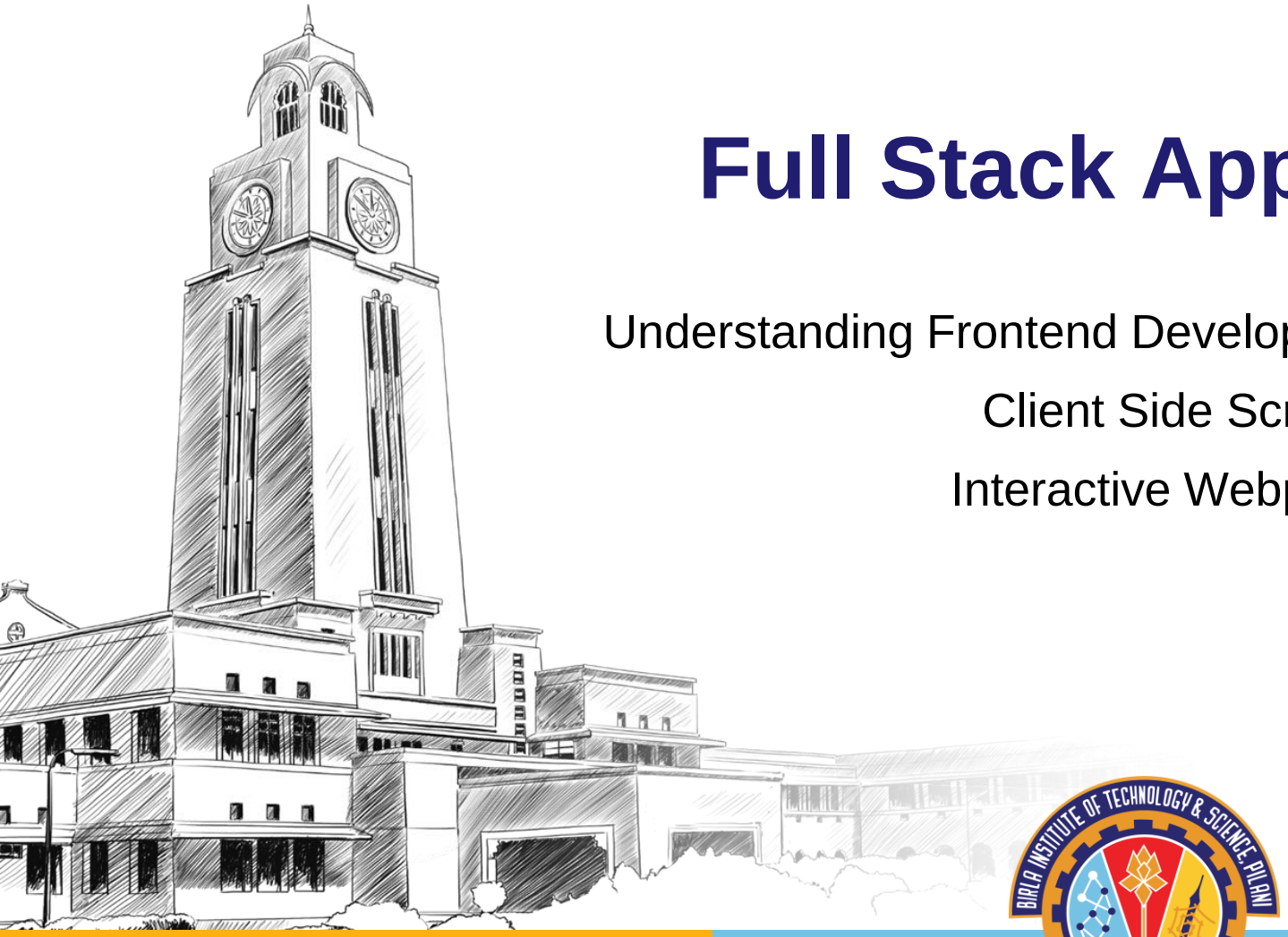


Full Stack Application Development

Understanding Frontend Development

Client Side Scripting

Interactive Webpages

Akshaya Ganesan

Frontend technologies

- What happens when you load your webpage in the browser?
- The code (the HTML, CSS, and JavaScript) are running inside an execution environment (the browser).
- HTML is the markup language that we use to structure and give meaning to our web content.
- CSS is a language of style rules that we use to apply styling to our HTML content, for example, setting background colors and fonts and laying out our content in multiple columns.
- JavaScript is a scripting language that enables you to create dynamically updating content and adds interactivity to your webpage.

