**JavaScript Developer Answer (B) Time 1 Hour**

1. Which of the below component Lifecycle hook is executed first?

c)ngOnChnages

1. How to pass data from parent to child ?

d)@Input()

1. What is the value of "1"+2+4

c) 124

1. What is the value of **(-'34'+10)**

c) -24

1. If you have var y = 1, x = y = typeof x; What is the value of x?
2. Undefined
3. ‘map’ keyword which part of library ...?

d)rxjs

7. Can we have nested controllers in AngularJS?

1. Yes
2. What is Associative Array? How do we use it?.

An associative array is declared or dynamically created

var arr = { "one": 1, "two": 2, "three": 3 };

var o = new Object();

o["one"] = 1;

o["two"] = 2;

o["three"] = 3;

for(var i in o)

{

document.write(i + "=" + o[i] + '<br>');

}

1. How to achieve inheritance in JavaScript with the help of a example?

JavaScript Prototypes. All JavaScript objects inherit the properties and methods from their prototype. Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype. The Object.prototype is on the top of the prototype chain.

- function Person(){}

Person.prototype.sayHello = function(){

alert ('I am function of person');

}

function Male(){}

Male.prototype = new Person();

Male.prototype.constructor = Male;

objectOfMale = new Male();

objectOfMale.sayHello();

(or)

var Person = {

sayHello : function () {

alert('Person object');

}

};

var objectOfMale = Object.create(Person);

objectOfMale.sayHello = function () {

alert('hello I am a male');

};

objectOfMale = new Male();

objectOfMale.sayHello();

1. Explain one way data binding, two way data binding and pipes in Angular? Also write a example of each.

Use of ngModel as is return with input tag :- <input type="text" [(ngModel)]="name">

- Use of {} as is in the view part :- {variable} like printing data fetched from server of static

- Use of Pipe like as :- <p>My name is <strong>{{ variable\_name | user\_define\_pipes }}</strong>.</p> and developer can create custome PIPES.

1. Explain Template Driven vs Model Driven or Reactive Forms?

- here should be use [ngModel], without ngmodel we can implement template driven.

<form #f="ngForm" (ngSubmit)="onSubmitTemplateBased()">

<p>

<label>First Name:</label>

<input type="text"

[(ngModel)]="user.firstName" required>

</p>

<p>

<label>Password:</label>

<input type="password"

[(ngModel)]="user.password" required>

</p>

<p>

<button type="submit" [disabled]="!f.valid">Submit</button>

</p>

</form>

- here should be import some require form properties - "FormGroup, FormControl, Validators, ..."

import { FormGroup, FormControl, Validators, FormBuilder }

from '@angular/forms';

@Component({

selector: "model-driven-form",

templateUrl: 'model-driven-form.html'

})

export class ModelDrivenForm {

form: FormGroup;

firstName = new FormControl("", Validators.required);

constructor(fb: FormBuilder) {

this.form = fb.group({

"firstName": this.firstName,

"password":["", Validators.required]

});

}

onSubmitModelBased() {

console.log("model-based form submitted");

console.log(this.form);

}

}

1. Explain $q service, deferred and promises .

‘Promises’ are post processing logics which are executed after some operation/action is completed whereas ‘deferred’ is used to control how and when those promise logics will execute.

We can think about promises as “WHAT” we want to fire after an operation is completed while deferred controls “WHEN” and “HOW” those promises will execute.

“$q” is the angular service which provides promises and deferred functionality.

1. What is Angular 2 Async Pipe?

Angular2 provides us special kinds of pipe that is called Async pipe and the Async pipe subscribes to an Observable or Promise and returns the latest value it has emitted.

The Async pipe allows us to bind our templates directly to values that arrive asynchronously manner and this is the great ability for the promises and observables.

this.userService.getUsers(this.userRestApiUrl).subscribe(data => this.users = data);

<ul> < li \* ngFor="let user of users | async"> Id: {{user.id }}, Name: {{user.name }} </li>< /ul>

1. Is it possible to have a multiple router-outlet in the same template? With brief and example.

is it possible to have a multiple router-outlet in the same template?

- Yes! We can use multiple router-outlets in same template by configuring our routers and simply add the router-outlet name. You can see in the example.

