05/05

Name :- Ajay Deshmukh
Roll no : 11
Div :- D15B

MPL Assignment 02

Define progressive webapp (PWA) and explain its significance in modern web development. Discuss the key characteristics of PWAs from traditional mobile apps.

⇒ A progressive webapp (PWA) is a type of web application that works like a mobile app but runs in a browser.

Significance of PWA in modern web development:

1) Cross-platform Compatibility.
2) Offline Support.
3) Fast performance
4) No app store required
5) Lower Development Cost

Difference in PWA & traditional apps:

Features	PWA	Traditional App
Installation	Direct from browser	Download from app store
Internet Required	work offline with Caching	Usually requires internet
Performance	Fast with Service workers	Fast but need installation
updates	Automatic	Manual
Development Cost	Lower	Higher

Q.2. Define ~~regu~~ responsive web design and explain its importance in the context of progressive web apps. Compare and contrast responsive, fluid and adaptive web design approaches.

⇒ Definition of Responsive web design:-

Responsive web design (RWD) is a ~~techniq~~ technique that makes web Pages adjust automatically to different screen size and devices it ensures a good user experience on mobile on mobile, tablets and desktops without needing separate version of website.

Importance of Responsive Design in PWAs

1) Better user Experience
2) Faster Load time
3) SEO benefits
4) Cost effective

* Comparison :-

Approach	How it works	Pros	Cons
~~Res~~ Responsive	use flexible grids and css media queries	works on all devices	Com
Fluid	use % based widths and fixed pixels on different screen size	works well	?

Key Differences :-

1) Responsive adapts dynamically to all screens
2) Fluid resizes smoothly but may not be fully optimized
3) Adaptive loads different layouts based on device type.

Describes the lifecycle of service workers including registration, installation and activation phases

=> Lifecycle of Service workers :-

A services worker is a script that runs in the ~~browsers~~ background and helps a web app work offline, load faster and send push notifications, its lifecycle has three main phases

1) Registration Phase :- The browser registers the Service workers using javascript.

e.g.:

```
if ('serviceworker' in navigator) {
    navigator.serviceworker.register("/sw.js")
    .then () => console.log('Service Registered");
    .catch(error => console.log ("Registered Failed",
                                 errors));
}
```

2) Installation phase:
i) The service worker downloads necessary files (HTML, and stores the cache

ii) if successful, it moves to the activation phase.

Code eg:-

```
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open('app-cache').then(cache =>
            return cache.addAll(['/', '/index.html',
                'style.css']);
        })
    );
});
```

3) Activation phase:
→ The old service worker is replaced with new o
→ unused cache files from the previous version are deleted.

Final step: Fetch & nine
Once activated, the service worker intercepts network request. Serves cached files and types data.

Explain the use of indexed DB in the service worker for data storage.

⟹ Use of IndexDB in service worker for Data storage:-
IndexedDB is a browser database that stores large amount of structured data like json objects.
It helps PWAs work offline by saving and networking data efficiency.

Why use Indexed DB in Service workers?

1) Offline Support:- store data when offline and sync it later.

2) Efficient storage:- Saves structured data like user setting, cart items or form inputs

3) Faster Access:- Retrive Data quickly without needing a network registers

4] Persistent Data:- Data remains saved even after the browser is closed.

How Service workers use IndexedDB?

Opening the Database

```
let db;
let request = indexedDB.open('MyDatabase', 1);
request.onsuccess = function(event) {
    db = event.target.result;
};
```

Creating a store and adding Data

```
request.onupgradeneeded = function(event) {
    let db = event.target.result;
    let store = db.createObjectStore('users', { keyPath
    store.add({ id: 1, name: 'John Doe', age: 25 });
};
```

Fetching Data in Service Worker.

```
let transaction = db.transaction('users', 'readonly');
let store = transaction.objectStore('user');
let getUser = store.get(1);
get user.onsuccess = function() {
    Console.log(get.User result);
};
```