Slide Note

Social networking sites, job portals, news websites, online shopping sites, are few of the popularly visited websites.

Wikipedia is another website that is also popularly visited to gather information.

The sites that allow users to provide inputs are interactive ones and the others are non-interactive.

Social networking sites, job portals, online shopping sites are interactive websites.

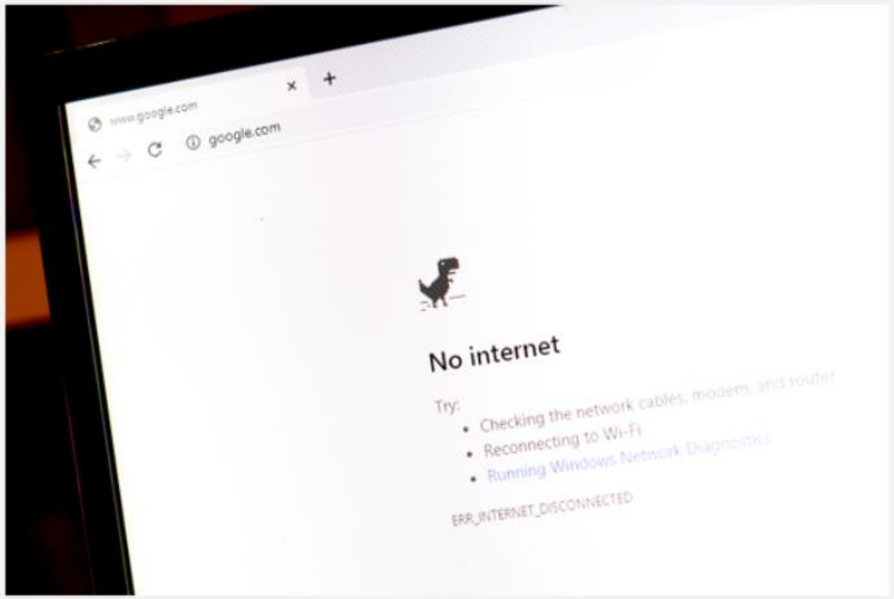
Wikipedia is a non-interactive website.

With HTML and CSS, responsive and state of the art websites can be designed, but these technologies do not help to make webpages interactive.

Time to Think

- Write down some of the websites you visit on a daily basis.
- Categorize the websites as interactive and non-interactive.
- How can you add interactivity to webpages with the tools and technologies learned so far?

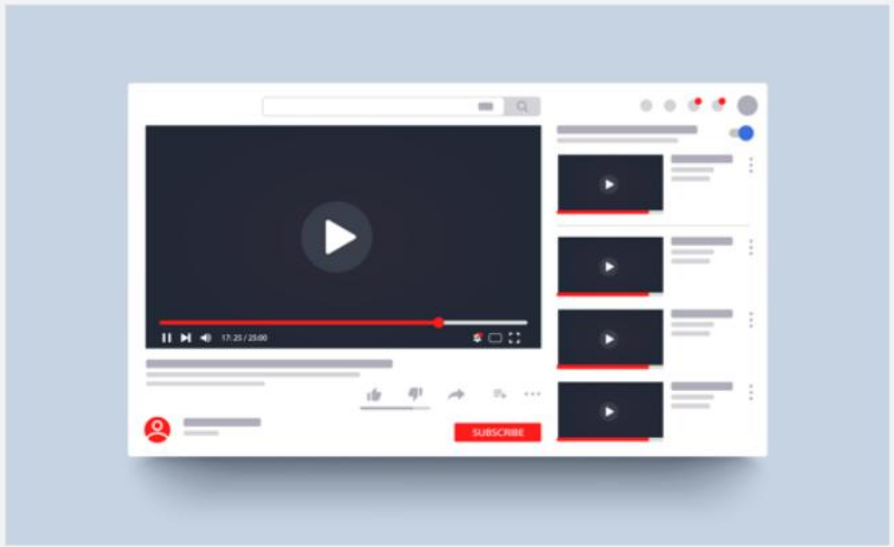
Slide Note



Playing Game in Browser

- Browsers were meant to view web pages.
- How are we able to play games?
- Ever wondered how Google Chrome's Dinosaur Game is implemented?
- Visit the link from Google Chrome browser to play the game.

Slide Note



Watching Online Videos

- How is it possible to play, pause, forward, and rewind a video on YouTube?

