Programming with JavaScript

Programming with JavaScript

Trainer's Guide

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2019

Aptech®
Unleash your potential

Aptech®
COMPUTER EDUCATION
Unleash your potential

Preface

The book 'Programming with JavaScript' covers various essential concepts in JavaScript such as pattern matching, preprocessing, grouping scripts, minification, and more.

The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer's Guide will provide guidance on the flow of the session and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 Certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team

Table of Contents

Sessions

Session 1 – Introduction to JavaScript

1.1 Pre-Class Activities

Before you commence the session, conduct an icebreaker.

Do: Start this session by playing a small game soon after the students assemble in the classroom.

Give 30 minutes for this activity.

This is a warm up activity that will help understand the students' views and thoughts to make the environment friendlier.

Ask the students to introduce themselves and ask the students what they like most about programming or working with computers. For example, they may answer, I am Samara. When I was in high school, I learnt C++ coding; that was my first programming experience. It was amazing when I created simple games and I fell in love with programming.

Say (After the introductions are done with): Let us move on to the Session objectives.

1.1.1 Objectives

By the end of this session, the learners will be able to:

- Define JavaScript
- Differentiate between JavaScript and Java
- Describe Web Applications and Websites
- Compare and contrast between server-side and client-side scripting
- Explain basic concepts of JavaScript

1.1.2 Teaching Skills

To teach this session successfully, you must know about HTML, Document Object Model, CSS, and basics of JavaScript. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

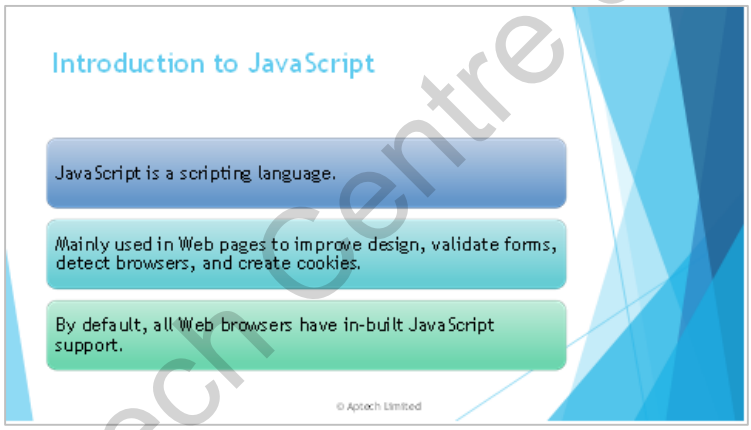
Follow the order given here during In-Class activities.

1.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 4 of the presentation. Tell them that they will be introduced to JavaScript, differences between JavaScript, and Java. They will learn about Web applications and Websites, and compare and contrast between server-side and client-side scripting. They will understand basic concepts of JavaScript.

1.2 In-Class Explanations

Slide 5
Understand HTML, CSS, and JavaScript.



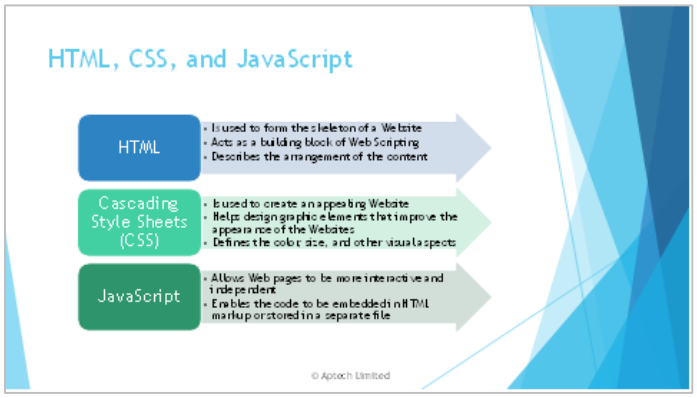
Ask the students, “What is JavaScript?”

Do: Give some time for them to answer. Consolidate the responses.

Say: Tell the students that JavaScript is a scripting language used in Websites. It is mainly used in Web pages to improve the design, validate forms, detect browsers, and create cookies. By default, all - Web browsers have built-in JavaScript support.

Slide 6

Understand JavaScript.



Ask: Do you know what HTML stands for?

Do: Wait for response. Consolidate the responses.

Say: HTML stands for Hyper Text Markup Language.

Ask: Do you know that HTML, Cascading Style Sheets (CSS), and JavaScript can work together?

Do: Wait for response. Consolidate the responses.

Discuss the difference between HTML and CSS.

Say:

HTML is often used to form the skeleton of a Website. HTML is the fundamental building block of Web Scripting and provides a skeletal frame for a Website. It describes an arrangement of the content.

These sites can be further enhanced by using CSS and JavaScript. CSS helps to design graphic elements that improve the appearance of Websites. It helps in defining the color, size, and other visual aspects of Web pages with advanced technology. It is an important tool to create an appealing Website.

JavaScript allows Web pages to be more interactive and independent of a Web server.

The JavaScript code can be embedded in HTML markup or stored in a separate file with a .js extension.

When used together, HTML, CSS, and JavaScript can render visually rich, efficient, and interactive sites. Shown here is an example of a Website created using JavaScript and CSS. Combination of all three provides a better user experience than a static Website. Briefly highlight various features of each of the versions.

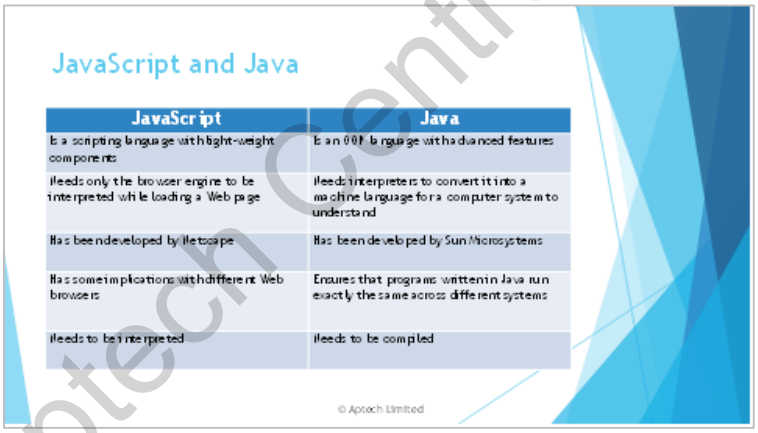
Slide 7

Understand the usage of JavaScript and CSS using an example Website.

Do: Show the picture and tell them that this is an example of a Website created using JavaScript and CSS.

Slide 8

Understand the difference between the JavaScript and Java.

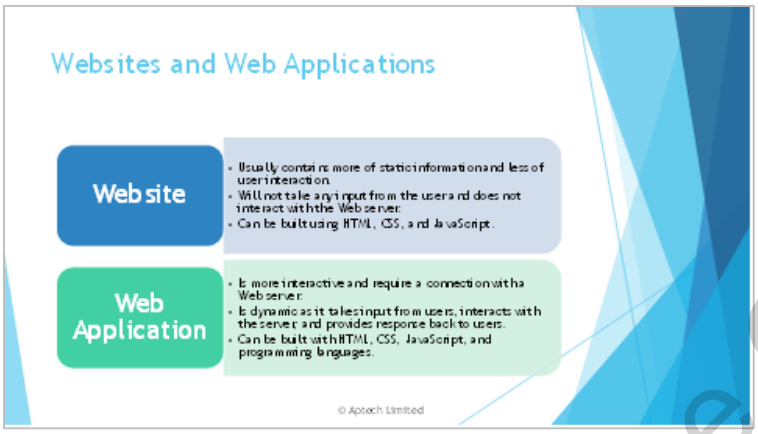


Say: Java is one of the most popular high-level scripting languages used by developers worldwide. JavaScript and Java are two different languages.

Do: Show the table on the screen and explain the differences between Java and JavaScript.

Slide 9

Understand the features of Websites and Web Application.



Ask the students, can you recall from an earlier slide, what is JavaScript?

Do: Give some time for them to answer. Consolidate the responses.

Say: JavaScript is a scripting language used in Websites to improve the design, validate forms, detect browsers, and create cookies.

Do: Explain that a Website:

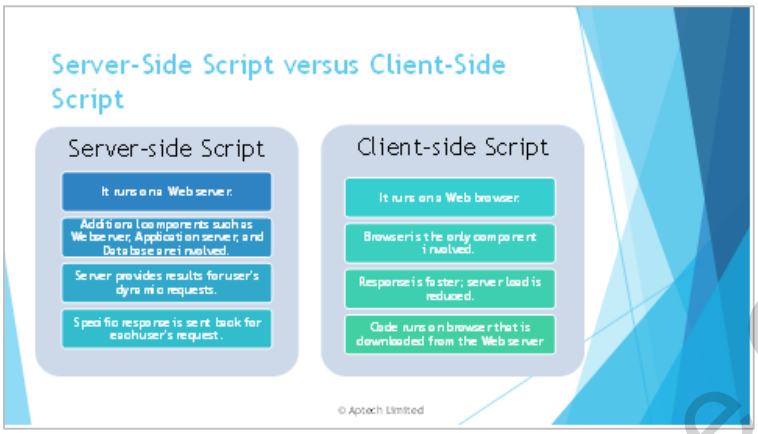
- Usually contains more of static information and less of user interaction. Will not take any input from the user and does not interact with the Web server if created as static Website. Web applications can be built using HTML, CSS, and JavaScript and does not require programming languages or database.
- Can be built using HTML, CSS, and JavaScript and does not require programming languages or database.

Do: Explain that a Web application:

- Is more interactive and requires a connection with a Web server, similar to desktop applications.
- Is dynamic as it takes input from users, interacts with the server, and provides response back to users.
- Can be built with HTML, CSS, JavaScript, and programming languages such as Java, PHP, Ruby, or Python.

Slide 10

Understand the difference between server-side script and client-side script.



Do: Compare and contrast the server-side script and client-side script.

Explain the following points:

Server-side script:

- A server-side script runs on a Web server that takes input from the client and interacts with it.
- There are additional components involved in a server-side scripting. This includes Web server, Application server, and Database (DB).
- The server provides results for the user's dynamic requests and displays the results in the Web browser.
- Specific response is sent back for each user's (client's) request to the browser.

Client-side script:

- A client-side script runs on a Web browser that acts as an environment, and the processing of the script happens at the end user's computer.
- Browser is the only component involved in client-side script.
- As the processing happens on the client-side, the response is faster. The server load is reduced saving time significantly for the client requests.
- The code runs on the browser that is downloaded from the Web server to the user's computer while accessing the Webpage.

Do: Explain how server-side and client-side scripts process requests from a user using certain steps.

Say: Server-side scripts process requests from a user using certain steps:

- The user makes a request from the browser.
- The request is processed by the Web server.
- The Web server passes the request to DB.
- The DB is queried, and the result is sent back to the server.

- The output is displayed in the browser for the user.

Client-side scripts process requests from a user using certain steps:
- The user accesses a Web page through a browser.
- It downloads the scripts embedded in the page to the browser.
- The user makes a request from the browser.
- The request is processed by the script in the browser itself.
- The output is displayed in the browser for the user.

Say: The server-side script connects the server and the client (browser).

Do: Compare and contrast server-side scripting and client-side scripting using the table given:
Server-side Scripting
Purpose: Works at the server-side using HTTP protocol
Processing: Requires application and Database servers to process the requests
Security: Secured as the code is hidden

Client-side Scripting
Purpose: Works at the browser (client) itself
Processing: No server-side components are required
Security: Not secured as the code can be viewed

Say: Some languages that have been developed for server-side scripting include Python, JSP (and Servlets), and PHP.

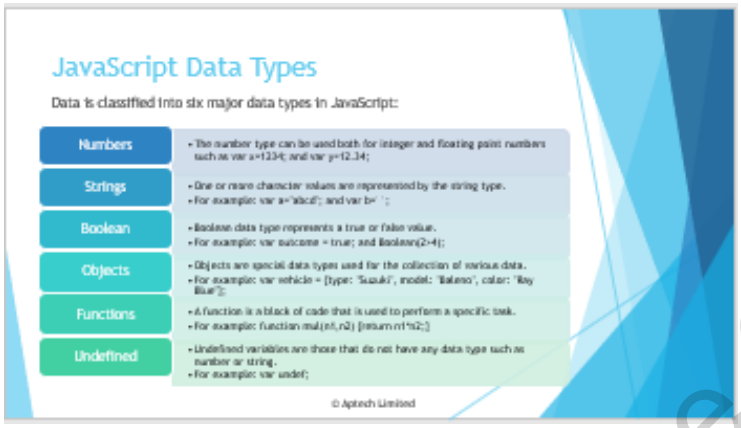
Do: Explain the different scripting languages.

Say: PHP is one of the best server-side languages used on the Web. It was designed to extract and manipulate information in the database. The language is used in association with SQL language for databases. It is used in Web applications such as Facebook, WordPress, and Wikipedia.

Python is a simple and user-friendly language, which is easy to learn. It functions well in the object-oriented environment and it is used in popular Web applications such as YouTube and Google.

A popular example of client-side programming language is JavaScript. JavaScript can be used to perform actions such as running checks on form values and sending alerts to the user's browser. Other programming languages used for client-side programming are VBScript, HTML, CSS, and AJAX.

Slide 11
Understand the datatypes used in JavaScript.



Ask: What do you know about data types?

Do: Consolidate the responses.

Say: Data type is one of the important features in JavaScript. These can be represented and manipulated in a programming language. Data is classified into six major data types in JavaScript:

- Numbers - Numeric values are defined as numbers in JavaScript. The **number** type serves both for integer and floating point numbers. For example, **345**, **123.34**.
- Strings - One or more character values are represented by the string type; for example, **"Good Morning"**.
- Boolean - Boolean Data type represents a true or false value. For example, **true** or **false**.
- Objects - Objects are data types used for the collection of various data.

Do: Point out the example on the slide:
Example:

```
var vehicle = {type: "Suzuki", model: "Baleno", color: "Ray
Blue"};
```

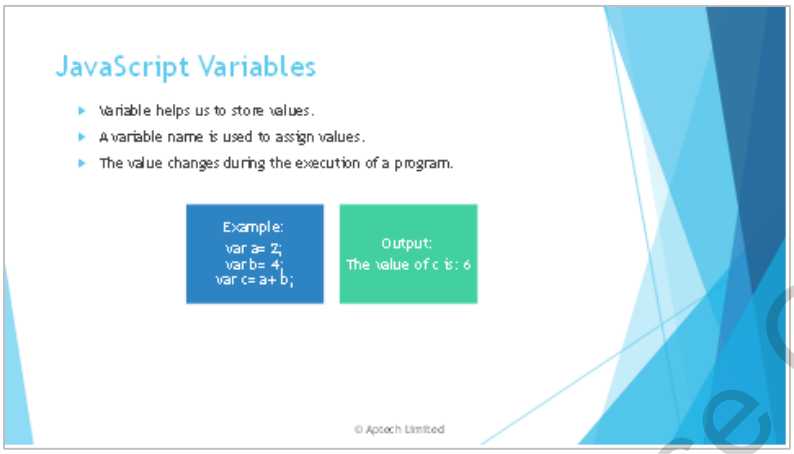
Say:
Functions - A function is a set of code which can be used to perform a specific task. It can be used wherever required in the script for performing a task repeatedly.

Do: Point out the example on the slide:
Example:

```
function multi(a1,a2) { return a1*a2; }
```

Say:
Undefined - undefined variables are those that do not have any data type such as a number or a string. For example, var **undef**;

Slide 12
Describe the uses of variables.



Do: Explain briefly about the JavaScript variables.

Say: Variable helps us to store the values. A variable name is used to assign values. The value changes during the execution of a program.
Example:
var a= 2;
var b= 4;
var c= a+ b;

Do: Tell the students, a, b, and c are variables in the example.

Say: Variable a is assigned value 2.
Variable b is assigned value 4.
Variable c is assigned value 6 after performing the addition operation.

Ask: Can anyone volunteer an example of using variables and getting an output from performing some operations on the variables?

Do: Allow a couple of volunteers to share their examples either using the whiteboard or orally.

Thank the volunteers. If none are forthcoming, write the following on the white board and ask the students to come up with the value of students in the example: (answer is 2+5+3+7=17)
var subj1 =2;
var subj2=5;
var subj3=3;
var subj4=7;
var totalmarks = subj1+subj2+subj3+subj4;

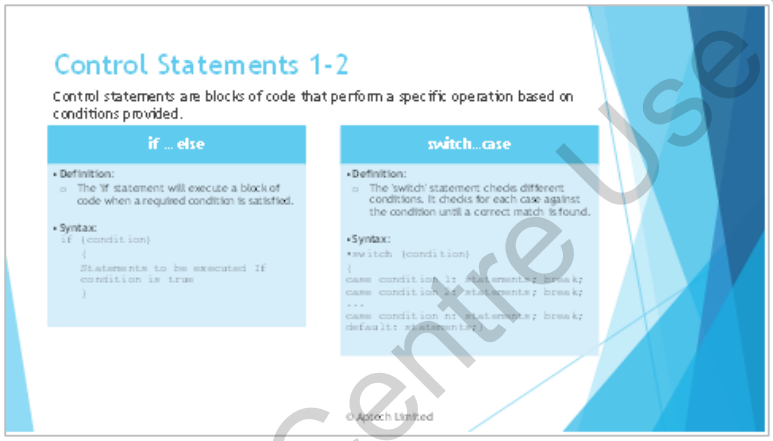
Thank the volunteers and ask the student who got the answer right to explain how he or she arrived at the result.

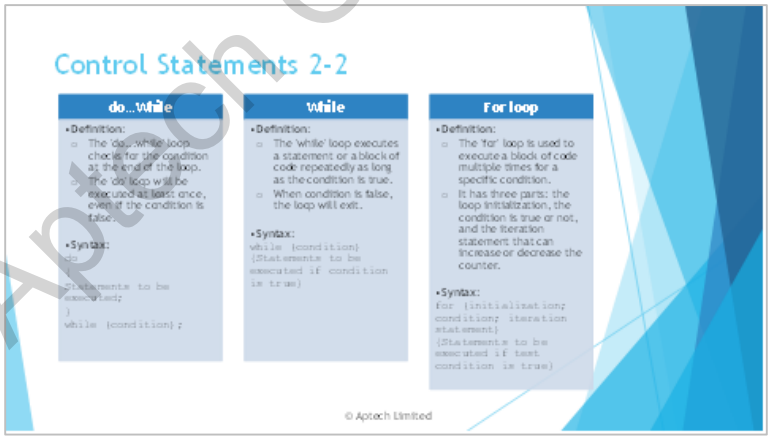
Say: Following points are important regarding variables:

- To declare a variable, JavaScript uses a reserved keyword 'var'.
- A variable must have a name of its own.
- To assign a value to a variable an operator equal to (=) is used.
- A variable must be declared before using it in the program.

Slides 13 and 14

Explain the control statements.





Say: Control statements are blocks of code that perform a specific operation based on conditions provided. Following are a few control statements in JavaScript:

- if … else
- switch…case
- do while loop
- while loop
- for loop

Do: Talk about the control statements in JavaScript. The students might be aware of control statements. You can ask few questions and derive the answer from them.

Ask: Let us say you go to see a movie, but you are not sure for which movie you will be able to get the tickets. So, do you think in such a case you can use an 'If-else' statement?

Answer: Yes
Say: The 'if' statement will execute a block of code when a required condition is satisfied. If accompanied by else block, it will execute another block of code when the condition is not satisfied.

Do: Show the syntax for if…else statement.

Ask: Can anyone take a guess as to how the 'switch…case' is different from the 'if-else' statement?

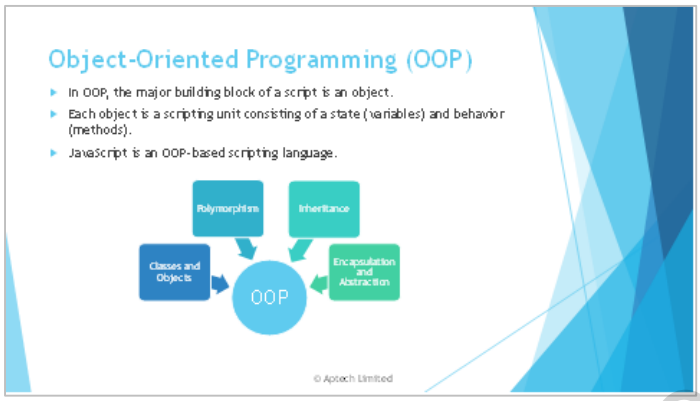
Answer:
The 'switch' statement will check for condition. It checks for each case until a correct match is found. If no matches are found, the default condition will be executed.

Do: Show the syntax for switch… case statement.

Say:
The 'do' loop will be executed at least once, even if the condition is false. The 'while' loop will execute the code repeatedly as long as the condition is true.
Do: Show the syntax for 'do…while' and 'while' loop.

Say:
The 'for' loop is used to execute a block of code multiple times for a specific condition. It has three parts:
The loop initialization, the condition is true or not, and the iteration statement that can increase or decrease the counter. If the condition is true, then the code given inside the loop will be executed; otherwise, the loop will exit. The initialization, condition, and iteration can be given in a single line.

Slide 15
Understand OOP.



Ask: Can anybody say what is the full form of OOP? Is anyone here familiar with OOP concept?

Do: If anybody is aware of OOP or has prior knowledge of any OOP language such as C++, ask them to briefly describe the concept of OOP. There could be a class with no student aware of OOP' in such a case, simply state that the students will learn about OOP next.

Answer: OOP is Object Oriented Programming.

Say: In OOP-based systems, the principal building block of a script is an object.
Each object is a scripting unit consisting of a state or variables, and behavior or methods.
JavaScript is an OOP-based scripting language.

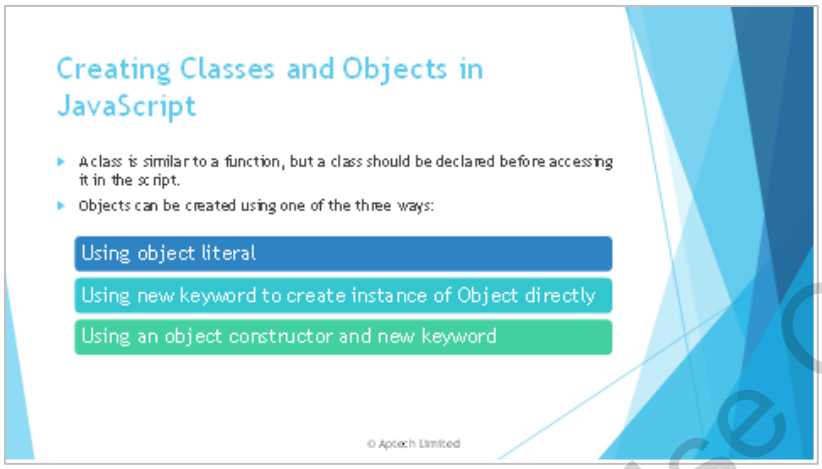
Do: Show the picture on the slide that depicts the basic OOP concepts.

Discuss the basic OOP concepts.

Say: Let us learn about some basic OOP concepts.

- Classes: A class defines its properties and functions. For example, let us take Bird as a class that consists of a type of living beings, which have body parts and can fly.
- Objects: An object is an instance of a class, and multiple instances are allowed in a class. An object has both data and function. For example, the woodpecker is a type of bird, so we can have an object Woodpecker that is an instance or an object of the class, Bird.
- Polymorphism: Polymorphism allows us to redefine the functionality of a particular object. It can have multiple forms.
- Inheritance: Object acquires the properties and behaviors of the parent object.
- Encapsulation: Binding the data and functions into a single unit is called encapsulation.

Slide 16
Explain creating classes and objects in Java.



Say: A class in JavaScript is similar to a function. The difference being that it should be declared before accessing in the script. Defining a class is as simple as defining a function.

Do: Write the following example on the black or whiteboard:

```
class Fruit()
{

}
```

Say: In the example, a new class named Fruit is defined.

Say: Objects can be created in JavaScript using one of the three ways.

Do: Read out the three ways from the slide: Using object literal, using new keyword to create instance of Object directly, and using an object constructor and new keyword.

Do: Then, using the following points and examples, explain how to create objects using each of the three method.

For example, one can create a new object using **new** keyword as follows:

```
var test = new Object;
```

Here, the test will be of type Object.

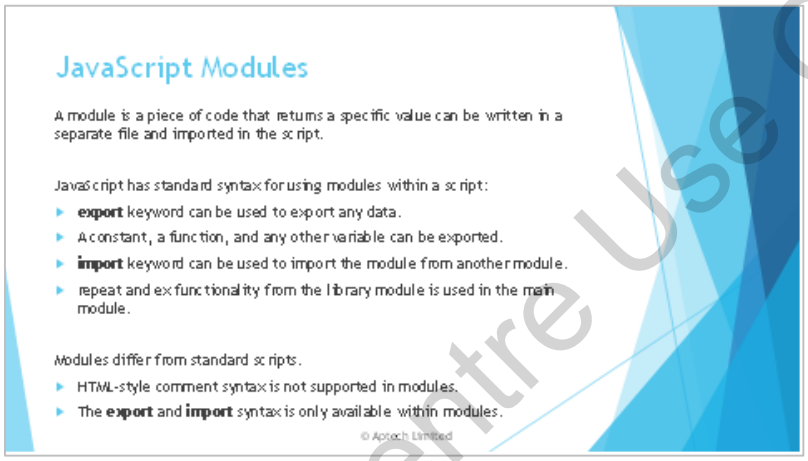
Example code:

In the following example, a class named Car is defined and two instances are created (car1 and car2).

```
function Car(name, color, brand) {
 this.carname = name;
 this.color = color;
 this.brand = brand;
}
var car1 = new car("Honda City", "Ray blue", 40, "Honda");
var car2 = new car("honda Accord", "blue", 40, "Honda");
```

Slide 17

Understand JavaScript modules.



Say: JavaScript modules are similar to libraries. A module is a piece of code that returns a specific value can be written in a separate file and imported in the script. JavaScript has standard syntax for using modules within a script.

Say: The **export** keyword can be used to export any data. A constant, a function, and any other variable can be exported.

Do: Write the following code on the board or whiteboard and demonstrate the use of the **keyword** export:

```
// lib.mJAVASCRIPT
export const repeat = (string) => '${string} ${string}';
export function ex(string) {
 return '${string.toUpperCase()}!';
}
```

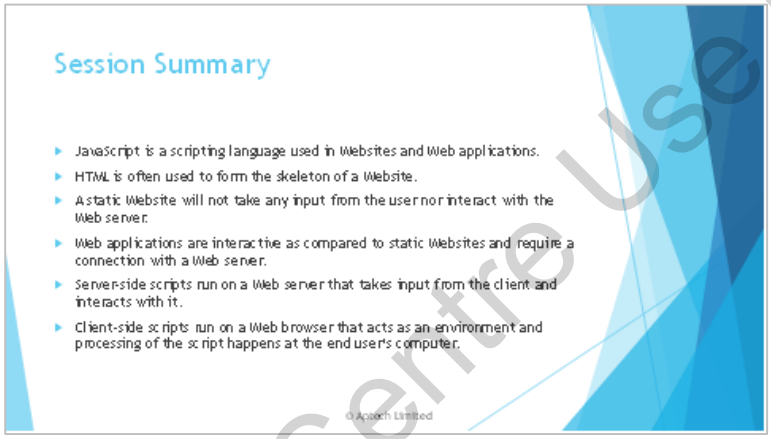
Say: The **import** keyword can be used to import the module from another module. The repeat and ex functionality from the library module are used in the main module.

Do: Explain how modules differ from normal scripts.

Say: Modules slightly differ from standard scripts. The HTML style comment syntax is not supported in modules, though it can work in classic-type scripts. The import and export syntax are only available within modules. It will not work in standard scripts.

It is because of these differences; the same JavaScript code would work differently when treated as a module versus a standard script. Therefore, the JavaScript runtime needs to know which script are components.

Slide 18
Revise the concepts learnt through a small activity.



Do: Conduct a small activity to engage the students in recalling the key learning points of the session. Divide the class into two or three groups depending on the class size. Ask each group to pose a concept-related question based on what they have learned in the session to the other teams. Allow the teams to pose questions to each other in turns. Allow not more than five minutes for this activity. At the end, thank and congratulate the students for a great recap. Then, debrief presenting the summary points.

Say: We are at the end of this session. Let us have a small activity to see how well you have learned the concepts presented. Please be a part of any one of <two/three> groups. In this activity, each team gets to ask a question based on the concepts presented in this session to the other teams. Try to make it hard for the others. If they are unable to answer, the team posing the question should be able to answer. Each team will get a chance to pose their questions in turns in a round-robin fashion.

(After the activity, Say): Well done, all of you. Let me now summarize the key points of the session.

- JavaScript is a scripting language used in Websites and Web applications. It allows Web pages to be more interactive and independent of a Web server.
- HTML is often used to form the skeleton of a Website.
- A static Website will not take any input from the user nor interact with the Web server.

- Web applications are interactive as compared to static Websites and require a connection with a Web server.
- Server-side scripts run on a Web server that takes input from the client and interacts with it.
- Client-side scripts run on a Web browser that acts as an environment and processing of the script happens at the end user's computer.

Say: Thank you for your participation in this session. The next session will be Matching Patterns in JavaScript.

1.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand the Matching Patters in JavaScript that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 2 – Matching Patterns in JavaScript

2.1 Pre-Class Activities

Before you commence the session, you should introduce the course in brief.

