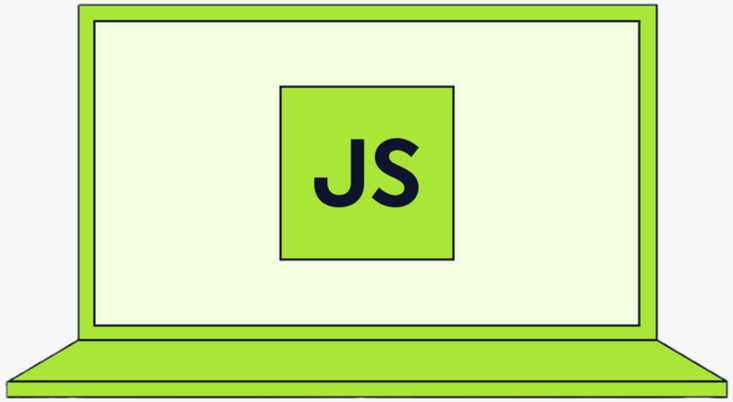
Newton School of Technology

The Complete Javascript Course



@newtonschool

Lecture 20: Functions,Array and Objects

- Team Newton

JS

Table of Contents

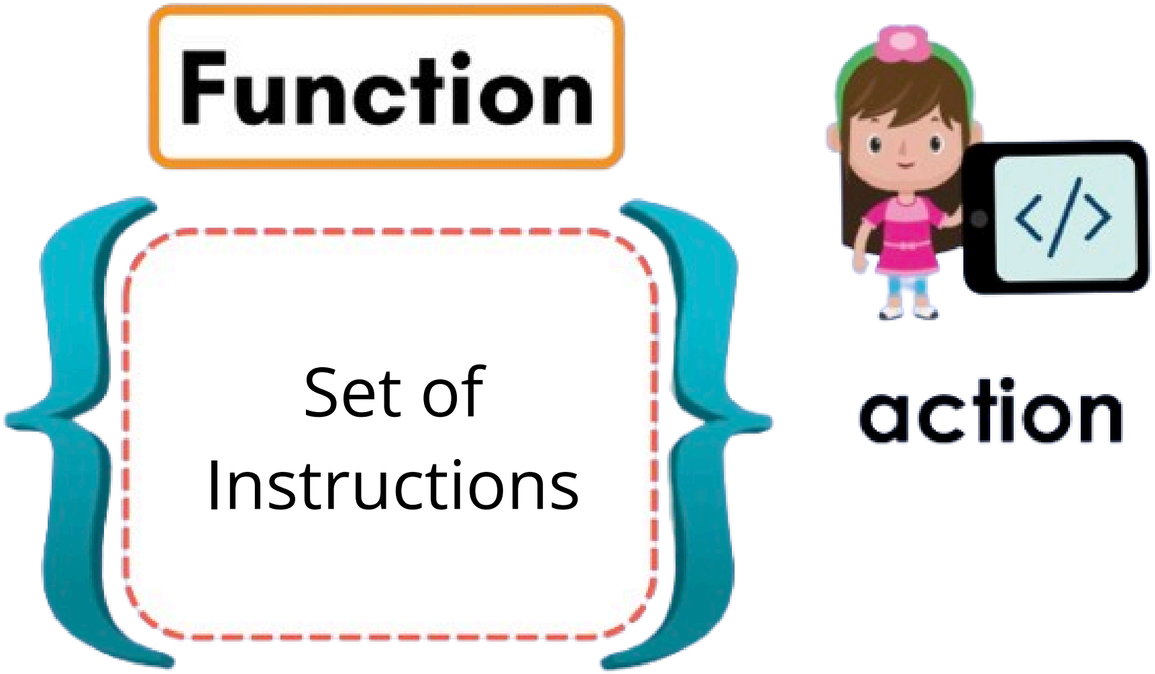
Newton School of Technology

Newton School of Technology

Function in JS

What is a Function?

A function is a block of code made out of a set of steps that result in a single specific action.



What does it mean??
Let’s understand with an analogy

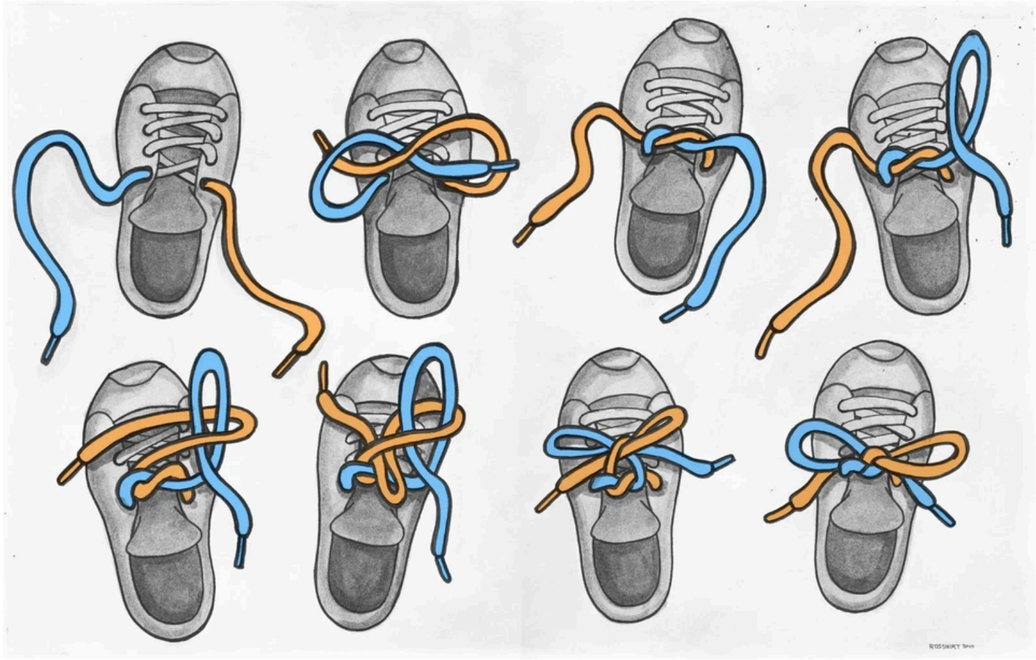
Functions: Like Tying your shoes

Newton School of Technology

A function is like tying your shoes—once you learn how, you can repeat the steps whenever needed without rethinking the process

Functions = *Tying your shoes*

1. Gather laces
2. Knot
3. Loop
4. Swoop
5. Pull tight



Similarly we use functions in our programs

Newton School of Technology

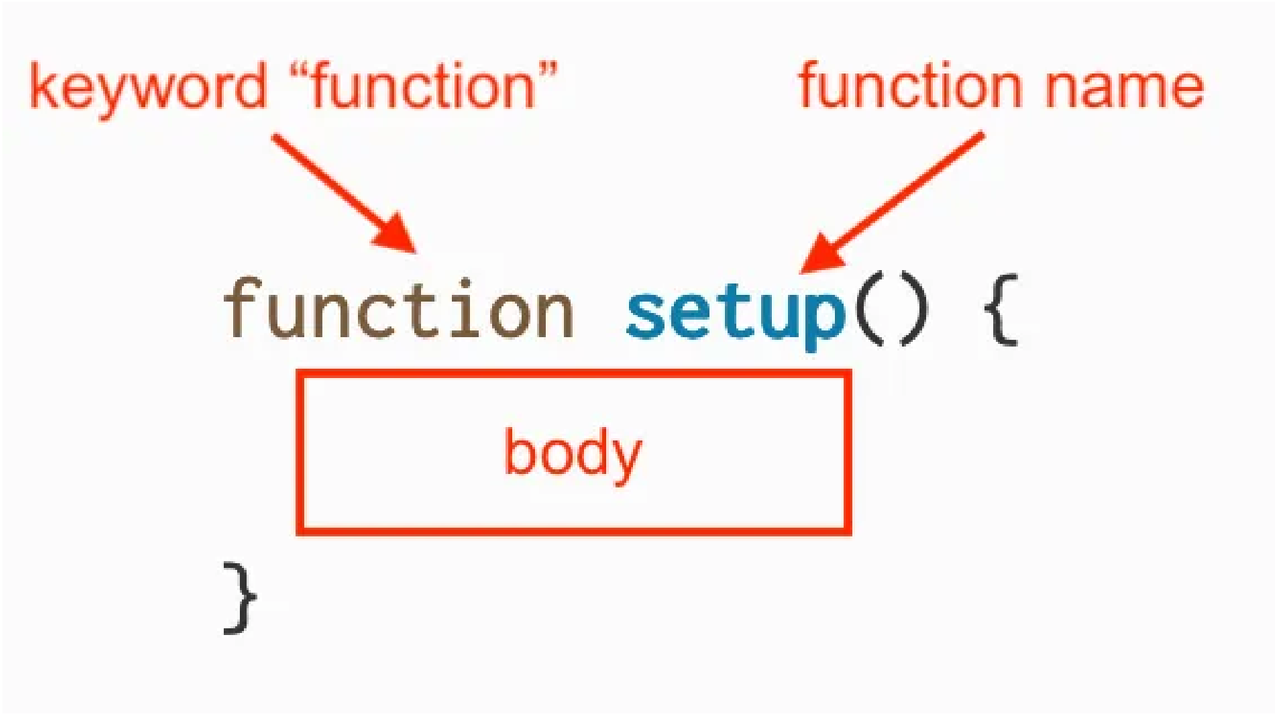
Let us define a tie-shoe function and reuse it multiple times:

```
// Function to tie your shoes
function tieShoes() {
    console.log("Cross the laces.");
    console.log("Make a loop with one lace.");
    console.log("Wrap the other lace around the loop.");
    console.log("Pull through and tighten.");
}

// Reusing the function multiple times
tieShoes(); // First time
tieShoes(); // Second time
```

Naming a function

Newton School of Technology

A programmer can give a function a name and be used again and again.



