Before we begin...

- Download VS Code
- Download Google Chrome
- Videos On!

Welcome

Some thank yous

- General Assembly
- All of you!

Introductions

Jack Jeffress

- *Your Lead Instructor*
- Developer || Hacker || Programmer
- Currently...
- Previously...
- I have been teaching for a long time
  - General Assembly Distinguished Faculty Member
  - Google Developer Expert
- Email, Twitter and Instagram
- Outside of coding...

How I teach

- The more questions the better
- Interruptions and rabbit holes are always welcome
- I'll try to keep it interesting
- I always want to keep learning
- I add far more into the slides than necessary
- I love teaching (so always happy for extras etc.)

Stacey Brosnan

- *Your Instructor Associate*
- Take it away, Stacey!

Now for the important people

Who are you?

- What is your name?
- What do you do?
- What is your coding experience?
- What do you want to get out of this course?
- What do you enjoy? Hobbies, interests etc.

Thank you!

This Course

How we work at GA

- We start from the beginning
- We adapt as necessary
- We teach industry standards and best practices
- We appreciate your feedback
- We get you to base camp

Course Overview

This is JavaScript, from the ground up. We will start from the beginning and get you proficient in the most popular programming language in the world.

There are 4 main parts:

- JavaScript Overview
- JavaScript and the Browser
- APIs, AJAX and Persisting Data
- Front-end Framework (React)

But really...

- Learning JavaScript as a language
- Getting you excited about what you can do with it
- Introducing you to lots of new concepts
- Teaching you how to learn (in a coding context)
- Having fun and working as a team
- Providing you with a platform to continue from

A Typical Class

- Agenda
- Warmup
- Recap
- Class Content
- Break
- Class Content
- Sometimes there will be extras
  - Completely optional though!

Agenda

- Welcome
- Introductions
- Class Overview
- Web Development Overview
- JavaScript's Role and History
- JavaScript's Data Types
- JavaScript's Variables (maybe)
- JavaScript's Operators and Conditionals (maybe)
- JavaScript's Loops (maybe)
- Review

The Tools We Will Use

- Google Chrome - Our Browser
- VS Code - Our Text Editor
- Slack - For Chatting and Sharing Info
- Zoom - For Video Calls
- GitHub
  - Accessing my code and slides
  - Eventually, submitting homework and projects

Zoom Features

- Annotations
- Raise Hand
- Breakout Rooms
- No text chat in Zoom
  - Use Slack for that, please!

A Bit of Advice

Getting the most out of it

- Take it slowly
- Pretend you are a computer
- Find parallels and look for real-world applications
- Fail early and often
- Do more than is asked
- Embrace your inner nerd
- Enjoy the little wins
- Solve the problem before you start coding
- Consciously read code
- Work together and work harder than required

The Pitfalls

- Getting too advanced too quickly
- Judging yourself
- Comparing yourself to others
- Forgetting about the user
- Not taking advantage of what you have here:
  - Your classmates
  - Your TA
  - 1-on-1s

Overview of the Web

What is the Web?

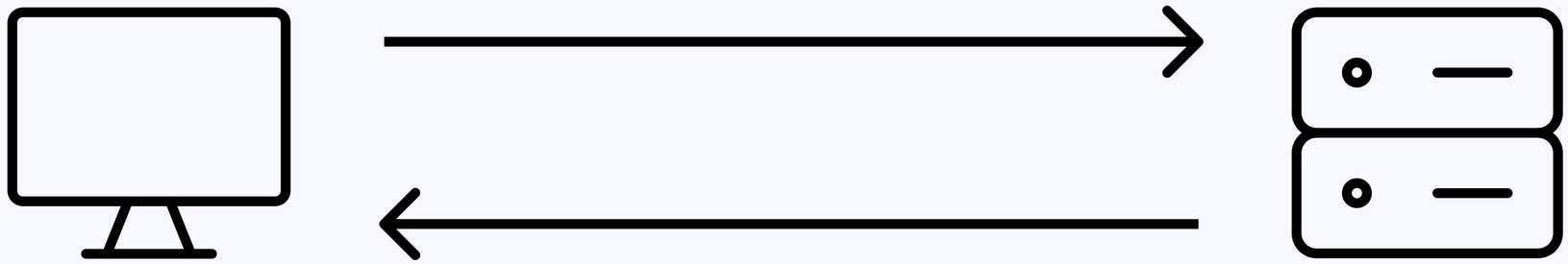
Billions of connected devices through a series of networks

