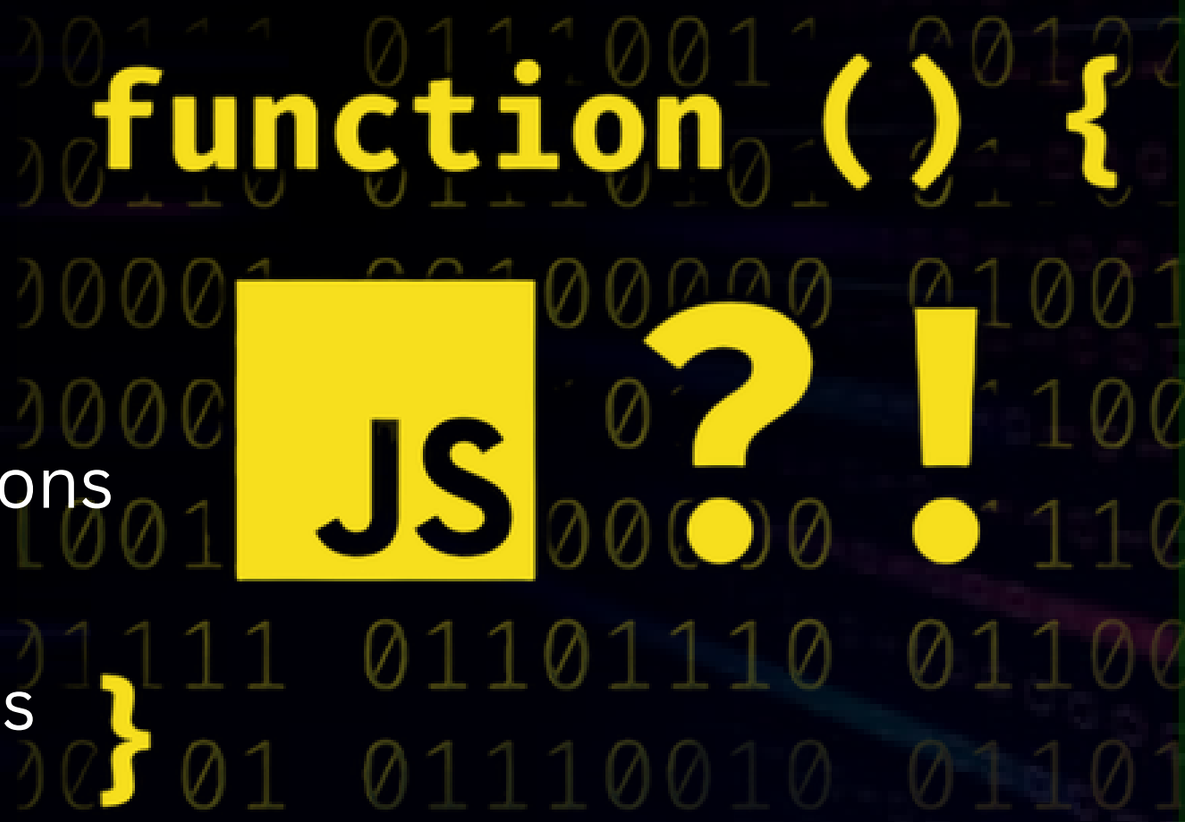
FUNCTIONS IN

JAVASCRIPT

- What is a function?
- Parameters vs Arguments
- Ways to implement a function
  - Function Declaration
  - Function Expression
- Different types of function expressions
  - Named function expressions
  - Anonymous function expressions
  - Arrow function
  - IIFE
  - Higher Order Function



@adityauke

What is a function?

A function is a reusable piece of code that takes some input, performs some tasks, and returns a value. A function stores a piece of code that we want to use repeatedly.

To use a function, we call that function by its name.

script.js > ...

```
// Defining a function
function getName(age){
    let name = 'Aditya';
    return `${name} is ${age}`;
}

// Calling a function
let name = getName(21)
console.log(name) // Aditya is 21
```

@adityauke

Parameters vs Arguments

When you define a function, the input values are called parameters.
Parameters are something that is accepted by a function.

When you invoke or call a function, the input values are called arguments.
Arguments are something that is passed to a function.

script.js > ...

```
1  // Here inputs are called parameters
2  function sum(a,b){
3      let c = a + b;
4      return c;
5  }
6
7  // Here inputs are called arguments
8  let result = sum(9,9)
9  console.log(result) // 18
```

@adityauke

Ways to implement a function

Function Declaration

While defining a function if the first keyword of the statement is a function keyword, then it will be called the function declaration.

```
// Function Declaration
function welcome(name){
    // some task
}
```

Function Expression

While defining a function if the first keyword of the statement is not a function keyword, then it will be called the function expression.

```
// Function Expression
const fun = function(name){
    console.log("Hey");
}

// To access the function use the variable name
fun() // Hey
```

@adityauke

Different types of function expressions

- Named function expressions

```
// Named Function Expression
const fun = function greet(name){
    console.log("Hey");
}

// To access the function use the variable name
fun() // Hey
```

- Anonymous function expressions

```
// Anonymous Function Expression
const fun = function(name){
    console.log("Hey");
}

// To access the function use the variable name
fun(); // Hey
```

@adityauke

Different types of function expressions

- Arrow function

  Arrow functions were introduced in ES6.

```
const getName = (age) => {
    let name = 'Aditya';
    return `${name} is ${age}`;
}

let name = getName(23);
console.log(name); // Aditya is 23
```

1. If the arrow function has only one statement, you can skip the curly braces and the return keyword.
2. if the arrow function has only one parentheses as well. parameter, you can skip the
3. If the arrow function has only one statement and if it is returning an object literal, use parentheses to wrap that object literal otherwise it will not work as expected.

@adityauke

Different types of function expressions

- Higher Order Function

A function that accepts functions as arguments and/or returns a function is known as a Higher-order-function.

```
function add(a,b){
    return a + b;
}
// Higher Order function
function calculation(callback){
    return callback
}
let sum = calculation(add)
console.log(sum(9,9)) // 18
```

Array.map(), setTimeout(), and Array.sort() are examples of Higher order Functions.

@adityauke

Different types of function expressions

- IIFE - Immediately Invoked Function Expression

It is a function that is invoked immediately as soon as it is defined without being explicitly called.

It is defined by wrapping the function in parentheses and then adding a set of parentheses at the end to immediately invoke it. We can pass arguments to the additional set of parentheses.

```
(function(age){
    let name = 'Aditya';
    console.log(`${name} is ${age}`);
})(21)

// Aditya is 21
```

If you will not write the additional set of parentheses at the end, your function will not be invoked.

@adityauke