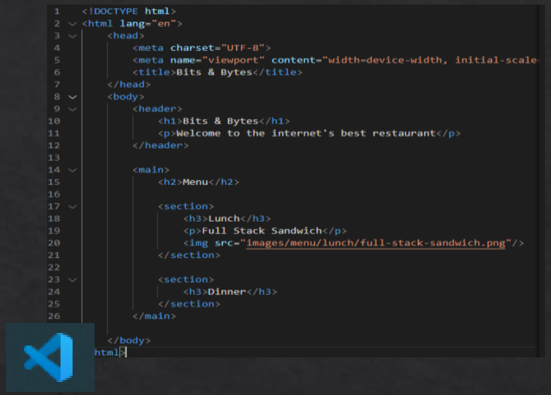
Module 3

JavaScript – Review



```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale
        <title>Bits & Bytes</title>
    </head>
    <body>
        <header>
            <h1>Bits & Bytes</h1>
            <p>Welcome to the internet's best restaurant</p>
        </header>

        <main>
            <h2>Menu</h2>

            <section>
                <h3>Lunch</h3>
                <p>Full Stack Sandwich</p>
                <img src="images/menu/lunch/full-stack-sandwich.png"/>
            </section>

            <section>
                <h3>Dinner</h3>
            </section>
        </main>

    </body>
html>
```

Session Objectives:

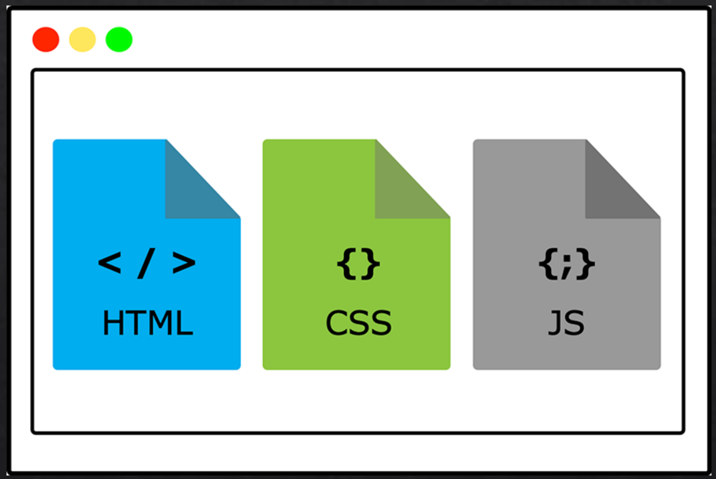
Build a “Math Practice” application!

Module 3

Introduction to JavaScript

HTML, **CSS**, and **JavaScript** are the building blocks of the web. Each of these languages play their own significant roles in building the web:

- **HTML**: Provides the basic structure or markup of a document.
- **CSS**: Provides formatting of the document to control presentation and layout.
- ⇨ **JavaScript**: Provides behavior to the document.



Module 3
Introduction to JavaScript

Variables - Variable Names

Naming rules for JavaScript variables:

- Variable names consist of letters A-Z, a-z, characters _, $, and digits 0-9.
- Variable names must start with a letter, _, or $.
- Variable names are case-sensitive.
- Variable names may not be a reserved keyword.

The following are considered best practices in JavaScript:

- Use camelCase for multi-word variable names.
- Use uppercase for constants and separate words with an underscore, _.
- Boolean variable names should begin with is.

Module 3
Introduction to JavaScript

Variables - Declaring Variables

There are two ways to declare a variable in JavaScript. In either case, the basic form is the reserved words, **let**, or **const**, a variable name, and a semi-colon, **;**

Using let

You use let when you know the value of the variable needs to be changeable. This is its basic form:

```
let age; // Declare without initializing
let breed = ’Poodle’; // Declare and initialize
```

Using const

The reserved word const—short for "constant"—is the alternative to let. Variables declared with const must be initialized with a value, and can't be reassigned later. Here's an example:

```
const PI = 3.14159;
```

Avoid using var
Avoid variable shadowing

Module 3
Introduction to JavaScript

Data types

The most important JavaScript data types to be familiar with are:

- Number
- String
- Boolean
- Object
- null
- Undefined

In JavaScript, variables aren't associated with any particular data type when you declare them. This makes JavaScript a loosely typed language.