Newton School of Technology

Challenge
Write a Function to Display Hello World

Were you able to do it??

Newton School of Technology

Let’s start writing:

```
function sayHello() {
    console.log("Hello, World!");
}

sayHello();
// Output: Hello, World!
```

How simple it is!!

Think of a more complex task/function

Newton School of Technology

Your dad wants you to calculate your daily expenses, but you find it difficult to do manually. You are crying for help!!

Your Rescue: Write an add function

Newton School of Technology

Don’t worry, functions are here for rescue.

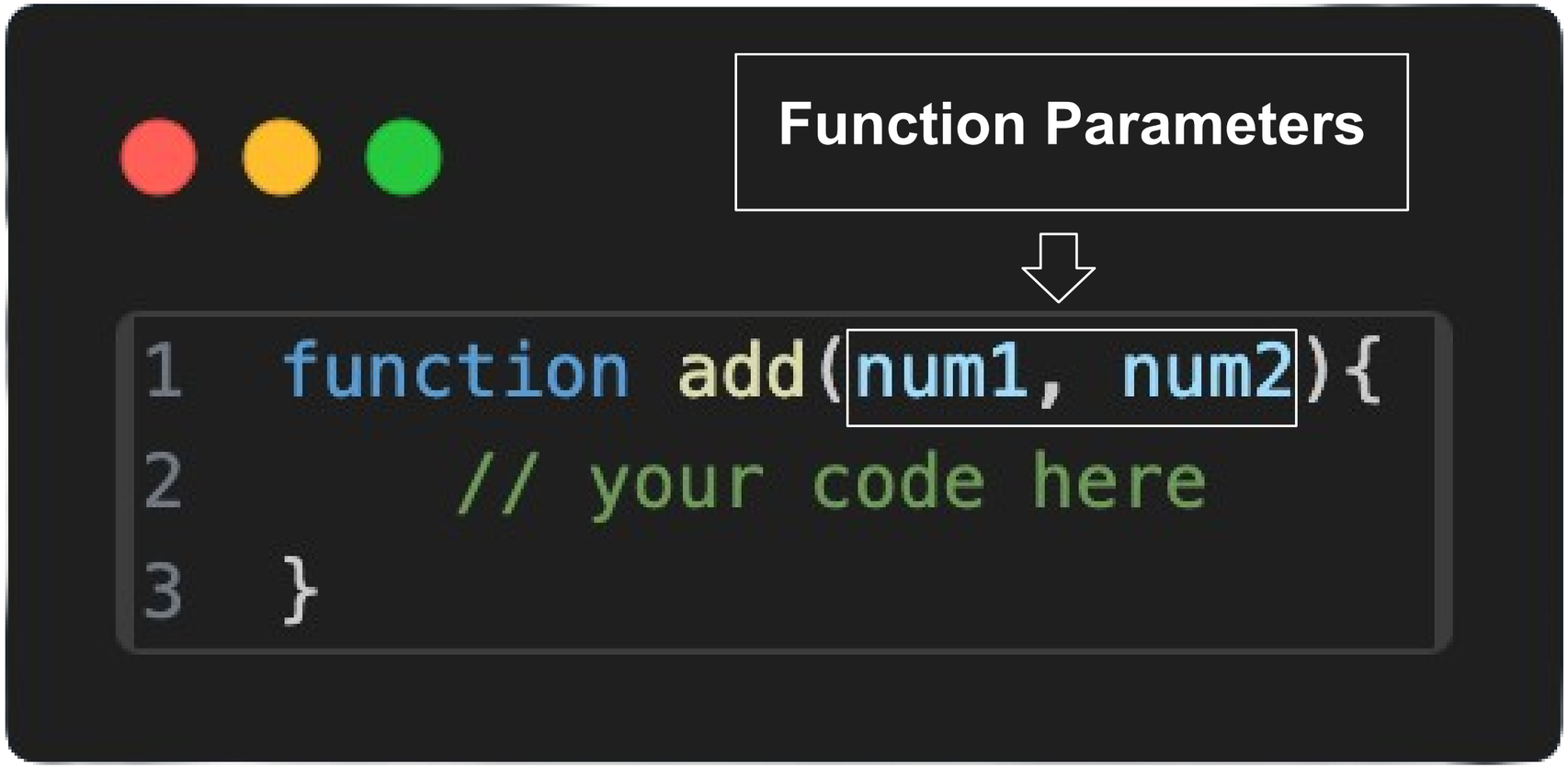
```
function add(){
    // your code here
}
```

But how to tell the function what to add??

Function: parameters

Newton School of Technology

We can define parameters in a function to receive values.



Here “*num1*” and “*num2*” are function parameters.

Function: using parameters to add

Newton School of Technology

Let’s use parameters to calculate the sum. And print it:

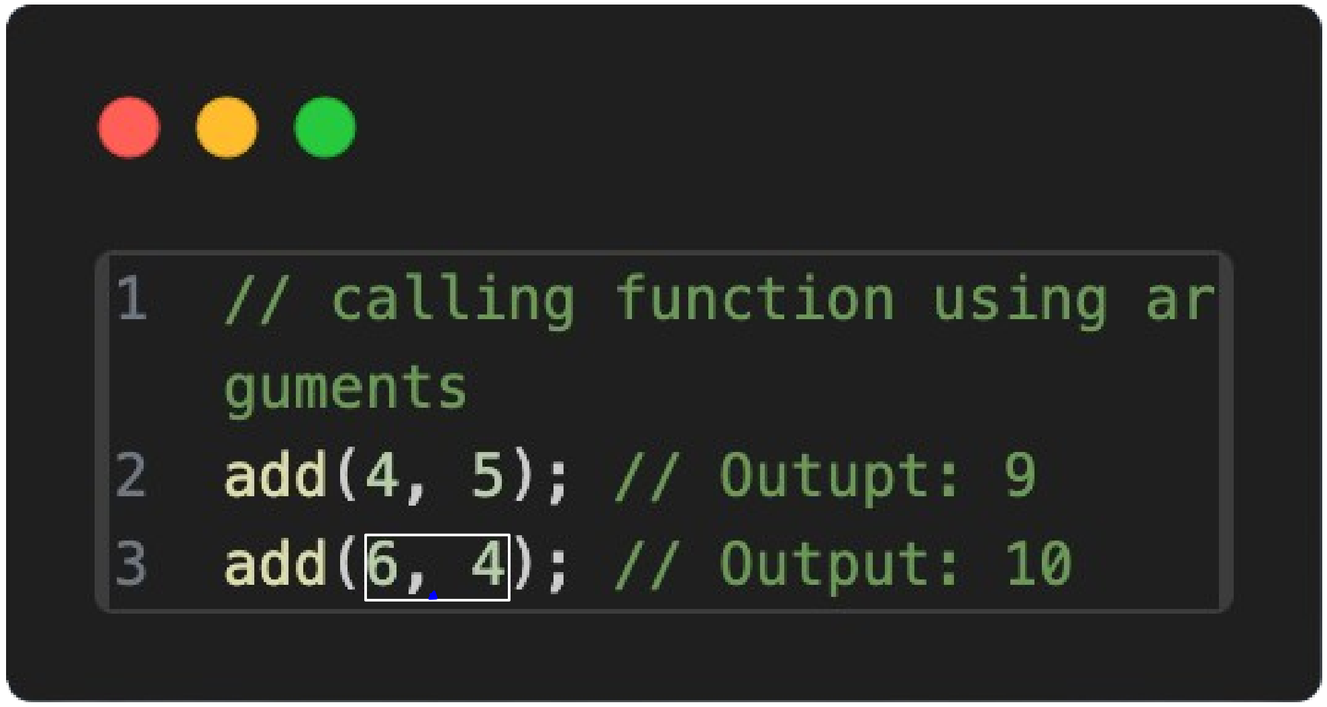
```
1  function add(num1, num2){
2      let result;
3      result = num1 + num2;
4
5      // printing the result
6      console.log(result);
7  }
```

Using parameters to perform addition

Function: calling using arguments

Newton School of Technology

Let’s just call the function by passing values inside it:



```
// calling function using arguments
add(4, 5); // Outupt: 9
add(6, 4); // Output: 10
```

Arguments are the values passed to a function and assigned to its parameters.