2.1.1 Objectives

By the end of this session, the learners will be able to:

- Explain Regular Expressions in JavaScript and its uses
- Identify RegExp object and matching patterns in JavaScript
- Describe the uses of modifiers, brackets, and Metacharacters
- Describe properties and methods of RegExp in JavaScript

2.1.2 Teaching Skills

To teach this session successfully, you must know about RegExp in JavaScript. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

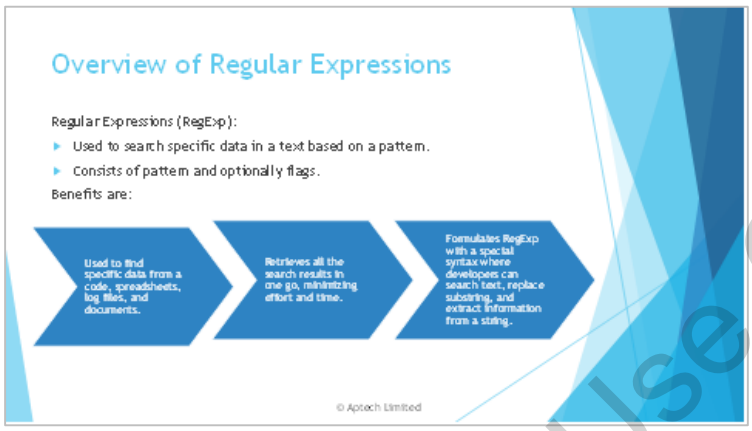
Follow the order given here during In-Class activities.

2.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about matching patterns using Regular Expressions in JavaScript and also explains the uses of modifiers, brackets, and metacharacters.

2.2 In-Class Explanations

Slide 3
Understand Regular Expressions.



Ask the students, do they have any idea about how to search and extract a string from a text?

Do: Give some time for them to answer. Consolidate the responses.

Say: Regular Expression (RegExp) is used in programming languages to search for specific data in a text based on a pattern. In JavaScript, a RegExp comprises a pattern and optionally, flags. The benefits of RegExp are as follows:

- It can be used to find specific data from a code, spreadsheets, log files, and documents.
- It retrieves all the search results in one go, minimizing the effort and time.

By formulating a RegExp with a special syntax, developers can:
- Search text in a string.
- Replace substring in a string.
- Extract information from a string.

Do: Show the syntax for regular expression on the white board as

```
/pattern/modifiers;
```

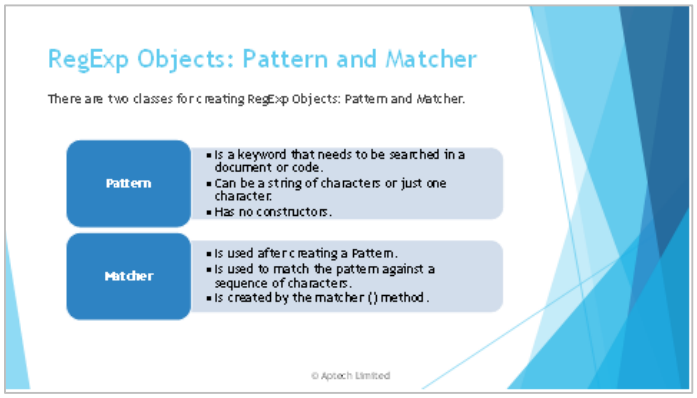
Demonstrate the use of Regular Expression using the example given:

```
var checklist = "Watching Birds";
var n = checklist. Search (/Birds/i);
```

This code will return 10, which is the position of the word ‘Birds’ in the string ‘Watching Birds’.

Slide 4

Understand the RegExp Objects – Pattern and Matcher.



Say: The RegExp object has predefined properties. Pattern and Matcher are the two classes involved in creating a RegExp Objects.

Explain the meaning of pattern and matcher as:

Pattern:

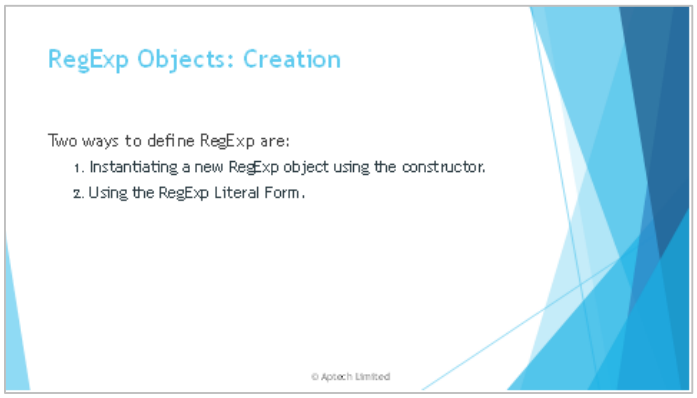
Pattern is a keyword that needs to be searched in a document or code and could be a string of characters or just one character. It has no constructors.

Matcher:

The Matcher is used after creating a Pattern. It is used to match the pattern against a sequence of characters. It is created by the matcher () method.

Slide 5

Understand the RegExp Objects Creation.



Say: RegExp is an object. It can be defined in two ways:

1. Instantiating a new RegExp object using the constructor.

Do: Write on the whiteboard

```
var regObj1 = new RegExp ('Madrid')
```

Say: In the statement, the pattern 'Madrid' is instantiated by the object.

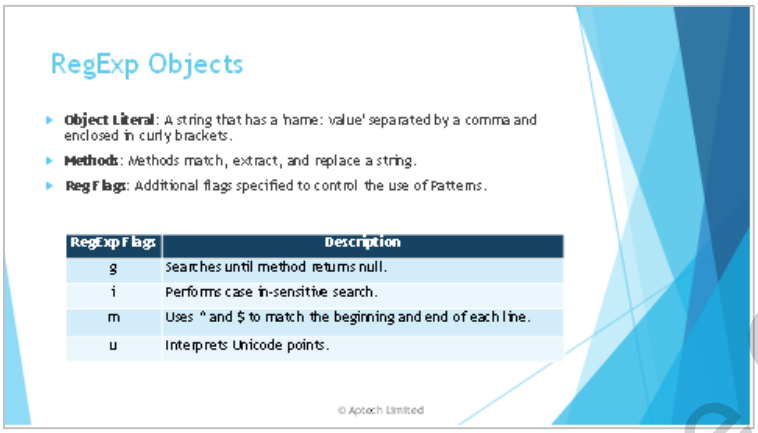
2. Using the RegExp Literal Form:

The pattern can be instantiated with the forward slashes using Literal as follows:

```
var regObj1 = /Madrid/
```

Slide 6

Understand the RegExp Objects.



Say: Regular Expression can be expressed as object literal: Object Literal is string that has a 'name: value' separated by a comma and enclosed in curly brackets.

Demonstrate using the example

```
var bike = {Name: "Unicorn", Capacity:" 150cc", Fueltype:
"Petrol"};
var output = bike.Name + "has" + bike.Capacity + "capacity.";
```

To match a string, RegExp Objects uses methods. Methods match, extract, and replace a string.

Do: Show the methods usage using an example. Write the example and syntax on the board.

Example:

```
matches
inputStr.match (regular expression)
index
inputStr.search (regular expression)
replacedStr
inputStr.replace (regular expression)
```

Say:

As shown in the example, RegExp.test() can be used to match a string and extract information from a string.

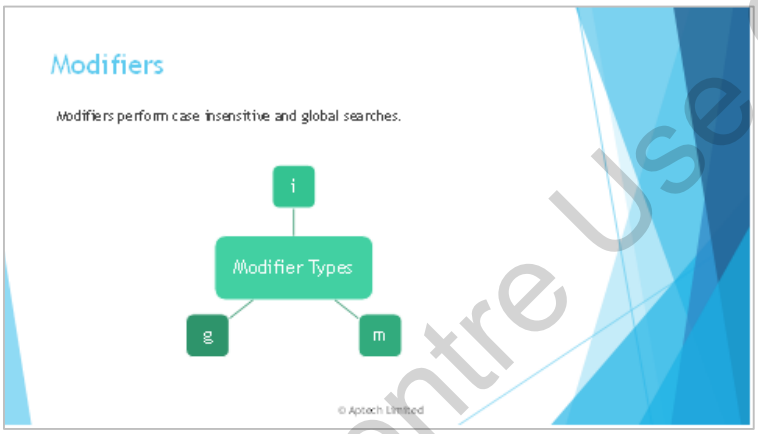
Syntax is `RegExp.exec()`.

Say: Regular expressions has optional flags can be specified to control the use of Patterns while matching against other strings.

Do: Show the table and explain the different RegExp Flags as shown:

- g allows to run RegExp.exec () numerous times to find each matching input string until the method returns null.
- i makes the pattern case insensitive so that it matches all the strings of different capitalizations.
- m Is necessary to have newline characters (/n). This flag allows the start and the end metacharacters (^ and $ respectively) to match at the beginning and at the end of each line.
- u Interprets the regular expression as Unicode points.

Slide 7
Understand modifiers.



Ask: Have you heard of Modifiers in JavaScript?

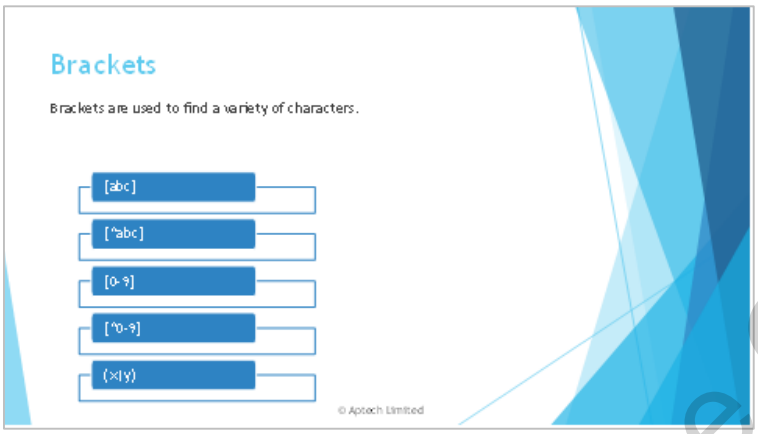
Do: Wait for response. Consolidate the responses.

Say: Explain briefly what a modifier is and give the description of the various modifiers as follows:

Modifiers are used to perform case insensitive and global searches. Different types of modifiers are:

- i is used for case-insensitive match
- g for global matching
- m for multiline matching

Slide 8
Understand brackets.



Ask: What is the use of brackets?

Do: Wait for response.

Say: Explain briefly what are brackets and give the description of various bracket expressions as follows:

Brackets are used to find a variety of characters.

- [abc] searches any character between the brackets
- [^abc] is used to search any of the character outside the brackets
- [0-9] is used to search any digit between the brackets
- [^0-9] is used to search any of the digit outside the brackets
- (x|y) is used to search any of the alternatives specified

Do: Write the code snippet on the board.

```
var inputTxt = "Do you have money?";
var srchPatt = /[^a-o]/g;
var toWrite = inputTxt.match(srchPatt);
```

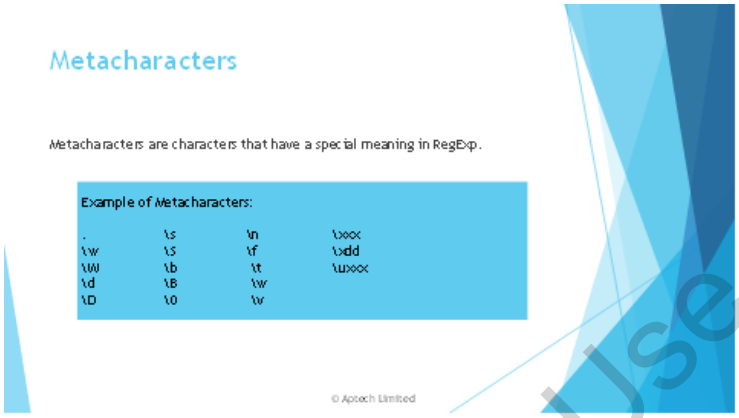
Ask: What will be the output of the code?

Do: Wait for the response. Consolidate the response.

Say:
The output will be:
"D", " ", "y", "u", " ", "v", " ", "y", "?"
Here,

[^a-o] acts as search pattern on the string inputTxt to print the characters not between lowercase a and o.

Slide 9
Understand metacharacters.



Say: Metacharacters are characters with a special meaning in RegExp.

Discuss: the meaning of various metacharacters as follows:

- . - Search for a single character
- \w - Search a word character
- \W - Search a non-word character
- \d - Search a digit
- \D - Search a non-digit character
- \s- Search a Whitespace character
- \S-Search a non-whitespace character
- \b-Search a match at the beginning/end of a word
- \B-Search a match not at the beginning/end of a word
- \0-Search a NULL character
- \n-Search a new line character
- \f-Search a form feed character
- \r-Search a carriage return character
- \t-Search a tab character
- \v-Search a vertical tab character
- \xxx-Search the character specified by an octal number xxx
- \xdd-Search the character specified by a hexadecimal number dd
- \uxxxx-Search the Unicode character specified by a hexadecimal number xxxx

Do: Write the code snippet on the board.

```
var strPatt = "Hello! Good Morning!! How is the market?";
var srchPatt = /\w+/g;
var toWrite = str.match(srchPatt);
```

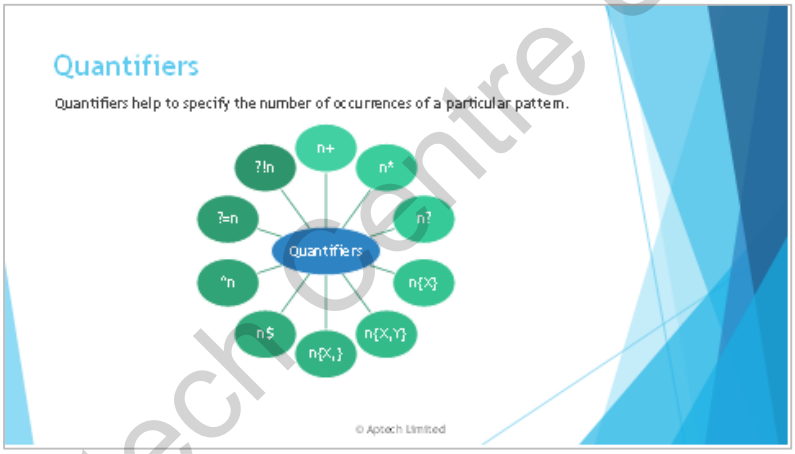
Ask: What will be the output?

Do: Wait for the response.

Say:
Output will be:
Hello, Good, Morning, How, is, the, market

Here,
'\w' searches word characters. '
/g' does global matching.
The non-word characters are removed in the output.

Slide 10
Describe quantifiers.



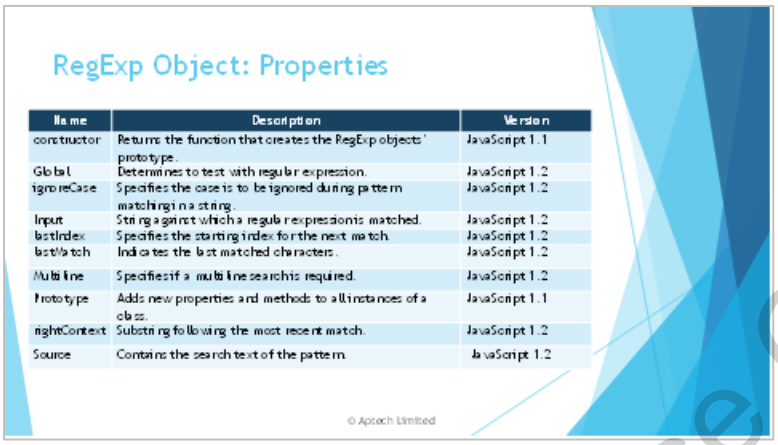
Say: Quantifiers help to specify the number of occurrences of a particular pattern.

Discuss the meaning of each of the quantifier as follows**:**

- n+: Co-ordinates any string that contains at least one n
- n*: Co-ordinates any string that contains zero or more occurrences of n
- n?: Co-ordinates any string that contains zero or one occurrences of n
- n{X}: Co-ordinates any string that contains a sequence of X n's
- n{X,Y} :Co-ordinates any string that contains a sequence of X to Y n's
- n{X,}: Co-ordinates any string that contains a sequence of at least X n's
- n$: Co-ordinates any string with n at the end of it
- ^n: Co-ordinates any string with n at the beginning of it
- ?=n: Co-ordinates any string that is followed by a specific string n
- ?!n: Co-ordinates any string that is not followed by a specific string n

Slide 11

Explain the RegExp Object Properties.

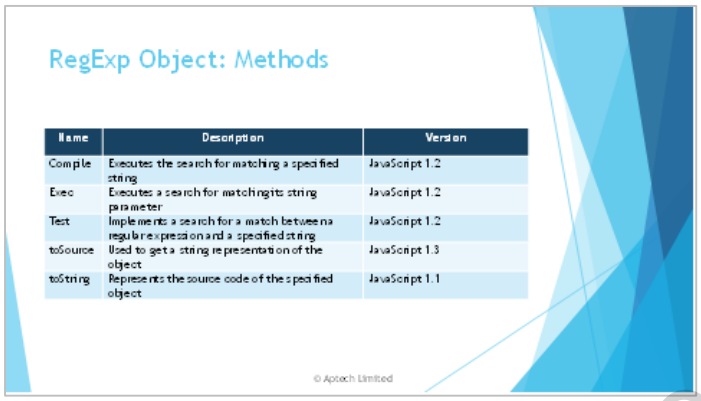


Say: Let us at look at the properties of RegExp Object and in which version of JavaScript it was implemented.

Do: Show the table and explain the properties of RegExp Object as follows:

- Constructor returns the function that created the RegExp objects' prototype and was implemented in JavaScript 1.1
- Global determines to test with regular expression and was implemented in JavaScript 1.2
- ignoreCase determines the case is to be ignored during pattern matching in a string and was implemented in JavaScript 1.2
- Input is a string against which a regular expression is matched and was Implemented in JavaScript 1.2
- lastIndex specifies the index from where to start the next match and was implemented in JavaScript 1.2
- lastMatch indicates the last matched characters and was implemented in JavaScript 1.2
- Multiline describes whether a search in strings should be performed across multiple lines and was implemented in JavaScript 1.2
- Prototype is used to add new properties and methods to all instances of a class and implemented in JavaScript 1.1
- rightContext is the substring following the most recent match. It was implemented in JavaScript 1.2
- Source is a read-only property that has the text of the pattern. It was implemented in JavaScript 1.2

Slide 12
Understand RegExp Object Methods.



Say: Let us at look at the methods of RegExp Object and the version of JavaScript in which it was implemented.

Do: Show the table and explain the methods of RegExp Object as follows:
- Compile is used to execute the search for matching a specified string and was implemented in JavaScript 1.2
- Exec executes a search for matching its string parameter and was implemented in JavaScript 1.2
- Test implements a search for a match between a regular expression and a specified string and was implemented in JavaScript 1.2
- toSource is used to get a string representation of the object and was implemented in JavaScript 1.3
- toString represents the source code of the specified object and was implemented in JavaScript 1.1

Do: Write the code snippet on the board.

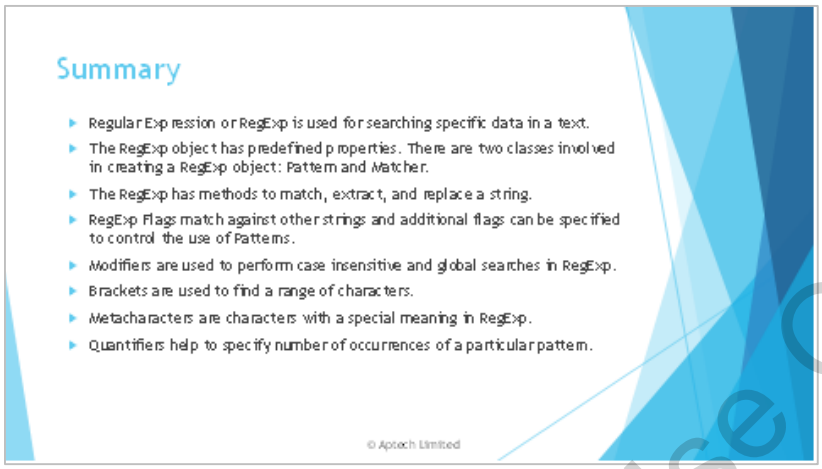
```
var numtoConvert = 100;
var toPrint = numtoConvert.toString();
```

Ask: What will be the output of the code?

Do: Wait for the response. Consolidate the response.

Say: Output is number 100 is converted to a string.

Slide 13
Summarize the session.



Do: Conduct a small activity to engage the students in recalling the key learning points of the session. Divide the class into two or three groups depending on the class size. Ask each group to pose a concept-related question based on what they have learned in the session to the other teams. Allow the teams to pose questions to each other in turns. Allow not more than five minutes for this activity. At the end, thank and congratulate the students for a great recap. Then, debrief presenting the summary points.

Say: We are at the end of this session. Let us have a small activity to see how well you have learned the concepts presented. Please be a part of any one of <two/three> groups. In this activity, each team gets to ask a question based on the concepts presented in this session to the other teams. Try to make it hard for the others. If they are unable to answer, the team posing the question should be able to answer. Each team will get a chance to pose their questions in turns in a round-robin fashion.

(After the activity, Say): Well done, all of you. Let us now summarize the key points of the session.

- Regular Expression or RegExp is used for searching specific data in a text.
- The RegExp object has predefined properties. There are two classes involved in creating a RegExp object: Pattern and Matcher.
- The RegExp has methods to match, extract, and replace a string.
- RegExp Flags match against other strings and additional flags can be specified to control the use of Patterns.
- Modifiers are used to perform case insensitive and global searches in RegExp.
- Brackets are used to find a range of characters.
- Metacharacters are characters with a special meaning in RegExp.
- Quantifiers help to specify number of occurrences of a particular pattern.

Say: Thank you for your participation in this session. The next session will be preprocessing in JavaScript.

2.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand the preprocessing in JavaScript that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 3 – Pre-processing JavaScript Files

3.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

3.1.1 Objectives

By the end of this session, the learners will be able to:

- Describe what a preprocessor is
- Explain supported preprocessors in detail
- Explain how preprocessing happens in JavaScript

3.1.2 Teaching Skills

To teach this session successfully, you must know about pre-processing in JavaScript and various tools used to support pre-processing. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

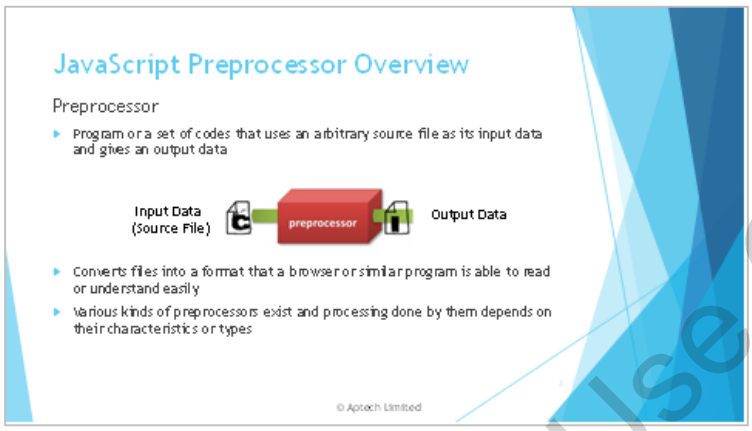
Follow the order given here during In-Class activities.

3.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the overview of preprocessing of JavaScript files and the process of preprocessing in detail.

3.2 In-Class Explanations

Slide 3
Understand Preprocessors.



Say: Let us get started with the first topic within this session - Introduction to Preprocessor in JavaScript.

Ask: What do you understand by preprocessor?

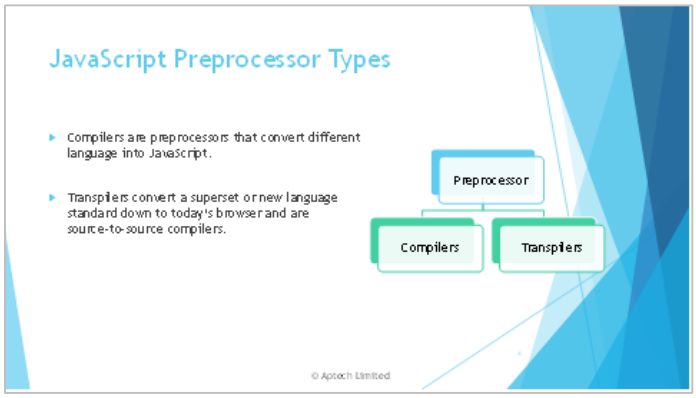
Do: Give the students some time to answer. Consolidate the responses. Define the term preprocessor. State the definition by reading out first point in the PPT. Explain the diagram.

Say: A preprocessor takes an arbitrary source file and converts it into something that the browser understands. Another program can use this output data as an input data. Preprocessors convert files into the format that a browser or a similar program is able to read or understand easily.

There are various kinds of preprocessors and the kind of processing they do depends on their characteristics/type. They can perform a simple task such as run a macro as well as perform highly complex tasks such as substituting a part of an entire programming language.

Slide 4

Understand preprocessor types.



Ask: Can any of you share what you know about compilers and transpilers?

Do: Wait for response. Consolidate the responses. Then, continue with the discussion. It could be possible that none of the students know these terms or may not be able to speak it out. In such a case, simply state that they will learn about these terms next and continue with the explanation.

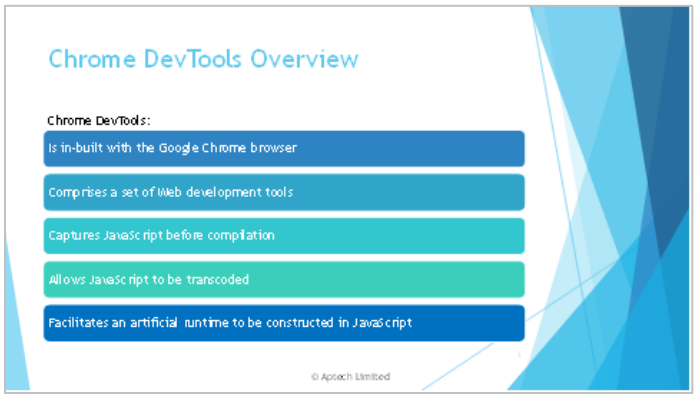
Say: JavaScript preprocessors are usually of two types. They are either compilers or transpilers. Compilers are preprocessors that convert an entirely different language into JavaScript.

Transpilers are the preprocessors that convert (transpile) a superset or new language standard down to today's browser. Languages that transpile to JavaScript are compile-to-JavaScript languages. Transpilers read source code written in one programming language and produce the equivalent code in another language. Transpilers can be referred to as source-to-source compilers. ECMAScript 6 (ES6) and CoffeeScript are popular examples of transpilers.

No preprocessors are specifically designed for JavaScript. Lexical (language) preprocessors are flexible, which means, they are preprocessors with lexical analysis rules that can be customized and can used with JavaScript. It is advisable to use preprocessors that have been designed for a language having a similar syntax as JavaScript. For example, C or C++ preprocessors.

Slide 5

Understand Chrome DevTools.



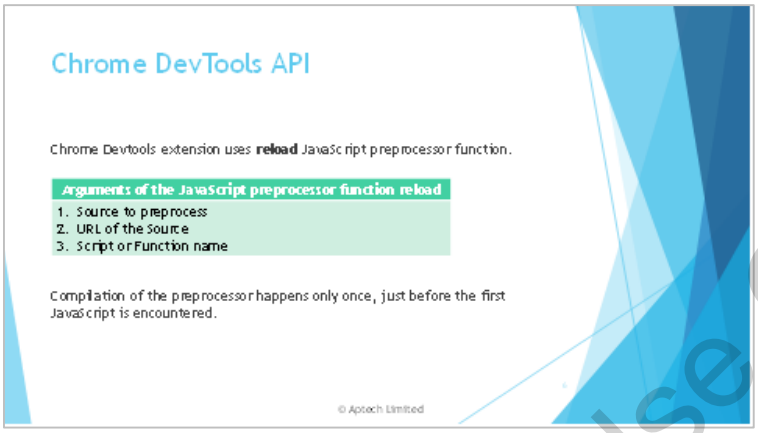
Say: Let us now explore the Google Chrome development tool, also called Chrome DevTools, which has JavaScript preprocessing feature.

Do: Explain the students that Google Chrome development tool, which is also called as Chrome DevTools, has JavaScript preprocessing feature.

Say: Chrome DevTools is in-built with the Google Chrome browser. Chrome DevTools comprises a set of Web development tools that facilitate easier development. Chrome DevTools captures JavaScript before compilation and allows JavaScript to be transcoded. Then, along with JavaScript that is injected into the page, the preprocessor facilitates an artificial runtime to be constructed in JavaScript. This in combination with other functions present in Chrome DevTools extension API enables developers to create new JavaScript-related tools.

Slide 6

Understand the API of Chrome DevTools.



Say: To use the script preprocessor, developers usually create code for Chrome Devtools extension that will reload the Web page having the preprocessor installed. I will share a sample skeleton structure for such a script.

Do: Write the given code on the whiteboard and explain the code:

```
chrome.devtools.inspectedWindow.reload({
     ignoreCache: true,
     injectedScript: runThisFirst,
     preprocessorScript: pre-processor
});
```

Do: Describe the structure of JavaScript preprocessor function.

Say: The JavaScript preprocessor function reload, takes three arguments:

- Source to preprocess: The loader bypasses the cache for all inspected page resources loaded, before the load event start
- URL of the source: The string will override the value of the User-Agent HTTP header, which sent while loading the resources of the inspected page.
- Script or function name of the event handler: Script will inject into every frame of the inspected page immediately upon load, before any of the frame's scripts.

Do: Share the tip mentioned:

Tip: Developers can visit the link http://developer.chrome.com/extensions/samples.html to view an example that illustrates the API call in a simple extension. Developers can download and extract the zip file. Then, they can use chrome://extensions in Developer Mode and load the extracted extension. After this, when they open or reopen DevTools, they can see the Preprocessor panel having a reload button that triggers a simple preprocessor.

