Week 1 Exploration

- Zainab Nusaiba

Objective:

To explore and document your view on call-stack, event loop, and how it helps modern-day browsers to render content. Also, document your understanding of

- How es6 is changing the landscape of web experiences
- Is PWA still relevant today?

Synopsis:

-——> Call-stack
-——> Event loop
-——> Call-stack and Event loop for rendering content
-——> ES6
-——> PWA

Call-stack:

Call-stack is used by an interpreter to manage all the calls that have been made. The call stack is generally used by Javascript in web browsers. As the name suggests, it keeps track of all the function calls that are made, are running, and are supposed to be done next in the form of a stack. Hence it's one thread, one stack that runs one thing at a time.

1. Basic Functioning:
-——> It is like a stack data structure with the push and pop operations
-——> Stack is knowing what operation and the function call is to be carried out
-——> ***push*** operation leads to pushing the presently called function onto the stack
-——> ***pop*** operation leads to popping out or removing the function once it's done or due to the command
-——> It works for the ***LIFO*** principle which refers to Last In First Out, it means the last function that gets pushed into the stack is the first to be popped out when the function returns

2. Mechanism
-——> Script is calling a function
-——> The calling function gets added to the call stack
-——> The function is carried out

-——> Additional functions within a function or next in the script are added to the stack on top
-——> Based on the LIFO principle, function calls on top of the stack are executed first
-——> After the current function is executed, the stack pops that function out of the stack
-——> Incase the stack takes more space than the allocated stack space, it shows an error of "Stack Overflow"

3. Example

-——>

```
function greeting() {
  morningMessage();
}

function getUsername() {
  document.write("Enter Username");
 //input username-scanf
}

function  printUsername() {
  return username;
}

function morningMessage() {
  sayHi();
  printUsername();
  document.write( "Good Morning");
}

function sayHi() {
  return "Hi! ";
}

getUsername();
greeting();
```

Event Loop:

The implementation of Event Loop is asynchronous one which means it waits for a message to arrive in case not mentioned earlier, and executes after it receives the acknowledgment.

1. Functioning - JavaScript:

-——> Javascript is a single-threaded synchronous programming language
-——> The asynchronous parts are handled by the Web APIs
-——> ***Single-threaded*** means that the interpreter runs only one line at a time and not in multiple lines, ways, or threads

-——> It works on the principle of ***FIFO*** - First In First Out
-——> Event Loop gives the illusion of it being ***multi-threaded***
-——>When the function stack becomes empty, event loop pulls the function from the queue and puts it in the execution stack

2. Mechanism - JavaScript:
-——> Consider Stack, Web API, and Event queue
-——> The functions in the stack get executed and in the case of callback functions, are waiting for the setTimeout() to get over so that the function can be executed.
-——> The setTimeout() then does a callback to the Web API considering there is a callback
-——>Then when the stack is empty, the function call which has been waiting in the event queue is acknowledged
-——>After acknowledging, the function calls of event queue is sent to the execution stack
-——>This hence acts like a secret loop

Call-stack & Event Loops for rendering content:

1. **Callback queue and event loop**

-——> After the timer gets expired, the callback function is put inside ***callback queue***
-——> Event loop checks if call-stack is empty if so, it pushes the callback function from the callback queue to the call stack
-——> Then that callback function is removed from the callback queue

2.Role of Web APIs
-——> Javascript is synchronous so for it to execute in a multithreaded, asynchronous manner, Web APIs are used.
-——> Web APIs are the additional contents that the browser contains for it to function collaboratively
-——> Web APIs include

- setTimeout() - access to timers and completing function calls after a certain period
- DOM APIs - connecting scripts/ webpages to the structure of the document
- fetch() - to fetch that argument from its source or helps connect to other servers
- local storage
- Console
- location (website URL)

-——> Browser allows the Javascript engine to use all of these Web APIs through a keyword called “Window” which is the global object

3. Basic Functioning
-——> Event loop constantly monitors the call stack and the callback queue

-——> Event loop sees whether the call stack is empty and that the callback queue has a function waiting to be executed
-——> When this happens, the event loop pushes the function from the callback queue into the call stack
-——> The function pushed from callback queue is removed from the queue
-——> The current function in call-stack is executed inside the console and after executing it is removed from the call-stack

ES6:

ES6 is the newer version of Javascript ever since 2015. ES6 is changing web experiences as it has newer syntaxes, and features that are easy to use and more readable. It defines the baseline conditions for Javascript developers. Following this, the developers can achieve the same rendering across various browsers. The main features making it different include:

- Transpilers - It can translate es6 to es5 if that browser is accustomed to it
- Keyword ' const ' - unchangeable assignment of variable
- Template literals - avoids adding excessing string concatenation('+')
- Arrow functions - single line fn for small operations
- Iterators - forEach
- Default params - assign values in params
- Internationalization - can format & display currency, time, and numbers for various countries

PWA:

It is also known as the Progressive Web Apps. It has been in the market for a long time. It is also considered the future as they have an additional advantage over the regular web app. It uses certain technologies to be able to serve both web apps and native web apps.

Progressive Web Apps use the Service worker API, web app manifests, and other progressive enhancements to provide the native app users with the web app experience.

Gmail, Starbucks, and Amazon are some examples.

Service worker API act as the proxy servers that sit between the web application, web browser, and the network. This is the essential feature that provides a smooth offline experience to the user. It updates its data on the server whenever necessary and also manages to send notifications to the users as per requests. It takes up and performs various tasks depending on the actions asked and whether the network is available for that task.

Web app manifests are the link to websites that are directly downloaded to one's home screen without having to download the app. In the backend, it provides the information about the web application in a JSON text file

progressive enhancements mean creating a simple easy-to-use design whose main basis is accessibility. It is to ensure that the old devices, small screens, and acceptable accessibility features and devices with limited capabilities are all considered. At the same time, it is to consider that the new versions and the advanced devices get the best experience possible.

Week 1 Exploration (Mentor) - Zainab