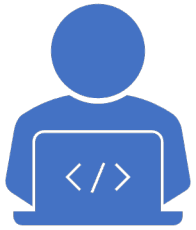

openai.mjs
openai.mjs > openai > apiKey
import OpenAI from 'openai';
const openai = new OpenAI({
apiKey: process.env['OPENAI_API_KEY'], // This is the default and can be omitted
});
async function main() {
try{
const chatCompletion = await openai.chat.completions.create({
messages: [{ role: 'user', content: 'Say this is a test' }],
model: 'gpt-3.5-turbo',
});
chatCompletion.data.choices.forEach(choice => {
console.log(choice.text)
});
} catch(e){
console.log(e.message)
}
}
main();
JavaScript
A Programming Language – An Introduction

What will we cover:

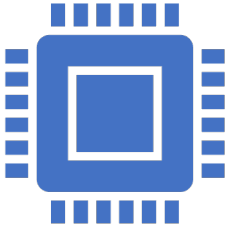
- What is JavaScript?
- What is Dynamic Typing?
- An introduction to JavaScript variables
- An introduction to JavaScript types

What is JavaScript?

JavaScript is a high-level, versatile, and dynamically-typed programming language primarily used for building interactive and dynamic content on the web.



It is a crucial component of web development alongside HTML and CSS. JavaScript allows developers to add functionality, interactivity, and behaviour to web pages.



JavaScript can also run on outside of the context of a browser thanks to the Node.js runtime that is built on top of Google Chromes V8 JavaScript engine

Dynamically Typed?

Dynamically typed means that the type of a variable is not explicitly declared and can change at runtime.

This contrasts with statically typed languages like Java, where variable types are explicitly declared and enforced at compile-time.

Dynamic Typing


```java
public static void main(String[] args) {
    String name = "john";
    name = 25;
    System.out.println(name);
}
```

Result: java: incompatible types: int cannot be converted to java.lang.String

Note that this a “compile time” error, the code can never actually run.


```javascript
let name = "john";
name = 25;
console.log(name);
```

Result: 25

The variable changes from a String to a Number at “runtime”

Compile time? Runtime?

Compile time and runtime are two distinct phases in the lifecycle of a program, referring to when specific processes occur during the development and execution of software.

Compile Time

- Occurs before program execution.
- Involves checking and translating source code.
- Detects and reports compile-time errors.
- Responsibility lies with the developer.

Runtime

- Occurs during program execution.
- Involves memory allocation, dynamic behavior, and error handling.
- Runtime errors may occur during execution.
- Developers handle runtime errors using appropriate mechanisms.

Putting that all together

JavaScript is a dynamically typed programming language.

It is an interpreted language (not compiled) that is interpreted at runtime.

(Some versions use ”Just in Time” (JIT) compilation, but we won’t go there yet)

Java is a statically typed programming language that is compiled before execution.

