

Speaking about Experience

1. AWS Lambda experience

As an AWS cloud engineer with 4+ years practical experience, I have used Lambda service on the various purposes of my previous projects.

Mostly, I built server less API endpoint for my web & mobile apps using Lambda (Node.js, Python or .NET Core) with API Gateway.

I am very familiar with its core concepts, advanced usages including concurrency management, throttling, lambda layer, efficient use of memory/cpu resource, diagnosing metrics, etc.

Also, extensively used it in connection with other services like SQS/SNS and Kinesis for data streaming.

2. DynamoDB/Cognito experience

DynamoDB: With its fast, scalable and NoSQL based features, I often used DynamoDB in my projects to support the data storage of WIP routines (temporary data store). I am quite familiar with creating/managing tables and indexes, running queries on various scenarios and also using it as a trigger of lambda functions.

Cognito: Used cognito user pool for user authentication / management and identity pool for authorization which means restricting access to various resources (like S3, DynamoDB, etc).

3. Serverless architecture

Combining the above mentioned services and others like RDS, SQS/SNS, API Gateway, Step functions, etc, I built several serverless architecture of large scale web / mobile platforms on AWS cloud.

Last year, I worked as a back-end developer/cloud engineer on a construction bid retriever service which runs in serverless architecture with extensive usage of lambda functions, dynamodb and simple queue service.

This platform serves as a SaaS product and in order to process unpredictable and highly scalable amount of requests in an efficient and reliable way, introduced the serverless architecture and orchestrated on the AWS cloud.

4. GoLang experience

In the past, I have been committed to some projects as a backend developer. At that time, I have developed microservice APIs using GoLang.

Briefly, I developed microservices to simplify the process of job advertisement on google ads and also bing ads.

And also I created internal GoLang modules to help eliminate boilerplate code and enhance developer productivity.

Wrote unit tests & developed integration tests using docker-compose.

That's my experience in GoLang.

5. React Experience

1. Designed CSS templates for use in all pages on the website working with CSS background, positioning, text, border, margin, padding, and table
2. Applied optimization techniques to reduce page size and load times to enhance user experience using sprites
3. Developed user interfaces by using ReactJS, Flux for SPA development
4. Used React-Router to turn the application into Single Page Application
5. Worked in using ReactJS components, Forms, Events, Keys, Router, Animations, and Flux concept.
6. Used Web services (SOAP and RESTful) for transmission of large blocks of XML.JSON
7. Implemented the Drag & Drop functionality using React-Draggable
8. Used React-Autocomplete for creating google maps location search on the webpages.
9. Used OOP concepts to develop UI components that could be reused across the Web Application
10. Used JIRA as the bug tracking system to track and maintain the history of bugs/issues on an everyday basis
11. Involved designing in web pages using HTML5, CSS3, JS, Bootstrap, SASS, LESS, React, Redux, Flex, MongoDB
12. Worked on ReactJS Virtual Dom and React views, rendering using components that contain additional components called custom HTML tags.
13. Implemented stable React components and stand-alone functions to be added to any future pages
14. Used ReactJS for templating for faster compilation and developing reusable components
15. Involved in development of User Interface using HTML5, CSS3, JS, jQuery, Angular
16. Extensively used Angular UI(Angular Bootstrap) for ng-grid, and UI components
17. Extensive experience using Bootstrap for layout
18. Followed SCRUM methodology
19. Produced content pages with CSS3 layout and style markup presentations and also used JS methods and properties
20. Updated the website from time to time for special Requirements

5. .Net + C#

- React + .NET experience

React and .NET are powerful technologies that can be used to create highly dynamic and scalable e-commerce shopping systems.

1. Front-end development with React:

React is a JavaScript library used for building user interfaces. In an e-commerce shopping system, React can be used to create a highly interactive and responsive shopping experience for users. I have experience using React to create components such as product listings, shopping carts, and checkout forms.

2. Back-end development with .NET:

.NET is a framework for building web applications, including e-commerce shopping systems. I have experience using .NET to develop the server-side logic for processing user requests and managing data. This includes creating APIs for handling user authentication, product search, and order processing.

3. Integration of React and .NET:

One of the key benefits of using React and .NET together is the ability to create a seamless user experience by integrating the front-end and back-end components of the shopping system. I have experience using tools such as Webpack and Babel to compile React components and integrate them with .NET APIs.

4. Testing and debugging:

As with any complex software project, testing and debugging are critical components of ensuring a high-quality e-commerce shopping system. I have experience using tools such as Jest and Enzyme for testing React components, as well as Visual Studio and Postman for testing .NET APIs.

Overall, my experience with React and .NET in e-commerce shopping systems has allowed me to create dynamic and scalable systems that provide users with a seamless shopping experience.

- http status 200, 300

HTTP status codes are used by servers to communicate the status of an HTTP request to the client.

HTTP status code 200 - OK: This status code indicates that the server has successfully processed the request and the response contains the requested resource. This is the most common HTTP status code and indicates that everything went well with the request. When a client receives a 200 response, it typically means that the request was successful and that the server has responded with the requested data.

HTTP status code 300 - Multiple Choices: This status code indicates that the requested resource has multiple representations available, each with its own URI, and the client must choose which representation to use. For example, a server may provide multiple language options for a web page,

and the client can choose which language to display. The response typically includes a list of the available representations and the URIs for each representation.

Other status codes in the 3xx range include:

301 Moved Permanently: The requested resource has been permanently moved to a new location, and the client should update its bookmarks or links to the new URL.

302 Found: The requested resource has been temporarily moved to a new location, and the client should use the new URL for the current request only.

304 Not Modified: The client's cached copy of the requested resource is still valid, and the server has not modified the resource since the last request.

307 Temporary Redirect: The requested resource has been temporarily moved to a new location, and the client should use the new URL for all subsequent requests.

In summary, HTTP status code 200 indicates a successful response, and HTTP status code 300 indicates a redirection response with multiple choices available.

- http 400 vs 500

HTTP status codes are used by servers to communicate the status of the HTTP request to the client. The two status codes that you have mentioned, 400 and 500, have different meanings and indicate different types of errors:

HTTP status code 400 - Bad Request: This status code indicates that the server cannot process the request due to a client-side error, such as invalid syntax or missing parameters in the request. The client should modify the request and try again. Some common causes of a 400 error include malformed JSON requests or missing required request parameters.

HTTP status code 500 - Internal Server Error: This status code indicates that the server encountered an unexpected error while processing the request. The error could be due to a programming error or a problem with the server's configuration. This error is typically not the client's fault, and the client does not need to take any action other than to notify the server administrator of the issue. Some common causes of a 500 error include database connectivity issues, server-side programming errors, and insufficient server resources.

In summary, a 400 error indicates a client-side error that can be corrected by modifying the request, while a 500 error indicates a server-side error that requires the attention of the server administrator.

- CORS error

CORS (Cross-Origin Resource Sharing) is a security mechanism implemented by web browsers to restrict access to resources on a web page from a different domain than the one that served the page. It is a common error that occurs when a web application tries to access a resource from a different domain or origin, and the server that hosts the resource doesn't allow the request.

When a web application running in a web browser tries to access a resource (e.g., an API endpoint or an image) on a different domain than the one that served the web page, the web browser sends a preflight request to the server to check if the request is allowed. The server responds with a set of headers that indicate whether the request is allowed or not. If the server does not allow the request, the browser will throw a CORS error, preventing the web application from accessing the resource.

The CORS error typically includes an error message that provides more information about the cause of the error, such as:

- Access-Control-Allow-Origin header missing: The server is not configured to allow requests from the domain that the web application is hosted on.
- Invalid CORS request: The request violates one or more of the CORS security policies, such as making a request with invalid headers or methods.
- CORS request blocked: The server blocked the request due to security concerns, such as the request being sent from a malicious website or the resource being protected by authentication.

To resolve a CORS error, the server administrator must configure the server to allow requests from the domain that the web application is hosted on. This can be done by setting the appropriate headers on the server response. Alternatively, the web application developer can use a proxy server to bypass the CORS restriction and access the resource from the server-side instead of the client-side.

- reflection in C#

In C#, reflection is a powerful feature that allows you to inspect and manipulate metadata information about types, classes, methods, properties, and other constructs within your code at runtime.

Reflection provides a way to examine and analyze the structure and behavior of objects at runtime, which enables developers to write dynamic code that can adapt to different situations.

Using reflection, you can dynamically create new objects, invoke methods on existing objects, and retrieve information about the properties and methods of a given object, even if you don't know the exact type of the object at compile-time.

Reflection is commonly used in C# to build frameworks, libraries, and other reusable components, where you want to create code that is flexible and can be used in a variety of scenarios.

For example, you can use reflection to:

Create instances of classes dynamically

Invoke methods dynamically

Get information about the methods, properties, and fields of an object

Load and inspect assemblies at runtime

Create dynamic types and objects

Reflection is a powerful tool that can help you write more flexible and reusable code, but it comes with some overhead and complexity, so it's important to use it judiciously and with care.

- What is .NET core 2.2?

.NET Core is a free, open-source, and cross-platform framework that is designed for developing modern applications. It is a successor to the .NET Framework and is actively developed by Microsoft.

.NET Core provides a lightweight and modular platform for building applications that can run on multiple operating systems, including Windows, Linux, and macOS. It supports a variety of programming languages, including C#, F#, and Visual Basic.

Some of the key features of .NET Core include:

High performance: .NET Core is optimized for performance and can deliver faster execution times than the .NET Framework.

Cross-platform support: .NET Core can run on multiple operating systems, allowing developers to build and deploy applications on different platforms.

Modular architecture: .NET Core is designed with a modular architecture that allows developers to use only the components they need, reducing the size of the application and improving performance.

Open-source: .NET Core is an open-source project, which means that the source code is available for anyone to view, modify, and contribute to.

Cloud-ready: .NET Core includes features that make it well-suited for building cloud-native applications, including support for microservices, Docker containers, and Kubernetes.

Overall, .NET Core is a modern and versatile framework that provides developers with the tools and features they need to build high-performance, cross-platform applications.

- What is ASP.NET

ASP.NET is a web development framework developed by Microsoft, used for building dynamic web applications and web services. ASP.NET allows developers to build web applications using C# or VB.NET on top of the .NET framework.

ASP.NET provides a rich set of features that make it a popular choice for building web applications, including:

Server-side programming: ASP.NET allows developers to write code that executes on the server, enabling dynamic web applications with rich functionality.

Web Forms: ASP.NET includes a powerful UI framework called Web Forms that provides drag-and-drop controls and easy data binding.

MVC: ASP.NET also includes the Model-View-Controller (MVC) architectural pattern, which provides a structured approach to building web applications.

Authentication and Authorization: ASP.NET provides a built-in security system for handling user authentication and authorization.

Caching: ASP.NET provides caching features that allow web applications to improve performance by caching frequently accessed data.

Cross-platform support: With the introduction of .NET Core, ASP.NET has become cross-platform, which means that developers can build and run ASP.NET applications on different platforms, including Windows, Linux, and macOS.

Overall, ASP.NET is a robust and powerful framework for building web applications that provides developers with a wide range of features and tools to create dynamic and responsive applications.

- Abstract class vs interface in C#

In C#, both abstract classes and interfaces provide a way to define a set of methods, properties, and events that can be implemented by derived classes. However, there are some key differences between the two:

1. **Implementation:** An abstract class can contain both abstract and non-abstract methods, whereas an interface can only contain method signatures without implementation.
2. **Inheritance:** A class can only inherit from one abstract class, but it can implement multiple interfaces.
3. **Access modifiers:** Abstract classes can have access modifiers for their members (e.g., public, protected, private), whereas interfaces can only have public members.
4. **Constructors:** An abstract class can have constructors, but an interface cannot.
5. **Default implementation:** Starting from C# 8.0, interfaces can include default implementations of methods using the default keyword. In contrast, an abstract class can provide a default implementation for a method, but it is not a part of the interface definition.

When to use an abstract class or an interface depends on the situation. Use an abstract class when you want to provide a default implementation and share code among derived classes. Use an interface when you want to define a contract for a set of methods that can be implemented by different classes, regardless of their inheritance hierarchy.

- When we use abstract class and interface in C#?

In C#, you can use abstract classes and interfaces when you want to define a set of methods, properties, and events that can be implemented by derived classes. However, the choice between using an abstract class and an interface depends on the situation. Here are some scenarios where you might use each one:

Abstract Class:

When you want to provide a default implementation for some methods, while leaving other methods to be implemented by derived classes.

When you want to share code among related classes in an inheritance hierarchy.

When you want to define a base class that can be used as a template for derived classes.

Interface:

When you want to define a contract for a set of methods that can be implemented by different classes, regardless of their inheritance hierarchy.

When you want to provide a common API for unrelated classes.

