CYDEO

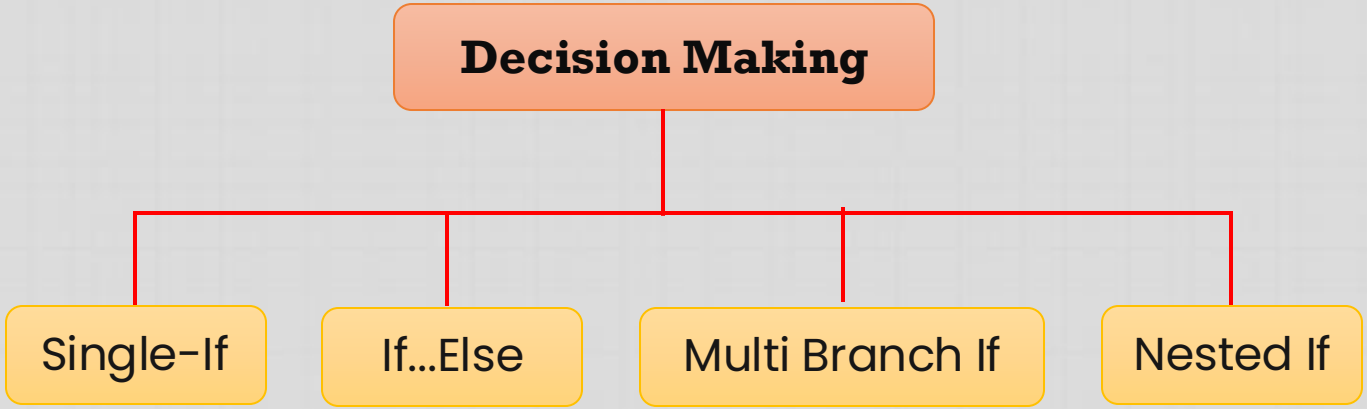
JavaScript Programming Day02

Content

- If Statements
- Loops
- String
- Arrays
- Functions
- Class & Object
- Tabnine - AI Coding Tool

If Statements

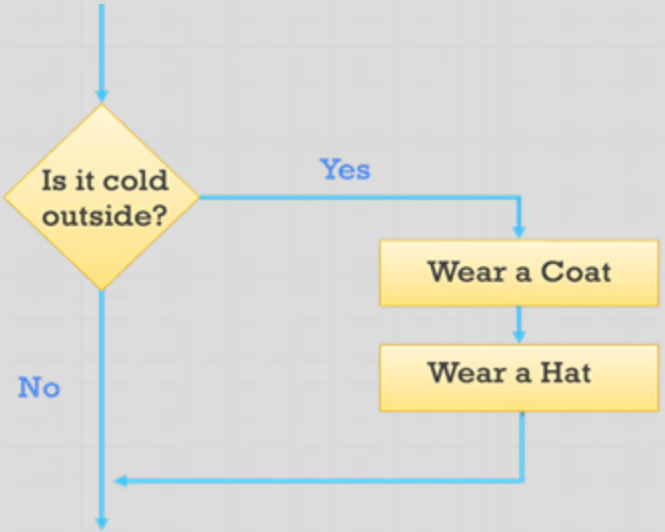
- Used for making decisions based on specified criteria



Single If

- The if statement evaluates a condition
- If the condition evaluates to true, any statements in the subsequent code block are executed

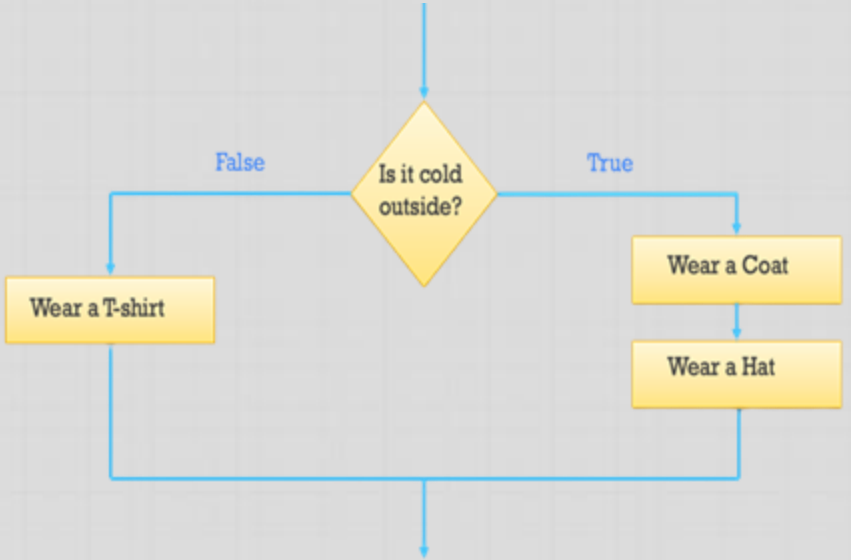
```
if(Condition){
    Statements
}
```



If...Else

- The if...else statement checks a condition
- If it resolves to true the first code block is executed
- If the condition resolves to false, the second code block is run instead

