Name:- Ajinkya Sunil Patil [D15B-42]

05/05

MPL Assignment 02

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

Soln:-

A progressive web app (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 responsive web design and explain its importance inthe context of progressive web. Apps.
Compare and contrast responsive, fluid and adaeptive web design approaches.

Solns:-

Definition of Responsive web design :-
Responsive web design (RWD) is a technique that makes web pages adjust automatically to different screen size and devices it ensures a good user experience onmobile 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.

Comparisons :-

| Approach | How it works | Pros | Cons. |
|---|---|---|---|
| Responsive | Use flexible grids and CSS media queries. | works on all devices | Can be Complex |
| Fluid | Use % based widths instead of fixed pixels | work well on different Screen size | Less Control over |

Key difference :-
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.

Q.3] Describes the lifecycle of services workers, including registration, installation and activation phases.

Soln :-
Lifecycle of services workers :-
A services worker is a script that runs in the backgro-
-und and helps a web app work offline, load faster and send push notifications. its lifecycle has three main phases :-

1] Registration phase.
:- The browser register the service workers using Javascript.

e.g. :-
```
if ('serviceworker' in navigator) {

    navigater.serviceworkers.register ("/sw.js")
      .then () => console.log ('service Registered"))

      .catch (error => console.log ("Registered failed:",
                                                   error));
}
```

2] Installation phase.

i) The service worker downloads necessary files (HTML, CSS, JS) and stores them in 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',
                                 'styles.css']);
        })
    );
});
```

3] Activation phase.

→ The old services worker is replaced with new one.

→ unused cache files from the previous version are deleted

Final step: Fetch & sync.

Once activated, the services worker intercept network request, serves cached files and syncs data.

Q4] Explain the use of indexed DB in the. services worker for data storage.

Soln :-

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 retrieving data efficiency.

Why use IndexedDB in service workers?

1] offline support - store data when offline and sync its later.

2] Efficient storage - saves structure data like user setting, cart items or form inputs

3] faster Access - Retrieves Data quickly without needing a network request

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 ('My Database', 1);

request.onsuccess = function (event) {
    db = event.target.result;
  };
```

# Creating a store & adding Data

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

# fetching Data in service worker.

```
let transaction = db.transaction ('users', 'readonly');
let store = transaction.objectStore ('users');

let getUser = store.get(1);
get User.onsuccess = function (){
      console.log (get User result);
  };
```