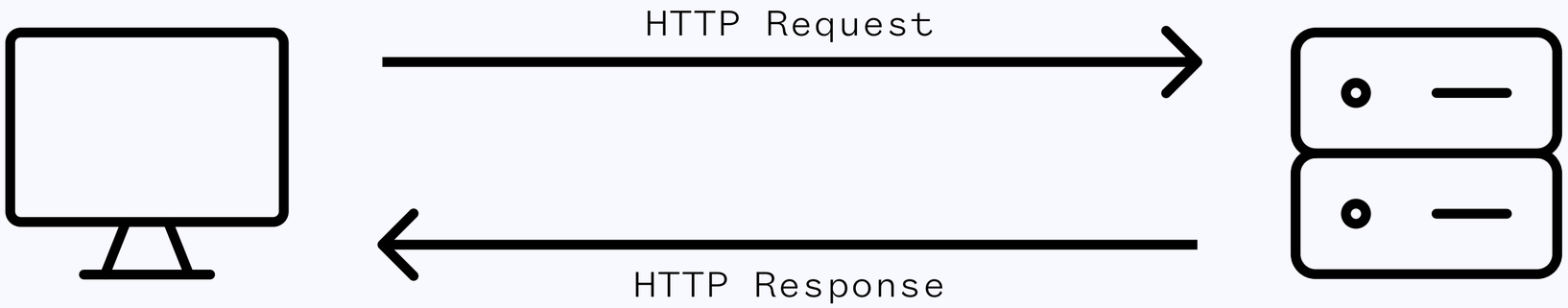
The web and the internet are different. The web is the physical network of devices, the internet is the virtual network of information

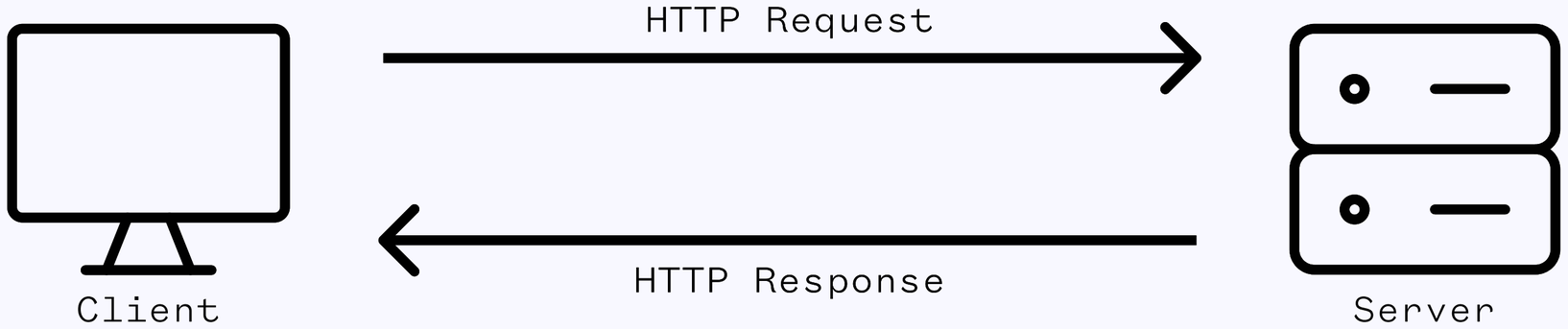
A bit of history...