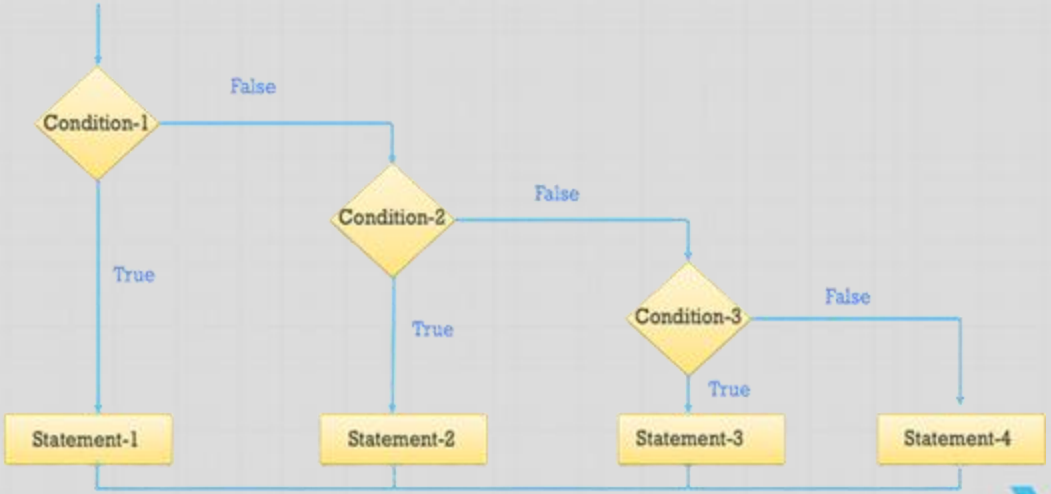
```
if(Condition){
    Statements
}else{
    Statements
}
```



Multi-branch If

- Multi-branch if statement can be used to create an else if clause
- It is used to make decision among several alternatives

```
if(Condition1){
    Statements
}else if(Condition2){
    Statements
}else{
    Statements
}
```



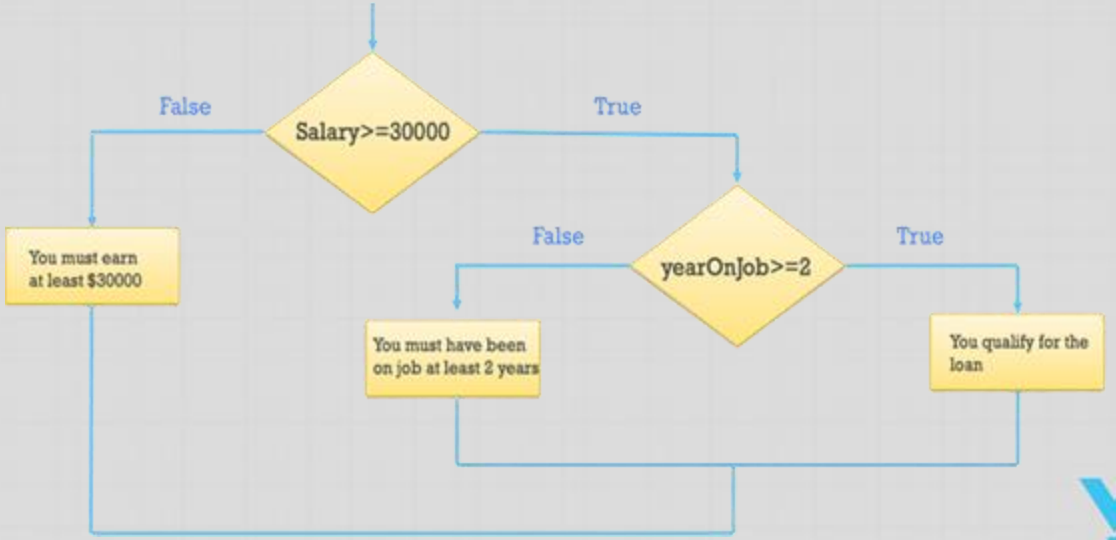
Multiple else if blocks can be given

Nested If

- Nested if statements can be used for creating a pre-condition
- It's used if one condition can be evaluated to several alternatives

```
if(Condition){
    if(Condition){
        Statements
    }
}
```

Outer and Inner If statements can be any type of if statement



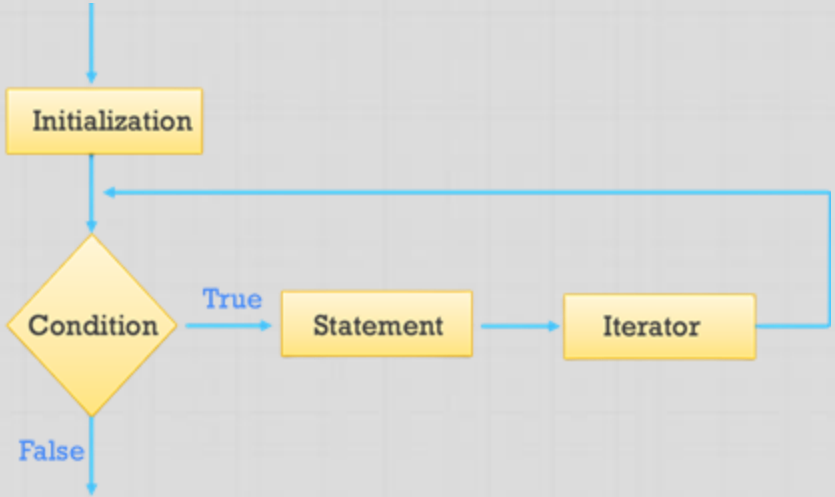
Loops

Loops

- Used for repeating a set of statements
- There are two types of loops:
  - For Loop
  - While Loop
  - Do-While Loop