Javascript

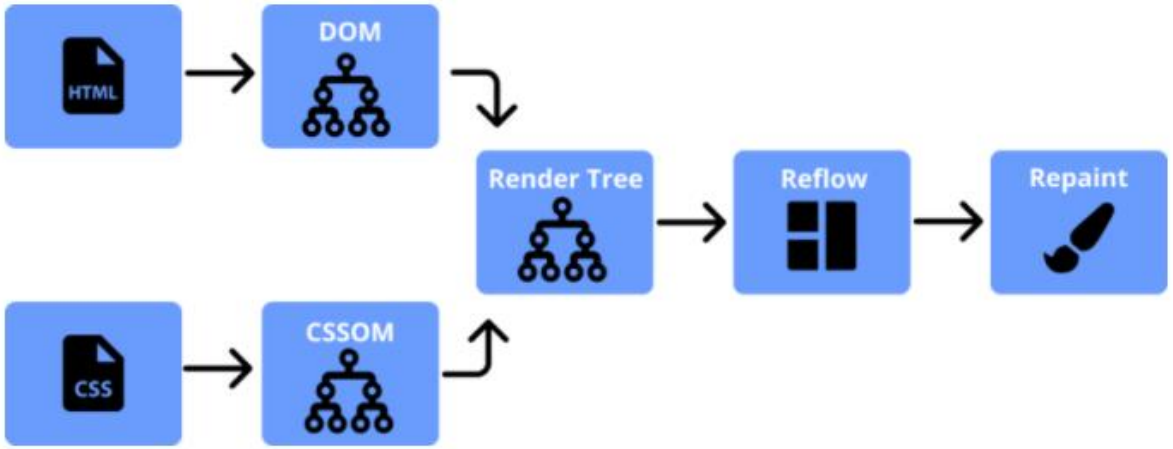
- JavaScript is the programming language of the web.
- JavaScript is a single-threaded interpreted language.
- Every browser has its own JavaScript engine. Google Chrome has the V8 engine, Mozilla Firefox has SpiderMonkey.
- The core client-side JavaScript language consists of some common programming features like variables, objects, functions etc.,

Host Environment

- Browsers- Initially only implemented in web browsers
- Now it can be used on Server Side

Client Side(Browser)	Server Side(NodeJS)
• Different browsers provide their own JS engines • Allows interaction with web page(HTML and CSS) • Interact with browser API s (History, Location)	• Google' s V8 was extracted to run anywhere- Node.js • Node is a fast C++-based JavaScript interpreter • Work with file systems, Web servers • Knowledge Reuse

The Critical Rendering Path(Recap)



Execution in Javascript

JavaScript is a single-threaded programming language.

An Execution Context is an abstract concept of an environment where the JavaScript code is evaluated and executed.

