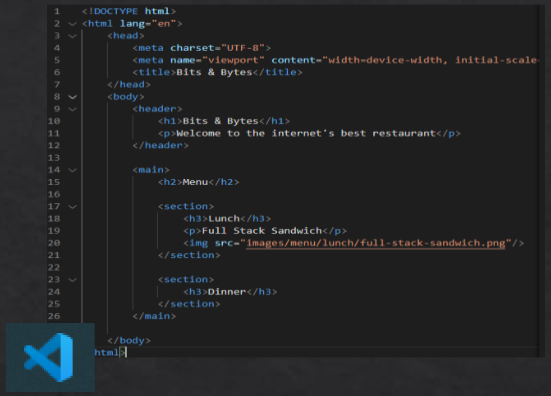
Module 3

JavaScript Functions



```html
<!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:

- Functions Introduction
- Named Functions
- Function Parameters
    - Optional Parameters
    - Parameter Default Values
    - arguments variable
- Anonymous Functions
- Array Functions
- Function Documentation

Module 3
JavaScript Functions

Functions

To write maintainable code, one of the primary things to avoid is code duplication. Programming languages provide many features and constructs to achieve this, and one of the most common is functions.

In the context of programming, a "function" is a way to package up a block of code, allowing you to reuse that block over and over again. This is especially helpful when you have a piece of logic that's needed in more than one place in a system.

There are two types of functions that you'll learn about in JavaScript: named functions and anonymous functions.

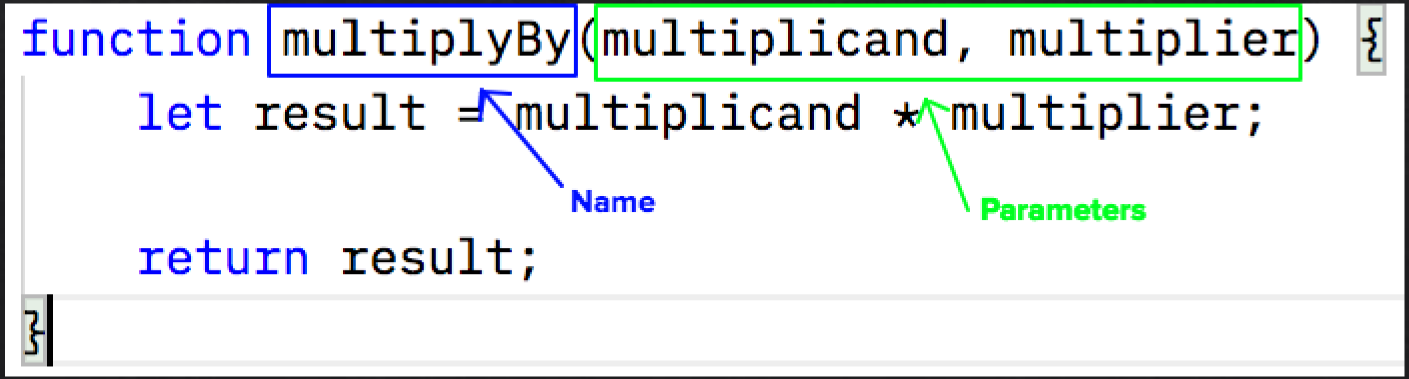
Module 3
JavaScript Functions

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

Named functions – Function Signature

Function name
Like variables, functions have names that can be used to reference them. Also, like variable names, careful consideration should be given to choosing names for functions. Function names should be:

- **descriptive** - it should be clear what type of action or calculation the function performs when invoked
- **camelCase** - the first letter of the name is lowercase and the first letter of each subsequent word is uppercase
- **unique** - function names need to be unique across all JavaScript code that's loaded into the page. If a name conflicts with another function, the one that's loaded last overwrites the other one

```
function addTwoNumbers(x, y){
  //Logic
  let result = x + y;
  return result;
}
```

Module 3
JavaScript Functions

Named functions – Function Parameters

Function parameters
Parameters are variables that can provide input values to a function. When functions are created, parameter lists indicate what inputs the function can accept.

Optional parameters
Parameters in JavaScript are always optional. So what if a value isn't provided for a parameter when calling a function? As you saw in the previous discussions, if you don't assign a value to a variable, the variable is set to undefined.

Default parameter values
In some cases, there's a reasonable default value that you can use if a parameter value isn't supplied. You can use default parameters to make your functions more robust and useful in varying scenarios.

Module 3
JavaScript Functions

Named functions – Function Parameters

Handling an unknown number of parameters
There may be times when you want to handle an unknown number of parameters. A good example of this is writing a function that concatenates an unknown number of strings together. If you call a function like this:

```
let name = concatAll(firstName, lastName);
```

What if you want/need to call concatAll like the following:

```
let name = concatAll(honorific, firstName, mothersMaidenName,
 '-', fathersLastName);
```

Module 3

JavaScript Functions

Named functions – Function Parameters

Handling an unknown number of parameters

The issue here is that there's no way to know how many parameters you have for that function. But in JavaScript, you can use a special variable to get at all of the given parameters, whether you expect them or not. This variable is called arguments.

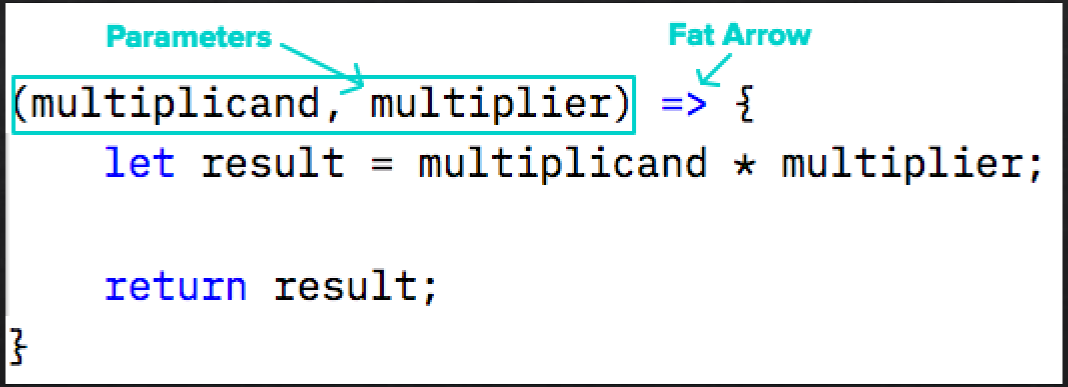
With arguments, you can treat all parameters that have been passed to the function as an array, even if you don't have any parameters defined in the actual function definition.

```
function concatAll() { // No parameters defined, but we still might get some
  let result = '';
  for(let i = 0; i < arguments.length; i++) {
      result += arguments[i];
  }
  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

Array functions using anonymous functions

Arrays in JavaScript have many useful functions themselves that use anonymous functions.

forEach
Performs like a for loop, running a passed in anonymous function for every element of an array.

```
let numbers = [1, 2, 3, 4];

numbers.forEach( (number) => {
    console.log(`This number is ${number}`);
});
```

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

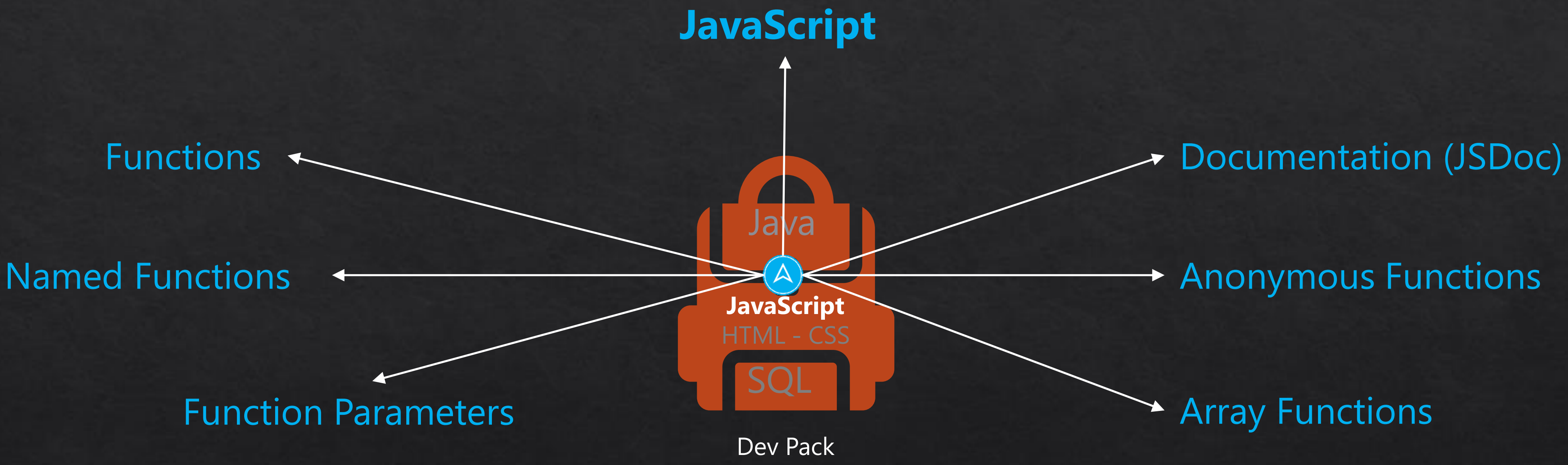
Comments on functions are typically integrated into the IDE and are used to create documentation of your code for other programmers to use. They follow a standard format called JSDoc.

```
/**
 * 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

JavaScript Functions

New Tools!



JavaScript
Functions
Documentation (JSDoc)
Named Functions
Anonymous Functions
Function Parameters
Array Functions
Java
JavaScript
HTML - CSS
SQL
Dev Pack