For Loop

- Runs the given code a specific number of times
- Initialization is the starting point of the loop
- Condition is the ending point of the loop
- Iterator is responsible for making the condition false



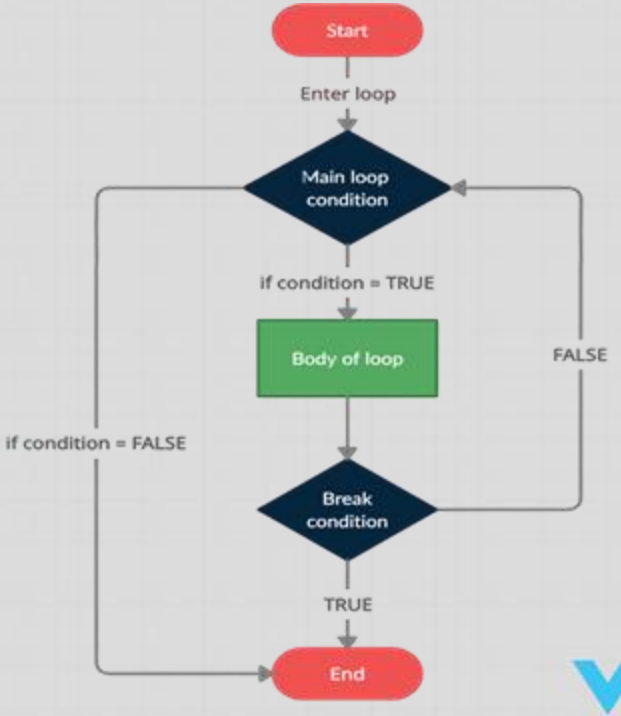
For Loop Syntax

```
for(initialization; condition; iterator){
        Statements
}
```

- The initialization expression initializes the loop
- When the condition expression evaluates to false, loop stops running
- The Iteration gets executed after each iteration through the loop

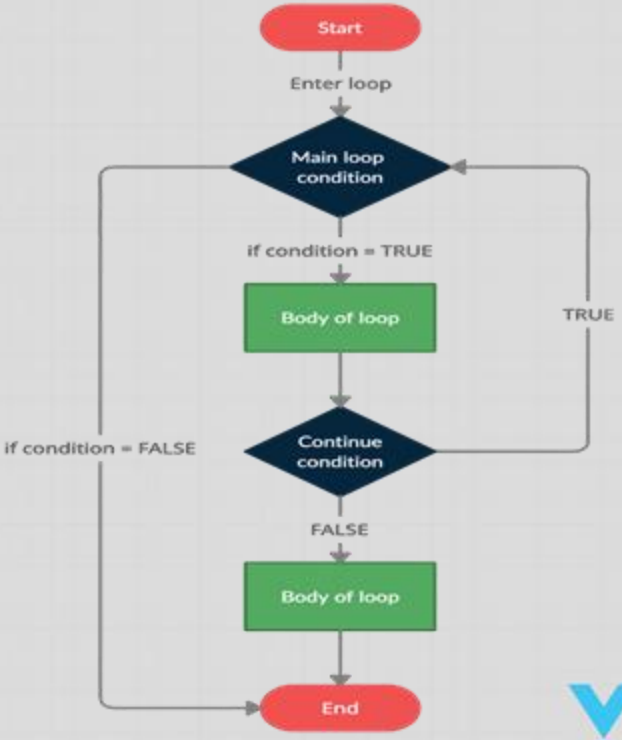
Break Statement

- Causes the termination of the loop
- Tells the interpreter to go on to the next statement of code outside of the loop



Continue Statement

- Skips the current iteration of the loop
- Tells the interpreter to jump to the next iteration



String

String

- String is a sequence of characters, surrounded by double quotes (“”) or single quotes (‘’)
- A string object is immutable; once it is created, it can’t be altered

```
let greeting = "Hello World";

let name = "Wooden Spoon";
```

```
let greeting = 'Hello World';

let name = 'Wooden Spoon';
```

String: Sequence of Characters

- Strings are ordered sequences of characters, and each character has a unique index number

```
let s = 'CYDEO !';
//index: 0123456
```

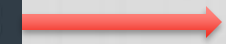
String Methods

Method Name	Method Name	Method Name
charAt()	length	toLowerCase()
toUpperCase()	trim()	indexOf()
lastIndexOf()	replace()	substring()
repeat()	split()	startsWith()
endsWith()	includes()	

String Concatenation

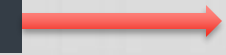
- The action of linking things together by using “+” or “${}” operator

```
let age = 20;
console.log("I am " + age + " years old");
```