Execution in Javascript

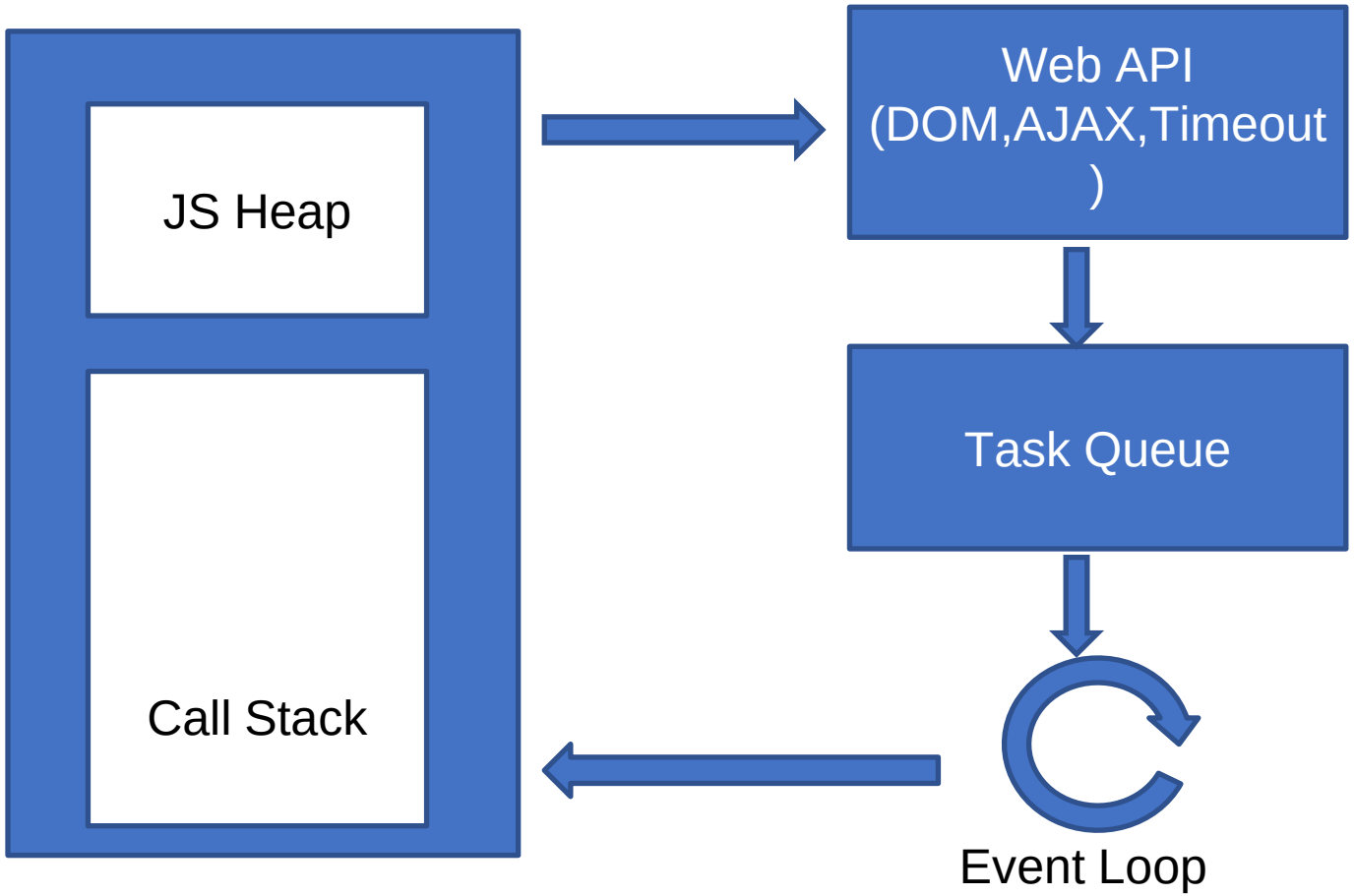
The JavaScript Engine consists of two main components:

Memory Heap: this is where the memory allocation happens

Call Stack: this is where your stack frames are as your code executes. As its name implies, the call stack has a LIFO (Last in, First out) structure, which stores all the execution context created during the code execution.