Menu

00:03:00:00:35 03/ 35

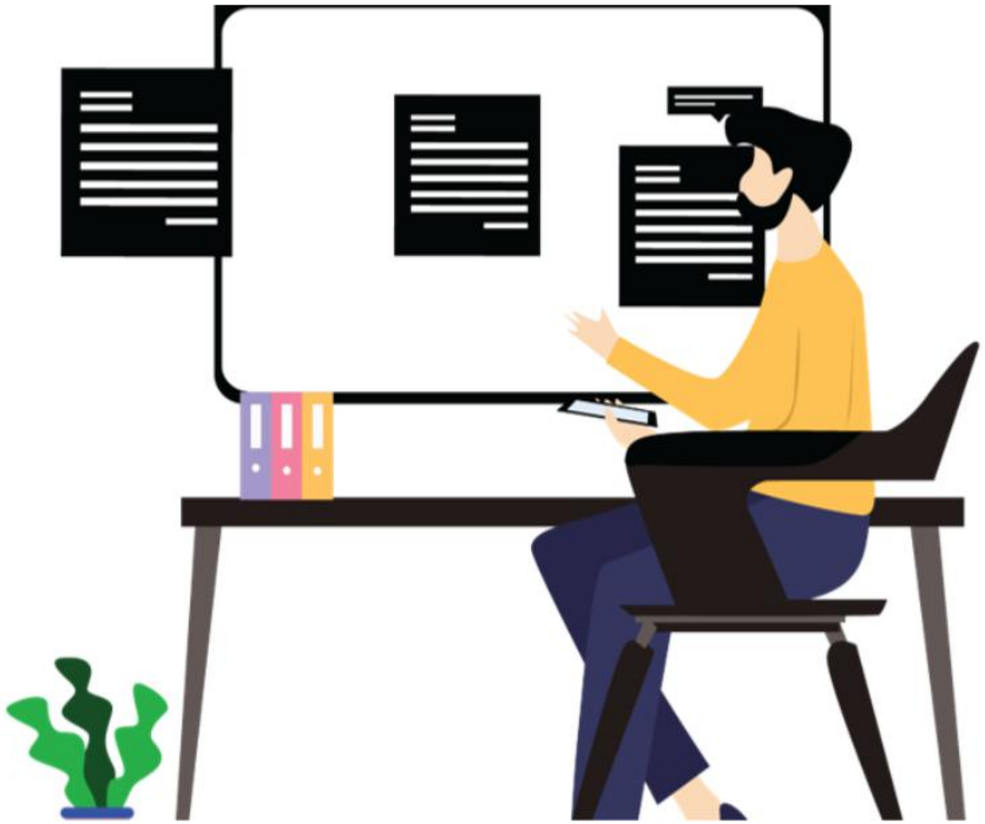
Slide Note

JavaScript is a scripting language that helps to add interactivity to the webpages designed using HTML and CSS

The common thing for all websites discussed is that they are all interactive.

Slide Note

Getting Started With JavaScript



Slide Note

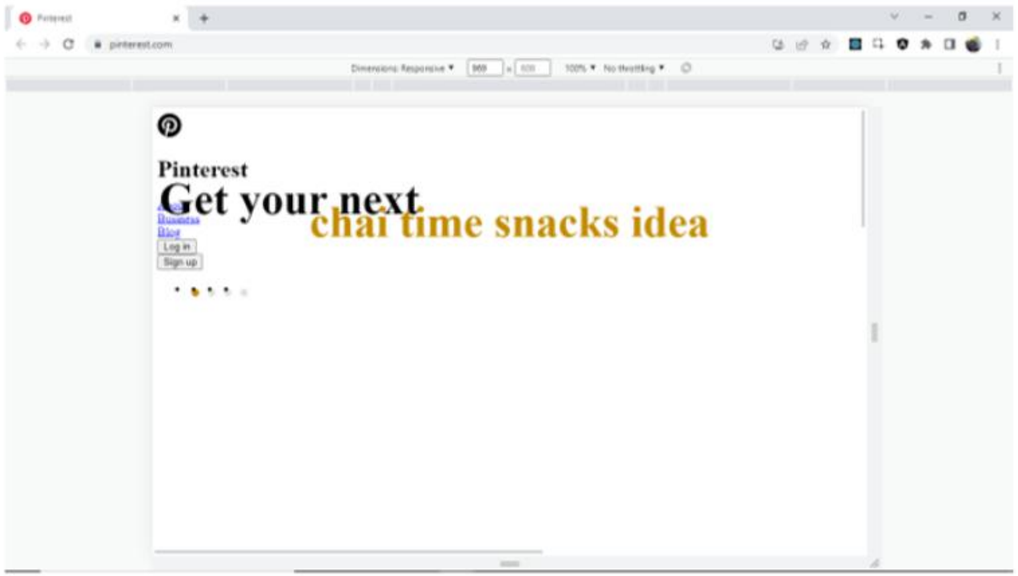
Learning Objectives

- Write Hello World program in JavaScript
- Define datatypes and literals
- Use variables and constants to store data
- Define template literals
- Explain `typeof` operator
- Implement conditional constructs for writing decision making code
- Implement loop constructions for writing iterative programs

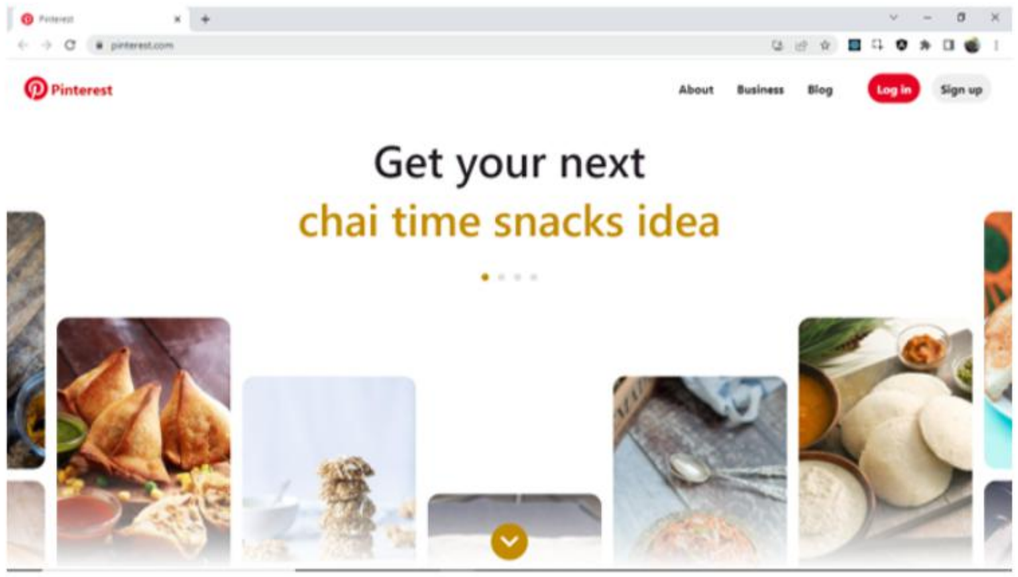
Slide Note

Web Page Design

Role of HTML – Structuring Web Page



Role of CSS – Styling Web Page



Slide Note

Play the video clip on slide and experience the interactivity in a web page

Need for Interactive Websites

An interactive website communicates and allows for interaction with users.