I am 20 years old

```
let age = 20;
console.log(`I am ${age} years old`);
```



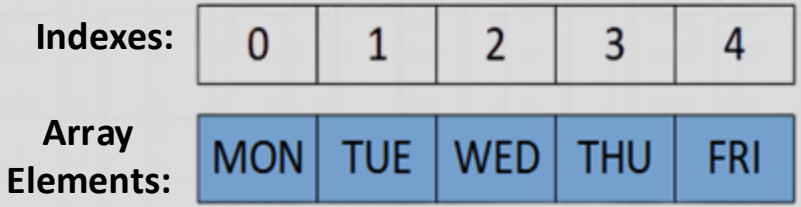
I am 20 years old

Arrays

Arrays

- Special type of variable that's used to store multiple values of any types
- The values in the array are ordered, changeable, can be duplicated, and can be of any data type
- Each element has a unique index number

```
let days = ["MON", "TUE", "WED", "THU", "FRI"];

let fruits = ["Cherry", "Lemon", "Cherry"];

let scores = [75, 78, 85, 90, 93, 95, 85];

let myArray = ["A", "B", 1, 2, true, false];
```



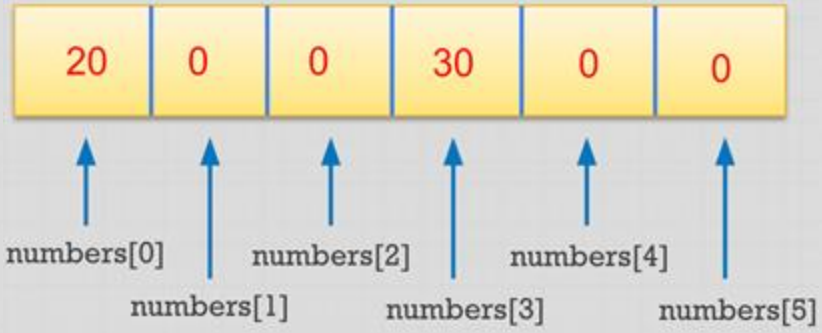
Accessing Array Elements

- Elements of an array can be accessed by using the square brackets [ ]
- Index number needs to be provided within the square brackets

```
numbers[0] = 20;
```

arrayRefVar, index, value

```
numbers[3] = 30;
```

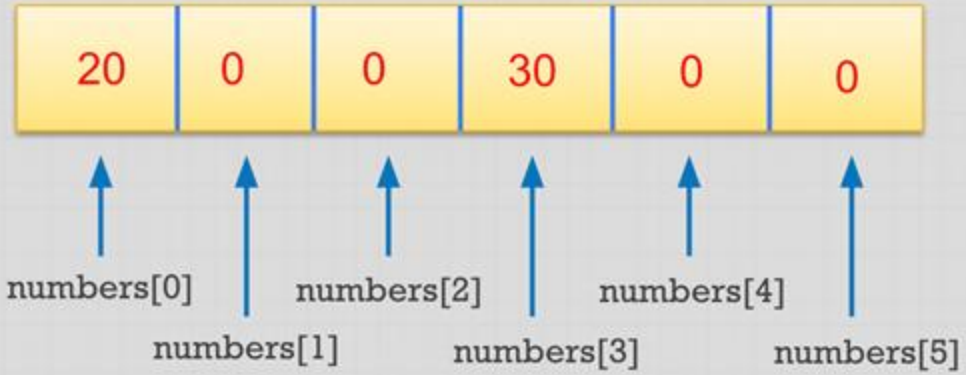


Assigning Values to Array Indexes

```
numbers[0] = 20;
```

arrayRefVar — index — value

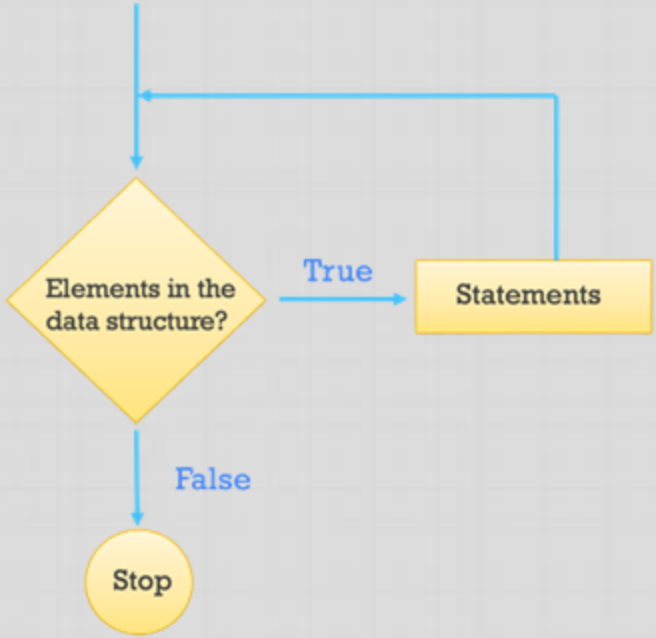
```
numbers[3] = 30;
```



For...of Loop

- Used to access each successive value of a sequence
- Iteration order and number of iterations are fixed
- Does not have index numbers

```
for (let element of sequence) {

    // Statements

}
```



Array Methods

Method Name	Description
push(element)	Adds the specified element to the end of the array
unshift(element)	Adds the specified to the beginning of the array
splice(index, # of elements)	Remove the specified number of element starting from index
shift()	Removes the first element from the array
pop()	Removes the last element from the array

