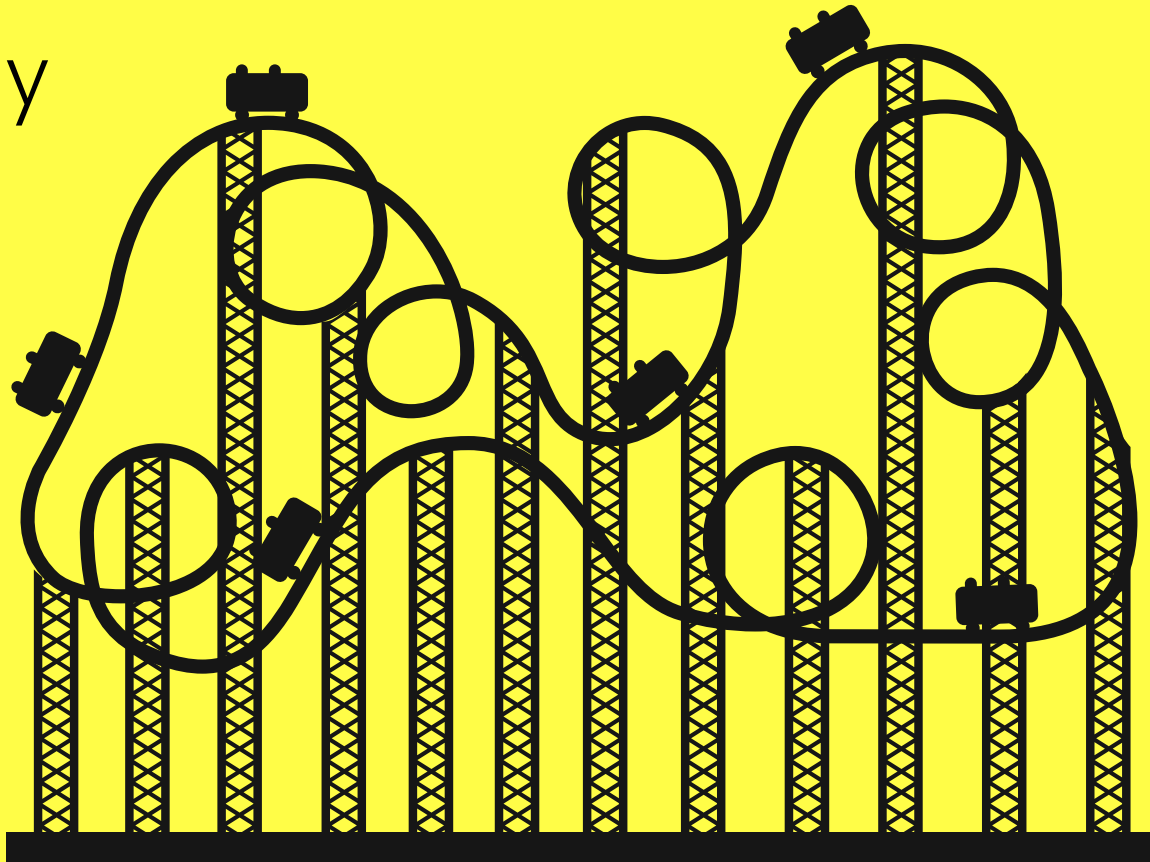
OBJECTS

- Objects are collections of **properties**.
- Properties are a key-value pair
- Rather than accessing data using an index, we use custom keys.



LOOPS

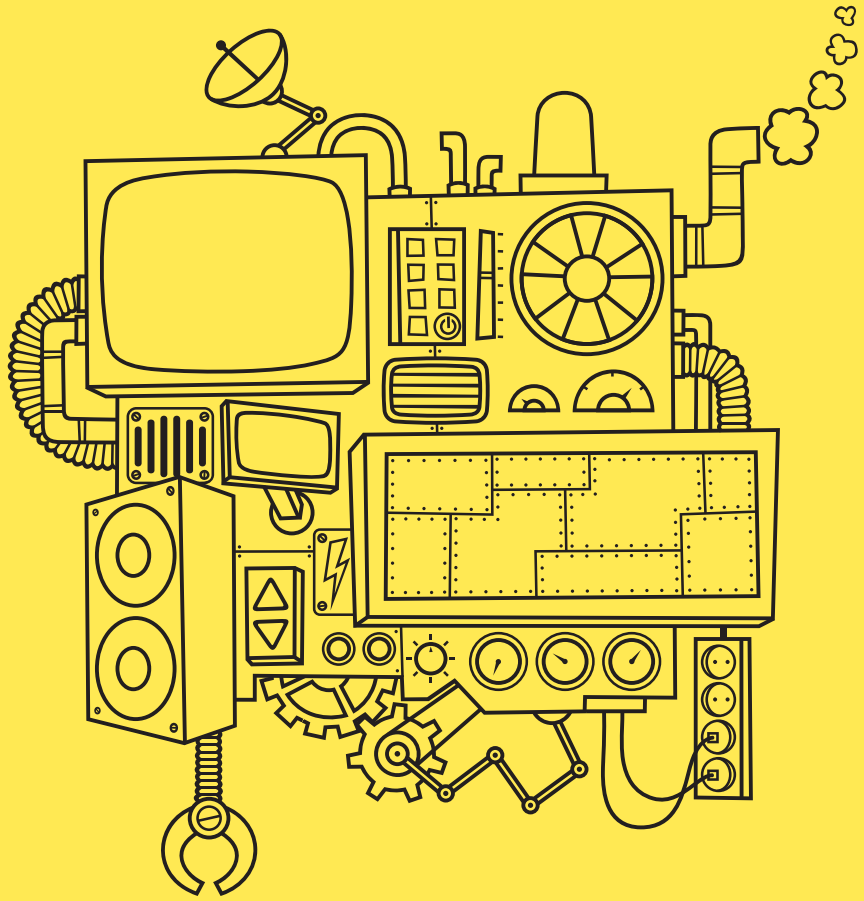
- Loops allow us to repeat code
  - "Print 'hello' 10 times
  - Sum all numbers in an array
- There are multiple types:
  - for loop
  - while loop
  - for...of loop
  - for...in loop