Module 3

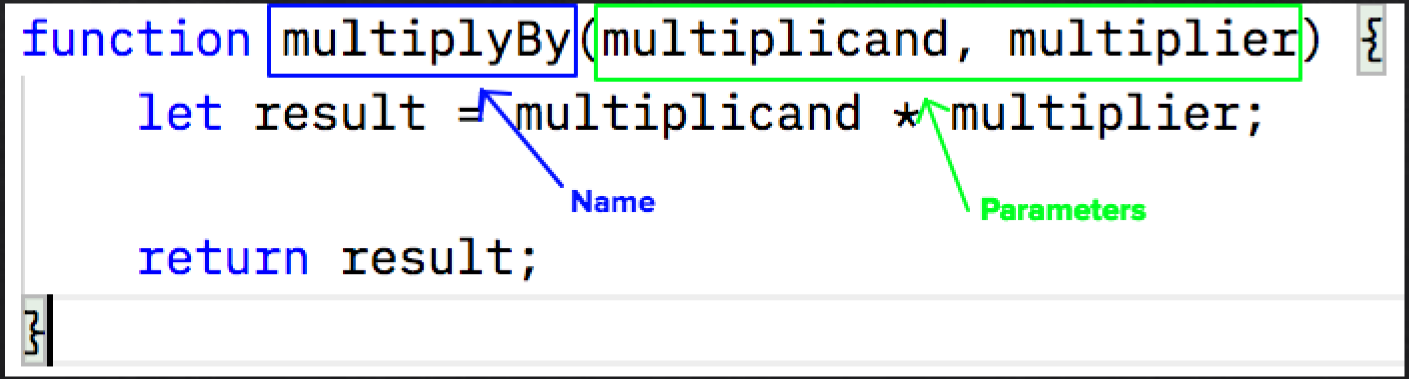
JavaScript Functions

Named functions

Two parts of a named function: the **function signature** and the **function body**.

The components of a function signature are:

- The function name
- The function parameters



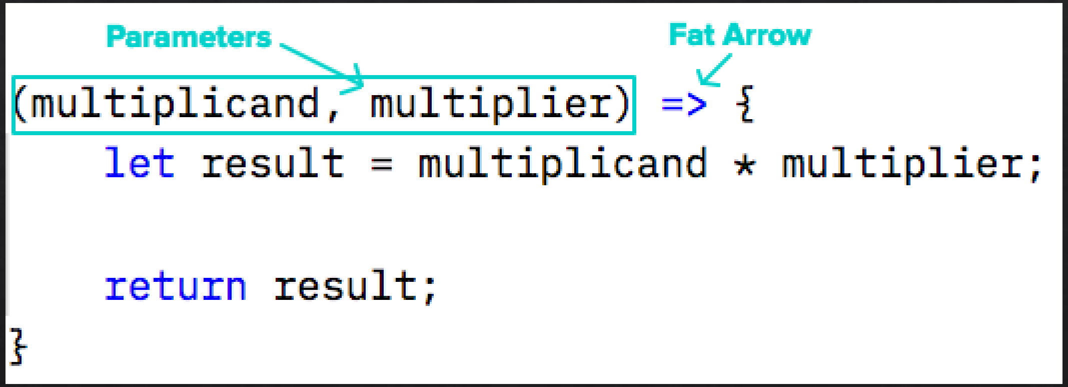
```
function multiplyBy(multiplicand, multiplier) {
    let result = multiplicand * multiplier;

    return result;
}
```

Module 3
JavaScript Functions

Anonymous functions

Anonymous functions are functions that don't have a name. Functions in JavaScript can be used like any other value, so creating a function without a name is possible. You create an anonymous function with the following syntax:



```
(multiplicand, multiplier) => {
    let result = multiplicand * multiplier;

    return result;
}
```

Module 3

JavaScript Functions

Documentation

One of the core responsibilities of a programmer is writing documentation for the code they create. Documentation is more than comments on the code, and there are a lot of comments that are considered bad practice.

Line Comments

```
// Set number of phones to one
let number = 1;
```

```
// Set number of phones to one
let numberOfOwnedPhones = 3;
```

✓

```
let numberOfOwnedPhones = 3;
```

Self-Documenting Code

✓

```
// Needed later to build the display table
let numberOfOwnedPhones = 3;
```

Module 3

JavaScript Functions

Documentation

Function Comments - JSDoc

Comments on functions are typically integrated into the IDE and are used to create documentation of your code for other programmers to use. They follow a standard format called JSDoc.

```
/**
 * Takes two numbers and returns the product of
 * those two numbers.
 *
 * Will return NaN if exactly two numbers are not
 * given.
 *
 * @param {number} multiplicand a number to multiply
 * @param {number} multiplier a number to multiply by
 * @returns {number} product of the two parameters
 */
```

Module 3

DOM

The DOM

DOM stands for the Document Object Model. It's the browser's internal representation of the structure of the current web page.

