Session 7

Web Workers

Objectives

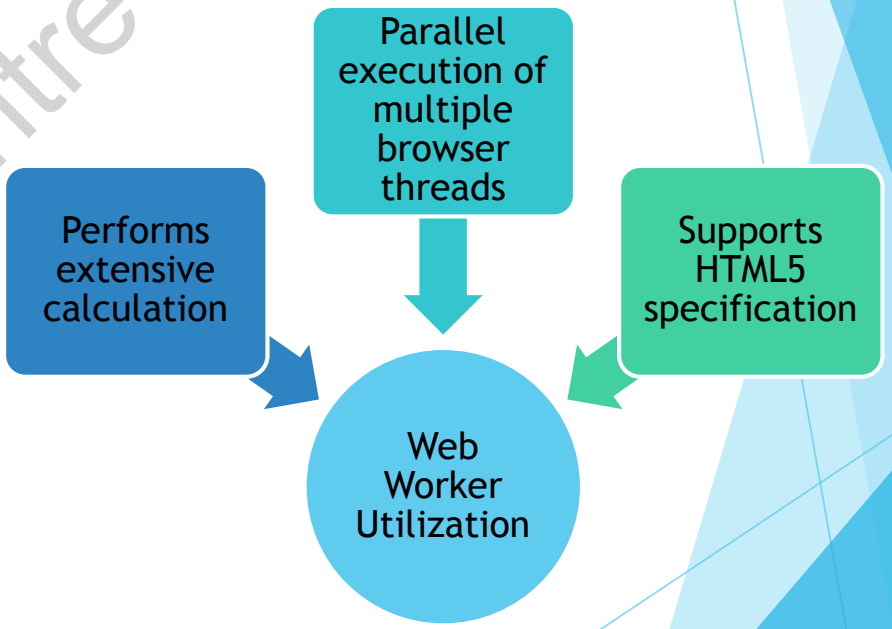
By the end of this session, students will learn to:

- 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

Overview of Web Workers

Web workers:

- Are JavaScript codes executed from an HTML page that run in the background
- Are not broken up by user interface scripts
- Allow tasks to be easily performed without any interruptions



Web Worker Limitations

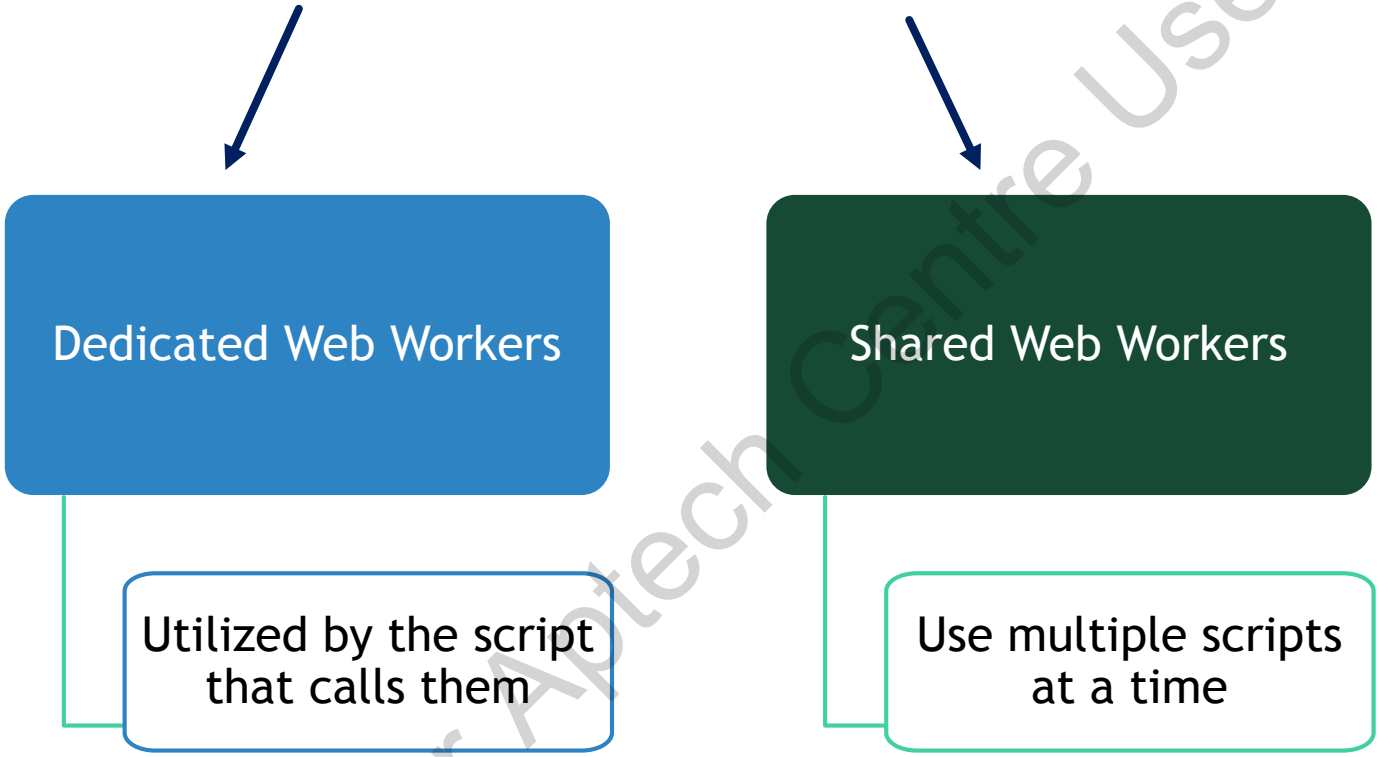
- Run in the background and are relatively heavy weight
- Do not have direct access to the Web page and the DOM API
- Consume the CPU cycles and make the system less responsive

Prerequisite to Run a Web Worker

Ensure that prerequisites to run a Web worker are in place.

Dedicated Web Workers

Web Workers are broadly categorized into two types:



Shared Web Server

There are two ways to send a message:

By copying the message

- The message is sequenced, copied, sent over, and then, randomized at the other end.

By transferring the message

- The original sender cannot re-use the message if it has been already transferred.

Checking for JavaScript Feature Access by Web Workers

Access Allowed	Access not Allowed
• 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	• Window object • Document object • Parent object • DOM (because it is not thread-safe)

Creating a Web Worker

Initialize with the URL of a JavaScript file that contains the code to be executed

Initiate a new worker thread, which is asynchronously downloaded

Import through importScripts() method

Establish communication between the Web worker and its parent page

Creating Web Worker File and Objects

- Consider a Web worker as an external JavaScript that needs to be created.
- Create a script to count the delay in posting a message.
- Write a code to check if the Worker already exists and if not, then create a new Web Worker object.
- Add an 'onmessage' event listener to the Web worker.

Terminating and Reusing a Web Worker

A Worker can be destroyed by calling the terminate() method.

After terminate() is called, the Web worker is demolished immediately with no chance to complete any ongoing or pending operation.

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

If the Worker variable is set to undefined, after it has been terminated, the code can be reused.

Handling Errors

Filename	Line number	Message
• Name of the Worker script where the error has occurred	• Line number where the error might have occurred	• Description of the error

Session Summary

- 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 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