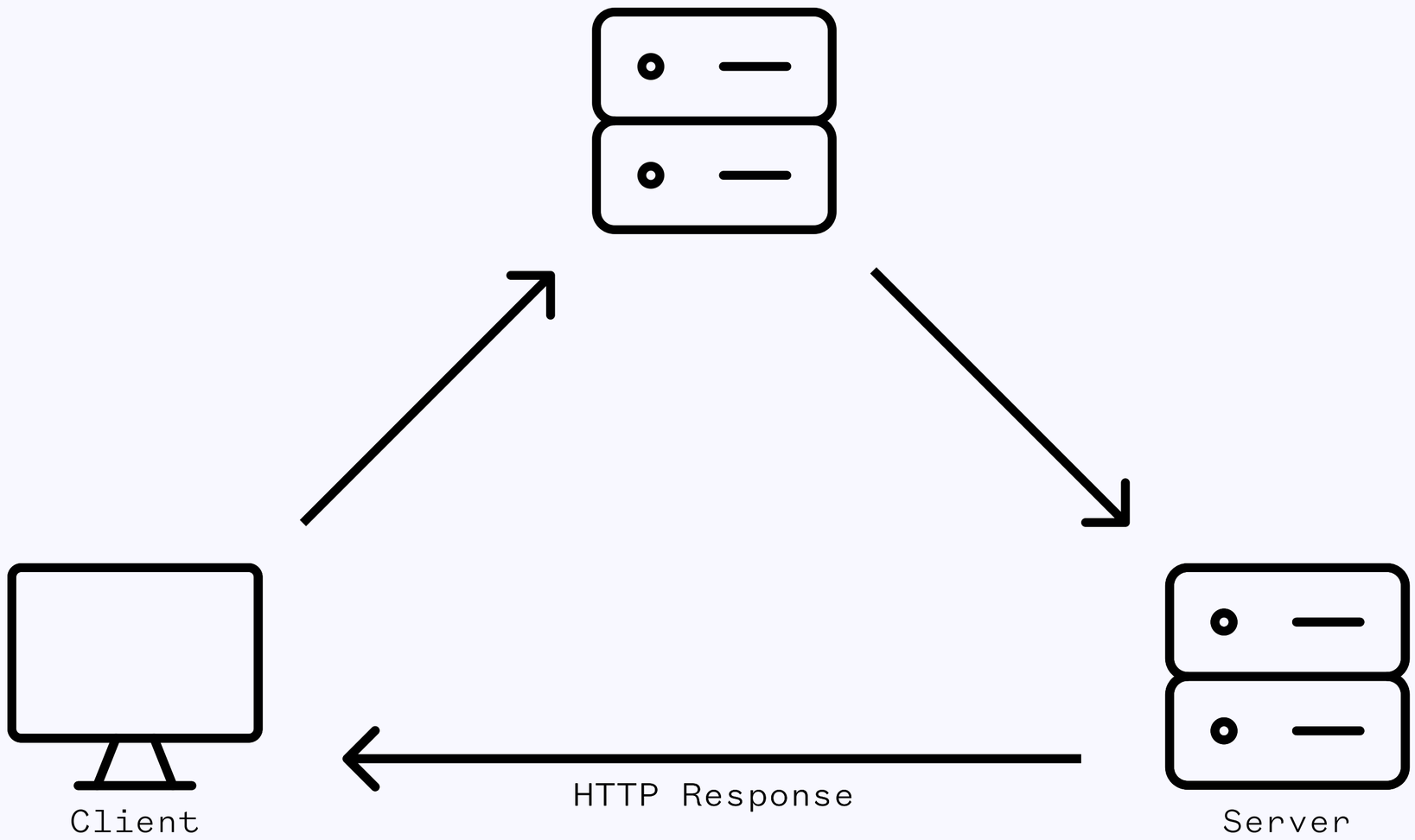
- J.C.R. Licklider came up with 'The Galactic Network' (Aug. 1962)
- ARPANet in the 1960's for the US Government
- Vint Cerf and Bob Kahn invented TCP/IP in 1974
- Then, Tim Berners-Lee released the 'World Wide Web' in 1991, with this site

