

Q.1]

Define Progressive web App and explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps.

⇒

A progressive web App (PWA) is a type of web application that works like a mobile app but runs in a browser. It can be installed on a device, work offline and provides a fast and smooth user experience.

Significance of PWA in Modern web development:-

- 1] cross-Platform compatibility:- Works on both mobile and desktop with a single codebase.
- 2] offline support:- Can function without the internet using cached data.
- 3] Fast Performance:- loads quickly, even on slow networks.
- 4] NO App Store Required:- Users can install it directly from browser.
- 5] lower Development cost:- One PWA can replace separate Android and iOS apps.

Key Differences Between PWA and Traditional Mobile apps:-

Feature	PWA	Additional Mobile App
Installation	Direct from browser	Download App from App store
Internet Required	Works offline with caching	Usually requires internet
Performance	Fast with service worker	Faster but needs installation
updates	Automatic, no app store approval	Manual updates needed.
Development cost	Lower (one codebase for all)	Higher (separate apps for each platform)

Q. 2] Define responsive web design and explain its importance in 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 technique that makes web pages adjust automatically to different screen sizes and devices. It ensures a good user experience on mobiles, tablets and desktops without



needing separate versions of website.

Importance of Responsive Design in PWAs :-

- 1] Better user experience :- PWAs work smoothly on any device.
- 2] Faster load time :- optimized design improves speed.
- 3] SEO Benefits :- google ranks responsive sites higher.

comparison of ~~web~~ Web Design Approaches :-

Approach	How It works	Pros	Cons
Responsive	Uses Flexible grids and CSS media queries to adjust layout.	works on all devices improved SEO.	can be complex to design
Fluid	Uses percent-based widths instead of fixed pixels, so elements resize smoothly	works well on different screen sizes, easy to implement	less control over layout on large screens.
Adaptive	uses fixed layout that change at	optimized for known	more efforts required

screen size

to design  
for each  
screen size.

### Key Differences:-

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

Q. 3]

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

⇒

lifecycle of service workers

A service worker is a script that runs in 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 worker using Javascript.



code Example :-

```
if ('service worker' in navigator) {
  navigator.serviceWorker.register('sw.js')
    .then() => console.log('Service Worker
    Registered')
    .catch(error => console.log('Registration
    failed:', error));
}
```

- This tells the browser to install and activate the Service Worker.

2] Installation Phase

- The Service Worker downloads necessary files (HTML, CSS, JS) and stores them in cache.
- If successful, it moves to the activation phase.

code Example :-

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

- This ensures the app loads even without the internet.

### 3] Activation Phase

- The old service worker is replaced with the new one.
- Unused cache files from previous version are deleted.

code Example :-

```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(keys => {
      return Promise.all(keys.map(key => {
        if (key !== 'app-cache') {
          return caches.delete(key);
        }
      }));
    })
  );
});
```

- The service worker is now fully active and controls network request.

~~Final Step : Fetch & Sync~~

~~Once activated, the service worker intercepts network requests, serves cached files and syncs data when the internet is available.~~



Q.4]

Explain the use of IndexedDB in the service worker for data storage.

=>

Use of IndexedDB in service worker for data storage

IndexedDB is a browser database that stores large amounts of structured data like JSON objects. It helps PWs work offline by saving and retrieving data efficiently.

Why use IndexedDB in service workers?

- 1] Offline support :- stores data when offline and syncs it later.
- 2] Efficient storage :- saves structured data like user settings, 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);
```

```
getUser.onsuccess = function() {
  console.log(getUser.result);
};
```