When you want to allow multiple inheritance of behavior.

It's worth noting that in C#, a class can only inherit from one abstract class, but it can implement multiple interfaces. This means that if you need to define a set of methods that can be implemented by different classes that have no relationship to each other, using an interface might be a better choice. However, if you want to define a set of methods that are closely related to each other and share some common implementation, using an abstract class might be a better choice.

- Delegates in C#

In C#, a delegate is a type that represents a reference to a method with a specific signature. A delegate can be used to encapsulate a method call and pass it around as a first-class object, just like a variable. When a delegate is invoked, it invokes the method that it references.

Here's an example of defining a delegate type and using it to reference a method:

csharp

Copy code

```
// Define a delegate type that takes two integers and returns an integer
```

```
delegate int MyDelegate(int x, int y);
```

```
// Define a method that matches the delegate signature
```

```
int Add(int x, int y)
```

```
{  
    return x + y;  
}
```

```
// Create a delegate instance that references the Add method
```

```
MyDelegate myDelegate = Add;
```



```
// Invoke the delegate, which will call the Add method
```

```
int result = myDelegate(1, 2);
```

Delegates are commonly used for callback methods, event handling, and asynchronous programming. For example, when you register an event handler for a UI control in C#, you are actually creating a delegate instance that references the method that handles the event.

In C#, you can also use lambda expressions to create delegate instances inline, without defining a separate method. Here's an example:

```
csharp
```

Copy code

```
// Create a delegate instance using a lambda expression
```

```
MyDelegate myDelegate = (x, y) => x + y;
```

```
// Invoke the delegate, which will call the lambda expression
```

```
int result = myDelegate(1, 2);
```

This code creates a delegate instance that adds two integers using a lambda expression, which is a concise way of defining a method with a single expression.

- String builder vs concatenating method in C#?

In C#, you can concatenate strings using the '+' operator or the string.Concat method. However, if you are concatenating many strings or concatenating strings in a loop, using a StringBuilder can be more efficient.

When you concatenate strings using the '+' operator or the string.Concat method, a new string object is created each time, which can lead to a lot of unnecessary memory allocations and performance overhead. For example, consider the following code:

```
csharp
```

```
string result = "";
```

```
for (int i = 0; i < 10000; i++)
```

```
{
```

```
    result += i.ToString();
```

```
}
```

This code concatenates 10,000 integer values into a single string using the '+' operator. However, since a new string object is created each time, this code can be very slow and memory-intensive.

Using a StringBuilder, you can avoid creating unnecessary string objects and improve performance. Here's the same code rewritten using a StringBuilder:

```
csharp
```

```
StringBuilder sb = new StringBuilder();
```

```
for (int i = 0; i < 10000; i++)
```

```
{
```

```
    sb.Append(i.ToString());
```

```
}
```

```
string result = sb.ToString();
```

In this code, a StringBuilder object is used to accumulate the integer values, and the ToString method is called only once to create the final result string.

In general, you should use string concatenation when you are concatenating a small number of strings, or when you don't need to worry about performance. However, if you are concatenating many strings or concatenating strings in a loop, using a StringBuilder can be much more efficient.

- Downsides of using an ORM like entity framework

ORMs (Object-Relational Mappers) like Entity Framework are powerful tools for developers to interact with databases by providing an object-oriented interface to perform CRUD (Create, Read, Update, Delete) operations on database tables. However, there are some downsides of using an ORM like Entity Framework:

1. **Performance Overhead:** ORM's add a layer of abstraction between the application code and the database. This layer of abstraction can add performance overhead, which can impact application performance, especially in high-transaction applications.
2. **Learning Curve:** ORM's are complex tools that require some learning before they can be used effectively. Developers need to learn not only the syntax and features of the ORM, but also the underlying database concepts.
3. **Query Performance:** The SQL queries generated by an ORM may not be optimal in terms of performance. Developers need to have a good understanding of the database schema and query optimization techniques to write efficient queries.
4. **Vendor Lock-In:** ORM tools are typically tightly coupled with the database vendor and may not be compatible with other databases. This can make it difficult to switch to a different database vendor in the future.
5. **Limited Control:** ORM's provide limited control over the database operations, which can be a problem in complex scenarios. Developers may need to resort to raw SQL queries to achieve the desired behavior, which can defeat the purpose of using an ORM.

Overall, while ORMs like Entity Framework can be useful in simplifying database interactions, they come with some downsides that developers should be aware of and mitigate.

- What is Dapper framework?

Dapper is an open-source, lightweight object-relational mapping (ORM) framework for .NET developers. It was developed by Sam Saffron of Stack Overflow in 2011 as a simple and efficient way to map database results to strongly typed objects.

Dapper is designed to be fast, flexible, and easy to use. It provides a simple API for executing SQL queries and mapping the results to objects, without the overhead of traditional ORMs. Dapper uses dynamic SQL and provides support for mapping to both scalar values and complex object graphs.

One of the key benefits of using Dapper is its performance. Dapper is significantly faster than traditional ORMs like Entity Framework, because it avoids generating complex SQL statements and focuses on simple, direct queries. Dapper also supports asynchronous operations, allowing developers to execute database queries in a non-blocking manner.

Overall, Dapper is a great choice for developers who need a lightweight, fast, and flexible ORM solution for their .NET applications.

- Dapper vs Entity framework

Dapper and Entity Framework are both object-relational mapping (ORM) frameworks for .NET developers, but they have some key differences.

1. **Performance:** Dapper is generally considered to be faster than Entity Framework, especially for simple queries. This is because Dapper uses raw SQL queries and is optimized for performance, while Entity Framework generates complex SQL queries and has more overhead.
2. **Complexity:** Entity Framework is a more feature-rich ORM than Dapper, with support for advanced features like code-first migrations, lazy loading, and complex object graphs. This complexity comes at a cost, however, as it can make Entity Framework harder to learn and use than Dapper.
3. **Flexibility:** Dapper is more flexible than Entity Framework, because it allows developers to write raw SQL queries and map results to custom objects. Entity Framework, on the other hand, relies on its own query language (LINQ) and can be more restrictive in terms of what types of queries it can support.
4. **Ease of use:** Dapper is generally considered to be easier to use than Entity Framework, because its API is simpler and more straightforward. Dapper also has excellent documentation and a large community of users, which can make it easier to get help when you need it.

In summary, Dapper is a good choice for developers who need a lightweight, high-performance ORM with a simple API. Entity Framework is a better choice for developers who need a more feature-rich ORM with support for advanced features and complex object graphs.

- What is migration in C#?

In C#, migration refers to the process of updating a database schema to match changes in the codebase. This typically involves adding or modifying tables, columns, and relationships in the database to match changes in the object model.

Migrations are commonly used in object-relational mapping (ORM) frameworks like Entity Framework, which use migrations to keep the database schema in sync with the codebase as the application evolves. The migration process involves generating a set of SQL scripts that describe the changes to be made to the database, and then executing those scripts to update the database schema.

Migrations can be performed manually, by writing and executing the SQL scripts directly, or they can be automated using a migration framework like Entity Framework Migrations. Automated migrations can simplify the process of keeping the database schema in sync with the codebase, but they require careful planning and testing to ensure that the migrations are performed correctly and safely.

Overall, migrations are an important tool for managing the evolution of an application's database schema over time, and they are essential for maintaining the integrity and consistency of the data.

- Generate migration in C#

In C#, generating a migration typically involves using a migration framework like Entity Framework Migrations to create a set of SQL scripts that describe the changes to be made to the database schema.

Here are the basic steps to generate a migration using Entity Framework:

Install the Entity Framework package: Open the Package Manager Console and enter the following command to install the Entity Framework package:

Install-Package EntityFramework

Enable migrations: Enter the following command to enable migrations in your project:

Enable-Migrations

Add a migration: Enter the following command to create a new migration.

Add-Migration <migration_name>

Replace <migration_name> with a descriptive name for the migration.

Review the migration script: After running the Add-Migration command, Entity Framework will generate a migration script in the Migrations folder of your project. Review the script to ensure that it contains the expected changes to the database schema.

Update the database: To apply the migration and update the database schema, enter the following command:

Update-Database

This will apply the pending migrations to the database.

- Advantages of CLI in C# Entity framework

CLI tools (Command Line Interface) in C# Entity Framework offer several advantages, including:

Automation: CLI tools can automate repetitive tasks, making them faster and more efficient. For example, you can use CLI tools to create and migrate databases automatically, without having to write code for each step.

Portability: CLI tools can be used on any operating system and can be executed from any command line interface, making them highly portable. This makes it easier to collaborate with other developers, as everyone can use the same toolset.

Flexibility: CLI tools can be used to perform a wide range of tasks related to Entity Framework, such as generating code from an existing database schema, creating and updating databases, and scaffolding controllers and views.

Integration: CLI tools can be integrated with other tools and services, such as CI/CD pipelines, to automate the deployment process. This can help streamline the development process and reduce errors.

Learning curve: CLI tools can be easier to learn and use than a graphical user interface (GUI), as they often require less mouse clicking and navigation. This can be particularly useful for developers who are comfortable working in a command-line environment.

Overall, CLI tools provide a convenient and powerful way to manage Entity Framework projects and can help streamline.

- Execution plan in SQL

An execution plan in SQL is a detailed description of the steps the database management system (DBMS) will take to execute a specific SQL query. The execution plan provides insight into the algorithms and data structures used by the DBMS to execute the query.

The execution plan includes information such as:

1. The order in which the tables in the query will be accessed.
2. The access method used to read data from each table, such as a full table scan or an index lookup.
3. The type of join operation used to combine tables, such as a nested loop join or a hash join.
4. The filters and predicates that will be applied to the data.
5. The way the results will be sorted, grouped, or aggregated.

To view the execution plan for a SQL query, you can use a command such as EXPLAIN or EXPLAIN ANALYZE (in PostgreSQL). This command will display the execution plan in a structured format, showing the steps the DBMS will take to execute the query, the estimated cost of each step, and other relevant information. The execution plan can help you understand how the query is executed and identify potential performance issues or optimization opportunities.

- What is Cursors in SQL?

In SQL, a cursor is a database object that allows you to traverse the records returned by a SELECT statement one at a time. Cursors are often used when you need to process rows returned by a query in a specific order or when you need to perform complex calculations or operations on the data.

When you define a cursor, you typically provide a SELECT statement that defines the result set to be returned. Once the cursor is defined, you can use FETCH statements to retrieve one row at a time from the result set. Cursors also allow you to move forward and backward through the result set, and to skip over rows that do not meet certain criteria.

Cursors can be useful in certain situations, but they can also be slow and resource-intensive. In many cases, it is better to use set-based operations to manipulate data in SQL rather than cursors.

- Props of Tailwind CSS

Tailwind CSS is a popular utility-first CSS framework that makes it easy to create responsive, customizable, and accessible user interfaces. Some of the key properties of Tailwind CSS include:

Utility-first approach: Tailwind CSS is designed to be used with utility classes that define specific styles and behaviors, such as font sizes, colors, spacing, and flexbox layouts. This approach makes it easy to create custom designs without writing any custom CSS.

Responsive design: Tailwind CSS includes a set of responsive utility classes that allow you to create designs that adapt to different screen sizes and devices.

Customizable: Tailwind CSS is highly customizable, with configuration options that allow you to customize everything from the color palette to the spacing scale to the generated CSS output.

Accessible: Tailwind CSS is designed with accessibility in mind, with utility classes for common accessibility needs such as screen reader text and focus styles.

Comprehensive documentation: Tailwind CSS has comprehensive documentation that covers all of its features and provides examples and best practices for using the framework.

Growing community: Tailwind CSS has a growing community of developers who contribute plugins, tools, and resources that extend the framework's functionality.

Integration with other tools: Tailwind CSS integrates with other popular tools and frameworks such as React, Vue, and Next.js, making it easy to use Tailwind CSS in modern web development workflows.

Overall, Tailwind CSS is a powerful and flexible CSS framework that can help you create custom, responsive, and accessible user interfaces quickly and efficiently.

- Tailwind CSS vs Bootstrap

Tailwind CSS and Bootstrap are both popular CSS frameworks that are widely used in web development. While they share some similarities, they also have some important differences:

1. **Utility-first vs component-based:** Tailwind CSS is a utility-first CSS framework, which means that it provides a set of utility classes that you can use to create custom styles and layouts. Bootstrap, on the

other hand, is a component-based framework, which means that it provides pre-built components that you can use to create layouts and styles.

2. Customization: Tailwind CSS is highly customizable, with configuration options that allow you to customize everything from the color palette to the spacing scale to the generated CSS output. Bootstrap is also customizable, but it is more opinionated and provides fewer options for customization.

