**Client Side Assessment Answers**

1. **Closures:**

a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

JavaScript closure example

function Outer\_function() {

// variable defined outside the inner function

let name = 'Govardhan';

// inner function

function Inner\_Function() {

// accessing name variable

return console.log( 'Hi' + ' ' + name);

}

return Inner\_Function;

}

const test = Outer\_function();

test // returns the function definition

test() // returns the value

1. **Shallow Copy:**

* In Shallow copy, a copy of the original object is stored and only the reference address is finally copied.
* The changes made in the copied object also reflect the original object.
* It stores references of the object in the main memory.

Example:

let person *=* {

  firstName: 'Govardhan Reddy',

  lastName: 'Morsu',

};

let bperson *=* person;

bperson.firstName *=* 'Ravindar';

console.log(bperson);     *// { firstName: 'Ravindar', lastName: 'Morsu' }*

**Deep Copy:**

* In Deep copy, the copy of the original object and the repetitive copies both are stored.
* There is no reflection on the original object when the changes are made in the copied object.
* It stores copies of the object values.

Example:

let person *=* {

  firstName: 'Govardhan Reddy',

  lastName: 'Morsu ',

};

let bPerson *=* JSON.parse(JSON.stringify(person));

bPerson.firstName *=* 'Ravindar';

bPerson.lastName *=* 'Reddy';

console.log(person);      *//{ firstName: 'Govardhan Reddy', lastName: 'Morsu ' }*

console.log(bPerson);     *//{ firstName: 'Ravindar', lastName: 'Reddy' }*

1. **Object Destructuring:**

The object destructuring is a useful JavaScript feature to extract properties from objects and bind them to variables.

* object destructuring can extract multiple properties in one statement.
* It can access properties from nested objects, and can set a default value if the property doesn't exist.

Example :

const hero *=* {

  name: 'Iron Man',

  realName: 'Tony Stark'

};

var name     *=* hero.name;

var realName *=* hero.realName;

 console.log(name,  realName);   *//classical way*

const { name, realName } *=* hero;

console.log(name,  realName);   *//object destructuring way*

1. **Arrow Function:**

We have to write less code with the Arrow Function while providing the same functionality, which can help to declutter and simplify your code.

* It reduces a lot of code and makes it more readable.
* The greatest advantage of having contextual “this”, no longer need to “bind” functions any more.
* Most modern browsers support Arrow Functions out of the box. Although I will still advice using a transpiler like Bable to polyfill for backward compatibility.

Example:

*//Arrow Function*

let squareA *=* (a) => a *\** a

console.log(squareA(2))   *// Output is 4*

*// Clasical Function*

function squareB(x){

*return* x*\**x

}

console.log(squareB(2))   *// Output is 4*

1. **Higher Order Function:**

In JavaScript, functions are treated as first-class citizens. We can treat functions as values and assign them to another variable, pass them as arguments to another function, or even return them from another function.

* Basically, a function which takes another function as an argument or returns a function is known as a higher order function.

Example:

1. **Event Looping:**

JavaScript is a single-threaded synchronous programming language. It means that the main thread where JavaScript code is run, runs in one line at a time manner and there is no possibility of running code in parallel.

Event loop: An event loop is something that pulls stuff out of the queue and places it onto the function execution stack whenever the function stack becomes empty.

The event loop is the secret by which JavaScript gives us an illusion of being multithreaded even though it is single-threaded

1. **Building Blocks of Angular:**

* Modules
* Components
* Templates
* Metadata
* Data binding
* Directives
* Services
* Dependency injection
* Modules:

A module is a container for a group of related components. There are two types of modules one is encapsulating block of function within the single component and the other is encapsulating block of function within a single or group of components by providing exposure in an interface. we need to divide our app module into sub smaller modules and each module is responsible for a specific section.

* Components:

Its component represents a unique "View" and "View Model" in **Model-View-ViewModel** (MVVM) pattern or exactly like what components do in Angular. The "View" or Template shows how the complete component will look–when displayed on the browser. "View Model" has all required logic parts to provide "View" with rich functionality and data.

To generate component we use

ng generate component <component name>

it will create html, css,ts,spec.ts files in the component directory.

* Templates:

The templates' view represents a view that improves HTML with Angular functionality like data binding. In the template to encapsulate the data and when the data changes here at runtime, Angular automatically updates it in the browser view as well.

* MetaData:

Metadata helps in connecting everything in the applications, a "View" with a "View Model" with styles.

* + - * + the selector is Metadata in Angular the custom HTML to be included with the component.
        + template URL is the exclusive URL for the template to be used when processing the component,
        + an array of directives with this kind of Metadata will tell Angular which other Component or the directives should be included in that component.
        + an array of style URLs with this type of Metadata you can easily define customized styles for the component.
        + Angular how to inject dependencies like services in a component or @RouterConfig that helps you to configure routing in your project application.
    - DataBinding:

Data binding is a technique, where the data stays in sync between the component and the view.

There are two different types of data bindings

* + - * + one-way
        + two-way binding.
* Directives:

There are three types of Directives present in Angular

* + - * + **Components Directive** The component can be used as a directive. Every component has **an**and Output option to pass between the component and its parent HTML elements.
        + **Structural Directive** is like \*ngFor and \*ngIf which enables you to make changes to DOM with the help of adding or by the Input moving nodes.
        + **Attribute Directive** helps in adding behavior or do a change in the look or appearance of a specific element just like ngmodel directive which implements two-way data binding is an Attribute Directive.
* Services:

Services simply help in making reuse of service. As the project becomes bigger naturally more components will add to it and which also will require data to access. So, every time making a copy-paste of code it will create a single reusable singleton data service.