Functions

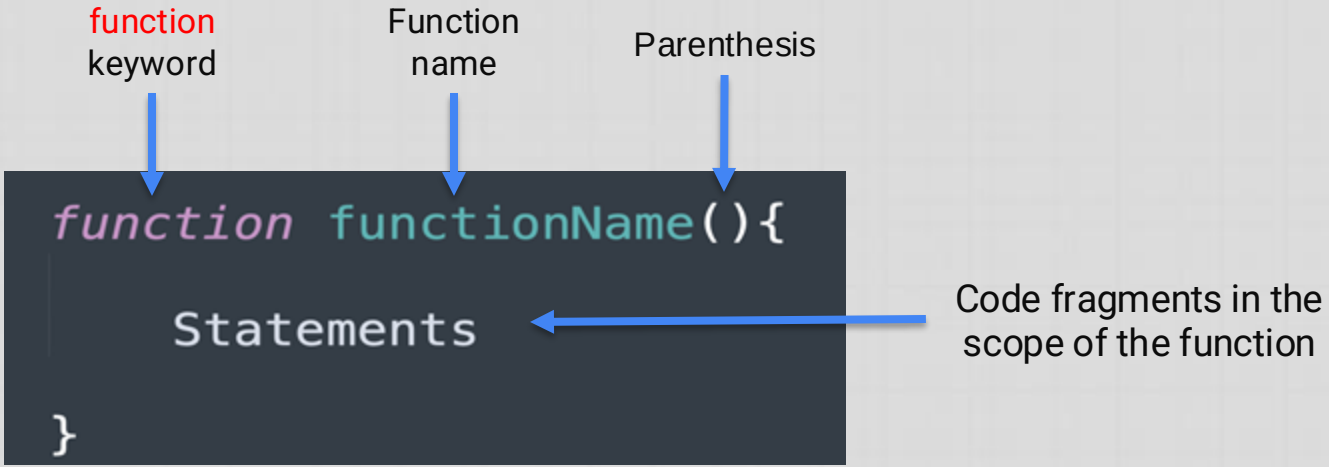
Function

- Grouping a series of code fragments to perform a task
- Allows us to reuse the function rather than repeating the same set of statements
- Improves the reusability and efficiency of our codes

```
printMessage();

cube(10);

execute();

reverse('CYDEO');

takeScreenSHort();

...
```

Declaring A Function

- A function must be declared before we call it and use it
- To create a function, we need to use the function keyword

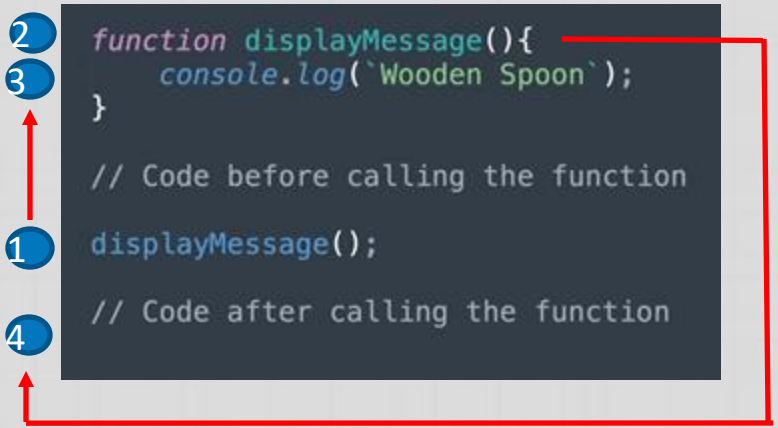


Components of Function

- The function keyword: indicates that the start of the function
- The function name: Descriptive name of the function
- The parenthesis: function/method name is always followed by a set of parenthesis, can be capable of receiving arguments

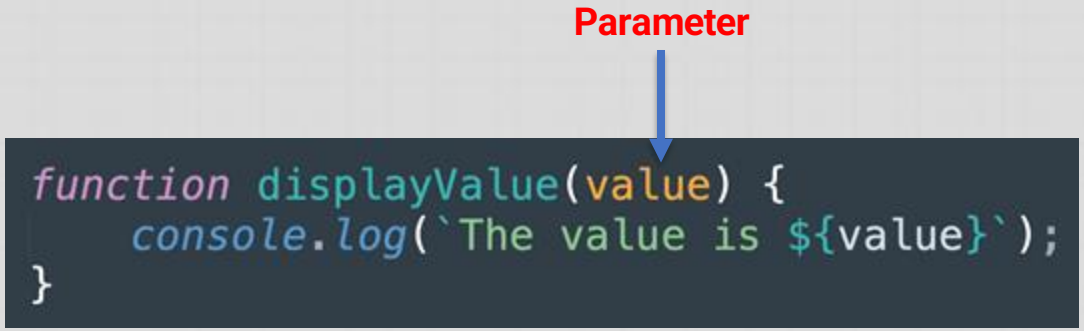
Calling a Function

- When we need to script to perform the task the function does
- The function executes the codes in its scope from top to button
- When it has finished, the code continues to run from the point where it was initially called