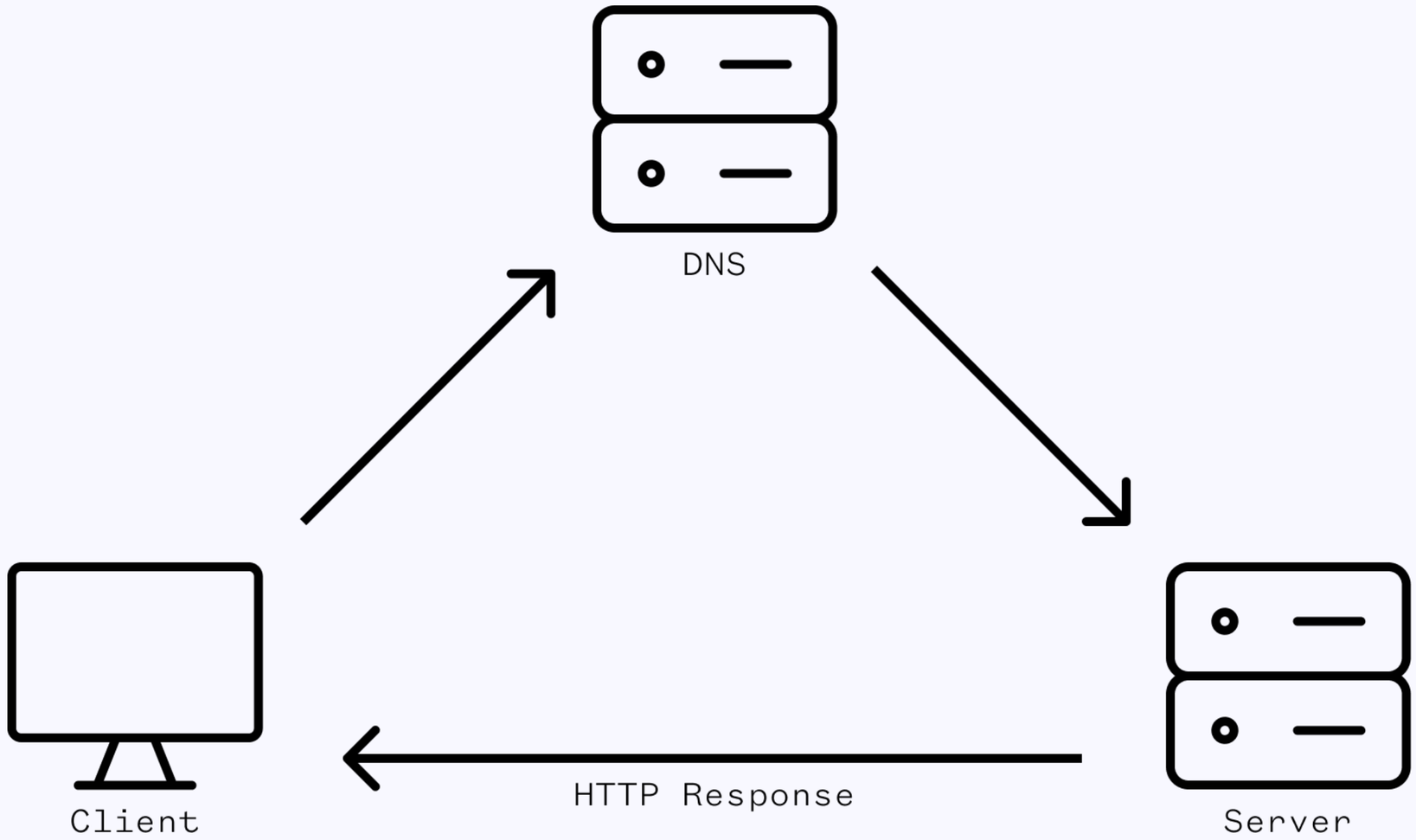
How does it work?