FUNCTIONS

Reusable procedures

- Functions allow us to write reusable, modular code
- We define a "chunk" of code that we can then execute at a later point.
- We use them ALL THE TIME



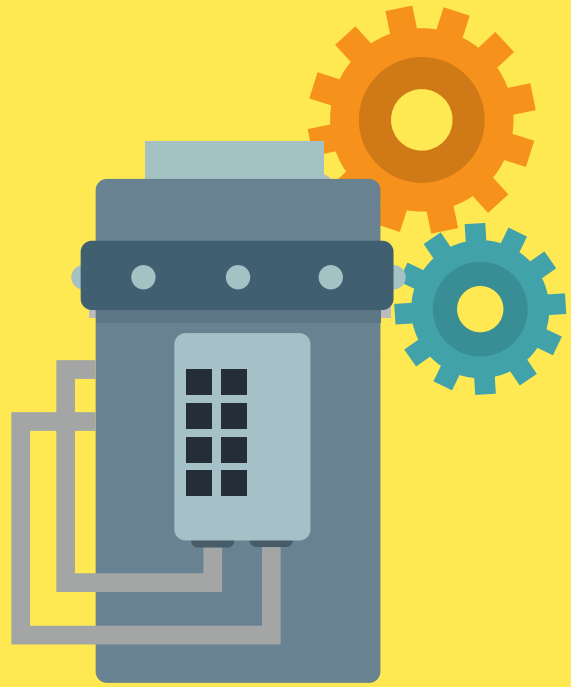
ARGUMENTS

ARGUMENTS

We can also write functions that accept inputs, called arguments

ARGUMENTS

`greet('Tim')` → → `"Hi Tim!"`

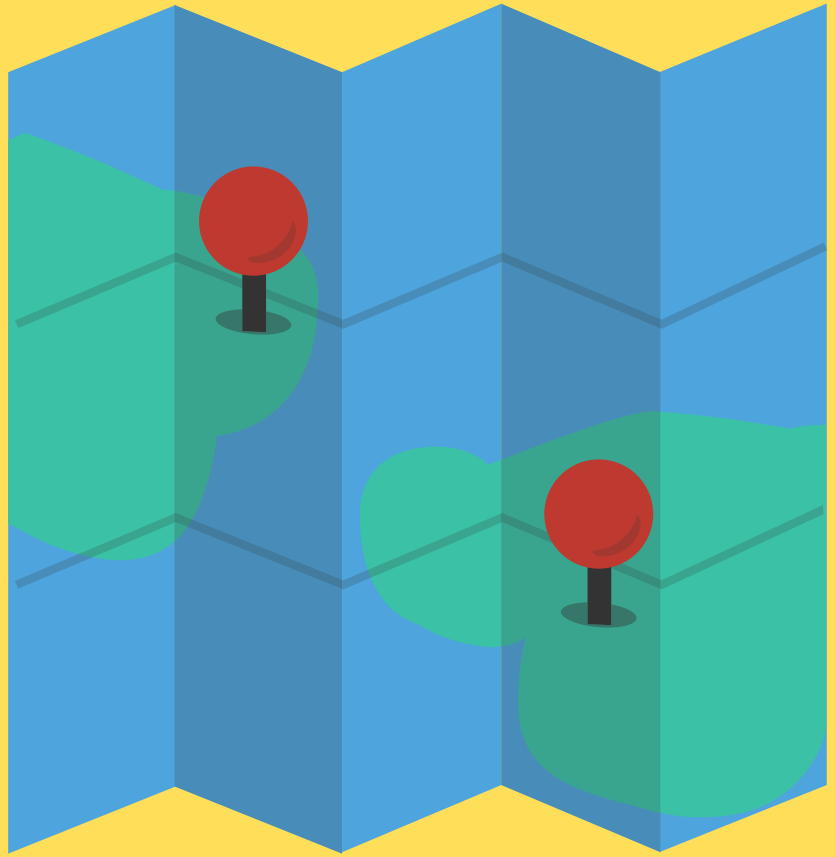


`greet('Anya')` → → `"Hi Anya!"`

FUNCTIONS ARE... OBJECTS!