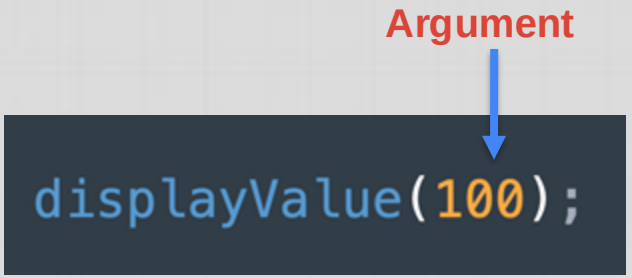
Passing Parameters to Function

- When we declare a function, parameters can be given
- Parameters passed to the function act like variables within the function's scope
- Used for providing additional information the function must have to perform its task



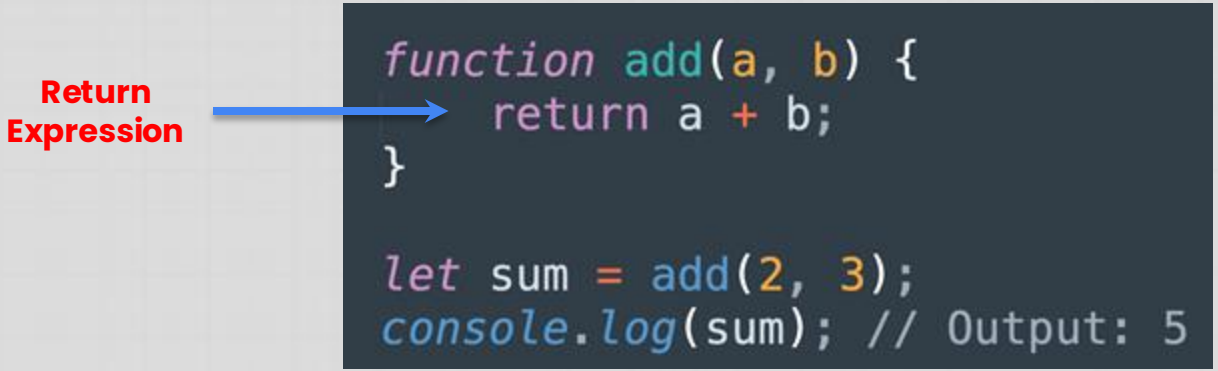
Calling a Function that Needs Information

- Must specify the values the method should use
- Values need to be given in the parentheses that follow the function name
- The values we passed to the method are called arguments
- Arguments can be provided as values or as variables



Return Values From Functions

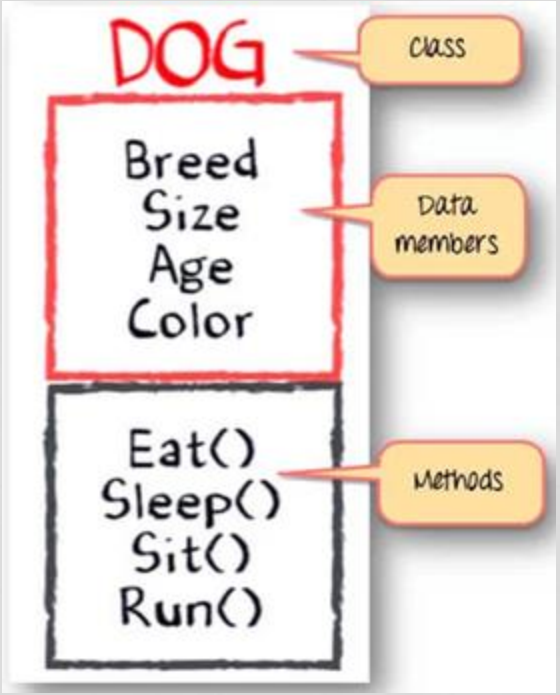
- A function can return a value by using a return statement
- The return statement ends function execution and specifies a value to be returned to the function caller



Class & Object

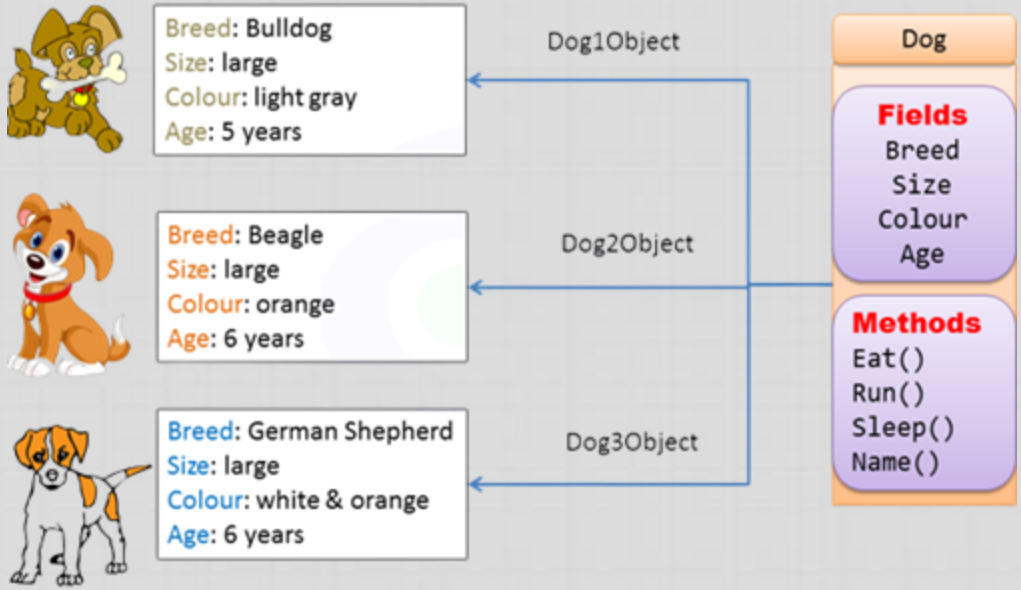
What is A Class?

