BITS Pilani
Pilani Campus

Full Stack Application Development- SE ZG503

Akshaya Ganesan

CSIS, WILP

innovate achieve lead

BITS Pilani
Pilani Campus

Lecture No: 12 Frontend- Web API

Client-side JavaScript API

- Client-side JavaScript has many APIs available.
- They are not part of the JavaScript language itself.
- But they are built on top of the core JavaScript language
- Client-side JavaScript API s fall into two categories
    - Browser APIs
    - Third Party APIs

Browser API

- A Browser API can extend the functionality of a web browser.
- All browsers have a set of built-in Web APIs to support complex operations, and to help accessing data.
- For example, the Web Audio API, Geolocation API

Third-party APIs

- **Third-party APIs** are not built into the browser by default.
- Third-party APIs are constructs built into third-party platforms (e.g. Google Maps API, Facebook APIs)
- They allow you to use some of those platform's functionality in your own web pages

Common APIs

- **APIs for manipulating documents**
  - Example: DOM API
- **APIs that fetch data from the server**
  - Example: Fetch API
- **APIs for drawing and manipulating graphics**
  - Example Canvas API
- **Audio and Video APIs**
  - Example: Web Audio API
- **Device APIs**
  - Example: Geolocation API
- **Client-side storage APIs**
  - Example: Web Storage API

Client-side storage

- There are numerous methods for storing data locally in the users' browser
  - Cookies
  - Web Storage (Local and Session Storage)
  - IndexedDB
  - Centralized data store ; State management libraries can be used.

Cookies

- A cookie is a small piece of data that a server sends to the user's web browser.
- The browser may store it and send it back with the next request to the same server.
- It remembers stateful information for the stateless HTTP protocol.
- They're the earliest form of client-side storage commonly used on the web.

Web Storage API

- The Web Storage API provides mechanisms by which browsers can store key/value pairs

- The two mechanisms within Web Storage are as follows:
  - sessionStorage maintains a separate storage area for each given origin that's available for the duration of the page session
  - localStorage does the same thing, but persists even when the browser is closed and reopened.

- The two types of storage areas are accessed through global objects named “window.localStorage” and “window.sessionStorage”.

Web Storage API

- Data is stored as key/value pairs, and all data is stored in string form.
- Data is added to storage using the setItem() method.
- setItem() takes a key and value as arguments.
- If the key does not already exist in storage, then the key/value pair is added.
- If the key is already present, then the value is updated.
- sessionStorage.setItem("foo", 3.14);
- localStorage.setItem("bar", true);

Reading Stored Data

- To read data from storage, the getItem() method is used.
- getItem() takes a lookup key as its sole argument. If the key exists in storage, then the corresponding value is returned.
- If the key does not exist, then null is returned.

- var number = sessionStorage.getItem("foo");
- var boolean = localStorage.getItem("bar");

Removing Stored Data

- To delete individual key/value pairs from storage, the removeItem() method is used.
- The removeItem() method takes the key to be deleted as its only parameter.
- If the key is not present then nothing will happen.
- sessionStorage.removeItem("foo");
- localStorage.removeItem("bar");

The storage Event

- A user can potentially have several instances of the same site open at any given time.
- Changes made to a storage area in one instance need to be reflected in the other instances for the same domain.
- The Web Storage API accomplishes this synchronization using the “storage” event.
- When a storage area is changed, a “storage” event is fired for any other tabs/windows that are sharing the storage area.
- Note that a “storage” event is *not* fired for the tab/window that changes the storage area.

Indexed DB

- IndexedDB is a transactional database embedded in the browser.
- The database is organized around the concept of collections of JSON objects similar to NoSQL databases
- IndexedDB is useful for applications that store a large amount of data (for example, a catalog of DVDs in a lending library) and applications that don't need persistent internet connectivity to work (for example, mail clients, to-do lists, and notepads).
- Each IndexedDB database is unique to an origin
- IndexedDB is built on a transactional database model.

Indexed DB

- Database - This is the highest level of IndexedDB. It contains the object stores, which in turn contain the data you would like to persist.
- Object store - An object store is an individual bucket to store data. Similar to tables in traditional relational databases.
- Operation - An interaction with the database.
- Transaction - A transaction is wrapper around an operation, or group of operations, that ensures database integrity.