The DOM is an internal data structure that browsers use to represent the structure and content of a web page. When the browser loads an HTML document, it needs to translate that into something that it can use to draw a graphical representation of the page.

Understanding the DOM is important. HTML is static. It's only read once when the page loads, and then it's converted into a DOM. CSS and JavaScript perform their tasks using the DOM, not the static HTML.

Module 3
DOM

Selecting DOM elements
JavaScript has many built-in functions that you can use to manipulate the DOM.   The following is a list of the most commonly used functions.

getElementById()
The first function you'll learn about is getElementById(). This function gets a single HTML element from the DOM and returns a reference to it.

querySelector()
Is for selecting single elements that don't have an ID. querySelector() takes a standard CSS selector and returns the first element it finds that matches that selector.

querySelectorAll()
If you want to get all of the list items, you can use querySelectorAll() instead. This returns a NodeList of all the elements, which you can use as an array:

Module 3

JavaScript – Event Handling

Important Definitions

- **Event Handling**
  A style of user interaction that browsers use to allow developers to react and interact with a user's use of their site.

  Event handling is managed all in JavaScript and it not written into CSS or into the HTML. It is how UIs are managed on the web.

- **Event Handler**
  A function that can be attached to a DOM element and run when a defined type of event happens on that DOM element.

  The function can be an anonymous function or a named function.

- **Event Objects**
  An object given to every Event Handler that defines properties about that event, like location, which DOM element the event happened on, and key presses or mouse click information.

  Event object are always passed in as the first parameter to an Event Handler function.

- **Event Propagation**
  The process of an event firing on a DOM element and every parent of the DOM element up to the window object.

  You can stop propagation at a certain level by calling event.stopPropagation() at that DOM element.

- **Default Action**
  The default process that a browser will perform if no Event Handler prevents it from happening.

  You can stop default behavior by calling event.preventDefault().

  Default behavior is most important to watch out for on buttons, forms, and a elements.

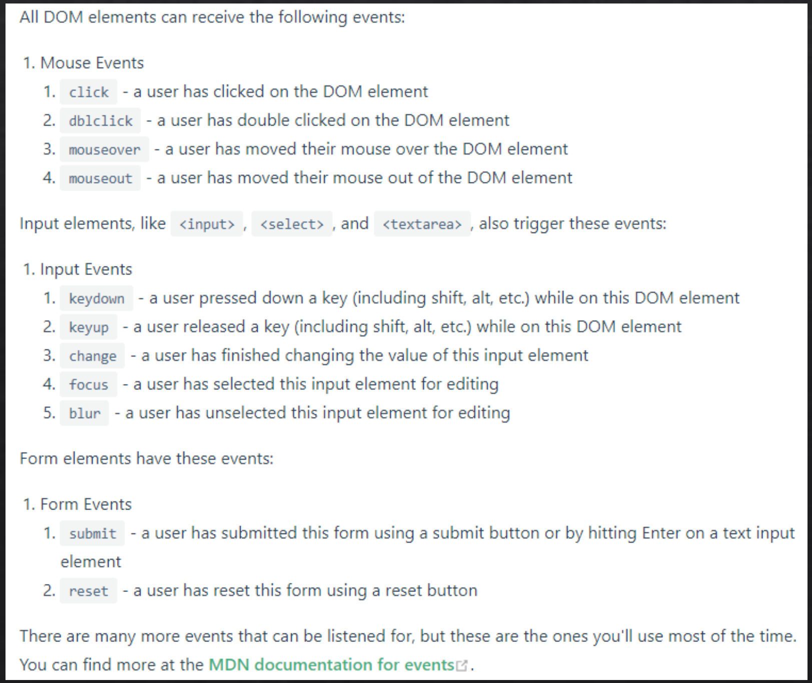
Module 3

JavaScript – Event Handling

Listening for events in JavaScript

Reacting to events in JavaScript requires three things:

1. A DOM element that you want to listen to events on
2. A specific event that you want to listen to
3. A function that holds the logic that you want to execute



All DOM elements can receive the following events:

1. Mouse Events
   1. `click` - a user has clicked on the DOM element
   2. `dblclick` - a user has double clicked on the DOM element
   3. `mouseover` - a user has moved their mouse over the DOM element
   4. `mouseout` - a user has moved their mouse out of the DOM element