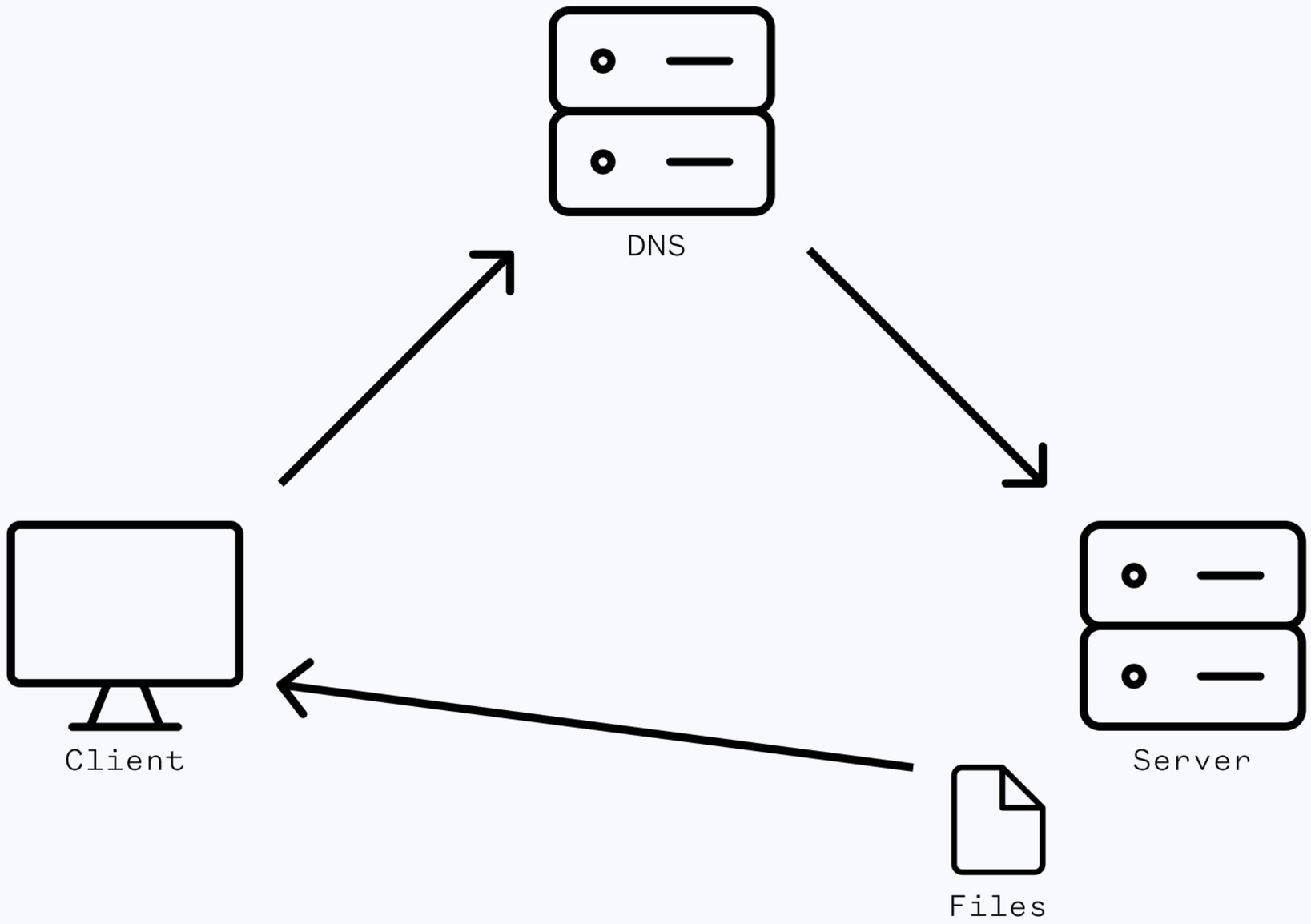


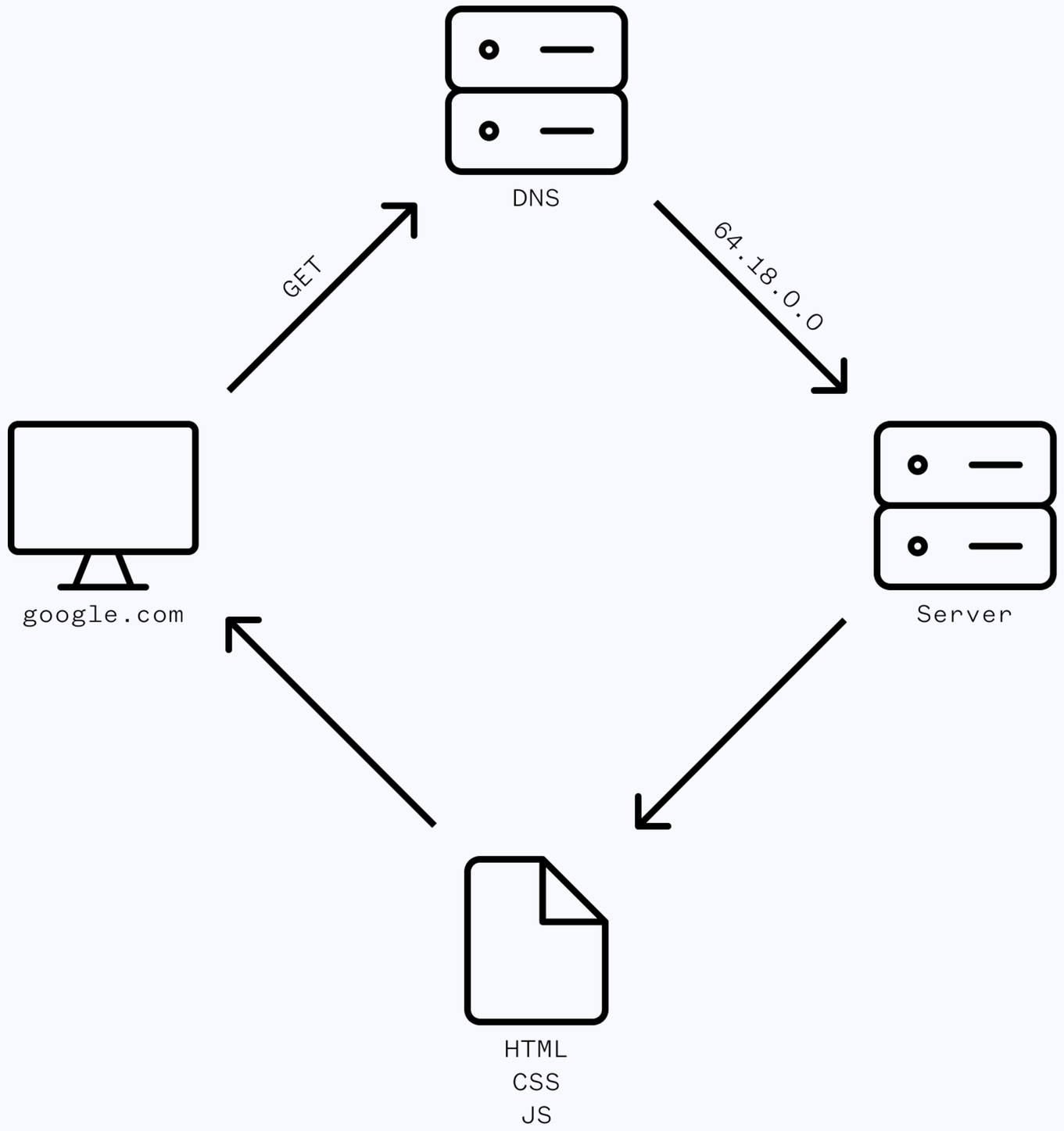

HTTP Request
HTTP Response


HTTP Request
HTTP Response
Client
Server


HTTP Response
Client
Server


DNS
Client
HTTP Response
Server


DNS
Client
Server
Files


DNS
GET
64.18.0.0
google.com
Server
HTML
CSS
JS

Web Development

What is it?

- The creation and management of websites and web-based applications
- Made up of the **Front-End** and the **Back-End**
- Most developers will try to convince you that it's magic

The Front-End

The Front-End

- What the user sees
- It powers the visuals and interactions of the web
- Made up of three languages
  - HTML
  - CSS
  - JS

HTML → Content

The bones

CSS → Style

The appearance

JavaScript → Functionality

The brain

The Back-End

The Back-End

- What goes on behind the scenes
- Consists of databases, servers, processes etc.
- Can make payments, send emails, store assets etc.
- Lots of different programming languages
  - Node.js, C++, Python, Ruby, Elixir, ASP.net, Go etc.
- We will see this a little bit (but not very much)

JavaScript Overview

What is JavaScript?

- The most popular programming language (according to Stack Overflow, GitHub and GitHut)
- Originally created to make websites alive
- It is a very flexible programming language
  - Runs in the front-end (the browser), in the back-end (as Node.js), and other places too

What is JavaScript?

- The programs in JavaScript are called scripts, and have the file extension of `.js`
  - They are just ordinary files - no preparation or compilation needed
- It is a loosely typed and dynamic language (we will go into this later)

What can JavaScript do?

- Validating information
- Change a page's HTML and CSS
- Request information from APIs
- Adding interactivity and animations
- Visualise data
- Can be used for art
- 3D, Games, Augmented and Virtual Reality
- Internet of Things and Hardware