- Where objects came from
- A blueprint or set of instructions to build a specific type of Object
- No memory allocated for a class



What is An Object?

- An instance of a class
- Multiple objects can be created from a class
- Each object has its own memory
- The data stored in an object are called fields



Writing A Custom Class

Class Name	Dog
Fields (Attributes)	name breed size age color ...
Methods (Actions)	eat() drink() play() ...

keyword

Class Name

```
class Dog {

  constructor(name, breed, age, color) {
    this.name = name;
    this.breed = breed;
    this.age = age;
    this.color = color;
  }

  eat() {
    console.log(`${this.name} is eating.`);
  }

  drink() {
    console.log(`${this.name} is drinking.`);
  }

  play() {
    console.log(`${this.name} is playing.`);
  }

}
```

The Constructor Method

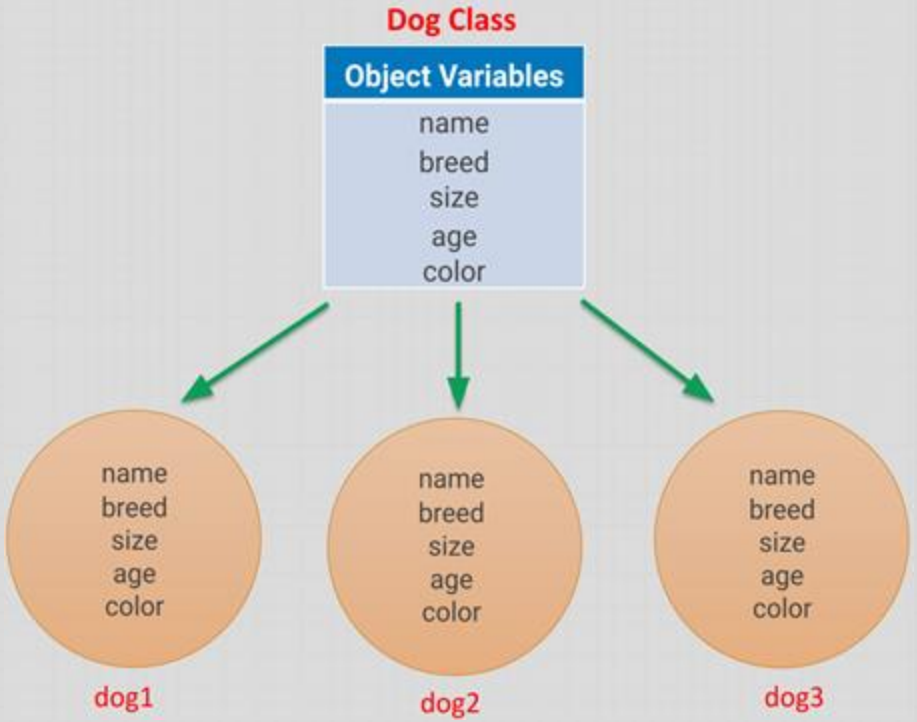
- Built-in constructor method used for defining & initializing the attributes
- Belongs to the object and each object has its own memory
- Gets executed when an object is created from the class

```
class Dog {

  constructor(name, breed, age, color) {
    this.name = name;
    this.breed = breed;
    this.age = age;
    this.color = color;
  }

}
```

Object Variables

- Belongs to the object, each object has a different copy of the instance variable
- Used by the objects to store their data members

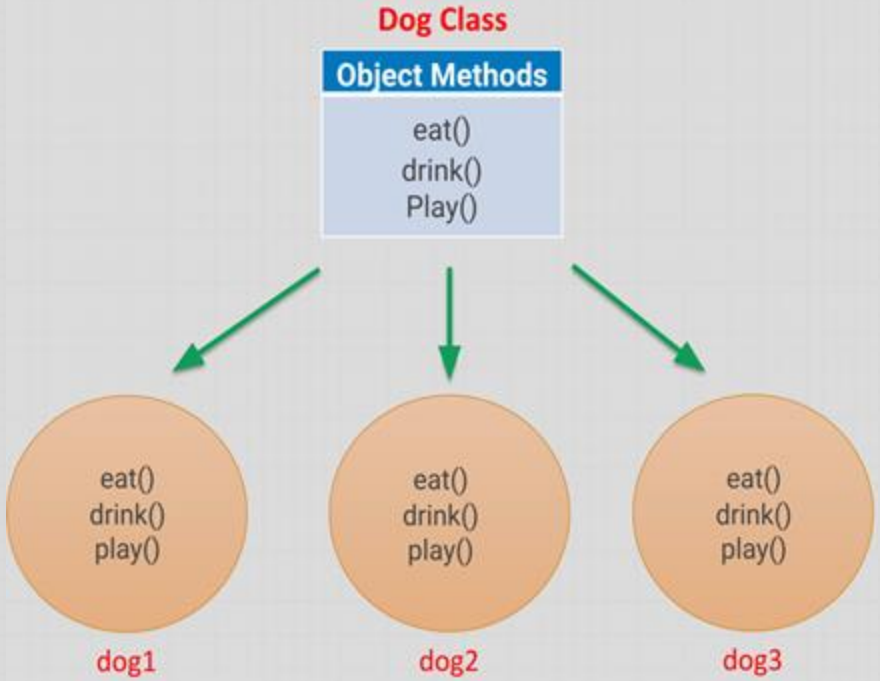
```
constructor(name, breed, age, color) {
  this.name = name;
  this.breed = breed;
  this.age = age;
  this.color = color;
}
```