Do: Explain about the effect of the pre-processor during navigating the Web pages.

Say: Remember that compilation of the preprocessor happens only once, just before the first JavaScript is encountered. The preprocessor remains active until the page reloads or navigate. If the Web page is navigated back and then forward, it will not result in any preprocessing.

Demonstrate: Do and show the steps shared here.

Say: You can visit the link http://developer.chrome.com/extensions/samples.html to view an example that illustrates the API call in a simple extension.

You can then download and extract the zip file. Then, use chrome://extensions in Developer Mode and load the extracted extension. After this, when you open or reopen DevTools, you can see the Preprocessor panel having a reload button that triggers a simple preprocessor.

Slide 7

Understand the supported Preprocessors and Transcompilers.

Say: We now come to the second topic in this session - Supported Preprocessors and Transcompilers.

JavaScript development has become better with introduction of various new frameworks.

For server-side Web development, JavaScript is mostly used along with several other languages, technologies, and frameworks. Whereas, for client-side programming, JavaScript is the primary and often, the only option, as all browsers support it. To overcome this problem, development of newer languages took place, which can compile down to JavaScript.

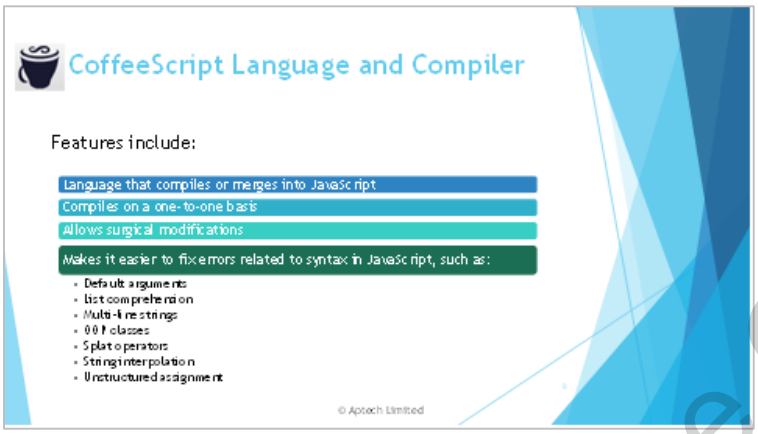
These new languages offer improved syntax that hides the complexity of new language. They can offer better ways to write and manage large code, while automatically generating 'correct' JavaScript as an alternative.

Do: Explain that there are various transcompilers available, which can do source to source compilation into JavaScript languages.

Say: Different transcompilers available are CoffeeScript, TypeScript, Traceur, Babel, and UglifyJS. Let us understand each of transcompilers in detail. Let us begin with CoffeeScript.

Slide 8

Understand CoffeeScript.



Do: Introduce students to CoffeeScript. Begin with the definition and then, point out its features.

Say: CoffeeScript is defined as a language in its own self that compiles or merges with JavaScript. It brings out the best of JavaScript in a simple way.

CoffeeScript compiles itself on a one-to-one basis and is equivalent to JavaScript, without requiring interpretation during runtime.

There are a few macroscopic errors introduced when developers make functional changes in existing projects. By functional, it simply means that JavaScript code changes because when a project is put together, there are many items to be synchronized. It is easier in CoffeScript as it works similar to modern day scripting languages.

Ask: So, given this background, do you feel CoffeeScript must have been very popular when it was introduced?

Do: Look for a yes, 'no', or 'we do not know' response from the students. In case someone is aware (says no), encourage the person to volunteer what he or she knows.

Say: Initially, after its introduction, CoffeeScript met with some resistance. Then, it evolved with time. In 2013, it was ranked 17th as the most used programming languages. This proved that JavaScript had lot of scope, but needed modernization. With more and more new scripts and languages coming up, the use of CoffeeScript declined.

Ask: Here is something else to ponder: does anyone know what is meant by the term surgical modification?

Do: Look for a 'no', or 'we do not know' response from the students. In case someone is aware (says yes), encourage the person to volunteer what he or she knows. The idea of these small interactions is to break the monotony. Ensure that you do not spend more than two minutes on such interactions.

Say: Sometimes, developers create another language that is extremely similar to an existing native language (in this case, JavaScript).

Other developers can easily learn these languages/scripts. Such languages have syntax similar to the native language. Hence, they are used to modify the native language and eliminate or lessen the flaws of the native language.

This method of modifying the native language using scripts/language that is equally understandable and fluent known as a surgical modification.

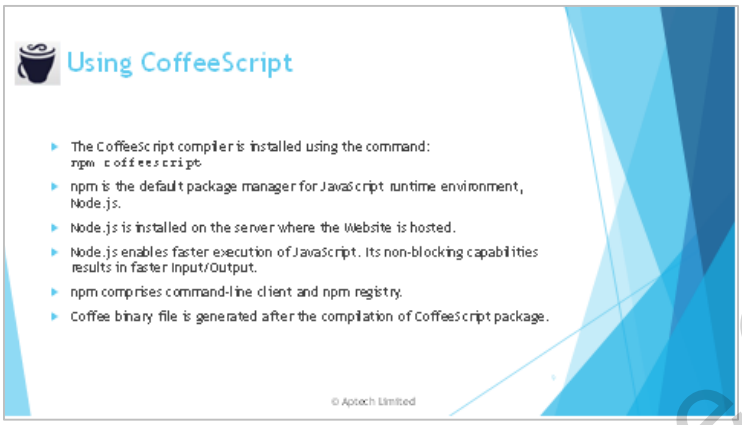
Do: Introduce students to the syntax errors in JavaScript.

Say: CoffeeScript is primarily capable of fixing errors related to syntax in JavaScript. JavaScript has a few syntax errors such as:

- Default arguments
- List comprehension
- Multi-line strings
- OOP classes
- Splat operators
- String interpolation
- Unstructured assignment

Slide 9

Understand how to use CoffeeScript.



Do: Explain the details about how to compile the CoffeeScript code.

Say: To compile CoffeeScript code into JavaScript, install CoffeeScript package using the command: `npm coffeescript`

Here, `npm` refers to the command-line client for the package manager for JavaScript, which also bears the same name, npm. npm is the default package manager for JavaScript runtime environment, Node.js.

Node.js is installed on the server on which the Website will be hosted and enables developers to execute JavaScript faster and in an easier manner. Node.js is primarily used due to its non-blocking capabilities, which results in faster Input/Output.

npm comprises the command-line client and an online database of public and private packages, known as the npm registry.

After the compilation of coffeeScript package, coffee binary file is generated, which is a pre-compiled file and provides an option of producing Source Maps.

Demonstrate: Type the command at the command prompt to install coffeeScript:

```
npm install -g coffeescript
```

Say: The command to compile a CoffeeScript file is: coffee --compile --map --output <filename>.

Do: Create a CoffeeScript file and save as test.coffee:

```
author = "J K Rowling"
booktitle = "Harry Potter and The Philosophers Stone -- #{
author }"
```

```
sentence = "#{ 22 / 7 } is an approximation of Pi"
```

Demonstrate: Show them the sample CoffeeScript file and show them how to compile them. The command is:

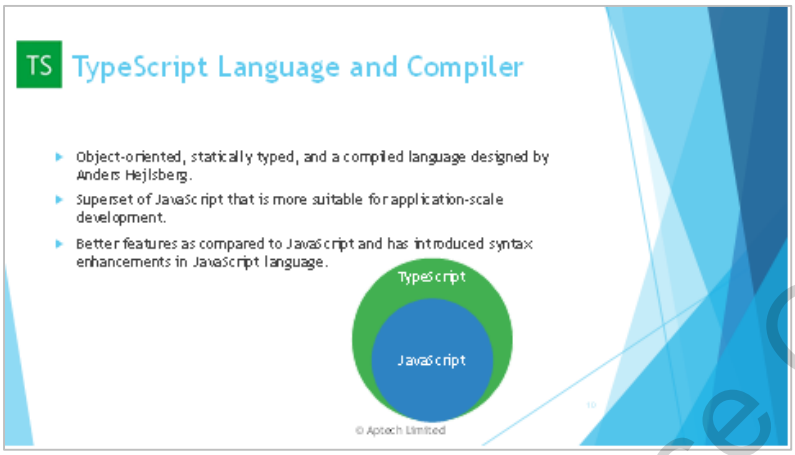
```
coffee --compile test.coffee
```

Show them the generated JavaScript file: test.js.

```
// Generated by CoffeeScript 2.3.2
(function() {
 var author, booktitle, sentence;
 author = "J K Rowling";
 booktitle = 'Harry Potter and The Philosophers Stone --
${author}';
 sentence = '${22 / 7} is an approximation of Pi';
}).call(this);
```

Say: To integrate and simplify the compilation processes of a larger build system, additional plug-ins may be required, such as is grunt-contrib-coffeescript or gulp-coffee.

Slide 10
Understand TypeScript.



Do: Introduce and define TypeScript.

Say: Let us move on to the next type of transcompiler, TypeScript. It is object-oriented, statically typed, and a compiled language designed by Anders Hejlsberg (of Microsoft who is well-known for designing C#). The first version of TypeScript (0.8) was released for public use in October 2012.

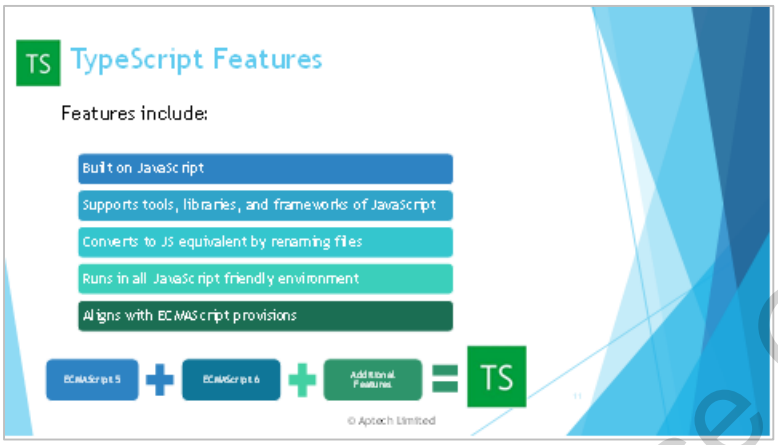
Do: Explain the image and simplify the definition of TypeScript.

Say: TypeScript can be defined as a language and superset of JavaScript that is more suitable for application-scale development. It is a language as well as a set of tools that are compiled in JavaScript itself.

It presents better features as compared to JavaScript. TypeScript has introduced syntax enhancements in JavaScript language.

Slide 11

Understand features of TypeScript.



Say: Some of the features of TypeScript are as follows:

Built on JavaScript

TypeScript is built on JavaScript and it adopts all basic building blocks of any program that is written in JavaScript. Those who can program in JavaScript can easily use TypeScript without any additional learning.

Library support

All the files compiled with TypeScript are in JavaScript and hence, they can be consumed by JavaScript codes. Therefore, compiled files can work with all tools, libraries, and frameworks of JavaScript.

JS and TS are equivalent

For executing the program, TypeScript codes are converted to JavaScript equivalent, just by renaming .js files as .ts. .ts files can compile along with TypeScript files.

TypeScript is light

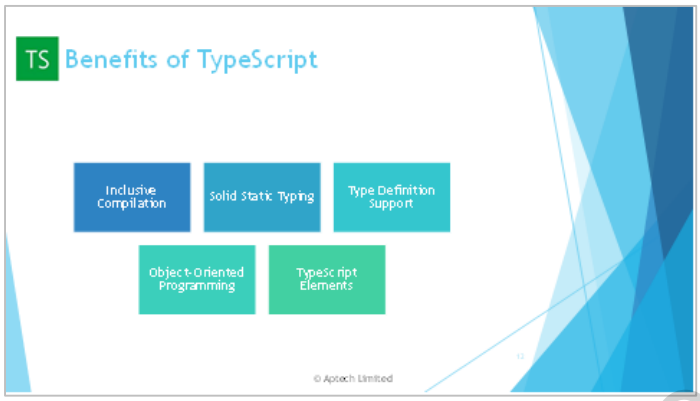
TypeScript is compact and does not need any specific runtime environment or a dedicated VM to execute. It runs on all environments that are
JavaScript-friendly including devices, operating systems, and browsers.

TypeScript aligns with ECMAScript provisions

TypeScript derives its fundamental language features from ECMAScript5, whereas its class-based orientation and modules are as per specifications of ECMAScript 6 (codenamed as 'Harmony'). JavaScript official specifications are in ECMAScript 5. Furthermore, TypeScript also holds features such as generics and type annotations. These do not form part of the ECMAScript 6 specification.

Slide 12

Understand benefits of TypeScript.



Do: Explain various benefits of TypeScript.

Say: TypeScript is better than languages such as Dart or CoffeeScript. While these languages need a language-specific execution environment, TypeScript has no such requirements, as it is JavaScript based. Let us look at various benefits of TypeScript.

Inclusive Compilation

JavaScript being an interpreted language needs many runs and reruns to decode the bugs. Unlike JavaScript, TypeScript offers a transpiler with an error-detection feature. TypeScript not only runs the code but also generates a compilation error for all the syntax errors. The highlighted errors are easier to rectify, and bug removal is less cumbersome before the script is tested.

Solid Static Typing

JavaScript needs developers to define every single variable. However, TypeScript comes with TypeScript Language Service (TLS). TLS offers static typing as well as type inference system. In case, a variable not defined or declared by the programmer, the value of the same is fetched through TLS.

Type Definition Support

TypeScript comes with a separate file for TypeScript Definition that has .d.ts extension. These files contain the definition for external JavaScript libraries thus, enabling TypeScript code to contain such external libraries.

Object-Oriented Programming

Object-oriented programming is supported by TypeScript for concepts such as inferences, inheritance, and so on.

TypeScript Elements

Core components of TypeScript are language, compiler, language service, and declaration files. Type annotations, keywords, and syntax form a part of the language. TSC or the TypeScript

compiler is the converter that converts TypeScript instructions into JavaScript equivalents. Language service aids a common set of standard editor operations such as signature help, code formatting, statement completion, colorization, and so on.

Note: A declaration file is generated when the TypeScript is compiled. Declaration files end with *.d.ts* extension and act as interfaces between components compiled in JavaScript. TypeScript declaration files are similar to the header of C/C#. These files display IntelliSense for function calls, types, and variable support for JavaScript libraries such as MooTools, jQuery, and so on.

Demonstrate: Show them how to incorporate classical inheritance.

Say: With TypeScript, it is possible to create classes that incorporate classical inheritance. As seen earlier, principal fundamentals of TypeScript permit static-typing as well as compile-time checking of JavaScript code via annotations.

Demonstrate: Show them a working example of TypeScript:

```
// addition.ts
function add(left: number, right: number): number {
 return left + right;
}
var result = add(1, 1);
result.foobar;
```

Say: To use this code, first type the command at the command prompt that will install TypeScript:

```
npm install –g typescript
```

Then, run the following command:

```
tsc addition.ts
```

Demonstrate: Show a code snippet having erroneous code that will undergo compile-time checking:

```
// addition.ts
function add(left: number, right: number): number {
 return left + right;
}
add("Foo");
add(1);
var result = add(1, 1);
result.foobar;
```

Type the following command at the command prompt:

```
tsc addition.ts
```

Demonstrate: Show them the errors.
addition.ts(5,1): error TS2346: Supplied parameters do not match any signature of call
target.addition.ts(6,1): error TS2346: Supplied parameters do not match any signature of call
target.addition.ts(9,8): error TS2339: Property 'foobar' does not exist on type 'number'.

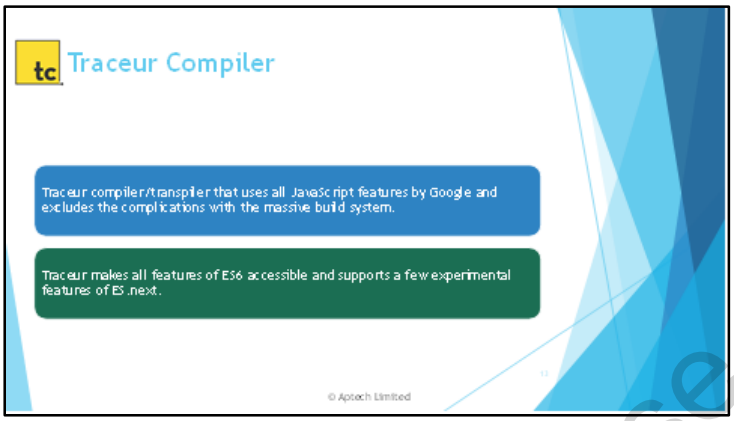
Do: Talk about Duck-Typing.

Say: When an object is passed through a combination of properties and methods to determine its semantics and belonging, it is known as duck-typing. TypeScript uses duck-typing to define structures that are complex.

Demonstrate: Show an example of duck-typing.

```
interface ObjectWithTitle {
 title: string;
}
function printTitle(titledObject: ObjectWithTitle) {
 console.log(titledObject.title);
}
var article = {
 title: "This is awesome!",
 publishedAt: new Date(),
 content: "Foo bar"
};
  printTitle(article);
```

Slide 13
Introduce Traceur.



Do: Introduce Traceur.

Say: Traceur happens to be a compiler/transpiler by Google. It uses all JavaScript features that have been tested and are not part of the native offering or future features of modern Web browsers. It is not only easy to utilize, but also excludes the complications with the massive build system.

ES modules can load easily when Traceur incorporated in a page. Apart from making all features of ES6 accessible, Traceur also supports a few of the experimental features of ES.next.

Do: Show them the Traceur online to understand how Traceur compiles down the ES6 code and ask the students to try.

Say: To test ES6 module, a simple HTML file setup with Traceur.js is required. It is possible to begin testing at once with JavaScript.

Demonstrate: Show them how to test ES6 module. Show them that the HTML code fragment gives a better understanding of various available functionalities.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>ES6 Modules</title>

 <!--Include Traceur to compile ES6 script content on the fly-->
 <script src="traceur.js"></script>
 <script src="BrowserSystem.js"></script>
 <script src="bootstrap.js"></script>
```

```
 <!--Load the base.js module-->
 <script>
  System.import('./base.js');
 </script>
</head>
<body>
</body>
</html>
```

Say: The base.js file is added to the main entry point of modules. This is done to avoid declaring multiple script tags required to include JavaScript files for loading modules.

The source files for JavaScript to be incorporated in HTML are located at Traceur.js github page (https://github.com/google/traceur-compiler/tree/master/example). They can be manually downloaded or can be installed via NPM.

Demonstrate: Show the use of system import.

Say: To dynamically determine the module to the loaded system, Import API is used.

Demonstrate: Show them the code and explain.

```
System.import('./base.js');
```

Here, System.Import is used to import the base.js module.

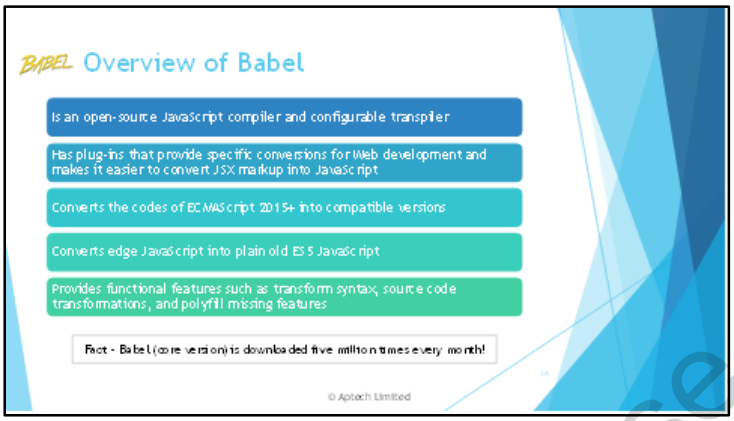
As soon as a module or file is loaded using ES6 module loader, whatever code is contained in that module is executed spontaneously. The module contained is executed only the first time it is loaded.

Do: Summarize the entire use of Traceur.

Say: There are many other ways to work with modules such as Babel, Browserify, Rollup, Webpack, and so on. It is an easy option of embedding ES6 modules while using Traceur.js.

Developers are still experimenting with the JavaScript ecosystem. On the evolution journey, ES6 tool is still evolving. Some tools can be consolidated and eventually, a standardized and stable release of such a tool will emerge.

Slide 14
Understand Babel transcompiler.



Say: Let us look at the transcompiler Babel.

Do: Introduce Babel and its features.

Say: Babel, is also known as Babel.js, is an open-source JavaScript compiler. It is a free and configurable transpiler used for Web development. Software developers can write source code in a preferred markup language or programming language. However, they need Babel, as a source-to-source compiler (transpiler) to translate it to JavaScript so that the modern Web browsers can read it.

Babel plug-ins provide specific conversions for Web development. For example, a program in React.js can be converted into Babel preset 'react'. Using Babel to convert JSX (JavaScript extension) markup into JavaScript is easy. Babel is a JavaScript compiler that converts the codes of ECMAScript 2015+ into compatible versions of JavaScript in the present and the older environments or browsers.

Babel converts edge JavaScript into plain old ES5 JavaScript that can run in any new as well as old browsers. Babel brings in synergy between new versions of JavaScript and the new ES6 specification that is, including classes, multiline strings, and fat arrows.

Babel provides functional features such as transform syntax, perform source code transformations (codemods), and find polyfill missing features in the target environment (via @babel/polyfill) and so on. Syntax transformers support Babel for the latest version of JavaScript. Plug-ins enable the use of new syntax instantly even in the absence of browser support.

Demonstrate: Using an example show ECMAScript 2015 arrow function, which is supported by all modern browsers.

```
[1, 2, 3].map((n) => n + 1);
```

After compilation, the output will be JavaScript ES5 equivalent syntax.

```
[1, 2, 3].map(function(n) { return n + 1; });
```

Say: Syntax transformers support Babel for the latest version of JavaScript. Plug-ins enable the use of new syntax instantly even in the absence of browser support.

Demonstrate: Type annotations with flow or TypeScript.

Say: Babel does not support type checking by itself; however, it can highlight type annotations with Flow or TypeScript preset to get started.

Demonstrate: Type the following command on command prompt to install the flow preset:

```
npm install --save-dev @babel/preset-flow
```

Or

The typescript preset:

```
npm install --save-dev @babel/preset-typescript
```

Following example demonstrates flow:

```
// @flow
function square(n: number): number {
 return n * n;
}
```

Following example demonstrates typescript preset:

```
// @TypeScript Preset
function Greeter(greeting: string) {
 this.greeting = greeting;
}
```

Say: Babel is made of plug-ins. Transformation pipeline can be composed using existing plug-ins or can be composed separately as well. Plug-ins can be easily used either by using an existing preset or by creating one. Plug-ins or plug-in templates can be created easily with the help of astexplorer.net and generator-babel-plug-in. Developers need to have a working copy of the latest stable version of Node.js.

Demonstrate: Show them the code:

```
// A plugin is just a function
export default function ({types: t}) {
 return {
 visitor: {
  Identifier(path) {
  let name = path.node.name; // reverse the name: JavaScript ->
tpircSavaJ
  path.node.name = name.split('').reverse().join('');
  }
 }
 };
}
```

Say: Availability of the Source Maps enables easy debugging of the compiled code.

Say: While Babel remains compliant to the ECMAScript standards, it also has few specific options that are more compliant with specifications to optimize performance.

Say: Babel is compact as it uses the minimalistic code and is not dependent on the significant runtime. However, where specific transforms use some 'loose' options, the spec compliance adjusted for speed, readability, and file size.

Say: Babel package is a Node module and can be installed using a simple command.

Demonstrate: Write the following code on the white or blackboard and inform the class about the command to install Babel:

```
npm install –save-dev 'babel-cli'
```

Say: Babel offers plug-ins for grunt, gulp, Webpack, Webstorm, Sublime, and so on. It can be used or inserted in almost any development toolchain. The new ES6 specification includes classes, multiline strings, and fat arrows functionality.

Say: JavaScript offers no classes and objects inherited from other present objects. Any object can be superclass or parent object to any other object. Any function can define as a constructor function. Calling the same function with a different keyword yields a new object. ES6 introduced the class keyword. It permits defining functions that can be only used as constructors. A class is a special cookie cutter object. It is often used to define other objects and acts as a special case of prototypical inheritance. Usually, only objects are created from functions that have been specifically decided. Let us look at classes in Babel in detail.

Do: Explain about classes in Babel.

Say: An ES6 class expression to create a new class is shown as follows:

Demonstrate:

```
class Person{ }
var dave = new Person
```

Say: When the code runs through Babel, it is referring to a constructor function.

Demonstrate:

```
@Constructor function
"use strict";
function _classCallCheck(instance, Constructor) {
 if (!(instance instanceof Constructor)) {
 throw new TypeError("Cannot call a class as a function");
 }
}
var Person = function Person() {
 _classCallCheck(this, Person);
};
```

Say: In the code snippet, Person function is used as a standard prototypical constructor. It can be safely checked with _classCallCheck function, which is called within the Person constructor. It will return errors if the Person function is not the constructor function.

Do: Explain about multiline strings.

Say: ES6 also defines strings in a novel manner. The back tick symbol (`) present helps in creating multiline strings. It is quite useful when defining JavaScript templates. Babel can create multiline strings using back ticks.

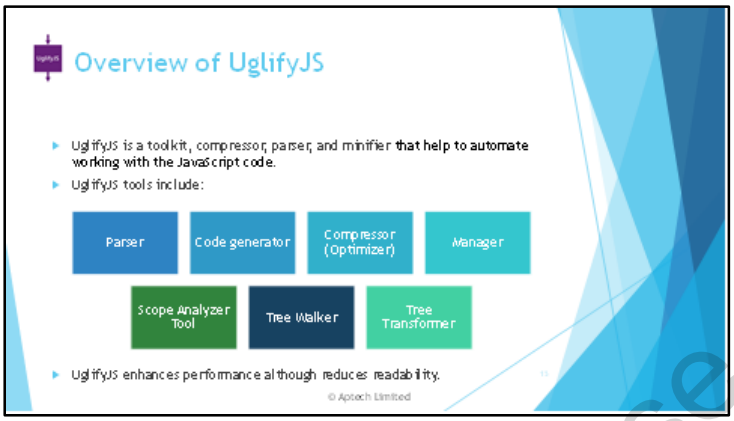
Demonstrate:

```
var template = `
<div>
 <h1>hello {{name}}</h1>
</div>`
```

Explain that the compilation result is:

```
var template = "
 <div>
 <h1>hello {{name}}</h1>
</div>
";
```

Slide 15
Understand Uglify.



Do: Explain the next toolkit – Uglify. Talk about its overview and features.

Say: Uglify or UglifyJS is a tool that compresses a JavaScript file, removes whitespaces, and even replaces long variable names with smaller ones. The file size is reduced and performance is enhanced.

Note that workflow tools such as gulp/grunt and Webpack automatically 'uglify' the files when they are built for production. This eliminates uglified code when the debug takes place on a local machine.

UglifyJS is a toolkit and a compressor, parser, and minifier, written in JavaScript. It also contains tools that help to automate working with the JavaScript code. These include:

- A parser, which produces Abstract Syntax Tree (AST) from a JavaScript code.
- A code generator outputs JavaScript code from an AST. In addition, Source Maps can generate by providing an option.
- A compressor (optimizer) — An AST can be optimized by using the transformer API.
- A manager — the local variable name is reduced to single-letters.
- A scope analyzer is a tool, which gives information about where the variables are defined/referenced and so on, that augments the AST.
- A tree walker — a simple API allows to do something on every node in the AST.
- The tree transformer — the tree transforms with another API.

Do: Next state that UglifyJS only supports ECMAScript 5 (ES5).

Say: Support for const is present, but incomplete and may not be transformed properly. To use the minify ES2015+ (ES6+) should install uglify-es package from npm.

Do: Next talk about the advantages and end with disadvantages.

Say: Advantages are:
Enhanced performance: The larger files take more time to transfer. Uglify reduces the size of the file and hence the time taken to transfer is reduced as well, resulting in enhanced performance.
The disadvantage:
Reduces Readability: Entire code is minimized and whitespaces are deleted. Hence, overall readability of code is affected. It is also difficult to debug. Thus, Uglify should be implemented only when there is an urgency for production. To achieve improved performance and sustained readability, which is not possible with UglifyJS along with debugging.

Do: Explain about the options available.

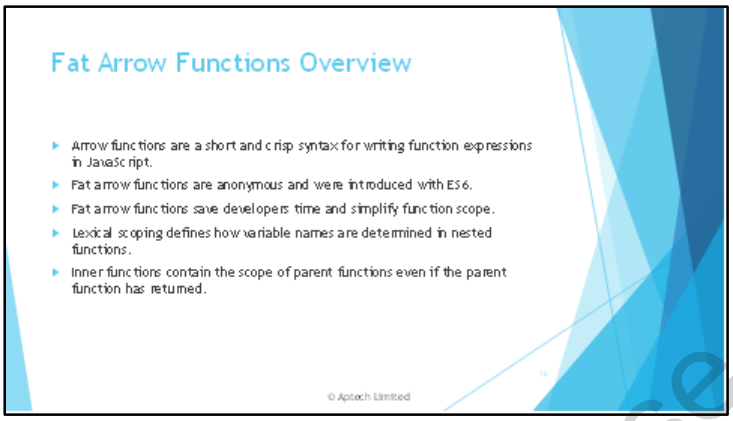
Say: There are many useful options available in UglifyJS. Following is a list of usage options:
Minify options
Parse options
Compress options
Mangle options
Output options

Demonstrate: Following is the command syntax: Uglifyjs [input files] [options]

Say: UglifyJS2 can take multiple input files. Input files are passed first and then, the options are passed. UglifyJS will parse input files in a sequence and apply any compression options. The files are parsed in such a way that a reference from a file to some variable/function declared in another file matches properly.

The usage of --compress will result in a certain level of size savings; however, using only this option will not achieve required level of savings. The JS files can be further minimized. It needs more options, such as dead code, to remove an unreachable code, sequences to join consecutive simple statements using the comma operator, and so on.

Slide 16
Understand Fat Arrow Functions.



Say: This is the third topic in today's session - Fat Arrow Functions.

Say: Arrow functions are a short and crisp syntax for writing function expressions in JavaScript. They utilize a new symbol, =>, which looks like a fat arrow and hence, are also called 'fat arrow' functions. Fat arrow functions are anonymous. They were introduced with ES6. They save developer's time and simplify function scope.

Demonstrate: Typically, the function expression with fat arrows is written in this syntax:

```
(param1, param2, paramN) => expr
```
where param signifies parameters and expr is an expression.

Consider an example:
```
const sum = (num1, num2) => { return num1+num2};
```

The right-side function has no name, it is anonymous, and it uses a fat arrow.

Earlier, before ES6 fat arrows feature, this same function would be written as:
var sum = function(num1, num2) { return num1 + num2;};

Thus, one can observe that fat arrows feature has shortened the code. This function is not called or invoked like other functions. Instead, it can be passed on as a parameter to a callback or promise or can be saved to a variable.

Do: Talk about the fat arrows with just one parameter.

Say: When there is only one parameter present, braces preceding the arrow can be omitted.

Following statement demonstrates an example:

```
num1 => { return num1 + 1 };

     The ES5 equivalent of this is: (function(num1) {
 return num1 + 1;
});
```

Do: Talk about the Fat Arrows with just one line of code.

Say: If a function contains only one line of code and it ends with a semi-colon, then curly brackets are not required.