* Dependency injection

Dependency Injection to inject the necessary dependencies for a given component/service at runtime and provide flexibility, extensibility, and testability to the framework and your application.

We need to explicitly instruct the Angular to create an instance of Service and passes to our AppComponent. This concept is called Dependency Injection. So, we should instruct Angular to inject the dependency of this component into its constructor.

1. **Advantages of Angular in Web Application:**

### **1. Automatic Synchronization With Two-Way Data Binding**

The two-way data binding enables the synchronization of data between the Model and the View. As a result, these two components are automatically updated when data is altered or amended. Furthermore, this occurs in real-time, avoiding engineers from devoting additional time and effort to manual changes.

### **2. Optimized Server Communication**

With Angular, caching becomes a breeze! By producing static files, Angular eliminates the extra load on CPUs. Furthermore, the framework’s response time to API calls is quite fast. In the case of an Angular application, the pages in the document object model are rendered for user actions. It provides a rapid user experience and does not need users to wait for the app to become fully engaged.

### **3. POJO Model to Eliminate External Dependencies**

The Plain Old JavaScript Objects (POJO) Model is used by Angular to make the code structure portable and independent. This enables us to avoid introducing complicated functions or methods to the programme. It also eliminates the need for third-party frameworks or plugins. Because POJO needs less coding, apps produced with Angular load quickly and give outstanding user accessibility. To add to this, the model allows us to keep the scripts clean, making the framework goal-oriented.

### **4. Testing in Angular – An Integral Part**

The Angular framework includes testing as a standard feature. In Angular, all JavaScript code is subjected to a set of tests. The ease of testing allows us to start from scratch and test all of the components at the same time. Thanks to Angular’s dependency injection which is in charge of all scopes and controllers. Because of this requirement, Angular’s unit test feature may force dependency injection by injecting mock/dummy data into the controller to do the testing.

### **5. Angular Material – A Comprehensive Modern UI**

Angular Materials follow Google’s Material Design standards. It comes with navigation elements, layout, buttons, indicators, and pre-installed data tables. Furthermore, Angular Material not only creates a pleasing user experience but also generates apps such as Google. Nevertheless, apps such as Google Drive, Android OS, Gmail, and others have become everyday necessities for consumers. One of the best benefits of Angular.

### **6. Angular & Single Page Applications**

A single page application’s seamless operation is dependent on two factors. One is that when the website loads, all of the required JavaScript, CSS, and HTML codes are obtained at the same time. The second is that they are retrieved as and when needed, based on the user’s actions and behaviour. While users are surfing the app, there is no page reloading, which necessitates dynamic contact with servers.

### **7. Code Consistency and Robust Ecosystem**

The Angular CLI tool allows developers to establish basic projects, perform tests, and add new functionalities in a single project while keeping the entire team informed. Angular’s style guide is frequently appreciated by the developers. This makes team communication more comfortable and is less likely to be misconstrued. Thanks to the framework’s extensive ecosystem, which has generously provided the developer community with the power of dependency injection and other resources.

### **8. Model-View-ViewModel (MMVC) Architecture to Unify Independent Structures**

Developers are more confident in building a clean user interface with seamless business logic as the framework separates business logic from UI components. This is because of the controller’s rapid communication between the model and the view component. As a result, data is shown as rapidly as feasible.

### **9. Ivy Renderer – Next-Generation Renderer for Enhanced Performance**

Ivy renderer strives for exceptional code debugging and a user-friendly app experience. It makes the framework more accessible and sets a good example by lowering file sizes while still maintaining the framework as a feature-rich platform. It wasn’t in its final shape when it was unveiled at NG-CONF 2018, however Angular 8 presently offers the best of it. With the introduction of Angular 9, the contemporary Angular engine will be in its full form.

### **10. Active Community & Easy-to-Access Information**

Because of its numerous advantages, Angular has been popular among developers and engineers since its inception. Angular is simple to incorporate for those who want to get the most out of it, with a wealth of resources such as study materials, reference articles, FAQs, and much more.

1. **Single Page Application:**

The major difference between multi-page and single-page applications is that single-page apps download the page a single time once it is executed.

As soon as a user interacts with the app, only the required component will be modified, not the complete application, which makes a single-page app much faster in terms of interactivity. Single-page applications offer a much better user experience (UX), meaningthat users can navigate easily between the different pages of an app without waiting for the pages to load.

Advantages:

An important feature of single-page applications is performance. They get a performance boost by loading HTML, CSS, and JavaScript resources as soon as the website is loaded.

* Battery reusability
* Optimization
* Client-side rendering
* User experience
* Easy debugging
* Performance
* Less complex implementation
* Better caching
* Better SEO optimization

1. **DataBinding:**

Data binding is a process that creates a connection between the application’s UI and the data. When the data changes its value, the UI elements that are bound to the data will also change.

Types:

**One way data binding** is a change in the state affects the view from component to view template or change in the view affects the state from view template to component.

**Two-way data binding** is a change from the view can also change the model and similarly changes in the model can also change in the view from component to view template, and also from view template to the component.

There are several ways to bind data between components and view in Angular.

* Interpolation
* Property binding
* Class binding
* Style binding
* Attribute binding
* Event binding
* Two-way binding