Input elements, like `<input>`, `<select>`, and `<textarea>`, also trigger these events:

1. Input Events
   1. `keydown` - a user pressed down a key (including shift, alt, etc.) while on this DOM element
   2. `keyup` - a user released a key (including shift, alt, etc.) while on this DOM element
   3. `change` - a user has finished changing the value of this input element
   4. `focus` - a user has selected this input element for editing
   5. `blur` - a user has unselected this input element for editing

Form elements have these events:

1. Form Events
   1. `submit` - a user has submitted this form using a submit button or by hitting Enter on a text input element
   2. `reset` - a user has reset this form using a reset button

There are many more events that can be listened for, but these are the ones you'll use most of the time. You can find more at the MDN documentation for events.

Module 3
JavaScript – Event Handling

*Best practice

Adding event handlers to DOM *

```
function changeGreeting() {
  let greetingHeader = document.getElementById('greeting');
  greetingHeader.innerText = 'Goodbye';
}

let changeButton = document.getElementById('change-greeting');

changeButton.addEventListener('click', (event) => {
  changeGreeting();
});
```

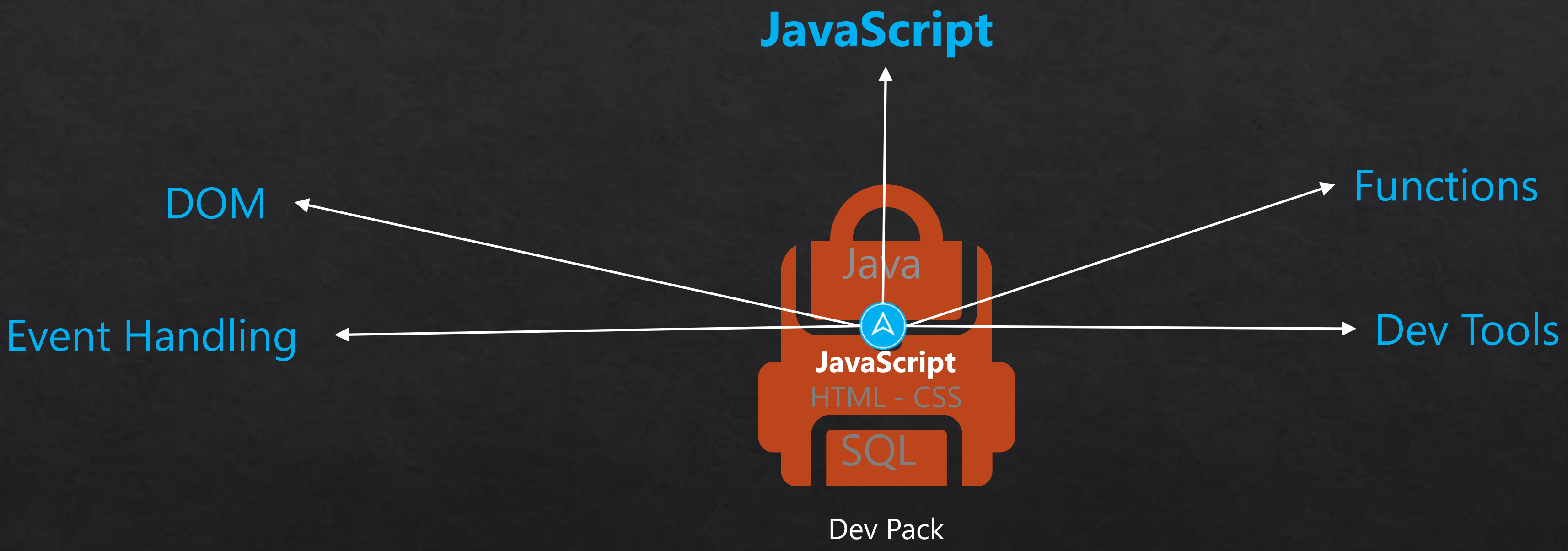
Event handling using anonymous functions
You could also get the same functionality by attaching an anonymous function as the event listener instead of calling a named function:

```
changeButton.addEventListener('click', (event) => {
  let greetingHeader = document.getElementById('greeting');
  greetingHeader.innerText = 'Goodbye';
});
```

Module 3

JavaScript – Review

New Tools!


JavaScript
DOM
Functions
Event Handling
Dev Tools
Java
JavaScript
HTML - CSS
SQL
Dev Pack