Session 4

Grouping Scripts and Non-Blocking Scripts

Objectives

By the end of this session, students will learn 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

Grouping Scripts

- Developers have different files of JavaScript codes with multiple functions and handlers. Sometimes, it may require grouping.
- Grouping scripts allow to merge different JavaScript codes to form one script.
- Developer needs to fix inline or external before grouping scripts.
- Developer must be aware of the inline and external JavaScript files where it can be utilized.

Inline JavaScript

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.

External JavaScript

The external file can be executed from anywhere in the HTML code.

Developers can use an external JavaScript code stored in a .js file.

The Website takes additional time to load an external JavaScript.

To maintain the load speed, a developer can reduce the number of JavaScript files.

Balance between Requests and Cache-Ability 1-2

When an inline JavaScript code is embedded, no additional **requests** are made to the JavaScript resources placed externally.

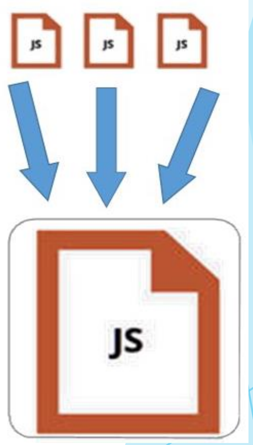
If the JavaScript file is bulky and is not changed too frequently, the file should be stored as an external file.

The number of external script files should be limited.

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

Reduce the time required to download external files from the browser by limiting the number of external scripts.

Reduce the number of external files by concatenating several external files into a single file.



Balance between Requests and Cache-Ability 2-2

- To concatenate several files with different code, copy and paste code present in all JavaScript files into a single file.
- At that time, 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.
- Online tools for merging files include **require.js** (https://requirejs.org/) and **merge.js** (https://www.filesmerge.com/merge-javascript-files).These tools can merge files at runtime as well.
- ‘Grouping’ or ‘Concatenating’ JavaScript helps in decreasing load time of a Web page that in turn gives a better user experience.

Blocking and Non-Blocking JavaScript Operations

A JavaScript operation is categorized as a Blocking operation and Non-Blocking operation.

Blocking

- It waits for the process to execute until an I/O operation is completed.

Non-Blocking

- It does not wait for an I/O operation to be executed.

Node.js and Callback

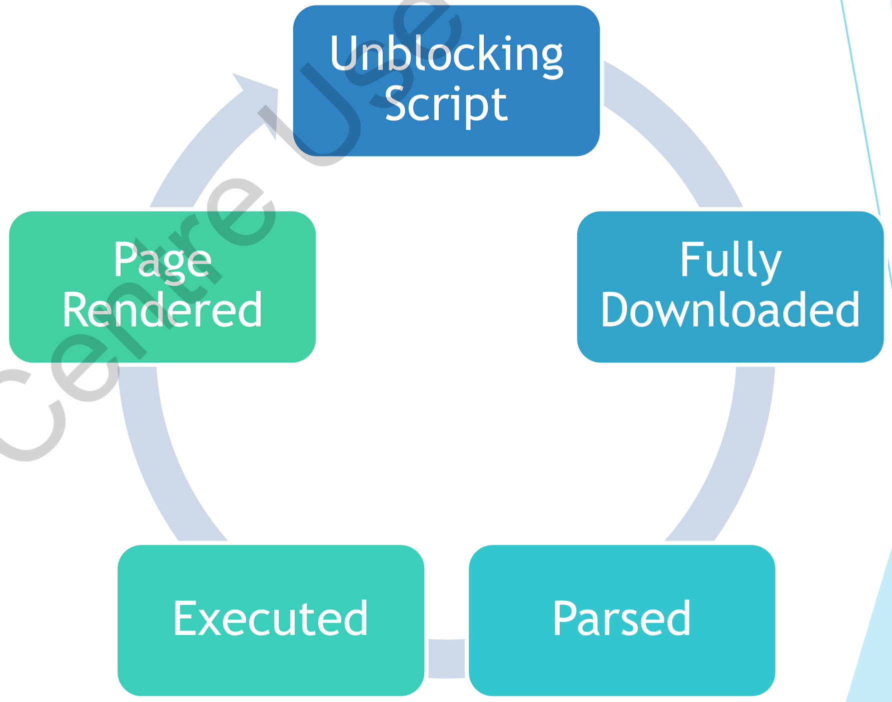
Node.js	Callback
• Cross-platform runtime environment that delivers certain in-built JavaScript functionalities. • Node functions do not perform I/O directly and processing happens uninterrupted. • As there are no blockings, Node functions prove to be extremely effective in developing 'scalable systems.'	• 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.