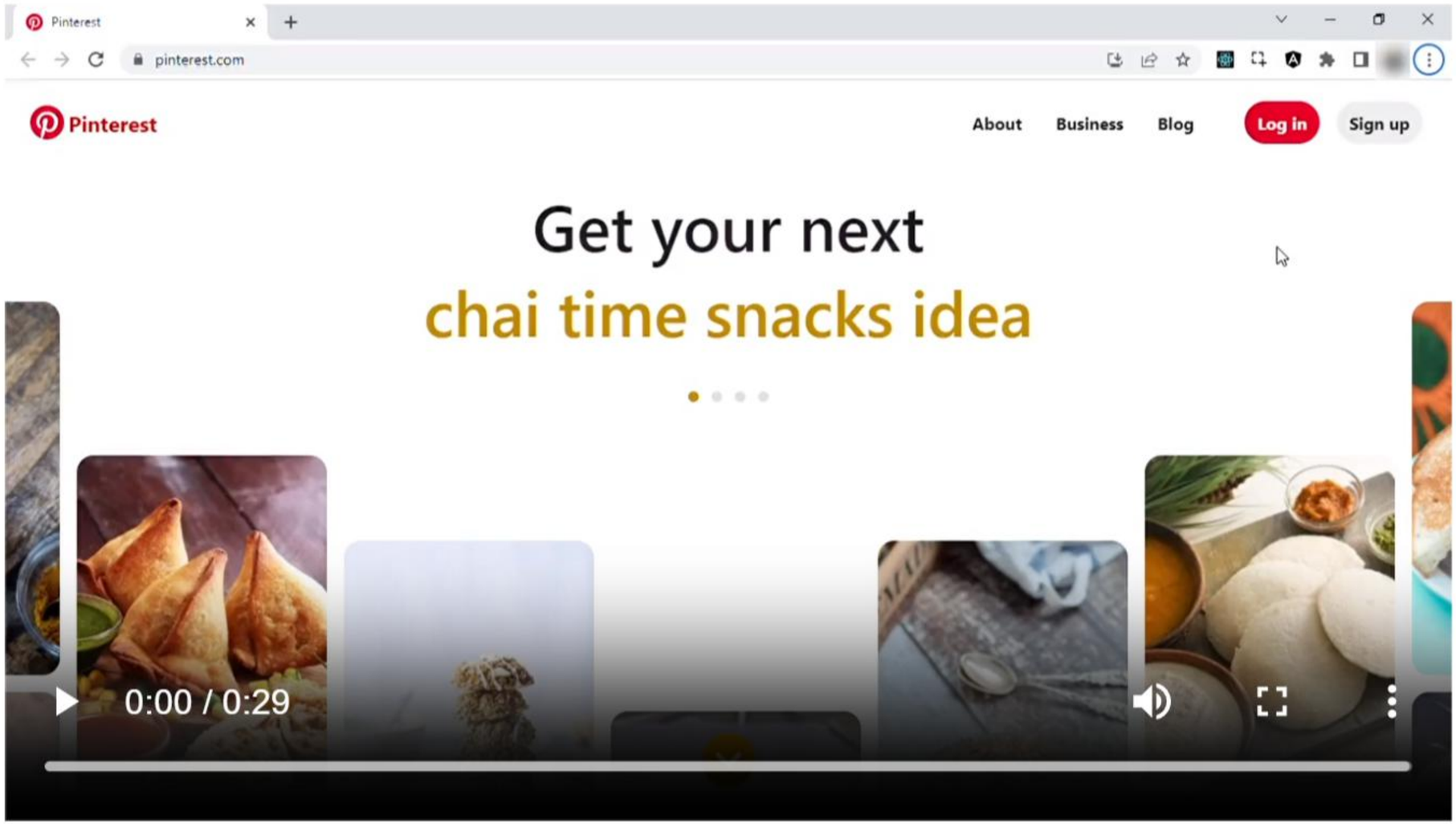
Interactive business websites can make website less boring.

Users will spend more time on a website that interacts with them.

Interactive sites provide more personalized user experience.

Improves brand awareness by creating a lasting effect in the users' minds.

Interactive Web Page



Slide Note

How to Make a Web Page Interactive?

- Implementing interactivity in a web page requires programming.
- Programming is all about writing instructions and getting them executed.
- **JavaScript** is a tool that allows developers to write programs and make a page interactive.
- Designing interactive pages helps to
  - Dynamically modify content and style of the web page
  - Provide response to user's interaction

Slide Note

JavaScript engine is a software component residing inside browser that executes JavaScript code.

Can JavaScript also be used for server-side programming?

Yes. (however, this discussion is beyond the scope of this program)

What is JavaScript?

