Creating Your First Progressive Web App with Angular

simplilearn

You Already Know

Course:

Learn to build Progressive Web Apps using JavaScript

Recap

- Explain the basics of PWA
  - Install node
  - Introduction to PWA

- Explore different tools
  - Service workers
  - Fetch API
  - Cache

- Build a PWA
  - Starting HTML
  - Finding woeids
  - Manifest file

A Day in the Life of a MEAN Stack Developer

In this sprint, Joe has to develop the frontend of the application using an Angular framework. Since he was recently trained in working on this framework, he has been chosen to lead this project and complete the initial part of the application. He agrees to complete the task as early as possible, as the release of the application is next week.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

- List the importance of an Angular framework
- Build an Angular application with customized requirements
- Create responsive forms for user input and validation
- Implement the routing mechanism in your Angular application
- Upgrade an AngularJS project to Angular 2+

Set Up PWA

Components of PWA

There are four major components in every progressive web application:

HTTPS
PWA will only run over HTTPS and not on HTTP. Local host is the only exception.

Web App Manifest
It is a JSON file which describes the basic information of the application.

Responsive Design
PWAs are required to be responsive as they run on different devices simultaneously.

Service Worker
It is a script to cache the application.

Arrow Function and Promises

Arrow functions and promises are the core concepts complimenting PWA.

Arrow Functions:
It is called at the time of its definition.

Promises:
It is an object returned from asynchronous function providing basic operation.

```
const phraseSplitterEs6 =
phrase => phrase.split(" ");

console.log(phraseSplitterEs6("ES6
Awesomeness"));
// ["ES6", "Awesomeness"]
```

```
var promise1 = new
Promise(function(resolve, reject) {
        setTimeout(function() {
        resolve('foo');  }, 300);});
        promise1.then(function(value) {
        console.log(value);
// expected output: "foo"});
console.log(promise1);
```

Service Worker

It is a script. It runs in your browser and helps manage caching of an application. It supports browser versions, Chrome >60 and Firefox >55.

It responds to the HTTP requests by interrupting outgoing requests of the application.

It works as a network proxy. One can completely program service worker-based caching without relying on server-based caching headers.

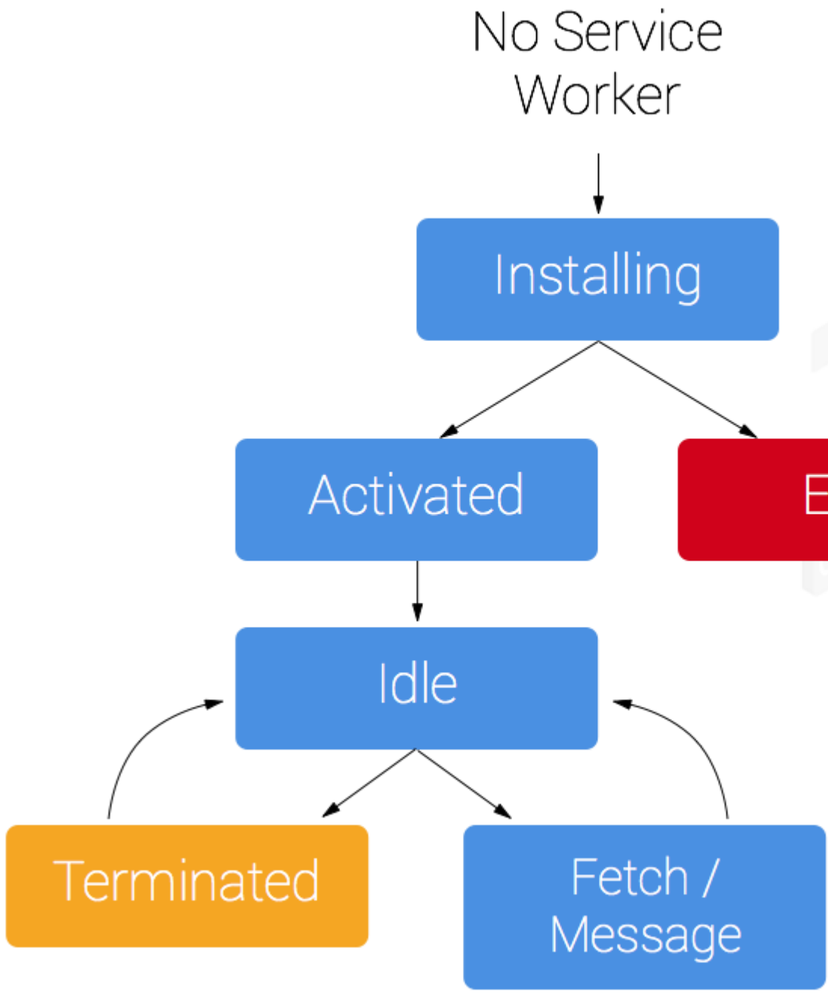
It helps to load the complete application locally without internet.