MAP

Creates a new array with the results of calling a callback on every element in the array



ARROW FUNCTIONS

"syntactically compact alternative"
to a regular function expression

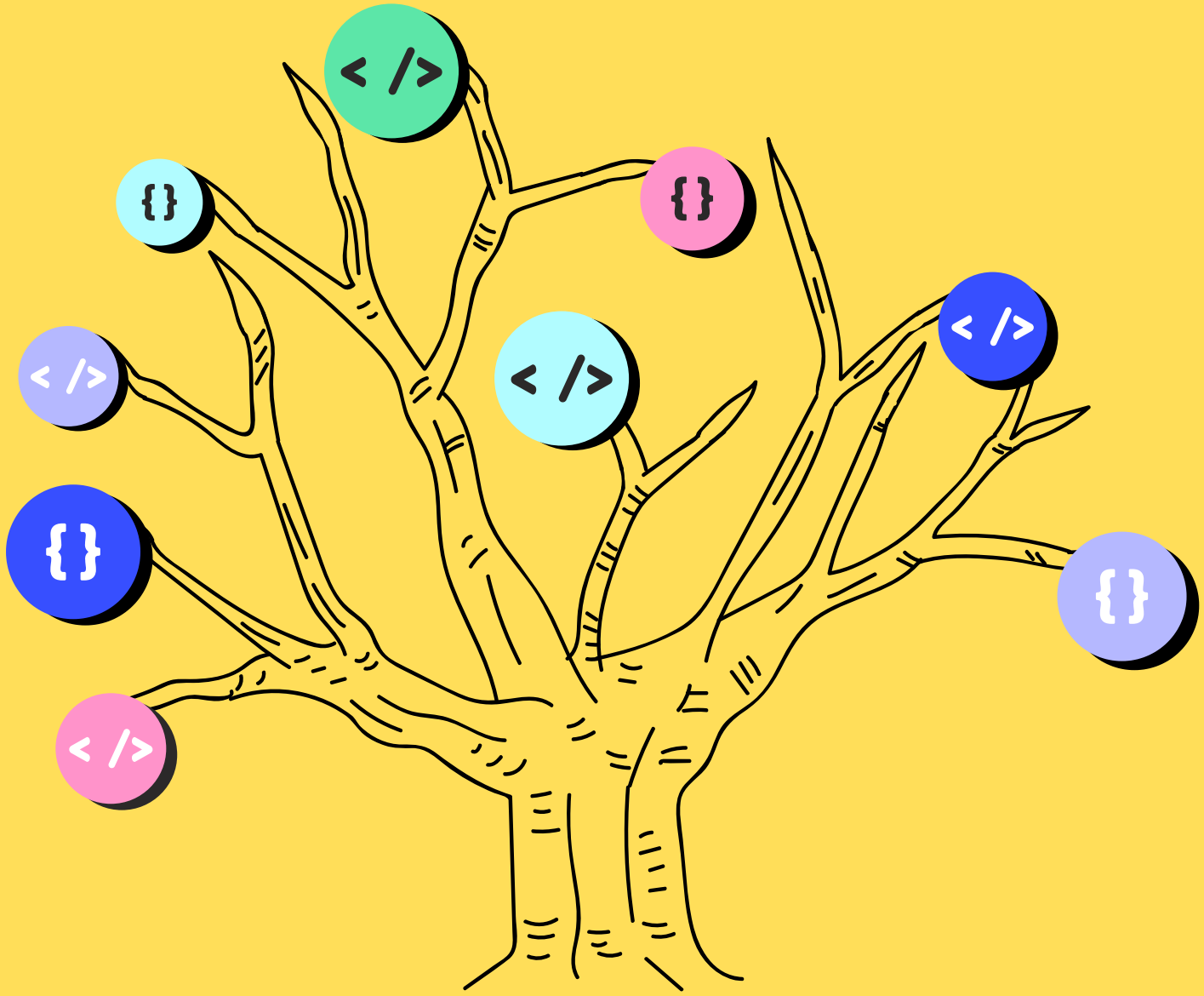
```
const square = (x) => {
  return x * x;
}

const sum = (x, y) => {
  return x + y;
}
```

SPREAD

Spread syntax allows an iterable such as an array to be **expanded** in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.

