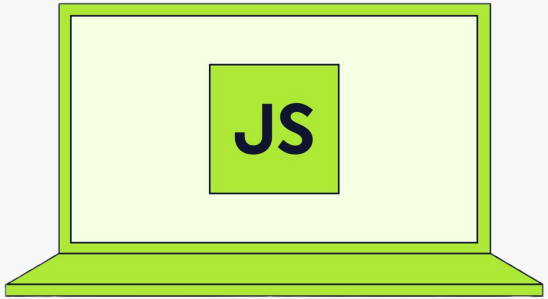
Newton School of Technology

The Complete Javascript Course



@newtonschool

Lecture 2: Data Types and Operators

-Vishal Sharma

JS

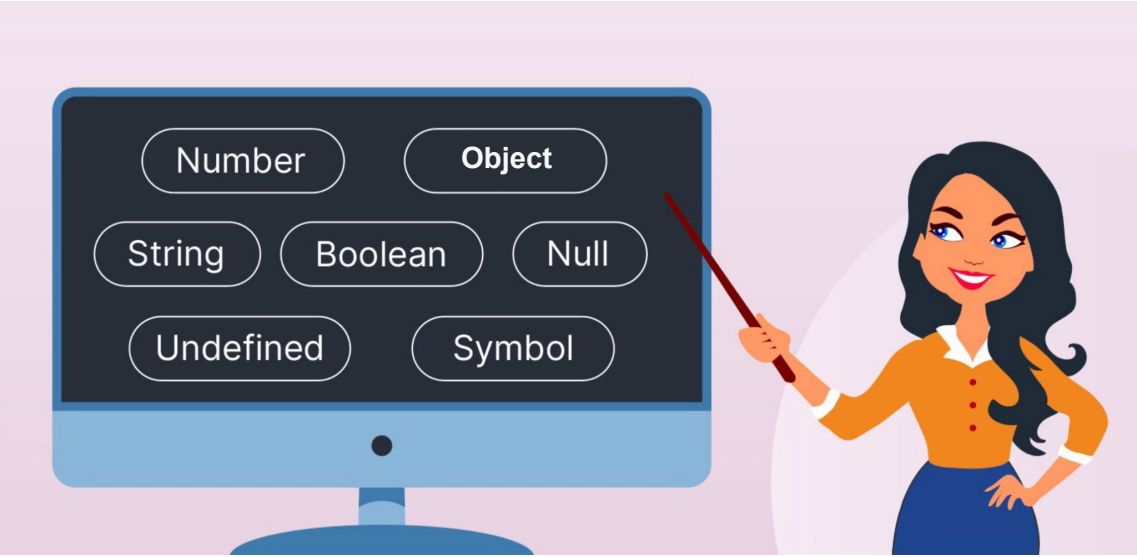
Table of Contents

Newton School of Technology

What are Data Types in javascript?

Newton School of Technology

In JavaScript, **data types** are the classification of data values that tell the type of data being dealt with.



Data types indicate whether data is a number, string, boolean, etc.

Don't worry if you're unfamiliar with them yet.

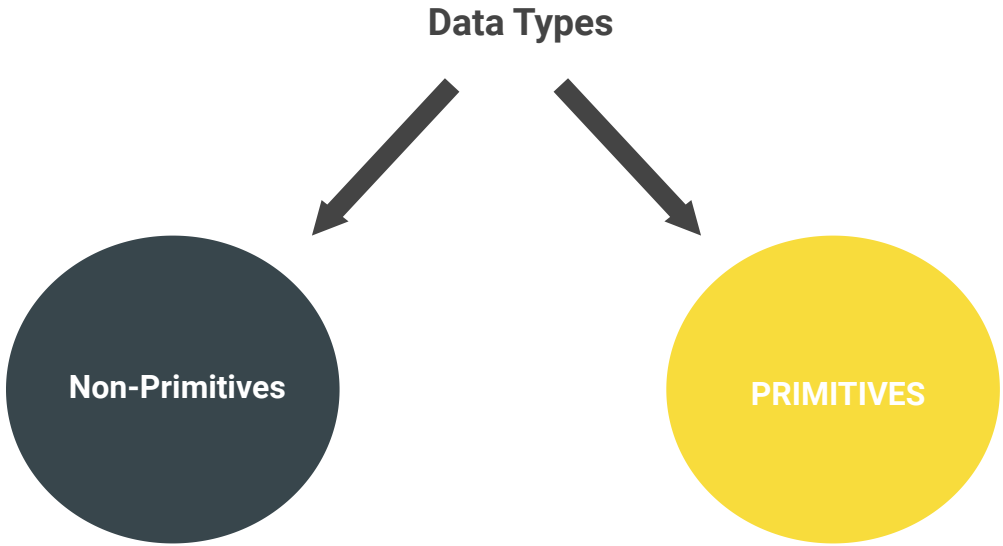
Newton School of Technology

Primitive and Non-Primitive Data Types

Categories of Data Types

Newton School of Technology

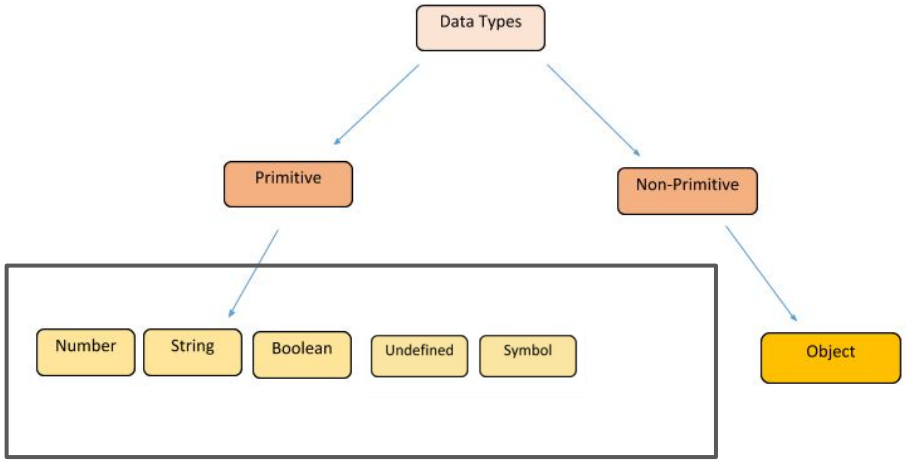
In JavaScript, data types are primitives (simple, immutable) and non-primitives (complex, mutable, and multi-valued).