3. Learning curve: Tailwind CSS has a steeper learning curve than Bootstrap, as it requires you to learn a new set of utility classes and their syntax. Bootstrap, on the other hand, is more beginner-friendly, as it provides pre-built components that can be easily customized.

4. File size: Tailwind CSS generates more CSS code than Bootstrap, which can result in larger file sizes and slower page load times. Bootstrap generates less CSS code, but it also provides fewer options for customization.

5. Design philosophy: Tailwind CSS is designed to be a low-level framework that provides building blocks for creating custom designs, while Bootstrap is designed to provide pre-built components and a consistent design language out of the box.

Ultimately, the choice between Tailwind CSS and Bootstrap depends on your specific needs and preferences. Tailwind CSS can be a good choice if you value customization, flexibility, and a more low-level approach to building user interfaces. Bootstrap can be a good choice if you value ease of use, consistency, and a more component-based approach to building user interfaces.

- Event loop?

In the context of JavaScript, the event loop is an essential part of the browser's runtime environment, and is responsible for managing the execution of asynchronous code. JavaScript is a single-threaded language, which means that it can only execute one task at a time. However, the event loop allows JavaScript code to execute asynchronously by delegating long-running tasks to other parts of the browser, such as web APIs or the browser's rendering engine.

The event loop works by continuously monitoring the call stack, which is a data structure that tracks the execution of function calls in the program. When the call stack is empty, the event loop checks the task queue for any pending tasks, such as timer callbacks or user events. If there are tasks in the queue, the event loop retrieves the next task and adds it to the call stack, where it can be executed. If the task is asynchronous, the event loop will delegate the task to another part of the browser and continue to monitor the call stack for new tasks.

By using the event loop to manage asynchronous code, JavaScript can execute multiple tasks simultaneously without blocking the execution of other code. This allows JavaScript code to be more responsive and efficient, and is essential for building modern web applications that rely heavily on asynchronous operations such as network requests and user interactions.

```
add2byneras(2)(5); //7
```

```
const add2numbers = a => b => b? add2numbers(a+b): a
```

Intro Call Questions

It's my pleasure to introduce myself.

I am an innovative Senior <Front-end, Full Stack> developer with 10 years of experience managing all aspects of the development process for a startup to big-sized companies.

I have received Bachelor's degree in computer science from [university name] Nanyang Technological university.

My career as a web developer gets started by learning React, Angular, etc.

My major tech stack includes <React, Angular, JavaScript, Type Script, and React Native for the front end side and then Node, and Python for the back end as well as server less architecture using AWS Lambda.>

As a front-end developer, I've developed, customized, and launched easy-to-use, highly responsive, user-friendly websites and applications from scratch.

I have developed several web projects including E-commerce, fin-tech and gambling platforms.

When I worked at <Bamboo Group, Estonia> I worked as a full stack developer, so I am familiar with the team working with the agile scrum methodology and environment.

Since that, it equipped me with the ability to work with CI/CD pipelines and the Version Control Systems such as Github, GitLab, and BitBucket.

For ticket management tools, I am familiar with Jira, Trello, Asana and so on

I have good experience in unit test and end-to-end test such as Jest (Karma) and Cypress.

Especially, I led 3 junior members when I worked as a full stack developer in [Great IT, Sweden].

My last contract with my last company has been finished successfully & at the same time, my contract period was fulfilled. So I am currently self-employed so that I'm looking for new challenges to broaden my horizons.

I have experience in several projects and worked from several teams but I wish to expend and utilize my skills and expertise in web development and I am really interested in your company vision, so I applied to this job position and I'd like to bring my skills to this company.

And I feel that your company is the perfect place for me to challenge myself.

Thank you for giving me this opportunity to introduce myself!

- Talking about hotel booking system using React

<https://www.hotelscombined.com/>

Sure, I'd be happy to discuss my experience building a hotel booking system using React.

Firstly, I believe React is an excellent choice for building a hotel booking system. It allows for efficient rendering of components and can handle large amounts of data with ease. Additionally, React's modular structure makes it easy to break down the application into smaller, manageable pieces.

When building a hotel booking system with React, there are several key components that I would recommend including. These components would include a search bar, a list of available hotels, and a booking form.

The search bar would allow users to input their desired location, check-in and check-out dates, and other search criteria. This component would be responsible for sending a request to the server and returning a list of available hotels based on the user's input.

The list of available hotels would display the hotels that match the user's search criteria. Each hotel would be displayed as a separate card or tile, which would include details such as the hotel name, location, room rates, and room availability.

The booking form would allow users to select a specific room at their chosen hotel and book their stay. This form would include fields for the user's personal information, payment information, and any special requests they may have.

In terms of the technical aspects of building a hotel booking system with React, there are several tools and libraries that I would recommend using. These would include React Router for handling client-side routing, Axios for making API requests, and Moment.js for managing dates and times.

Overall, building a hotel booking system with React can be a challenging but rewarding experience. By breaking down the application into smaller, modular components and utilizing the right tools and libraries, it's possible to create a powerful and user-friendly booking system that meets the needs of both hoteliers and travelers.

- Talking about challenges to build hotel booking system

Certainly! There are several challenges that come with building a hotel booking system using React, especially when it comes to integrating with external APIs and managing large amounts of data.

One of the main challenges is handling API requests and responses. When building a hotel booking system, you'll need to integrate with third-party APIs such as hotel booking engines or payment gateways. These APIs can have complex authentication requirements, rate limits, and error handling procedures, which can be challenging to navigate. I will need to ensure that my React components are able to handle different types of responses and errors, and provide appropriate feedback to the user when something goes wrong.

Another challenge is managing the state of the application. When building a hotel booking system, I will be dealing with large amounts of data, such as hotel and room availability, user details, and payment information. I will need to ensure that React components are able to efficiently manage and

update this data, without causing unnecessary re-renders or impacting performance. I'll also need to consider how to handle complex user flows, such as booking multiple rooms or modifying an existing booking.

Security is another challenge when building a hotel booking system. I will need to ensure that sensitive user data such as personal and payment information is securely transmitted and stored. This may involve using encryption, tokenization, and other security measures to protect user data.

Finally, testing and debugging can be challenging when building a hotel booking system with React. With so many components and dependencies, it's important to have a robust testing strategy in place to catch errors and ensure the system is functioning as expected.

Overall, building a hotel booking system with React requires careful planning, attention to detail, and a deep understanding of how to manage complex data and external API integrations. But with the right approach, it's possible to build a powerful and user-friendly booking system that meets the needs of both hotels and travelers.

===== Professional accomplishments =====

as I told you earlier, in my last company, I developed cloud-based hotel management project.

this hotel management project supports the essential features such as Guest Profile, Email automation, Housekeeping management, Reservation management, Room management, Group bookings & reservations, Invoicing, Payment processing, and so on.

Among these essential features, I am responsible for dashboard, Housekeeping Management, Reservation Management, and Room Management.

I worked on not only frontend but also backend together.

Backend is (Node) ruby on rails, and frontend is react.js project.

frankly, at first time, the frontend was jquery. we converted jquery into react components for 6 months.

for backend, I developed REST api.

for frontend, I got a figma design from UI designer, convert the figma design into pixel-perfect react components and integrate backend api with frontend.

at that time, The project is running under agile methodology.

We have a sprint every 2 weeks.

During the sprint, We plan the stories, and design, develop, test and discover.

We have a daily standup every day, and talk what I worked on yesterday, what I am working on today, what issues are blocking me.

We used slack for communication and jira for project tracking.

for testing, I write the rspec(BDD or Python Behave(python), Jest(Node) for backend testing. also I used Jest and Cypress for the frontend testing.

for background job, We used sidekiq gem.

1. Housekeeping management

It allow the housekeeping team to mark rooms as clean, dirty, or out of order.

Having this information in one central place gives the front desk visibility of room status, so that they can accommodate early check-in's and room moves without needing to radio the housekeeping supervisor.

2. Reservation managment

By housing all of the reservations digitally, hoteliers can make a paper reservation book a thing of the past.

Within reservations management, You are able to set rates, control availability, offer promotins, and visualize reservations on a calendar.

3. Room management

This is the virtual front desk. This functionality lets front desk agents check guests in and out, and cancel or extend reservations with just a few simple clicks.

===== Challenges =====

I have got 2 challenges in the backend, and 1 challenge for frontend.

--- Frontend challenge(customer action history)

We need to render huge data inside the data grid component.

at least 10 thousands records need to render in the grid component.

It is very slow because of a lot of component rendering.

I suggest to use react virtualized and react window

React Virtualized creates a window frame that allows records to slide in and out of view as the user scrolls through it.

Therefore, It only renders the components what we see in the grid component, according to scroll position.

And it becomes faster.

--- First challenge is

for backend, I have got a race-condition for reservation management.

Race conditions are undesirable situations caused when the concurrent execution of the code causes negative consequences comparing to the scenario of having them executed sequentially.

Let me give you a example about race-condition

currently, there is a one room available in the hotel.

10 customers are trying to book this one room at the same time.

in this situation, only one customer must book the room. but unfortunately, some of the customers book this one room together.

Therefore, there was a duplication record of the same room.

actually, we can not reproduce this kind of race-condition on the development server. It happens only on the production server.

We have got this error-reports from the hotelier about this race-condition issues, and figured out we need to lock the records to prevent race-condition.

normally, rails frameworks supports Pessimistic lock.

Pessimistic lock is actually lock the record. so that others need to wait until this record is unlocked.

We need to handle a lot of customers transactions everytime. It is not useful for us.

We decided to use Optimistic Lock instead of Pessimistic lock.

The reason why we choose the Optimistic Lock is, it use lock version of the record.

so it check the lock version before update the record, if lock version is different, It gets the record again and update.

Therefore, We can prevent race-condition successfully without any technical issues.

--- Second challenge is

for backend, dashboard api is too low. It tooks more than 10 seconds.

We used database view to select and join the tables. I suggest to use materialized view.

The materilzed view is similar to view, but it actually store the data in the disk.

When we call materilzed view, It returns the data from the cache instead of executing the SQL view query.

after all, dashboard api only took 1 second. It's amazing

- Talking about challenges to build e-commerce online shopping system using Angular + Python for interview?

Anniecloth.com

Sure, here are some potential challenges that you could mention when discussing building an e-commerce online shopping system using Angular and Python:

1. Scalability: As an e-commerce website grows and attracts more customers, it will need to handle a higher volume of traffic and more transactions. This can place a strain on the system's backend, which needs to be designed to handle large amounts of data and requests.

2. Security: With e-commerce, security is a top concern. The system must be designed to protect sensitive customer data, such as credit card information, and prevent fraudulent activities.
3. User experience: E-commerce systems need to be designed with the user in mind, providing a smooth and intuitive shopping experience. This involves designing the front-end UI to be responsive, fast, and easy to navigate.
4. Integration with third-party systems: An e-commerce system needs to integrate with various third-party services such as payment gateways, shipping carriers, and inventory management systems.
5. Performance optimization: E-commerce systems must be optimized for performance to ensure fast page load times and quick checkout experiences. This involves optimizing code, minimizing network requests, and caching data where appropriate.
6. Data analytics: E-commerce systems generate a lot of data, which can be used to improve the user experience, personalize product recommendations, and inform business decisions. The system must be designed to collect, store, and analyze this data effectively.
7. Testing and quality assurance: Testing is critical for e-commerce systems to ensure that they are functioning as intended and that there are no errors or bugs. Automated testing frameworks should be implemented to ensure that new features and updates don't break existing functionality.
8. Search functionality: E-commerce systems require robust search functionality to help users find products quickly and easily. This involves designing a search algorithm that takes into account factors such as relevancy, popularity, and price.
9. Inventory management: Inventory management is an important aspect of e-commerce systems, which must track inventory levels and ensure that products are always in stock.
10. Order fulfillment and tracking: An e-commerce system must have an efficient order fulfillment process, which includes tracking orders and providing updates to customers on the status of their orders.

- how to manage stress when working on a project?

1. Set realistic goals: It's important to set achievable goals for my project and break them down into smaller, manageable tasks. This will help me feel less overwhelmed and make it easier to track my progress.
2. Prioritize tasks: Prioritize the tasks that are most important and tackle them first. This will help me feel more in control and reduce stress.
3. Take breaks: Taking short breaks throughout the day can help me recharge and refocus. Try taking a walk outside, doing some stretching exercises, or listening to music.
4. Practice mindfulness: Mindfulness techniques such as meditation or deep breathing can help me manage stress and stay focused on my work.
5. Get support: Talk to my colleagues, friends, or family members about my project and the challenges I am facing. Sometimes just talking things through can help me feel more relaxed and better able to handle stress.

6. Exercise regularly: Regular exercise can help reduce stress and improve my overall well-being. Even just a few minutes of physical activity each day can make a difference.

7. Stay organized: Keeping my workspace organized and clutter-free can help reduce stress and make it easier to focus on my work.