Service Worker Life Cycle

1. **Registration:** This is the starting point of the service worker. It can be registered as a JavaScript code or as an Angular package**.**

2. **Installation:** It is installed once the script is downloaded automatically. Service worker is installed only if it is not registered before or its code changes.

3. **Activation:** A Service worker is activated if there is no service worker active or **self.skipWaiting()** is called during install, or user refreshes the page.



Set Up PWA

Duration: 30 min.

Problem Statement:
You are given a project to install and add PWA functionality in an Angular application.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Demonstrate the Setup of PWA

1. Create an Angular application.
2. Install PWA and http-server.
3. Build and run the application.
4. Push the file to the GitHub repositories.

PWA: Deep Dive

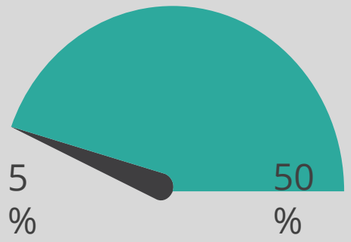
Service Worker in Angular

1. Service worker are compatible with Angular version greater than 5.0.
2. It prevents Angular developer to code for low-level APIs.
3. Angular service worker reduce network dependencies.
4. Angular service worker run on HTTPS and not on HTTP. It can be added to your Angular application with the command:
   ng add @angular/pwa
5. One can implement offline functionality to the application using service worker.

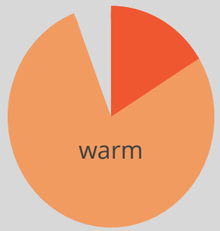
Service Worker Version Management

The collection of resources, files, plug-ins, and packages is called as a version of an application.

Service workers version depends on the app version or changes done in the app to update the service worker.



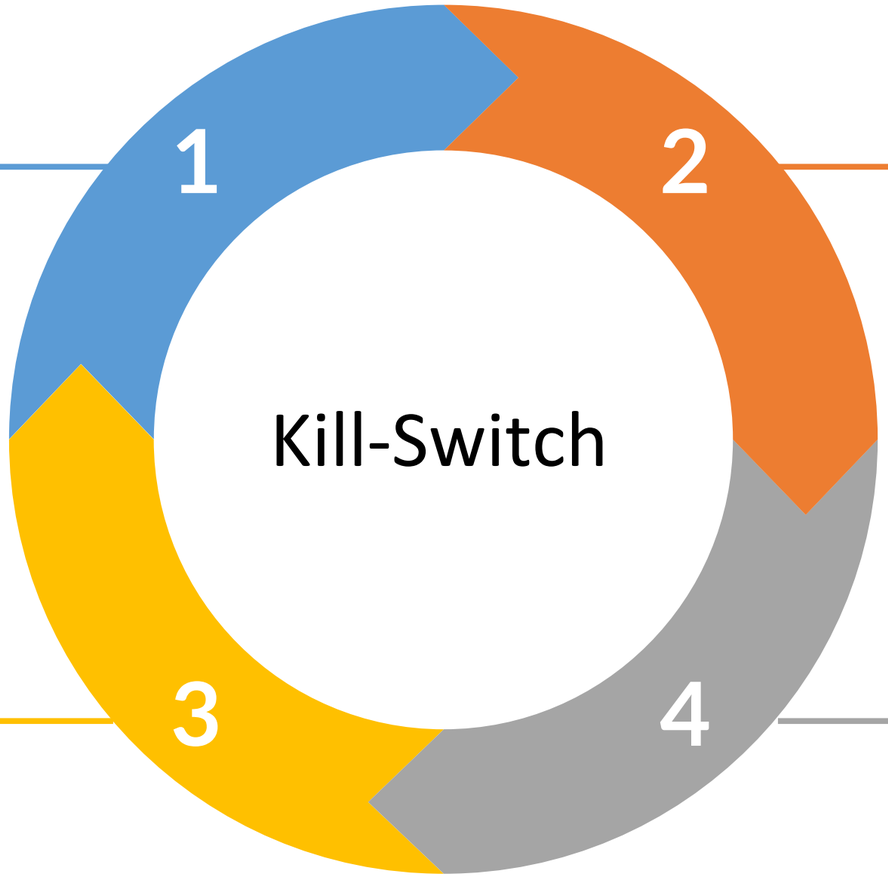
Service workers are updated every time there is new build made with the application.



User shall not hinder with the internal working of service worker as it automatically detects the version changes.

Service Worker Kill Switch

Kill-Switch phenomenon is useful when your service worker fails resulting in breaking of the web pages. The below sequence explains the working of Kill Switch:



Version Change

You keep the version number for appropriate caching.