- Artificial Intelligence
- Plus, much more!

How will we run JavaScript?

- Through our **own files**
  - We will have an HTML file that references a `.js` file
- Through a **REPL** (mostly useful for debugging, testing and learning)
  - A **R**ead, **E**valuate, **P**rint **L**oop
- Through the **Command Line/Terminal** (eventually)

Chrome Developer Tools

- This is the *REPL* we will be using
- Exceptionally useful for learning and debugging
  - Not for writing projects!
- To access it:
  - Mac: `CMND + OPTION + J`
  - Windows/Linux: `CTRL + SHIFT + J`
  - `View -> Developer -> JavaScript Console`
- Let's take a look

Our Own Files

1. Create a folder somewhere memorable
2. Open that folder up in **VS Code**
3. In VS Code create two files, an `index.html` and an `index.js`
4. Set up your HTML file (hint: `!` then tab) and *save it*
5. In your JS file, write some code and *save it*
6. In your HTML file, use a script tag to reference the JS file

The `script` tag

```
<script src="index.js"></script>
```

The `.js` file

```
console.log("Hello World");
```

`console.log`?

- This is one of the ways we debug and test applications
- Anything that is provided to the `console.log` function will appear in he Developer Tools (and nowhere else!)
- It's a tool for development, not something that is used once applications are deployed
- Useful for leaving notes, seeing the order of how things execute, seeing the current value of variables etc.

Let's take a break!

See you in 10 minutes

JavaScript Basics

What does JavaScript give us?

- A syntax/structure
  - The skeleton of our code (what characters go where)
- Data Types
  - What we use to build applications
- Ways to save, access and manipulate data
  - Variables - saving things for later on!
- Ways to work with APIs

Time for some transparency...

We need to lay the foundations first, and that takes a while.