- how can I handle negative feedback from the QA team?

1. Stay calm and professional: Try not to let negative feedback affect my emotions. Stay calm and professional in my interactions with the QA team.

2. Listen and ask questions: Listen carefully to the feedback and ask clarifying questions if I am unsure about anything. This will help me understand what went wrong and how I can improve.

3. Take ownership: Take ownership of my mistakes and don't try to shift the blame onto others. This will help me build trust with the QA team and show that I am committed to improving my work.

4. Learn from your mistakes: Use the feedback as an opportunity to learn and improve. Identify the areas where I need to improve and take steps to address them.

5. Communicate with the team: Keep the QA team updated on my progress as I work to address the issues they raised.

I think negative feedback is an opportunity to grow and improve, and it's a common part of the development process.

- How to mentor junior developers?

I encourage Google (or any other search tool).

If I know the answer can be found online easily, I tell them to look it up .

So I want to teach them how to teach themselves, and not have them become dependent on me.

I have to make myself available to answer questions.

I do code reviews regularly to tell them what they are doing right/wrong.

I use code reviews as an opportunity to point out best practices.

Get them started early with planning/documentating what they are going to be doing instead of letting them begin by writing code.

I am always open to learning from them.

They probably spend more time than I learning, and it's possible that they learn something that I didn't know.

I help them learn from their mistakes.

There are going to be mistakes, so I will show them that mistakes are part of learning, and that they should use them as an opportunity to learn.

Instead of telling them how to do something, I try to help them figure it out themselves.

This will help improve their ability to logically work through a problem, and increases their learning ability.

Instead of telling them what they did wrong, I tell them of ways they can improve it.

- Strengths:

Strong problem-solving skills - I have the ability to break down complex problems into smaller, more manageable tasks, and find effective solutions to them.

Attention to detail - I am meticulous in my work and take pride in producing high-quality code that is efficient and error-free.

Strong communication skills - I am able to communicate technical information to non-technical stakeholders in a clear and concise manner.

Passion for learning - I have a constant desire to learn and improve my skills, and I am always eager to take on new challenges.

Collaborative - I work well with others and value teamwork, and I believe that working together is essential to achieving successful project outcomes.

- Weaknesses:

Time management - Sometimes I struggle with managing my time effectively, especially when I get deeply immersed in a project. However, I have learned to overcome this weakness by setting realistic timelines and breaking down tasks into smaller, more manageable pieces.

Public speaking - While I am confident in my ability to communicate with my team and colleagues, I sometimes struggle with public speaking. However, I have been taking steps to improve this weakness by attending public speaking workshops and practicing my presentation skills.

Accepting criticism - In the past, I have sometimes had difficulty accepting constructive criticism on my work. However, I have learned that feedback is crucial to growth and improvement, and I now make a conscious effort to take feedback in stride and use it to improve my work.

Lack of expertise in certain technologies - While I have experience with a range of programming languages and technologies, there are still some areas where I lack expertise. However, I am always willing to learn and am open to expanding my skill set to include new technologies.

Delegating tasks - Sometimes I have trouble delegating tasks to other team members because I feel that I can complete them better or faster myself. However, I have learned that delegation is an important part of teamwork, and I am working to become better at delegating tasks and trusting my team members to handle them effectively.

- How do you handle urgent deadlines?

I am used to working under tight deadlines, so I set my most urgent tasks at the top of my to-do list every morning when I get to work. Then, I establish a clear deadline for myself that's usually a day in

advance of company due date. Recently, I had to shift around my workload to accommodate an urgent product order. The client wanted custom modifications to the product completed in a week, but our normal lead time for implementing modifications is around 10 to 14 days. I communicated with my production department and manufacturing team to implement urgent changes in the production timeline. These adjustments allowed us to ship the product to the client on time, effectively reducing the time it took to apply the customization by three days.

- How would you describe our company culture?

Tell about company vision and mission.

Yeah, challenging and collaborating. You, I mean your company seems that developers are encouraged to explore the full potential of their skill sets and that they're likely to grow through job experience.

Regarding the collaborating, I have noticed that software engineers and your teams are working well together cross-functionally to accomplish your goals.

- Would you rather work alone or with a team?

Of course, in a team. As you know, anyone can not do anything alone because everyone has limited knowledge and experience.

So, without any help, without any collaboration, you are nothing. That's my phrase. haha.

- How do you feel about working in a team environment?

My ideal work environment is one that focuses on teamwork.

I like companies like yours that put an effort in gelling the group together and making sure everyone is supported. I like that sort of structure and formality of knowing what you're supposed to do and when you are aware of the situation. I work best when I have a group of positive people working around me and where I can focus on putting my talent to work.

Concept in my resume.

-SDLC

Software Development Life Cycle (SDLC) is a methodology used in software engineering to design, develop, test, and deploy software systems. It is a structured approach that outlines the various phases involved in building software.

The different phases of the SDLC are:

1. Requirement Gathering: The first phase is to gather the requirements from the client or end-user. The requirements are documented, analyzed, and prioritized.
2. Design: In this phase, the software architecture is designed based on the requirements. The design is documented, reviewed, and approved.
3. Implementation: In this phase, the software is developed using a programming language or a development platform. The code is tested and reviewed for quality.

4. Testing: In this phase, the software is tested for functionality, performance, and security. Testing includes unit testing, integration testing, system testing, and acceptance testing.

5. Deployment: Once the software is tested and approved, it is deployed in the production environment. The deployment process involves installing the software on the client's servers and configuring it.

6. Maintenance: After the software is deployed, it needs to be maintained to ensure it continues to function properly. Maintenance includes bug fixing, upgrades, and enhancements.

Each phase of the SDLC has specific deliverables and goals that must be achieved before moving to the next phase. The SDLC ensures that the software is developed according to the client's requirements and meets the desired quality standards.

- STLC

Software Testing Life Cycle (STLC) is a methodology used in software testing to ensure the quality of the software before it is deployed to production. It is a structured approach that outlines the various phases involved in testing software.

The different phases of the STLC are:

1. Requirement Analysis: The first phase is to analyze the software requirements to identify the test cases that need to be developed. This involves reviewing the software requirements, creating test scenarios, and identifying the testing scope.

2. Test Planning: In this phase, the test strategy is developed, and a test plan is created. The test plan includes the objectives, scope, testing types, and resources required for testing.

3. Test Case Development: In this phase, the test cases are developed based on the test scenarios identified in the requirement analysis phase. Test cases are designed to cover all the functionality of the software.

4. Test Environment Setup: In this phase, the testing environment is set up, including the hardware and software requirements.

5. Test Execution: In this phase, the test cases are executed, and the software is tested for functionality, performance, and security. Test results are recorded, and defects are reported.

6. Test Reporting: In this phase, the test results are compiled and analyzed, and a test report is generated. The test report includes the testing status, test results, and defects found during testing.

7. Defect Tracking: In this phase, the defects found during testing are tracked, and a defect report is created. Defects are assigned a priority and severity, and the defect resolution process is initiated.

8. Test Closure: In this phase, the testing is completed, and the software is ready for deployment. A final report is created, and the testing team prepares for the next software release.

Each phase of the STLC has specific deliverables and goals that must be achieved before moving to the next phase. The STLC ensures that the software is thoroughly tested and meets the desired quality standards before it is deployed to production.

- LESS

LESS (Leaner Style Sheets) is a preprocessor scripting language that extends the capabilities of CSS (Cascading Style Sheets). It is a dynamic stylesheet language that can be compiled into CSS code, which is then interpreted by web browsers.

The main features of LESS are:

1. **Variables:** LESS allows the use of variables to store reusable values such as colors, font sizes, and margins. This simplifies the process of making changes to the styles of a website.
2. **Nesting:** LESS allows the nesting of CSS rules, making it easier to organize and maintain the style sheets.
3. **Mixins:** LESS allows the creation of mixins, which are reusable sets of CSS properties. Mixins can be used to simplify the process of applying styles to multiple elements on a web page.
4. **Functions:** LESS includes built-in functions that can be used to perform calculations, manipulate colors, and convert units of measurement.
5. **Importing:** LESS allows the importing of external style sheets, making it easier to reuse styles across multiple web pages.

LESS is widely used in web development to create more organized, efficient, and maintainable CSS code. It simplifies the process of writing and managing styles, resulting in faster development times and fewer errors. Additionally, the use of LESS can help improve website performance by reducing the amount of CSS code that needs to be downloaded by web browsers.

- SASS

SASS (Syntactically Awesome Style Sheets) is a preprocessor scripting language that extends the capabilities of CSS (Cascading Style Sheets). It is a dynamic stylesheet language that can be compiled into CSS code, which is then interpreted by web browsers.

The main features of SASS are:

1. **Variables:** SASS allows the use of variables to store reusable values such as colors, font sizes, and margins. This simplifies the process of making changes to the styles of a website.
2. **Nesting:** SASS allows the nesting of CSS rules, making it easier to organize and maintain the style sheets.
3. **Mixins:** SASS allows the creation of mixins, which are reusable sets of CSS properties. Mixins can be used to simplify the process of applying styles to multiple elements on a web page.
4. **Functions:** SASS includes built-in functions that can be used to perform calculations, manipulate colors, and convert units of measurement.

5. Importing: SASS allows the importing of external style sheets, making it easier to reuse styles across multiple web pages.

6. Control directives: SASS includes control directives such as `@if`, `@for`, and `@each`, which allow developers to write more powerful and flexible style sheets.

SASS is widely used in web development to create more organized, efficient, and maintainable CSS code. It simplifies the process of writing and managing styles, resulting in faster development times and fewer errors. Additionally, the use of SASS can help improve website performance by reducing the amount of CSS code that needs to be downloaded by web browsers.

- SCSS

SCSS (Sassy CSS) is a preprocessor scripting language that extends the capabilities of CSS (Cascading Style Sheets). It is a dynamic stylesheet language that can be compiled into CSS code, which is then interpreted by web browsers.

SCSS is a superset of CSS3 syntax, which means that all valid CSS3 code is also valid SCSS code. It provides additional features that make it easier to write and maintain complex CSS stylesheets. Some of the main features of SCSS are:

1. Variables: SCSS allows the use of variables to store reusable values such as colors, font sizes, and margins. This simplifies the process of making changes to the styles of a website.

2. Nesting: SCSS allows the nesting of CSS rules, making it easier to organize and maintain the style sheets.

3. Mixins: SCSS allows the creation of mixins, which are reusable sets of CSS properties. Mixins can be used to simplify the process of applying styles to multiple elements on a web page.

4. Functions: SCSS includes built-in functions that can be used to perform calculations, manipulate colors, and convert units of measurement.

5. Importing: SCSS allows the importing of external style sheets, making it easier to reuse styles across multiple web pages.

6. Control directives: SCSS includes control directives such as `@if`, `@for`, and `@each`, which allow developers to write more powerful and flexible style sheets.

SCSS is widely used in web development to create more organized, efficient, and maintainable CSS code. It simplifies the process of writing and managing styles, resulting in faster development times and fewer errors. Additionally, the use of SCSS can help improve website performance by reducing the amount of CSS code that needs to be downloaded by web browsers.

- TDD

TDD stands for Test Driven Development. It is a software development process where tests are written before writing the actual code.

In TDD, the process typically works as follows:

1. Write a test for a small piece of functionality that is required by the application.
2. Run the test, which should fail because the functionality is not yet implemented.
3. Write the code to make the test pass.
4. Refactor the code as necessary to make it clean and maintainable.
5. Repeat the process for the next small piece of functionality.

The main benefit of TDD is that it helps ensure that the code being written is correct and meets the requirements. By writing tests first, developers are forced to think about the requirements and edge cases before writing any code. This can help catch bugs early in the development process when they are less expensive to fix.

TDD also promotes good software design practices, such as writing modular and reusable code. Since tests are written first, the code is designed to be testable, which often leads to more modular and loosely coupled code.

TDD is often associated with agile software development methodologies, where small iterations and frequent feedback are key. However, TDD can be used in any software development process and is often used in conjunction with other software development practices, such as continuous integration and deployment.

- BDD

BDD stands for Behavior Driven Development. It is a software development methodology that focuses on writing tests in a way that they describe the expected behavior of the system in a human-readable format. BDD is based on the principles of Test Driven Development (TDD) and Domain Driven Design (DDD).

BDD tests are written in a specific format called Gherkin, which uses natural language to describe the behavior of the system in terms of features, scenarios, and steps. Gherkin syntax includes keywords such as Given, When, Then, And, and But, which are used to describe the steps of the test in a structured manner.

BDD encourages collaboration between developers, testers, and business stakeholders to ensure that everyone understands the expected behavior of the system. BDD tests can be used to drive the development process, and can help to ensure that the system meets the requirements of the stakeholders.