Updating Cache

Add the version number with the update method to update the service worker.

Cross Site Scripting

Malicious code resulting into service worker bypassing.

Kill-Switch

To save the website, one clears the website information on user's device which is called Kill-Switch.

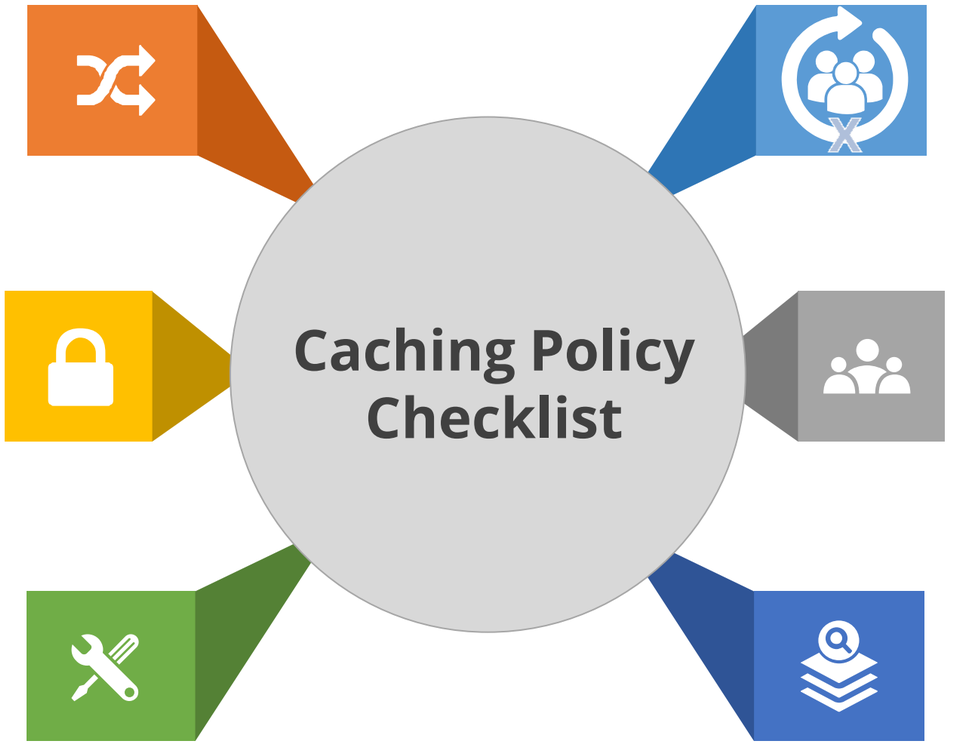
Performance Caching Policy

Caching technique and policies depends on various parameters like traffic pattern, type of data served, and application-specific resources.

One should provide separate file for the resources, which gets frequent updates to minimize the amount of downloaded content.

One should control the pace of updates by combining resource URLs.

Developer should determine the best cache lifetime for every resource.



User should avoid serving same content on different URL, as it may result in caching it multiple times.

It is one of the best practices to make use of validation token (Etag) to mitigate transfer of same data when server is same.

Make use of CDN and other intermediaries to cache common responses valid for all users.

Application to Home Screen

Duration: 35 min.

Problem Statement:
You are given a project to install and add PWA application to your local system.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Add PWA to Desktop Application

1. Create an Angular Application.
2. Install PWA and HTTP server.
3. Build and run the application.
4. Add application to home screen.
5. Push the files to the GitHub repositories.

Angular Application Shell

Angular Application Shell

Angular application shell is a collection of initial designs of the webpage required to be rendered and sent with the HTTP response.

Minimal
Specifying the minimal design in the first response helps in making the core data content without worrying about the peripheral components.

Fast
Once cached, loading is quick.

Economic
Once cached, the browser downloads less data which is economical for most of the users.

Native
Using app shell, one can easily provide native-like experiences with offline support.

App Shell and Angular Universal

Angular app shell compliments Angular universal code that helps in quick rendering of pages at user end.

Helps accelerate the server-side rendering process.