Actual values passed to function are called ***arguments***

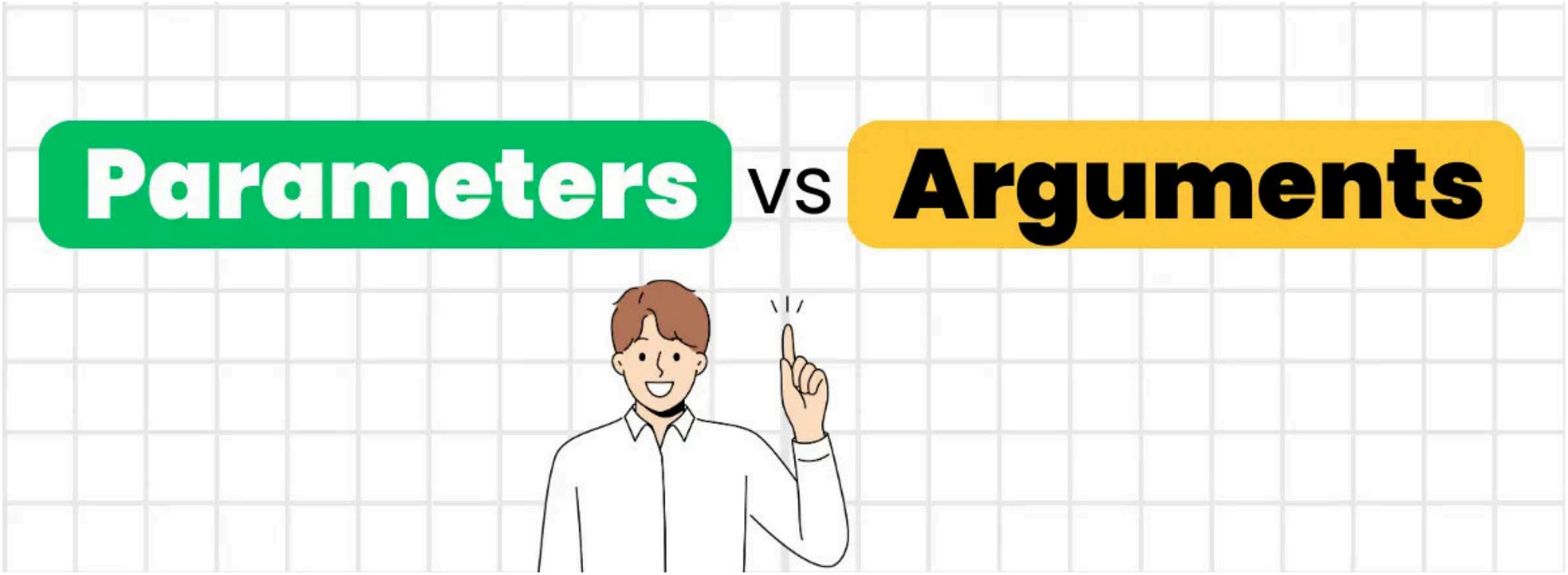
Newton School of Technology

Parameters vs Arguments

Difference: parameters and arguments

Newton School of Technology

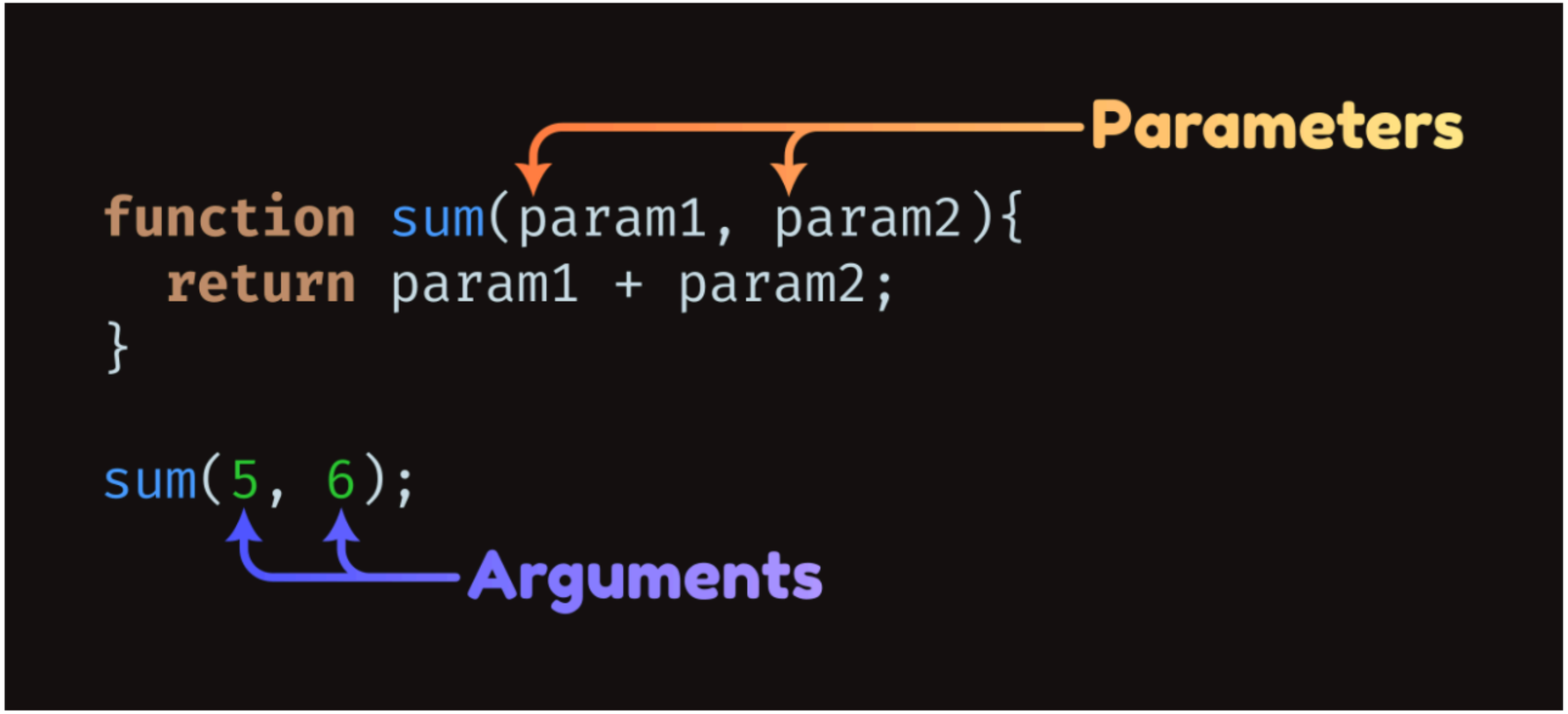
Now, can you tell me the difference between these two??



Difference: parameters and arguments

Newton School of Technology

Parameters are placeholders in a function declaration, while arguments are the actual values passed during a function call.



Newton School of Technology

Function Return Values

Function: return values

Newton School of Technology

You expect someone to return something whenever you give them a task; this could be the task's status or final result, among other things.

Return Values

Function: return value syntax

Newton School of Technology

Similarly, functions do return values on completion.

```
1   function findSquare(num) {
2       let result = num * num;
3       return result;
4   }
5
6   let square = findSquare(3);
```

We want to return the result value to where the function was called from.

Function: return values with Example

Newton School of Technology

Displaying returned values by calling the function and storing the result in a variable.

```
1  function add(num1, num2){
2      let result;
3      result = num1 + num2;
4
5      // returning the result
6      return result;
7  }
8
9  let sum = add(2, 3);
10 console.log(sum); // Output: 5
```

Returning values is preferred over displaying the result inside the function.

Function: JS vs Python

Newton School of Technology

Operation	JavaScript	Python
Define a function	function greet() {}	def greet():
Define with parameters	function greet(name) {}	def greet(name):
Return a value	return "Hello";	return "Hello"
Call a function	greet();	greet()

In Class Quiz

Newton School of Technology

Arrays

Array: Lists in javascript

Newton School of Technology

Lists in javascript are called “Array”.

Definition: An array in JavaScript is a special object used to store multiple values in a single variable.