THE DOM

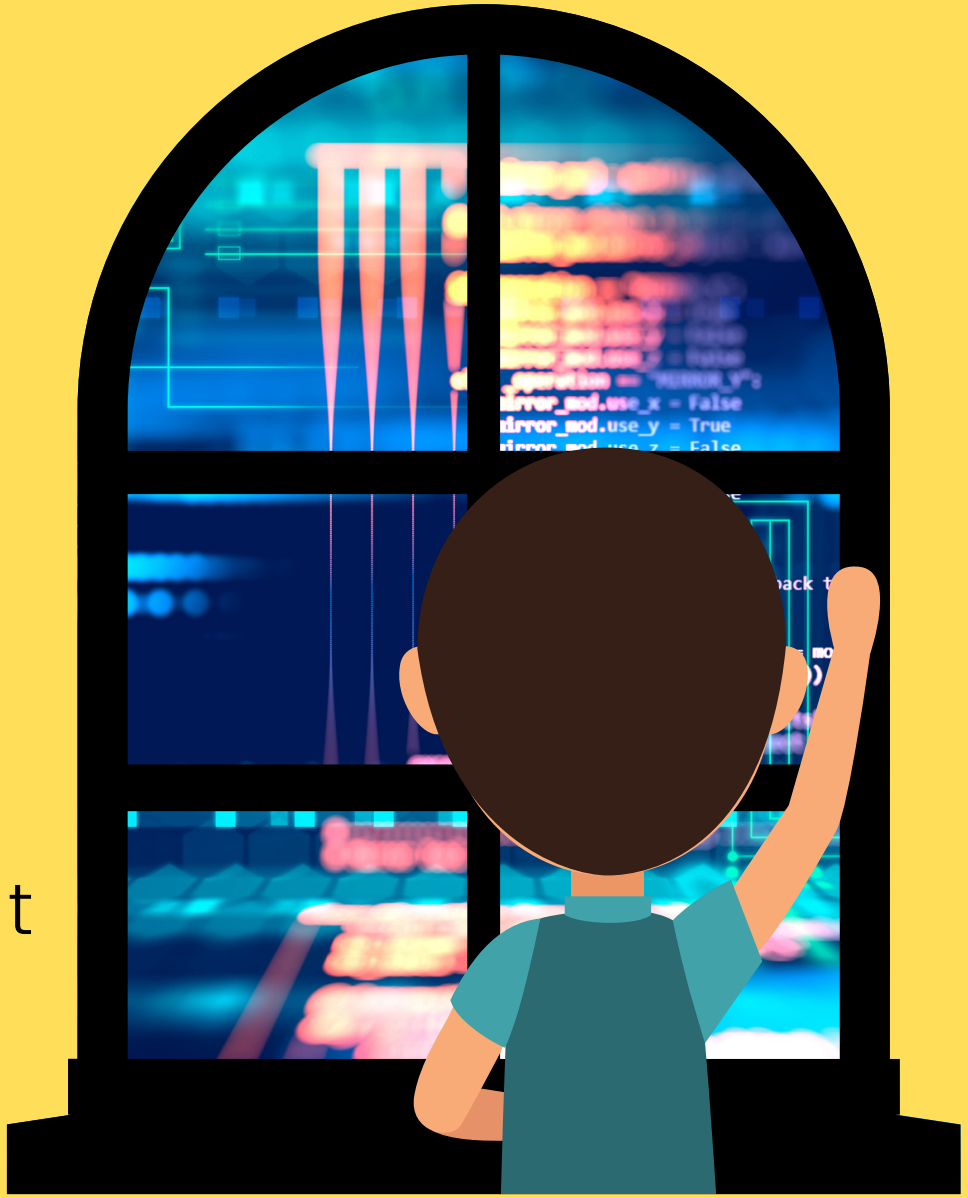


DOM

Document Object Model

WHAT IS IT?

- The DOM is a JavaScript representation of a webpage.
- It's your JS "window" into the contents of a webpage
- It's just a bunch of objects that you can interact with via JS.



PROPERTIES & METHODS

(the important ones)

- classList
- getAttribute()
- setAttribute()
- appendChild()
- append()
- prepend()
- removeChild()
- remove()
- createElement

- innerText
- textContent
- innerHTML
- value
- parentElement
- children
- nextSibling
- previousSibling
- style

EVENTS

Responding to user inputs and actions!

A SMALL TASTE

- clicks
- drags
- drops
- hovers
- scrolls
- form submission
- key presses
- focus/blur

- mouse wheel
- double click
- copying
- pasting
- audio start
- screen resize
- printing