Some benefits of using BDD include:

1. Clarity: BDD tests are written in a human-readable format, which makes it easier for everyone to understand the expected behavior of the system.
2. Collaboration: BDD encourages collaboration between developers, testers, and business stakeholders, which helps to ensure that everyone is on the same page.
3. Automated testing: BDD tests can be automated, which can save time and improve the quality of the tests.

4. Regression testing: BDD tests can be used for regression testing, which ensures that changes to the system do not break existing functionality.

Overall, BDD is a useful methodology for developing software that is focused on the expected behavior of the system. By using BDD, developers can ensure that the system meets the requirements of the stakeholders, and that the tests accurately reflect the behavior of the system.

- DDD

DDD stands for Domain-Driven Design, a software development approach that focuses on the core domain and business logic of the software system. The goal of DDD is to create a model of the domain that is based on a deep understanding of the problem domain and the needs of the users.

DDD emphasizes the use of a shared language between developers and business stakeholders, and encourages collaboration between different roles to create a cohesive and well-defined model of the domain. The domain model is the central focus of DDD, and is used to drive the design of the software system.

Some key concepts of DDD include:

1. Ubiquitous Language: DDD promotes the use of a shared language that is used by both developers and business stakeholders to describe the domain model.
2. Bounded Contexts: DDD recognizes that different parts of the system may have different models of the domain, and defines boundaries around these models to ensure that they are consistent and well-defined.
3. Entities, Value Objects, and Aggregates: DDD defines different types of objects that represent the domain model, including entities (objects with a unique identity), value objects (objects without identity), and aggregates (groups of related objects).
4. Domain Events: DDD promotes the use of domain events to represent important changes in the domain model, which can be used to trigger other actions or update other parts of the system.
5. Context Mapping: DDD provides techniques for mapping the relationships between different bounded contexts, which can help to ensure that the models are consistent and well-integrated.

Overall, DDD is a powerful approach to software development that emphasizes the importance of understanding the domain and business logic of the system. By focusing on the domain model and using a shared language, developers can create software systems that are more closely aligned with the needs of the users and stakeholders.

- SOLID Design pattern

SOLID is a set of design principles for object-oriented programming that are designed to make software systems more maintainable, flexible, and scalable. The principles are named after the first letter of each principle:

1. Single Responsibility Principle (SRP): A class should have only one reason to change, meaning it should have only one responsibility.

2. Open-Closed Principle (OCP): A class should be open for extension but closed for modification. This means that new behavior can be added to a class without changing its existing code.
3. Liskov Substitution Principle (LSP): Subtypes should be substitutable for their base types. This means that objects of a subclass should be able to replace objects of the parent class without causing errors.
4. Interface Segregation Principle (ISP): Clients should not be forced to depend on interfaces they do not use. This means that interfaces should be designed to be as small and focused as possible.
5. Dependency Inversion Principle (DIP): High-level modules should not depend on low-level modules. Instead, they should both depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

By following these principles, developers can create software systems that are easier to understand, modify, and maintain over time. SOLID principles help to reduce code complexity, increase code reusability, and improve testability.

- Monolithic application

In software engineering, a monolithic application is an architecture where all the components and modules of the application are tightly integrated and interdependent. In other words, a monolithic application is a single, self-contained unit of code that handles all the functionalities of an application.

In a monolithic application, all the modules are combined into a single executable file or binary, and all the components share a single codebase, database, and runtime environment. This means that any changes to one part of the application may require re-deploying the entire application, which can be time-consuming and error-prone.

Monolithic applications are often used for small to medium-sized projects or when the application's functionalities are limited. They are relatively easy to develop, test and deploy in a single environment. However, as the application grows in size and complexity, it becomes difficult to maintain, scale and evolve.

With the rise of microservices, many organizations are now moving away from monolithic architectures to more modular and scalable architectures. Microservices allow for more flexibility and agility, as individual services can be developed and deployed independently, making it easier to maintain, scale and evolve the application over time.

- Microservices Architecture

Microservices architecture is a software design pattern that structures an application as a collection of small, independent, and loosely coupled services, each performing a specific business function. Each microservice is designed to be independently deployable and scalable, and communicates with other microservices through a lightweight protocol like HTTP or message queues.

Some key features of microservices architecture include:

1. Decentralized Data Management: Each microservice owns and manages its own data, reducing dependencies on other services and allowing for greater autonomy.

2. **Service Autonomy:** Each microservice is designed to be self-contained and independent, making it easier to deploy, maintain, and evolve.
3. **Resilience:** Microservices are designed to be fault-tolerant and resilient, with each service able to handle failures and continue functioning.
4. **Continuous Integration and Deployment:** Microservices architecture enables continuous integration and deployment, making it easier to deploy new features and updates without disrupting the entire application.
4. **Polyglot Programming:** Microservices architecture allows different microservices to be written in different programming languages and technologies, allowing teams to choose the best tool for each specific function.

Overall, microservices architecture is designed to enable greater agility, scalability, and resilience in software development. By breaking down a large monolithic application into smaller, independent services, teams can develop and deploy applications more quickly, respond more quickly to changing business requirements, and maintain a high level of service availability and performance.

- PrimeNG

PrimeNG is an open-source user interface (UI) component library for Angular, a popular web application framework. It provides a set of rich and responsive UI components that can be used to build modern and professional-looking web applications.

PrimeNG includes a wide range of UI components such as buttons, menus, dialogs, input fields, data tables, charts, and more. All of these components are designed to be highly customizable and easy to use. They are built with accessibility and internationalization in mind, making them suitable for a global audience.

PrimeNG also includes a number of themes that developers can use to style their applications quickly and easily. These themes are based on popular design frameworks such as Bootstrap and Material Design and provide a consistent and modern look and feel to the UI components.

- Angular-animate

angular-animate is a module in the Angular framework that provides support for animations in Angular applications. It provides a set of directives and services that can be used to animate elements and components in response to user actions or other events.

angular-animate allows developers to create animations using CSS3 transitions and keyframe animations. It also provides support for JavaScript animations using the \$animate service, which allows developers to create custom animations that can be triggered by any event.

The main features of angular-animate are:

1. **Animations for AngularJS directives:** angular-animate provides a set of pre-built animations that can be used with AngularJS directives such as ng-show, ng-hide, ng-repeat, and ng-view.
2. **CSS3 animations:** angular-animate supports CSS3 transitions and keyframe animations, making it easy to create animations using standard CSS code.

3. JavaScript animations: angular-animate provides the \$animate service, which allows developers to create custom animations using JavaScript code.
4. Animation callbacks: angular-animate provides callbacks for animation events such as start, end, and cancel, allowing developers to perform additional actions before, during, or after an animation.
5. Animation configuration: angular-animate allows developers to configure animation properties such as duration, easing, and delay, making it easy to customize the look and feel of animations.

- Angular-filter

Angular-filter is a module in the AngularJS framework that provides a set of filters that can be used to manipulate data in AngularJS applications. It includes a variety of filters such as orderBy, filter, limitTo, uppercase, lowercase, and currency.

The main purpose of Angular-filter is to simplify the process of filtering and formatting data in AngularJS applications. It provides a set of reusable filters that can be used to sort and filter arrays, format strings and numbers, and perform other common data manipulations.

Some of the most commonly used filters in Angular-filter are:

1. orderBy: Used to sort arrays based on one or more properties.
2. filter: Used to filter arrays based on a search string or a set of criteria.
3. limitTo: Used to limit the number of items displayed in an array.
4. uppercase and lowercase: Used to convert strings to uppercase or lowercase.
5. currency: Used to format numbers as currency.

Angular-filter is easy to use and can be added to an AngularJS application by including the angular-filter.js file and adding the ngFilter module as a dependency. Once added, filters can be used in HTML templates or in JavaScript code by using the | operator to apply the filter to data.

For example, to filter an array of objects based on a search string, the following code can be used:

```
<div ng-repeat="item in items | filter:searchText">
  {{item.name}}
</div>
```

- Angular-Mocks

angular-mocks is a module in the AngularJS framework that provides a set of utilities for testing AngularJS applications. It includes a variety of services, directives, and helper functions that make it easier to write unit tests and integration tests for AngularJS applications.

The main purpose of angular-mocks is to simplify the process of writing and running tests for AngularJS applications. It provides a set of tools and utilities that can be used to mock dependencies, simulate user interactions, and test AngularJS directives and services.

Some of the most commonly used features in angular-mocks are:

1. `$httpBackend`: A service that can be used to mock HTTP requests and responses, making it easier to test AngularJS services that interact with remote servers.
2. `$compile`: A service that can be used to compile AngularJS templates, making it easier to test AngularJS directives and components.
3. `$rootScope`: A service that provides access to the root scope of an AngularJS application, making it easier to test event handlers and scope functions.
4. `spyOn`: A helper function that can be used to spy on functions and methods, making it easier to test how they are called and what they return.
5. `inject`: A helper function that can be used to inject AngularJS services and dependencies into tests.

angular-mocks is easy to use and can be added to an AngularJS application by including the `angular-mocks.js` file and adding the `ngMock` module as a dependency in the test file. Once added, the utilities in angular-mocks can be used to write tests for AngularJS applications.

Overall, angular-mocks is an essential tool for testing AngularJS applications, making it easier to write and run tests and ensure that applications are functioning as expected.

- Angular-Route

Angular-Route is a module in the AngularJS framework that provides routing and navigation capabilities for single-page applications (SPAs). It allows developers to create routes and views, and then map URLs to those views. When a user navigates to a specific URL, Angular-Route loads the corresponding view and updates the browser's URL.

The main purpose of Angular-Route is to provide a structured way to manage navigation in AngularJS applications. It allows developers to create multiple views and map them to different URLs, making it easier to create complex SPAs with multiple pages and states.

Some of the key features of Angular-Route are:

1. **Routing**: Angular-Route allows developers to define routes for their application, and map those routes to views.
2. **Navigation**: Angular-Route provides built-in navigation directives, such as `ng-view` and `ng-route`, that allow developers to navigate between views and update the browser's URL.
3. **History**: Angular-Route provides a built-in history service that allows developers to navigate back and forth between different states in their application.
4. **Animations**: Angular-Route provides a way to animate views as they are loaded and unloaded, making it easy to create engaging user experiences.
5. **HTML5 mode**: Angular-Route supports HTML5 mode, which removes the hash (#) from the URL and allows for cleaner URLs.

Angular-Route is easy to use and can be added to an AngularJS application by including the `angular-route.js` file and adding the `ngRoute` module as a dependency in the main app module. Once added,

developers can define routes and views using the `$routeProvider` service, and use built-in directives such as `ng-view` to load views based on the current route.

- Angular-Sanitize

Angular-Sanitize is a module in the AngularJS framework that provides a set of directives and services for sanitizing and filtering HTML and CSS content in web applications. It is used to prevent cross-site scripting (XSS) attacks and other security vulnerabilities by sanitizing user-generated content before rendering it in the browser.

The main purpose of Angular-Sanitize is to protect applications from malicious attacks by removing potentially harmful HTML and CSS content. It provides a set of tools and services that can be used to clean up user-generated content, strip out unwanted tags and attributes, and ensure that only safe content is rendered in the browser.

Some of the most commonly used features in Angular-Sanitize are:

1. `ng-bind-html`: A directive that can be used to bind HTML content to a DOM element, while sanitizing the content to prevent XSS attacks.
2. `$sanitize`: A service that can be used to sanitize and filter HTML and CSS content, removing potentially harmful content and ensuring that only safe content is rendered in the browser.
3. `$sce`: A service that provides a way to mark strings as trusted or untrusted, and to create trusted HTML and CSS content that can be rendered in the browser.
4. `$sceDelegate`: A service that provides a way to configure the trusted content policy for an application, and to define a whitelist of allowed URLs and domains.

Angular-Sanitize is easy to use and can be added to an AngularJS application by including the `angular-sanitize.js` file and adding the `ngSanitize` module as a dependency in the main app module. Once added, the `ng-bind-html` directive and `$sanitize` service can be used to sanitize and filter HTML and CSS content in the application.

Overall, Angular-Sanitize is an important module for ensuring the security and integrity of AngularJS applications, and should be used whenever user-generated content is displayed in the browser.

- Angular-Touch

Angular-Touch is a module in the AngularJS framework that provides support for touch-based user interactions in web applications. It is used to create mobile-friendly web applications that can be used on touch-based devices such as smartphones and tablets.

The main purpose of Angular-Touch is to provide a set of directives and services that can be used to detect touch-based user interactions and respond to them in a meaningful way. It provides support for gestures such as tapping, swiping, and dragging, and allows developers to create custom touch-based interactions for their applications.