Let’s start with primitives

Newton School of Technology

Primitive data types in JavaScript are simple, immutable values like numbers, strings, booleans, symbols.



Declaring Primitive Data types

Newton School of Technology

Primitive data types in JavaScript are simple, immutable values like numbers, strings, booleans, undefined, null etc. Let's have a look:-

```
// Primitive Data Types in JavaScript

// Number
let age = 25; // Integer
console.log("Age:", age); // Output: Age: 25

// String
let firstName = "John"; // Sequence of characters
console.log("First Name:", firstName);
// Output: First Name: John

// Boolean
let isActive = true; // Logical value
console.log("Is Active:", isActive);
// Output: Is Active: true
```

In JavaScript, the data type of a variable is implicitly determined by the value assigned to it, such as a `Number` for numbers, a `String` for text, and a `Boolean` for true/false values

Declaring Primitive Data types

Newton School of Technology

Primitive data types in JavaScript are simple, immutable values like numbers, strings, booleans, symbols.

```
1  // Primitive Data Types in JavaScript
2
3  // Undefined
4  let userLocation;
   // Variable declared but not assigned
5  console.log("User Location:", userLocation);
   // Output: User Location: undefined
6
7  // Null
8  let car = null; // Intentional absence of a value
9  console.log("Car:", car); // Output: Car: null
```

If a variable has no value, it's automatically undefined. To intentionally keep it empty, assign it null.

Sounds overwhelming! Don’t worry

Newton School of Technology

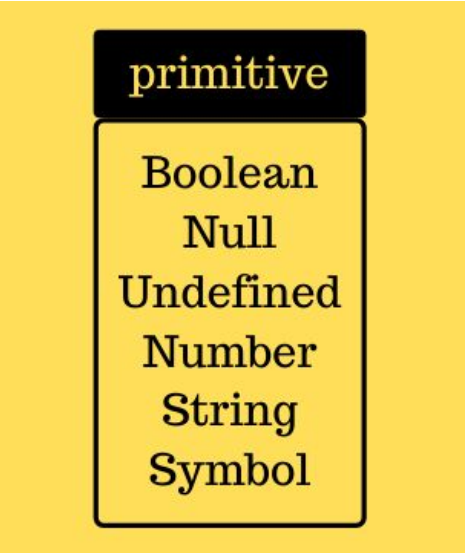
We all find it difficult when we start but it gets easier as we go along

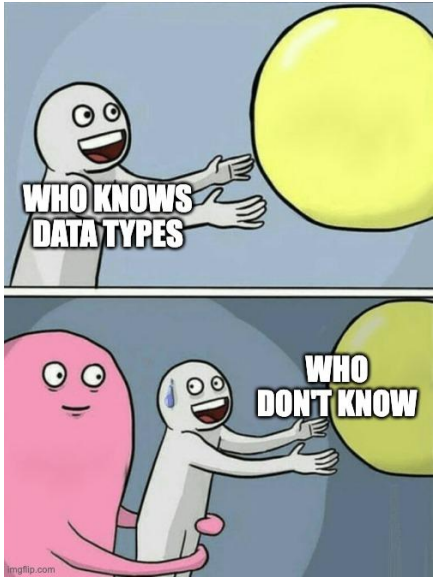
Ask questions when you're unsure, code when things feel confusing, and remember—the more you code, the better you'll understand!

Understanding Data types

Newton School of Technology

Today, we begin our journey through data types, starting with the most common and progressing to the more complex and powerful.

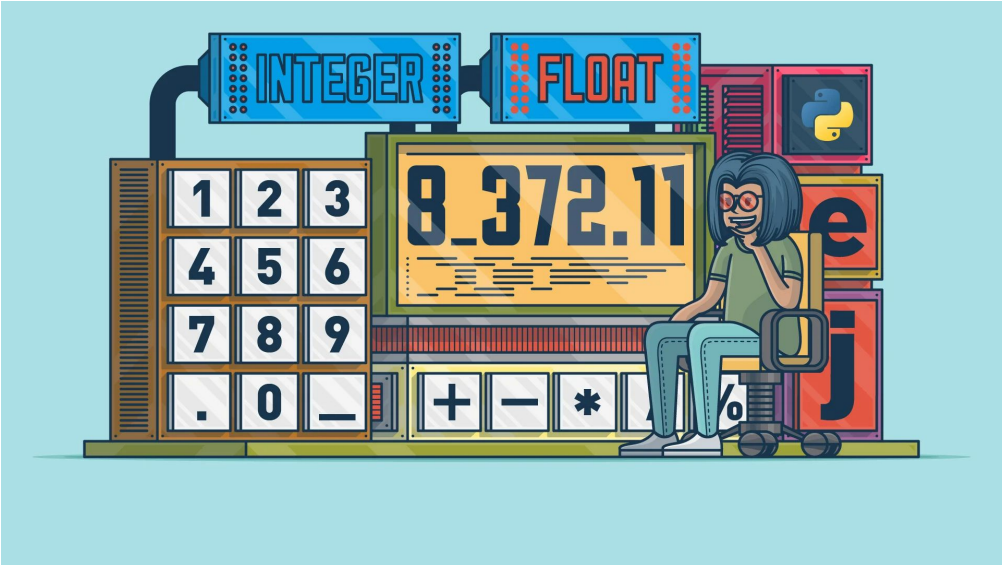




Meet the Numbers

Newton School of Technology

They represent quantities, from counting objects to calculating complex formulas. It could be either simple integers or decimals.