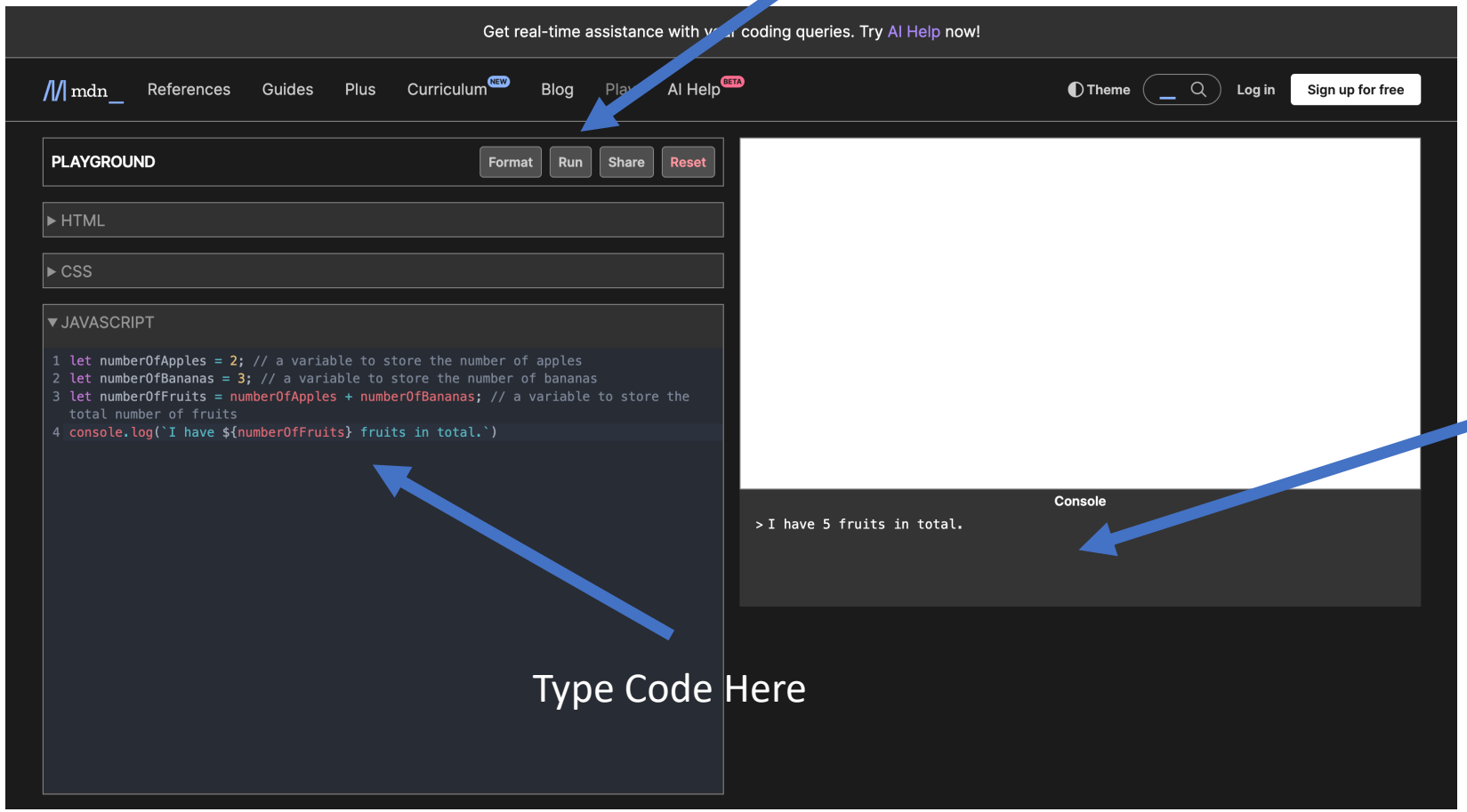
Learning JS: Variables - const

- As with other programming languages, JavaScript has variables
- Variables are containers for all the values we use within our programmes

```
const numberOfApples = 2; // a variable to store the number of apples
const numberOfBananas = 3; // a variable to store the number of bananas
const numberOfFruits = numberOfApples + numberOfBananas;
console.log(`I have ${numberOfFruits} fruits in total.`)
```

Can I try this out?

