Javascipt:

* How to insert an div element in between two divs in Javascript
* Event bubbling and how to stop propagating
* how to browser will load resouces from website
* Difference between for and foreach and what is preferred method and why //
* what is shadow Dom
* Write singleton design pattern
* ES6 features
* Closures
* Iterators
* Arrow functions and what is the use
* How browser will create DOM
* Generators
* Prototype
* Inheritance in JavaScript

Angular:

* how to style particular elements in the template
* change detection strategy
* Directives
* How to verify roles and show/hide particular menu items
* How to provide security in client side
* what is Zone.js and what it will do in angular
* View encapsulation //
* Polyfills
* how to get url params
* ActivatedRoute //
* HostListener  //
* Create feature selector in NgRx.
* What is switchMap, mergeMap, concatMap, scan, withLatestFrom //
* Difference between hot and cold observable. //
* How to rethrow errors from nested observable.
* Difference between BehaviorSubject and Subject. //
* At what point of time meta reducer is executed?
* How to dynamically create feature store object/action/reducer/effect?
* What is decorator? //
* Difference between constructor and OnInit methods. //
* Sequential 'CanActivate'.
* What is application budget and how can you improve on it with lazy loading modules with(without) Router. //

Topics

* Components
* Data Binding
* Directives
* Services and Dependency Injection
* Routing
* Observables (Hot vs Cold) //
* Forms: Template/Reactive
* Pipes(Pure and Impure) //
* HTTP
* Authentication and Route Protection
* Dynamic Components
* Angular Modules and Optimizing Angular Apps
* Angular Animations
* Service Workers
* Unit Testing
* Unit Testing (Jasmine, Karma)
* Unit test coverage (Istanbul)
* NgRx
* RXJS Operators
* View encapsulation
* Change Detection Strategy
* AOT vs JIT compilation //
* Lifecycle Hooks //
* Decorators //
* Promises vs Observables
* Weak Map, Weak Set
* ES6 concepts
* Zone.js
* SOLID Principles
* Polyfills
* Code Quality (linting, static code analyzers, Styles Guides )
* Performance Optimization
* This, let vs var vs const //
* Closures Practical Implementation
* Shallow copy vs deep copy in js
* How to use HTTP without observables //
* Component communication (Parent, child, nested) //

Questions

* Data Types in JavaScript
* Spread Operator //
* Rest Operator //
* Destructuring //
* ES6 Concepts
* Object.freeze //
* Routing in Angular //
* Lazy Loading //
* Auth Gaurds
* Promises
* Promises vs Observable //
* Generators
* Async Await //
* Change detection Strategy
* Unit Testing in Angular
* Services and Dependency Injection
* Passing Data Between Components
* Subjects and Behaivour Subjects //
* Session Storage and Local Storage
* Performance Optimizations in JavaScript
* SOLID Principles
* Design Patterns
* Redux Workflow
* Code Review Process
* Write polyfill for Array.find();
* Service Workers and Web Workers and their usage
* " Write a complete Angular component to render data from the endpoint ‘/getUsers’ which gives data
* [{id:1,name:’pradeep’},{id:2,name:’matharaju’},{id:3,name:’sridivya’},{id:4,name:’kumar’}]."
* "interface a {
* id:Number,
* name:String
* }
* Class b implements a{
  + //class code
* }
* b instanceOf a ---> output?."
* "let arr = [‘john’,’David’,[‘SriDivya’],[‘pradeep’,’Matha Raju’]]
* function findLargestString(arr){
* //implement function//
* }"
* Sets, WeakSets, Maps and Weak Maps
* How to make a particual property of an object readable?
* Hoisting in JavaScript
* Let vs Var vs Const //
* Arrow functions vs normal functions
* different ways to create objects in Javascript
* different ways to invoke functions in Javascript(call apply bind) //
* Intercomponent communication using subjects in services
* Event loop in javascript (callstack, webapi,event queue, microtask queue) //
* Promises, Promise.all
* how does inheitence work in javacript(protype based inheritence explain with example)
* routing in Angular
* Change detection Strategy in angular
* Unit Testing in Angular
* Maps and sets
* Generators
* Data types in Angular
* closures In Javascript (expalin with example)
* Difference between AngularJS vs Angular
* What do you think which is better? Difference between Javascript and Typescript (JS is functional and TS is OOP language)
* What OOPs features typescript supports?
* Why do we use Services, why don’t we use components everywhere (purpose of service)
* Which principle (from SOLID)/ design is used in Angular services
* What contribution have you made which was major and used by the team (Core utilities)
* Data sharing between components
* What is difference between asyncronous and synchrounous
* What is difference between get and post
* What is variables

AWS Topics: 22/01/2021

* AWS Lambda
* AWS Api Gateway
* AWS Cloudfront
* AWS RDS
* AWS cognito
* AWS Secrets manager
* AWS DynamoDB
* AWS S3
* AWS Cloudfront

AWS Questions: 22/01/2021

* How to host static website using AWS s3?
* How to manage passwords in AWS lambda functions?
* How to connect AWS services with client side applications?
* How to join two tables using primary & foreign key?
* What are the types of engines supports AWS RDS?
* how to use specific node version in lambda function?

**Immutability in Javascript**

- Splice, Slice

**Prototypal inheritance**

- Beyond basics, setPrototypeOf, defineProperties

**Lexical vs Dynamic scope**

- How does “this” keyword works in javascript different scenario

**Hoisting**

- Partial hoisting vs Full hoisting

**@ViewChild and @ContentChild**

- Angular - ElementRef, ViewContainer

**ChangeDetection Strategies**

- ngZone

# LSEG

## Project Discussions

Rxjs:

-- Different Rxjs operators, switchmap, mergemap, concatmap, forkjoin, ExhaustMap, Interval, and Timer Operator

-- Scenario-based questions when to use which operator, In the same scenario how different operators will behave

Javascript:

-- Generator function

-- Promises and observables

-- How async-await works under the hood

-- callback hell

Angular:

-- Authentication and Authorization in Angular apps

-- Different Auth Guards in Angular

-- Questions related to token management, Scenario-based questions, where will you store the tokens, what if token gets expired, what will happen if we modify the token

-- What is refresh token, How will you refresh the token

-- Local storage, Session Storage, Cookies

-- Reactive forms

HTML CSS:

-- CSS Preprocessors LESS and SASS

-- What will you do if your CSS does not work

-- Semantic HTML

Unit Testing :

Basic Questions related to Unit testing, mocking, stubbing

E2E:

Basic questions related to Protractor

CI-CD:

How CI-CD works How you have implemented CI-CD in your project, Jenkins Pipeline

# Equifax

1. What is dependecy injection in angular?  
   2. what will you do if you see white page  on application load, when client hits URL.  
   3. How do you do code reviews and PR  
   4. What are CI CD pipelines

* How to optimize application for better performance
* Difference between ajax and rest
* What is the main usage of ajax
* Any challenges in your whole experience and how did you solve them.
* Why do you consider particular technology/framework is the most suitable for your challenge( for example, In my case, I said, I have used angular material to solve my challenge. So they asked about angular).
* Was there any ambiguity in getting requirements.
* How did you communicated requirements  ambiguity to product owner for getting proper requirements.
* What are rules you fallow while writing code/developing application
* In general, how you make application user friendly to engage the user more time.
* What is the difference between http post and put
* What is your contribution on particular project and answer dependent questions.
* How do you showcase unit test reports to  product owner. Either through, adding at story or any other way.