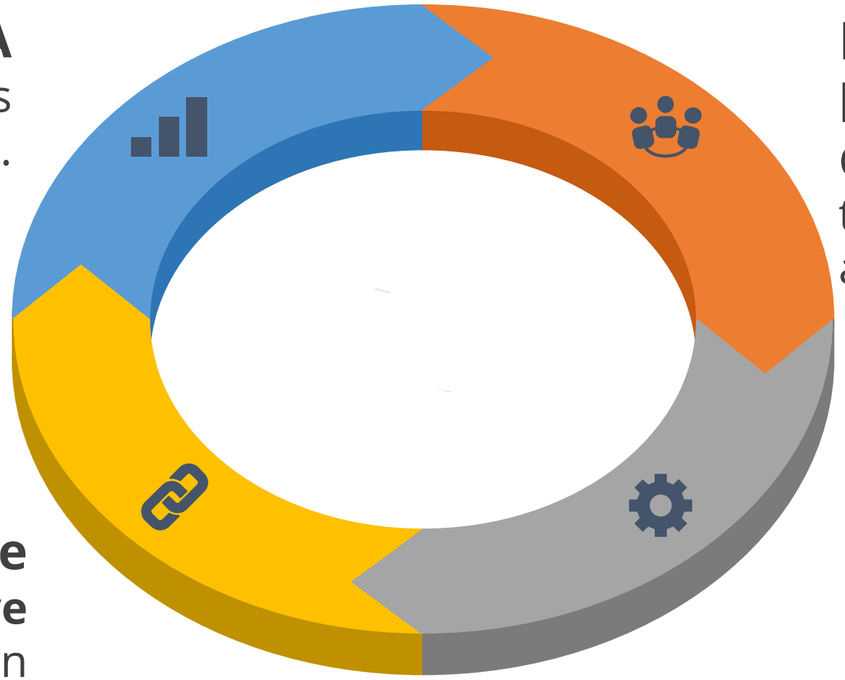
Instead of sending an empty app to the end user, you can add minimal design for the initial loading.

Provides control to pre-render components that are common for the complete application.

Working with Application Shell

Application shell can be added to the application using the following methods:

Scaffolding Angular PWA
Adding routing functionality is essential with app shell.



Profiling Application Startup before App Shell
Checking page startup time in Dev tools to get the components to be included in app shell.

Generating Production Mode
Using **ng run <appshell>** and **ng serve –prod ,** one can serve app in production mode.

Adding App Shell with CLI
Adding app shell in application by ng command for both Angular universal and app shell.

Adding App Shell in Current Application

Go through the below code in order to add app shell in your current application.

- Use the below code to generate app shell.

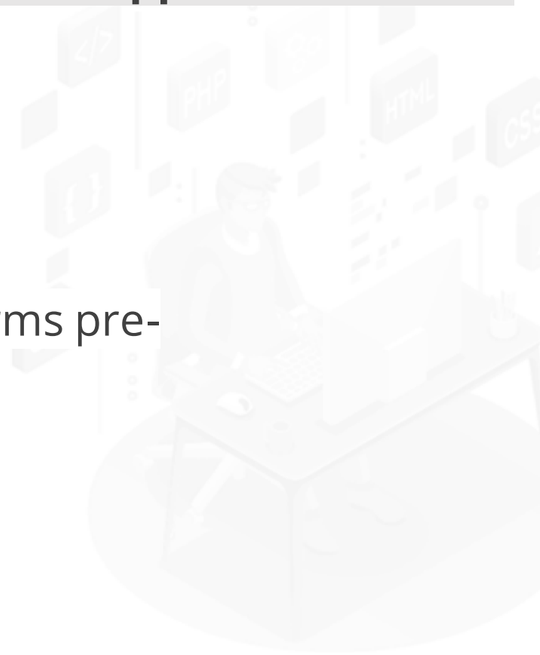
```
ng generate app-shell -client-project app-with-appshell -universal-project app-with-appshell-server
```

- **ng generate app-shell** code will create an app shell.
- **-client-project** code specifies the project to which app shell shall be created.
- **-universal-project** code specifies the universal/parent/server application that performs pre-rendering.
- To verify app shell use below commands:

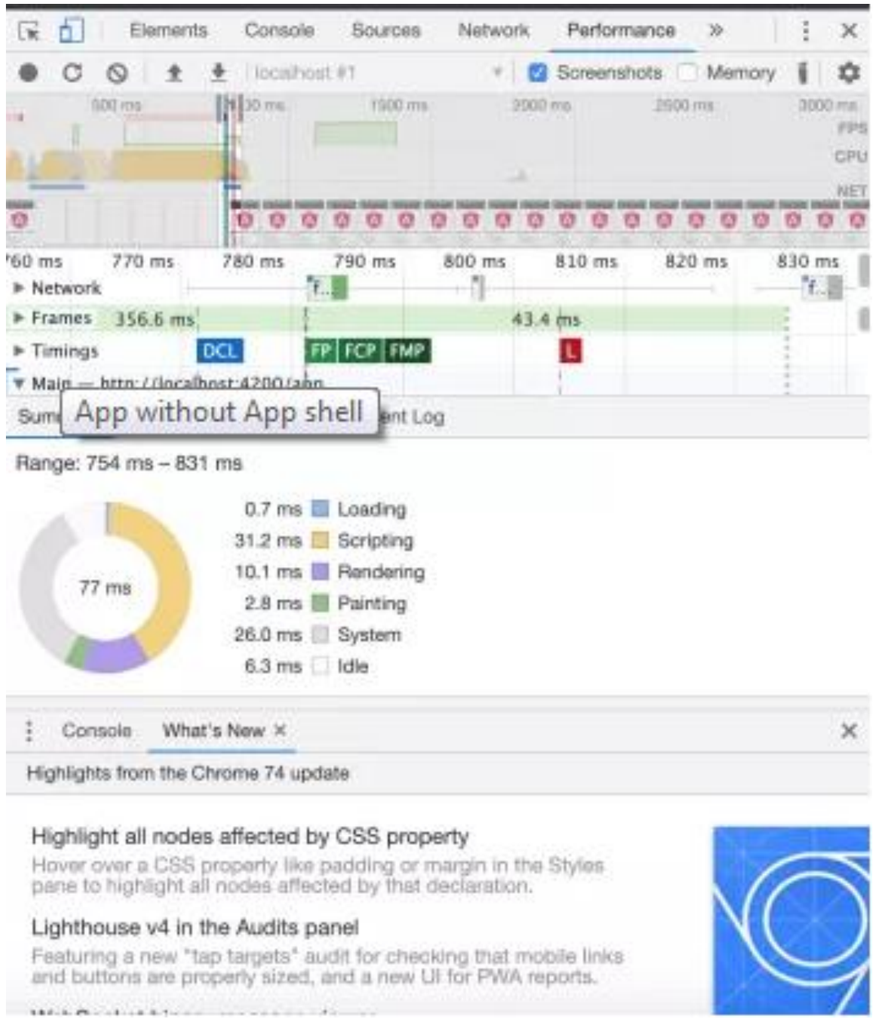
```
ng run app-with-appshell:app-shell

OR

ng build --prod
```

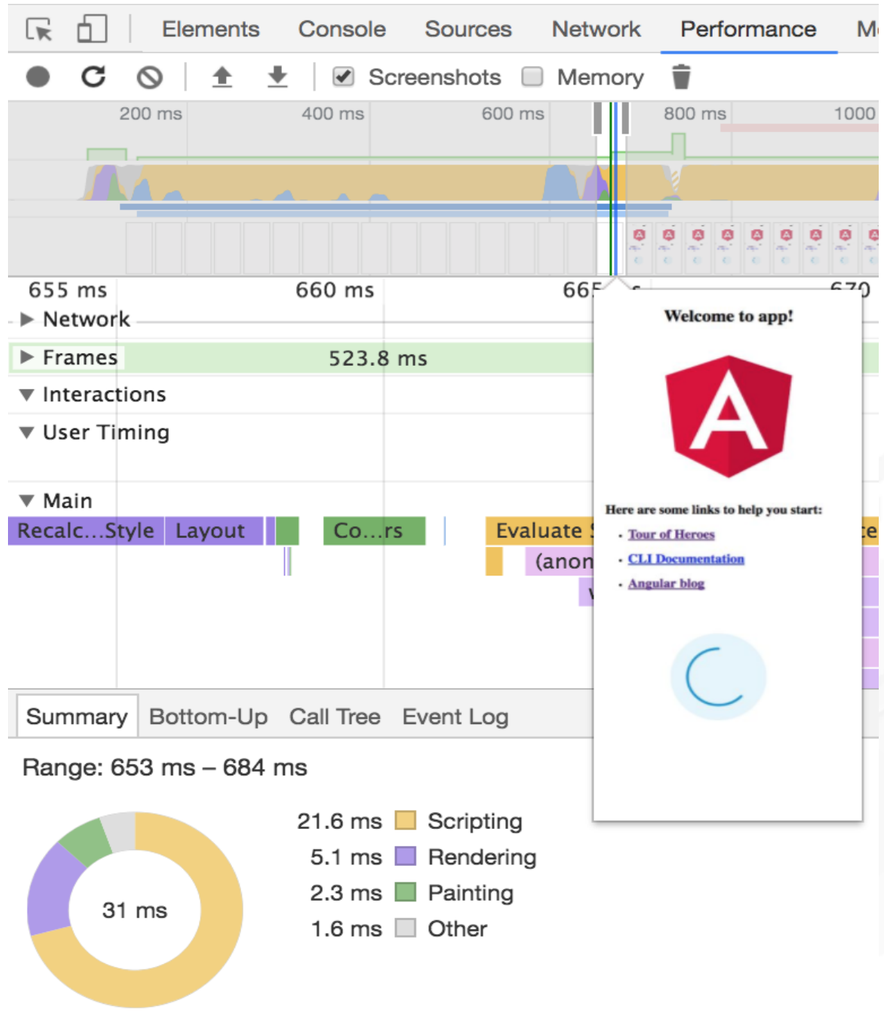


Measuring Performance of App Shell



Application without app shell

There is a significant improvement in the loading time of the application to the end user.