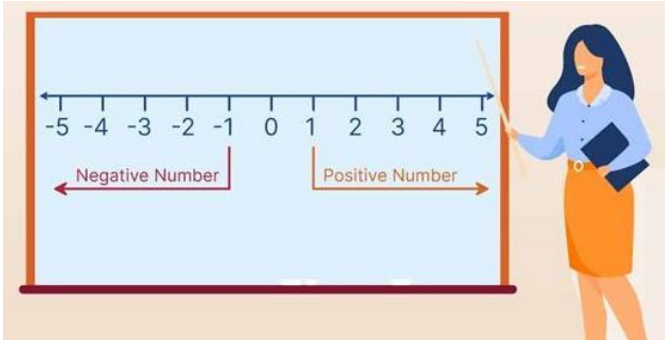
They could either be integers or fractions.

Declaration and Assignment

Newton School of Technology

Let’s declare and assign some numbers:-

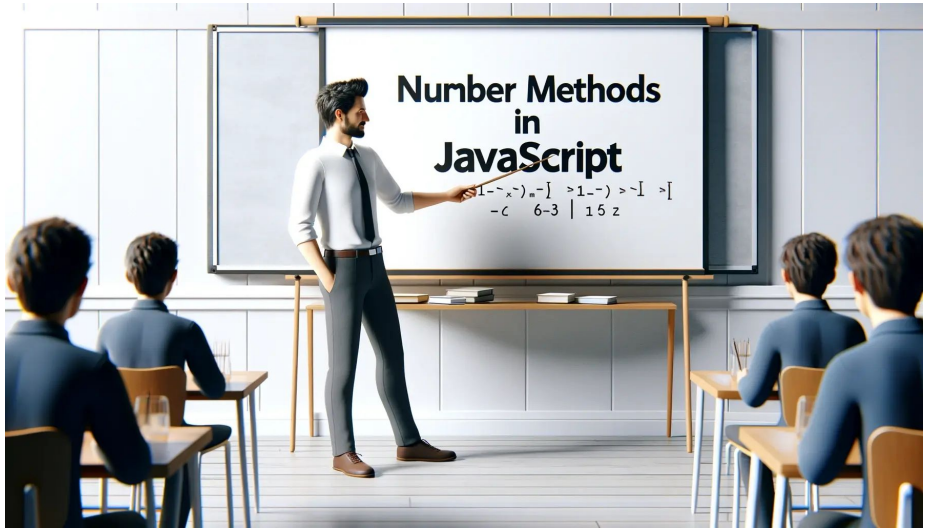
```
let intNum = 10;      // Positive integer
let negInt = -5;      // Negative integer
let floatNum = 3.14; // Positive float
let negFloat = -2.5; // Negative float
```



Number methods

Newton School of Technology

Number methods in JavaScript help manipulate numeric values efficiently. For instance, ceil() rounds fractional value to nearest smaller integer, while floor() rounds the number to nearest integer.

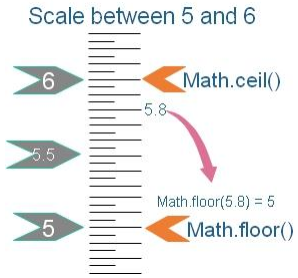


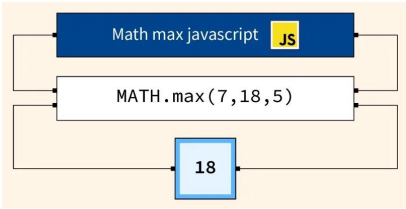
Number Methods

Newton School of Technology

JavaScript provides a variety of built-in methods and operations for working with numbers. Let's start with rounding and converting:-

```
// Math.ceil() - Rounds a number UP to the nearest integer
console.log("Math.ceil(4.3) =", Math.ceil(4.3));  // 5

// Math.floor() - Rounds a number DOWN to the nearest integer
console.log("Math.floor(4.7) =", Math.floor(4.7));  // 4

// Math.max() - Returns the largest of the given numbers
console.log("Math.max(1,5,3,9,7) =", Math.max(1,5,3,9,7));
// 9

// Math.min() - Returns the smallest of the given numbers
console.log("Math.min(1,5,3,9,7) =", Math.min(1,5,3,9,7));
// 1
```



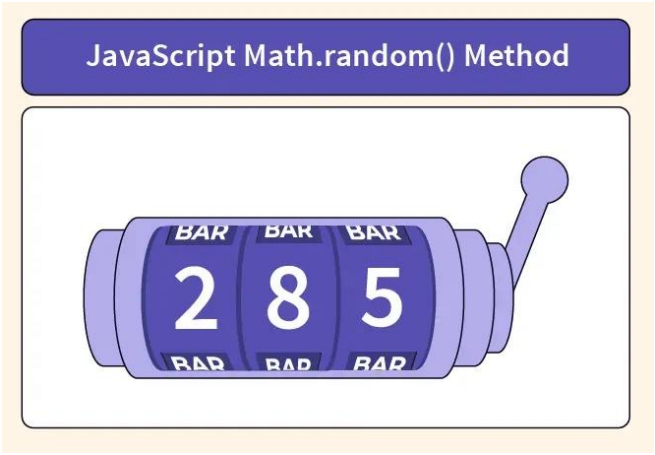


Number Methods

Newton School of Technology

Let’s look at few more:-

```
// Generate a random number between 1 and 10
let randomValue = Math.random();
// Generates a number between 0 (inclusive) and 1 (exclusive)

// Map the random value to a number between 1 and 10
let randomInRange = Math.floor(randomValue * 10) + 1;

console.log("Random number between 1 and 10:", randomInRange);
// Output: Random number between 1 and 10
```