JavaScript Execution in browser

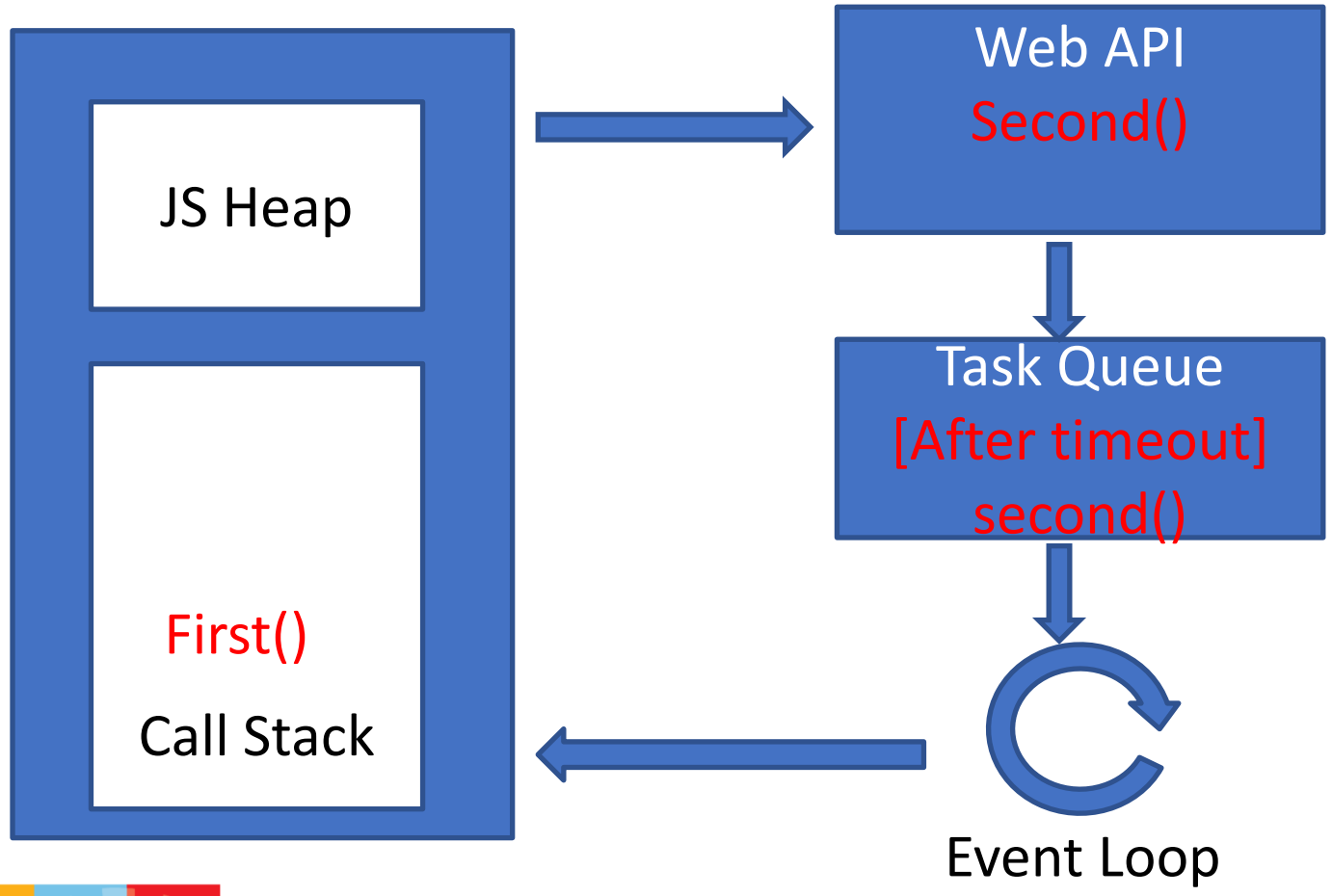
- Synchronous Execution




```
<script>
        const second = () => {
            console.log('Hello there!');
        }
        const first = () => {
            console.log('Hi there!');
            second();
            console.log('The End');
        }
        first();
    </script>
```

JavaScript Execution in browser

- Asynchronous Execution




```
<script>
    const second = () => {
        console.log('Hello there!');
    }
    const first = () => {
        console.log('Hi there!');
        setTimeout(second, 1000);
        console.log('The End');
    }
    first();
</script>
```

WebAPIs

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.

Node Ecosystem

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.

Node JS Ecosystem

- **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

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.

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

Thank You!