Blocking

- Blocking means execution of an additional JavaScript in the code. Execution must wait until a non-JavaScript operation completes.
- Usually, synchronous methods in standard library of Node.js and libuv are common blocking operations.
- The operation of reading a file is a single thread operation and should be executed fully before another function can take 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.

Non-Blocking

- Non-blocking in JavaScript means that the request for the I/O has been made and will be processed once the resources are freed.
- If there is a script that is blocking, the page will not render the operation until the script is:
  - Fully downloaded
  - Parsed
  - Executed
- When an external script is loaded using a <script> tag, the load time required is far more than expected.



Functional sequence of Non-Blocking Script

Understanding Non-Blocking I/O in JavaScript

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'. It is possible to use the same thread for a different function until the readable data becomes available.

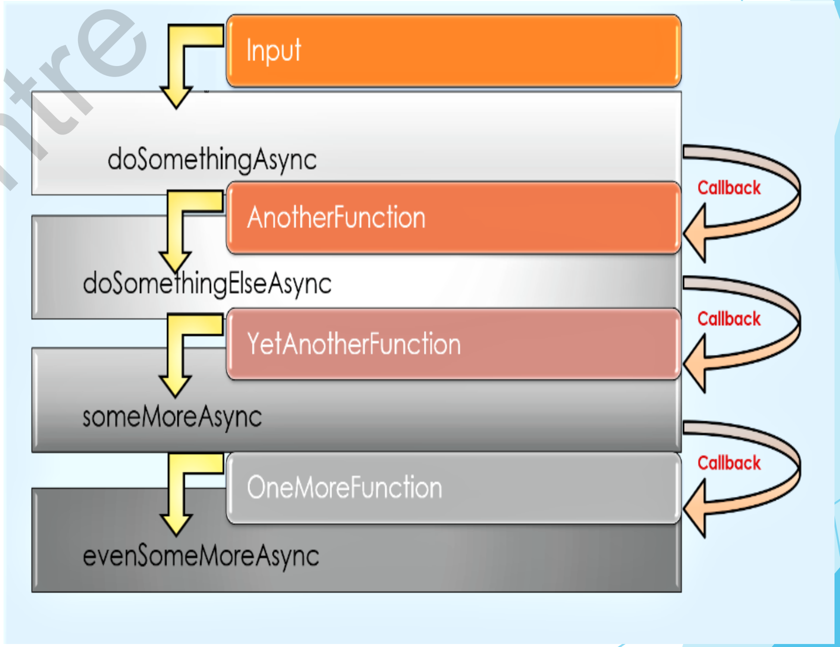
The event loop speeds up the processing time, improves the time required to read and write data, and improves error handling.

The blocking approach operates synchronously.

The non-blocking approach operates asynchronously.

Creating Non-Blocking JavaScript 1-3

- Embedding asynchronous scripts in C or in Java needs threads.
- Non-blocking or asynchronous scripts offer the best advantage of JavaScript.
- Consider an example to understand the functional flow of a non-blocking script.



Creating Non-Blocking JavaScript 2-3

Single-Threading

- 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.

Synchronous Functions

- Synchronous functions are concerned, each statement completes the task before the next statement is run.

Asynchronous Functions

- Asynchronous tasks can be called on as and when required and can also be put aside when data or call is not available.

Creating Non-Blocking JavaScript 3-3

Event Loop

- 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 function, an additional background thread is called up on. This thread helps manage the event loop.

Investigating the Exception of a Long Running Script

- When a maximum statement count exceeds while running JavaScript in certain browsers, an exception regarding unresponsive script is triggered.

Session Summary

- 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 that 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.