Planning a party

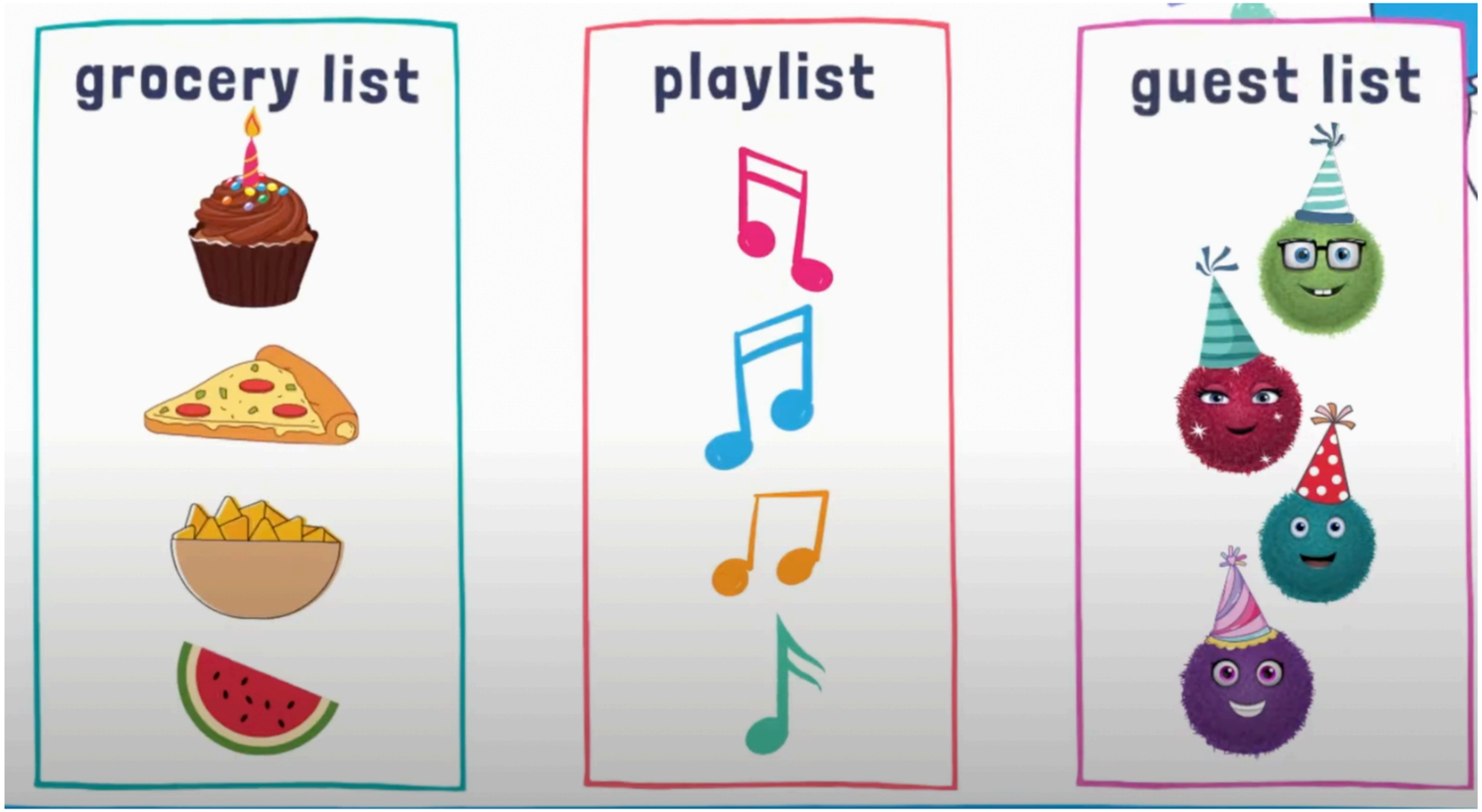
Newton School of Technology

Imagine you are organizing a party, for that you need to purchase lot of items.

You will need, cake, music player, food and decorative items. So you might want to make a list out of it!!

Planning a party: making lists

Let’s make lists, separate list for separate group of items:-

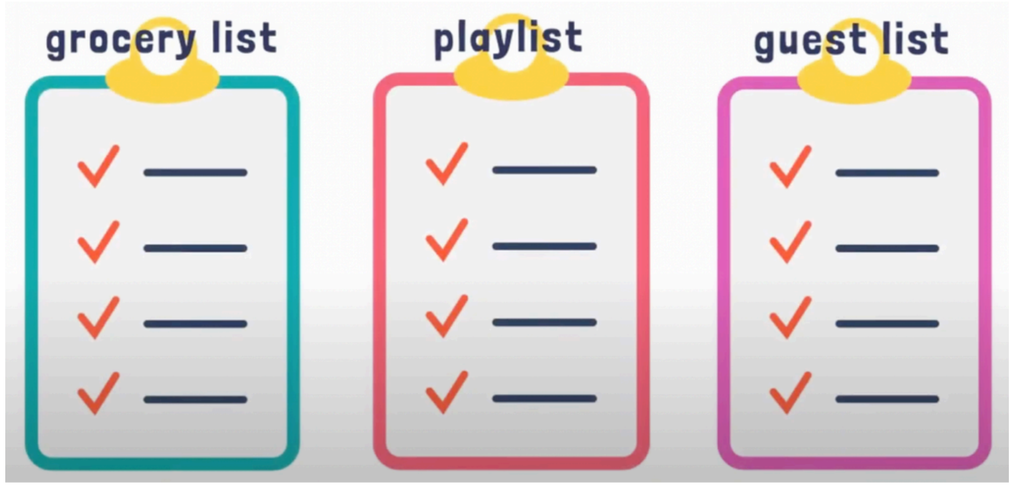


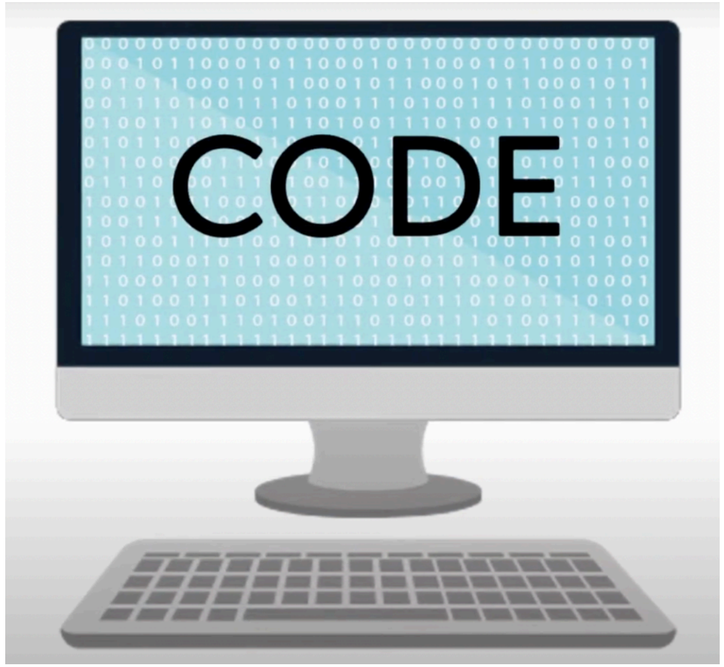
Lists help us to keep our tasks organized!!!

Planning a party: making lists

Newton School of Technology

Just like lists help you keep your tasks organized. Lists are very helpful in programming too:-