- JavaScript is the most popular programming language for **web development**.
- JavaScript is a Scripting Language.
- A web browser interprets JavaScript instructions and makes the page interactive.
- Each browser has its own JavaScript engine that executes the JavaScript code.
- A web browser is running at the client end, and therefore JavaScript executing in the browser is termed as **Client-side Scripting Language**.
- JavaScript is a case-sensitive language.
    - For eg. `console` in JavaScript IS NOT EQUAL TO `Console` or `CONSOLE`
- Although not mandatory, all JavaScript statements must end with a semicolon ";".
    - A semicolon marks the end of a statement, so if multiple statements are written in a single line, JavaScript can determine the end of one statement and beginning of next instruction.

Slide Note

Hello World is typically the first program that learners write whenever they start learning a new programming language

Snippets are scripts that you author in the Sources panel of Chrome's Developer Tools

To create new code snippet:

Open a blank page in browser

Open DevTools,

Select the "Sources" tab,

Then the "Snippets" tab on the left.

Right click anywhere in the snippets' list and select "New".

To execute a code snippet, press CMD+Enter.

The point to be highlighted here is JavaScript is a case-sensitive language.

Hello World in JavaScript

Display the text Hello World on web console.

Click here for demo solution.

DEMO

Menu

00:11:00:00:35

11/ 35

Slide Note

Operations can be simple if they involve the same types of data

Operations can be complex if they involve different types of data

For complex operations, data from one type gets converted into another type.

More on complex operations shall be discussed later in the sprint.

Data Types

- Data types define a set of values and set of operators that are used to perform operations on those values and generate a result.
- For example, in JavaScript:
    - `5` and `3` are two numbers which result into a number when they are added, subtracted, multiplied or divided
    - "`Hello`" and "`World`", which when concatenated produces a string "`Hello World`"
- Data types in JavaScript are classified into two categories:
    - Primitives – immutable values
    - Objects – value is memory referenced by an identifier

Slide Note

JavaScript Primitive Types

Data Types	Description	Example
String	Represents textual data within single or double quotes	'hello', "hello world!" etc
Number	An integer or a floating-point number	3, 3.234, 3e-2 etc.
BigInt	An integer with arbitrary precision	900719925124740999n, 1n etc.
Boolean	Any of two values: true or false	true and false
undefined	A data type whose variable is not initialized	let a;
null	Denotes a null value	let a = null;
Symbol	data type whose instances are unique and immutable	let value = Symbol('hello');

Slide Note

Literals in JavaScript

- In JavaScript, literals are to be taken literally and they represent values.
- They are the fixed values and are associated with a specific data type.
- The table below lists a few types of literals with examples.

Literal type	Description	Examples
String	A string literal is zero or more characters enclosed in double (") or single (') quotation marks.	'hi', "how are you", "A-17101", "1425559870"
Numeric	Numeric literals are integer literals as well as floating-point literals	0, -10, 3456
Floating-point	A floating-point literal has integer, decimal point, fraction and an exponent	3.1415, .3412, 3.1E+2,
Boolean	Boolean literal has two literal values `true` and `false`	

Slide Note

Using const and let thoughtfully, will help you clearly denote which variables should change and which should not.

Following this principal helps improve code readability as other developers will have a clear sense of how variables change throughout program execution .

Naming Conventions

In computer programming, a naming convention refers to a standard practice being followed for choosing the character sequence to be used for identifiers like variables and constants.

The code continues to execute if naming conventions are not followed, however, naming conventions help:

§ To reduce the effort needed to read and understand source code.

§ To enable code reviews to focus on issues more important than syntax and naming standards.

In JavaScript, for variable names use **lowerCamelCasing**,

Use names that are *concise*, *human-readable*, and are *semantic*.

Examples of good variable names: count, firstName, laptopBagPrice

Examples of bad variable names: a, Lastname, thisIsAVeryLongVar

Variables and Constants in JavaScript

- Data in program are stored in **variables or constants**.
- They are the named storage for data.
- Variable can be re-assigned with new data whereas constants cannot be.
- In JavaScript, variables are declared using the `let` keyword and constants using `const` keyword.
- There are two limitations on names in JavaScript:
  - The name must contain only letters, digits, or the symbols like dollar or underscore.
  - The first character must not be a digit.
- Because JavaScript is a case-sensitive language, variable names `Firstname` ≠ `firstname`

```
//let keyword
let radius = 10;
console.log(radius); // 10
const pi = 3.14;
radius = 5;
pi = 31; // error
```

Slide Note

JavaScript sees $firstname and $lastname assigned with strings and therefore it concatenated the two values using + operator

JavaScript sees value1 and value2 assigned with numbers and as a result it added the two values using + operator

Dynamically Typed JavaScript

- A language is dynamically typed if the type is associated with run-time values, and not named variables or fields.
- This means that the programmer can write a little quicker because there is no requirement to specify types every time.
- Most scripting languages have this feature.

```
let firstName = 'John'; // firstName is string since it is assigned a string value.
let lastName = 'Smith'; // lastName is string since it is assigned a string value.

let value1 = 100; // value1 is number since it is assigned a numeric value.
let value2 = 200; // value2 is number since it is assigned a numeric value.

console.log(firstName + lastName);  // Outputs JohnSmith, + concatenates strings
console.log(value1 + value2);       // Outputs 300, + adds numbers
```

Slide Note

Print Values of Radius and Pi of Circle

For a circle with a radius of 10, store and print the values of radius and pi. (Note: pi is equal to 3.14).

Click here for the demo solution.

