

Session: 11

JavaScript - I

For Aptech Centre Use Only

Objectives

- Explain scripting
- Explain the JavaScript language
- Explain the client-side and server-side JavaScript
- List the variables and data types in JavaScript
- Describe the JavaScript methods to display information
- Explain escape sequences and built in functions in JavaScript
- Explain events and event handling
- Explain jQuery
- Describe how to use the jQuery Mobile
- Explain operators and their types in JavaScript
- Explain regular expressions in JavaScript
- Explain decision-making statements in JavaScript
- Explain while loop
- Explain for loop
- Explain do..while loop
- Explain break and continue statement
- Explain single-dimensional arrays
- Explain multi-dimensional arrays
- Explain for..in loop

Introduction

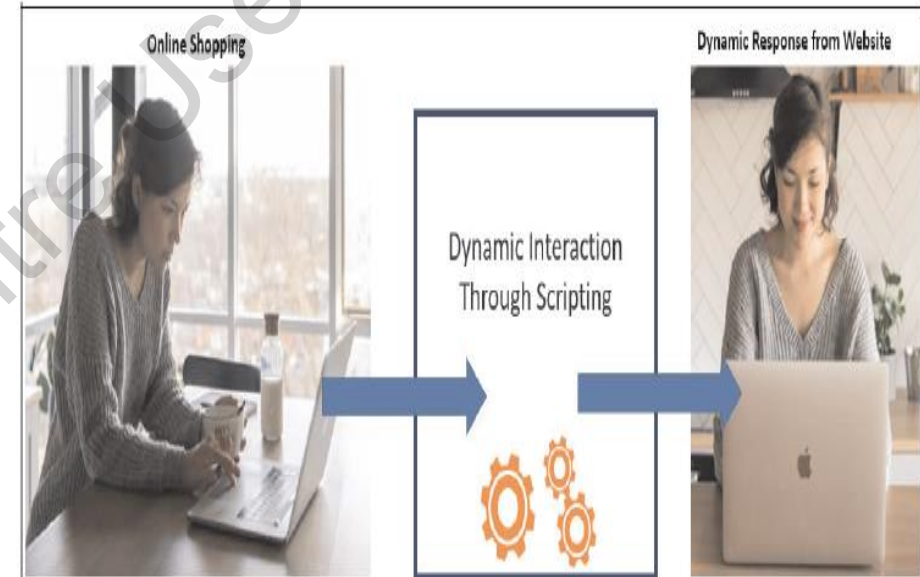
Consider an organization that provides a Website that allows its customers to view their products.

The company has received frequent customer feedbacks to provide the shopping facility online.

For example, details such as credit card number, email, and phone number entered by the customer must be in a proper format.

The developer can handle all these critical tasks by using a scripting language.

A scripting language refers to a set of instructions that provides some functionality when the user interacts with a Web page.



Scripting

Scripting refers to a series of commands that are interpreted and executed sequentially and immediately on occurrence of an event.

This event is an action generated by a user while interacting with a Web page.

Examples of events include button clicks, selecting a product from a menu, and so on.

A scripting language refers to a set of instructions that provides some functionality when the user interacts with a Web page.

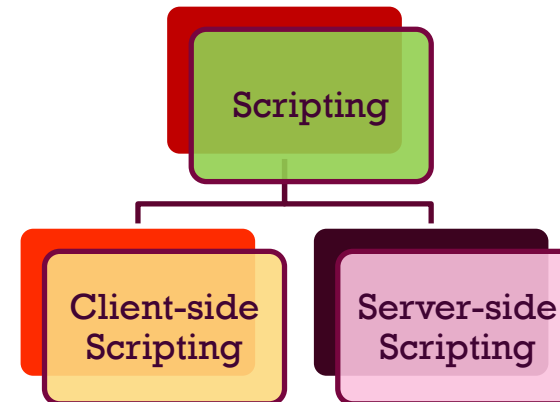
Scripting languages are often embedded in the HTML pages to change the behavior of the Web pages according to the user's requirements.

Client-side Scripting:

- Refers to a script being executed on the client's machine by the browser.

Server-side Scripting:

- Refers to a script being executed on a Web server to generate dynamic HTML pages.



JavaScript

JavaScript is a scripting language that allows building dynamic Web pages by ensuring maximum user interactivity.

JavaScript language is an object-based language, which means that it provides objects for specifying functionalities.

In real life, an object is a visible entity such as a car or a table having some characteristics and capable of performing certain actions.

Similarly, in a scripting language, an object has a unique identity, state, and behavior.

The identity of the object distinguishes it from other objects of the same type.

The state of the object refers to its characteristics, whereas the behavior of the object consists of its possible actions.

The object stores its identity and state in fields (also called variables) and exposes its behavior through functions (actions).

Versions of JavaScript

- The first version of JavaScript was developed by Brendan Eich at Netscape in 1995 and was named JavaScript 1.0.
- Following table lists various versions of JavaScript language:

Edition	Name	Description
1	ECMAScript 1 (1997)	First edition
2	ECMAScript 2 (1998)	Supported by Internet Explorer from version 4.0
3	ECMAScript 3 (1999)	Added regular expressions and try/catch and was supported by Internet Explorer 5.0, Netscape Navigator 4.0, and Opera 5.0 onwards
5	ECMAScript 5 (2009)	Added features such as 'strict mode' JSON support, String.trim(), Array.isArray(), and Array iteration methods. Is supported by Internet Explorer 6.0 and Mozilla Firefox 1.0 onwards
6	ECMAScript 2015	Was published in 2015 and added features such as let and const, default parameter values, Array.find(), and Array.findIndex()
7	ECMAScript 2016	Was published in 2016 and added exponential operator and Array.prototype.includes
8	ECMAScript 2017	Was published in 2017 and added features such as string padding, Object.entries, Object.values, async functions, and shared memory
9	ECMAScript 2018	Was published in 2018 and added features such as rest/spread properties, asynchronous iteration, and Promise.finally()
10	ECMAScript 2019	Was published in 2019 and added features such as Array.prototype.flat, Array.prototype.flatMap, and Object.fromEntries
11	ECMAScript 2020	Was published in June 2020 and introduces a BigInt primitive type for arbitrary-sized integers, nullish coalescing operator, and globalThis object
12	ECMAScript 2021	Was published in June 2021 and introduces replaceAll method for Strings, Promise.any, AggregateError, WeakRef, and other features

Client-side JavaScript

A Client-side JavaScript (CSJS) is executed by the browser on the user's workstation.

A client-side script might contain instructions for the browser to handle user interactivity.

These instructions might be to change the look or content of the Web page based on the user inputs.

Examples include displaying a welcome page with the user name, displaying date and time, validating that the required user details are filled, and so on.

A JavaScript is either embedded in an HTML page or is separately defined in a file, which is saved with .js extension.

In client-side scripting, when an HTML is requested, the Web server sends all the required files to the user's computer.

The Web browser executes the script and displays the HTML page to the user along with any tangible output of the script.

Server-side JavaScript

A Server-side JavaScript (SSJS) is executed by the Web server when an HTML page is requested by a user and the output is displayed by the browser.

A server-side JavaScript can interact with the database, fetch the required information specific to the user, and display it to the user.

Server-side scripting fulfills the goal of providing dynamic content in Web pages.

Unlike client-side JavaScript, HTML pages using server-side JavaScript are compiled into bytecode files on the server.

A JavaScript is either embedded in an HTML page or is separately defined in a file, which is saved with .js extension.

Compilation is a process of converting the code into machine-independent code.

This machine-independent code is known as the bytecode, which is an executable file that the Web server runs to generate the desired output.

<script> Tag

The <script> tag defines a script for an HTML page to make them interactive.

The browser that supports scripts interprets and executes the script specified under the <script> tag when the page loads in the browser.

You can directly insert a JavaScript code under the <script> tag.

You can define multiple <script> tags either in the <head> or in the <body> elements of an HTML page.

In HTML5, the type attribute specifying the scripting language is no longer required as it is optional.

- The Code Snippet demonstrates the use of the tag.

```
<!doctype html>
<html>
  <head>
    <script>
      document.write("Welcome to the Digital World");
    </script>
  </head>
  <body>
    .....
  </body>
</html>
```

There are two main purposes of the <script> tag, which are as follows:

Identifies a given segment of script in the HTML page.

Loads an external script file.

Variables in JavaScript

A variable refers to a symbolic name that holds a value, which keeps changing.

For example, age of a student and salary of an employee can be treated as variables.

In JavaScript, a variable is a unique location in computer's memory that stores a value and has a unique name.

The name of the variable is used to access and read the value stored in it.

A variable can store different types of data such as a character, a number, or a string.

Declaring Variables

- Declaring a variable refers to creating a variable by specifying the variable name.
- For example, one can create a variable named to store the name of a student.

In JavaScript,

the var keyword is used to create a variable by allocating memory to it.

a keyword is a reserved word that holds a special meaning.

the variable can be initialized at the time of creating the variable or later.

initialization refers to the task of assigning a value to a variable.

once the variable is initialized, you can change the value of a variable as required.

variables allow keeping track of data during the execution of the script.

one can declare and initialize multiple variables in a single statement.

while referring to a variable, you are referring to the value of that variable.

- Following syntax demonstrates how to declare variables in JavaScript:

Syntax:

```
var <variableName>;
```

where,

- **var**: Is the keyword in JavaScript.
- **variableName**: Is a valid variable name.

Variable Naming Rules

- JavaScript is a case-sensitive language.
- The variables X and x are treated as two different variables.
- JavaScript consists of certain rules for naming a variable as follows:

In JavaScript, a variable name

can consist of digits, underscore, and alphabets.

must begin with a letter or underscore character.

cannot begin with a number and cannot contain any punctuation marks.

cannot contain any kind of special characters such as +, *, %, and so on.

cannot contain spaces.

cannot be a JavaScript keyword.

Data Types in JavaScript 1-2

- To identify the type of data that can be stored in a variable, JavaScript provides different data types.
- Data types in JavaScript are classified into two broad categories namely, primitive and composite data types.
- Primitive data types contain only a single value, whereas the composite data types contain a group of values.

➤ PRIMITIVE DATA TYPES

- A primitive data type contains a single literal value such as a number or a string.
- A literal is a static value that you can assign to variables.
- Following table lists the primitive data types:

Primitive Data Type	Description
boolean	Contains only two values namely, true or false
null	Contains only one value namely, null. A variable of this value specifies that the variable has no value. This null value is a keyword and it is not the same as the value, zero
number	Contains positive and negative numbers and numbers with decimal point. Some of the valid examples include 6, 7.5, -8, 7.5e-3, and so on
string	Contains alphanumeric characters in single or double quotation marks. Single quotes are used to represent a string, which itself consists of quotation marks. A set of quotes without any characters within it is known as the null string

Data Types in JavaScript 2-2

➤ COMPOSITE DATA TYPES

- A composite data type stores a collection of multiple related values, unlike primitive data types.
- In JavaScript, all composite data types are treated as objects.
- A composite data type can be either predefined or user-defined in JavaScript.
- Following table lists the composite data types:

Composite Data Type	Description
Objects	Refers to a collection of properties and functions. Properties specify the characteristics and functions determine the behavior of a JavaScript object
Functions	Refers to a collection of statements, which are instructions to achieve a specific task
Arrays	Refers to a collection of values stored in adjacent memory locations

Methods

- JavaScript allows you to display information using the methods of the document object.
- The document object is a predefined object in JavaScript, which represents the HTML page and allows managing the page dynamically.
- Each object in JavaScript consists of methods, that fulfill a specific task.
- There are two methods of the document object, that display any type of data in the browser:
 - `write()`: Displays any type of data.
 - `writeln()`: Displays any type of data and appends a new line character.
- The syntax demonstrates the use of `document.write()` method, which allows you to display information in the displayed HTML page.