Some of the most commonly used features in Angular-Touch are:

1. **ngTouch**: A module that provides support for touch-based user interactions, including tapping, swiping, and dragging.
2. **ngSwipeLeft** and **ngSwipeRight**: Directives that can be used to detect left and right swipe gestures, and trigger an action in response.
3. **\$swipe**: A service that can be used to detect swipe gestures and provide information about the direction and speed of the swipe.
4. **\$drag**: A service that can be used to detect drag gestures and provide information about the position and movement of the drag.

Angular-Touch is easy to use and can be added to an AngularJS application by including the `angular-touch.js` file and adding the `ngTouch` module as a dependency in the main app module. Once added, developers can use the built-in directives and services to detect touch-based user interactions and respond to them in a meaningful way.

Overall, Angular-Touch is an important module for creating mobile-friendly web applications that can be used on touch-based devices. It provides a set of tools and services that can be used to detect touch-based interactions and respond to them in a way that provides a great user experience.

- Angular-Navigation

Angular-Navigation is a module in the AngularJS framework that provides support for navigating between different views or pages in a web application. It is used to create a navigation menu, manage the state of the application, and handle URL routing.

The main purpose of Angular-Navigation is to provide a set of directives and services that can be used to create a navigation menu and manage the state of the application as the user navigates between different views or pages. It provides support for URL routing, which allows users to bookmark specific pages or share them with others.

Some of the most commonly used features in Angular-Navigation are:

1. **ngRoute**: A module that provides support for URL routing, which allows users to navigate between different views or pages in the application.
2. **\$routeProvider**: A service that can be used to define the routes for the application, including the URL path, the template to use for the view, and the controller to use for the view.
3. **\$routeParams**: A service that provides access to the parameters in the URL, which can be used to customize the content displayed in the view.
4. **ng-view**: A directive that can be used to display the content of the current view, which is determined by the URL route.

Angular-Navigation is easy to use and can be added to an AngularJS application by including the `angular-route.js` file and adding the `ngRoute` module as a dependency in the main app module. Once added, developers can use the built-in directives and services to create a navigation menu and manage the state of the application as the user navigates between different views or pages.

Overall, Angular-Navigation is an important module for creating a seamless and intuitive user experience in an AngularJS application. It provides a set of tools and services that can be used to manage the state of the application and handle URL routing, allowing users to navigate between different views or pages with ease.

- NgRx

NgRx is a state management library for Angular applications. It provides a set of tools and techniques for managing the state of an application in a consistent and predictable way, and is based on the principles of Redux, a popular state management library for React applications.

The main purpose of NgRx is to provide a predictable and scalable way to manage the state of an Angular application. It provides a set of building blocks, including actions, reducers, selectors, and effects, that work together to manage the state of the application in a single, centralized location. This makes it easier to reason about the behavior of the application, and helps to prevent common issues such as race conditions and data inconsistencies.

Some of the key features of NgRx include:

1. **Single source of truth:** The application state is managed in a single, centralized location, making it easier to reason about the behavior of the application and maintain consistency across different components.
2. **Predictable state changes:** All state changes are handled through a series of well-defined actions and reducers, which makes it easier to understand and debug the behavior of the application.
3. **Time-travel debugging:** NgRx provides tools for debugging and inspecting the state of the application at different points in time, which makes it easier to identify and resolve issues.
4. **Scalability:** NgRx is designed to scale with the complexity of an application, making it a good choice for large or enterprise-level projects.

Overall, NgRx is a powerful tool for managing the state of an Angular application. It provides a consistent and predictable way to manage the application state, making it easier to reason about the behavior of the application and maintain consistency across different components.

- Redux-Saga

Redux Saga is a middleware library for Redux that enables handling of side effects, such as asynchronous API calls, in a more organized and testable way. It uses a generator-based approach to handle asynchronous tasks, providing a cleaner and more readable way to manage complex asynchronous flows.

The main purpose of Redux Saga is to provide a declarative and composable way to handle side effects in a Redux application. It allows developers to separate the logic for handling side effects from the actual Redux reducers, which makes the code more modular and easier to test. Redux Saga uses a generator-based approach to handle side effects, which means that asynchronous tasks can be written in a synchronous style, without resorting to nested callbacks or promises.

Some of the key features of Redux Saga include:

1. Declarative side effect handling: Redux Saga provides a set of declarative and composable effects, such as take, put, call, and fork, that make it easy to handle side effects in a more organized way.
2. Testability: Redux Saga makes it easy to test asynchronous flows, as they can be run step-by-step and checked for expected behavior.
3. Error handling: Redux Saga provides a comprehensive error handling mechanism, which allows developers to handle errors in a more graceful way and avoid breaking the application.
4. Advanced features: Redux Saga provides more advanced features, such as canceling and debouncing effects, which enable more complex asynchronous flows to be handled more easily.

Overall, Redux Saga is a powerful tool for managing side effects in a Redux application. It provides a declarative and composable way to handle side effects, making the code more modular and testable. Redux Saga's generator-based approach also makes it easier to handle complex asynchronous flows, which can be difficult to manage with traditional callbacks or promises.

- Redux Tool Kit

Redux Toolkit is a package that simplifies the process of writing Redux code. It provides a set of tools and utilities that help developers write cleaner and more concise Redux code, while also improving the performance and maintainability of the code.

Some of the key features of Redux Toolkit include:

1. Simplified Redux setup: Redux Toolkit provides a simpler way to set up a Redux store, reducing boilerplate and improving the overall readability of the code.
2. Immutable updates: Redux Toolkit makes it easy to create immutable updates to the store, which helps to avoid common pitfalls and improve the overall performance of the application.
3. Simplified action creation: Redux Toolkit includes a set of utilities for creating actions, making it easier to define and manage actions within the application.
4. Thunk integration: Redux Toolkit includes built-in support for thunks, which simplifies the process of writing and managing asynchronous code within the application.
5. DevTools integration: Redux Toolkit integrates with the Redux DevTools extension, providing a more powerful debugging experience for developers.

Overall, Redux Toolkit is a powerful tool for simplifying the process of writing Redux code. It provides a set of utilities and tools that reduce boilerplate and improve the overall performance and maintainability of the code. By simplifying the process of writing Redux code, Redux Toolkit makes it easier for developers to build scalable and maintainable applications.

- Chakra UI

Chakra UI is a React component library that provides a set of accessible and customizable UI components for building web applications. It is designed to be easy to use, with a focus on accessibility and flexibility.

Some of the key features of Chakra UI include:

1. Accessibility: Chakra UI follows the WCAG 2.0 standards for accessibility and provides keyboard navigation, screen reader support, and other accessibility features out of the box.
2. Customizability: Chakra UI provides a theme system that allows developers to customize the look and feel of the components to fit their application's branding and design.
3. Responsiveness: Chakra UI components are designed to be responsive and work well on different screen sizes and devices.
4. Easy to use: Chakra UI provides a set of easy-to-use components with clear documentation and examples, making it easy for developers to get started with the library.
5. Extensibility: Chakra UI provides an API that allows developers to extend and modify the behavior of the components to fit their specific needs.

Overall, Chakra UI is a powerful and flexible React component library that provides a set of accessible and customizable UI components. It is designed to be easy to use and provides a great starting point for building modern web applications.

- Material UI

Material UI is a popular React component library that provides a set of pre-designed and customizable UI components based on Google's Material Design guidelines. It is built with the goal of providing a simple and intuitive way to create beautiful and responsive user interfaces.

Some of the key features of Material UI include:

1. Theming: Material UI provides a robust theming system that allows developers to customize the look and feel of the components to match their application's branding and design.
2. Responsive design: Material UI components are designed to be responsive and work well on different screen sizes and devices.
3. Accessibility: Material UI components follow the WCAG 2.0 standards for accessibility, providing keyboard navigation, screen reader support, and other accessibility features out of the box.
4. Easy to use: Material UI provides a set of easy-to-use components with clear documentation and examples, making it easy for developers to get started with the library.
5. Active community: Material UI has a large and active community of developers who contribute to the library and provide support for others.

Overall, Material UI is a powerful and flexible React component library that provides a set of customizable UI components based on Google's Material Design guidelines. It is designed to be easy to use and provides a great starting point for building modern web applications.

- Jest

Jest is a popular open-source JavaScript testing framework developed by Facebook. It is designed to be easy to use and provides a fast and reliable way to test JavaScript code, including React applications.

Some of the key features of Jest include:

1. Easy to set up: Jest is easy to set up and configure, with minimal setup required to get started with testing.
2. Fast and reliable: Jest is designed to be fast and reliable, with built-in features like parallel test execution and snapshot testing to make testing faster and more efficient.
3. Mocking: Jest provides built-in support for mocking and simulating JavaScript modules and functions, making it easier to isolate and test individual components.
4. Code coverage: Jest provides built-in code coverage reports, making it easier to track and improve code coverage for your applications.
5. Large community: Jest has a large and active community of developers who contribute to the framework and provide support for others.

Overall, Jest is a powerful and flexible testing framework that provides an easy-to-use and reliable way to test JavaScript code. It is well-suited for testing React applications and is widely used by developers around the world.

- Karma

Karma is a popular open-source JavaScript testing tool that is commonly used to run unit tests for web applications. It is designed to make it easy to set up and run tests in different browsers and environments, making it well-suited for testing web applications that need to be compatible with different browsers.

Some of the key features of Karma include:

1. Browser compatibility: Karma supports a wide range of browsers, making it easy to test applications across multiple browsers and environments.
2. Easy to use: Karma is easy to set up and configure, with minimal setup required to get started with testing.
3. Real-time testing: Karma provides real-time feedback on test results, allowing developers to quickly identify and fix issues.
4. Extensible: Karma is highly extensible, with support for a wide range of testing frameworks, including Mocha, Jasmine, and QUnit.
5. Large community: Karma has a large and active community of developers who contribute to the tool and provide support for others.

Overall, Karma is a powerful and flexible testing tool that provides an easy-to-use and reliable way to test web applications. It is widely used by developers around the world and is well-suited for testing web applications that need to be compatible with different browsers and environments.

- Jasmine

Jasmine is a popular behavior-driven development (BDD) testing framework for JavaScript. It provides a clean and expressive syntax for writing tests, and can be used for both front-end and back-end testing.

Jasmine's syntax is designed to be easy to read and write, and is based on the concept of "specs", which define a specific behavior or feature to be tested. Each spec consists of one or more "expectations", which are assertions that test a specific condition.

Here's an example of a simple Jasmine spec:

```
describe("Calculator", function() {  
  it("should add two numbers together", function() {  
    var calculator = new Calculator();  
    var result = calculator.add(2, 3);  
    expect(result).toEqual(5);  
  });  
});
```

In this example, we define a spec for a "Calculator" object that tests the "add" method. We create a new Calculator object, call the "add" method with the arguments 2 and 3, and then use the "expect" function to assert that the result is equal to 5.

Jasmine provides a rich set of matchers and other testing utilities that make it easy to write comprehensive and readable tests. It can also be easily integrated with other JavaScript tools and frameworks, such as Karma and Protractor.

Overall, Jasmine is a powerful and flexible testing framework that can help developers write better, more reliable JavaScript code.

- Cypress

Cypress is a popular open-source end-to-end testing framework for web applications. It is built on top of JavaScript and uses a powerful architecture that allows it to run tests in a real browser, interact with the application under test, and make assertions about its behavior.

Cypress comes with several features that make it a popular choice for developers:

1. Real-Time Reloads: Cypress watches changes in the code and automatically reloads the application whenever a change is detected.

2. Automatic Waiting: Cypress automatically waits for elements to become available and animations to complete, ensuring that tests are not flaky.
3. Time Travel: Cypress provides a unique feature called "time travel" that allows developers to see what happened at any point during the test, making it easier to debug issues.
4. Consistent Results: Cypress runs tests in the same browser environment every time, ensuring consistent and reliable results.
5. Easy Debugging: Cypress has built-in debugging tools that make it easy to troubleshoot test failures.
6. Cross-Browser Testing: Cypress can be configured to run tests in multiple browsers, making it easy to ensure that an application works correctly across different browsers.

Cypress is easy to install and use and provides a powerful and flexible testing solution for web applications.

- SOAP

SOAP stands for Simple Object Access Protocol. It is a messaging protocol used for exchanging structured information between networked devices over the Internet.

SOAP is a protocol that uses XML (Extensible Markup Language) as its message format and typically operates over HTTP or HTTPS. It is commonly used in web services to facilitate communication between different software systems, allowing them to share data and functionality.

SOAP messages are made up of an envelope, which describes the message structure, a header, which contains additional information such as authentication details, and a body, which contains the actual message content.

SOAP is platform and language-independent, meaning it can be used by applications written in any programming language and running on any platform that supports HTTP or HTTPS.