DEMO

Slide Note

Template Literals

- Template literals are also known as template strings.
- A Backtick (`) character is used to delimit the template literals.
- Useful for defining multi-line strings for string interpolation with embedded expressions.
- Embedded expressions are delimited by a dollar sign and curly braces: `${expression}`
- Template literals make writing `console.log()` messages with values convenient, since the complete message with expressions are part of single set of backticks.

```
let radius = 10;
const pi = 3.14;

console.log(`radius = ${radius} and pi = ${pi}`);
```

Template literal

Embedded expression

Slide Note

Ternary operator will be discussed in next sprint.

Operators

- In programming, an operator is a character or characters that determine the action that is to be performed or considered.
- Operators operate on operands and generate a result.
- For example, in an addition operation like:
- x + y
  - x and y are operands
  - + is an operator
- JavaScript operators can be classified into 3 categories:
  - **Unary operators:** These operators processes on single operand
  - **Binary operators:** These operators operate on two operands.
  - **Ternary operator**: This is a special type of operator that takes 3 operands

Slide Note

Basic Operators in JavaScript

Types of Operators	Operators	Meaning
Arithmetic Operators	`+, -, *, /, %, **`	An arithmetic operator takes numerical values (either literals or variables) as their operands and returns a single numerical value.
Assignment Operators	`=, +=, -+, *=, /+, %=, **=`	An assignment operator assigns a value to its left operand based on the value of its right operand.
Unary Operators	`+, -, ++, --, delete`	A unary operation is an operation with only one operand.

Slide Note

The alternative syntax of typeof operator allows omitting ()

typeof(x) = typeof x

typeof Operator

- The `typeof` is an operator that helps determine the data type of operand.
- It returns a string which is the type of unevaluated operand.
- When an undeclared variable is passed, the `typeof` operator returns ‘undefined’.
- The developers can use this operator to validate the type of incoming data.

```
console.log(typeof(34)); // prints number

console.log(typeof("34")); // prints string

console.log(typeof(false)); // prints boolean

console.log(typeof(x)); // prints undefined
```

Slide Note

From among the above listed operators, JavaScript supports 2 sets of comparison operators for equality and inequality checks

== (equal) and === (strict equal)

!=(not equal) and !== (not strict equal)

The strict equal and strict not equal operators not only checks for values but also the types

For example, 3 == '3' returns true, but 3 === '3' returns false, since their types are different. 3 is a number where as '3' or "3" is a string

Comparison Operators

Operator	Description	Examples Returning True

Equal (==)	Returns true if the operands are equal.	`3 == value1`
		`"3" == value1`
		`3 == '3'`

Not equal (!=)	Returns true if the operands are not equal.	`value1 != 4` `value2 != "3"`
Strict equal (===)	Returns true if the operands are equal and are of the same type.	`3 === value1`
Strict not equal (!==)	Returns true if the operands are of the same type but not equal or are of a different type.	`value1 !== "3"` `3 !== '3'`
Greater than (>)	Returns true if the left operand is greater than the right operand.	`value2 > value1` `"12" > 2`
Greater than or equal (>=)	Returns true if the left operand is greater than or equal to the right operand.	`value2 >= value1` `value1 >= 3`
Less than (<)	Returns true if the left operand is less than the right operand.	`value1 < value2` `"2" < 12`

Note: value1 and value2 are two variables initialized with the values 3 and 4 respectively.

Slide Note

Logical Operators

Operator	Usage	Description
Logical AND (&&)	`expr1 && expr2`	When used with Boolean values, && returns true if both operands are true; otherwise, returns false.
Logical OR (||)	`expr1 || expr2`	When used with Boolean values, || returns true if either operand is true; if both are false, returns false.
Logical NOT (!)	`!expr`	Returns false if the operand can be converted to true; otherwise, returns true.

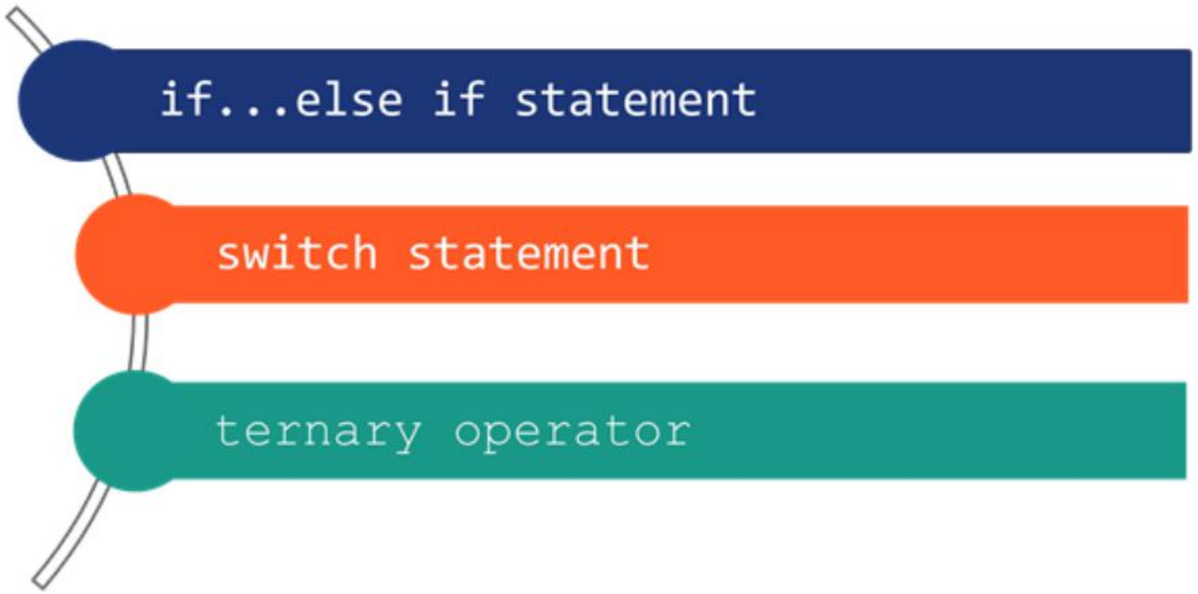
Note::

- Comparison operators have higher precedence than Logical Operators.
- Both set of these operators have Left-to-Right associativity.