CALL STACK

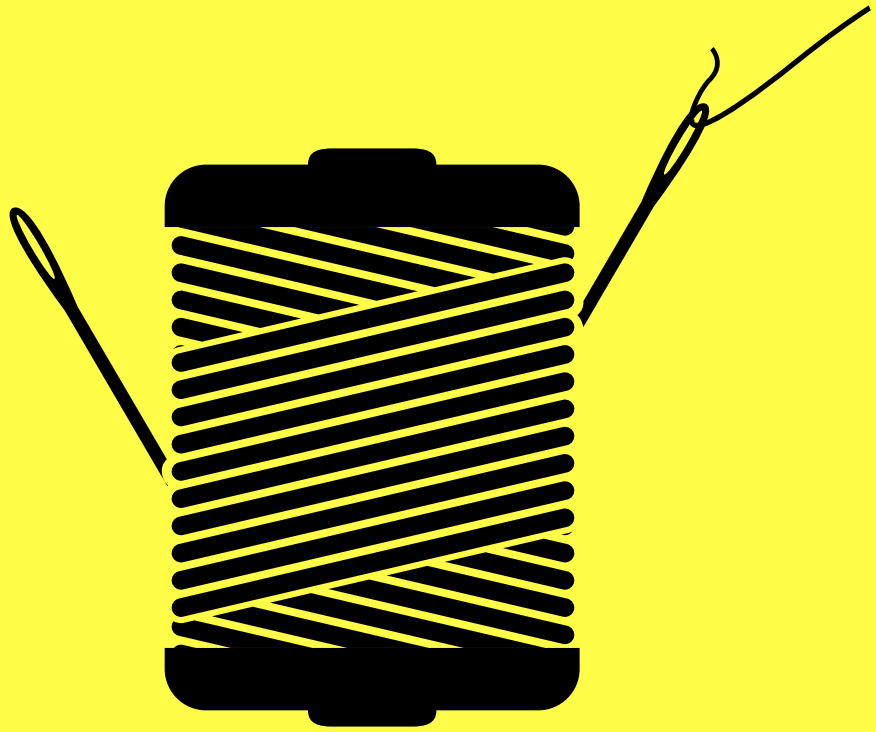
The mechanism the JS interpreter uses to keep track of its place in a script that calls multiple functions.

How JS "knows" what function is currently being run and what functions are called from within that function, etc.

LAST THING IN...

FIRST THING OUT...

JS IS SINGLE THREADED



WHAT DOES THAT MEAN?

At any given point in time, that single JS thread is running at most one line of JS code.

PROMISES

A pattern for writing async code.

RESOLVE

REJECT

A promise is a returned object to which you attach callbacks, instead of passing callbacks into a function