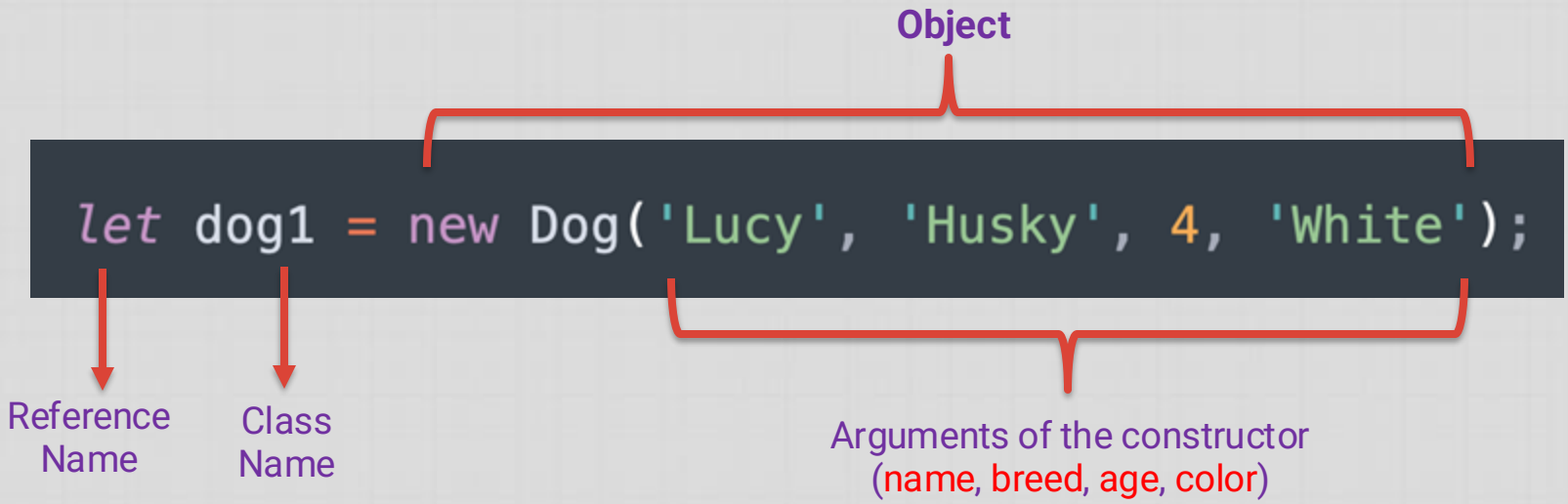
Object Methods

- Objects can share the methods created within the class
- Methods can be called through the object once it's instantiated

```
eat() {
  console.log(`${this.name} is eating.`);
}

drink() {
  console.log(`${this.name} is drinking.`);
}

play() {
  console.log(`${this.name} is playing.`);
}
```



Creating an Object



Accessing an object's data and methods

- An Object's members refer to its data fields and methods. After the object is created its data can be accessed and its methods can be invoked using the dot operator (.)

```
let dog1 = new Dog('Lucy', 'Husky', 4, 'White');

console.log(dog1.name); // Lucy
console.log(dog1.breed); // Husky
console.log(dog1.color); // White

dog1.eat(); // Output: Lucy is eating.
dog1.drink(); // Output: Lucy is drinking.
dog1.play(); // Output: Lucy is playing.
```

Class vs Object

Class	Object
Class is a collection of similar objects	Object is an instance of a class
Class is conceptual (is a template)	Object is real
No memory is allocated for a class	Each object has its own memory
Class can exist without any objects	Objects can not exist without a class

AI Tools In Coding

Applications of AI in Software Development

- **Automated Code Generation**: AI tools which can write codes snippets or entire programs based on given descriptions
- **Code Completion**: AI systems that suggest the next part of the code you are writing
- **Bug Detection and Fixing**: AI systems that identify bugs and suggest fixes

Benefits of using AI tools

- **Increases Productivity**: AI tools can help programmers write code faster and automating repetitive tasks
- **Improved Code Quality:** AI tools can suggest best practices and common patterns to write cleaner and more maintainable codes
- **Error Reduction:** AI can catch potential errors and bugs early in the development process
- **Learning and Skill Enhancement:** AI tools exposes developers to new coding techniques and patterns