It may seem a little boring... But I promise that'll change soon enough!

There's going to be a lot of terminology too - don't feel like you need to remember all of it right away!

Data Types

What are Data Types?

- The building blocks of every application
- The types of information that the language provides
- Any value in JavaScript has a certain data type
  - e.g. a number or a string

How many Data Types are there?

There are 9 data types in JavaScript, broken down into 3 categories:

- Primitive Data Types
- Structural Data Types (sometimes called Composite Types)
- Structural Root Primitive (this is `null`)

Primitive Data Types

What are Primitive Data Types?

They are the basic building blocks of the language. They are **immutable** - this means that *they cannot be changed*

What Primitive Types are there?

- **`Number`**
- **`Boolean`**
- **`String`**
- **`undefined`**
- **`BigInt`***
- **`Symbol`***

* Very new, and rarely used. We will ignore them for the time being

Structural Data Types

What are Structural Data Types?

Data types that are *mutable* - this means that they can be changed. They are constructed by combining one or more other primitive or structural data types. They aren't really data - they just provide structure to data

What Structural Data Types are there?

There are technically only two structural data types:

- **Objects**, which includes things like `Arrays`, `Objects`, `Dates`, etc.
- **Functions**

But for our purposes...

Structural (or composite) data types can take one of three forms:

- Arrays
- Objects
- Functions

We will learn more down the track though!

Primitive Data Types

Numbers

Numbers

The number type in JavaScript represents both **integers** (whole numbers) and **floats** (a.k.a. decimals, or floating points)

```
42;

124152152152;

-15;

-101245125;

1.52;

-18.92;
```

Numbers

We can perform operations such as addition (**+**), subtraction (**-**), multiplication (*****) and division (**/**)

```
// Addition
15 + 6;

// Subtraction
25.12 - 13.25;

// Multiplication
18 * 7;

// Division
24 / 4;
```

Did you notice the...

Semicolon at the end of the line?

This is how we end *most lines* in a JavaScript file.

```
14 + 29;
```

Did you notice the...

The lines that started with //?

This is one of the ways we *write comments* in JavaScript. The computer ignores these lines!

```
// Very useful for writing notes!
```

You can easily create comments by pressing **CMND + /** on Mac, or **CTRL + /** on Windows/Linux.

Booleans

Booleans

The logical type in JavaScript. It comes in two forms: `true` and `false`. It's commonly used to *make decisions* about what should occur.

`true` means "yes, correct, continue", and `false` means "no, incorrect, skip or stop"

```
true;

false;
```

Strings

Strings

Used to represent textual data. It can be an empty string, contain one character or have many characters!

They must be some form of quotation marks (and they are *mostly interchangeable*)

```
"I am a string"; // Double Quotes

'She said "This, here, is single quoted"'; // Single Quotes

`I am a very fancy string - more on me later`; // Back Ticks
```

Strings

To **escape** special characters in strings, use the **back slash**!

```
"But he said \"What if I need to escape the quote?\"";

'What\'ll I do if I need to escape?';

`I said: "I don't have these problems, plus I can interpolate"`;
// More on this later...
```

Concatenation

This is how we combine multiple strings. If you want to add something in that isn't a string, that is okay too (JavaScript will try to convert it for you!).

```
"Hello " + "World";

"We are learning " + "JavaScript.";

"The result of 1 + 1 is " + 2;
```

You wouldn't typically use concatenation in this way, you'd use dynamic data. That's where *variables* come into play! You'll see them soon

Properties and Methods

When you create a piece of data in JavaScript, *properties* and *methods* get attached to it.

Properties are static pieces of information about the data.

```
"Hello".length;
```

Methods are operations you can ask JavaScript to perform on the data. JavaScript will return the result of that operation for us!

```
"my string".toUpperCase(); // Notice the parentheses!
```

undefined

`undefined`

It means that there is nothing here! Either the value does not exist yet, or it does not exist anymore. I treat this as *implicitly empty* and mostly just let JS use it, rather than me using it directly...

```
undefined;
```

```
null
```

`null`

Again, it means that there is nothing here. But this, to me, is explicitly empty. I use this regularly if I want to say to myself and other developers that something has absolutely no value.

```
null;
```

*`null` is technically a structural root primitive

Structural Data Types

Arrays

Arrays

An ordered collection of any information, where you can access data with a *zero-based index* (the first item is at index 0, the second is at index 1 etc.)