Syntax:

```
document.write("<data>" + variables);
```

where,

- **data**: Specifies strings enclosed in double quotes.
- **variables**: Specify variable names whose value should be displayed on the HTML page.

Using Comments

- Comments provide information about a piece of code in the script.
- When the script is executed, the browser identifies comments as they are marked with special characters and does not display them.
- JavaScript supports two types of comments. These are as follows:

➤ SINGLE LINE COMMENTS

- Single-line comments begin with two forward slashes (//). You can insert single-line comments as follows:

```
// This statement declares a variable named num.  
var num;
```

➤ MULTI-LINE COMMENTS

- Multi-line comments begin with a forward slash followed by an asterisk (/*) and end with an asterisk followed by a forward slash (*).
- You can insert multiple lines of comments as follows:

```
/* This line of code  
declares a variable */  
var num;
```


Escape Sequence Characters

- An escape sequence character is a special character that is preceded by a backslash (\).
- Escape sequence characters are used to display special non-printing characters such as a tab space, a single space, or a backspace.
- In JavaScript, the escape sequence characters must be enclosed in double quotes.
- Following table lists the escape sequence characters:

Escape Sequence	Non-Printing Character
<code>\b</code>	Back space
<code>\f</code>	Form feed
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal tab
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\\</code>	Backslash
<code>\\aaa</code>	Matches a Latin-1 encoding character using octal representation, where aaa are three octal numbers. For example, <code>\251</code> represents the copyright symbol
<code>\\xaa</code>	Matches a Latin-1 encoding character using hexadecimal representation, where aa are two hexadecimal numbers. For example, <code>\x61</code> represents the character 'a'
<code>\\uaaaa</code>	Represent the Unicode encoding character, where aaaa are four hexadecimal numbers. For example, the character <code>\u0020</code> represents a space

Built-in Functions

- A function is a piece of code that performs some operations on variables to fulfill a specific task.
- It takes one or more input values, processes them, and returns an output value.
- Following table lists the built-in JavaScript functions:

Function	Description	Example
<code>alert()</code>	Displays a dialog box with some information and OK button	<code>alert("Please fill all the fields of the form");</code> Displays a message box with the instruction
<code>confirm()</code>	Displays a dialog box with OK and Cancel buttons. It verifies an action, which a user wants to perform	<code>confirm("Are you sure you want to close the page?");</code> Displays a message box with the question
<code>parseInt()</code>	Converts a string value into a numeric value	<code>parseInt("25 years");</code>
<code>parseFloat()</code>	Converts a string into a number with decimal point	<code>parseFloat("10.33");</code> Returns 10.33
<code>eval()</code>	Evaluates an expression and returns the evaluated result	<code>eval("2+2");</code> Returns 4
<code>isNaN()</code>	Checks whether a value is not a number	<code>isNaN("Hello");</code> Returns true
<code>prompt()</code>	Displays a dialog box that accepts an input value through a text box. It also accepts the default value for the text box	<code>prompt("Enter your name", "Name");</code> Displays the message in the dialog box and Name in the text box.

Events

- An event occurs when a user interacts with the Web page.
- Some of the commonly generated events are mouse clicks, key strokes, and so on.
- The process of handling these events is known as event handling.
- Following figure displays the event:



Event Handling

Event handling is a process of specifying actions to be performed when an event occurs. This is done by using an event handler.

An event handler is a scripting code or a function that defines the actions to be performed when the event is triggered.

When an event occurs, an event handler function that is associated with the specific event is invoked.

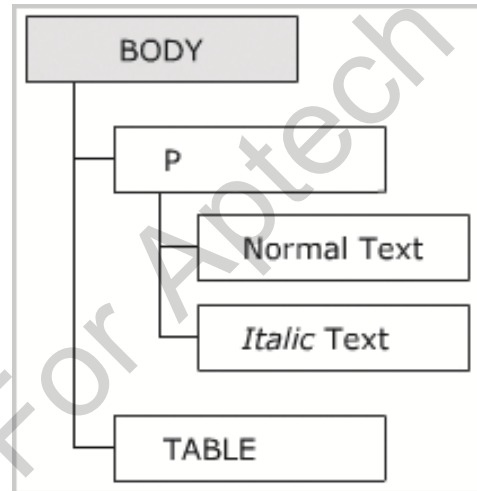
The information about this generated event is updated on the `event` object.

The `event` object is a built-in object, which can be accessed through the `window` object.

It specifies the event state, including information such as the location of mouse cursor, element on which an event occurred, and state of the keys in a keyboard.