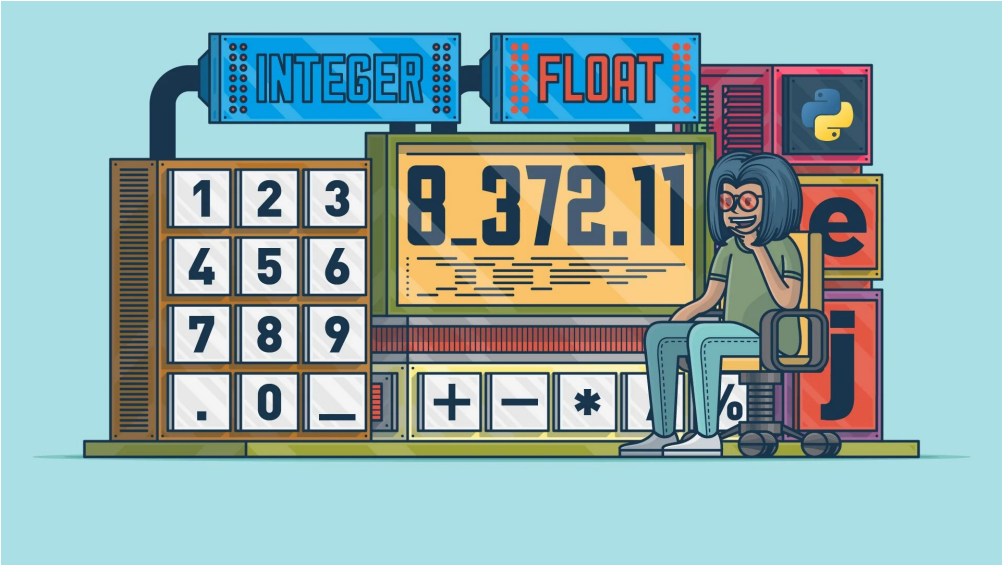
Interactive Activity

Using Math.random() find a number between 2 and 8

Enter the Strings

Newton School of Technology

They are the storytellers of JavaScript. They carry text, allowing us to write messages, descriptions, and even dialogue for our characters.



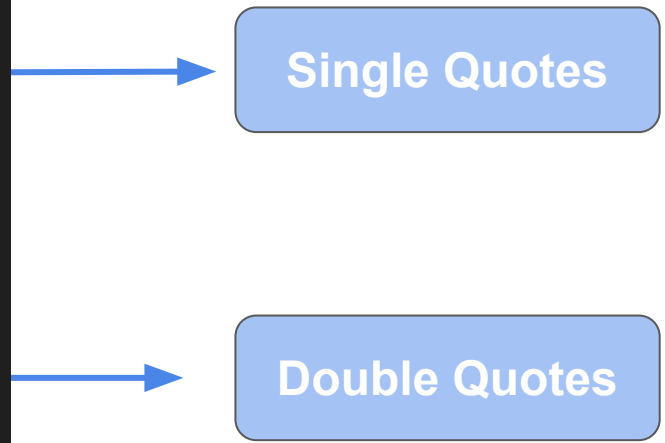
They could either be integers or fractions.

Declaration and Assignment

Newton School of Technology

Let’s declare and print some strings:-

```
1  // Using Single Quotes ('')
2  let singleQuoteString = 'Hello, world!';
3  console.log(singleQuoteString);
4  // Output: Hello, world!
5
6  // Using Double Quotes ("")
7  let doubleQuoteString = "Hello, world!";
8  console.log(doubleQuoteString);
9  // Output: Hello, world!
```



Declaration and Assignment

Newton School of Technology

Let’s learn how to use backticks and how it is different from ‘...’ and “...”

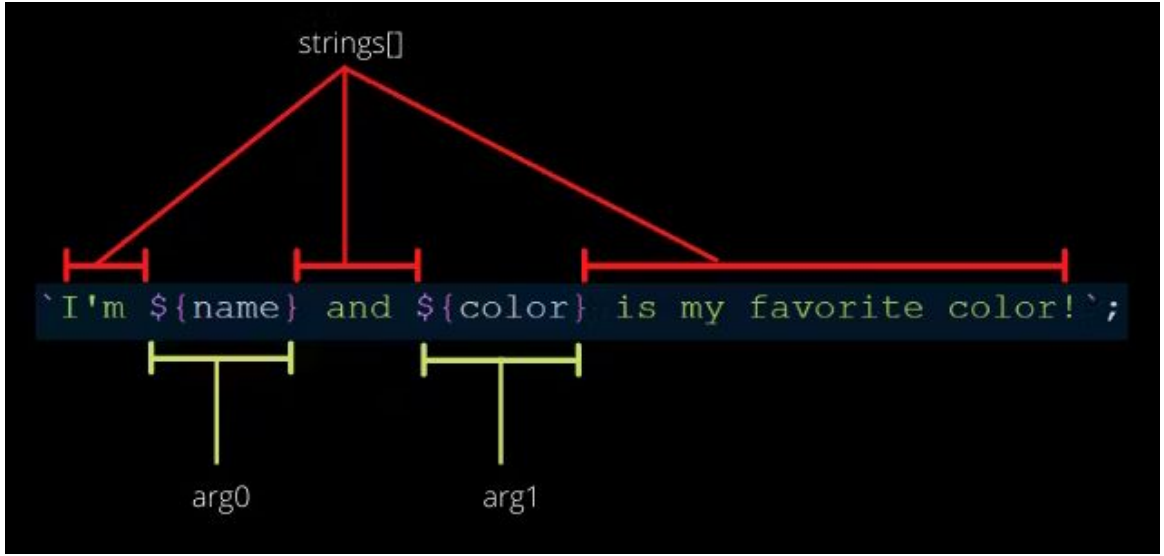
```
// Using Backticks (``)
let backtickString = `Hello, world!`;
console.log(backtickString);
// Output: Hello, world!

// Multi-line string using backticks
let multiLineString = `This is a string
that spans multiple
lines.`;
console.log(multiLineString);
// Output:
// This is a string
// that spans multiple
// lines.
```

Backticks Quotes

Backticks: for template literals

Newton School of Technology

Template literals use **backticks** (``) instead of single or double quotes for defining strings. With them we can embed variables inside the string. Let's see how:-



Here name and color are variables.