```
num1 => num1 + 1;
```

Demonstrate: The result into compilation as:

```
(function(num1) {
 return num1 + 1;
)}
```

Following example uses one variable to display the output, which represents all the elements in an array:

```
[1, 2, 99].map(num => console.log(num));
```

The longer equivalent of this would be:

```
[1, 2, 99].map(function(num) {
 return console.log(num));
});
```

Do: Talk about Fat Arrows and 'this' (Lexical Scoping).

Say: In simple terms, lexical scoping defines how variable names are determined in nested functions. Inner functions contain the scope of parent functions even if the parent function has returned. Arrow functions determine their 'this' based upon current 'this' value when the function is declared and not based on its placement within an object. With arrow functions, developers can access the parent function's values, not the global values (such as window) and hence, these values can be updated inside the function itself by referring 'this'.

Demonstrate: Consider an example that demonstrates the concept:

```
 function Count() {
 this.number = 0;
 this.CountDown = setInterval(() => {
  this.number++;
  console.log(this.number);
 }, 1000);
}
var obj= new Count();
```

Say: This function will log the value every second, behaving like a countdown. Here, if the developer had used normal function in place of fat arrows, they would get NaN instead of a countdown. This is because in normal functions 'this' will refer to window and here, window does not have any 'number' variable.

Slide 17
Understand Preprocessing JavaScript using CoffeeScript.

Say: This is the final topic in today's session - Preprocessing JavaScript.

Let us discuss the preprocessing of JavaScript. So far, several preprocessors have been explored. First, we will be demonstrating CoffeeScript transcompiler.

Do: Remind them of the earlier example of CoffeeScript that was demonstrated.

Demonstrate: Show another example of a CoffeeScript code. It is saved with .coffee extension.

```
mood = greatlyImproved if singing
if happy and KnowsIt
 clapsHands( )
  chachacha( )
else
  showIt( )
lunch = if friday then pizza else tacos
```

After compilation, CoffeeScript code turns into this JavaScript:

```
(function( ) {
  var lunch, mood;
  if (singing) {
   mood = greatlyImproved;
  }
```

```
  if(happy && KnowsIt) {
   clapsHands( );
   chachacha( );
  } else {
   showIt( );
  }
  lunch = friday ? pizza : tacos;
} call(this));
```

Say: There are two ways to execute or run CoffeeScript:

Manually compile the CoffeeScript code and then, run the output JavaScript code as a Node.js application. Run the original CoffeeScript code through the Node.js run configuration and compile it during run.

Demonstrate: How to compile CoffeeScript manually and run the generated JavaScript code:

Say: Step 1: The CoffeeScript code is compiled to JavaScript. After compilation, a file is generated separately to store the output. Depending upon the compiler type, the file has the name of the source CoffeeScript file and the extension as js or js.map. By default, the generated files will be stored in the folder where the original file is located. This default location can change and the generated files can be stored in the location or folder newly defined.

Step 2: Create Node.js configuration. Define settings as per requirements such as engine to use, working directory, and path of JavaScript file generated from original CoffeeScript file during compilation.

Step 3: Save the configuration.

Step 4: Run Node.js application.

Demonstrate: Next, show how to compile CoffeeScript on the fly during a run.

Say: This mode needs the register.js file, which is part of the coffee-script package that should be located inside the project. Therefore, install the CoffeeScript package on the Node.js page.

Step 1: Open the CoffeeScript file in the editor.

Step 2: Choose **Create <CoffeeScript_file_name>** on the context menu.

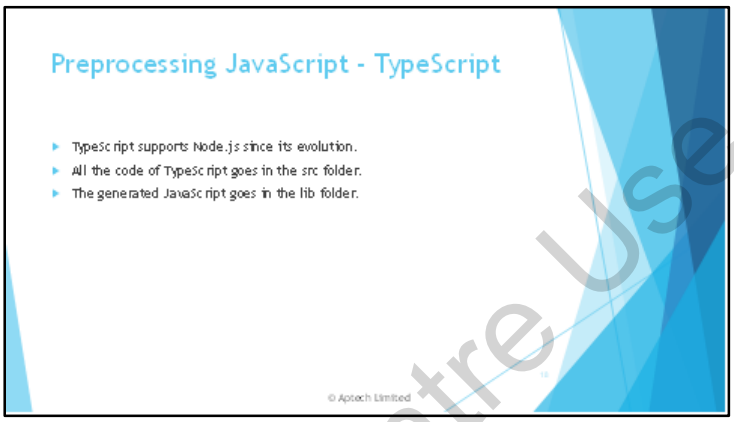
Step 2: Save the configuration and define the settings as per the requirement such as an engine to use, working directory, and the path of JavaScript file generated from the original CoffeeScript file during compilation.

Step 3: Run Node.js application.

For both these approaches, Node.js is the runtime environment for executing the application in JavaScript.

Next, we will be demonstrating TypeScript.

Slide 18
Understand use of TypeScript.



Do: Remind them of the earlier example of TypeScript that was demonstrated.

Say: TypeScript has support for Node.js since its evolution.

Demonstrate: Following are steps to set up a quick Node.js project:

Install the following packages to set up Node.js project:

```
package.json or npm init -y, npm

Add TypeScript package:
npm install typescript --save-dev

Add node.d.ts package for TypeScript options:
npm install @types/node -save-dev

Init a tsconfig.json package:

npm tsc --init --rootDir src --outDir lib --esModuleInterop --
resolveJsonModule --lib es6, dom --module commonjs
```

Say: All the code of TypeScript go in the src folder and the generated JavaScript goes in the lib folder.

Demonstrate: How to compile and run.

dd ts-node - use to compile and run in node (npm install ts-node --save-dev)
Whenever a file is changed, nodemon will invoke ts-node (npm install nodemon --save-dev).
Add a script to package.json based on the application entry.

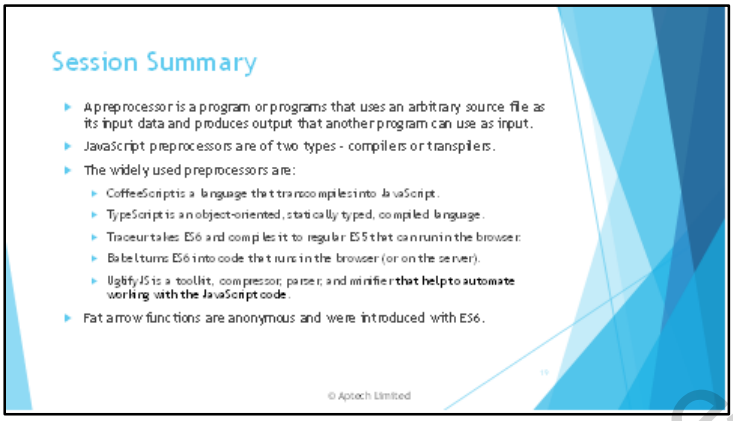
Show the index.ts contents.

```
 "scripts": {
 "start": "npm run build:live",
 "build": "tsc -p .",
 "build:live": "nodemon --watch 'src/**/*.ts' --exec 'ts-node'
src/index.ts"
 },
```

Say: The ts-node transpilers automatically pick up tsconfig.json and the ts-node runs the output JavaScript through Node.js.

JavaScript application is now ready to run ‘npm run build’.

Slide 19
Summarize the session.



Do: Ask volunteers to recap what they have learned in the session. To encourage more students to participate, ask students randomly to share one or two points about a specific preprocessor. Have the interaction not longer than five minutes. At the end, thank the students and summarize the key learning points.
Say:

- A preprocessor is a program or programs that uses an arbitrary source file as its input data and produces output that another program can use as input.
- JavaScript preprocessors are usually of two types - compilers or transpilers.
- The most popular and widely used preprocessors are CoffeeScript, TypeScript, Babel, and UglifyJS.
- CoffeeScript is a language that transcompiles (compiles from one source language to another) into JavaScript.
- TypeScript is an object-oriented, statically typed, compiled language.
- Traceur is a compiler that takes ECMAScript 6 and compiles it to regular JavaScript (ECMAScript 5) that can run in the browser.
- Babel is a transpiler for JavaScript best known to turn ES6 into code that runs in the browser (or on the server).
- UglifyJS is a parser, compressor, minifier toolkit written in JavaScript. It also contains tools that allow automating, working with JavaScript code.
- Arrow functions are a short and crisp syntax for writing function expressions in JavaScript.
- Fat arrow functions are anonymous and were introduced with ES6.

Say: Thank you for your participation in this session. The next session will be grouping and non-blocking scripts.

3.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand the grouping and non-blocking scripts that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 4 – Grouping Scripts and Non-Blocking Scripts

4.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

4.1.1 Objectives

By the end of this session, the learners will be able to:

- Explain the concept of grouping scripts through inline and external JavaScript files
- Define blocking and non-blocking JavaScript operations
- Differentiate between blocking and non-blocking operations in JavaScript

4.1.2 Teaching Skills

To teach this session successfully, you must know about grouping and non-blocking scripts. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

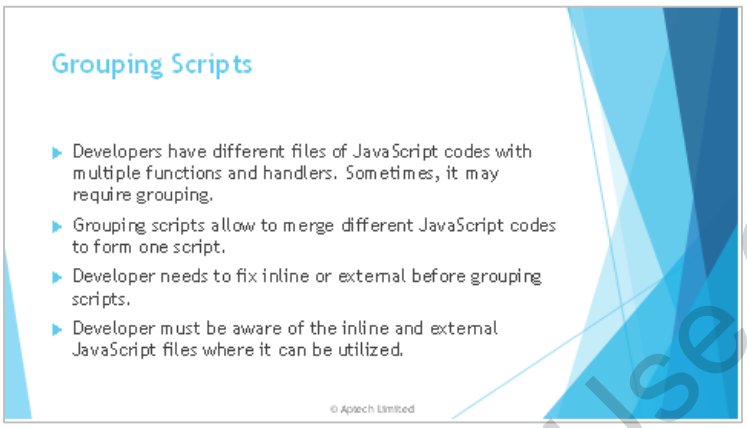
Follow the order given here during In-Class activities.

4.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the basic concepts of grouping scripts. The session explains blocking and non-blocking JavaScript operations and how they affect the UI. The session will also highlight the difference in the way both these scripts work.

4.2 In-Class Explanations

Slide 3
Understand the concept of grouping scripts.



Ask the students, what are Grouping Scripts?

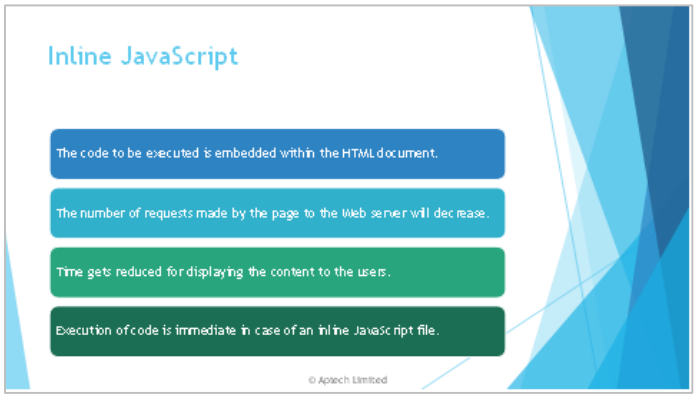
Do: Give some time for them to answer. Consolidate the responses.

Say: Tell the students that developers have different files of JavaScript codes with multiple functions and handlers. However, at certain times, they may need them to be grouped together into one script.

Grouping scripts allows to merge different JavaScript codes to form one script.
Before grouping different scripts, a developer needs to first decide whether the script file that is to be executed should be inline or external. Developers should know about inline and external JavaScript files to understand situations where each of them can be best utilized.

Slide 4

Understand inline JavaScript.



Ask: What do you know about Inline JavaScript?

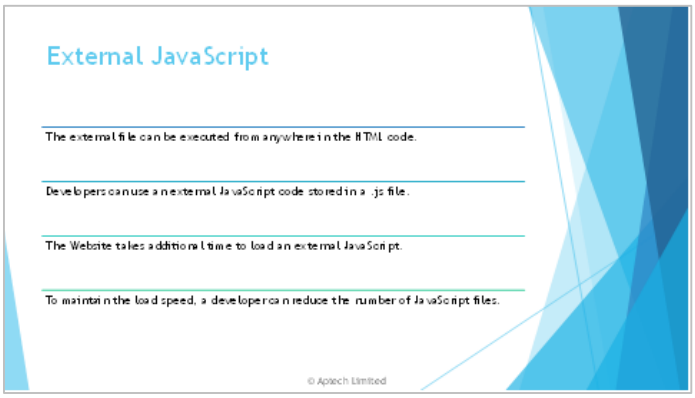
Do: Wait for response. Consolidate the responses.

Say:

- The code to be executed is embedded within the HTML document.
- The number of requests made by the page to the Web server will decrease.
- Time gets reduced for displaying the content to the users.
- Execution of code is immediate in case of an inline JavaScript file.

Slide 5

Understand external JavaScript.



Ask: What do you know about External JavaScript?

Do: Wait for response. Consolidate the responses.

Say:

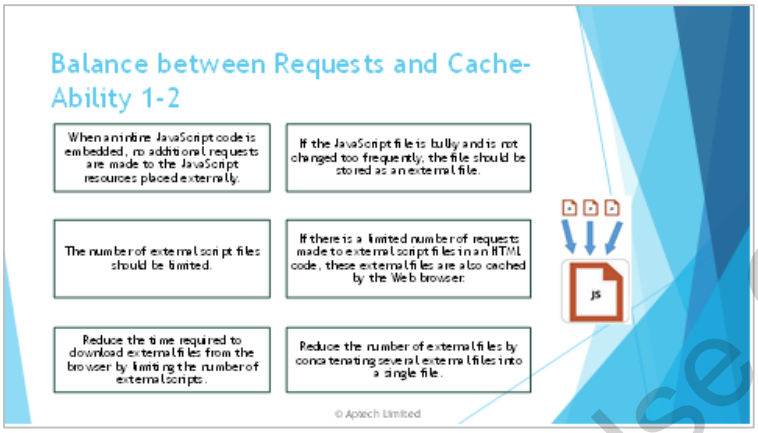
- The actual JavaScript code is placed in a file external to the HTML code.
- The external file can be executed from anywhere in the HTML code.
- The Website takes additional time to load an external JavaScript, because there are multiple files to be fetched from an external source.
- Developers can use an external JavaScript code stored in a .js file and request for all the mentioned CSS and .js files when the server processes the main HTML script.

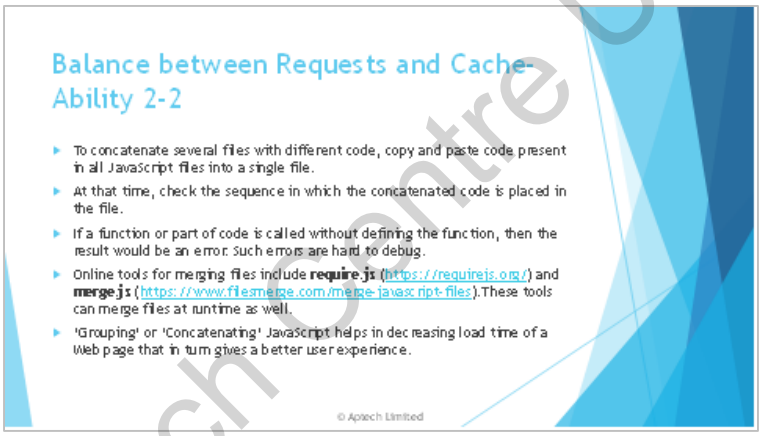
Refer to the following link for more information:

https://www.guru99.com/all-about-internal-external-javascript.html

Slides 6 and 7

Understand the balance between requests and cache-ability.





Say: When an inline JavaScript code is embedded within an HTML code, no additional requests are made to the JavaScript resources placed externally. In case the JavaScript file is bulky and is not changed too frequently, the file should be stored as an external file at a location different from the HTML code on the Web page.

The number of external script files should be finite. If there is a finite number of requests made to external script files in an HTML code, these external files are also cached by the Web browser.

Ask: How can the developer reduce the time required to download external files?

Do: Give some time for them to answer.

Say: Developer can reduce the time required to download external files from the browser by simply limiting the number of external scripts. Developers can further reduce the number of external files by concatenating several external files into a single file. Such a single file can be called with a single <script> tag.

Ask: Do you have any idea of concatenating files? How will you do it?

Do: Let the students come out with different answers, give some time for them. Consolidate the responses.

Say:

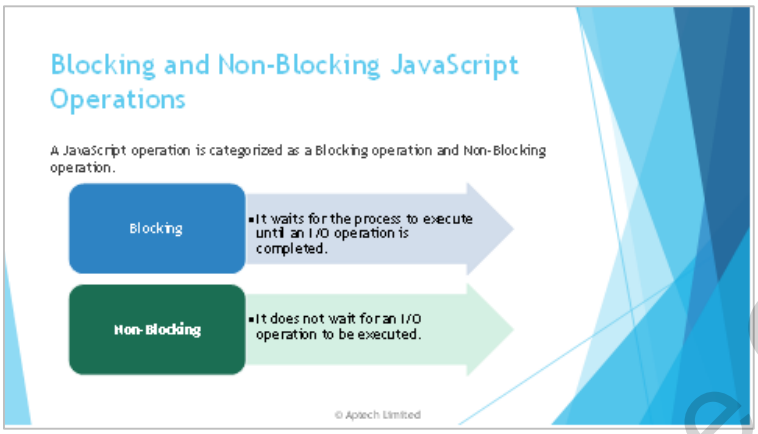
- The best possible way to concatenate several files with different code is to copy and paste the code present in all the JavaScript files into a single file. While doing so, make sure to check the sequence in which the concatenated code is placed in the file. If a function or part of code is called without defining the function, then the result would be an error. Such errors are hard to debug.
- There are online tools for merging files, such as require.js (https://requirejs.org/) and merge.js (https://www.filesmerge.com/merge-javascript-files). These tools can merge files at runtime as well.
- For example, developers can create a 'main.js' file which calls all other scripts and then, invoke the main using require.js as follows:

```
<script data-main="scripts/main.js"
 src="scripts/require.js"></script>
```

- 'Grouping' or 'Concatenating' JavaScript helps in decreasing the load time of a Web page that in turn gives a better user experience.

Slide 8

Understand blocking and non-blocking JavaScript operations.



Say: Next, we will discuss about blocking and non-blocking operations.

A JavaScript operation is categorized as a Blocking operation and Non-Blocking operation. If it waits for the process to execute until an I/O operation is completed. On the other hand, a non-blocking operation does not wait for an I/O operation to be executed.

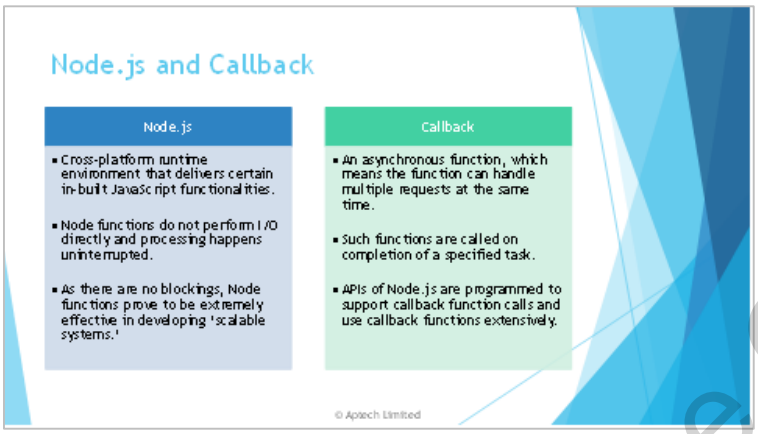
Before understanding the concept of blocking and non-blocking operations in JavaScript, let us see what 'node.js' and 'callback' are and how they are implemented for blocking and non-blocking operations in JavaScript.

Refer to the following link for more information:

https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/

Slide 9

Understand Node.js and Callback.



Ask: What is Node.js?

Do: Consolidate the responses.

Say: Tell the students that Node.js is a cross-platform runtime environment that provides certain in-built JavaScript functionalities.

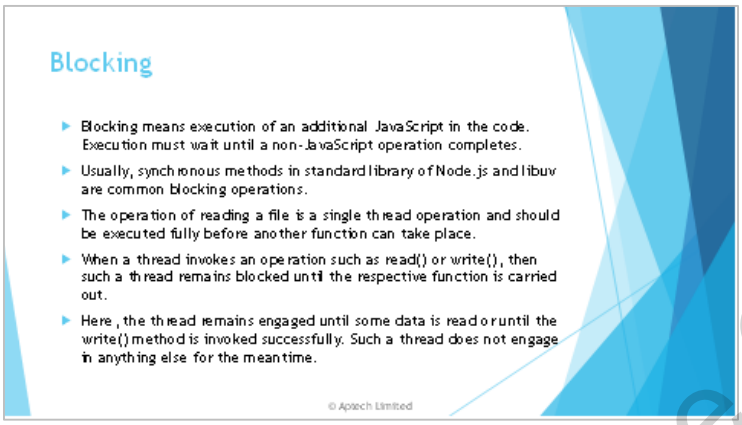
Usually, Node functions do not perform I/O directly and hence, the processing happens uninterrupted. As there are no blockings, Node functions prove to be extremely effective in developing ‘scalable systems’.

Ask: Do you have any idea about callback function?

Do: Consolidate the responses.

Say: A callback is an asynchronous function, which means the function can handle multiple requests at the same time. Such functions are called on completion of a specified task. APIs of Node.js are programmed to support callback function calls and use callback functions extensively.

Slide 10
Understand blocking.



Say: Talk about Blocking. Explain the following:

Blocking
Blocking means execution of an additional JavaScript in the code. Execution must wait until a non-JavaScript operation completes. Typically, delay in code execution occurs when an event loop (the order in which the code is executed) is unable to process JavaScript when a blocking operation is in progress.

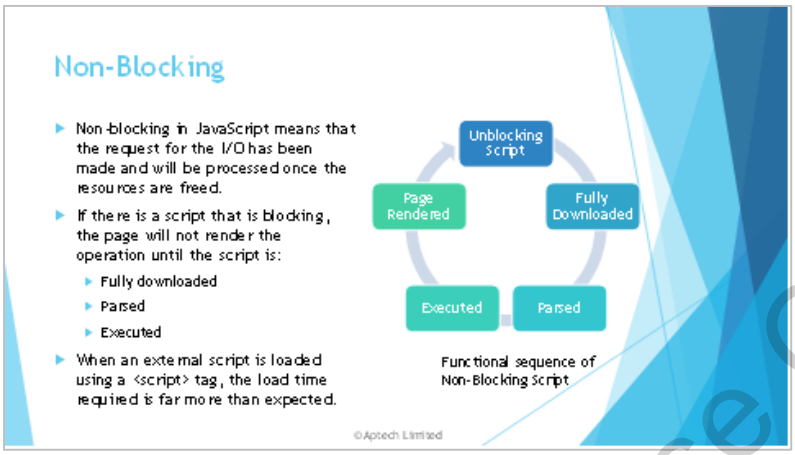
Usually, synchronous methods in standard library of Node.js and libuv (I/O library with in-built event loops) are common blocking operations.

At times, in Node.js, CPU-intensive scripts of JavaScript perform I/O functions poorly. Such functions are not referred to as blocking.

Note: 'I/O' signifies the interaction between the network that is supported by libuv and the system disk.

The operation of reading a file is a single thread operation and should be executed fully before another function takes place. When a thread invokes an operation such as read() or write(), then such a thread remains blocked until the respective function is carried out. Here, the thread remains engaged until some data is read or until the write() method is invoked successfully. Such a thread does not engage in anything else for the meantime.

Slide 11
Describe non-blocking.



Say: Talk about Non-Blocking. Explain the following:

Non-blocking in JavaScript means that the request for the I/O has been made and will be processed once the resources are freed. There are more than one way to load JavaScript in a non-blocking manner. The most basic approach to do so is to dynamically create a script node.

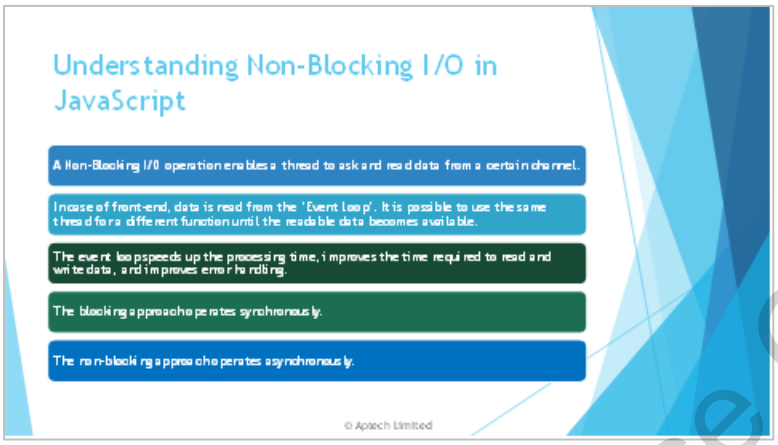
Do: Point out to the picture.

Say: If there is a script that is blocking, the page will not render the operation until the script is:
- Fully downloaded
- Parsed
- Executed

The load time of a blocked JavaScript is roughly equivalent to the amount of time a browser requires to render an operation. When an external script is loaded using a <script> tag, the load time required is far more than expected.

Slide 12

Understand Non-blocking I/O in JavaScript.



Ask: Can anyone list the functional sequence of Non-Blocking Script?

Do: Consolidate responses from them.

Answer: Unblocking Script, Fully Downloaded, Parsed, Executed, Page Rendered.

Say: Tell them that, at times, certain data is reserved for use of a process and no other process is allowed access to it. Though the data might or might not be in interaction with the disk, it would still block operation over I/O. Input as well as output, both kinds of operations are blocked.

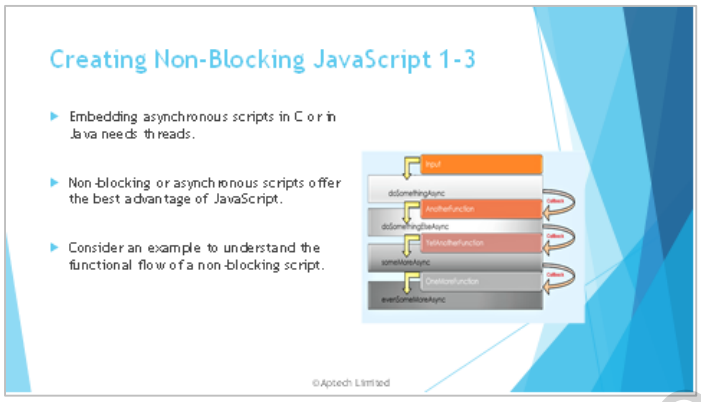
A non-blocking I/O operation enables a thread to ask and read data from a certain channel. In case of front-end, data is read from the 'Event Loop'. If data is available, then it is fetched; otherwise, nothing is gathered.

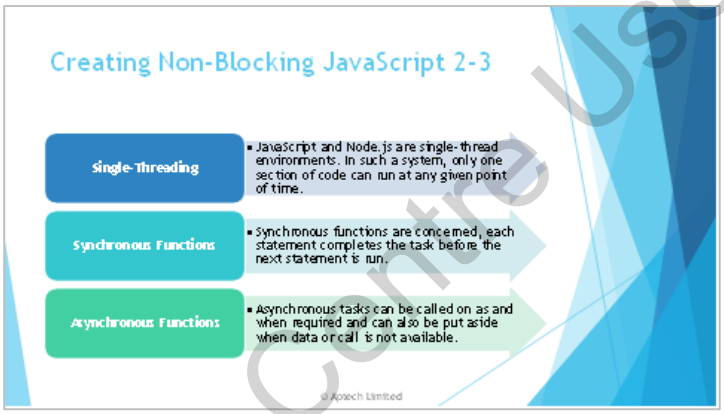
It is possible to use the same thread for a different function until the readable data becomes available. Threads, when not blocked in I/O calls, perform I/O on other request or on other channels present in the event loop. The event loop speeds up the processing time, improves the time required to read and write data, and improves error handling.

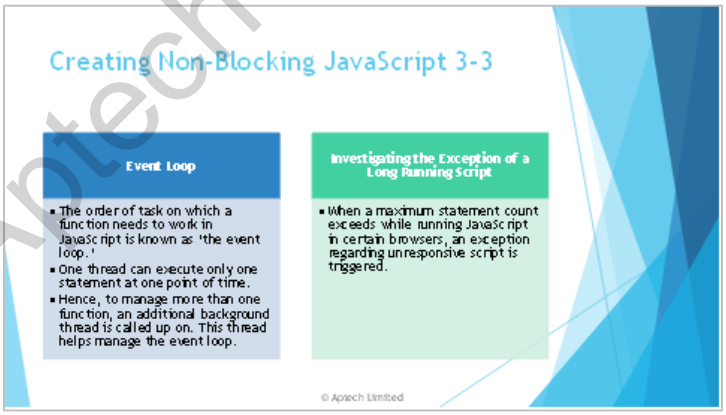
The blocking approach operates synchronously, that is, it finishes the full process, from the beginning till the end, in one go. The non-blocking approach operates asynchronously. The non-blocking approach allows sending and receiving callbacks as tokens to and from where their operations are performed.

Slides 13 to 15

Understand how to create non-blocking JavaScript.







Say: Tell them that embedding asynchronous scripts in C or in Java needs threads, which can prove to be challenging. Non-blocking or asynchronous scripts offer the best advantage of JavaScript.

Do: Show the picture and explain the flow.

Input→ Another function→Yet another function→One more function.

Ask: What do you mean by Single threading?

Do: Consolidate the responses.

Answer: JavaScript and Node.js are single-thread environments. In such a system, only one section of code can run at any given point of time.

Ask: What is Synchronous function?

Do: Consolidate the responses.

Answer: Synchronous functions are concerned. Each statement completes the task before the next statement is run. This results in the program being evaluated in the exact order of appearance of each statement, function, and command.

Ask: What is Asynchronous function?

Do: Consolidate the responses.

Answer: Asynchronous tasks can be called on as and when required and can also be put aside when data or call is not available. Synchronous functions usually are not difficult to execute as the latest browsers and most event handlers execute them instantly, resulting in fast execution of operations. When the task is to make an API call or query third-party servers for data, then asynchronous functions serve better than synchronous functions do.

Ask: What is meant by Event Loop?

Do: Consolidate the responses.

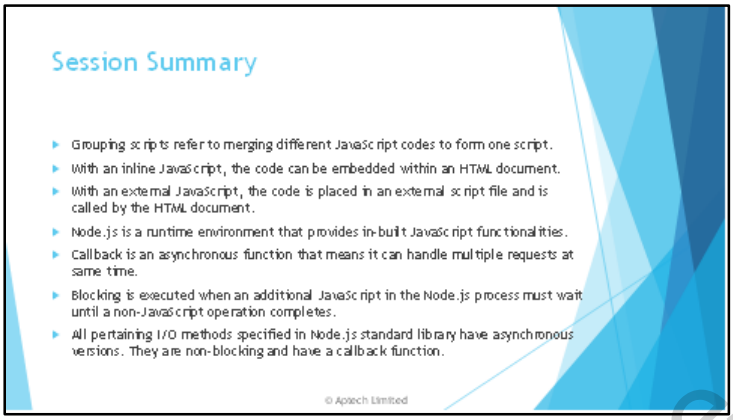
Answer: The order of task on which a function needs to work in JavaScript is known as 'the event loop'. One thread can execute only one statement at one point of time. Hence, to manage more than one functions, an additional background thread is called upon. This thread helps manage the event loop. However, the code always runs a single main thread, hence, JavaScript is said to be single-threaded.

Say: Investigating the Exception of a Long Running Script means when a maximum statement count exceeds while running JavaScript in certain browsers, an exception regarding unresponsive script is triggered.

For example, whenever the JavaScript engine of Internet Explorer 8 executes more than five million statements as part of an event thread, it triggers such an exception. This exception informs users that a script on the page is causing Internet Explorer to run slowly. The user can stop the script and get back control on the speed of the system.

To stop such exceptions from appearing, a code would need to be written. The exception is handled to terminate certain JavaScript operations from affecting the UI where all user actions and events must run in a sequence. Hence, asynchronous functions work better in such scenarios as a thread is yielded periodically and is active only when the UI thread is inactive. Asynchronous functions can provide relief in situations where lengthy scripts must be run on client end.

Slide 16
Summarize the session.



Do: Show the slide to the students and tell them that, we are going to revise the points covered. Now, you can list the points one by one.

- Grouping scripts refer to merging different JavaScript codes to form one script.
- With an inline JavaScript, the code can be embedded within an HTML document.
- With an external JavaScript, the code is placed in an external script file and is called by the HTML document.
- Node.js is a runtime environment that provides in-built JavaScript functionalities.
- Callback is an asynchronous function, which means it can handle multiple requests at same time.
- Blocking is executed when an additional JavaScript in the Node.js process must wait until a non-JavaScript operation completes.
- All pertaining I/O methods specified in Node.js standard library have asynchronous versions. They are non-blocking and have a callback function.

Say: Thank you for your participation in this session. The next session will be interacting with html objects using JavaScript.

4.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand how to interact with html objects using JavaScript that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 5 – Interacting with HTML Objects Using JavaScript

5.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

5.1.1 Objectives

By the end of this session, the learners will be able to:

- Describe data access in brief
- Describe how JavaScript events work with HTML objects
- Describe how HTML Document Object Model (DOM) elements work on JavaScript
- Explain how JavaScript is added in HTML DOM
- Define JSON and its uses

5.1.2 Teaching Skills

To teach this session successfully, you must know about HTML Objects and interacting with them using JavaScript. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

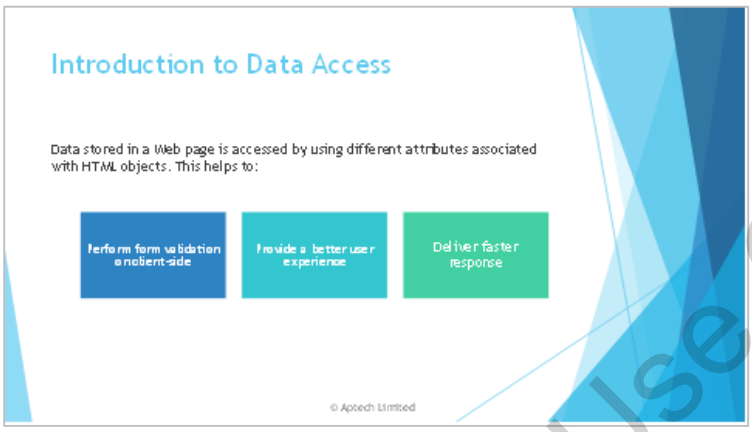
Follow the order given here during In-Class activities.

5.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the data access. The session explains how to interact with HTML objects using JavaScript and the purpose of JSON in JavaScript.

5.2 In-Class Explanations

Slide 3
Understand the concept of data access.



Ask: Have they heard of data access?

Do: Wait for response. Consolidate the responses.

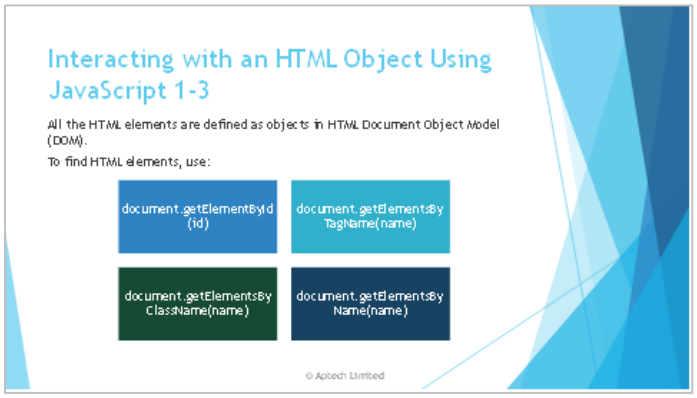
Say: JavaScript can access data stored in a Web page by using different attributes associated with HTML objects. This helps to perform form validation on client-side giving a better user experience, as response is faster when compared to fetching data from the database.

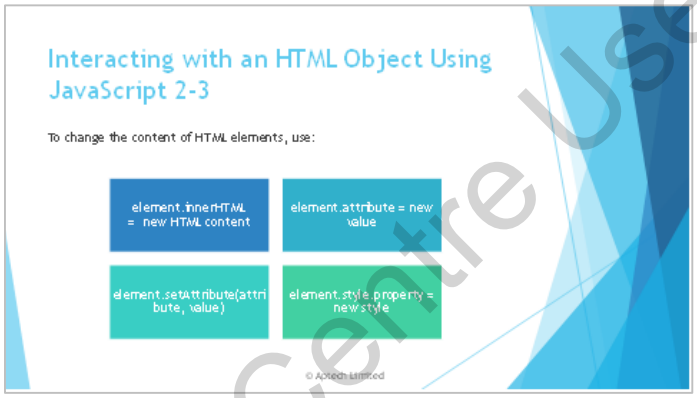
Do: Write the script on the whiteboard:

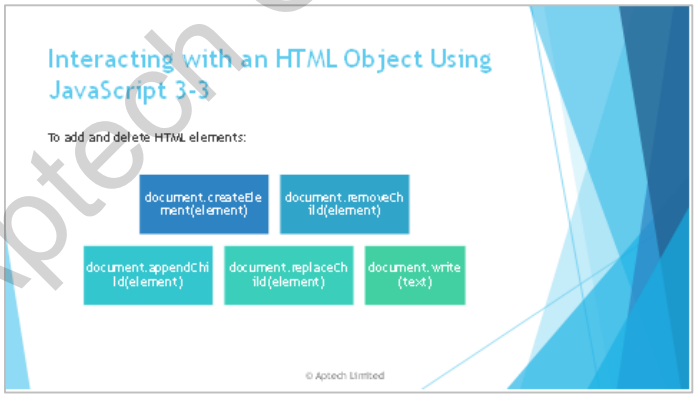
```
<script>
function getSport(sport) {
  var sportName = sport.getAttribute("data-sport-name");
  alert(sportName);
}
</script>
<a href="#" onClick="getSport(this)" data-sport-
name="Football">Ronaldo</a>
<a href="#" onClick="getSport(this)" data-sport-
name="Tennis">Federer</a>
```

Slides 4 to 6

Understand how to interact with HTML objects.







Say: The HTML Document Object Model (DOM) is created when an HTML page is loaded in the browser and all HTML elements are defined as objects in DOM. To find HTML elements use:

- document.getElementById(id): Finds an element by element id
- document.getElementsByTagName(name): Finds elements by tag name
- document.getElementsByClassName(name): Finds elements by class name
- document.getElementsByName(name): Finds elements by name

Do: Ask the class to come up some examples for each of these.

Say: To change the content of HTML elements, use the following:

element.innerHTML = new HTML content: Updates inner HTML of an element
element.attribute = new value: Updates attribute new value of an HTML element
element.setAttribute(attribute, value): Updates attribute value of an HTML element
element.style.property = new style: Updates style of an HTML element

Do: Write an example of the code on the whiteboard:

Using element.innerHTML = new HTML content Method

```
<html>
<body><p id="Para">Hello Script! </p>
<script>
document.getElementById ("Para").innerHTML = "New text!";
</script>
</body>
</html>
```

Ask: The learners to write an example of the remaining functions.

Say: To add and delete HTML elements:

document.createElement(element): Helps in creating an HTML element
document.removeChild(element): Helps in removing an HTML element
document.appendChild(element): Helps in adding an HTML element
document.replaceChild(element): Helps in replacing an HTML element
document.write(text): Helps in writing into the HTML output stream

Do: Give examples of the codes and write them on the whiteboard:

Using document.removeChild(element) Method

```
<!DOCTYPE html>
<html>
<body>
<ul id="Writing tools">
  <li>Pen</li>
  <li>Pencil</li>
</ul>
<p>Click the button to append an item to the end of the
list.</p>
<button onclick="myMethod()">Append</button>
<script>
function myMethod() {
  var node = document.createElement("LI");
  var textnode = document.createTextNode("Color pencils");
```

```
  node.appendChild(textnode);
  document.getElementById("Writing tools").appendChild(node);
}
</script>
</body>
</html>
```

Using document.removeChild(element) Method

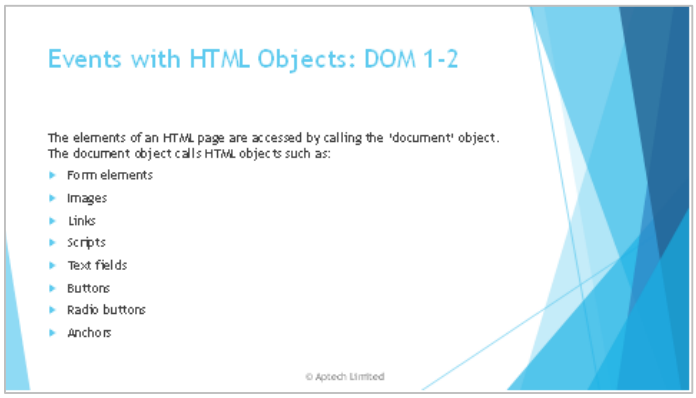
```
<!DOCTYPE html>
<html>
<body>
<ul id="Writing tools">
 <li>Pen</li>
 <li>Pencil</li>
 <li id="cp">Color Pencils</li>
</ul>
<p>Click the button to delete an item in the list.</p>
<button onclick="myMethod()">Delete</button>
 <script>
function myMethod() {
var ele = document.getElementById("cp");
ele.parentNode.removeChild(ele)
}
</script>
</body>
</html>
```

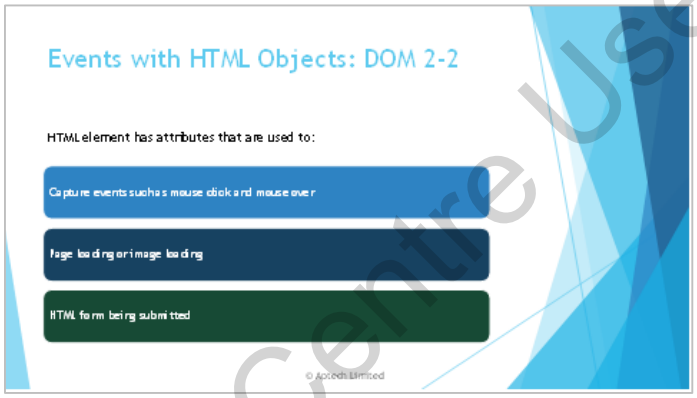
Refer to the following link for more information:

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Slides 7 and 8

Understand the external JavaScript.





Say: All the elements of an HTML page can be accessed by calling the 'document' an object. HTML objects such as form elements, images, links, scripts, text fields, buttons, radio buttons, and anchors can be called by the document object.

Say: Every HTML element has attributes that can be used to capture the events such as mouse click and mouse over. Page loading or image loading and an HTML form being submitted are examples of HTML events triggered when user clicks the mouse.

Do: Write the script on the whiteboard:

```
<!DOCTYPE html>
<html><body>
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-
color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
<script>
function mOver(obj) {
  obj.innerHTML = "Cursor is on DIV"
}
function mOut(obj) {
```

```
  obj.innerHTML = "Bring Cursor here"
}
</script>
</body>
```

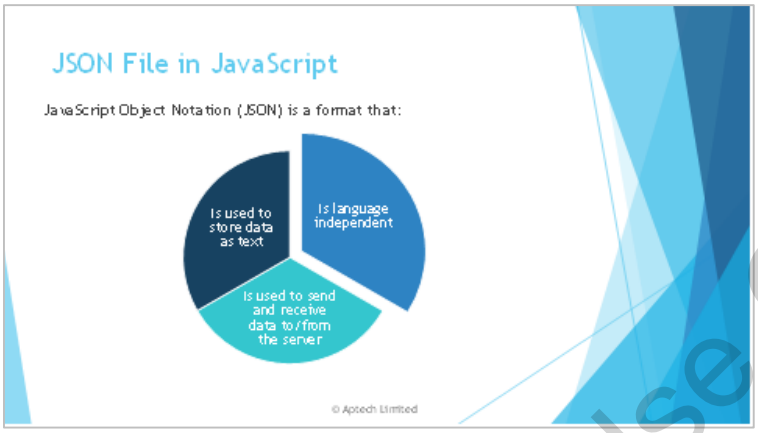
Say: This is an example that demonstrates using onmouseover event. Methods help to fetch the content of an HTML element by using its ID or name. 'document.getElementById' (ID) and 'getElementsByTagName' (name) are commonly used methods to obtain the data from an element.

Refer to the following link for more information:

https://www.w3schools.com/js/js_events.asp

Slide 9

Understand the JSON file in JavaScript.



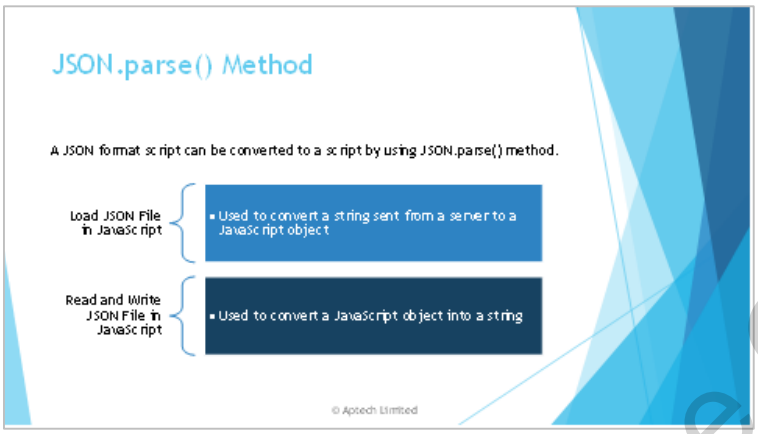
Say: JavaScript Object Notation (JSON) is a format used to store data as text. It is language independent and is commonly used to send and receive data to/from the server. Thus, it can also be used to send/receive scripts or data for HTML elements to/from the server.

Do: Write the script on the whiteboard:

```
<!DOCTYPE html>
<html><body>
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-
color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
<script>
function mOver(obj) {
  obj.innerHTML = "Cursor is on DIV"
}
function mOut(obj) {
  obj.innerHTML = "Bring Cursor here"
}
</script>
</body>
```

Say: This is an example that demonstrates using onmouseover event.
Methods help to fetch the content of an HTML element by using its ID or name.
'document.getElementById' (ID) and 'getElementsByTagName' (name) are commonly used methods to obtain the data from an element.

Slide 10
Understand the JSON.parse() Method.



Say: If a script is received from the server in JSON format, it can be converted to a script by using JSON.parse() method. There are two methods:

1. Load JSON File in JavaScript where JSON.parse() is used to convert a string sent from a server to a JavaScript object.
2. Read and Write JSON File in JavaScript where to send a script to the server, it should be converted to a string format.

Do: Write an example as:
The string *{"Name": "White Stork", "Species": "Bird","Type": "Migrant"}* can be converted as follows:

```
var toConvert = JSON.parse('{"Name": "White Stork", "Species":
"Bird", "Type": "Migrant" )};

And JSON.stringify() method can be used to convert a JavaScript
object into a string.

var toConvert = {Name: "White Stork", Species: "Bird", Type:
"Migrant" };
var toJSON = JSON.stringify(toConvert);
```

Do: Write the complete code on the whiteboard:

```
<!DOCTYPE html>
<html>
<body>
<p id="migrantbirds"></p>
<script>
var toConvert = {Name: "White Stork", Species: "Bird", Type:
"Migrant"};
var toJSON = JSON.stringify(toConvert);
```

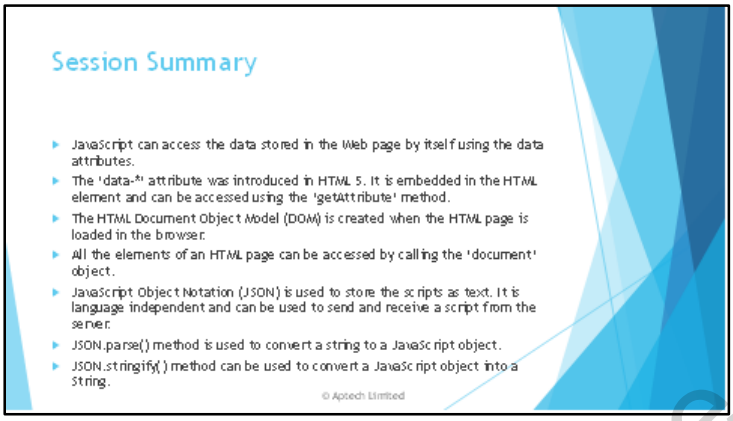
```
document.getElementById("migrantbirds").innerHTML= toJSON;
</script>
</body>
</html>
```

Say: The outcome of this is that the string {"Name":"White Stork", "Species": "Bird", "Type": "Migrant"} will be stored in the variable, to JSON.

Refer to the following link for more information:

https://www.w3schools.com/js/js_json_files.asp

Slide 11
Summarize the session.



Do: Ask for volunteers to recap what they have learned in the session. To encourage all students to participate, ask students randomly to share one or two points about a specific developer tool. Have the interaction not longer than five minutes. At the end, thank the students and summarize the key learning points.

Say:
- JavaScript can access the data stored in the Web page by itself using the data attributes.
- The ‘data-*’ attribute was introduced in HTML 5. It is embedded in the HTML element and can be accessed using the ‘getAttribute’ method.
- The HTML Document Object Model (DOM) is created when the HTML page is loaded in the browser.
- All the elements of an HTML page can be accessed by calling the ‘document’ object.
- JavaScript Object Notation (JSON) is used to store the scripts as text. It is language independent and can be used to send and receive a script from the server.
- JSON.parse() method is used to convert a string to a JavaScript object.
- JSON.stringify() method can be used to convert a JavaScript object into a String.

Say: Thank you for your participation in this session. The next session will be JavaScript Minification.

5.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand JavaScript Minification that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 6 – JavaScript Minification

6.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

6.1.1 Objectives

By the end of this session, the learners will be able to:

- Explain what JavaScript minification is
- Explain how JavaScript minification process works
- List the benefits of JavaScript minification
- Explain the online tools available for JavaScript minification process
- Explain in detail how Google Closure Compiler tool for JavaScript minification works

6.1.2 Teaching Skills

To teach this session successfully, you must know about JavaScript minification and online tools available for minification. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

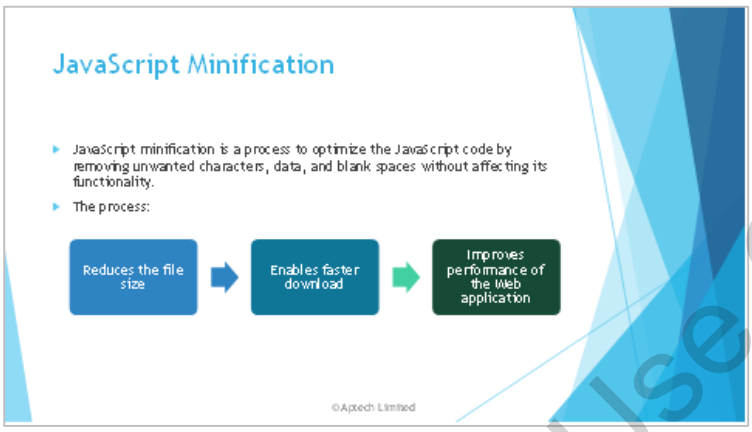
Follow the order given here during In-Class activities.

6.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the basic concepts of JavaScript minification. The session explains the features, tools, and benefits of minification process. It also describes the working of Google Closure Compiler tool, which is commonly used for JavaScript minification.

6.2 In-Class Explanations

Slide 3
Understand the concept of JavaScript Minification.



Ask the students, what do you understand by the term Minification?

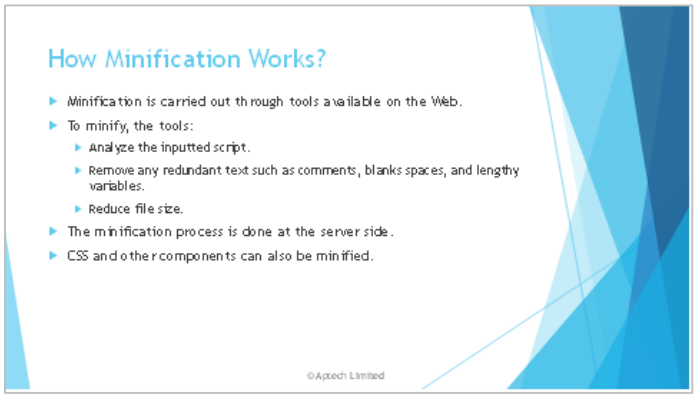
Do: Wait for the response and consolidate the responses.

Say: JavaScript minification is a process of optimizing the JavaScript code by removing unwanted characters, data, and blank spaces without affecting its functionality. This process reduces the file size of the code, enabling faster download of the file, which in turn improves the performance of the Web application. Therefore, the minification process is highly beneficial to Websites that have multiple user interactive functionalities incorporated through JavaScript codes.

Refer to the following link for more information:
https://blog.stackpath.com/glossary/minification/

Slide 4

Understand how minification works.

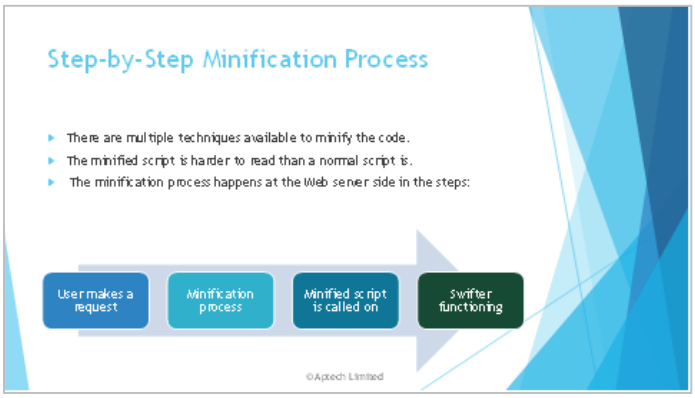


Say: There are many tools available on the Web that help minify a JavaScript code. These tools will take the code as the input and provide the minified code as the output. To minify, the tools analyze the inputted script, remove any redundant text, such as comments, blank spaces, and lengthy variables, and thereby, reduce the file size. This minification process is done at the server side before the response is sent to the browser.

CSS and other components that are utilized by Web browser to render a Website can also be minified.

Slide 5

Understand external JavaScript.



Ask: Let us consider a real-life example. You are asked to shorten an essay to make the words fit in one page of paper. What will you remove from the passage to make this happen?

Do: Look for responses such as 'reduce some words', 'shorten the essay', 'remove unnecessary words' and so on. Keep writing these kinds of responses on the white or blackboard. Take responses from few volunteers. Thank the students for the responses and explain the different techniques of minification.

Say: There are multiple techniques available to minify the code, such as shortening the variable names, removing the whitespaces, and making the functions crisper. Essentially, all techniques of minification ensure that the functionality of the script is kept intact, which in turn ensures that the Web browser will function as required.

Ask: What do you think would be drawbacks of minification?

Do: Wait for response. Consolidate the responses.

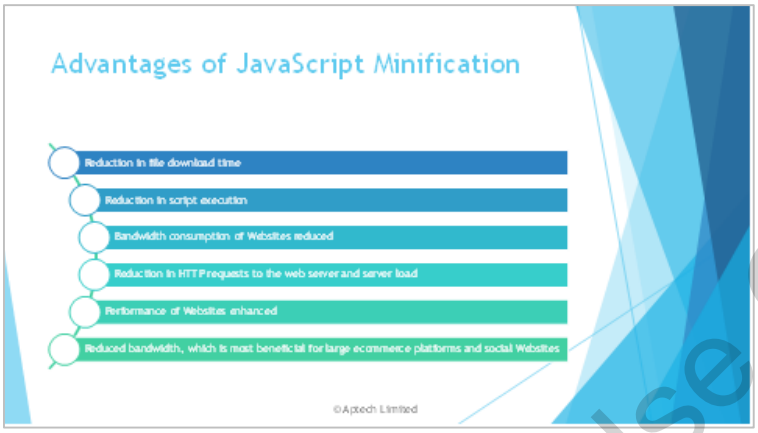
Say: The minified script is harder to read than a normal script is. The minification process happens at the Web server side that deploys the file and responds to a request from the end user.

Do: Explain the steps involved in minification process as:

- User makes a request: A user makes a request from a Web browser to a Web page.
- Minification Process: Web server lets the page be processed by an online Minification tool.
- Minified script is called on: Web server deploys the minified file and responds to the browser.
- Swifter functioning: The Web page functions as expected using a lesser bandwidth.

Slide 6

Understand the advantages of JavaScript Minification.



Ask: So, why should we adopt JavaScript minification?

Do: Wait for the responses. Consolidate the responses.

Do: Explain the advantages of JavaScript minification.

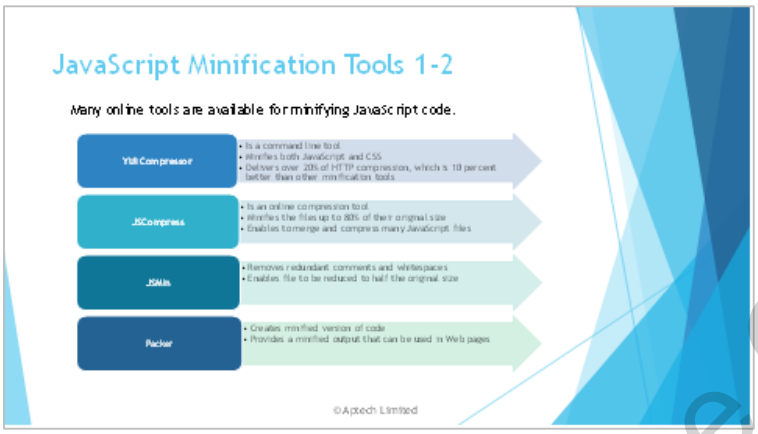
Say: Benefits of minification are:

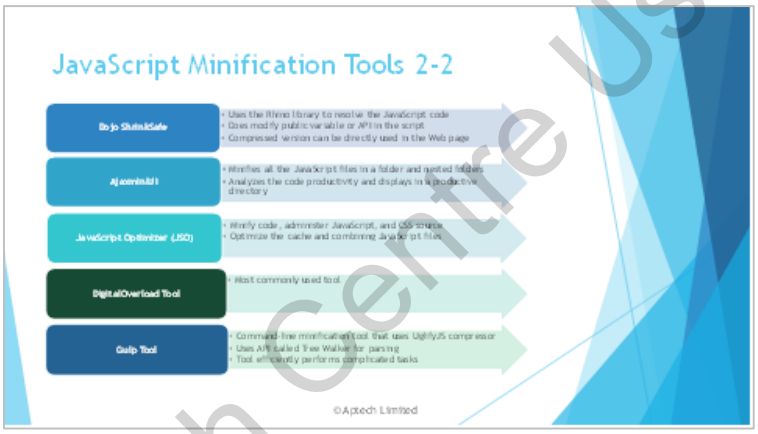
- The download time of a Web page is decreased as the file size is reduced by removing unwanted text from the code.
- The script execution time is reduced.
- The bandwidth consumption of the Website is reduced.
- The number of HTTP requests to the Web server is reduced, which reduces the server load and enables more users to access the Website simultaneously.

As Websites now need to download smaller files, they load faster and use lesser bandwidth. Therefore, the performance of Websites is enhanced due to minification. The benefit accrued by a reduced bandwidth usage could be minimal for Websites that have only few users.

However, the reduction in bandwidth usage due to minification makes a significant difference for large e-commerce platforms and social Websites, such as Facebook and Twitter.

Slides 7 and 8
Understand the JavaScript Minification tools.





Say: Many online tools are available for minifying JavaScript code. The primary goal of JavaScript minification is to retain the functional quality of the code while reducing the file size. As part of the HTTP protocol, usually, all JavaScript and CSS files transferred by Web servers are zipped.

Do: Explain the functionality of each tool.

Say:
YUI Compressor
The YUI Compressor is a command-line tool developed by Yahoo. It is written in Java and can be used to minify both JavaScript and CSS. The compression ratio is much better than any other JavaScript minification tool. YUI Compressor delivers over 20% of HTTP compression, which is 10 percent better than what other minification tools offer.

JSCompress
JSCompress is an online compression tool that is used to minify the files up to 80% of their original size. Several JavaScript files can be merged and compressed by uploading them to the JSCompress site.

JSMin
The JSMin JavaScript minifying tool is used to remove redundant comments and whitespaces from JavaScript files. The file size of a JavaScript file can be reduced to half its original size, which helps in downloading a Web page faster.

Packer
Packer is a familiar JavaScript minifying tool. It creates a minified version of the code. The Web developer can paste the original code in the minifier toolbox and click 'Pack' button to get the minified output. This output can be used in the Web page.

Do: Explain briefly about each of the tools.

Dojo ShrinkSafe
The Dojo ShrinkSafe uses the Rhino library to resolve the JavaScript code. It does not modify a public variable or an API used in the script. So, the compressed version of the JavaScript code can be used as such in the Web page.

AjaxminUi
AjaxminUi minifies all the JavaScript files in a folder and nested folders. The minifying operation is charged in the background. The tool also analyzes the code productivity and shows the results in a productive directory at the end of the minifying operation.

JavaScript Optimizer (JSO)
Apart from minifying the code, the JSO can administer JavaScript and CSS source. It can even reduce the exchange of data between the server and the client cache. It helps in optimizing the cache and combining JavaScript files.

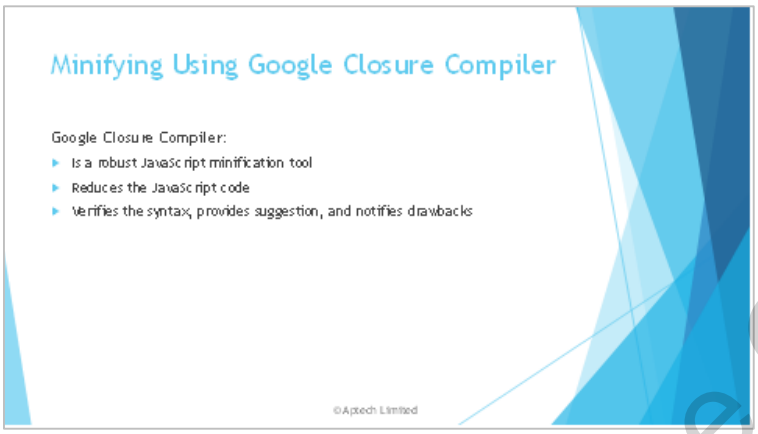
DigitalOverload Tool
The most commonly used minification tool is the JavaScript minifier, DigitalOverload. The results are exclusive, tactical, and well prepared.

Gulp Tool
Gulp is an easy to use command-line minification tool that uses the UglifyJS compressor. The compressor provides an API called Tree Walker that parses through every node of the JavaScript code. Thus, the tool efficiently performs complicated tasks.

Refer to the following link for more information:
https://www.fusioncharts.com/blog/5-excellent-javascript-minification-tools-to-improve-your-code-performance/

Slide 9

Understand how to use Google Closure Compiler.



Ask: What is the work of compiler?

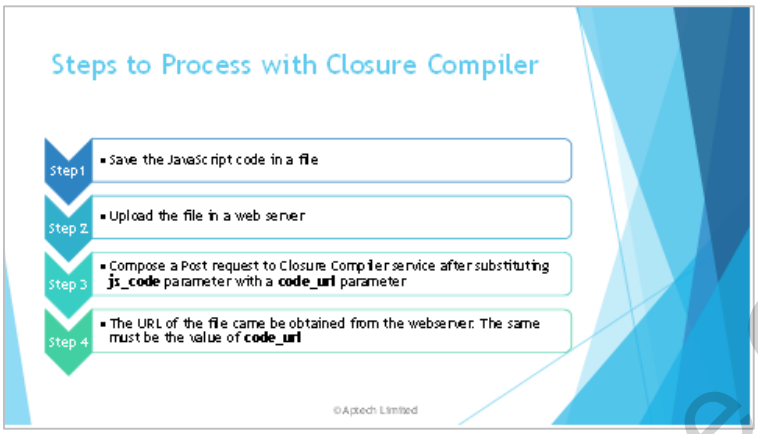
Do: Wait for response. Consolidate the responses.

Say: Google Closure Compiler is a robust JavaScript minification tool. It reduces the JavaScript code by assessing it, eliminating the dull code, modifying badly written codes, and removing the leftovers. It verifies the syntax, provides suggestions, and notifies the drawbacks in the code.

Refer to the following link for more information:
https://developers.google.com/closure/compiler/

Slide 10

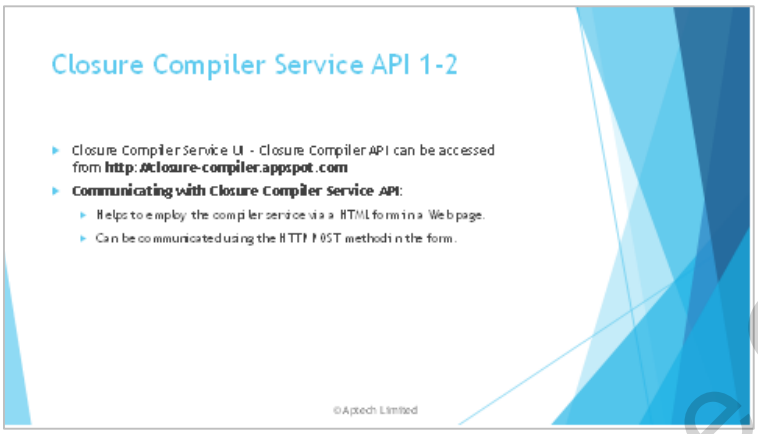
Understand how to process with Closure Compiler.

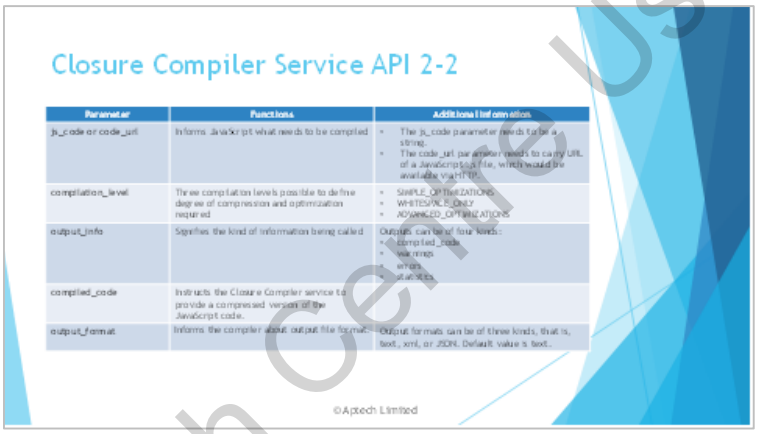


Say: There are a few steps to be followed to process a file in Closure Compiler.

Do: Show the image in the slide and explain the steps.

Slides 11 and 12
Understand the Closure Compiler Service API.





Say: Closure Compiler Service UI uses Closure Compiler API. The Closure Compiler API can be accessed from http://closure-compiler.appspot.com.

The Closure Compiler service API helps to employ the compiler service via a HTML form in a Web page. Closure Compiler service can be communicated using the HTTP POST method in the form.

Say: With each request, atleast one parameter has to be sent to communicate with Closure Compiler Service API. Let us learn about different parameters, what they do, and some additional information.

Do: Show the table on the slide and explain as follows:

- Parameter ***js_code or code_url:*** The value of this parameter informs JavaScript what needs to be compiled. At least one of these parameters, if not both, must be included in the request.
  The js_code parameter needs to be a string; for example, alert('hello').
  The code_url parameter needs to carry URL of a JavaScript .js file, which would be available via HTTP.

- ***compilation_level:*** Three possible compilation levels can be applied to the JavaScript code that defines the degree of compression and optimization required.. They are SIMPLE_OPTIMIZATIONS, WHITESPACE_ONLY, ADVANCED_OPTIMIZATIONS.

- ***output_info*****:** The value of this parameter signifies the kind of information being called from the compiler. Outputs can be of four kinds:
  compiled_code
  warnings
  errors
  statistics

- ***compiled_code***: It instructs the Closure Compiler service to provide a compressed version of the JavaScript code.

- ***output_format:*** *It* Informs the compiler about output file format. Output formats can be of three kinds, that is, .text, xml, or JSON. The default value of the output_format parameter is text.

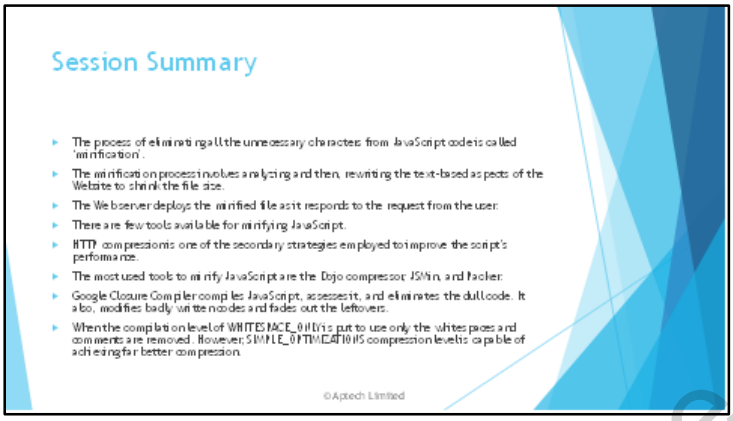
Say: When the compilation_level is WHITESPACE_ONLY, whitespaces and comments are only removed. SIMPLE_OPTIMIZATIONS compression level can achieve better compression than WHITESPACE_ONLY.

Say: To know what Is the amount of compression achieved, the data saved has to be calculated. To do this, change **output_info** in request parameters
from **compiled_code** to **compiler_statistics**.

The result will be compressed JavaScript, that is less than half in size as compared to the original code. This is inferred from the statistic obtained.
Compilation Time: 0
Actual Size: 1372
Compressed Size: 677

Slide 14

Summarize the session.



Do: Conduct a small activity to engage the students in recalling the key learning points of the session. Divide the class into two or three groups depending on the class size. Ask each group to pose a concept-related question based on what they have learned in the session to the other teams. Allow the teams to pose questions to each other in turns. Allow not more than five minutes for this activity. At the end, thank and congratulate the students for a great recap. Then, debrief presenting the summary points.

Say: Let us summarize the session as follows:

- The process of eliminating all the unnecessary characters from JavaScript source code is called 'minification'.
- The minification process involves analyzing and then, rewriting the text-based aspects of the Web site to shrink the file size. Style sheets, scripts as well as other components utilized by the Web browser to render the site can be minified.
- The Web server deploys the minified file as it responds to the request from the user. As a result, functionality is not changed while data transfer bandwidth is reduced.
- There are few tools available for minifying JavaScript.
- Apart from Minification of the JavaScript code, HTTP compression is one of the secondary strategies employed to improve the script's performance.
- The most used tools to minify JavaScript are the Dojo compressor, JSMin, and Packer. However, each of these tools has its drawbacks.
- Google Closure Compiler is another JavaScript minification tool, a real compiler that takes JavaScript to a superior level. The Closure Compiler compiles JavaScript, assesses it, and eliminates the dull code. It also, modifies badly written codes and fades out the leftovers.
- When the compilation level of WHITESPACE_ONLY is put to use only the whitespaces and comments are removed. However, SIMPLE_OPTIMIZATIONS compression level is capable of achieving far better compression.

Say: Thank you for your participation in this session. The next session will be about Web Workers.

6.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand Web Workers that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 7 – Web Workers

7.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

7.1.1 Objectives

By the end of this session, the learners will be able to:

- Define Web workers
- Differentiate between the types of Web workers
- Explain how to create a Web worker file and object
- Explain how a Web worker object is terminated and reused

7.1.2 Teaching Skills

To teach this session successfully, you must know about Web Workers. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

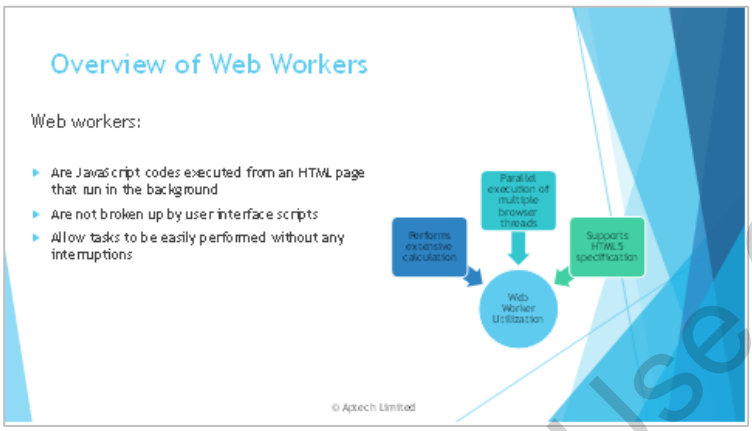
Follow the order given here during In-Class activities.

7.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the overview of Web workers. This session focusses on how a Web worker object is created, terminated, and reused.

7.2 In-Class Explanations

Slide 3
Understand concept of Web Workers.



Ask: What is the full form of W3C?

Do: Wait for the response and consolidate the response.

Say: W3C refers to the World Wide Web Consortium and WHATWG refers to Web Hypertext Application Technology Working Group. They have defined Web worker as a JavaScript script executed from an HTML page that runs in the background, independently of user-interface scripts that may also have been executed from the same HTML page.

Put in simple terms, the W3C and WHATWG see Web workers as long-running scripts that have not been broken up by user interface scripts.

Ask: What is the meaning of user interface scripts?

Do: Wait for the response and consolidate the response.

Say: User interface scripts are those scripts that respond to clicks or interactions made on the interface. Since, Web workers run scripts in the background, all other tasks can be easily performed without any interruptions to the user interface.

Do: Discuss what is Web worker utilization.

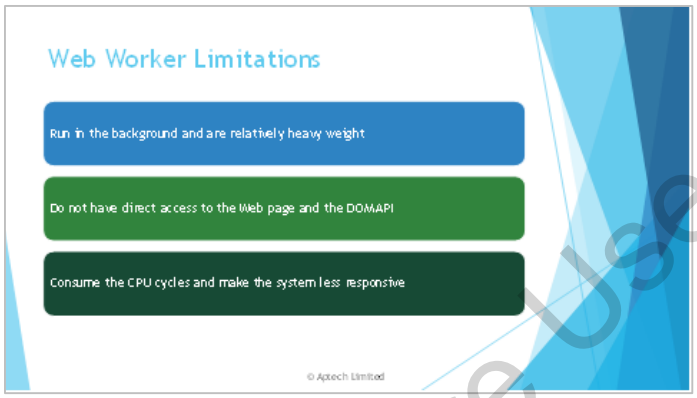
Say: Web workers are best utilized for performing an extensive calculation. Web workers use multi-core CPUs efficiently.

Web workers permit the parallel execution of multiple browser threads while one or more JavaScript threads run in the background. A browser that allows a single thread of execution will

have to wait for all JavaScript programs to be completely executed. This process takes significant time, which the programmer may not want to disclose to the user.

The specification of a Web worker is different for HTML5 specification, but can be used with HTML5 as well.

Slide 4
Understand limitations of Web Workers.

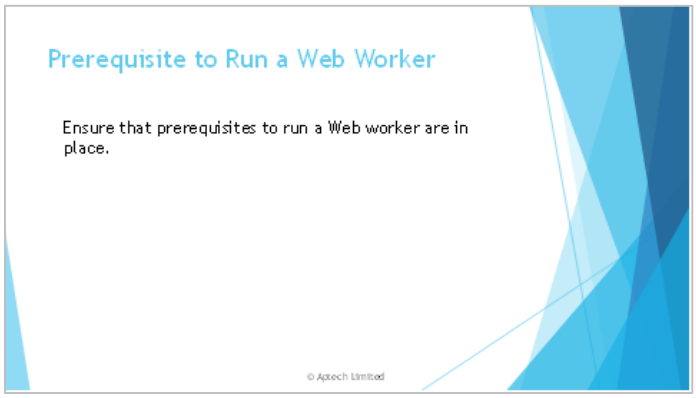


Say: Web Workers have a few limitations. They are:

- Web workers run in the background and are relatively heavy weight. They need not be used in large numbers. For example, it would be unsuitable to launch one worker for each pixel of a four-megapixel image.
- While a script is executed inside a Web worker, the Web page's window object (window.document) is not accessible, which means that Web workers do not have direct access to the Web page and the DOM API.
- Even though Web workers cannot block the UI, they can consume the CPU cycles and make the system less responsive.

Slide 5

Understand prerequisites to run a Web Worker.



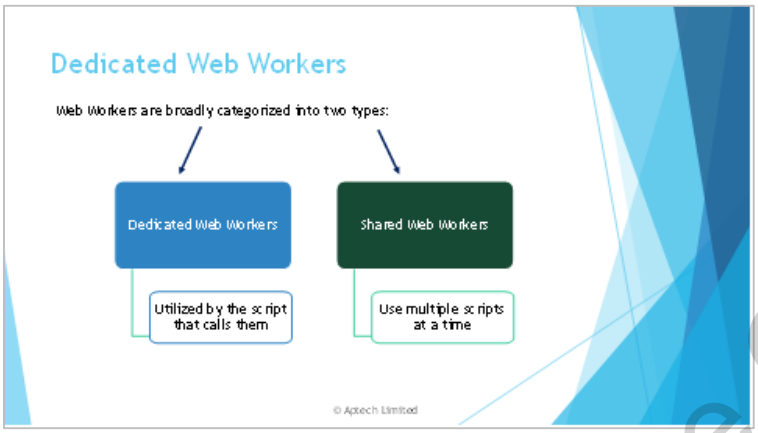
Say: Let us see what the prerequisites are to run a Web Worker.

To run a Web worker, a Web server is required. Developers can run a local server using XAMPP, which is a free and open-source Web server package.

Demonstrate using a system installation process of XAMPP. Explain every step as follows:
Step 1: To download XAMPP, go to https://www.apachefriends.org/download.html and choose the version as per your requirements (such as PHP and Operating System). Next, click Download and Open the .exe file to begin the installation process. The setup window is opened.
Step 2: Click Next and select features to be installed. Note: Most of them will be auto-selected.
Step 3: Click Next. Developer will be prompted to the folder in which XAMPP has to be installed. Developers are advised to save it to the C drive (C:\xampp).
Step 4: Uncheck the window for Bitnami for XAMPP and then, click Next.
Step 5: Click Next to begin the installation process.
Step 6: Navigate to the Start menu and search for XAMPP. Click it to launch the XAMPP Control Panel. In the Control Panel, click Start, which is beside Apache to start the server.
Step 7: To check if it XAMPP is working, start a browser and type the URL http://localhost/ as the URL. Apache dashboard will be seen. This verifies that the server is installed and running.

Slide 6

Understand Dedicated Web Workers.



Say: Web Workers are broadly categorized into two types:

- Dedicated Web Workers
- Shared Web Workers

Ask: What do you understand by Dedicated Web workers?

Do: Wait for response. Consolidate their response.

Say: Dedicated workers are Web workers utilized by the script that calls them.

Do: Demonstrate the use of dedicated Web workers by writing the following code snippet on the board.

Say: The code must be entered in the HTML file titled *WebWorkerExample.htm.* This file should be added to a Web application. Even if no Web application exists at present, placing the HTML file in the path *C:\xampp\htdocs* will make the file available to the Web server.

Demonstrate: Write the following code on the whiteboard and explain it:

```
<!DOCTYPE html>
<head>
  <title>Web Workers</title>
  <script type="text/javascript">
    window.onload=function () {
      if (typeof(Worker) === 'undefined')
      {
    alert("Web workers are not supported by this
Browser!");
    return;
      }
      else{
        var webworker = new Worker("DeepCalculation.js");
```

```
      webworker.addEventListener("message",
function (evnt) {
          alert("prime no. count "+evnt.data);
        },false);
        webworker.postMessage(2000);
      }
      };
  </script>
</head>
<body>
<h1>Working with Web Workers! It will work by itself, you do not
need to do anything.</h1>
</body>
```

Say:
In the code snippet, the <script> block comprises the event handler window.onload, which is responsible for loading the HTML page and any related files. The event handler checks whether the HTML document is ready to be executed and implements the Worker object to run a script in the background.

The 'if' conditional statement verifies whether the Web browser supports a Web worker or not. This is verified by checking the type of Worker object. If the type of worker object returned by the browser is undefined, an alert window will pop up with the message "Web workers are not supported by this Browser!". Then, the page will stop loading until the alert window is closed and all subsequent code is aborted without execution.

However, if workers are supported by the browser, then a Web worker object is defined which takes value from the corresponding script (in this case, DeepCalculation.js). The script sends some value back to the Web page. The message event handler is called by the addEventListener() method. The method implements the message event handler and accesses the value that is returned using 'evnt.data' property. Finally, the postMessage() method is invoked on the Worker object to initiate the execution of DeepCalculation.js. The postMessage() method also lets data be passed to the related script.

Do: Enter the following code in a file titled DeepCalculation.js.

```
self.addEventListener("message", function (evnt) {
  var count_prime = 0;
          for (var i = 3; i < evnt.data; i++) {
               isprime = true;
               for (var j = 2; j < i; j++) {
                    if (i % j === 0) {
                         isprime = false;
                         break;
                    }
               }
          if (isprime) {
```

```
                count_prime += 1;
                }
            }
    postMessage(count_prime);
  }, false);
```

Do: Explain the working of the code. State that this code in DeepCalculation.js file, runs in background.

Say: The code implements the message event handler of the worker thread. The message event is called on when the main page calls the postMessage() method. The message event handler executes a huge process by running a 'for' loop. Once the loop is completed, the code calls postMessage() to pass the result of processed data back to the main page. In the code, the number of prime numbers between 0 to 2000 are calculated. The total number of prime numbers are passed through count_prime variable to postMessage() to be executed on the main page. The output of the DeepCalculation.js will be displayed as an alert on the browser.

Say: Now, we move on to execution of the code.
For executing the code, the HTML file titled *WebWorkerExample.html* and JavaScript file *DeepCalculation.js* must be pasted in C:\xampp\htdocs. Developers can also create a new folder under *htdocs* and place both files in it. To execute the code, run the server using XAMPP Control Panel. In the browser, enter the URL http://localhost/WebWorkerExample.html and the worker will be executed.

Ask: What do you understand by the term shared Web server?

Do: Wait for response and consolidate the response.

Say: Contrary to dedicated Web workers, shared workers use multiple scripts at a time. The example used for a dedicated Web worker can be used as a shared Web worker to display the difference between the two. A shared Web worker is represented by a SharedWorker object.

Do: Show the working of shared Web server using the following code snippet. Write the code on the board.

```
<!DOCTYPE html>
<head>
  <title>Web Workers</title>
  <script type="text/javascript">
    window.onload=function (){
 if (typeof(SharedWorker) === 'undefined')
 {
  alert("Web workers are not supported by this Browser!");
  return;
 }
 else{
```

```
   var sharedworker = new SharedWorker("DeepCalculation.js");
   sharedworker.port.start();
   sharedworker.port.postMessage(2000);
   sharedworker.port.addEventListener("message", function
(evnt){
    alert(evnt.data);
   }, false);
 }
 };
  </script>
</head>
<body>
<h1>Working with Web Workers! It will work by itself, you do not
need to do anything.</h1>
</body>
</html>
```

Do: Explain the code snippet.

Say: In the code snippet, the code section in bold letters focuses on creating an instance of SharedWorker and passes somesharelengthytask.js as an argument to the constructor. The code then combines the message event handler with the port object of the SharedWorker object. The message handler function also performs the same job as explained in the dedicated Web worker example. The code then gives a call to the start() method on the port object. Finally, a call to the postMessage() method is given on the port object, so that the data can be retrieved to be passed to the shared worker.

Do: Write the following code on the board.

```
self.addEventListener("connect", function (evnt) {
  console.log(evnt.data);
  var conn_port = evnt.ports[0];
  console.log(conn_port);
  conn_port.start();
  conn_port.addEventListener("message", function (evnt) {

  var count_prime = 0;

        for (var i = 3; i < evnt.data; i++) {
             isprime = true;
             for (var j = 2; j < i; j++) {
                  if (i % j === 0) {
                       isprime = false;
                       break;
                  }
             }
             if (isprime) {
                  count_prime += 1;
             }
```

```
            }
        conn_port.postMessage(count_prime);

}, false);
}, false);
```

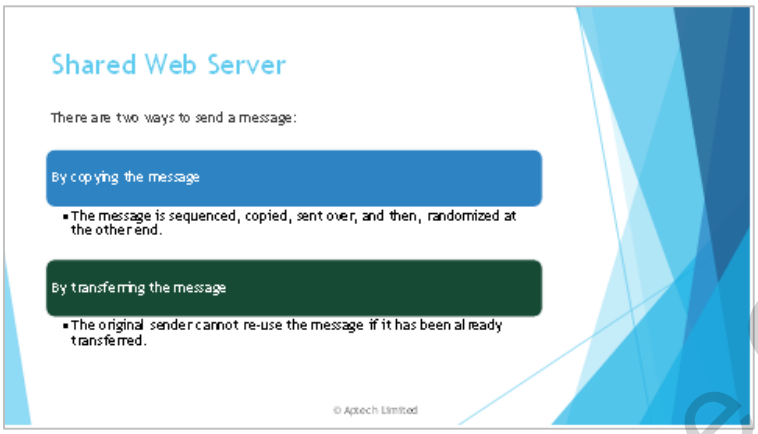
Do: Explain the code snippet.

Say: This code has to be entered in DeepCalculation.js file.
In the code snippet, a variable named conn_port for storing a reference of a port object is declared first. Two events that get handled in this code snippet are: connect and message. The connect event is called when a connection to the shared Web worker is established. The connect event handler takes 'evnt.ports[0]' object and places it in the conn_port variable. It combines the message event handler with the conn_port object (indicated through the statement 'conn_port.addEventListener("message", function (evnt))'.

The start() method of the conn_port object (conn_port.start()) is given a call so that listening to a message on that object can be initiated. The event handler for the message is almost similar to the one written in the code to count prime numbers, except that the event handler here is attached to the conn_port object. Also, postMessage() is called on the conn_port object to send the processed result to the main page.

Slide 7

Understand Shared Web Workers.



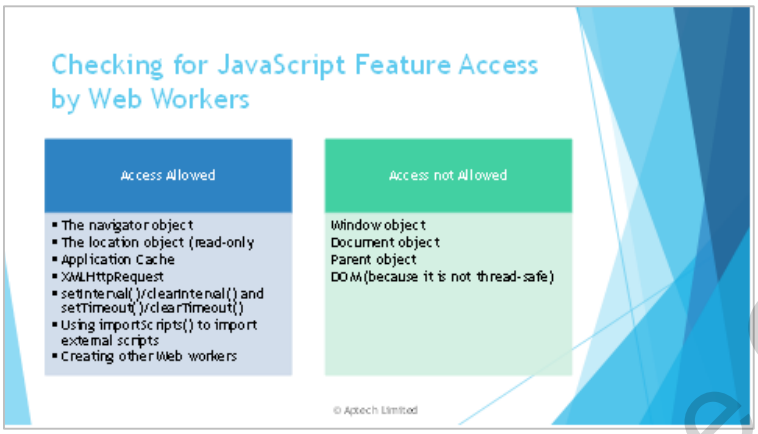
Say: Users can send messages to Web workers in following two ways:

By copying the message: The message is sequenced, copied, sent over, and then, randomized at the other end. The worker and page do not share the same instance, so the result is duplicated at each pass. Most browsers implement this feature by automatically encoding or decoding the value of JSON (format used to send data from a server to a Web page). These operations include overhead to the message transmission significantly.

By transferring the message: The original sender cannot re-use the message if it has been already transferred. Also, data transmission is almost instant. However, the limitation is that only ArrayBuffer is convenient enough to be passed on.

Slide 8

Understand how to check for JavaScript Feature Access.



Say: Only a subset of JavaScript features can access Web workers due to their multi-threaded nature. Features that can be accessed include:

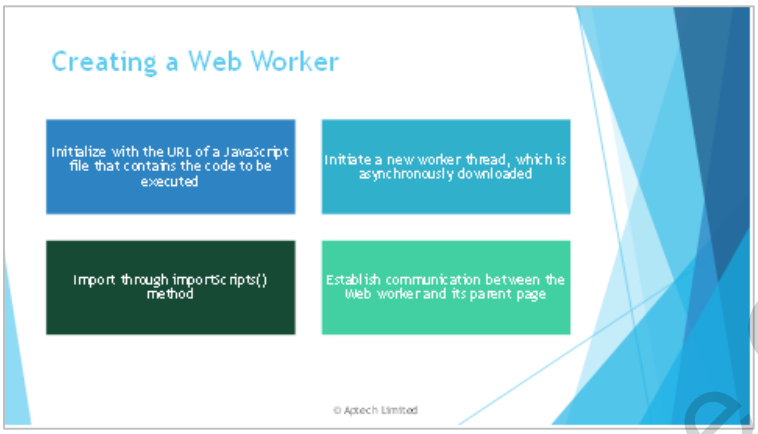
- The navigator object
- The location object (read-only)
- Application Cache
- XMLHttpRequest
- setInterval()/clearInterval() and setTimeout()/clearTimeout()
- Using importScripts() to import external scripts
- Creating other Web workers

Some essential JavaScript features to which Web workers do not have access are:

- window object
- document object
- parent object
- DOM (because it is not thread-safe)

When the browser is rendered into a Web page, a DOM is created. Every element —<p>, , or — is represented in the DOM by a different object. Note that there are global objects, such as window and document, which are not tied to specific elements. This means that a Web worker cannot manage the DOM (and thus the UI). Once developers understand how to use Web workers efficiently, they can use the workers to do heavy processing tasks. In the meantime, the UI related tasks are performed by the code in the page. After Web workers finish the processing task, the outcome can be passed on to the page responsible for rendering the processed results.

Slide 9
Understand how to create a Web Worker.



Say: Let us see how to a create a Web worker.
As a prerequisite, Web workers must be initialized with the URL of a JavaScript file first that contains the code to be executed. This code initializes an event listener and communicates with the script that it has produced from the main page.

Do: Write an example of creating a Web worker on the whiteboard.