They are able to be iterated, or looped through

```
[]; // An empty array

["a", "b", "c"];

[1, "", null, undefined]; // Primitives we have seen

["First Todo", "Second Todo"];
```

Objects

Objects

An unordered collection of *key-value pairs* (like a word and definition in a dictionary) that can store any data

```
{}; // An empty object

{ name: "Bob" }; // An object with one key and value

{
  course: "JavaScript Development",
  provider: "General Assembly",
  topics: [ "JavaScript", "React", "APIs" ],
  numberOfHours: 60
};
```

Functions

Functions

The main building blocks of applications - they are reusable sections of code.

They are very useful for when we need to perform a single action in many places of the script, and for reducing repetition.

```
function greetClass() {
  alert("Hello Everyone!");
}

greetClass();
```

Variables

What are variables?

- Named storage for storing, accessing and manipulating data
- They are ways to store information in memory
  - You assign a name to a certain piece of data. Think of them as named containers
- To create them, often we use the `let` keyword (we can also use `const` and `var`)

The `let` keyword

Creating variables using `let`

To create a variable, we *declare* it with a name and then we *assign* a value to it.

```
let message; // Declaration
let animal;
animal = "Orca"; // Assignment

// Declaration and Assignment (suggested approach)
let course = "JavaScript Development";

let alphabet = ["a", "b", "c"];
```

Changing variables values

```
let animal = "Orca";
animal = "Wolf";

let favouriteNumber = 1;
favouriteNumber = 42;
```

```
// JavaScript is a dynamic programming language
// which means variables can change types too!

let myVariable = null;
myVariable = 10; // Don't need let for reassignment!
myVariable = "String";
myVariable = false;
myVariable = [];
```

Variable Naming

- The name must contain only letters, digits, $ or _
- The first character must not be a digit and there can be no spaces
- Use `camelCase` when the name contains multiple words. It looks like this:

```
let myMultiWordVariable;
let thisCanGoOnAndOnAndOnAndOn;
let favouriteBook;
```

Some common gotchas

- There are some reserved words that cannot be used. See here for more details
- Names are case sensitive, meaning animal and Animal are not the same variable
- Declaring a variable with the same name twice in one scope will throw an error

```
let animal = "Orca";

let animal = "Wolf";
// SyntaxError: 'animal' has already been declared
```

Some good-to-follow rules

- Name your variables in a human-readable way
- Steer clear of abbreviations
- Be descriptive AND concise (if possible)
- Use `camelCase`

The `const` keyword

What is `const`?

It's very similar to `let`, the only difference is that it has an **immutable binding** (meaning the place in memory that it points to *can't change*)

```
const favouriteNumber = 42;

const course = "JavaScript Development";

course = "Marine Biology"; // This won't work!
```

The `var` keyword

We are going to skip this for the moment...

What is `var`?

`var` is relatively similar to `let` (essentially the older way of doing things), but it has a fair few differences. The main ones are:

- It is function-scoped (rather than block-scoped)
- Tolerates re-declarations (no temporal dead zone)
- They are hoisted (can be declared below their use)

```
var message = "Hello";
```

My (biased) advice

Only use `let` and `const`

Only use letters in variable names

Use `camelCase`, though some people use UPPER_SNAKE_CASE for `const`

Loose vs. Strict Typing

In strictly-typed programming languages - when you create a variable, you must say what type it is too (and that can't change).

In loosely-typed programming languages, you don't have to explicitly say what type something is and the type can change

JavaScript is loosely-typed!

typeof

typeof

Sometimes, you need to find out what type something is! `typeof` is often the solution.

```
typeof "string";

typeof 42;

typeof undefined;

let course = { name: "JavaScript Development" };
typeof course;
```

`typeof` considerations

Sometimes, `typeof` returns values that you may not expect.

```
typeof []; // What does this give us?
```

The missing piece of the puzzle is `instanceof`.

```
const letters = ["a", "b", "c"];
letters instanceof Array; // true
```

That's all for tonight!

Homework

- Variables and Primitive Data Types
- Read Eloquent JavaScript
- Read JavaScript.info's variable page and data types page

Homework: Extra

- Read Eloquent JavaScript
- Read JavaScript.info
- Read Speaking JavaScript

A Quick Note on Homework

- For the moment, could you please:
  - Send your homework to us on Slack
- Eventually, we will be using GitHub for this

Another Note on Homework

- There is more than we expect you to get through
- As long as you have a serious go, that is okay!
  - We are here to answer questions too. Feel free to message us
- The biggest thing is that you are going to get out what you put in

What's Next?

- Comparison Operators
- Logical Operators
- Conditionals
- Loops
- Arrays

Thank You!