Backticks: for template literals

Newton School of Technology

Template literals use **backticks** (``) instead of single or double quotes for defining strings. With them we can embed variables inside the string. Let's see how:-

```
let nam = 'Narendra';
let color = 'blue';

let str = `I'm ${nam} and ${color}
 is my favourite color!`;

console.log(str);
// Output: I'm Narendra and blue is my favourite color!
```

With the help of template literals we embedded variables in the string.

Accessing character values in string

Newton School of Technology

Each character in the string is indexed starting from 0. Consider string as row of mailboxes where each mailbox can hold a character.

For example, let's take string "**Codechef**":

	C	o	d	e	c	h	e	f
Index numbers are:	0	1	2	3	4	5	6	7

Strings are indexed

Newton School of Technology

Let’s try to access these character values using index.

```
let str = "codechef";

// Accessing characters by its index (mailbox number)
console.log("At index 0:", str[0]); // Output: c
console.log("At index 1:", str[1]); // Output: o
console.log("At index 2:", str[2]); // Output: d
console.log("At index 3:", str[3]); // Output: e
console.log("At index 4:", str[4]); // Output: c
console.log("At index 5:", str[5]); // Output: h
console.log("At index 6:", str[6]); // Output: e
console.log("At index 7:", str[7]); // Output: f
```

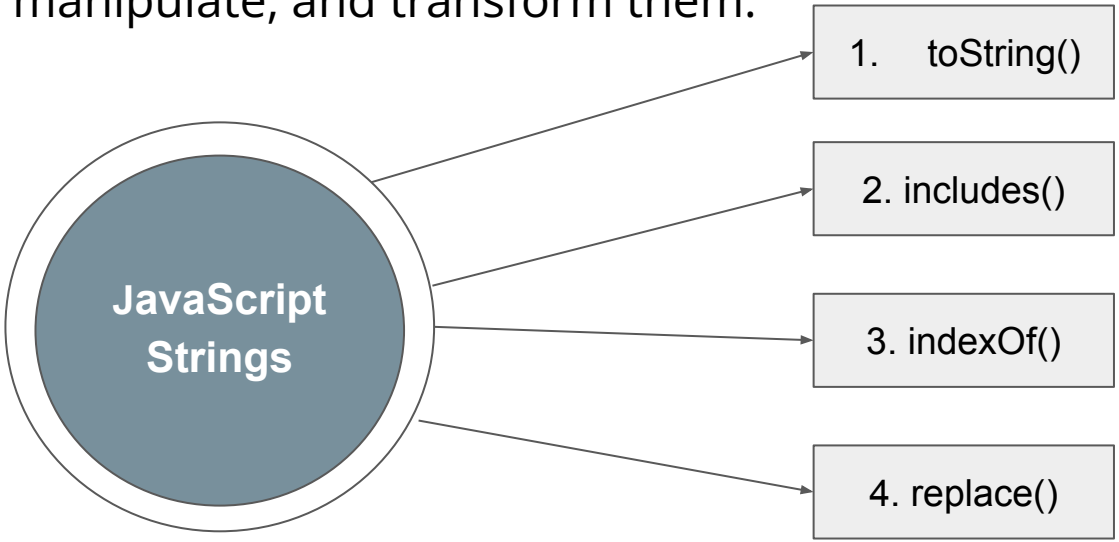
We stored codechef in str variable and extracted each character using their respective indexes.

charAt() function achieves the same using index values.

Is that it! There is so much more!

Newton School of Technology

Strings in JavaScript come with a toolbox full of powerful methods to explore, manipulate, and transform them.



String method: toString()

Newton School of Technology

The toString() method in JavaScript converts a value (such as a number, array, or object) into a string representation.

```
let num = 123;
console.log(num.toString());
// Output: "123"

let arr = [1, 2, 3];
console.log(arr.toString());
// Output: "1,2,3"
```

Converting different data types to strings

Here we are converting numbers and arrays to strings. Observer how they are being converted.

String method: trim()

Newton School of Technology

The trim() method in JavaScript removes whitespace from both ends of a string without affecting the spaces in between words.

```
let str = "  Hello, world!  ";
console.log(str.trim());
// Output: "Hello, world!"
```

Trimming down the white spaces

Can you observe the little white spaces and how they have gotten trimmed?

String method: replace()

Newton School of Technology

The replace() method in JavaScript is used to replace a specified substring or pattern within a string with another substring.

```
let str = "Hello, world!";
let newStr = str.replace("world", "JavaScript");
console.log(newStr);
// Output: "Hello, JavaScript!"
```

We have successfully replaced the word. Repeat it in your code editor.

String method: toString()

Newton School of Technology

The toString() method in JavaScript converts a value (such as a number, array, or object) into a string representation.

```
let num = 123;
console.log(num.toString());
// Output: "123"

let arr = [1, 2, 3];
console.log(arr.toString());
// Output: "1,2,3"
```

Can you observe the little white spaces and how they have gotten trimmed?

String method: includes()

Newton School of Technology

The includes() method in JavaScript checks if a string contains a specified substring. It returns true if the substring is found, otherwise false.

```
let sentence = 'JavaScript is awesome!';
console.log(sentence.includes('is'));
// true
console.log(sentence.includes('awesome'));
// true
console.log(sentence.includes('Python'));
// false
console.log(sentence.includes('Java', 5));
// false (search starts from index 5)
```

It is checking if string includes the word and giving out boolean. Observer the last one, why is it false??