```
var objworker = new Worker('largeLoop.js');
```

Say: Point to the example on the board and say this is the way to initialize a new Web worker. If the specified JavaScript file already exists, the browser will initiate a new worker thread, which is asynchronously downloaded. If the path to the worker is incorrect, a 404 error will be produced and the worker fails.

In case an application requiring several JavaScript files, a developer can import them through the importScripts() method. This method takes the file names as arguments separated by commas as follows:

```
importScripts("fileone.js", "filetwo.js", "filethree.js");
```

As soon as the Web worker is initiated, communication between the Web worker and its parent page is established using the postMessage() method. Based on the browser/version, postMessage() takes either a string or JSON object as its argument.

The message sent by Web worker is accessible using the onmessage event in the main page.

Do: Write the following code snippet on the board.

```
<!DOCTYPE HTML>
<html>
  <head>
```

```
   <title>Complex loop with Web worker</title>
    <script>
    function WelcomeUser() {
       var objworker = new Worker("Largeloop.js");
objworker.addEventListener("message", function (evnt) {
       alert("Completed" + evnt.data + "iterations" );
      },false);
alert("Welcome User");
   objworker.postMessage(99999999);
      }
   </script>
  </head>
  <body>
  <input type = "button" id="Welcome_btn" onclick =
"WelcomeUser();" value = "Click This"/>
  </body>
</html>
```

Say: This is an example of code that uses a Web worker.

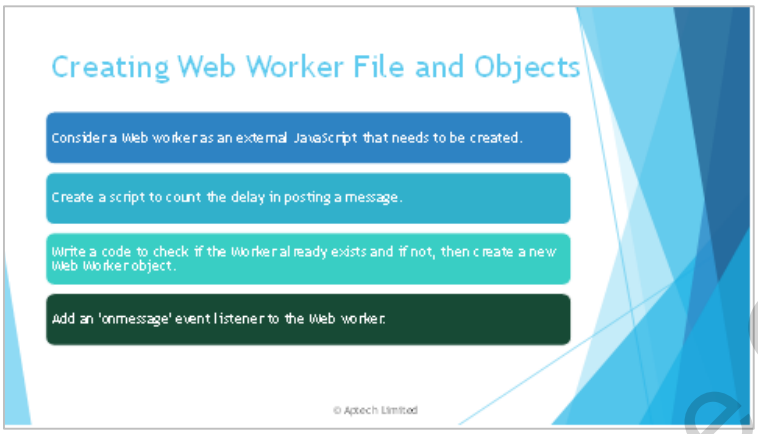
Do: Write this code on the board.

```
 Self.onmessage=function(evnt){
for (var k = 0; k <= evnt.data; k += 1) {
  var s = k;
}
postMessage(s);
};
```

Say: In this code the content of largeLoop.js file makes use of postMessage()API to pass the communication back to the main page. When the code is run, it will display specific alert messages when the button is clicked.

Slide 10

Understand how to create Web Worker file and objects.



Say: Let us see how to create a Web worker file and object. Consider a Web worker as an external JavaScript that needs to be created.

Do: Write the following script on the board.

```
var  k = 0;

function timedCount() {
 k = k + 1;
 postMessage(k);
 setTimeout("timedCount()",500);
}
self.onmessage=timedCount();
```

Say: Explain the code. In the code, a script to count the delay in posting a message is created first. Assume that the script is stored as **new_worker.js**. Here, the postMessage() method posts a message back to the HTML page.

Now, the Web worker file is ready. However, the HTML page may be unaware of its existence.

Do: Write the code on the board:

```
 if (typeof (objnewWorker) == "undefined") {
  objnewWorker = new Worker("new_worker.js");
}
```

Say: Hence, this code snippet first checks if the Worker already exists and if not, then a new Web Worker object is created. Now, one can send and receive messages from the Web worker.

Do: Write the following code on the board:

```
 objnewWorker.onmessage = function(evnt){
 document.getElementById("msg").innerHTML = evnt.data;
};
```

Say: This code adds an 'onmessage' event listener to the Web worker. The data from the Web worker is stored in 'evnt.data' as shown in the code.

Do: Write the following code on the board:

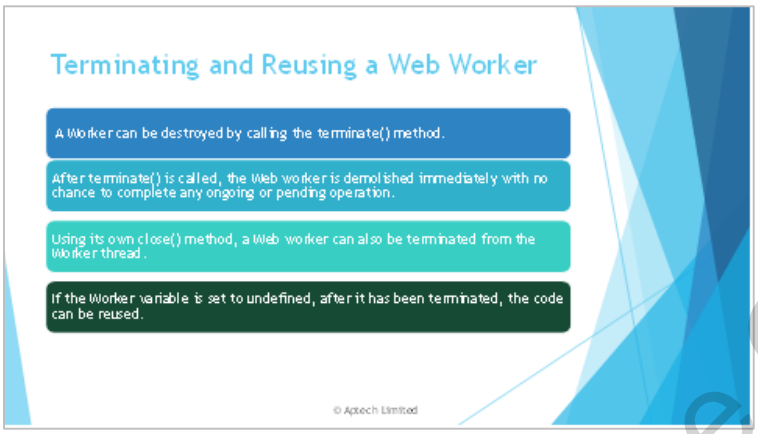
```
<!DOCTYPE HTML>
<html>
  <head>
   <title>Complex loop with Web worker</title>
    <script>
    function Welcomeuser() {
       var objworker = new Worker("Largeloop.js");
objworker.addEventListener("message", function (evnt) {
      },false);
   objworker.postMessage(99999999);
      var objnewWorker;
      if (typeof (objnewWorker) == "undefined") {
 objnewWorker = new Worker("new_worker.js");
}
objnewWorker.onmessage = function(evnt){
 document.getElementById("result").innerHTML = evnt.data;
};
     }
   </script>
  </head>
  <body>
  <input type = "button" id="Welcome_btn" onclick =
"Welcomeuser();" value = "Click This"/>
  <p>Count numbers: <output id="result"></output></p>
  </body>
 </html>
```

Say: The complete HTML markup including the script code will look like as shown. When this code is run the output will be:

Click This
Count numbers 109

Slide 11

Understand how to terminate Web Worker.



Ask: What do you mean by terminating a Web worker?

Do: Wait for response. Consolidate the response.

Say: At times, developers may want to cancel the task being executed in a Worker. To do so, a Worker can be destroyed by calling the terminate() method. Once a Worker is terminated, the developer cannot reuse or restart it. Developers can always create another instance of a worker and use it. Developers can terminate a Web worker by calling the terminate() method of the Web Workers API from the main thread as shown:

```
worker.terminate();
```

After terminate() is called, the Web worker is demolished immediately with no chance to complete any ongoing or pending operation. The Web worker gets no time to clean up. Thus, demolishing a Web worker abruptly may lead to several memory leaks (pool of underutilized free space in memory).

Using its own close() method, a Web worker can also be terminated from the Worker thread.

Once the close() method is called, any queued tasks present in the event loop is called off and the Web worker scope is closed forever.

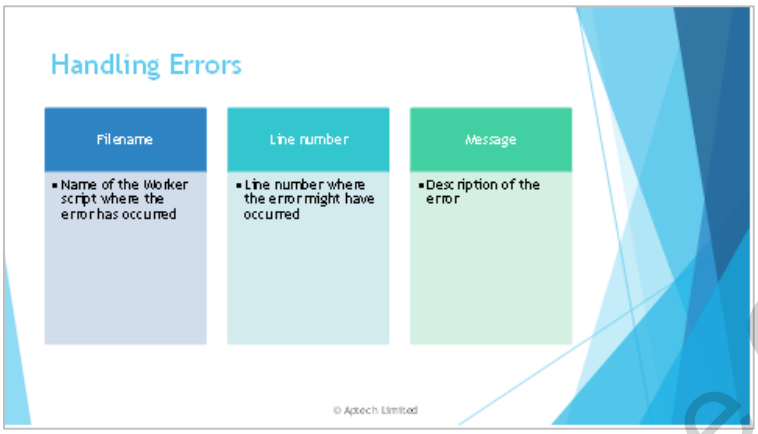
Reusing the Web Worker: If the Worker variable is set to undefined, after it has been terminated, the code can be reused. A Worker variable can be set to undefined as shown:

```
objnewWorker = undefined;
```

However, a Web worker cannot access DOM in JavaScript, although there are alternative ways such as using weakref.

Slide 12

Understand how to handle errors.



Say: As with any JavaScript logic, if an error occurs while executing a Worker, an ErrorEvent is fired. Three useful properties to figure out what went wrong in the interface are:

1. Filename: Name of the Worker script the error has occurred
2. Line number: Line number where the error might have occurred
3. Message: Description of the error

Do: Write the code snippet as example of setting up an error event handler to print the properties of the error.

```
 window.onload=function () {
 var webworker = new Worker("scripts/DeepCalculation.js");
 webworker.onmessage=function (evnt) {
     alert("some output "+evnt.data);
 }
 webworker.addEventListener("error", function (evnt) {
  alert("Line #" + evnt.lineno + " - " + evnt.message + " in " +
evnt.filename);
 }, false);
 webworker.postMessage(10000);
};
```

Say: Point to the examples and say this is an example of setting up an error event handler to print the properties of the error.

If the Web worker is performing certain complex operations, developers might need to add error handling to the main Web page code such that if any unhandled error occurs inside the Worker, a suitable action for the same can be taken. By handling the error event of the Worker object, developers can detect the error in the Worker thread. This code demonstrates this feature.

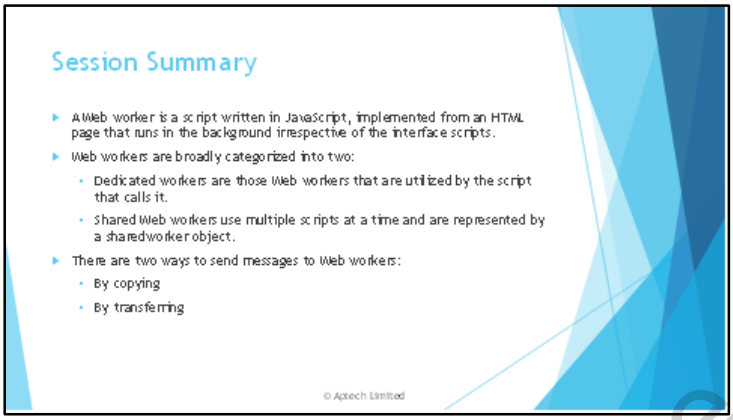
Do: Write the code on the board.

