**Functions**

1. **Function Declaration & Calling**

Be aware of the hoisting feature in JavaScript, which allows access to function declarations even before they’re defined (use functions before declarations)

greetWorld(); // Output: Hello, World!  
  
function greetWorld() {  
  console.log('Hello, World!');  
}

1. **Parameters and Arguments**

Diagram

Description automatically generated

Default parameters: will be used if no argument at that position is passed.

1. **Return**

By default, the return value is undefined.

Diagram

Description automatically generated

1. **Function Expressions**

This is another way to define a function. In a function expression, the function name is usually omitted 🡪 anonymous function. The **function** expression is instead **stored in a variable.**

Diagram, timeline

Description automatically generated with medium confidence

To call this function, we call with the variable (identifier) and pass the arguments as normal.

1. **Arrow Functions**

This is a syntax, a shorter way to use the ‘fat arrow’ notation () =>

This removes the need to type out the keyword function every time we need to create a function. Instead, we include the parameters in the () and then add an arrow => that points to the function body surrounded by {}

const rectangleArea = (width, height) => {  
  let area = width \* height;  
  return area;  
};

1. **Concise Body Arrow Functions**

There are a few techniques to shorten function body

+) Functions that take only 1 parameter will not require (). However, functions with 0 or 2+ will.

+) Function body composed of single-line block does not need curly braces. The return should be omitted, so that we’ll have an implicit return

const squareNum = num => num \* num;

+) Include ternary operator to return: we **CANNOT** do things like

**day === ‘Wednesday’ ? return ‘yea’ : return ‘nonono’**

We instead have to integrate ternary operator as:

**return day === ‘Wednesday’ ? ‘yea’ : ‘nonono’**