String method: indexOf()

Newton School of Technology

The indexOf() method in JavaScript returns the index of the first occurrence of a substring in a string or -1 if not found. It is case-sensitive.

```
let sentence = 'JavaScript is awesome!';

console.log(sentence.indexOf('is'));
// 11 (index where 'is' starts)
console.log(sentence.indexOf('awesome'));
// 15 (index where 'awesome' starts)
console.log(sentence.indexOf('Python'));
// -1 (not found)
console.log(sentence.indexOf('Java', 5));
// -1 (search starts from index 5)
```

Last one returns -1 since Java word is not available after 5th index.

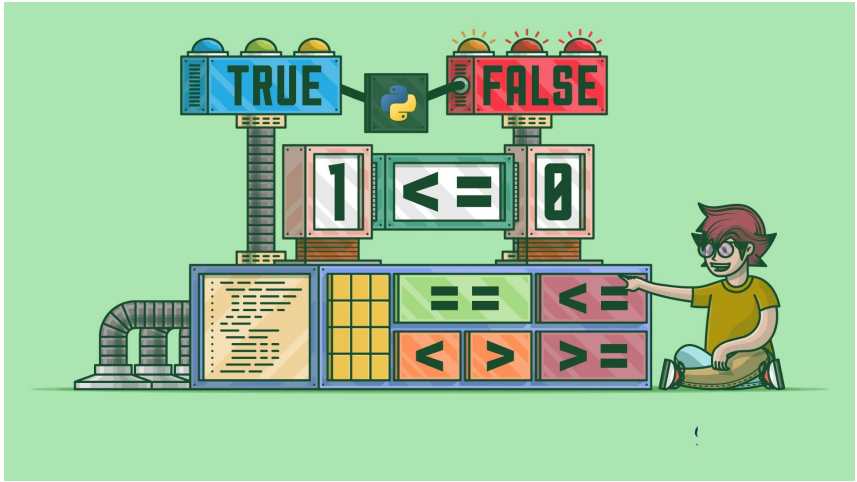
Newton School of Technology

Unraveling the Basics: boolean, null, and undefined

boolean

Newton School of Technology

Think of it as a switch—only two states: true or false.




```
1  let isSunny = true;
2  let isRaining = false;
```

Assigning boolean values

null

Newton School of Technology

This represents "nothing" or "empty". You can assign it intentionally when a variable is meant to have no value.


```
let car = null;
// This variable is explictly set to have no value

console.log(car);
// Output: null
```

undefined

Newton School of Technology

n JavaScript, undefined means a variable has been declared but has not been assigned a value yet. It also represents the default value of uninitialized variables.

undefined

defined

```
// Undefined example
let userAge;
console.log("userAge:", userAge);
// Output: undefined
// (variable is declared but not assigned a value)

// Defined example
let userName = "John Doe";
console.log("userName:", userName);
// Output: "John Doe"
// (variable is assigned a string value)
```

undefined vs defined

Any difference in null & undefined??

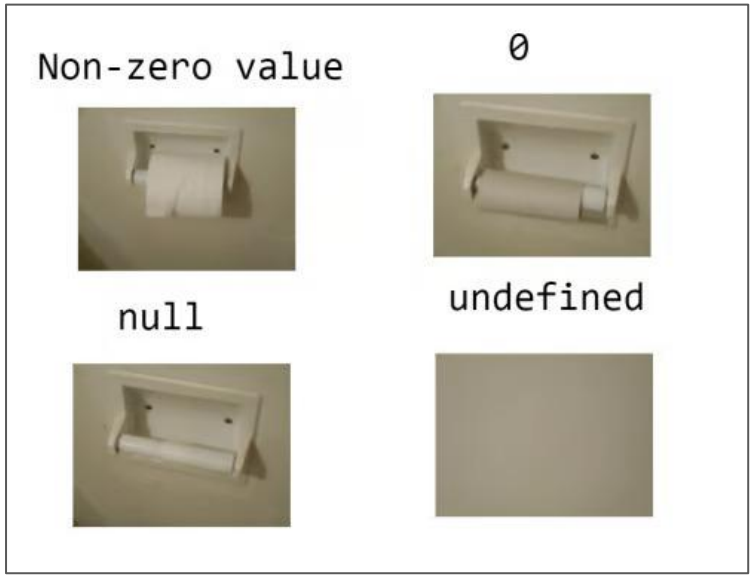
Newton School of Technology

Even though they look similar but they are not same at all. You have fallen in same trap like everyone else:-

Understanding differences

Newton School of Technology

Let’s understand difference between zero, null and undefined



undefined	null
has not been assigned	could be assigned
typeof undefined	typeof object

Truthy and Falsy values

Newton School of Technology

Any value which is not falsy is considered true.

Falsy Expressions
False
NaN
Undefined
Null
Empty String (“”/ ‘’)
0

Apart from these values all the values are considered true.

How to check truthy/falsy values??

Newton School of Technology

We can check truthy/falsy values using Boolean function.

```
let value = "Hello";
// Example truthy value

console.log(Boolean(value));
// Output: true

let anotherValue = "";
// Example falsy value
console.log(Boolean(anotherValue));
// Output: false
```

Here we used:-

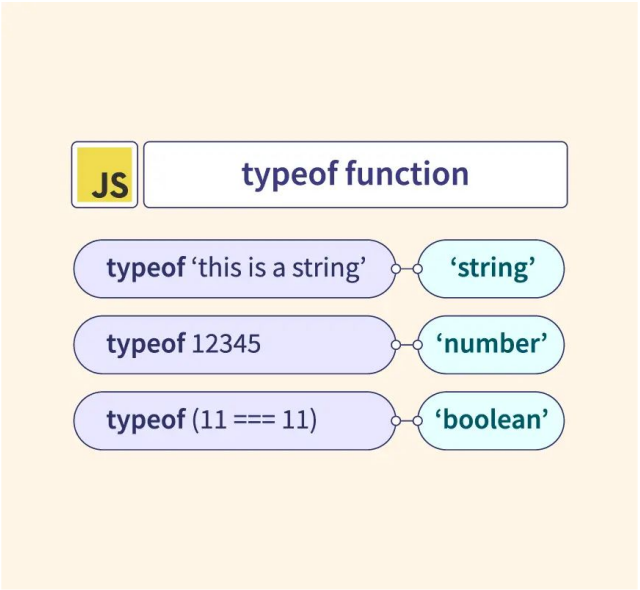
1. “Hello”: Evaluated as true
2. “”: Evaluated as false

Try few other examples by your own.