Indexed DB Example

- https://mdn.github.io/dom-examples/indexeddb-api/index.html

BITS Pilani
Pilani Campus

Node JS Ecosystem

Node.js

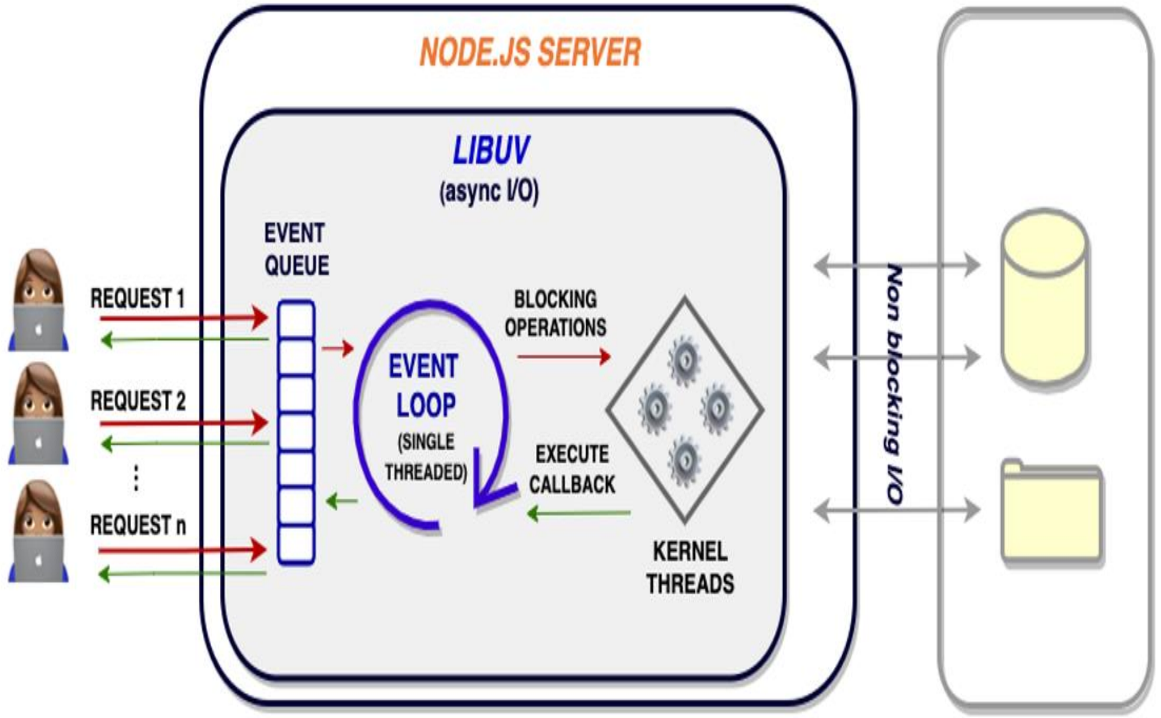
- Node.js is an open-source and cross-platform JavaScript runtime environment.
- Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser.
- A Node.js app runs in a single process, without creating a new thread for every request.

Why Node.Js is single threaded

- Node.js runs your JavaScript code in a single thread, which
- It doesn’t block the main thread for slow tasks.
- Instead, it uses non-blocking I/O and delegates slow work to background threads.
- This makes Node.js lightweight, efficient, and scalable — perfect for web servers, APIs, and real-time apps.

Why Node.Js is single threaded

- The event loop is a special mechanism in Node.js that:
  - Receives tasks (like I/O requests).
  - Passes slow tasks (e.g., file reads, DB calls) to background threads (not the main one!).
  - Continues running other code.
  - When the slow task finishes, the result is sent back to the main thread, which then runs your callback function.



Differences between Node.js and the Browser

- Both the browser and Node.js use JavaScript as their programming language.

Feature	Node.js	Browser
Purpose	Server-side JavaScript	Client-side JavaScript
Global Object	global	window
Access to DOM	No	Yes
File System Access	Yes (using fs module)	No (for security reasons)
Networking (TCP/UDP/HTTP)	Full access (sockets, HTTP servers, etc.)	Limited to HTTP/HTTPS via fetch, XHR
Modules	Uses CommonJS (require) or ES Modules	Uses ES Modules (import)
Execution Environment	Runs on a server, terminal, or command line	Runs in a web browser (Chrome, Firefox, etc.)
Security	Has full system access, so requires caution	Sandboxed (limited access to protect user data)
APIs Available	Node APIs (fs, http, path, etc.)	Web APIs (DOM, fetch, localStorage, etc.)
Use Cases	Backend apps, APIs, scripts, tools	UI, interactivity, animations, form handling
Event Loop	Handled by Node.js with libuv	Handled by browser (e.g., V8 in Chrome)
Threading	Single-threaded with background thread pool	Single-threaded with Web Workers for multithreading

Node JS Ecosystem

- **Node.js Core**
- Node.js was built using the V8 JavaScript Engine
- Compiles JavaScript to machine code

- **Node.js Runtime**
- event-driven, non-blocking I/O model

- **NPM (Node Package Manager)**
- Command-line tool
- Manages Dependencies

- **Express.js**
- Express.js is a popular, minimalistic web application framework for Node.js.

