# JavaScript Basics: Part 1

Sprint 4

## Connecting JavaScript to HTML Files

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!--page metadata -->
  <link href="style.css" rel="stylesheet" />
</head>
<body>
  <!-- page elements -->
  <script src="script.js"></script>
</body>
</html>
```

`<script></script>` tags are used to link JavaScript to HTML.

Note the differences between `<script>` and `<link>` tags:

- `<link>` tags are self-closing, `<script>` tags aren't
- `<link>` tags accept `href` and `rel` attributes, `<script>` tags accept `src`
- `<script>` tags are often placed as the last child of the `<body>` tag

## Connecting JavaScript to HTML Files

**There are two types of comments:**

- Single-line comments, starting with `//`

```
// everything to the right is a comment
console.log("Not a comment"); // everything to the right is a comment
```

- Multi-line comments, enclosed in `/* ... */`

```
console.log("Not a comment");
/* everything between these
characters is a comment
*/  console.log("Not a comment");
```

## Arithmetic Operators

- Addition (`+`), Subtraction (`-`), Multiplication (`*`), and Division (`/`)
- Exponentiation operator: (`a ** b`) returns `a` raised to the power of `b`
- Modulus operator: (`a % b`) returns the remainder of `a / b`
  - Example: `4 % 3` equals `1` because the remainder of `4 / 3` is `1`
- Parentheses are used to group expressions. Code inside parentheses is run first
- Order of operations (PEMDAS): parentheses, exponents, multiplication and division, then addition and subtraction

## Strings

A datatype representing text data:

- Allowable quotes: single quotes (`' '`), double quotes (`" "`), and backticks (`` ` ` ``)
- Strings can be concatenated with the `+` operator
  - Example: `"ab" + "c"` equals `"abc"`
- The escape character (`\`) can be used to allow quotation marks that would otherwise cause an error
  - Example: `'Don\'t forget to escape your quotes'`

Template literals are strings that accept embedded JavaScript expressions:

- Template literals have to be enclosed in backticks
- Unlike other strings, they can take up multiple lines

```
// syntax
`plain text ${embeddedJavaScript}`

// example
`This embedded arithmetic will be inserted into
the string: ${1 + 1}`
```

# JavaScript Basics: Part 2

Sprint 4

## Blocks

Logically and functionally independent page components.

**Variables can be declared with two different keywords (excluding `const`):**

- Variables defined with `const` can't have new values assigned to them
- Variables defined with `let` can have new values assigned to them

**Examples:**

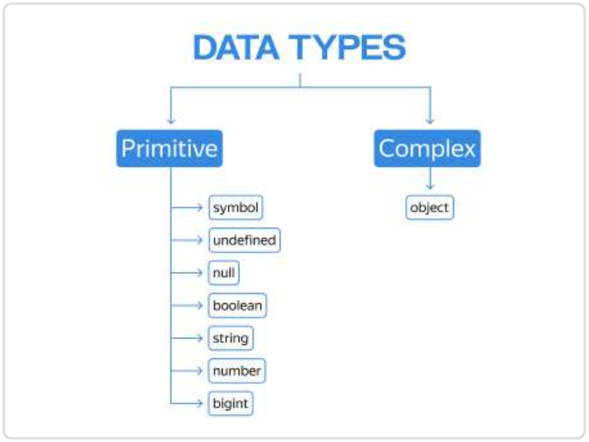
```
// declare a variable called name and assign it the value "Elise"
let name = "Elise";

// declare a variable without assigning it a value
let job;

// re-assign a value to a variable
name = "Elise";
job = "Software Engineer";

// variables declared with the const keyword cannot be reassigned
const constantString = "I cannot be reassigned";
```

**JavaScript variable naming rules:**

- Names can only contain alphanumeric characters and underscores (`_`)
- Numbers can't come first
- Names should be descriptive, and written in `camelCase`

## Primitive Data Types

- The five most common JavaScript primitive types:
  - `number` — includes both integers and floating-point numbers with decimals
  - `string` — text data
  - `boolean` — the logical values: `true` or `false`
  - `null` — variables with intentionally absent values and those assigned a value of `null`
  - `undefined` — variables that haven't had a variable assigned
- Use `typeof()` to check the type of a piece of data
- Use the appropriate conversion methods to convert data to a different type
  - Example: `Number("2")` converts the string `"2"` to the number `2`



## `null`, `undefined`, and `NaN`

Indicators of an absence of value:

- `undefined` is the value of a variable that has not been assigned a value yet
- The type of `undefined` is `"undefined"`

```
let meaningOfLife;
console.log(meaningOfLife);          // undefined
typeof(meaningOfLife);               // "undefined"
```

- `null` represents the intentional absence or non-existence of value
- The type of `null` is `"object"`

```
let someVariable = null;
console.log(someVariable);           // null
typeof(someVariable);                // "object"
```

- `NaN` is the number that is not a number. JavaScript uses it to represent uncomputable numerical calculations
- The type of `NaN` is `"number"`

```
console.log("string" * 2);           // NaN
```