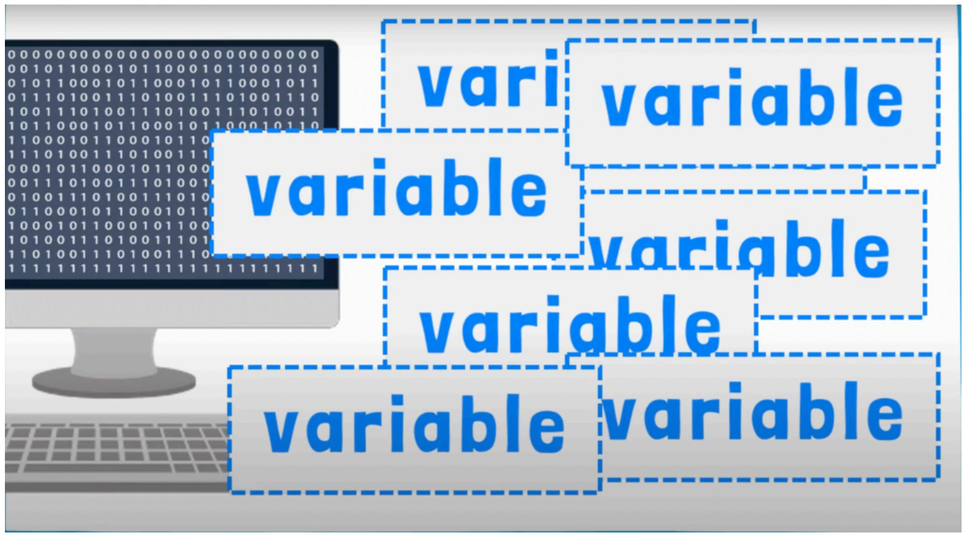


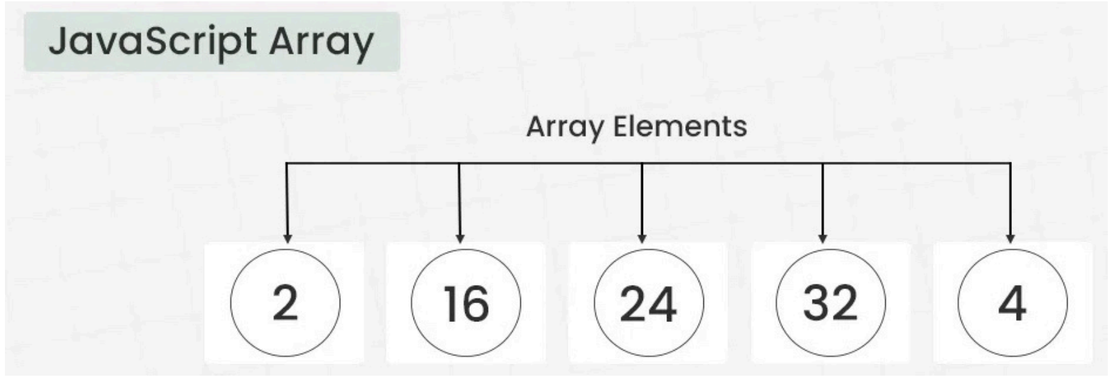


Why do we need array?

Newton School of Technology

Variables store data, but managing many of them is hard. Arrays simplify this by organizing multiple values in one place.



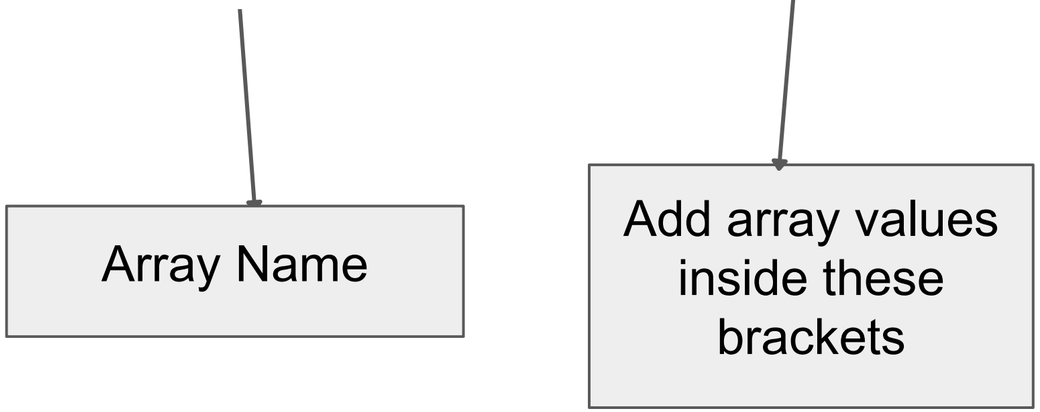


Array: Definition and Syntax

Newton School of Technology

An array in JavaScript is a special object used to store multiple values in a single variable.

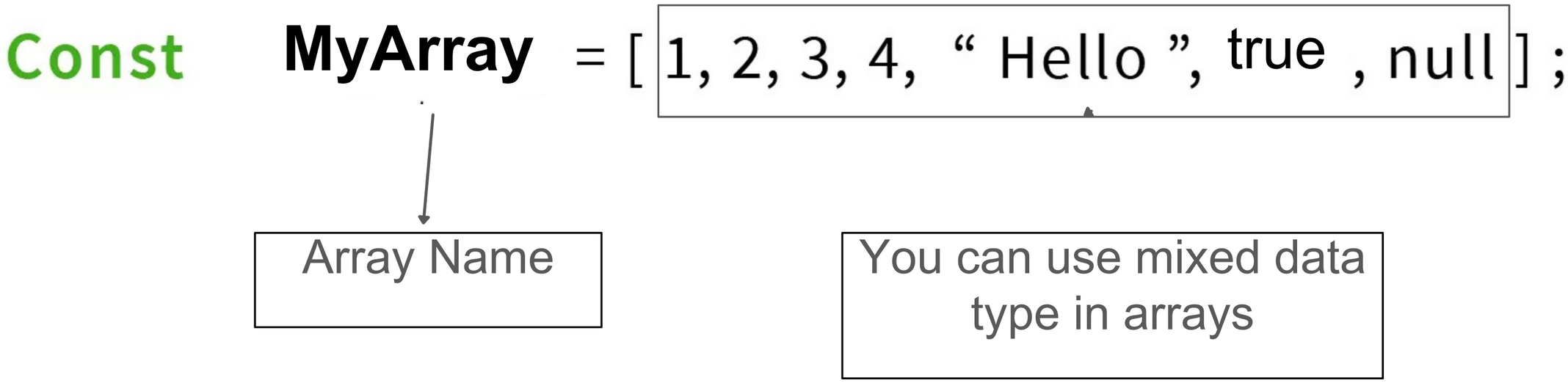
```
const MyArray = [ ];
```



Array: Storing values in arrays

Newton School of Technology

We can declare array as well as store values at the same time.



Array: Accessing the array values

Newton School of Technology

We can access values stored in array by using indexes.

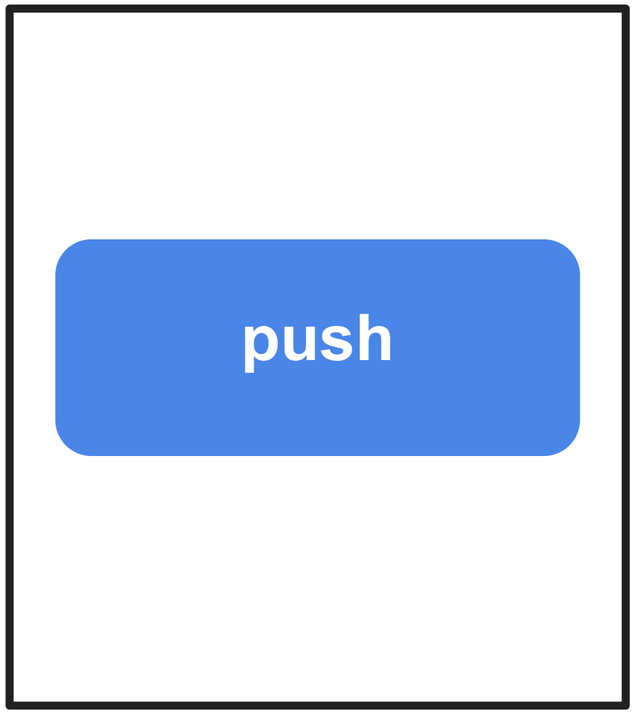
```
// Declaring and assigning array
const MyArray = [1, 2, 3, 4, "Hello", true , null];

// Printing array values
console.log(MyArray);
// Output: [1, 2, 3, 4, "Hello", true , null];
console.log(MyArray[0]); // Output: 1
console.log(MyArray[1]); // Output: 2
console.log(MyArray[4]); // Output: Hello
console.log(MyArray[5]); // Output: True
```

If we just type *MyArray* in console, entire array would get printed

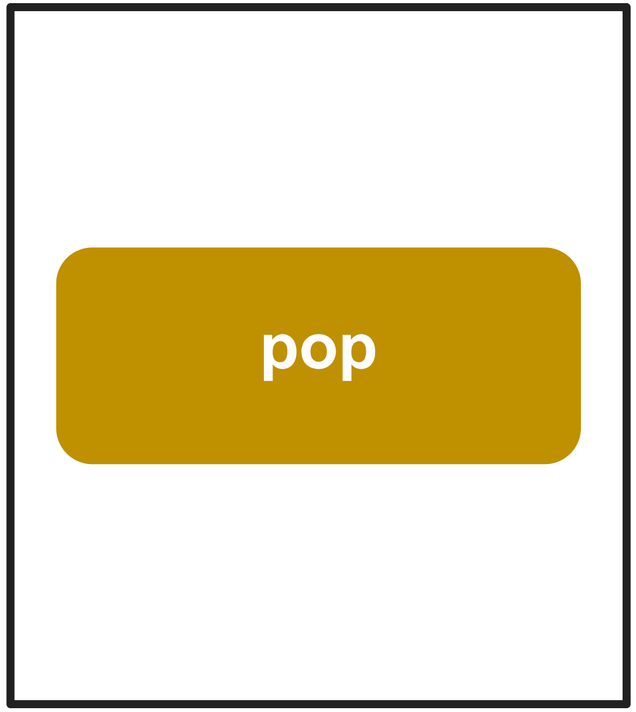
Common Javascript Methods

Newton School of Technology

Here are some commonly used array methods:



Add Items



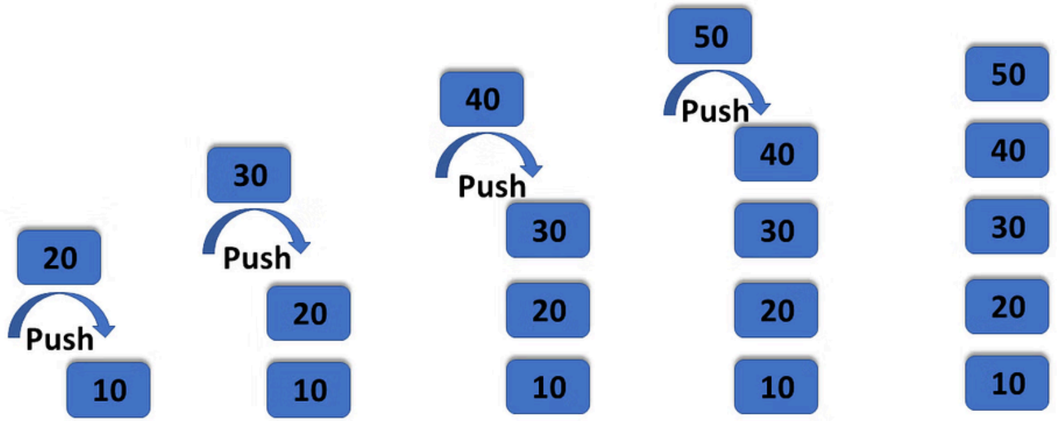
Remove items

Add Items: push

Newton School of Technology

Adds one or more items to the **end** of the array. Initially there was 10, however we could have started with empty array

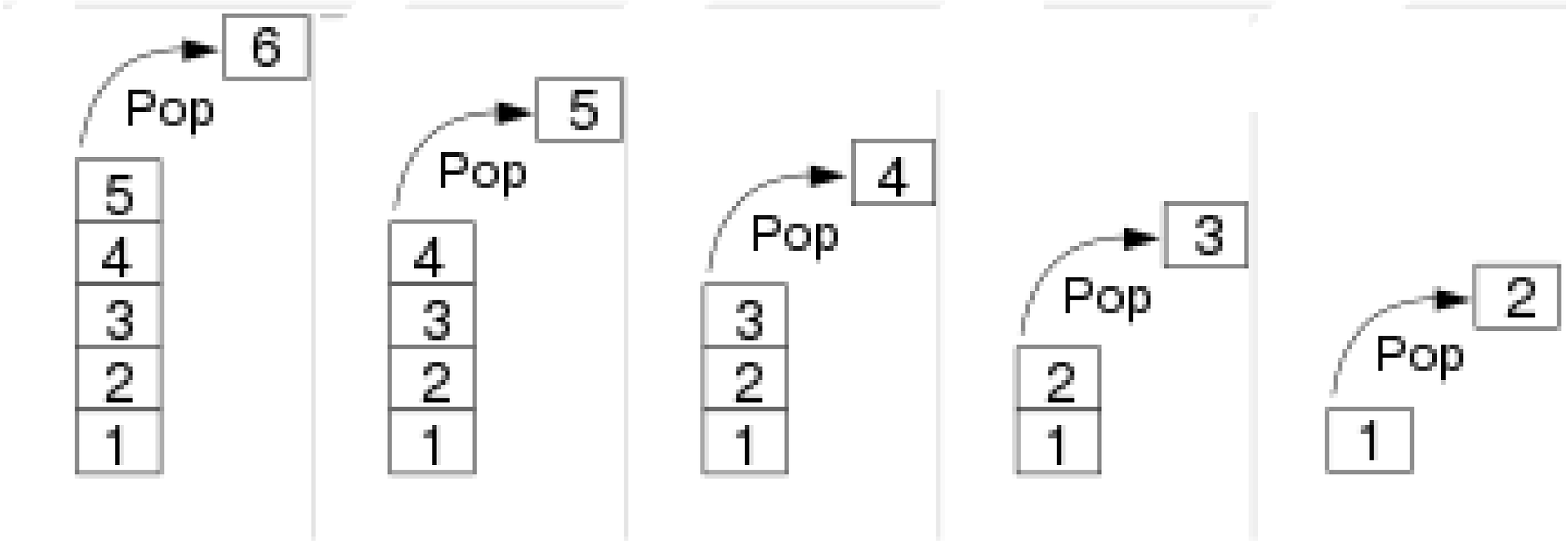
```
// Declaring an empty array
let numbers = [10];

// Adding items one by one
numbers.push(20);  // [20]
numbers.push(30);  // [20, 30]
numbers.push(40);  // [20, 30, 40]
numbers.push(50);  // [20, 30, 40, 50]

console.log(numbers); // [20, 30, 40, 50]

// Adding two items at once
numbers.push(60, 70);

console.log(numbers); // [20, 30, 40, 50, 60, 70]
```



Remove Items: pop

Newton School of Technology

Removes the **last** item from an array.

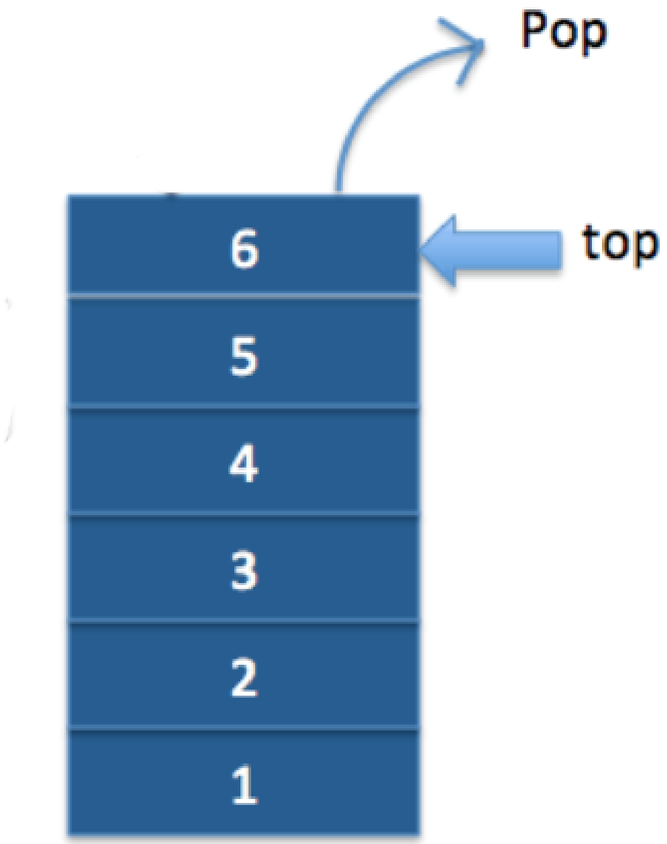


Remove Items: pop

Newton School of Technology

Let’s write javascript code for it:-

```javascript
// Initial array with 5 items
let numbers = [1, 2, 3, 4, 5];

// Using pop to remove all items
numbers.pop();  // [1, 2, 3, 4]
numbers.pop();  // [1, 2, 3]
numbers.pop();  // [1, 2]
numbers.pop();  // [1]
numbers.pop();  // []

console.log(numbers); // []
```



Arrays in Js vs Python

Newton School of Technology

Operation	JavaScript Code	Python Code
Declare a list/array	const/let nums = [ ];	nums = [ ]
Add an element to end	nums.push(4);	nums.append(4)
Remove last element	nums.pop();	nums.pop()
Get length of list/array	nums.length	len(nums)

In Class Quiz

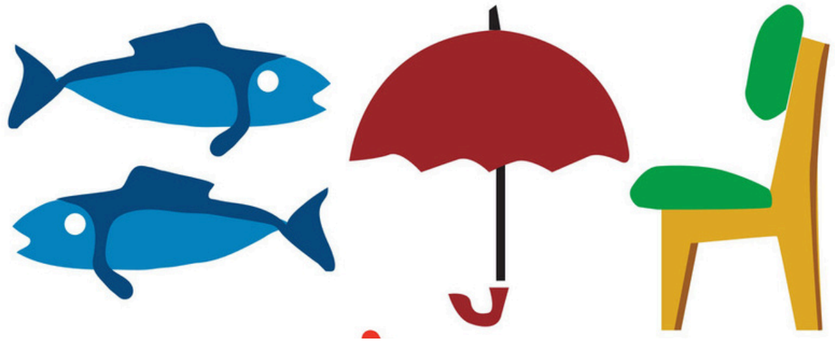
Newton School of Technology

Objects in Javascript

Objects in real world

Newton School of Technology

In the real world, **objects** are entities or things that have distinct characteristics and behaviors. Objects can be described by their **properties** (attributes) and can perform **actions** (methods).