<div class="row">

<div class="user">

<router-outlet name="userList"></router-outlet>

</div>

<div class="userInfo">

<router-outlet name="userInfo"></router-outlet>

</div>

</div>

const routes: Routes = [];

1. What is optional Parameters , Rest Parameters, arrow functions in TypeScript? With examples.

We can specify optional properties on interfaces and the property may be present or missing in the object.

interface User {

name: string;

age: number;

address?: string //Optional

}

let userInfo = function(user: User) {

let info = "Hello, " + user.name + " Your Age is - " + user.age + " and Address is -" + user.address;

return info;

}

let info = {

name: "Anil",

age: 30

};

console.log(userInfo(info));

- The Rest parameters do not restrict the number of values that we can pass to a function and the passed values must be the same type otherwise throw the error.

let addNumbers = function(...nums: number[]) {

let p;

let sum: number = 0;

for (p = 0; p < nums.length; p++) {

sum = sum + nums[p];

}

return sum;

}

//The Result

addNumbers(1, 2);

addNumbers(1, 2, 3);

addNumbers(1, 12, 10, 18, 17);

- The arrow function is additional feature in typescript and it is also known as a lambda function.

var addNum = (n1: number, n2: number) => n1 + n2;

1. Describe closure in JavaScript ? with example

While you create the JavaScript function within another function and the inner function freely access all the variable of outer function

How to create closure in JavaScript?

function ourterFun(i) {

var var1 = 3;

function innerFun(j) {

console.log(i + j + (++var1)); // It will return the 16.

}

innerFun(10);

}

ourterFun(2); // Pass an argument 2

1. Define routeProvider and stateProvider ?

routeProvider

HTML

<div ng-view></div>

Above tag will render the template from the $routeProvider.when() condition which you had mentioned in .config (configuration phase) of angular

1. The page can only contain single ng-view on page

2. If your SPA has multiple small components on the page that you wanted to render based on some conditions, $routeProvider fails. (to achieve that, we need to 3. use directives like ng-include, ng-switch, ng-if, ng-show, which looks bad to have them in SPA)

4. You can not relate between two routes like parent and child relationship.

5. You cannot show and hide a part of the view based on url pattern.

stateProvider

---------------------------

HTML

<div ui-view>

<div ui-view='header'></div>

<div ui-view='content'></div>

<div ui-view='footer'></div>

</div>

1 You can have multiple ui-view on single page

2 Various views can be nested in each other and maintained by defining state in routing phase.

3 We can have child & parent relationship here, simply like inheritance in state, also you could define sibling states.

4 You could change the ui-view="some" of state just by using absolute routing using @ with state name.

5 Another way you could do relative routing is by using only @ to change ui-view="some". This will replace the ui-view rather than checking if it is nested or not.

6 Here you could use ui-sref to create a href URL dynamically on the basis of URL mentioned in a state, also you could give a state params in the json format.

1. Describe DOM and BOM ?

The BOM (Browser Object Model) consists of the objects navigator, history, screen, location and document which are children of window. In the document node is the DOM (Document Object Model), the document object model, which represents the contents of the page. You can manipulate it using javascript.

1. Differences b/w Constructor and ngOnInit ?

1. The constructor is a default method runs when component is being constructed.

2. The constructor is a typescript feature and it is used only for a class instantiations and nothing to do with Angular 2.

3. The constructor called first time before the ngOnInit().

-

1. The ngOnInit event is an Angular 2 life-cycle event method that is called after the first ngOnChanges and the ngOnInit method is use to parameters defined with @Input otherwise the constructor is OK.

2. The ngOnInit is called after the constructor and ngOnInit is called after the first ngOnChanges.

3. The ngOnChanges is called when an input or output binding value changes.

1. What is Lazy Loading and How to enable Lazy Loading?  
   Lazy Loading - Lazy loading enables us to load only the module user is interacting and keep the rest to be loaded at run-time on demand.

Lazy loading speeds up the application initial load time by splitting the code into multiple bundles and loading them on demand.

1. Each and every Angular2 application must have one main module that is called “AppModule” and your code should be splitted into various child modules based on your applications.

2. We do not require to import or declare lazily loading module in root module.

3. Add the route to top level routing and takes routes array and configures the router.

4. Import module specific routing in the child module.

5. And so on.

1. Describe the use of @NgModule ?

The @NgModule metadata imports a single helper module, BrowserModule, which every browser app must import. BrowserModule registers critical application service providers. It also includes common directives like NgIf and NgFor, which become immediately visible and usable in any of this NgModule's component templates.