REQUESTS

- XMLHTTP
- FETCH
- AXIOS

AJAX

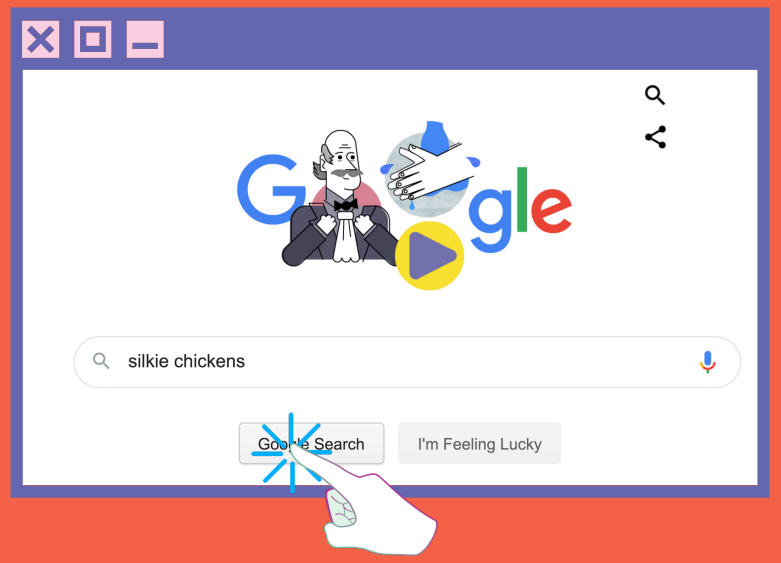
- ASYNCHRONOUS
- JAVASCRIPT
- AND
- XML

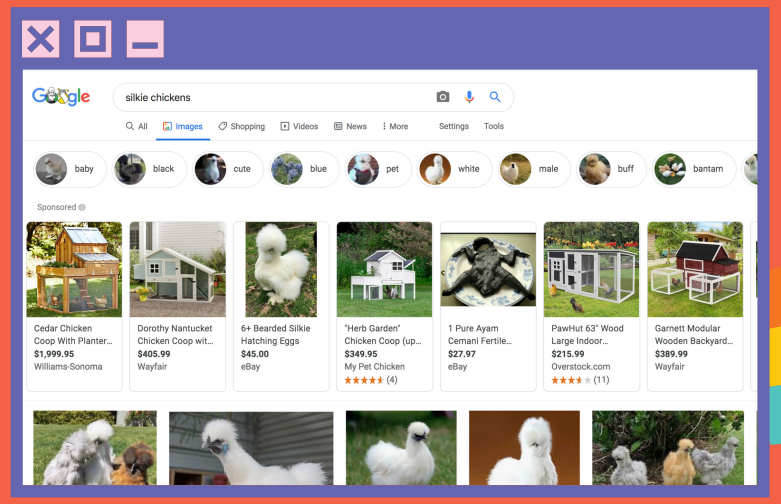
HTTP REQUESTS

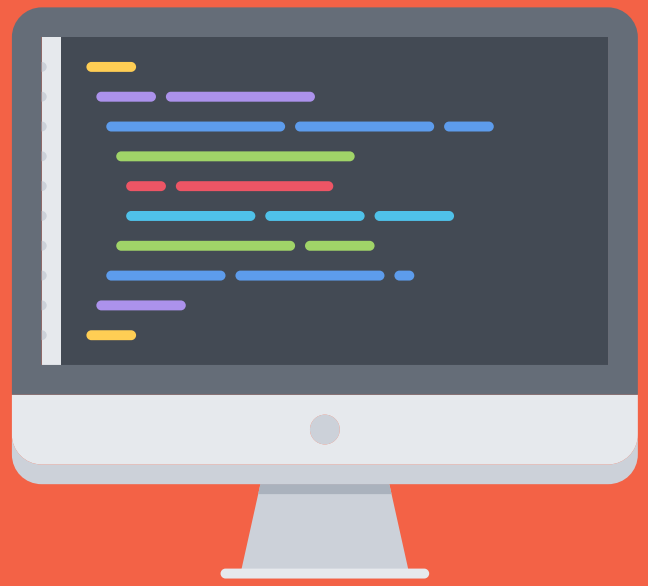
- Foundation of communication on the World Wide Web
- "Hyper Text Transfer Protocol"
- Request -> *I would like this information please*
- Response -> *Ok, here you go!*

PLEASE GIVE ME

GOOGLE.COM/SEARCH?Q=CHICKENS







PLEASE GIVE ME

API.CRYPTONATOR.COM/API/TICKER/BTC-USD

price is 11227.76

AJAX

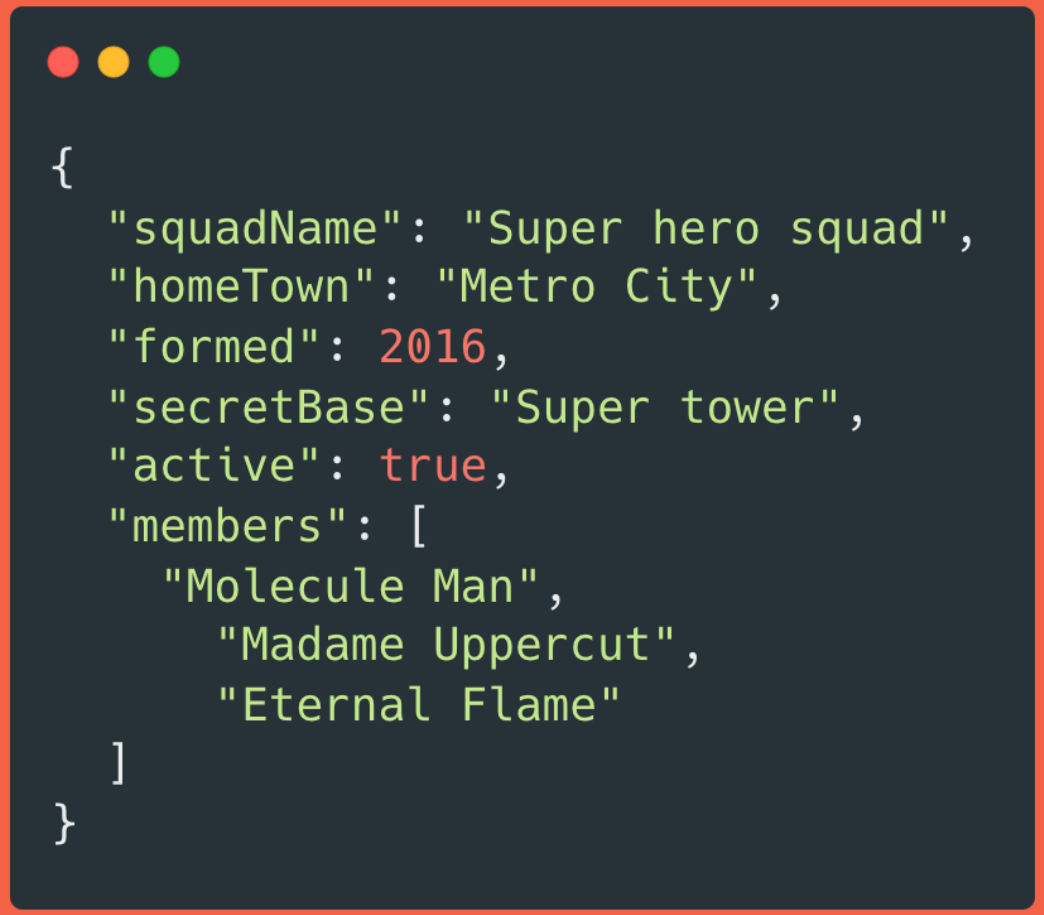
- ASYNCHRONOUS
- JAVASCRIPT
- AND
- XML

AJAJ

- ASYNCHRONOUS
- JAVASCRIPT
- AND
- JSON

JSON

- Java
- Script
- Object
- Notation



```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    "Molecule Man",
      "Madame Uppercut",
      "Eternal Flame"
  ]
}
```

XMLHttpRequest

- The "original" way of sending requests via JS.
- Does not support promises, so...lots of callbacks!
- WTF is going on with the weird capitalization?
- Clunky syntax that I find difficult to remember!