Each of these objects has some physical properties as well as some functionality. Can you name a few for each of them?

Objects in real world: Example

Newton School of Technology

Let’s have a look at this car. It has certain properties and some functions/methods it can perform.

Properties:-

1. color: red
2. make: Toyota
3. year: 2023

Methods/Functions:-

1. start
2. drive

Objects in Programming

Newton School of Technology

They are collections of related data and functionality grouped together, often mirroring real-world entities. Let’s write an object from the car from the previous

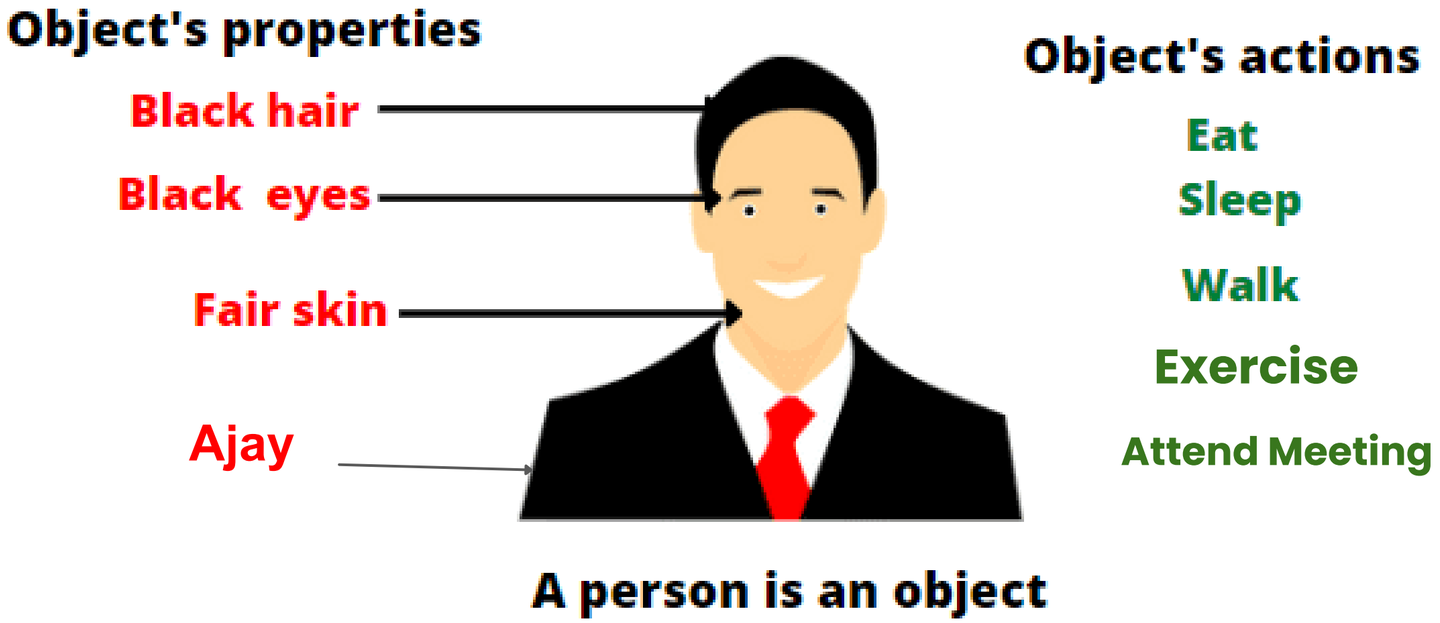
```
const car = {
    // Properties
    color: "red",
    make: "Toyota"
    year: 2023

    // Methods
    start: function() {
      console.log("The car has started.");
    },
    drive: function() {
      console.log("The car is driving.");
    }
};
```

In JavaScript, methods are also considered properties of an object.

Person as an Object

Newton School of Technology

Similarly, we can create an object to represent a person. Imagine Ajay, a busy professional juggling multiple tasks in a day. Alex has distinct traits like a name, black hair, fair skin, etc along with several actions and functions it can perform.



Can you transform this description into an object? Function might only contain console.log() for now?

The Persona of Ajay: A Living Object

Newton School of Technology

Here your implementation might differ; I have only taken a few actions as object methods.

```
const ajay = {
    // Properties
    name: "Ajay",
    hairColor: "black",
    skinTone: "fair",

    // Methods
    attendMeeting: function() {
        console.log(`${ajay.name} is attending a meeting.`);
    },
    exercise: function() {
        console.log(`${ajay.name} is cycling to stay fit.`);
    }
};

// Example Usage
ajay.attendMeeting();
console.log(ajay.name);
```

Now you should feel confident writing an object by yourself.

Object Creation Workshop

Workshop: Question

Newton School of Technology

Create an object to represent a smartphone with the following properties: brand, model, price, and features (as an array). And write some methods like messaging, calling, etc.

Be confident, and you will be able to write in one go.

Newton School of Technology

Objects vs Other Data Types

Newton School of Technology

Objects vs Primitives: A Unified Concept

Objects are derived from primitive types but differ in storing and managing data, grouping related data and behaviors for more flexibility.

```
const ajay = {
    // Primitive Data Types as Properties
    name: "Ajay",           // String (Primitive)
    age: 25,                // Number (Primitive)
    isEmployed: true,       // Boolean (Primitive)
    address: null,          // Null (Primitive)
    gender: undefined,      // Undefined (Primitive)

    // Methods related to person
    greet: function(){
        console.log(`Hello, my name is ${ajay.name}`);
    },
    updateEmploymentStatus: function(status){
        this.isEmployed = status;
        console.log(`Employment: ${ajay.isEmployed}`);
    }
}
```

Notice how various data types are included inside the object named 'ajay'.

How objects are different from primitives

Newton School of Technology

They differ in the following ways:-

Aspect	Primitives	Objects
Structure	Simple, single values like numbers or text.	Complex things that store many related values or actions.
Storage	Small and fast.	Bigger and slower.
Access	Accessed by identifier storing value.	Accessed via properties or methods (e.g., object.property).
Behavior	No inherent behavior; only stores data.	Can include methods for processing data.
Example	var car = “Toyota”	Visit slide 6 to see the example.

We will discuss access methods in later slides

Newton School of Technology

Objects as Key-Value Pairs

Objects as key-value pairs

Newton School of Technology

Objects are essentially collections of related information represented as **key-value pairs**. Values stored are accessible via that key.

key	value
name	Ajay
age	30
occupation	Engineer

```
let person = {
    name: "Ajay",
    age: 30,
    occupation: "Engineer"
};

console.log(person.name); // Output: "Ajay"
console.log(person.age);  // Output: 30
```

Objects as key-value pairs

Newton School of Technology

Methods also need to be accessed via its key as it is also considered as property.

```
let person = {
    name: "Ajay",
    greet: function() {
        console.log("Hello, ", + person.name);
    }
};

person.greet(); // Output: Hello, Ajay
```