- DAO(Data Access Object) pattern

The Data Access Object (DAO) pattern is a design pattern used in software engineering that provides a way to separate the data access logic from the business logic. It is typically used in applications that have a complex data model, where data is stored in a relational database or other persistent storage.

The DAO pattern involves creating an interface that defines a set of operations that can be performed on the data, such as create, read, update, and delete. This interface is then implemented by one or more classes that provide the actual implementation of the operations.

By separating the data access logic from the business logic, the DAO pattern allows for easier maintenance and scalability of the code. It also makes it possible to switch to a different data storage mechanism without affecting the rest of the application.

Overall, the DAO pattern promotes modularity and flexibility in software design, making it a popular choice for many applications that require complex data access logic.

- Jenkins

Jenkins is an open-source automation server that is used for building, testing, and deploying software applications. It is a highly popular tool for continuous integration and continuous delivery (CI/CD), and is widely used in DevOps environments.

Jenkins can be used to automate various aspects of the software development process, including building and testing code, deploying applications, and managing infrastructure. It supports a wide range of programming languages, build tools, and testing frameworks, making it highly versatile.

Jenkins operates as a web application and can be accessed through a web browser or a command-line interface. It has a large community of users and developers who contribute to its development, create plugins, and share knowledge and best practices.

Some of the key features of Jenkins include:

- Easy installation and configuration
- Support for a wide range of plugins and integrations
- Flexible and customizable build pipelines
- Distributed build and test support
- Security and access control
- Extensive logging and reporting capabilities

- Django ORM

Django ORM (Object-Relational Mapping) is a module in the Django web framework that provides a high-level, Pythonic interface for interacting with relational databases. It allows developers to interact with databases using Python code rather than writing SQL queries directly, making database interactions more intuitive and easier to manage.

The Django ORM module supports various database engines such as PostgreSQL, MySQL, SQLite, and Oracle. It uses a model-based approach where developers define classes that represent database tables and define fields on those classes that map to specific columns in the database.

Some of the key features of the Django ORM module include:

- Automatic creation of database tables based on model definitions
- Automatic management of database transactions
- QuerySet API for building complex database queries using Python code
- Support for aggregation, filtering, and sorting of query results
- Support for database relationships such as one-to-many and many-to-many
- Support for database migrations to manage changes to the database schema over time
- Integration with the Django web framework, including support for form validation and serialization of data to and from the database

Overall, the Django ORM module simplifies database interactions in Python web applications and is a popular choice for many developers.

- PyQt

PyQt is a set of Python bindings for the Qt application framework. Qt is a popular cross-platform application framework that is used to build graphical user interfaces (GUIs) for desktop, mobile, and embedded applications. PyQt provides Python developers with access to the Qt framework, allowing them to build native-quality GUI applications using Python code.

PyQt supports both Python 2 and Python 3 and can be used on various platforms including Windows, macOS, Linux, and Android. It provides a rich set of GUI widgets, graphics libraries, and multimedia tools, making it a versatile choice for building complex and feature-rich applications.

- Ag-Grid Community

Ag-Grid Community is an open-source JavaScript library that provides a powerful grid and data table solution for web applications. It is designed to be highly flexible and customizable, allowing developers to create complex data grids with advanced features and functionality.

Ag-Grid Community supports a wide range of features, including:

- Support for large datasets and virtual scrolling to improve performance
- Column resizing, filtering, and sorting
- Support for complex cell editors, including inline editing and popup editors
- Cell styling and formatting using CSS or custom renderers
- Built-in support for a wide range of data sources, including RESTful APIs and local data
- Localization and internationalization support
- Extensive documentation and community support

Ag-Grid Community is free and open-source, and can be used in both commercial and non-commercial applications. It is compatible with a wide range of JavaScript frameworks, including Angular, React, and Vue.js, and can be easily integrated into any web application.

- Apache Kafka

Apache Kafka is an open-source distributed event streaming platform that is used for building real-time data pipelines and streaming applications. It was originally developed at LinkedIn, and is now maintained by the Apache Software Foundation.

Kafka provides a high-throughput, low-latency platform for processing and streaming large amounts of data in real-time. It uses a publish-subscribe model, where data producers publish messages to topics, and data consumers subscribe to those topics to receive the messages.

Kafka is designed to be highly scalable, fault-tolerant, and distributed, with support for running on clusters of machines. It provides a number of key features, including:

- High throughput and low latency for processing large volumes of data
- Scalability and fault-tolerance through replication and partitioning of data across clusters
- A simple, efficient, and durable messaging system that can handle large amounts of data
- Support for multiple clients and languages, including Java, Python, and C++
- Integration with other data processing and storage systems, such as Hadoop and Spark
- Support for real-time stream processing and analytics through Kafka Streams and other stream processing frameworks

Overall, Apache Kafka is a powerful and versatile platform for building real-time data pipelines and stream processing applications, and is widely used in industries such as finance, healthcare, and e-commerce.

- How to implement Video Streaming in React using Kafka?

To implement video streaming in React using Kafka, you can use the Kafka API to create a producer that streams video data to a Kafka topic, and then use a Kafka consumer to consume and display the video data in your React application.

Here are the steps you can follow:

1. Create a Kafka producer that streams video data to a Kafka topic. You can use a Kafka client library for your preferred programming language to implement the producer.
2. In your React application, create a Kafka consumer that subscribes to the video stream topic and displays the video data in a video player component.
3. Render the VideoPlayer component in your React application to display the video stream.

- RDBMS

RDBMS stands for Relational Database Management System. It is a type of database management system that stores data in a structured manner using tables, columns, and rows. In an RDBMS, data is organized into tables, where each table represents an entity or concept in the real world. The relationships between tables are defined through the use of keys, which are used to link data between tables.

RDBMSs are widely used in many applications, including web applications, enterprise software, and data analytics. Some popular examples of RDBMSs include Oracle, MySQL, Microsoft SQL Server, PostgreSQL, and SQLite.

The benefits of using an RDBMS include:

Structured data: RDBMSs enforce structure on data, which makes it easier to organize and manage large amounts of data.

Scalability: RDBMSs can handle large volumes of data and can be scaled up or down depending on the needs of the application.

Security: RDBMSs provide robust security features to protect data from unauthorized access.

Data consistency: RDBMSs enforce consistency rules to ensure that data remains accurate and consistent across the system.

Data integrity: RDBMSs enforce referential integrity constraints to ensure that data relationships are maintained and that data is not lost or corrupted.

Overall, RDBMSs are a powerful tool for managing and organizing large amounts of data in a structured manner, and are widely used in many different types of applications.

- Amazon EC2 server

Amazon EC2 (Elastic Compute Cloud) is a web service offered by Amazon Web Services (AWS) that provides scalable computing capacity in the cloud. EC2 allows users to launch and manage virtual machines (called instances) in the AWS cloud.

EC2 instances are available in different sizes and configurations, and users can choose the instance type that best fits their needs. EC2 instances can be used to run applications, host websites, process data, or perform other computing tasks.

Why are you leaving your current job?

It was great experience with working with [name] company

Your company appears to fit well with the direction that I would like to take my career. I have researched and well-known great things about your company.

I am really excited to develop my expertise further with this opportunity.

What is most important to you in your next position?

Three things that I'm looking for in my next job are a collaborative, team-focused culture, opportunities to learn and grow my skills from a technical standpoint, and a chance to learn more leadership skills over time. After reading the job description for this role, it sounds like this could be a good match for what I'm looking for, so I was excited to come interview for the position and learn more.

What are you looking for in the next jobs?

Three things that I'm looking for in my next job are a collaborative, team-focused culture, opportunities to learn and grow my skills from a technical standpoint, and a chance to learn more leadership skills over time. After reading the job description for this role, it sounds like this could be a good match for what I'm looking for, so I was excited to come interview for the position and learn more.

Why Our Company

For many years, I have worked in a Front-end developer role with big and medium clients and companies. That's why I have proven experience in frontend development especially React and Typescript. You are now looking for a professional react/typescript developer, that's why I applied for your job.

What is your availability?

I'm flexible and available just about any time you need me to work. I am simply looking forward to joining the team and helping whenever I am most needed.

Why do you think that you will be successful in this job?

As my resume reflects, I have been successful at each of my previous places of employment. Given my research about your company, the job description outlined, and the information we've exchanged today, I believe I have the skills and experience to fulfill what you're looking for, and I'm eager to contribute as an employee.

What are your career objectives?

My career goal right now is to take the leadership and excellent track record I've obtained over the past decade and make the client's ideas real and grow them into a Fortune-500 company.

Do you consider yourself successful?

Yes, I do consider myself successful. I think my willingness to take on new challenges and work hard sets me up for success. For example, I volunteered to manage a project for my company, which involved managing 20 staff members. I had never managed such a large staff before. However, due to my hard work, effective communication, and clear goals, I effectively managed the team, and we completed our project ahead of schedule. I don't shy away from a challenge, and I know this would set me up for success at your company.

What is your greatest strength?

I have a solid work ethic. When I'm working on a project, I don't just want to meet deadlines. Rather, I prefer to complete the project well ahead of schedule. Last year, I even earned a bonus for completing planned products successfully one month ahead of time.

Asking about Company

Where are you located in?

Can you please explain me about Hiring Process? (How many steps in your Hiring Process?)

What does work/life balance look like at your company?

What kind of Industry do you wanna build? (What kind of Industry do you provide?)

How many developers are there in your team?

Can you provide me clear specifications and designs? Using Figma or Wireframe?

What is your provided compensation? (What is the range of Compensation?)

Don't you mind if you explain about compensation?

What is the Job Type? Full-time or Contract? (Are you seeking full-time developer or contractors only?)

What benefits do you provide for this job position?

What gets you excited about coming to work?

Yeah. Hmm.

The first exciting thing is that I will work on what I love and what I'm passionate about.

Top trending technologies like AI, TypeScript, React, Next.js. and so on.

And the next thing is that I will work with people I love working with, I mean you guys. I took some time to familiarize myself with you, I mean, your team and your company. I have noticed that all of you are very kind and have rich experience and skills in your field.

And finally, as same as anyone, it is that I will work for the amount of money I love. Haha. You're right. As you mentioned in your job description, a competitive salary is another exciting thing that I can get. Yup. That's all.

What was the last really great book you read?

Let me think, maybe, I think it's 'Harry Potter', a really exciting book. I like fantasy movies and books. My favorite fantasy movie is 'Naruto'.

Do you know Naruto Uzumaki? I remember some of his phrases like... "My friends were the first to accept me for who I am" "Failing doesn't give you a reason to give up, as long as you believe." something like that.

What surprises people about you?

-As I said before, I really like to work. I spend almost all my time working with challenging projects and learning new technologies.

-What else?

-Yeah, I am a night owl. lol. At SAP, I also worked as a professional remote worker in various global remote agile teams for foreign companies based in the US, UK, Germany, and so on. And the other reason is, you know, at night, there's nobody who bothers me, so I can all-in my time to think and work.

-Something else?

-Let me think. Yeah. The most important thing is writing 100% elegant and optimized code, provide 100% product. That's me.

If you were going to start your own business, what would it be?

Yeah, it's one of my dreams. I'd love to build my own IT company. I'd like to start my career again step by step, software developer, project manager, CTO, and finally CEO of my own company. Yeah. It is just my dream. haha. Yup. That's one of the reasons why I moved to Hong Kong from Japan. But unfortunately, because of a family emergency, I went back to here, Tokyo again. If things go better, I will move to Hong Kong again asap.

What's the biggest problem in most offices today?

As you know, pandemic. So everyone prefers remote work at home. I hope that everyone is safe.

What did you like most/least about your last company?

Hmm. maybe, working in a global agile team with different developers from all over the world is the most thing that I like at SAP. I worked as a senior software engineer and professional remote worker as I said before, so I have been given the opportunity to take part in various teams with no racism, no levels, no misunderstandings.

I never thought about the least thing at SAP. Everything was okay. Btw, I moved to Hong Kong to achieve my dream.

Where/when/how do you do your best work?

Haha. Anywhere, anytime, anyway you want.

As I said before, I always prefer 100% complete work. So I put great importance on listening carefully to the requirements and details. And also with a professional vision, I always do my best to deliver exceptional results on time and keep the team members updated with the project progress and any blockers to resolve through discussion.

When was the last time you made a big mistake at work?

Mistake... I didn't make any mistake till now. But if I must say about this, maybe, it is when I did not meet the timeline that the client wants because there are some misunderstandings between us. And anything else... I think there's no other mistake.

How could a manager best support you?

-Hmm. When I work as a software engineer at SAP, the team leader in a team gave me recognition and praise. And also he provided feedback, mentorship, and training.

-What else?

-Maybe, providing strong leadership and a clear vision is another main point. Yeah. That's all.

How do you handle stress or tight deadlines?