Application with app shell

Notification Management

Web Push Notifications

A web notification is a pop-up message that appears on the user's browser.

They are of two types: Notification API and Push API.

Allows end user to get timely updates.

One can build these on top of service worker.

Can be triggered locally by an application or can be pushed from the server to the user when app is not running.

Requires local server to test it.

Notification Terms

Push Notification
Response created for a push message

Notification API
Interface to display notification to the user

Push Message
Message sent to the client from server

Push API
Interface for a push service subscription

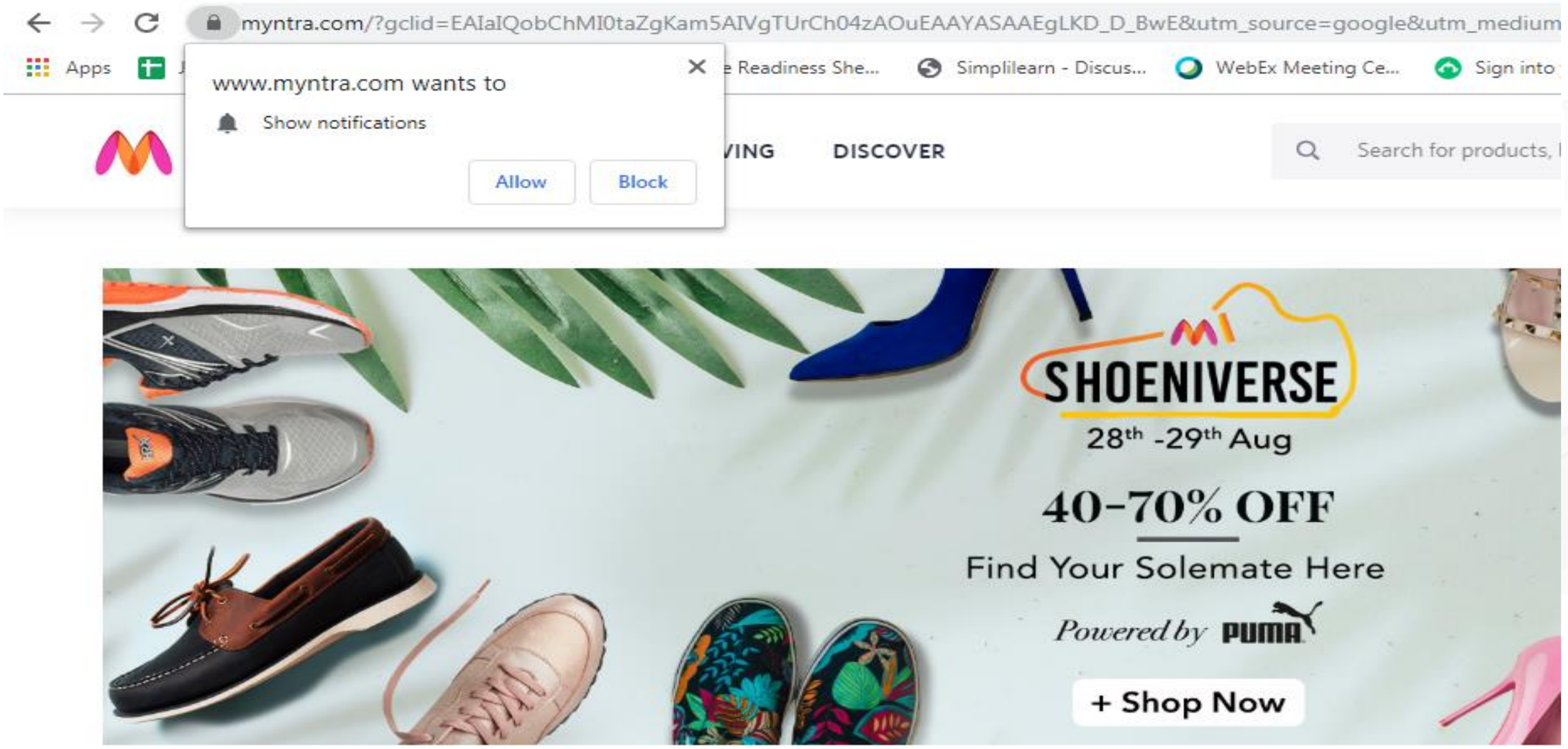
Web Push Protocol
Information on server and user interaction with push service

Web Push
Components used for pushing messages from server to client

Push Service
Mechanism for routing push message

Browser Push Notification Example

Below is a small example of how a push notification looks like:



Push Notifications Major Components

There are three major actors in the execution of push notifications.

Web Push Notification Server
It is a browser-specific in-built push server.