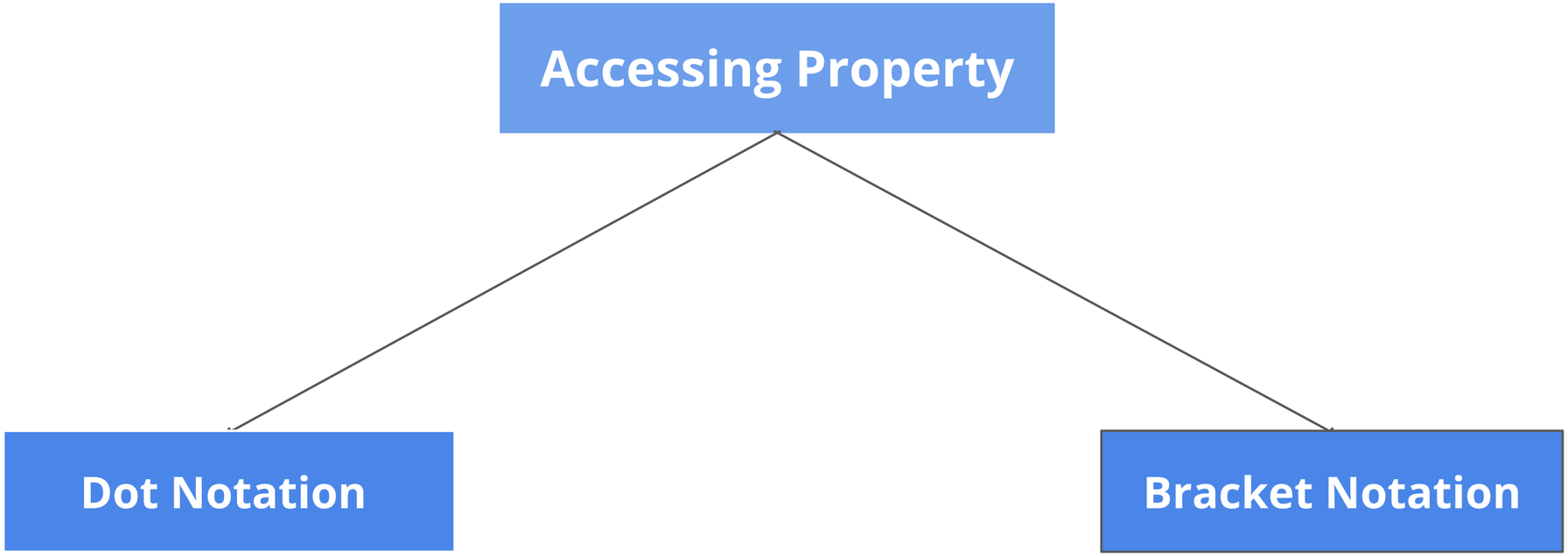
Newton School of Technology

Accessing Properties

Different ways to access properties

Newton School of Technology

There are two main ways to access the properties of an object



Accessing Property: Dot Notation

Newton School of Technology

In JavaScript, dot notation is the simplest and most common way to access the properties of an object. You use a dot (.) followed by the property name.

```
object.property
```

- object is the object you're working with.
- property is the name of the property you want to access.

Dot Notation: example

Newton School of Technology

Let's revisit the car object with the properties color, make, and year, and then access those properties using dot notation.

```
let car = {
    color: "Red",
    make: "Toyota",
    year: 2023
};

console.log(car.color);  // Output: Red
console.log(car.make);   // Output: Toyota
console.log(car.year);   // Output: 2023
```

Accessing properties using dot notation

Dot Notation: example

Newton School of Technology

Accessing the methods are no different.

```
let car = {
    color: "Red",
    make: "Toyota",
    year: 2023,
    displayInfo: function() {
        console.log(`This is a ${car.year} ${car.make} ${car.color} car.`);
    }
};

// Accessing and calling the method using dot notation
car.displayInfo();
// Output: this is a 2023 Toyota Red car.
```

Accessing Property: Bracket Notation

Newton School of Technology

Bracket notation is used to access both properties and methods of an object, especially when the property names are dynamic, contain spaces, or are stored in variables.

```
object[property]
```

Use single or double inverted commas in case property is a string.

Bracket Notation: example

Newton School of Technology

Let's reuse the car example, object creation part remains the same but we will use brackets instead of dot to access the object property.

```
// Accessing properties using bracket notation
console.log(car["color"]);  // Output: Red
console.log(car["make"]);   // Output: Toyota
console.log(car["year"]);   // Output: 2023

// Accessing and calling the method using bracket notation
car["displayInfo"]();
// Output: This is a 2023 Toyota Red car.
```

This notation is very useful in case property name is dynamic i.e. changes frequently maybe based on some condition.

Bracket Notation: example

Newton School of Technology

Accessing the methods are no different.

```
// Let's say the property name is dynamic, based on a condition
let propertyName = "make";
// This can change based on some condition

// Accessing the property dynamically using bracket notation
console.log(car[propertyName]);
// Output: Toyota

// Changing the condition
propertyName = "year";

// Accessing another property based on the updated condition
console.log(car[propertyName]);  // Output: 2023
```

Here we are using propertyName to dynamically access different properties of the car object.

Newton School of Technology

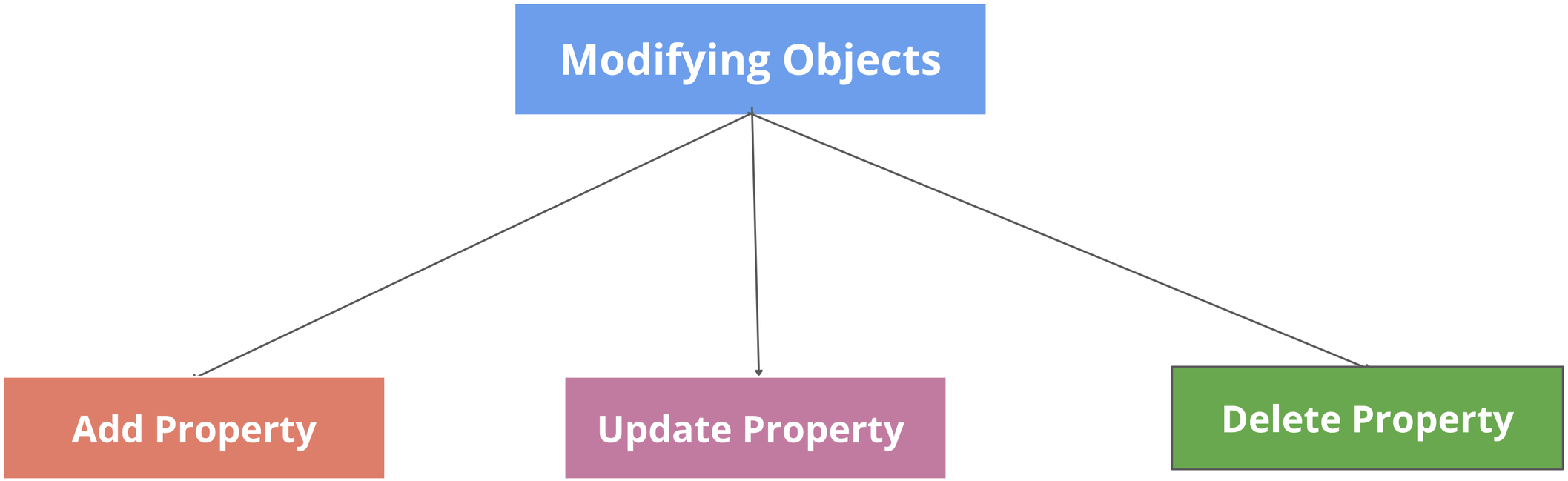
Modifying Objects

Why Modify Objects?

Newton School of Technology

In JavaScript, objects often need to change to reflect new information.

We'll explore three ways to modify objects:



How to add properties

Newton School of Technology

You can add new properties to an object anytime, expanding it dynamically. For example, if you forget to define the car color, you can add it later like this:

```
let car = { make: "Toyota", year: 2023 };

car.color = "Red";  // Adding a new property

console.log(car.color);  // Output: Red
```

We added a new property using the dot operator and assigned it the value "Red".

Adding a method property

Newton School of Technology

Here we have added a method property using dot operator.

```
let car = {
    make: "Toyota",
    year: 2023
};

// Adding a method using dot notation
car.startEngine = function() {
    console.log("The engine has started.");
};

// Calling the method
car.startEngine();
// Output: The engine has started.
```

Try adding one more method property by yourself.

How to Update Properties

Newton School of Technology

You can modify the value of an existing property just the way you create a new property.

```
let car = {
    make: "Toyota",
    year: 2023,
    color: "Red"
};

car.color = "Blue";
// Updating an existing property

console.log(car.color);
// Output: Blue
```

Earlier color was ‘Red’, we have later changed the value to ‘Blue’ by using assignment operator.

How to Update Properties

Newton School of Technology

In the similar fashion we can update method properties too.

```
let car = {
    make: "Toyota",
    year: 2023,
    startEngine: function() {
      console.log("The engine has started.");
    }
};

// Updating the startEngine method
car.startEngine = function() {
    console.log("The engine starts with a roar!");
};

// Calling the updated method
car.startEngine();
// Output: The engine starts with a roar!
```

Sounds simple right??

How to Delete Properties

Newton School of Technology

You can remove properties from objects using the delete keyword.

```
let car = {
    make: "Toyota",
    year: 2023,
    color: "Red"
};

delete car.color;
// Deleting a property

console.log(car.color);
// Output: undefined
```

We got undefined because the property was deleted.

Caution

- Deleting a property makes it completely unavailable in the object.
- The deleted property cannot be recovered, but the object itself remains intact.

How to Delete Properties

Newton School of Technology

Methods can be deleted from an object just like properties using the delete operator.

```
let car = {
    make: "Toyota",
    year: 2023,
    startEngine: function() {
      console.log("The engine has started.");
    }
};

// Deleting the startEngine method
delete car.startEngine;

// Trying to call the deleted method
console.log(car.startEngine);
// Output: undefined
```

That’s it!

In Class Quiz

Newton School of Technology

Thanks for watching!