NPM

- npm is the standard package manager for Node.js.
- npm installs, updates and manages downloads of dependencies of your project.
- Dependencies are pre-built pieces of code, such as libraries and packages, that your Node.js application needs to work.
- If a project has a package.json file, by running npm install, it will install everything the project needs, in the node_modules folder, creating it if it's not existing already.

Modules

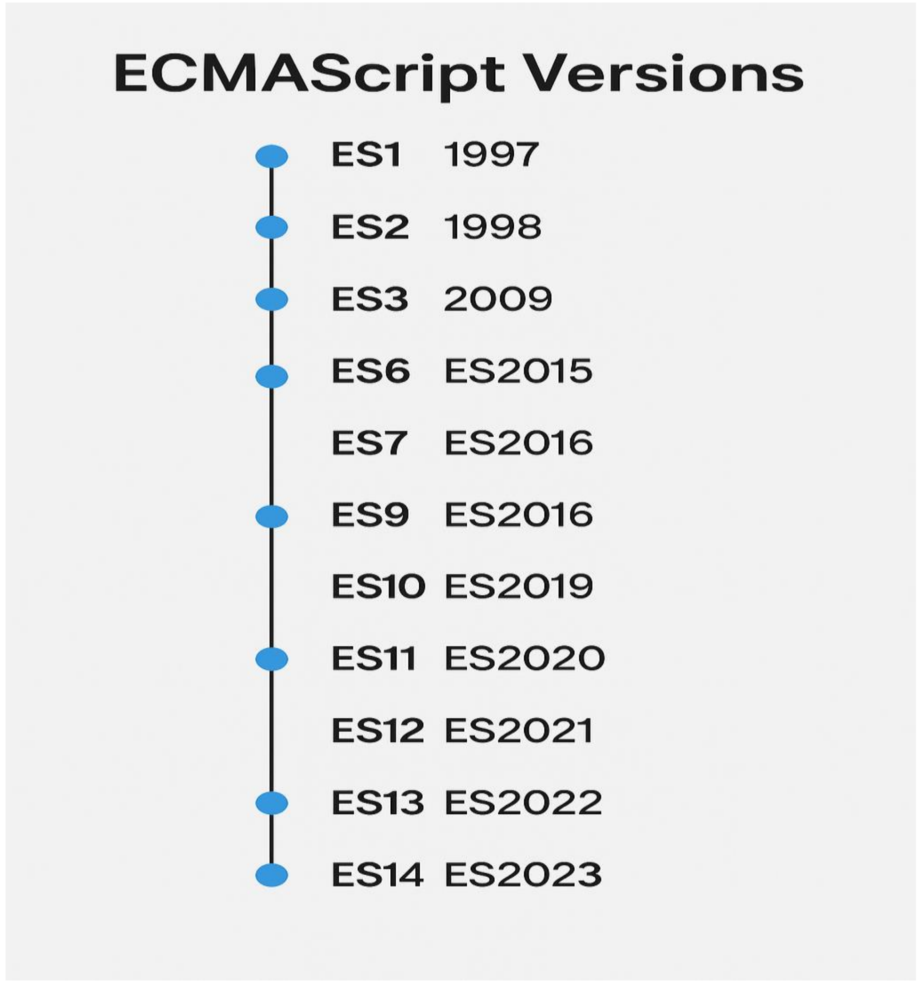
- CommonJS (CJS) – The Traditional Way (Used in Node.js)
  - Used by default in Node.js.
  - Uses require() to import.
  - Uses module.exports to export.
- ECMAScript Modules (ESM) – The Modern Standard (Used in Browsers and Node.js)
  - Official JavaScript module syntax (supported by both browser and Node).
  - Uses import / export.
  - You must use .mjs extension or set "type": "module" in package.json.

Webpack

- **Webpack** is a powerful **module bundler**
- **Webpack** is a tool that takes your **source code** (JavaScript, CSS, images, etc.) and **bundles** it into a single file (or multiple files) that can be easily deployed to a web server.
- Its main purpose is to **optimize** and **organize** your code, making it

ECMAScript

- **ECMAScript (often shortened to ES)** is the **official standard** that defines how the **JavaScript language** should work.
- **Modern JavaScript = ES6 and beyond**.



ECMAScript Versions

ES1	1997
ES2	1998
ES3	2009
ES6	ES2015
ES7	ES2016
ES9	ES2016
ES10	ES2019
ES11	ES2020
ES12	ES2021
ES13	ES2022
ES14	ES2023

Transpiling

- Transpiling refers to the process of converting ECMAScript 6 (ES6) code or the latest code into an older version of JavaScript that is more widely supported by browsers and environments.
- **Babel** is the most popular transpiler for ES6.

References

- https://www.jsv9000.app/
- https://eloquentjavascript.net/11_async.html
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Event_loop

Thank You!