Run Code with this Button

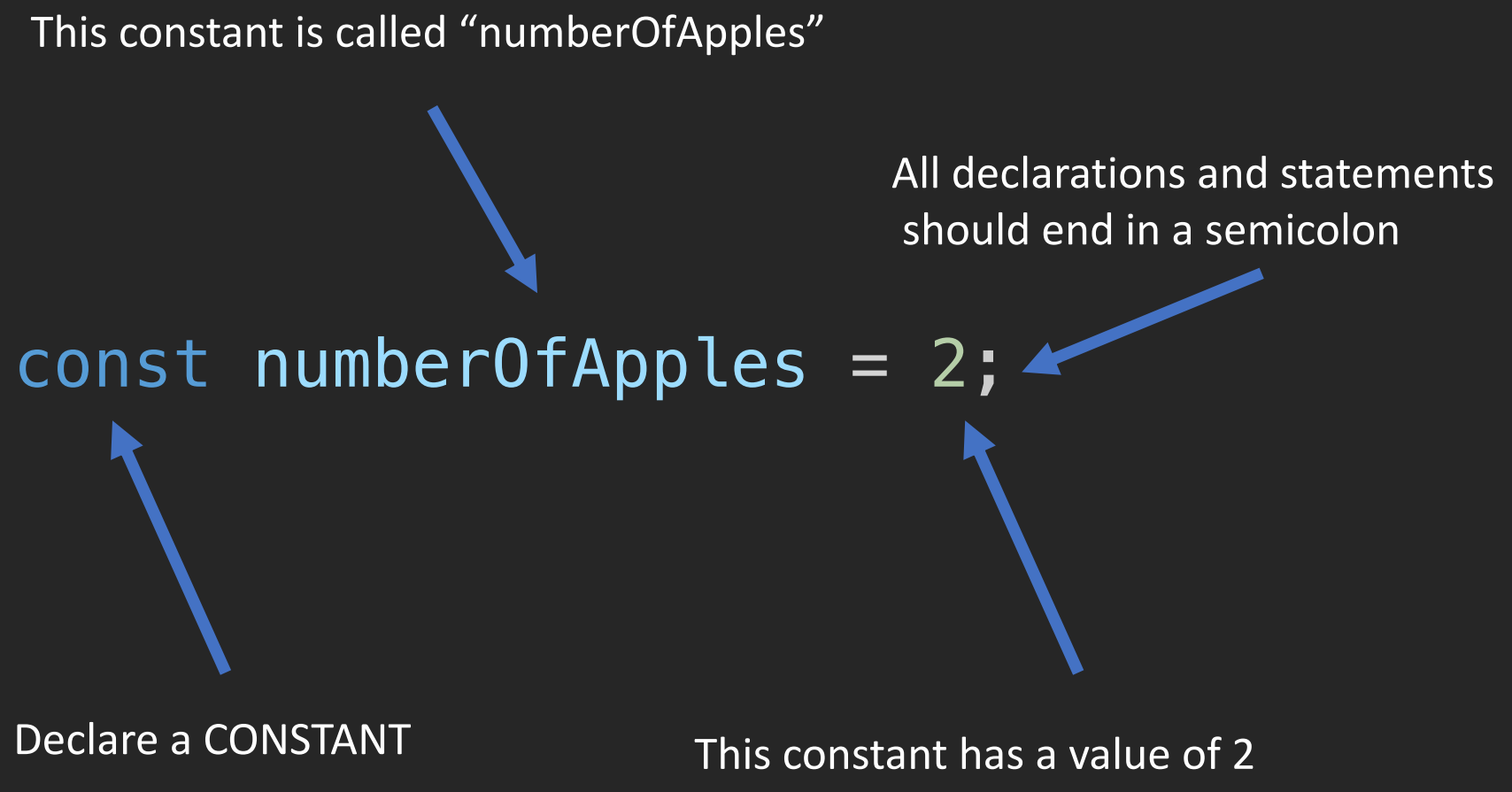


Review Output Here

Type Code Here

https://developer.mozilla.org/en-US/play

Deep Dive:


This constant is called “numberOfApples”
All declarations and statements should end in a semicolon
const numberOfApples = 2;
Declare a CONSTANT
This constant has a value of 2

What is a constant?

- A constant must be initialized when declared
- Once a constant is initialized, it's value cannot be changed
  - If the value is an object, the internals of the object can still change
- This is similar to declaring a variable "final" in Java

```
const numberOfApples = 2; // a variable to store the number of apples
const numberOfBananas = 3; // a variable to store the number of bananas
const numberOfFruits = numberOfApples + numberOfBananas;
console.log(`I have ${numberOfFruits} fruits in total.`)
```

```
final int numberOfApples = 2;
final int numberOfBananas = 3;
final int numberOfFruits = numberOfApples + numberOfBananas;
System.out.println("I have " + numberOfFruits + " fruits");
```

Constants, final....


```
final int numberOfApples = 2;
final int numberOfBananas = 3;
numberOfBananas = 4;
```

```
const numberOfApples = 2;
const numberOfBananas = 3;
numberOfBananas = 4;
```

```
public static void main(String[] args) {
    final int numberOfApples = 2;
    final int numberOfBananas = 3;
    numberOfBananas = 4;
    final int
    System.ou
}
```

Cannot assign a value to final variable 'numberOfBananas'

Make 'numberOfBananas' not final ⌥⇧⏎ More actions... ⌥⏎

The IDE tells us there's an error

```
const numberOfApples = 2; // a variable
const numberOfBananas = 3; // a variabl
numberOfBananas = 4;
const numberOfFruits = numberOfApples +
```

Compile Time Error: cannot assign a value to final variable numberOfBananas

Runtime Error:
variables.js:3
numberOfBananas = 4;
^
TypeError: Assignment to constant variable.