Event Bubbling

- Event bubbling is a mechanism that allows a user to specify a common event handler for all child elements.
- This means that the parent element handles all the events generated by the child elements.
- For example, consider a Web page that consists of a paragraph and a table. The paragraph consists of multiple occurrences of italic text.
- To change the color of each italic text of a paragraph when the user clicks a particular button, instead of declaring an event handler for each italic text, one can declare it within the `<P>` element.
- Following figure displays the event bubbling:



Life Cycle of an Event

- An event's life starts when the user performs an action to interact with the Web page.
- It finally ends when the event handler provides a response to the user's action.
- Steps involved in the life cycle of an event are as follows:

1. The user performs an action to raise an event.

2. The event object is updated to determine the event state.

3. The event is fired.

4. The event bubbling occurs as the event bubbles through the elements of the hierarchy.

5. The event handler is invoked that performs the specified actions.

Keyboard Events

- Keyboard events are the events that occur when a key or a combination of keys are pressed or released from a keyboard.
- These events occur for all keys of a keyboard.
- Different keyboard events are as follows:

Onkeydown

- Occurs when a key is pressed down.

Onkeyup

- Occurs when the key is released.

Onkeypress

- Occurs when a key is pressed and released.

Mouse Events

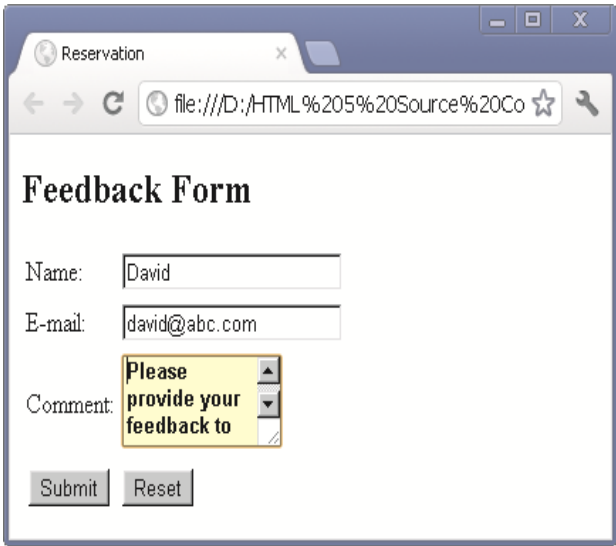
- Mouse events occur when the user clicks the mouse button.
- Following table lists the mouse events:

Event	Description
onmousedown	Occurs when the mouse button is pressed
onmouseup	Occurs when the mouse button is released
onclick	Occurs when the mouse button is pressed and released
ondblclick	Occurs when the mouse button is double-clicked
onmousemove	Occurs when the mouse pointer is moved from one location to other
onmouseover	Occurs when the mouse pointer is moved over the element
onmouseout	Occurs when the mouse pointer is moved out of the element

Focus and Selection Events

- The focus events determine the activation of various elements that uses the element.
- It allows a user to set or reset focus for different elements.
- The selection events occur when an element or a part of an element within a Web page is selected.
- Following table lists the focus and selection events:

Data Type	Description
onfocus	Occurs when an element receives focus
onblur	Occurs when an element loses focus
onselectstart	Occurs when the selection of an element starts
onselect	Occurs when the present selection changes
ondragstart	Occurs when the selected element is moved



jQuery

- Following are key features supported by jQuery:

Event Handling

jQuery has a smart way to capture a wide range of events, such as user clicks a link, without making the HTML code complex with event handlers.

Animations

jQuery has many built-in animation effects that the user can use while developing their Websites.

DOM Manipulation

jQuery easily selects, traverses, and modifies DOM by using the cross-browser open source selector engine named Sizzle.

Cross Browser Support

jQuery has a support for cross-browser and works well with following browsers:

- Internet Explorer 6 and above
- Firefox 2.0 and above
- Safari 3.0 and above
- Chrome
- Opera 9.0 and above

Lightweight

jQuery has a lightweight library of 19 KB size.

AJAX Support

jQuery helps you to develop feature-rich and responsive Web sites by using AJAX technologies.

Latest Technology

jQuery supports basic XPath syntax and CSS3 selectors.

Using jQuery Library

- To work with jQuery perform following steps:

1. Download the jQuery library from the <http://jquery.com/> Website.

2. Place the jquery-1.7.2.min.js file in the current directory of the Website.

- The user can include jQuery library in their file.
- The Code Snippet shows how to use a jQuery library.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>The jQuery Example</title>
    // Using jQuery library
    <script src="jquery-1.7.2.min.js">
      // The user can add our JavaScript code here
    </script>
  </head>
  <body>
  </body>
</html>
```

Calling jQuery Library Functions

- While jQuery is reading or manipulating the Document Object Model (DOM) object, the users can add the events when the DOM object is ready.
- If the user wants the event on their page then, the user has to call the event in the `$(document).ready()` function.
- Users also register the ready event for the document.
- Place the `jquery-1.7.2.min.js` file in the current directory and specify the location of this file in the `src` attribute.
- The Code Snippet shows how to call jQuery library function and ready event in DOM.

jQuery Mobile

- jQuery mobile is a Web User Interface (UI) development framework that allows the user to build mobile Web applications that work on tablets and smartphones.
- The jQuery mobile framework provides many facilities that include XML DOM and HTML manipulation and traversing, performing server communication, handling events, image effects, and animation for Web pages.
- Basic features of jQuery mobile are as follows:

Simplicity	Accessibility	Enhancements and Degradation	Themes	Smaller Size
This framework is easy to use and allows developing Web pages by using markup driven with minimum or no JavaScript.	The framework supports Accessible Rich Internet Applications (ARIA) that helps to develop Web pages accessible to visitors with disabilities.	The jQuery mobile is influenced by the latest HTML5, JavaScript, and CSS3.	This framework provides themes that allow the user to provide their own styling.	The size for jQuery mobile framework is smaller for CSS, it is 6 KB and for JavaScript library it is 12 KB.

Introduction to JavaScript Operators

An operator specifies the type of operation to be performed on the values of variables and expressions.

JavaScript provides different types of operators to perform simple to complex calculations and evaluations.

Certain operators are also used to construct relational and logical statements. These statements allow implementing decision and looping constructs.

Basics of Operators

An operation is an action performed on one or more values stored in variables.

The specified action either changes the value of the variable or generates a new value.

An operation requires minimum one symbol and some value.

Symbol is called an operator and it specifies the type of action to be performed on the value.

Value or variable on which the operation is performed is called an operand.

- Three main types of operators are as follows:

Unary operators - Operates on a single operand.

Binary operators - Operates on two operands.

Ternary operators - Operates on three operands.

Operators and their Types

Operators help in simplifying expressions.

JavaScript provides a predefined set of operators that allow performing different operations.

JavaScript operators are classified into six categories based on the type of action they perform on operands.

- Six categories of operators are as follows:
 - Arithmetic operators
 - Relational operators
 - Logical operators
 - Assignment operators
 - Bitwise operators
 - Special operators

Arithmetic Operators

Are binary operators.

Perform basic arithmetic operations on two operands.

Operator appears in between the two operands, which allow you to perform computations on numeric and string values.

- Following table lists arithmetic operators:

Arithmetic Operator	Description	Example
+ (Addition)	Performs addition. In case of string values, it behaves as a string concatenation operator and appends a string at the end of the other	45 + 56
- (Subtraction)	Performs subtraction. If a larger value is subtracted from a smaller value, it returns a negative numeric value	76-78
/ (Division)	Divides the first operand by the second operand and returns the quotient	24 / 8
% (Modulo)	Divides the first operand by the second operand and returns the remainder	90 % 20
* (Multiplication)	Multiplies the two operands	98 * 10

Increment and Decrement Operators

Increment and decrement operators are unary operators.

Increment operator (++) increases the value by 1, while the decrement operator (--) decreases the value by 1.

These operators can be placed either before or after the operand.

Operator if placed before the operand, expression is called pre-increment or pre-decrement. Operator if placed after the operand, expression is called post-increment or post-decrement.

- Following table lists increment and decrement operators:

Expressions	Type	Result
numTwo = ++numOne;	Pre-increment	numTwo = 3
numTwo = numOne++;	Post-increment	numTwo = 2
numTwo = --numOne;	Pre-decrement	numTwo = 1
numTwo = numOne--;	Post-decrement	90 % 20

Relational Operators

Are binary operators that make a comparison between two operands.

After making a comparison, they return a boolean value namely, true or false.

Expression consisting of a relational operator is called as the relational expression or conditional expression.

- Following table lists relational operators:

Relational Operators	Description	Example
== (Equal)	Verifies whether the two operands are equal	90 == 91
!= (Not Equal)	Verifies whether the two operands are unequal	99 != 98
=== (Strict Equal)	Verifies whether the two operands are equal and are of the same type	numTwo = 1
!== (Strict Not Equal)	Verifies whether the two operands are unequal and whether are not of the same type	90 % 20
> (Greater Than)	Verifies whether the left operand is greater than the right operand	97 > 95
< (Less Than)	Verifies whether the left operand is less than the right operand	94 < 96
>= (Greater Than or Equal)	Verifies whether the left operand is greater than or equal to the right operand	92 >= 93
<= (Less Than or Equal)	Verifies whether the left operand is less than or equal to the right operand	99 <= 100

Logical Operators

Are binary operators that perform logical operations on two operands.

They belong to the category of relational operators, as they return a boolean value.

- Following table lists the logical operators:

Logical Operators	Description	Example
&& (AND)	Returns true, if either of the operands are evaluated to true. If first operand evaluates to true, it will ignore the second operand	(x == 2) && (y == 5) Returns false
! (NOT)	Returns false, if the expression is true and vice-versa	!(x == 3) Returns true
(OR)	Returns true, if either of the operands are evaluated to true. If first operand evaluates to true, it will ignore the second operand	(x == 2) (y == 5) Returns true

Assignment Operators

Assignment operators assign the value of the right side operand to the operand on the left side by using the equal to operator (=).

Simple assignment operator - Is the '=' operator which is used to assign a value or result of an expression to a variable.

Compound assignment operator - Is formed by combining the simple assignment operator with the arithmetic operators.

- Following table lists the assignment operators:

Expressions	Description	Example
<code>numOne += 6;</code>	<code>numOne = numOne + 6</code>	<code>numOne = 12</code>
<code>numOne -= 6;</code>	<code>numOne = numOne - 6</code>	<code>numOne = 0</code>
<code>numOne *= 6;</code>	<code>numOne = numOne * 6</code>	<code>numOne = 36</code>
<code>numOne %= 6;</code>	<code>numOne = numOne % 6</code>	<code>numOne = 0</code>
<code>numOne /= 6;</code>	<code>numOne = numOne / 6</code>	<code>numOne = 1</code>

Bitwise Operators

Represent their operands in bits (zeros and ones) and perform operations on them.