Slide Note

Conditional Constructs

- A conditional construct consists of a condition and a task. When the condition is true, the application performs the task.
- Below are the JavaScript statements / operator that help build conditional constructs:



Slide Note

The if…else if Construct

```
let value = 111;

if (value >= 100) {
    console.log("Value is 3 or more digited number");
} else if (value >= 10) {
    console.log("Value is 2 digited number");
} else {
    console.log("Value is single digited number");
}
```

```
Output:

Value is 3 or more digited number
```

- In this scenario, we have two conditions.
- First expression is `value>=100`, if true, then the statement inside the first `if` block is executed.
- Second is `else if` expression `value>=10`, if true then the statement inside the second `if` block is executed.
- If none of the conditions are true, then the statement inside the `else` block is executed.
- As the first expression is true, i.e., (111>=100) the output is, "`Value is 3 or more digited number`".

Slide Note

Determine Grade

Write a program that assigns a grade based on the value of a test score.

Grade A for scoring 90 or above.

Grade B for scoring between 80 and 89.

Grade C for scoring between 70 and 79.

Grade D for scoring less than 70.

Click here for demo solution.

DEMO

Slide Note

```
let value = 100;

switch (value) {
    case 50:
        console.log("50");
        break;
    case 100:
        console.log("100");
        break;
    case 150:
        console.log("150");
        break;
    default:
        console.log("Value not in 50, 100 or 150");
}
```

```
Output:

100
```

The switch Construct

- If the switch expression (value) matches with the case value1 (50), the statement of case value1 is executed.
- Similarly, the code of case value2 (100) is executed if the expression (value) matches with value2.
- If there is no match, the code of the default case is executed.
- Each case statement can have an optional break statement.
- When the control reaches the break statement, the control jumps to the statement located after the switch expression.
- If a break statement is not found, it executes the next case. (Break is a statement that is used to break the current execution flow of the program.)

Slide Note

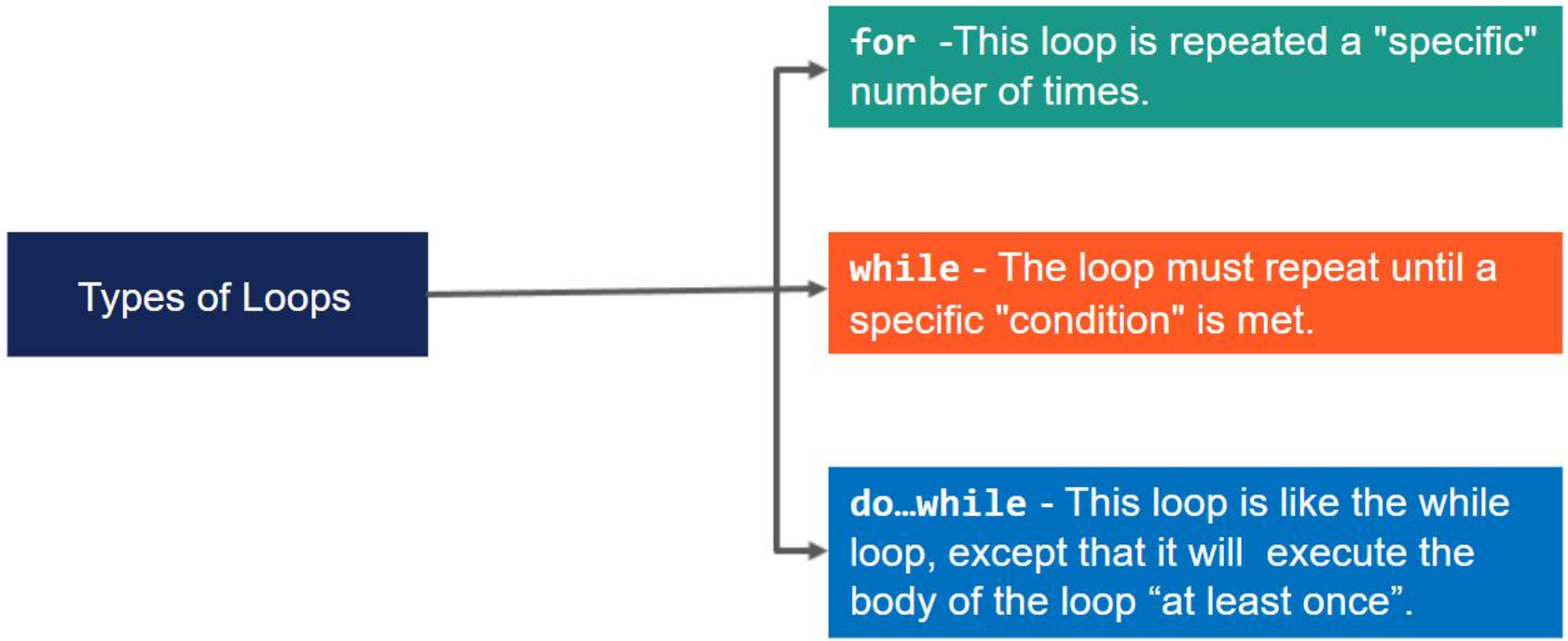
What Are Loop Constructs?