Learning JS: Variables - `let`

- Use the `let` keyword to create a variable that can change
- Another way of saying this is, Use the `let` keyword to create a variable that is ***mutable***

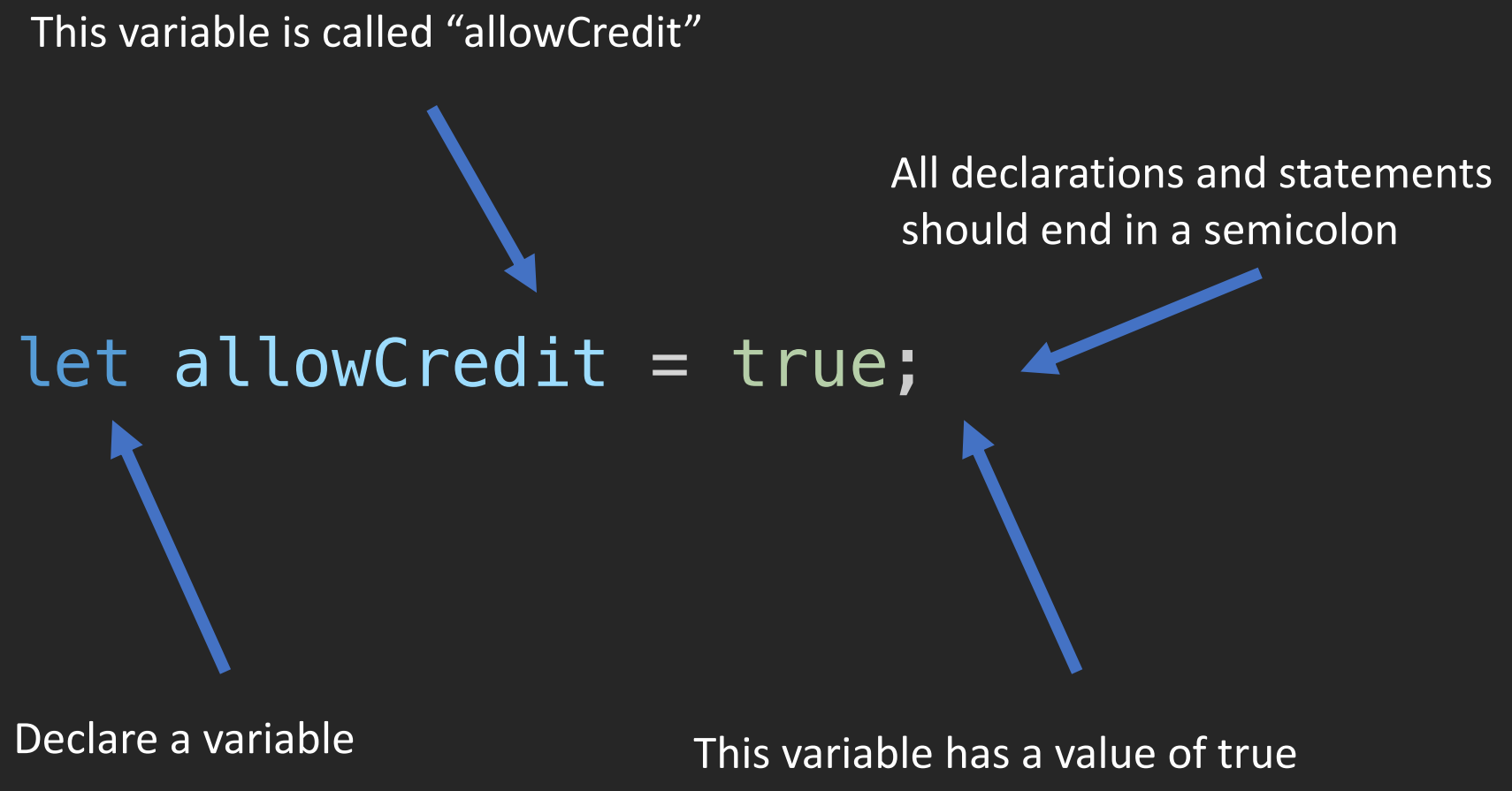
```
const price = 200;

let allowCredit = true;

if(price > 100) {
  allowCredit = false;
}

console.log(allowCredit); // false
```

Deep Dive:


This variable is called “allowCredit”
All declarations and statements should end in a semicolon
let allowCredit = true;
Declare a variable
This variable has a value of true

When to use `const` or `let`???