They return standard decimal values.

Compound assignment operator - Is formed by combining the simple assignment operator with the arithmetic operators.

- Following table lists the bitwise operators in JavaScript:

Bitwise Operators	Description	Example
& (Bitwise AND)	Compares two bits and returns 1 if both of them are 1 or else returns 0	00111000 Returns 00011000
~ (Bitwise NOT)	Inverts every bits of the operand and is a unary operator	~00010101 Returns 11101010
(Bitwise OR)	Compares two bits and returns 1 if the corresponding bits of either or both the operands is 1	00111000 Returns 00111100
^ (Bitwise XOR)	Compares two bits and returns 1 if the corresponding bit of either, but not both the operands is 1	a ^ b Returns 7 (00000111)

Special Operators

There are some operators in JavaScript which do not belong to any of the categories of JavaScript operators.

Such operators are referred to as the special operators.

- Following table lists the special operators in JavaScript:

Special Operators	Description
, (comma)	Combines multiple expressions into a single expression, operates on them in the left to right order and returns the value of the expression on the right.
?: (conditional)	Operates on three operands where the result depends on a condition. It is also called as ternary operator and has the form condition, ? value1:value2. If the condition is true, the operator obtains value1 or else obtains value2.
typeof	Returns a string that indicates the type of the operand. The operand can be a string, variable, keyword, or an object.

Operator Precedence

- Following table lists the precedence of the operators from the highest to the lowest and their associativity:

Precedence Order	Operator	Description	Associativity
1	()	Parentheses	Left to Right
2	++, --	Post-increment and Post-decrement operators	Not Applicable
3	typeof, ++, --, -, ~, !	Pre-increment and Pre-decrement operators, Logical NOT, Bitwise NOT, and Unary negation	Right to Left
4	*, /, %	Multiplication, Division, and Modulo	Left to Right
5	+, -	Addition and Subtraction	Left to Right
6	<, <=, >, >=	Less than, Less than or equal, Greater than, and Greater than or equal	Left to Right
7	==, ===, !=, !==	Equal to, Strict equal to, Not equal to, and Strict not equal to	Left to Right
8	&, , ^, &&,	Bitwise AND, Bitwise OR, Bitwise XOR, Logical AND, and Logical OR	Left to Right
9	?:	Conditional operator	Right to Left
10	=, +=, -=, *=, /=, %=	Assignment operators	Right to Left
11	,	Comma	Left to Right

Regular Expressions

Is a pattern that is composed of set of strings, which is to be matched to a particular textual content.

Allow handling of textual data effectively as it allows searching and replacing strings.

Allows handling of complex manipulation and validation, which could otherwise be implemented through lengthy scripts.

- There are two ways to create regular expressions which are as follows:

Literal Syntax:

- Refers to a static value
 - Allows specifying a fixed pattern, which is stored in a variable
- Syntax is as follows:
- ```
var variable_name = /regular_expression_pattern/;
```

## RegExp() Constructor:

- Is useful when the Web page designer does not know the pattern at the time of scripting.
  - Method dynamically constructs a regular expression when the script is executed.
- Syntax is as follows:
- ```
var variable_name = new RegExp("regular_expression_pattern", "flag");
```

RegExp Methods and Properties

- RegExp object supports methods that are used for searching the pattern in a string. They are as follows:

test(string) - Tests a string for matching a pattern and returns a Boolean value of true or false. The Boolean value indicates whether the pattern exists or not in the string. The method is commonly used for validation.

exec(string) - Executes a string to search the matching pattern within it. The method returns a null value, if pattern is not found. In case of multiple matches, it returns the matched result set.

Audio Attributes	Description
\$n	Represents the number from 1 to 9. It stores the recently handled parts of a parenthesized pattern of a regular expression.
aif	Indicates whether the given regular expression contain a g flag. The g flag specifies that all the occurrences of a pattern will be searched globally, instead of just searching for the first occurrence.
aifc	Indicates whether the given regular expression contains an i flag.
aiff	Stores the location of the starting character of the last match found in the string. In case of no match, the value of the property is -1.
asc	Stores the copy of the pattern.

Categories of Pattern Matching

- Different categories of pattern matching character that are required to create a regular expression pattern are as follows:

Position
Matching

Character
Classes

Repetition

Alternation
and Grouping

Back
Reference

Position Matching

Characters or symbols in this category allow matching a substring that exists at a specific position within a string.

- Following table lists various position matching symbols:

Symbol	Description	Example
<code>^</code>	Denotes the start of a string	<code>/^Good/</code> matches “Good” in “Good night”, but not in “A Good Eyesight”
<code>\$</code>	Denotes the end of a string	<code>/art\$/</code> matches “art” in “Cart” but not in “artist”
<code>\b</code>	Matches a word boundary. A word boundary includes the position between a word and the space	<code>/ry\b/</code> matches “ry” in “She is very good”
<code>\B</code>	Matches a non-word boundary	<code>/\Ban/</code> matches “an” in “operand” but not in “anomaly”

Character Classes

Characters or symbols in this category are combined to form character classes for specifying patterns.

These classes are formed by placing a set of characters within the square brackets.

- Following table lists various character classes symbols:

Symbol	Description	Example
[xyz]	Matches one of the characters specified within the character set	/^Good/ matches “Good” in “Good night”, but not in “A Good Eyesight”
[^xyz]	Matches one of the characters not specified within the character set	/[^BC]RT/ Matches “RRT” but, not “BRT” or “CRT”
.	Denotes a character except for the new line and line terminator	/s.t/ Matches “sat”, “sit”, “set”, and so on
\w	Matches alphabets and digits along with the underscore	/\w/ Matches “600” in “600%”
\W	Matches a non-word character	/\W/ Matches “%” in “800%”
\d	Matches a digit between 0 to 9	/\d/ Matches “4” in “A4”
\D	Searches for a non-digit	/\D/ Matches “ID” in “ID 2246”
\s	Searches any single space character including space, tab, form feed, and line feed	/\s\w*/ Matches “ bar” in “scroll bar”
\S	Searches a non-space character	/\S\w*/ Matches “scroll” in “scroll bar”

Repetition

Characters or symbols in this category allow matching characters that reappear frequently in a string.

- Following table lists various repetition matching symbols:

Symbol	Description	Example
{x}	Matches x number of occurrences of a regular expression	<code>/\d{6}/</code> Matches exactly six digits
{x,}	Matches either x or additional number of occurrences of a regular expression	<code>/\s{4,}/</code> Matches minimum four whitespace characters
{x,y}	Matches minimum x to maximum y occurrences of a regular expression	<code>/\d{6,8}/</code> Matches minimum six to maximum eight digits
?	Matches minimum zero to maximum one occurrences of a regular expression	<code>/\s?m/</code> Matches “lm” or “l m”
*	Matches minimum zero to multiple occurrences of a regular expression	<code>/im*/</code> Matches “i” in “Ice” and “imm” in “immaculate”, but nothing in “good”

Alternation and Grouping

Characters or symbols in this category allow grouping characters as an individual entity or adding the 'OR' logic for pattern matching.

- Following table lists various alternation and grouping character symbols:

Symbol	Description	Example
()	Organizes characters together in a group to specify a set of characters in a string	<code>/(xyz)+(uvw)/</code> Matches one or more number of occurrences of "xyz" followed by one occurrence of "uvw"