```
window.onload=function () {
 var webworker = new Worker("scripts/DeepCalculation.js");
 webworker.onmessage=function (evnt) {
     alert("specific output "+evnt.data);
 }
 webworker.addEventListener("error", function (evnt) {
  alert("Line #" + evnt.lineno + " - " + evnt.message + " in " +
evnt.filename);
 }, false);
 webworker.postMessage(10000);
});
```

Do: Discuss the code as follows.

Say: As observed from the code, an error handler has been reinforced to the error event of the Worker object. The error handler function receives an event object as an argument that provides error information such as the line number at which the error has occurred (evnt.lineno), error message (evnt.message), and file in which the error has occurred (evnt.filename).

Slide 13
Summarize the session.



Do: Conduct a small activity to engage the students in recalling the key learning points of the session. Divide the class into two or three groups depending on the class size. Ask each group to pose a concept-related question based on what they have learned in the session to the other teams. Allow the teams to pose questions to each other in turns. Allow not more than 5 minutes for this activity. At the end, thank and congratulate the students for a great recap. Then, debrief by presenting the summary points.

Say: Let us summarize the session as follows:

- A Web worker is a script written in JavaScript, implemented from an HTML page that runs in the background irrespective of the interface scripts.
- Web workers are broadly categorized into two: Dedicated Web workers and Shared Web workers.
  Dedicated workers are those Web workers that are utilized by the script that calls it.
  Shared Web workers use multiple scripts at a time and are represented by a sharedworker object.
- There are two ways to send messages to Web workers:
  By copying
  By transferring

Say: Thank you for your participation in this session. The next session will be about JavaScript Developer Tools.

7.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand different types of JavaScript Developer Tools that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 8 – JavaScript Developer Tools

8.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

8.1.1 Objectives

By the end of this session, the learners will be able to:

- Describe what are developer tools
- Explain the use of various browser developer tools with JavaScript

8.1.2 Teaching Skills

To teach this session successfully, you must know about the different types of JavaScript Developer Tools. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

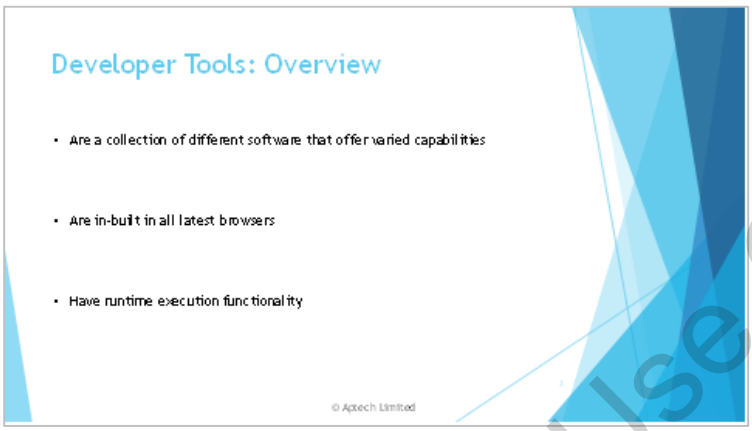
Follow the order given here during In-Class activities.

8.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about various types of developer tools such as Chrome developer tool, React developer tool for Chrome, Firefox developer tool, and Internet Explorer developer tool. This session also explains how to debug JavaScript code using these developer tools.

8.2 In-Class Explanations

Slide 3
Understand the concept of the Developer Tools.



Say: Let us get started with the first topic within this session - Overview of Developer Tools.

Ask: What do you understand by developer tools? Has anyone used any developer tool before?

Do: Give the students some time to answer. Consolidate the responses.

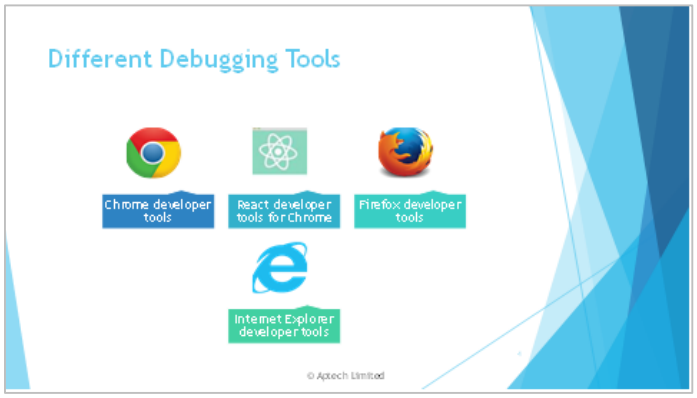
Say: The development tools or developer tools are a collection of different software that offer capabilities including event-handling, code reduction, data-rendering, and so on.
All mainstream browsers support JavaScript, which is proficient for Web application development based on browsers. Today's browsers are equipped with built-in development tools that are compatible with JavaScript and other Web technologies.

As JavaScript is a dynamic language, it is required by developer tools to have runtime execution functionality. This makes development with JavaScript quick and efficient, as less time is required for debugging and developers can immediately view results of changes they have made.

Refer to the following link to know more:
https://medium.com/javascript-scene/must-see-javascript-dev-tools-that-put-other-dev-tools-to-shame-aca6d3e3d925

Slide 4

Introduce different debugging tools.



Say: Let us move to the next topic within this session - Different Debugging Tools.

Say: Some of the browser developer tools that are used with JavaScript are as follows:

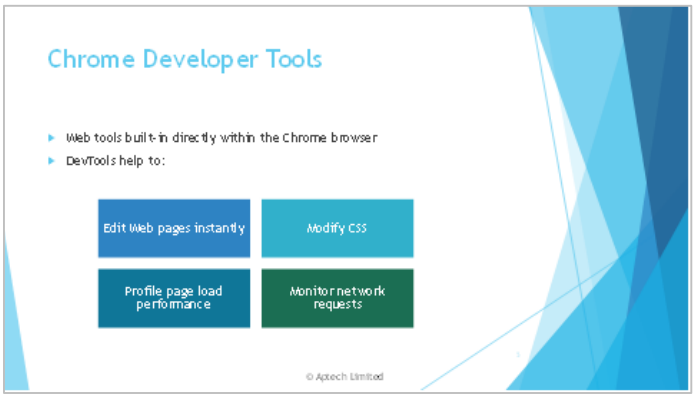
- Chrome developer tools
- React developer tools for Chrome
- Firefox developer tools
- Internet Explorer or IE developer tools

Refer to the following link to know more:
https://raygun.com/javascript-debugging-tools

Let us begin with the Chrome developer tool.

Slide 5

Introduce the Chrome Developer Tools.



Do: Talk about the Chrome Developer Tools.

Say: The Chrome developer tools, also called DevTools, are Web tools that are built-in directly within Chrome browser.

These tools enable developers to detect issues quickly. They also help developers in editing Web pages instantly, modifying CSS, profiling page load performance, and monitoring network requests.

Demonstrate: How to access the Chrome DevTools.

Say: Developers can access these tools using following ways based on their requirements. If developers need to work with DOM or CSS then you do the following steps.

Demonstrate: Step 1 - Right-click any element on the Web page and select **Inspect**.
Step 2 - Press **Ctrl+Shift+C** when using Windows or Linux.

Say: These methods will open the **Elements** panel.

Do: Show them the Elements panel.

Say: If developers need to run JavaScript or view logged messages then follow the step.

Demonstrate: Press **Ctrl+Shift+J** when using Windows or Linux.

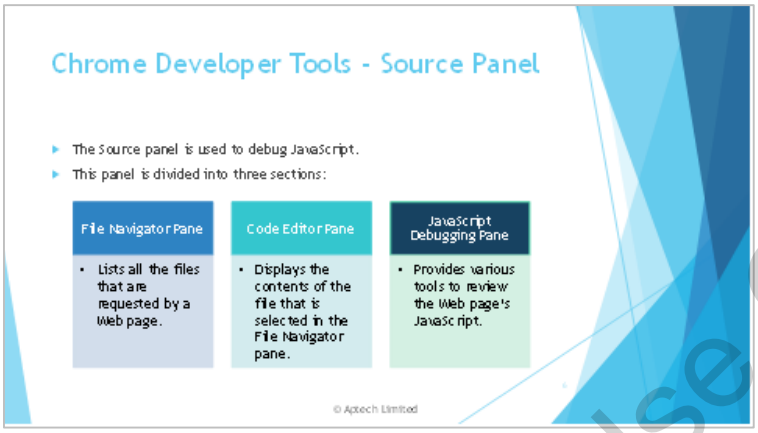
Say: This method will open the **Console** panel.

Do: Show the Console panel.

Refer to the following link to know more:
https://developers.google.com/web/tools/chrome-devtools/

Slide 6

Understand the Source Panel.

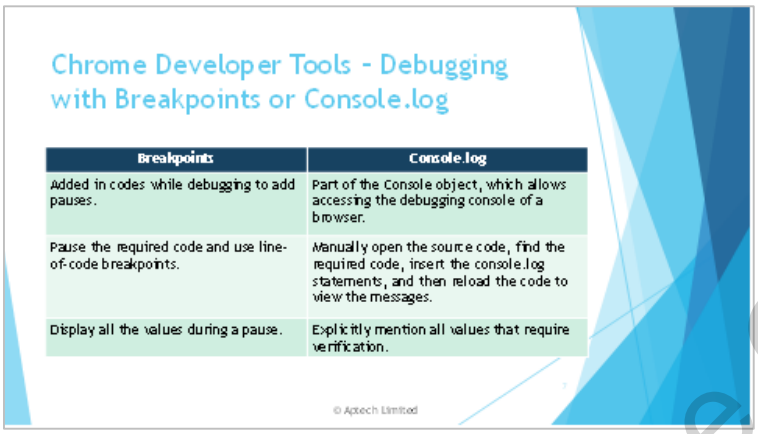


Say: Developers can use the **Source** panel to debug JavaScript. This panel is divided into three sections:

- File Navigator Pane - This pane lists all the files that are requested by a Web page.
- Code Editor Pane - This pane displays the contents of the file that is selected in the **File Navigator** pane.
- JavaScript Debugging Pane - This pane provides various tools to review the Web page's JavaScript.

Slide 7

Understand debugging and breakpoints.



Say: Developers can debug code using console.log method or breakpoints.

Breakpoints are a feature that help make debugging easier. They are added in codes while debugging to add pauses. The browser looks for these breakpoints to identify when to pause the code execution to debug it.

The Console.log method is part of the Console object, which allows accessing the debugging console of a browser. The log method with the console object is used to display messages in the debugging console. Developers have to insert many console.log statements in the code to verify values while the script executes.

Breakpoints are quicker to execute. This is because they allow pausing the code in the middle of execution and verifying all the values at that time. With breakpoints, developers can pause the required code without knowing how the source code is structured. Developers can also use line-of-code breakpoints, if they want to pause the code at a specific code line.

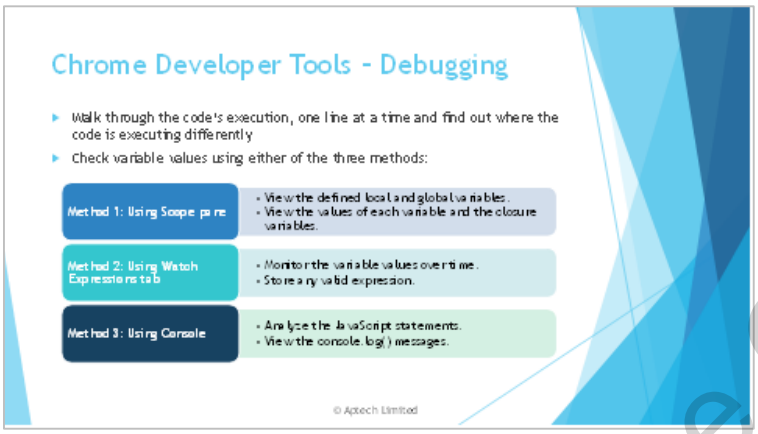
Developers have to manually open the source code, find the required code, insert the console.log statements, and then reload the code to view the messages.

With breakpoints, DevTools displays all the values during a pause.

Developers have to explicitly mention all values that require verification using console.log method.

Slide 8

Understand how to Debug scripts.



Say: A common cause for bugs is when the script executes in an incorrect manner. Developers can walk through their code's execution, one line at a time and find out where the code is executing differently.

Demonstrate: Developers can step through the code by performing the following steps in the Source panel:

Step 1 - Click **Step into next function call** to step through the execution of the ***onClick() function***, one line at a time. DevTools will highlight following line of code:
if (inputsAreEmpty()) {

Step 2 - Click **Step over next function call**. DevTools will execute inputsAreEmpty() without stepping into it.

Say: Developers can also check variable values using the DevTools. There are three main ways to do this:

Method 1: Using Scope pane - The **Scope** pane displays which local and global variables are defined when a code is paused. It also shows the values of each variable and the closure variables, when applicable. The **Scope** pane is empty when a code is not paused.

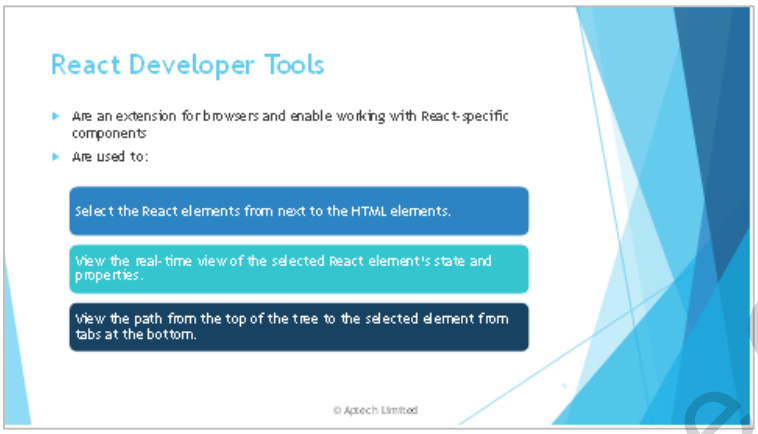
Method 2: Using Watch Expressions tab - The **Watch Expressions** tab enables the developers to monitor the variable values over time. The Watch Expressions store any valid expression.

Method 3: Using Console - Developers can use the **Console** to analyze the JavaScript statements, in addition to viewing the ***console.log()*** messages.

Once developers find the bug, they can edit the code directly from the DevTools UI.

Slide 9

Understand React developer tools.



Say: React developer tools or DevTools are an extension for browsers such as Chrome and Firefox. They enable developers to work with React-specific components. As an example, consider React DevTools for Firefox.

Do: Talk about how to install React DevTools for Firefox.

Say: Developers can install React DevTools from Firefox extension pages and these DevTools do not require any setup after installation.

Demonstrate: Developers should reopen the browser, if they are unable to access the toolkit after installing React DevTools. The next step would be to check whether DevTools could access the React instance on the Web pages. Clicking **React** logo that appears next to the Address bar on the browser can do this.

Do: Show how to access the React DevTools.

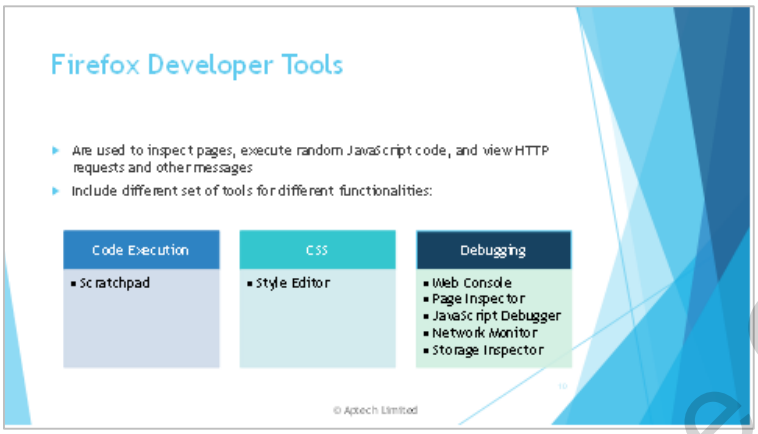
Demonstrate: Developers can access React DevTools by right-clicking the Web pages and selecting **Inspect**. Once the **Inspection** window opens, developers can navigate to the **React** tab and view the component hierarchy.

Say: These tools are similar to Chrome DevTools. However, there are some differences.

- Developers can select the React elements from next to the HTML elements.
- Developers can view the real-time view of the selected React element’s state and properties.
- Developers can view the path from the top of the tree to the selected element from tabs at the bottom.

Slide 10

Understand Firefox Developer Tools.



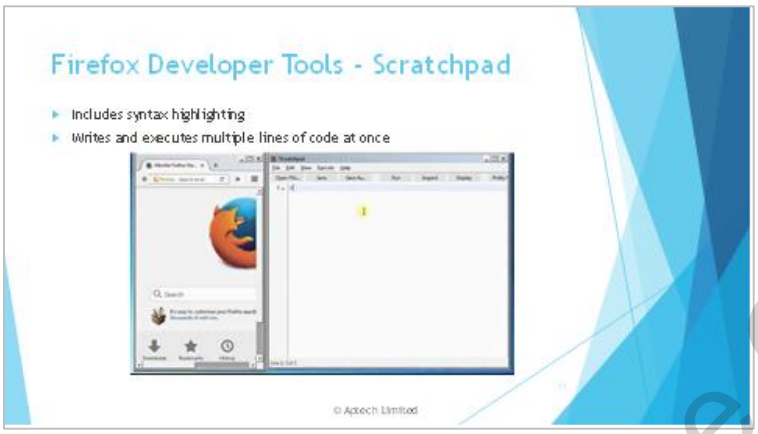
Say: Firefox's Web Developer menu contains tools that developers can use to inspect pages, execute random JavaScript code, and view HTTP requests and other messages.
Firefox Developer tools provide different set of tools for creating and debugging Websites and pages.

Developers can use Scratchpad and Style Editor to execute code and CSS style sheets, respectively. For debugging, developers can use tools such as Web Console, Page Inspector, JavaScript Debugger, Network Monitor, Storage Inspector, and so on. Developers can access all the tools from the Web Developer in the Firefox menu.

Let us cover some of the tools. Let us begin with Scratchpad.

Refer to the following link to know more:
https://developer.mozilla.org/son/docs/Tools

Slide 11
Understand Scratchpad.



Say: The updated Scratchpad includes syntax highlighting and developers can write and execute multiple lines of code at once.

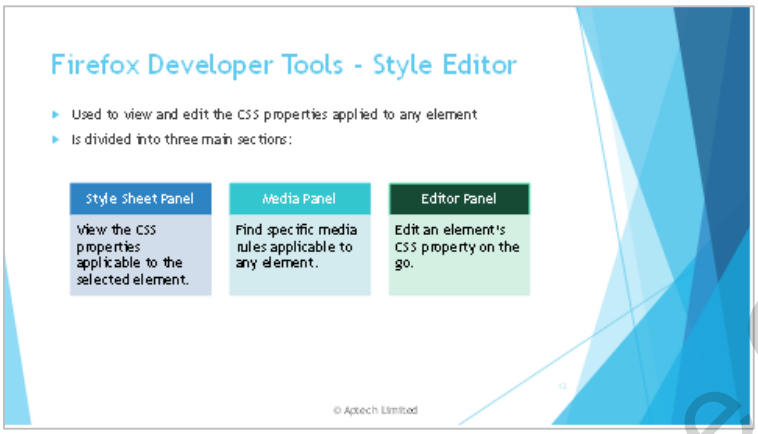
Demonstrate: Developers can access Scratchpad using the following ways based on their requirements:

- Click **Web Developer** menu and select **Scratchpad**.
- Click the **Wrench** icon and select **Scratchpad**.
- Press **Shift+F4** when using Windows.

Say: Once the code is written, developers can use the **Execute** menu and click **Run**. This will run the code in the current tab.

Refer to the following link to know more:
https://developer.mozilla.org/en-US/docs/Tools/Scratchpad

Slide 12
Understand Style Editor.



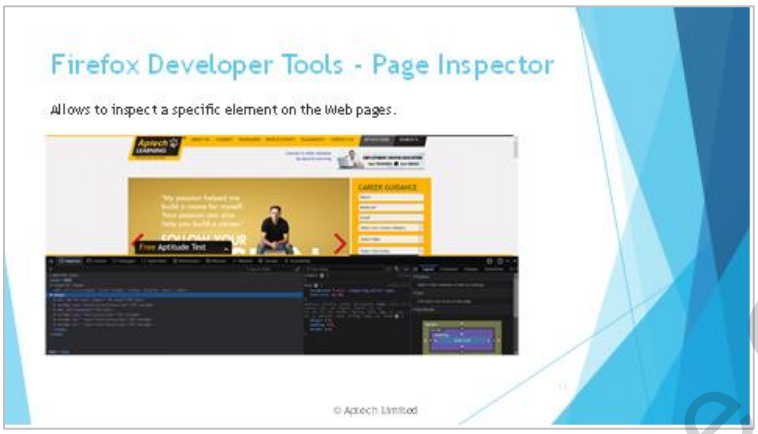
Say: Style Editor is used to view and edit CSS properties applied to any element.

Demonstrate: Developers can access the Style Editor from the Web Development menu.

Say: The Style Editor is divided into three main sections:
- Style Sheet panel – Enables to view all the CSS properties applicable to the selected element.
- Media panel – Enables to find specific media rules applicable to any element.
- Editor panel – Enables to edit an element's CSS property on the go.

Refer to the following link to know more:
https://developer.mozilla.org/en-US/docs/Tools/Style_Editor

Slide 13
Understand how to use Page Inspector.



Say: Developers can inspect a specific element on the Web pages using the Page Inspector in the following ways:

Demonstrate:
Right-click an element and select **Inspect** or press **Q**.
Launch the Inspector from the Web Developer menu.

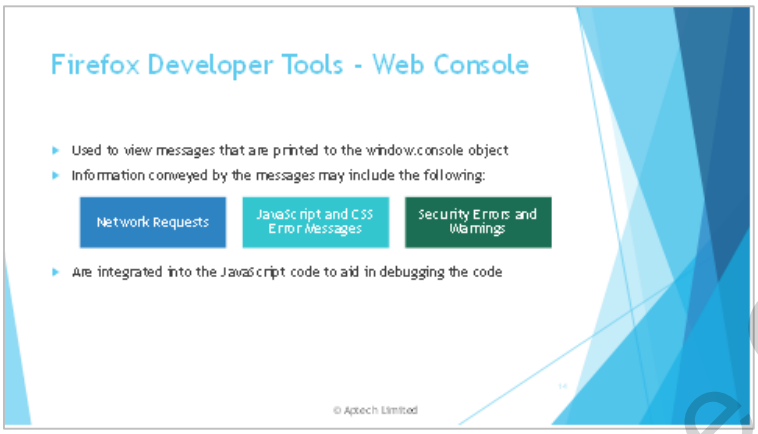
Say: A toolbar opens at the bottom of the screen, which the developers can use to control the inspector. The selected element gets highlighted.

Demonstrate: The steps to select a new element are:
- Click **Inspect** on the toolbar.
- Hover over the Web pages.
- Select element.

Say: Once these steps are performed, the element under the cursor gets highlighted. Developers can use the breadcrumbs on the toolbar to navigate between the child and the parent elements.

Slide 14

Understand how to use Web Console.



Say: Developers can use the Web Console to view messages that are printed to the ***window.console object***.

Demonstrate: Show a Web Console.

Say: Different types of messages displayed in the Web Console are related to following information:

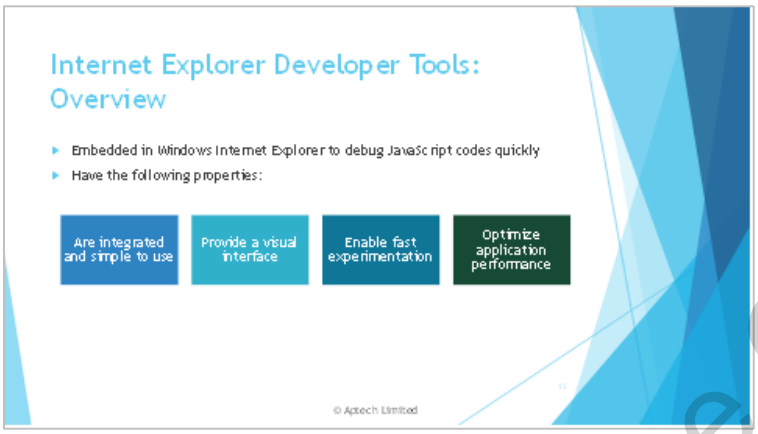
- Network Requests
- JavaScript and CSS Error Messages
- Security Errors and Warnings

These messages can be integrated into the JavaScript code to aid in debugging the code.

Demonstrate: Developers can access the Web Console in the following ways:

- Select **Web Console** from the **Web Developer** menu.
- Press **Ctrl + Shift + K**.

Slide 15
Understand Internet Explorer Developer Tools.



Say: The developer tools are embedded in Windows Internet Explorer or IE and enable the developer to debug JavaScript codes quickly.

The IE developer tools have the following properties:
- Are integrated and simple to use
- Provide a visual interface to the platform
- Enable fast experimentation
- Optimize application performance

Demonstrate: Developers can access the IE developer tools in the following ways:
Press **F12**.
Or Select **Developer Tools** from the **Tools** menu

The Developer Tool opens in a new window for each opened IE tab.

Developers can pin the tools to a tab. To do so:
Click **Pin** button or press **Ctrl + P**.

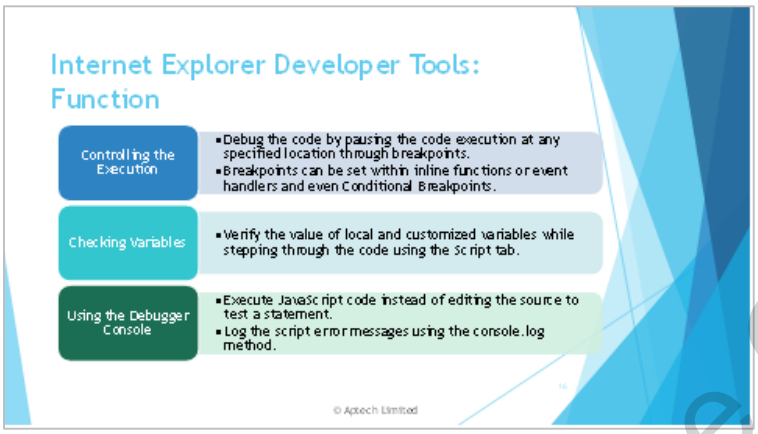
Demonstrate: Show how to start and stop the debugger.

Once developers have opened the developer tools, they can initiate debugging by navigating to the **Switch** tab and click **Start Debugging**.

Say: The developer tools refresh the Web pages and unpin the tools, if it is pinned, when debugging starts.

Slide 16

Understand Internet Explorer Developer Tools.



Say: Let us talk about what we can do using IE Developer Tools.

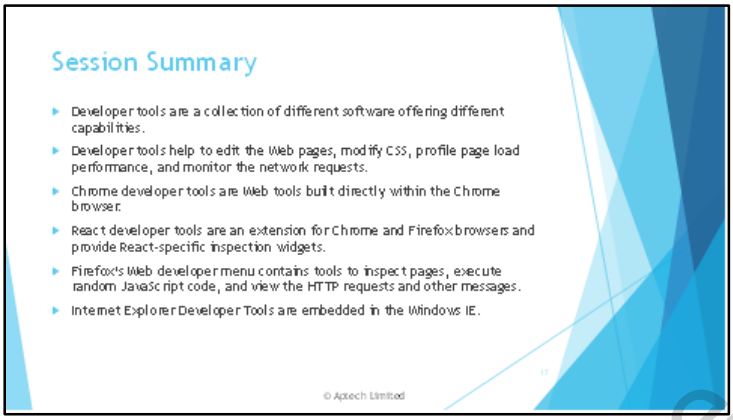
Controlling the Execution - With the IE developer tools, developers can debug the code by pausing the code execution at any specified location through breakpoints. Breakpoints can be inserted by clicking a line number or right-clicking the source and selecting Insert Breakpoint. Breakpoints can be set within inline functions or event handlers. In addition, the developers can also set Conditional Breakpoints.

Checking Variables - Developers can verify the value of variables while stepping through the code using the **Script** tab, which contains the **Locals** pane and the **Watch** pane. The **Locals** pane enables developers to view local variables. The **Watch** pane enables developers to monitor a customized list of variables. Variables can be added to this list. To add the variables, right-click the source and click **Add Watch**. Alternatively, you could type the variable name in the **Watch** pane.

Using the Debugger Console - Developers can execute JavaScript code in the Console pane, instead of editing the source to test a statement. Developers can also use Multi Line Mode to execute multiple lines at once. The code is executed immediately and the developers can test the outcome of adding that code at the same location as the current breakpoint. Console can also be used even when debugging is not required. Console also logs script error messages using the console.log method.

Refer to the following link to know more:
https://eclipsesource.com/blogs/2013/11/25/a-look-at-the-internet-explorer-11-developer-tools/

Slide 17
Summarize the session.



Do: Ask volunteers to recap what they have learned in the session. To encourage all students to participate, ask students randomly to share one or two points about a specific developer tool. Have the interaction not longer than five minutes. At the end, thank the students and summarize the key learning points.

Say:
- Developer tools are a collection of different software that offer different capabilities.
- Developer tools help in editing the Web pages, modifying CSS, profiling page load performance, and monitoring the network requests.
- Chrome developer tools are Web tools that are built directly within the Chrome browser and help detect issues quickly.
- React developer tools are an extension for Chrome and Firefox browsers. They provide with React-specific inspection widgets that aid in development.
- Firefox's Web developer menu contains tools that can be used to inspect pages, execute random JavaScript code, and view the HTTP requests and other messages.
- Developer tools embedded in the Windows Internet Explorer or IE enable the developer to debug the JavaScript codes quickly.

Say: Thank you for your participation in this session. The next session will be about using responsive APIs in JavaScript.

8.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should also explore and understand using responsive APIs in JavaScript that are offered in the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 9 - Using Responsive APIs in JavaScript

9.1 Pre-Class Activities

Before you commence the session, you should do a quick recap of the previous session with the class.

9.1.1 Objectives

By the end of this session, the learners will be able to:

- Define Responsive API
- Describe how an API works
- Explain the uses of APIs in JavaScript
- Describe how to use responsive APIs

9.1.2 Teaching Skills

To teach this session successfully, you must know about the responsive APIs JavaScript. You should teach the concepts in the theory class using slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities:

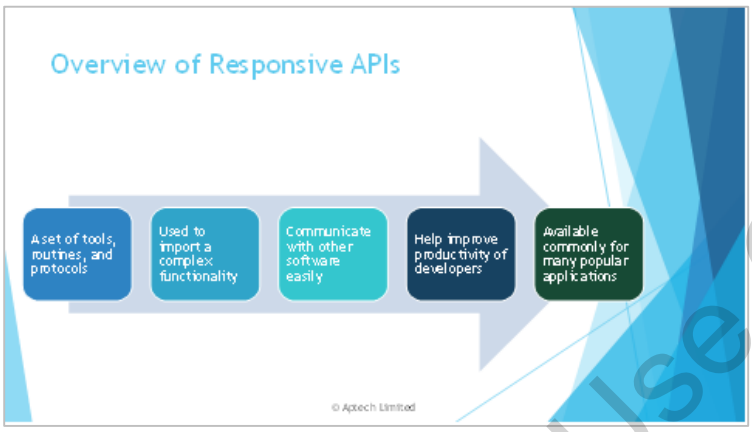
Follow the order given here during In-Class activities.

9.1.3 Overview of the Session:

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation. Tell them that they will learn about the APIs - how they work, what can be done with the help of APIs in JavaScript, and how to use responsive APIs.

9.2 In-Class Explanations

Slide 3
Understand the Responsive APIs.



Ask the students, what is an API? Give some time for them to think and answer.

Do: Consolidate the responses.

Say: Tell the students that in a real-world analogy of abstraction can be as follows:

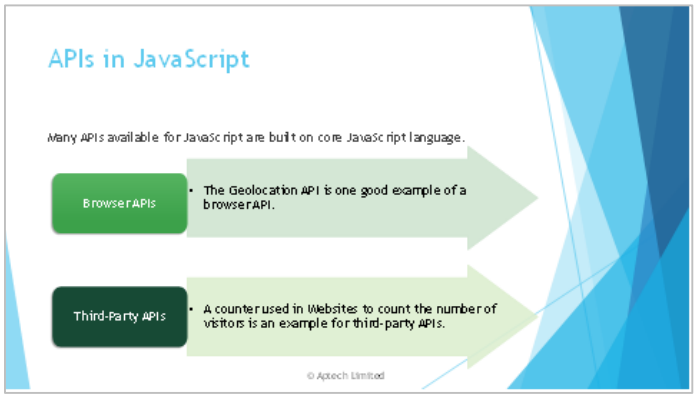
Getting water through a tap, wherein how the water flows from the water tank through the pipe is not known to the person drinking the water.

Application Programming Interface (API) is defined as a set of tools, routines, and protocols. One of its purposes is to help to import a complex functionality into a code. APIs abstract complex functionalities and allow them to be used in code using an easy syntax similar to our example given earlier.

In simple words, APIs allow developers to communicate with other software easily. They help improve productivity of developers. Today, APIs are commonly available for many popular applications.

Slide 4

Introduce the two types of APIs.



Ask: What do you know about APIs in JavaScript?

Do: Wait for responses. Consolidate the responses. Explain the two types of APIs.

Say: There are many APIs available for JavaScript that are built on the core JavaScript language. These APIs give programmers an option to use functionalities with client-side JavaScript code. Browser APIs and third-party APIs are the two categories of APIs available for JavaScript.

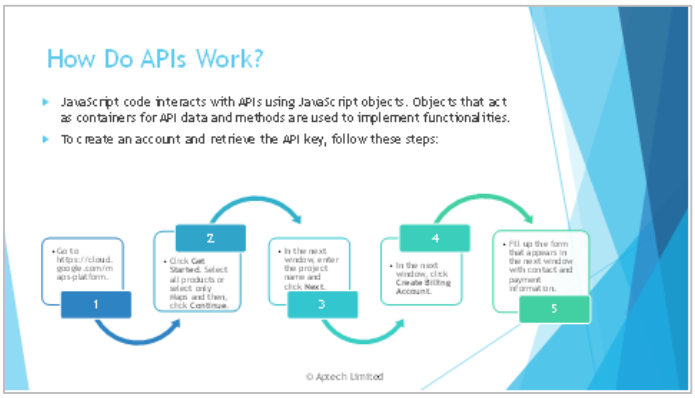
Browser API – Uses built-in functionalities of the Web browser that uses the built-in data of the computer. The Geolocation API is one good example of a browser API. It provides an easy JavaScript function to retrieve location data through the browser. This is done by interacting with the GPS hardware using a low-level code such as C or C++ in the background. The background process is abstracted here.

Third-Party APIs - Are available on the Internet and not built into the browser. A counter used in Websites to count the number of visitors is an example for third-party APIs. The code provided by the service provider needs to be embedded in the Web page.

Reference: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

Slide 5

Explain how APIs work.



Say: JavaScript code interacts with APIs using JavaScript objects. Objects act as containers for API data and methods are used to implement functionalities.

Demonstrate:

Following code is used to embed Google Maps API provided by Google within a script:

```
<script
src="https://maps.googleapis.com/maps/api/js?key=MY_KEY&callback
=myLoc">
```

Explain:

In this code, MY_KEY will be replaced with the key that the developer retrieves from the Google Maps service. This service is a paid one though a trial version is available for a limited period upon providing credit card details.

Do: Ask the students to form groups of five. Ask the members within each group to identify themselves with one of the five steps listed on the slide. Member 1 of each team needs to state the step and the others will do the same for the rest of the steps in sequence. Each team will repeat this cycle of stating the steps, but this time the member with step 2 will start with step 1 and each member will state the next step. The steps will be stated thus, for four more times. Like this, each team member will have stated the steps completely one time. The activity should not take more than five minutes. The idea of the activity is to bring some learner interaction and a fun way to introduce the steps and to make the students remember the steps.

Say: That was wonderful! Let me summarize. To create an account and retrieve the API key, follow these steps:

Go to https://cloud.google.com/maps-platform.

Click **Get Started**. A popup window is displayed with three types of products (Maps, Routes, and Places). Select all or select only Maps and then, click **Continue**.

In the next window, enter the project name and click **Next**.

In the next window, click **Create Billing Account**.

Fill up the form that appears in the next window with contact and payment information. This would require credit card details to be provided. The API key will appear in the next window. This API can then be used to find the location of a device.

Demonstrate: Write the following code on the whiteboard:

```
<!DOCTYPE html>
<html>
<head>
<title>Google Maps API</title>
</head>
<body>
<div id="MapDiv" style="width:100%;height:500px;"></div>
<script>
function myLoc() {
var mapProperties= {
  center:new google.maps.LatLng(28.4595,77.0266),
  zoom:10,
};
var googleMap = new
google.maps.Map(document.getElementById("MapDiv"),mapProperties)
;
}
</script>
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callba
ck=myLoc"></script>
<!-- Replace "YOUR_KEY" with API key, you will get after
registering your app -->
</body>
</html>
```

Explain: The tag is used to display the Google Map on the Web page. The **googleMap** variable is used to instantiate the Map object. The output is placed inside the tag using document.getElementById().

The google.maps.LatLng object is used to import the map of the specified location as provided in the parameters such as Latitude and Longitude. The parameters are the GPS coordinates of that location. It helps to load map of particular location on the Web page.

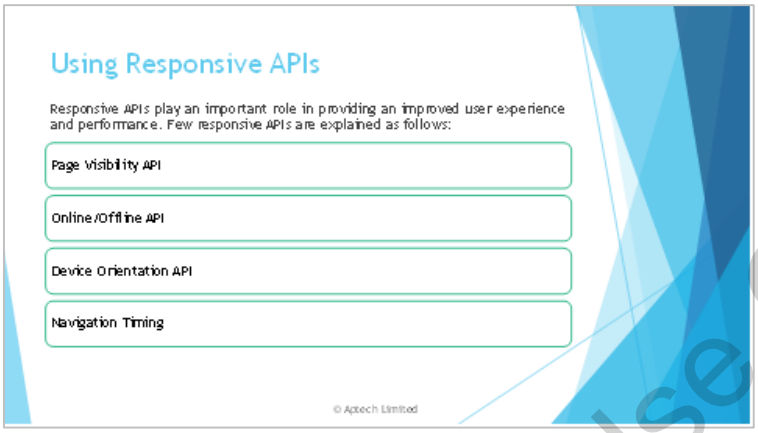
Demonstrate: Write the following code on the whiteboard:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
  <title>Ghibli API Demo</title>
</head>
<body>
  <div id="root"></div>
  <script>
const app = document.getElementById('root');
const container = document.createElement('div');
app.appendChild(container);
var request = new XMLHttpRequest();
request.open('GET', 'https://ghibliapi.herokuapp.com/people',
true);
request.onload = function () {
  var data = JSON.parse(this.response);
  if (request.status >= 200 && request.status < 400) {
    data.forEach(people => {
      const p = document.createElement('p');
      p.textContent = people.name;
      container.appendChild(p);
    });
  } else {
   alert('Sorry, this is not working.');
  }
}
request.send();
</script>
<div id="root">
  <div class="container">
  </div>
</div>
</body>
</html>
```

Slide 6

Explain how to use the Responsive APIs.



Ask: What is page visibility API?

Do: Give some time for them to think.

Say: Page Visibility API helps to listen to events and determine whether the Web page is being viewed by the user or the browser is being minimized. Behavior of the Web page can be changed based on this data.

For example, one can pause the background video or stop image carousel from moving on to the next slide in a slideshow if user is not viewing the page.

Demonstrate: Write the following code on the whiteboard as an example and explain it.

```
//for simplicity reason, just checking for webkitHidden vendor
// prefix
var hiddenSite, visibileSite;
if (typeof(document.hidden) !== "undefined") {
  hiddenSite = "hidden";
  visibleSite= "visibilitychange";
} else if (typeof(document.webkitHidden)!== "undefined") {
  hiddenSite = "webkitHidden";
  visibleSite = "webkitvisibilitychange";
}

if (typeof(document[hidden]) === "undefined") {
  console.log("Browser does not support the Page Visibility
API.");
    }
   else {
  // Handle page visibility change
```

```
  document.addEventListener(visibilityChange, function(){} ,
false);
     }
```

Ask: What is online/offline API?

Do: Give some time for them to think.

Say: The online/offline API helps to find whether the Internet is ON or OFF and save the work that the user has been doing instead of showing an alert message of connection being lost.

Demonstrate: Write the following code on the whiteboard as an example and explain it. Following is an example to run and verify the events:

In JavaScript:

```
window.addEventListener('load', function() {
  var rank = document.getElementById("rank");
  var logbook = document.getElementById("logbook");
  function updateStatus(event) {
    var con = navigator.onLine ? "online" : "offline";
    status.Name = condition;
    status.innerHTML = con.toUpperCase();
    logbook.insertAdjacentHTML("beforeend", "Event: " +
event.type + "; Status: " + condition);
  }
  window.addEventListener('online',  updateStatus);
  window.addEventListener('offline', updateStatus);
     });
```

In CSS:

```
#status {
  position: fixed;
  width: 100%;
  font: BoldItalics 1em Callibri;
  color: #EEF;
  padding: 0.5em;
}
#logbook{
  padding: 2.5em 0.5em 0.5em;
  font: 1em Courier new;
}
.online {
  background: Red;
}
.offline {
  background: Blue;
```

```
}

In HTML:
<div id="status"></div>
<div id="logbook"></div>
<p>Click here! This is the Sample!</p>
```

Ask: The learners if they have heard about Device Orientation API.

Do: Give some time to think.

Say: The device orientation API helps to retrieve orientation of the mobile device of the user by accessing the gyroscope and the compass data of the device. The mobile device can use this information to rotate the display and present a wide-screen view of the content.

Demonstrate: Write the following code on the whiteboard as an example and explain it. Following example shows how device orientation event works:

```
<!doctype html><html>
<head>   <div id="orientation-result"></div>
 <script>      var output =
document.getElementById("orientation-result");
function handleOrientation(event)
{      var output = document.getElementById("orientation-
result");
 var x = event.beta;  // In degree in the range [-180,180]
var y = event.gamma; // In degree in the range [-90,90]
var z = event.alpha;      var s = "";
  if((x==0) && (y==0) && (z==0))
{      s+='Device is stationary';      }
if(z>0){      s+='Device is lifted\n';      }
 if(x>0){  s+="Device's bottom panel is at lower height than top
panel\n";      }
 if(x<0){s+="Device's top panel is at lower height than Bottom
panel\n";      }
if(y>0){      s+="Device's left panel is at lower height than
right panel\n";      }
 if(x<0){      s+="Device's right panel is at lower height than
left panel\n";      }
output.innerHTML=s;}
window.addEventListener('deviceorientation',
handleOrientation);
 </script>   </head>   <body>   </body></html>
```

Ask: The learners if they have heard about navigation timing.

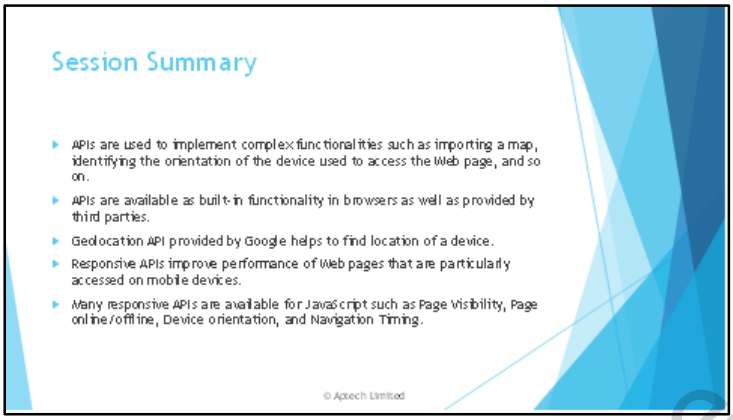
Do: Give some time to think.

Say: The navigation timing API helps to analyze performance of a Web page such as information on the unloading time of the prior page accessed and the total time required for loading the current page.

Demonstrate: Write the following code on the whiteboard as an example and explain it. Following code is used for measuring the loading time of the page with the help of Navigation Timing API:

```
<!doctype html>
<html>
   <head>
   <div id="time-result"></div>
   <div id="time-result1"></div>
   <div id="time-result2"></div>
   <div id="time-result3"></div>
      <script>
      response=document.getElementById('time-result');
      response1=document.getElementById('time-result1');
      response2=document.getElementById('time-result2');
      response3=document.getElementById('time-result3');
      var timevar =window.performance.timing;
      window.addEventListener('load', function() {
            var current_time = new Date().getTime();
            response.innerHTML='Perceived loading time (in
milliseconds): ' + (current_time - timevar.navigationStart);
});
      response1.innerHTML='Loading time:
'+(timevar.loadEventEnd-timevar.navigationStart);
      response2.innerHTML='DOM load time: '+(timevar.domComplete
- timevar.domLoading);
      response3.innerHTML='request response time:
'+(timevar.responseEnd - timevar.requestStart);
      </script>
   </head>
   <body>
   </body>
</html>
```

Slide 7
Summarize the session.



Ask: The learners to write down the key points learned in the session.

Do: Wait for the learners to write.

Do: Show the slide to the students and revise the points covered. Then, list the below points one by one.

- APIs are used to implement complex functionalities such as importing a map, identifying the orientation of the device used to access the Web page, and so on.
- APIs are available as built-in functionality in browsers as well as provided by third parties.
- Geolocation API provided by Google helps to find location of a device.
- Responsive APIs improve performance of Web pages that are particularly accessed on mobile devices.
- Many responsive APIs are available for JavaScript such as Page Visibility, Page online/offline, Device orientation, and Navigation Timing.

Say: Thank you for your participation in this session.

9.3 Post Class Activities for Faculty

As this is the last session, you can recap the entire course and concepts taught in various sessions of the course.

Tips:

You can also check the Articles/Blogs/Tips and Tricks uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the session.