FETCH

Fetch API

- The newer way of making requests via JS
- Supports promises!
- Not supported in Internet Explorer :(

AXIOS

A LIBRARY FOR MAKING HTTP REQUESTS

WHAT IS NODE?

"A JAVASCRIPT RUNTIME"

Until recently, we could only run JavaScript code in a web browser. Node is a JavaScript runtime that executes code **outside of the browser**.

We can use the same JavaScript syntax we know and love to write **server-side code**, instead of relying on other languages like Python or Ruby.

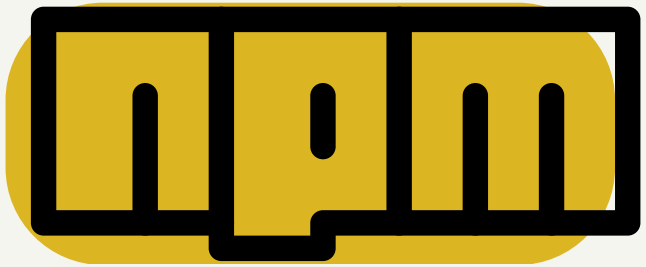
NODE JS VS CLIENT-SIDE JS

NOT INCLUDED IN NODE

Because Node does not run in a browser, we don't have access to all the browser "stuff". The window, document, and DOM API's are not a thing in Node!

NEW STUFF IN NODE!

Node comes with a bunch of built-in modules that don't exist in the browser. These modules help us do things like interact with the operating system and files/folders.



NPM

NODE PACKAGE MANAGER

NPM is really two things:

1. A library of thousands of packages published by other developers that we can use for free!
2. A command line tool to easily install and manage those packages in our Node projects

WHAT IS EXPRESS?

OUR FIRST FRAMEWORK!

Express is a "fast, unopinionated, minimalist web framework for Node.js:" It helps us build web apps!

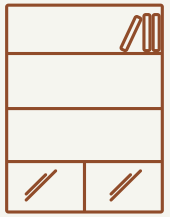
It's just an NPM package which comes with a bunch of methods and optional plugins that we can use to build web applications and API's

EXPRESS HELPS US...

- Start up a server to listen for requests
- Parse incoming requests
- Match those requests to particular routes
- Craft our http response and associated content

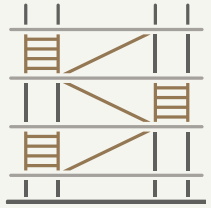
LIBRARIES VS. FRAMEWORKS

LIBRARY



When you use a library, you are in charge! You control the flow of the application code, and you decide when to use the library.