- **Use `const` by default**. If the value needs to remain constant, it helps prevent accidental reassignments.
- Use `let` when you know the value will change, such as in loops or when the variable's value needs to be reassigned.
- In general, it's good practice to use `const` when possible to make your code more robust and easier to reason about, only resorting to let when mutability is necessary.

When to use `const` or `let`???

- **Using `const` by default** helps to enforce ***immutability***
- If you are new to programming, immutability can be daunting, so we won't delve into it
- Just know that it has been shown that immutability can help reduce bugs and maintenance effort

adjective. im·mu·ta·ble (ˌ)i(m)-ˈmyü-tə-bəl. : **not capable of or susceptible to change**.

Learning JS: Variables

- We now have 2 ways of creating variables

`const`

- Use const when the value of the variable is not expected to change.
- It is often used for constants and values that remain constant throughout the program.

`let`

- Use let when the value of the variable is expected to change during the program execution.
- It is commonly used for loop counters and variables whose values may be reassigned.

Learning JS: Variables

```
// multiple declaration
const firstName = 'John', lastName = 'Doe';
let first = "John", last = "Doe";
```

Learning JS: Variables

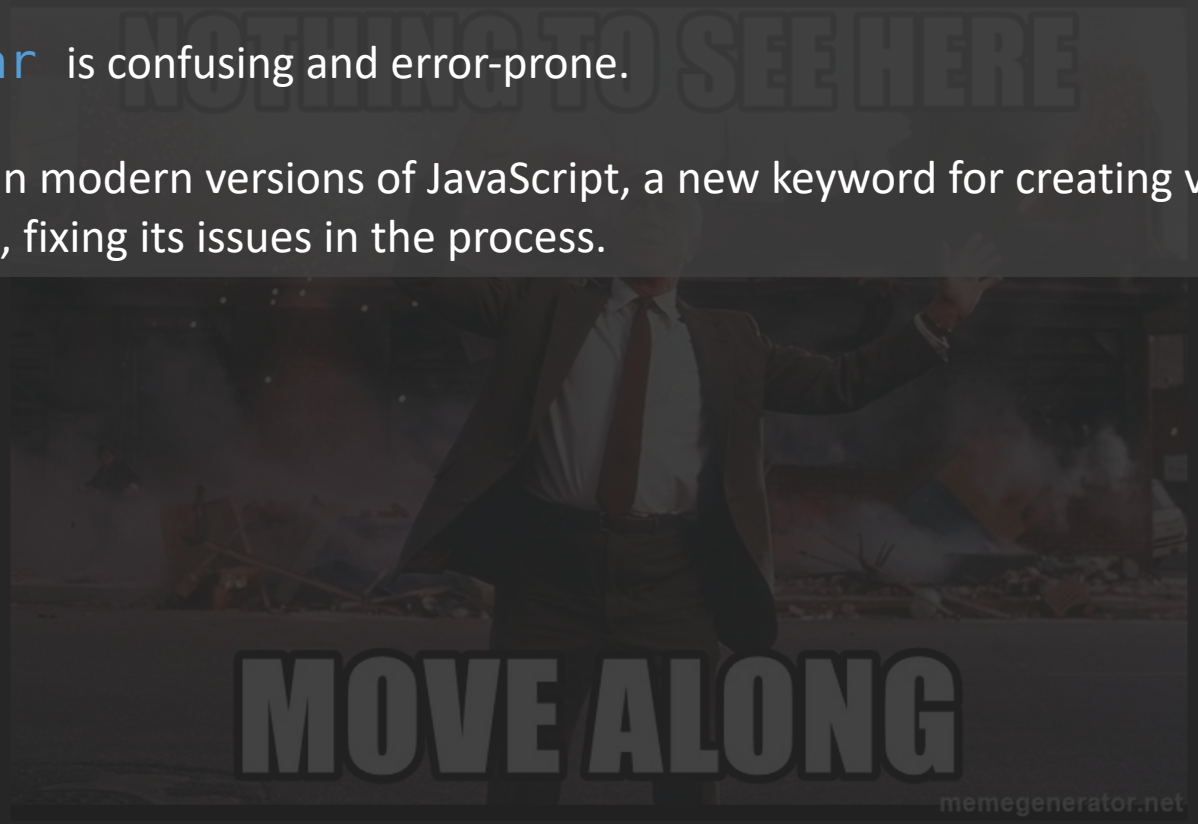
```
// initialize later
let myName;
let myAge;

if(someStatementIsTrue) {
  myName = "John";
  myAge = 25;
} else {
  myName = "Doe";
  myAge = 30;
}
```