	Combines sets of characters into a single regular expression and then matches any of the character set	<code>/(xy) (uv) (st)/</code> Matches "xy" or "uv" or "st"

Back References

Characters or symbols in this category allow grouping characters as an individual entity or adding the 'OR' logic for pattern matching.

- Following table lists various alternation and grouping character symbols:

Symbol	Description	Example
()\n	Matches a parenthesized set within the pattern, where n is the number of the parenthesized set to the left	<div>/(\\w+)\s+\\1/</div> <div>Matches any word occurring twice in a line, such as “hello hello”. The \\1 specifies that the word following the space should match the string, which already matched the pattern in the parentheses to the left of the pattern. To refer to more than one set of parentheses in the pattern, you would use \\2 or \\3 to match the appropriate parenthesized clauses to the left. You can have maximum nine back references in the pattern</div>

Decision-making Statements

Statements are referred to as a logical collection of variables, operators, and keywords that perform a specific action to fulfill a required task.

Statements help you build a logical flow of the script.

In JavaScript, a statement ends with a semicolon.

JavaScript is written with multiple statements, wherein the related statements are grouped together are referred to as block of code and are enclosed in curly braces.

Decision-making statements allow implementing logical decisions for executing different blocks to obtain the desired output.

They execute a block of statements depending upon a Boolean condition that returns either true or false.

if-else-if Statement

Allows you to check multiple conditions and specify a different block to be executed for each condition.

Flow of these statements begins with the if statement followed by multiple else if statements and finally by an optional else block.

Entry point of execution in these statements begins with the if statement.

If the condition in the if statement is false, the condition in the immediate else if statement is evaluated.

Also referred to as the if-else-if ladder.

Nested if Statement

Comprises multiple if statements within an if statement.

Flow of the nested-if statements starts with the if statement, which is referred to as the outer if statement.

Outer if statement consists of multiple if statements, which are referred to as the inner if statements.

Inner if statements are executed only if the condition in the outer if statement is true.

Each of the inner if statements is executed but, only if the condition in its previous inner if statement is true.

switch-case Statement

A program becomes quite difficult to understand when there are multiple if statements.

To simplify coding and to avoid using multiple if statements, switch-case statement can be used.

switch-case statement allows comparing a variable or expression with multiple values.

Introduction to Loops

Loops allow you to execute a single statement or a block of statements multiple times.

They are widely used when you want to display a series of numbers and accept repetitive input.

A loop construct consists of a condition that instructs the compiler the number of times a specific block of code will be executed.

If the condition is not specified within the construct, the loop continues infinitely. Such loop constructs are referred to as infinite loops.

- JavaScript supports three types of loops that are as follows:
 - `while` **Loop**
 - `for` **Loop**
 - `do-while` **Loop**

while Loop

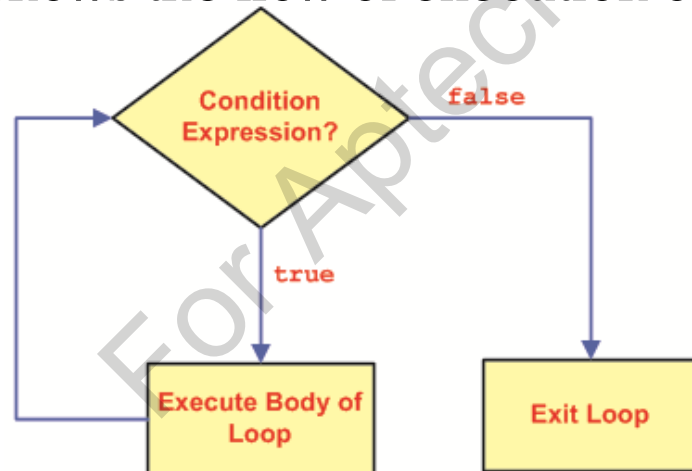
The `while` loop executes a block of code as long as the given condition remains `true`.

The while loop begins with the `while` keyword, which is followed by parentheses containing a `boolean` condition.

If this condition returns `true`, the block of statements within the while loop are executed.

Once the condition becomes `false`, the while statement stops the execution of loop and transfers the control to next statement appearing after the block.

- Following figure shows the flow of execution of the `while` loop:



for Loop

The `for` loop is similar to the `while` loop as it executes the statements within the loop as long as the given condition is true.

Unlike the `while` loop, the `for` loop specifies the loop control statements at the top instead in the body of the loop.

The `for` loop begins with the `for` keyword, which is followed by parentheses containing three expressions, each of which are separated by a semicolon.

The three expressions are referred to as **initialization expression**, **condition expression**, and **increment/decrement expression** respectively.

do-while Loop

The do-while loop is similar to the `while` loop. This is because both the do-while and `while` loops execute until the condition becomes false.

However, the do-while loop differs by executing the body of the loop at least once before evaluating the condition even if the condition is false.


The do-while loop starts with the `do` keyword and is followed by a block of statements.

At the end of the block, the `while` keyword is specified that is followed by parentheses containing the condition.

When the condition returns false, the block of statements after the `do` keyword are ignored and the next statement following the `while` statement is executed.

Jump Statements

break Statement



The `break` statement can be used with decision-making such as `switch-case` and loop constructs such as `for` and `while` loops.

The `break` statement is denoted by using the `break` keyword. It is used to exit the loop without evaluating the specified condition.

The control is then passed to the next statement immediately after the loop.

continue Statement



The `continue` statement is mostly used in the loop constructs and is denoted by the `continue` keyword.

It is used to terminate the current execution of the loop and continue with the next repetition by returning the control to the beginning of the loop.

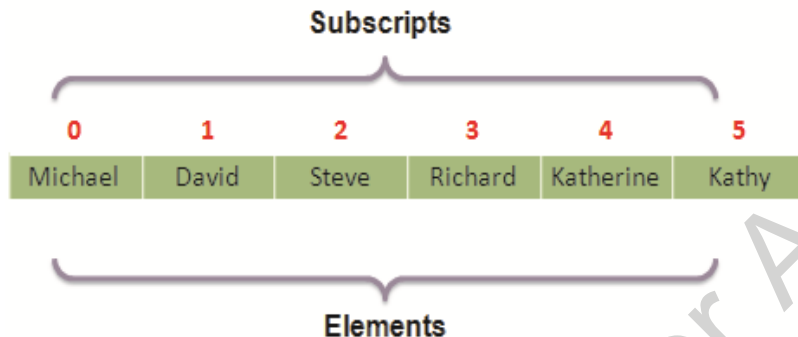
This means, the `continue` statement will not terminate the loop entirely, but terminates the current execution.

Arrays

- An array is a collection of values stored in adjacent memory locations.
- These array values are referenced using a common array name and must be of the same data type. These values can be accessed by using subscript or index numbers.
- JavaScript supports two types of arrays:
 - Single-dimensional array
 - Multi-dimensional array

Single-dimensional Array

In a single-dimensional array, the elements are stored in a single row in the allocated memory.



Multi-dimensional Array

- A multi-dimensional array stores a combination of values of a single type in two or more dimensions.
- These dimensions are represented as rows and columns similar to those of a Microsoft Excel sheet.

Employee Salaries →	0 BASIC	1 HRA	2 ALLOWANCE	3 TOTAL
↓				
0	14350	10500	1500	26350
1	34350	4050	1000	39400
2	6150	4500	3250	13900
3	4920	4500	2250	11670
4	12300	9000	2000	23300

Accessing Single-dimensional Arrays

➤ **Accessing Array Elements Without Loops**

➤ **Accessing Array Elements With Loops**

Array Methods

An array is a set of values grouped together and identified by a single name. In JavaScript, the `Array` object allows you to create arrays.

It provides the `length` property that allows you to determine the number of elements in an array.

various methods of the `Array` object allow to access and manipulate the array elements.

for..in Loop

The `for..in` loop is an extension of the `for` loop. It enables to perform specific actions on the arrays of objects.

The loop reads every element in the specified array and executes a block of code only once for each element in the array.

Syntax:

```
for (variable_name in array_name)
{
    //statements;
}
```

where,

- `variable_name`: Is the name of the variable.
- `array_name`: Is the array name.



Summary 1-2

- ❖ Scripting refers to a series of commands that are interpreted and executed sequentially and immediately on an occurrence of an event.
- ❖ JavaScript is a scripting language, which can be executed on the client-side and on the server-side.
- ❖ A variable refers to a symbolic name that holds a value, which keeps changing.
- ❖ A primitive data type contains a single literal value such as a number or a string.
- ❖ A function is a piece of code that performs some operations on variables to fulfill a specific task.
- ❖ Event handling is a process of specifying actions to be performed when an event occurs.
- ❖ Event bubbling is a mechanism that allows you to specify a common event handler for all child elements.
- ❖ jQuery mobile is a Web User Interface development framework that allows the user to build mobile Web applications that works on tablets and smartphones.
- ❖ An operator specifies the type of operation to be performed on the values of variables and expressions.
- ❖ JavaScript operators are classified into six categories based on the type of action they perform on operands.

Summary 2-2

- ❖ There are six category of operators namely, Arithmetic, Relational, Logical, Assignment, Bitwise, and Special operators.
- ❖ Operators in JavaScript have certain priority levels based on which their execution sequence is determined.
- ❖ A regular expression is a pattern that is composed of set of strings, which is to be matched to a particular textual content.
- ❖ In JavaScript, there are two ways to create regular expressions namely, literal syntax and RegExp() constructor.
- ❖ Decision-making statements allow implementing logical decisions for executing different blocks to obtain the desired output.
- ❖ A loop construct consists of a condition that instructs the compiler the number of times a specific block of code will be executed.
- ❖ JavaScript supports three types of loops that include: while loop, for loop, and do-while loop.
- ❖ The break statement is used to exit the loop without evaluating the specified condition.
- ❖ The continue statement terminates the current execution of the loop and continue with the next repetition by returning the control to the beginning of the loop.
- ❖ JavaScript supports two types of arrays namely, Single-dimensional array and Multi-dimensional array.
- ❖ The for..in loop is an extension of the for loop that enables to perform specific actions on the arrays of objects.