Service Worker Registration
It includes service worker registration.

User Subscription ID
It is generated when a user subscribes for a push notification.

Browsers Supporting for Push Notification

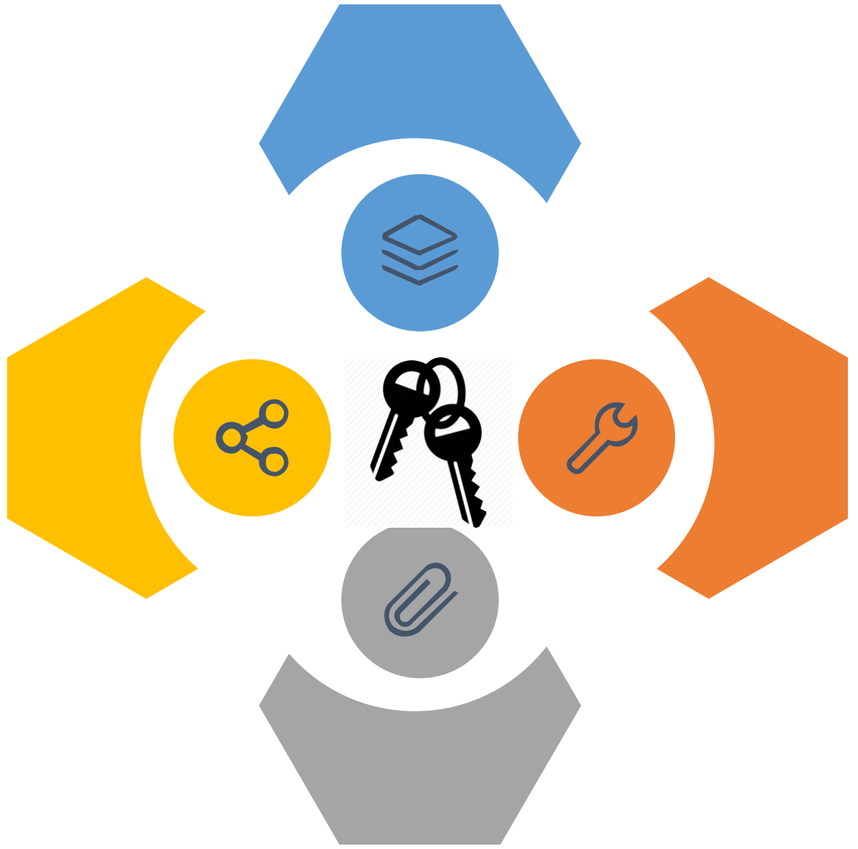
Device	OS	Browser
Mobile	Android	Chrome 50+, Firefox 44+, and Opera 46+
Mobile	iOS	Not supported
Desktop	Windows	Chrome 48+, Firefox 44+, Safari 7+, Opera 42+ and Edge 17+
Desktop	Linux	Chrome 48+, Firefox 44+, Safari 7+, and Opera 42+
Desktop	OS X	Chrome 48+, Firefox 44+, Safari 7+, and Opera 42+
Tablet	Android	Chrome 50+, Firefox 44+ and Opera 46+
Tablet	iOS	Not Supported

Source: https://www.izooto.com/web-push-notifications-explained

VAPID Key Pair Generation

VAPID stands for Voluntary Application Server Identification for push notification protocol.

It is also used to identify application server to analyze behavioral patterns of messages sent.



It is an encrypted public/private key pair.

Public key is unique for a server to register user for notification service.

Generating VAPID using Node Web Push

Steps to generate VAPID using node web push:

VAPID with Node Push

1. Install web push using CLI
2. Generate VAPID key pair with **web-push** command
3. Use VAPID key pair to subscribe to push notification

Notification Management

Duration: 60 min.

Problem Statement:
You are given a project to add web push notification in your application.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Demonstarte Notification Management

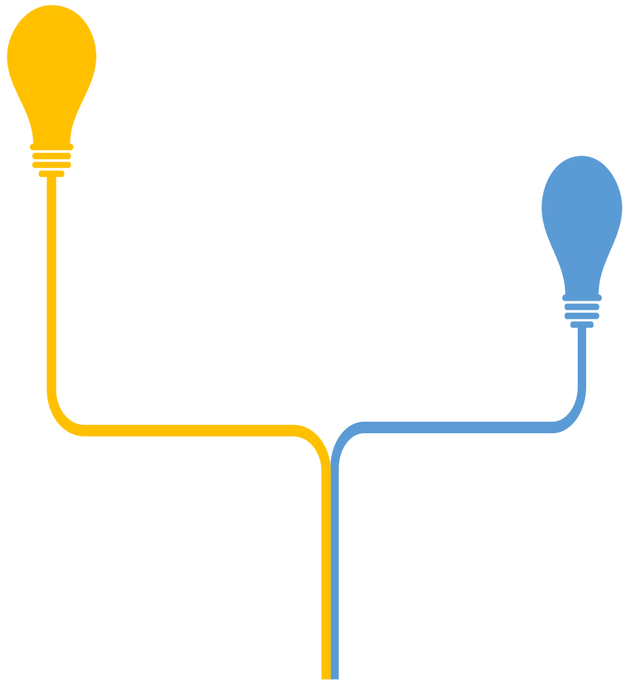
1. Create an Angular application.
2. Install PWA and HTTP server.
3. Build and run the application.
4. Add push notifications in the component.
5. Push the files to the Github repositories.