Learning JS: Variables – What is `var` ?

```
var myName;
var myAge;
```

- Back when JavaScript was first created, this was the only way to declare variables.
- The design of `var` is confusing and error-prone.
- `let` was created in modern versions of JavaScript, a new keyword for creating variables that works somewhat differently to `var`, fixing its issues in the process.



Learning JS: Variables – Naming Variables

```
var myName;
var myAge;
```

- You can call a variable pretty much anything you like, but there are limitations.
- Generally, you should stick to just using Latin characters (0-9, a-z, A-Z) and the underscore character.
- Do not use underscores, stuck to lower camel case – lowercase first letter, any “new word” begins with capital
  - firstName, lastName, finalOutputValue
- Make variable names intuitive, so they describe the data they contain.
- Don't just use single letters/numbers, or big long phrases.
- Avoid using JavaScript reserved words as your variable names
  - `var`,`function`,`let`, and `for` as variable names.
  - Browsers recognize them as different code items, and so you'll get errors.

Learning JS: Variables – Types - Number

- Number values represent floating-point numbers like `37` or `-2.95`.
- A number literal like `200` in JavaScript code is a floating-point value, not an integer.
- There is no separate integer type in common everyday use
- Numbers in JavaScript can exhibit some bizarre behaviour, for now, we'll keep it simple with the above

```
const price = 200;

price === parseInt("200", 10); // true

typeof price; // "number"

200 / 3 // 66.66666666666667

const distance = 2.5
```

Learning JS: Variables – Types - String

- The String object is used to represent and manipulate a sequence of characters.
- Strings are useful for holding data that can be represented in text
- Strings are **IM**mutable – **IM**possible to change
- Strings will be covered in more detail later

```
const string1 = "A string primitive";
const string2 = 'Also a string primitive';
const string3 = `Yet another string primitive`;
```

Learning JS: Variables – Types - Boolean

- The Boolean object represents a truth value: true or false.

```
const started = true;
const stopped = false;
```

Learning JS: Variables – Types - Object

- In programming, an object is a structure of code that models a real-life object.
- A simple object could represent a box and contain information about its width, length, and height
- An object could represent a person and contains data about their name, height, weight, what language they speak, how to say hello to them, and more.
- Objects are often used as associative arrays or “maps” in JavaScript

```
const dog = { name: "Spot", breed: "Dalmatian" };
console.log(dog.name); // Spot
typeof dog; // "object"
typeof dog.name; // "string"
```

Learning JS: Variables – Types - Array

- Like other languages, arrays can hold multiple values and are accessed via the index of the value
- Arrays in JavaScript are zero-indexed, that is, the first index is zero
- Technically, an array in JavaScript Array is syntactic sugar for an Object that behaves like an array in other languages
- If that doesn't make sense now, don't worry, it is initially beneficial to think of them separately

```
const foods = ["apple", "banana", "cherry"];

foods[0]; // apple
typeof foods; // "object"
Array.isArray(foods); // true
foods[4] // undefined

// can mix types inside a single array
const mixed = [1, "apple", true, { name: "John" }, [1, 2, 3]];
```

Learning JS: Variables – Types

- There are other types, but for now we are fine to stick with
  - Numbers
  - Strings
  - Arrays
  - Objects
  - Booleans

Recap: What did we cover?

- Dynamic Typing
- Compile time vs Runtime
- Variables – let & const
- Types – a brief intro to
  - Number
  - String
  - Array
  - Objects
  - Boolean