How to check data type: typeof

Newton School of Technology

The typeof operator is used to determine the type of a given variable or value. It returns a string indicating the type




```
// Declaring and assigning values to variables
let greeting = "This is a string";  // String
let num = 12345;                    // Number
let isEqual = 11 === 11;
// Boolean (True or False)

// Using typeof to check the type of each variable
console.log("Type of greeting:", typeof greeting);
// Output: string
console.log("Type of num:", typeof num);
// Output: number
console.log("Type of isEqual:", typeof isEqual);
// Output: boolean
```

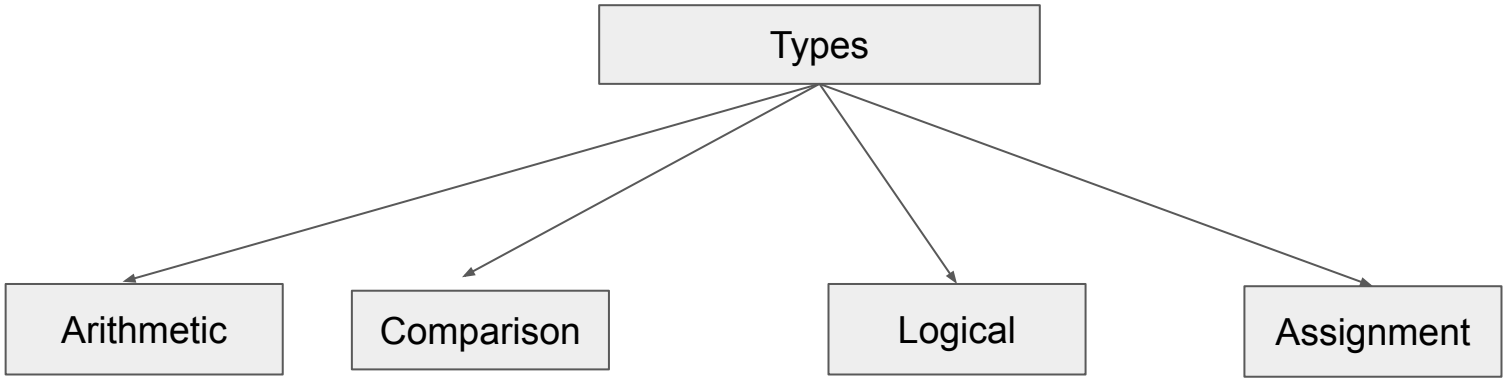
Newton School of Technology

Operators

What are operators?

Newton School of Technology

In JavaScript, **operators** are special symbols used to perform operations on values



Arithmetic Operators

Newton School of Technology

Arithmetic operators in JavaScript are used to perform mathematical operations on numbers.

Arithmetic Operator	Name	Example
+	Addition	a + b
-	Subtraction	a – b
*	Multiplication	a * b
/	Division	a / b
%	Modulus	a % b
++	Increment Operator	a++
--	Decrement Operator	a--

Arithmetic Operators: Example

Newton School of Technology

Here’s are few examples of arithmetic operations in JavaScript:

```javascript
// Declaring two variables
let num1 = 10;  // First number
let num2 = 5;   // Second number

// Addition
let addition = num1 + num2;
console.log("Addition: num1 + num2 =", addition);
// Output: 15

// Subtraction
let subtraction = num1 - num2;
console.log("Subtraction: num1 - num2 =", subtraction);
// Output: 5

// Multiplication
let multiplication = num1 * num2;
console.log("Multiplication: num1 * num2 =", multiplication);
// Output: 50
```

When you see your younger sibling struggling with basic algebra

Unexpected results we get sometimes

Newton School of Technology

There are few anomalies which we face due to impermissible operations. Let's understand with code:-

```
// 1. Division by zero = Infinity
console.log("1 / 0 =", 1 / 0);
// Output: Infinity

// 2. Number + String = string concatenation
console.log("1 + '2' =", 1 + '2');
// Output: '12'

// 3. String - String = numeric subtraction
console.log("'2' - '2' =", '2' - '2');
// Output: 0
```

Here you can observe that not all mathematical operations go as expected.

Data type of infinity

Newton School of Technology

In previous slide you can observer that divide by zero provides infinity, but if infinity is a number. Want to check? Use isNaN().

```
// Checking if value is a number using isNaN
let value1 = Infinity;
let value2 = -Infinity;
let value3 = 42;
let value4 = "Hello";

// Using isNaN() to check if it's a valid number
console.log(isNaN(value1));
// false (Infinity is a number)
console.log(isNaN(value2));
// false (-Infinity is a number)
console.log(isNaN(value3));
// false (42 is a number)
console.log(isNaN(value4));
// true (string is not a number)
```

You can observer infinity is a number.