Service Worker

Service Worker Registration

Service worker can be added to Angular application at two instances:

New Angular Application
Use below command to add SW while creating a new application:
ng new angular-pwa-app --service-worker



Existing Angular Application
Use below command to add SW to the existing application:
ng add @angular/pwa --project <name of project as in angular.json>

Service Worker Registration

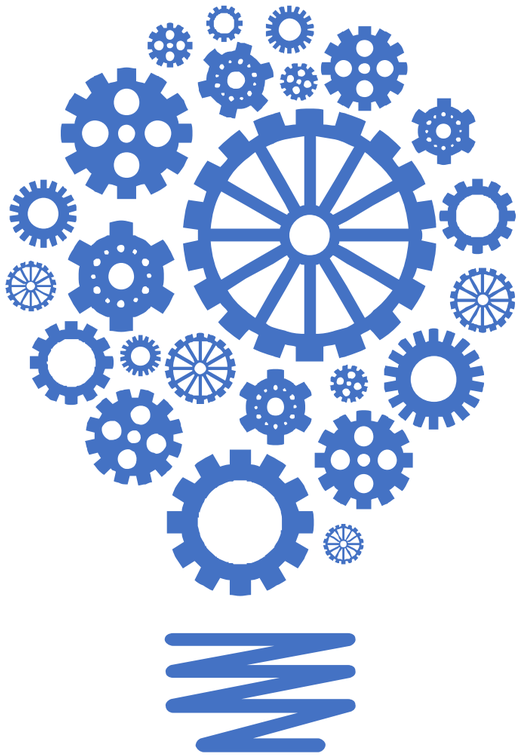
Cache Storage

Cache storage is the process where service worker download the entire angular application.

Cache is used to obtain back the cached application, store and delete objects.

CacheStorage is responsible for enabling SW to cache the application.

CacheStorage is often misunderstood as a **ServiceWorker**



Cache is stored in the form of request and response pair, where request is the key and response is the value.

Request object sends HTTP request on network and a response object is received from the server.

res and res in cache are represented as:

Cache

----

key | value

----

req | res

CacheStorage API Methods

CacheStorage API has several methods that can be used to work with various cache objects and their operations.

Deleting a Cache

Use **delete()** method to delete any existing cache object

Listing/Retrieving a Cache

keys(), match(), and matchAll() methods are used to fetch all the cache objects that are currently active

Checking a Cache

has() method is used to check the existing cache object

Creating Cache

open() method is used to create cache object

HTTP Request Interception

ServiceWorker handles caching in a web app by **penetrating into XHR requests.** It also provides flexibility to configure caching while keeping application code without caching logic. Follow the below sequence to perform it:

Create a PWA angular application

Cache static content

Cache API requests with performance

Cache API requests with freshness

Service Worker

Duration: 45 min.

Problem Statement:
You are given a project to register and monitor service worker in your browser.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Demonstrate Service Worker

1. Create an Angular application.
2. Install PWA and HTTP server.
3. Build and run the application.
4. Monitor service worker in the development tool of the browser.
5. Push the files to the GitHub repositories.

Working with Offline Page

Duration: 30 min.

Problem Statement:
You are given a project to enable offline feature in your application using service worker.

ASSISTED PRACTICE

Assisted Practice: Guidelines to Work with Offline Pages

1. Create an Angular application.
2. Install PWA and HTTP server.
3. Build and run the application.
4. Add service worker for offline access.
5. Push the files to the GitHub repositories.

Key Takeaways

- A Progressive Web App (PWA) is a website that appears and behaves like a mobile application.
- PWA is responsive and progressive. It provides offline usability.
- Service worker is a script that manages the caching of an application while running on a web browser.

Task Tracker

Duration: 45 min.

Problem Statement:

As a MEAN Stack Developer, you have to create a to-do list PWA application.

LESSON-END PROJECT

Before the Next Class

Courses: Node.js Absolute Beginners Guide-Learn Node

from Scratch Training

You should be able to:

- Install tools for a NodeJS application
- Create a basic website with Node and Bootstrap
- Push Node website to Heroku