- In a programming language, loops are programming constructs that help execute code multiple times by writing it only once.
- Loop constructs are composed of:
  - A loop statement that controls the loop iterations.
    - A loop statement is defined using a loop keyword and a condition expression.
  - A loop body that contains one more statement that executes every time a loop repeats.
    - A loop body defines a block of statements and is defined using delimiters like curly braces ({ }).

Slide Note

Loop Constructs

- Loops enable you to execute the same block of code several times.
- You can reduce a set of repetitive instructions down to one instruction by using a loop.



Slide Note

for loop generally is used as a counter loop

Step 1: Loop initializes with counter value set to the start value

Step 2: Condition is checked to ensure counter value is within the limit

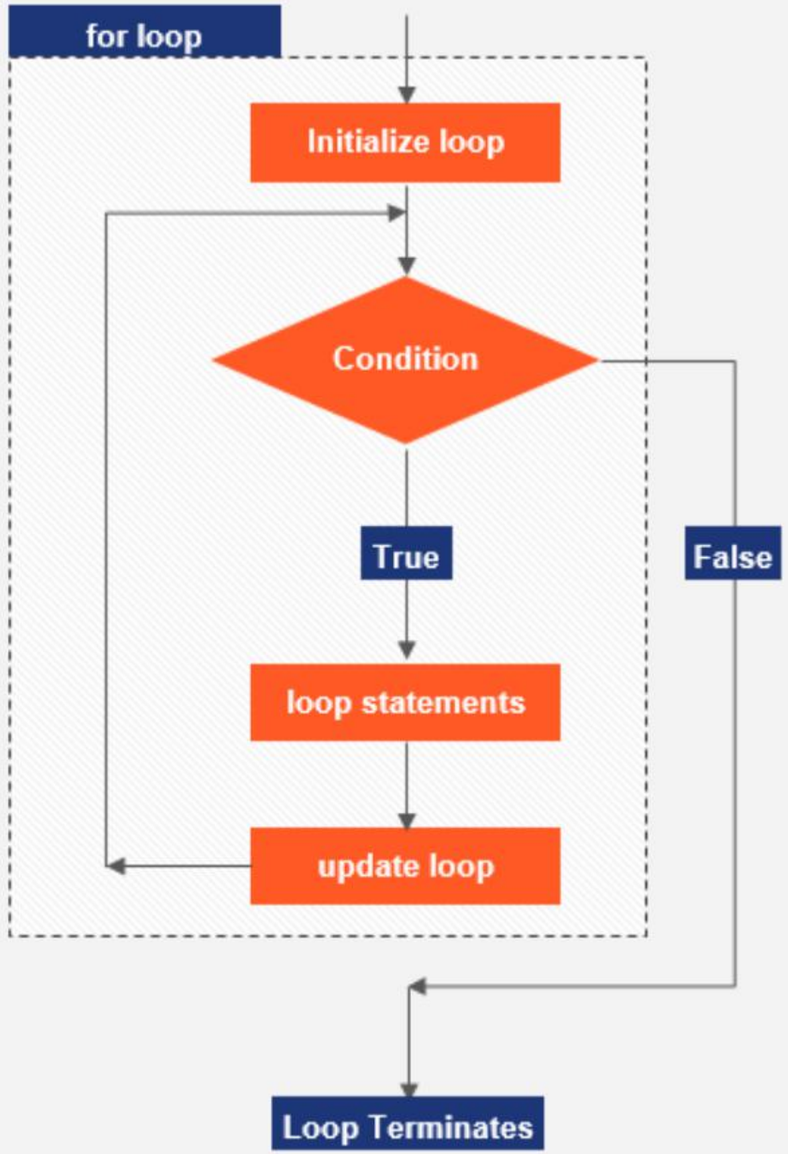
Step 3: If condition is true, loop statements execute, else loop terminates

Step 4: Counter value is updated

Steps 2, 3 and 4 repeat till the condition evaluates to true

for Loop

- A `for` loop repeats for a specific number of times.
- It loops through a block of code several times.
- The image on the slide explains the flow of control of the `for` loop.



Slide Note

`for` Loop Demos

Using `for` loop:

1. Calculate average of even numbers between 1 and 10.
2. Show the first 10 numbers in reverse order.
3. Demonstrate infinite loops.

Click here for the Demo Solution.

DEMO

Slide Note

A while loop is used only when the condition is known that will determine the iterations of the while loop.