Comparison Operators

Newton School of Technology

Comparison operators in JavaScript are used to compare two values and return a Boolean value (true or false), helping to evaluate relationships between

>	Greater Than
>=	Greater Than or Equal To
<	Less Than
<=	Less Than or Equal To
==	Equal To
===	Strict Equal To
!=	Not Equal To
!==	Strict Not Equal To

Comparison Operators: example

Newton School of Technology

Here’s a simple code example to demonstrate the use of comparison operators in JavaScript:

```
// Declaring two variables
let x = 10;  // First variable
let y = 5;   // Second variable

// Equality comparison (==)
console.log("x == y:", x == y);
// Output: false (x is not equal to y)

// Strict equality comparison (===)
console.log("x === y:", x === y);
// Output: false (x is not strictly equal to y)

// Inequality comparison (!=)
console.log("x != y:", x != y);
// Output: true (x is not equal to y)
```

Pay caution while choosing ‘==’ over ‘===’ as they have different outcomes sometimes.

Comparison Operators: example

Newton School of Technology

Continued...

```
// Strict inequality comparison (!==)
console.log("x !== y:", x !== y);
// Output: true (x is not strictly equal to y)

// Greater than comparison (>)
console.log("x > y:", x > y);
// Output: true (x is greater than y)

// Less than comparison (<)
console.log("x < y:", x < y);
// Output: false (x is not less than y)

// Greater than or equal to (>=)
console.log("x >= y:", x >= y);
// Output: true (x is greater than or equal to y)

// Less than or equal to (<=)
console.log("x <= y:", x <= y);
// Output: false (x is not less than or equal to y)

```

The correct placement of < or > before = is important for clarity.

Incorrect:- =<
Correct: <=

Logical Operators

Newton School of Technology

Logical operators in JavaScript are used to perform logical operations on values, typically boolean values (true or false).

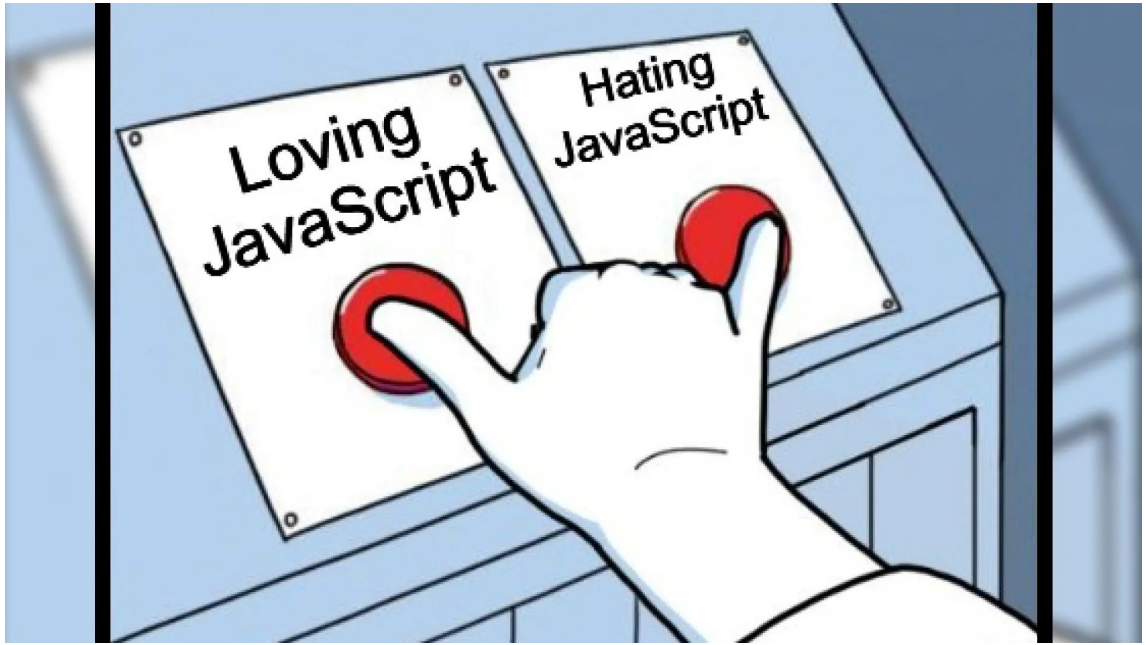
AND, OR, and NOT

Operator	Name	Example	Result
&&	AND	x < 10 && x !== 5	false
||	OR	y > 9 || x === 5	true
!	NOT	!(x === y)	true

Now how are you feeling??

Newton School of Technology

You must be having love-hate relationship now. Don’t worry, you will feel at home more you practice:-



Newton School of Technology

Some Real World Examples

Calculating Discounts and Special Offers

Newton School of Technology

When shopping, it's crucial to calculate the best price after discounts. For example, if an item costs $100 and has a 20% discount.

```
// Original price of the item
let originalPrice = 100;

// Discount percentage
let discountPercentage = 20;

// Calculate discount amount
let discountAmount = (originalPrice * discountPercentage) / 100;

// Calculate final price after discount
let finalPrice = originalPrice - discountAmount;

console.log("Original Price: $", originalPrice);
// Output: 100
console.log("Discount Amount: $", discountAmount);
// Output: 20
console.log("Final Price After Discount: $", finalPrice);
// Output: 80
```

Using arithmetic operators to determine the discount amount and the final price.

Job Offers: Making Right Choice

Newton School of Technology

To choose the better job offer, you can use comparison operators to evaluate salaries.

```
// Salary offers from two job companies
let offerA = 60000;  // Job offer A salary
let offerB = 75000;  // Job offer B salary

// Compare salaries using comparison operators
let isOfferAGreater = offerA > offerB;
// Checks if offer A is greater than offer B
let isOfferBGreater = offerB > offerA;
// Checks if offer B is greater than offer A

// Display the best offer
if (offerA > offerB) {
  console.log("Choose Offer A for a higher salary.");
} else if (offerB > offerA) {
  console.log("Choose Offer B for a higher salary.");
} else {
  console.log("Both offers are the same.");
}
```

Using a greater-than and smaller than to help you quickly see which offer provides a higher salary.

Practice! Practice! And become boss

Newton School of Technology

More you practice more you find easy way outs and more you will become confident in writing javascript code. Practice is the key...

In Class Questions

Newton School of Technology

Thanks for watching!