FRAMEWORK

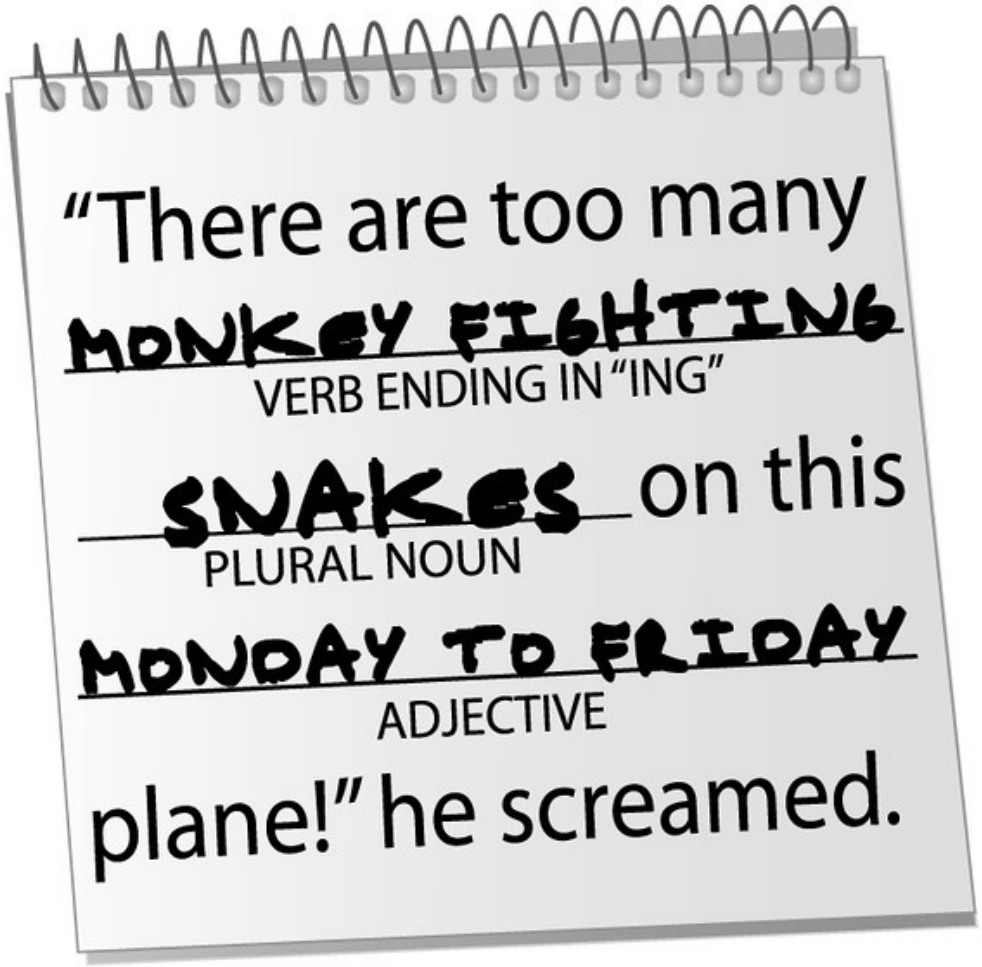


With frameworks, that control is inverted. The framework is in charge, and you are merely a participant! The framework tells you where to plug in the code.

TEMPLATING

Templating allows us to define a preset "pattern" for a webpage, that we can dynamically modify.

For example, we could define a single "Search" template that displays all the results for a given search term. We don't know what the term is or how many results there are ahead of time. The webpage is created on the fly.

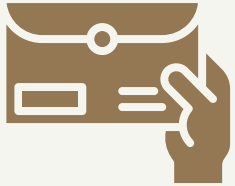


GET VS. POST

GET

- Used to retrieve information
- Data is sent via query string
- Information is plainly visible in the URL!
- Limited amount of data can be sent

POST



- Used to post data to the server
- Used to write/create/update
- Data is sent via request body, not a query string!
- Can send any sort of data (JSON!)

WHAT IS MONGO?

OUR FIRST DATABASE!

According to Mongo's homepage, it is "the most popular database for modern applications". It is commonly used in combination with Node.

Mongo is a **document database**, which we can use to store and retrieve complex data from.

WHY USE A DATABASE?

INSTEAD OF JUST SAVING TO A FILE?

- Databases can handle large amounts of data efficiently and store it compactly
- They provide tools for easy insertion, querying, and updating of data
- They generally offer security features and control over access to data
- They (generally) scale well.

SQL VS. NOSQL

SQL DATABASES

Structured Query Language databases are relational databases. We pre-define a schema of tables before we insert anything.

NO-SQL DATABASES

NoSQL databases do not use SQL There are many types of no-sql databases, including document, key-value, and graph stores.

POPULAR DATABASES

SQL DATABASES

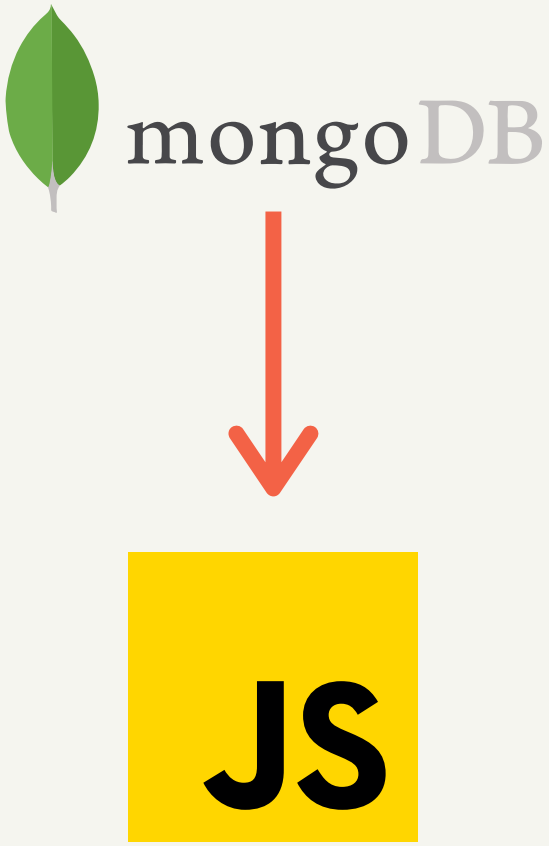
- MySQL
- Postgres
- SQLite
- Oracle
- Microsoft SQL Server

NO-SQL DATABASES

- MongoDB
- Couch DB
- Neo4j
- Cassandra
- Redis

WHY ARE WE LEARNING MONGO?

- Mongo is very commonly used with Node and Express (MEAN & MERN stacks)
- It's easy to get started with (though it can be tricky to truly master)
- It plays particularly well with JavaScript
- Its popularity also means there is a strong community of developers using Mongo.



ODM

OBJECT DATA MAPPER?
OBJECT DOCUMENT MAPPER?

ODMs like Mongoose map documents coming from a database into usable JavaScript objects.

Mongoose provides ways for us to model out our application data and define a schema. It offers easy ways to validate data and build complex queries from the comfort of JS.

MIDDLEWARE

Express middleware are functions that run during the request/response lifecycle.

MIDDLEWARE

REQUEST

- Middleware are just functions
- Each middleware has access to the request and response objects
- Middleware can end the HTTP request by sending back a response with methods like res.send()
- OR middleware can be chained together, one after another by calling next()

RESPONSE