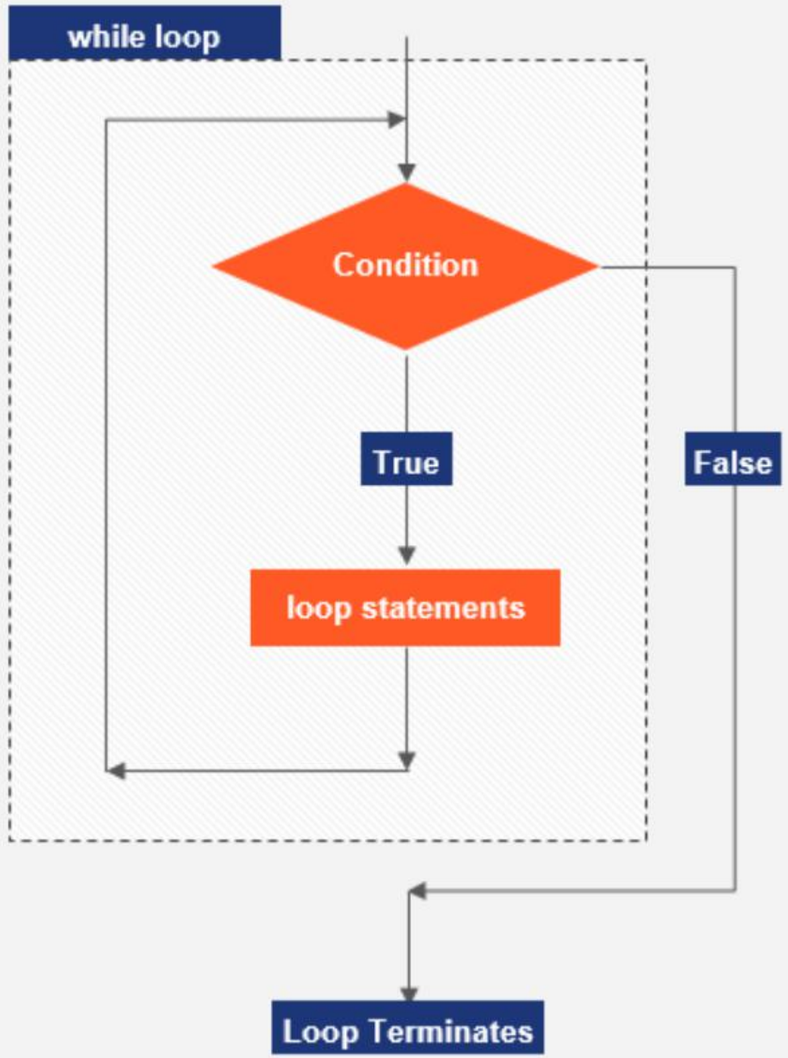
Step 1: Condition is evaluated

Step 2: If condition is true, loop statements execute, else loop terminates

Steps 1 and 2 repeat until the condition evaluates to true.

`while` Loop

- A `while` loop executes its statements as long as a specified condition evaluates to true.
- The figure on the slide explains the flow of control of the `while` loop.



Slide Note

Like while loop, do while loop iterations are also determined by the condition. The loop repeats until the condition is true

Step 1: Loop statements are first executed.

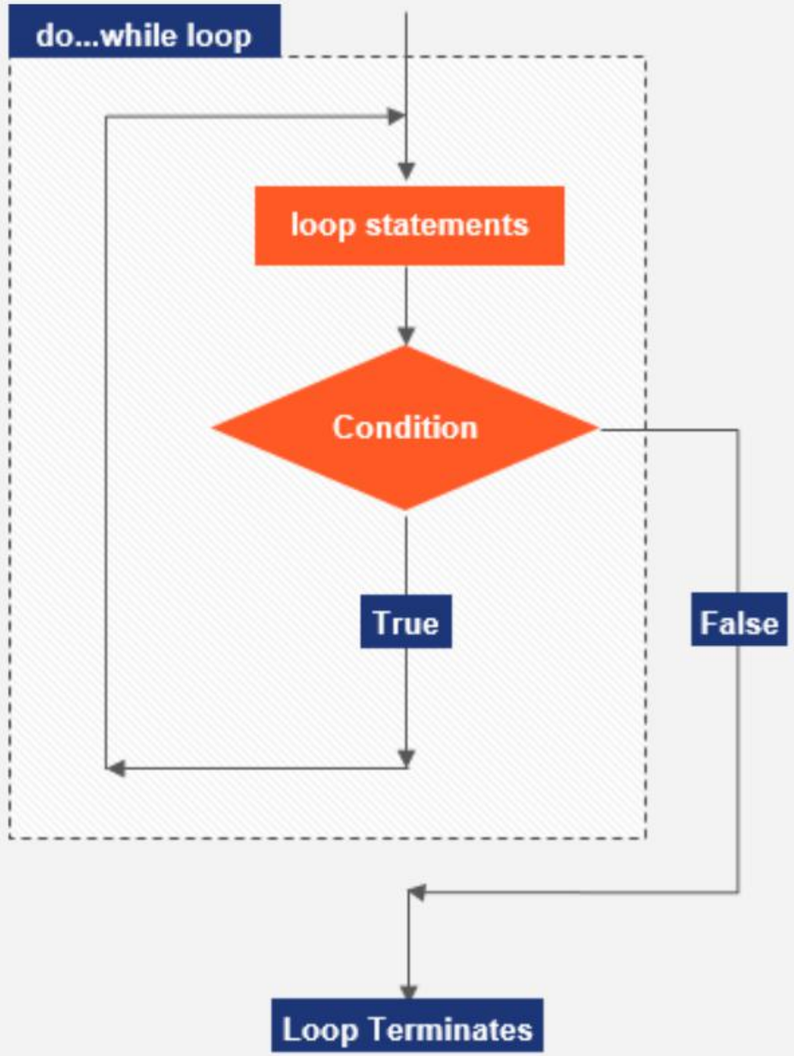
Step 2: Condition is evaluated

Step 3: If condition is true, loop repeats, else it terminates

Therefore, a do…while loop always executes its loop statements, at least once.

do…while Loop

- A `do…while` loop executes its statements until the specified condition evaluates to false.
- The figure on the slide explains the flow of control of the `do...while` loop.



Slide Note

Print Power of 2

Write a JavaScript program that prints powers of 2 up to 1024.

Expected output is:
1 2 4 8 16 32 64 128 256 1024

Click here for the demo solution.

DEMO