Stress... as you know, meeting the deadline is very important for us, I mean, software engineers. But, another important thing is the quality of production. I prefer both of us, but put more importance on quality.

So if the deadline is too tight, I explain the reason why I can not meet the deadline and why the quality is important, what else, how we can find an optimized way to build a perfect production. And also discuss it with teammates.

What are your plans for the next five years?

I'd like to be a project manager, team leader, or CTO for the next five years.

What three things do you need to succeed in this position?

Maybe, rich experience, communication skills, and the last thing is challenging to new things.

What kind of events do you attend outside of work hours?

Before the pandemic, I used to take part in some meetings to exchange experiences with each other and also I like a weekend party. But as you know, nowadays, we should avoid going outside of the home.

so in my free time, I enjoy fitness training and listening to music. I find that music is a wonderful creative outlet and stress-reliever, as well as a perfect balance for my demanding career. It allows me to set personal goals and achieve them, which is also true of physical fitness.

I've found that keeping my mind and body sharp improves every single facet of my life. I like cartoon songs like 'Flying to your heart' in TinkerBell, 'Let it go' in Frozen, 'Try Everything' in Zootopia, and so on.

What blogs or websites do you visit regularly?

Every day, I visit Twitter and I love to twit the phrases about programming.

How would you describe your group of friends?

Yeah. In one word, 'AWESOME'. We love to exchange the experience with each other and help. You know, if someone has any problems with work or life and he asks us to help, we do all our best to help him. That's our friendship and our property.

What do you do for fun?

Playing games. Watching cartoons.

What motivates you to do your best work?

Maybe, work ethic.

How do you prefer to communicate with coworkers?

I believe communication is the cornerstone of the success of a client-provided project. This is why I always emphasize the importance of listening carefully to the requirements.

In this way, I make sure my deliverance is more than satisfactory. I used to communicate with my teammates using Slack and for the project management, we used Jira and Trello.

If there's something that I need the design from the designers, I am very familiar with Figma.

Who inspires you and why?

Myself, as a software engineer who loves new challenges, I always prefer to be the 1st in my favorite field, not the 2nd or 3rd.

So, I can say that love for new challenges myself inspires me to do my best.

What superpower will you bring to our company?

100% clean and optimized code and 100% perfect product.

What has been the greatest disappointment of your life to date?

My biggest disappointment is that I wasn't able to follow my dream of being a professional photographer when I was a kid. Even though I was disappointed at the time, I realize now that if I had taken that direction, I would not have achieved my current degrees and developed a career that I love.

What tools or apps allow you to work more efficiently?

I used to communicate with my teammates using Slack and for the project management, we used Jira and Trello. If there's something that I need the design from the designers, I am very familiar with Figma.

How do you manage conflict with coworkers?

Good point. As you know, managing conflicts is very important in teamwork. So in that case, I find the conflicts part and take some time to review it.

And after that, I discuss with my teammates who wrote that part to manage these conflicts. After deep discussion, we can handle the conflicts together.

What would be your ideal work schedule?

I would probably want to start at 8 AM and finish up around noon before lunch. And start again at 3 PM and finish the work around 7 PM. From 12 PM to 3 PM, I should handle the problem with my family, as I said before, I came here, Tokyo for handling a family emergency.

What are the four types of burn-down charts?

The four types of burn-down charts are the product burn-down chart, the sprint burn-down chart, the release burn-down chart, and the defect burn-down chart.

What interested you about the position?

As same as all programmers, I can list the three things that are very important for me. They are production, team, and technology. Since you are building wonderful products for customers, you have a great team of high-skilled developers, and the technology that you are using is one of my main technical skills, I am very interested in this opportunity and I think I can be a good fit for this position.

Why are you leaving your current position?

I am currently working as a part-time developer at Sidero Startup company. Since all tasks are almost done, I only work 5 hours per week. So a few days ago, I started looking for a new opportunity as a full-time position.

What are you looking for long term?

My long-term objectives are to become a member of the elite team and work on enterprise-level projects.

Where do you hope your career will take you in the next 5 years?

For the next 5 or 7 years, I would like to be a project manager or team leader of the team. And after 10 years, I would like to build my own IT company.

What do you want to do in the next few years?

While working on top-trending tech-stacks with high-skilled developers from all over the world, I would like to learn more about how to make my dream come true.

What gets you really excited in a job?

As I described before, three things - products, team, and tech-stack.

What do you do in your current position? What do you like about it? What do you not?

Currently, I am working as a senior full-stack developer(part-time) at Sidero Startup company. You know, this company is my friend's company. I helped him to grow his company and his business. And also I mentored some Junior developers while working with this company. All tasks are almost done.

Are you comfortable leading a team and have you before? Give us examples of if you have and/or if you were to.

I have some valuable experience in working as a team lead or project manager but for now, I would like to work as an individual developer since I noticed that managing other people is not my main point.

What's important to you? If you had to pick what your top 6 "values" are in life and at work, what are they and why?

I always follow the core values below.

- Growth: Eagerly learn new things. Have broad interests and be curious about the world. Strive for perfection. Share my knowledge with others.
- Responsibility: Take on uncomfortable tasks that you can handle. Don't be afraid of the risk associated with decision-making. Don't run away from mistakes. Quickly communicate problems.
- Commitment: Take the initiative. Care about the success of the team. Go beyond what is expected. Find passion in what you do.
- Openness: Be brave and question decisions that you don't agree with. Openly talk about problems. Dedicate time to helping others. Approach change positively.
- Delivering value: Recognize what has to be done well from the very beginning. Make complex things simple and know how to choose 20% of work that provides 80% of the results. Deliver high-quality solutions while keeping the picture in mind.
- Love: Do what you love and love what you do. No matter what you are working on, you should be very invested.

Do you prefer to be told exactly what to do, or do you prefer to work with little direction?

Following the core values, I can say that I prefer to be told exactly what to do.

How do you stay up to date on all the latest technologies?

I always try to keep my tech-stack up-to-date to be ready for any kind of development while reading articles on LinkedIn, Medium, Dev.to and so on.

Can you rate yourself for the top 5 technologies you are most familiar with - from 1-10? 1 being not at all, and 10 being you could write an O'Reilly book on the tech.

React - 10, React Native - 10, Vue - 8, Node - 9, GoLang - 8

Are you comfortable in all technologies and would you be willing to work across all technologies? Or, do you prefer to stay in one technology?

As I mentioned before, I always try to learn new technologies so right now I am learning Python and Tensorflow.

What technologies are you most comfortable with?

Since I am a JS guy so I am most comfortable with JS frameworks and libraries, especially React.

What would you say your strengths are, and your weaknesses?

I can say my strengths in one word - 100%. I keep the core values in my mind so I always try to provide 100% perfect production. Because of this, sometimes it takes more time to complete the task or ticket.

Give us an example of when someone told you to do something that was technically wrong and/or you disagreed with the architect. How do you deal with the situation?

As you know, communication is the cornerstone of the success of a project so I always put great importance on listening carefully to the requirements and details from the customers and teammates. After listening carefully, I will explain why it is impossible or why it is wrong with good examples in articles.

Here, the most important thing is respecting for each other.

What do you think the hardest part about working remotely is?

From my perspective, the hardest part and also the most important part is communicating actively.

How do you plan on working remotely?

Overlapping working hours with client's business hours - 6 or 8 hours.

What are some of your pet peeves about working with people?

I think a clear specification is one of the most important aspects to produce a well-organized and clean solution.

What if you become really frustrated with a teammate? What are you going to do?

While working in a team, I know that I can meet such kinds of situations. In that case, I first try my best to keep professionalism. I will listen to his/her explanation carefully first and will give him my thought in a professional and kind manner. I am sure that losing temper is not the right option in any case.

What are you looking for financially?

120K - 140K per year

Looking forward to hearing from you very soon.

Cultural Questions

Would you like to work overtime or odd hours?

I know that in the company being asked to work for an extended number of hours comes with a good reason, so I am ok with it. It an extra effort means I am doing something for the company, I'll be happy to do it.

What is more important to you: the money or the work?

- "Money is always important, but the work is most important for me."
- "I would say that work is more important. If we work and achieve Company goals then obviously money would follow. I believe work to be prior."
- "Work is more important for me. Working just for money may not be fulfilled if I don't feel satisfied with my job. My work makes me stay productive, and money would naturally come along well."
- "I think money probably matters to me about as much as it does to anyone. It's vital and necessary for us to live and prosper but, at the same time, it's not my single most important driving force. I believe that money is rewarded for work."

Why should we hire you?

"I believe that everyone starts with a beginning, I need a platform to prove my abilities and skills. I think your company is the right place to explore my abilities. I need to be a part of your growth. I will do my level best."

"I have a good experience in that particular field (field of your specialization), and I think my talents will be a big contribution to the continuing pursuit of excellence of your company."

Assume you are hired, then how long would you expect to work for us?

"I will do my best for the growth of your company as long as I have the career growth, job satisfaction, respect and a healthy environment, then I don't need to change my company."

"I will work with the company as long as my presence benefits the company and I get ample opportunity to grow and develop both professionally and monetarily."

"Everyone looks for a bright future, healthy work environment, good salary, job satisfaction and I am pretty sure that your company gives such things, so I don't need to change the company."

"I will work with the company as long as my presence benefits both the company and mine in parallel. So your company gains good results, and I can be in a good position to improve my skills."

How would you rate yourself on a scale of 1 to 10?

"I will rate myself 8 out of 10 because I would never like to think that there should be a room left for putting in more efforts. That thought will create an interest in learning the things. Thank you very much for giving me this wonderful opportunity."

"I will answer this question based on some parameters. As far as hard work is concerned, I will rate myself as 8 because there should always be a scope to increase our skills which will create an interest in learning the things. When it comes to creativity, I would like to rate myself as 9. In the past, I have designed banners and brochures which were appreciated by the clients. To talk about patience, I will tag myself with 6 because I am an entry-level professional. Same as personal life, even professional life needs more experience for more patience. That is probably why in most companies, senior management looks more patient than entry level or even middle level. Overall, I would rate myself as 8 on a scale of 1 to 10. "

What is your objective in life?

"My short-term goal is to work in a reputed organization like yours where I can enhance my technical skills and knowledge.

My long-term goal is to see the company at a topmost position where I am one of the people responsible for that."

"My goal is to become a successful person and make my family proud of me."

What are your greatest strengths

"I am highly organized. I always take notes, and I use a series of tools to help myself stay on top of deadlines. I like to keep a clean workspace and create a logical filing method so I'm always able to find what I need.

I find this increases efficiency and helps the rest of the team stay on track as well."

"My main strengths are the ability to use my initiative to take on challenges. I am always proactive at what I do, and that keeps my mind stimulated and focused."

"My greatest strengths would be my intelligence and thoughtfulness. I believe that in every work environment you need to process every step and be detailed in your work. "

"My time management skills are excellent, and I'm organized, efficient, and take pride in excelling at my work."

"My greatest strength is my ability to focus on my work. I'm not easily distracted, and this means that my performance is very high."

“My biggest strength is my Confidence. Apart from that, I am Hardworking, self-motivated with a positive attitude towards my career and my life. If I have a problem, then I think its an opportunity for me to express my ability. “

What is your greatest weakness

“In the past, I had great opportunities to work with startups. Sometimes, they had very tight deadlines because they had to show MVP or PoC to investors on time. But I usually strived for the perfect result and pay too much attention to detail and sometimes, it caused missing the deadline.”

"I used to be very disorganized, always forgetting assignments and birthdays. But I managed to work out a computerized system of to-do lists and reminders that keep me on top of everything."

"I am a straightforward person, and I cannot say no when someone asks me for help."

What are your hobbies?

My hobbies are dancing, Internet surfing, playing Chess, listening to music, watching the news channel. In my spare time, I like to read news on my phone and traveling to my hometown. Thank you for giving this opportunity to introduce myself.

Explain, how would you be an asset to this organization?

"To become an asset for an organization, we have to punctual, dedicated, quickly adapt of the environment and positive working attitude I have all of these qualities so I will prove an asset for this company."

"My skill in XYZ company is outstanding. I have earned a lot of certificates and awards from my past employers. As an employee, I can handle pressure with ease and can work with minimal supervision."

Would you lie for the company?

"It depends on the situation if my lie creates a positive impact on the company and It was useful for many people, then I will lie."

How do you get to know about our company?

I get to know about your company from several online websites.

What does success mean to you?

“If I feel I am making a difference working with a team of people to make a more profitable company. It is a success for me.”

“If I put a smile on someone face and make him happy, it is a success for me.”

Describe yourself in one word?

Original, genuine, logical, incredible, focused, curious, active, quick, balanced, achiever, etc.

What is the difference between confidence and overconfidence?

Confidence is based on facts and knowledge, and overconfidence is based on speculation. There is a small difference between confidence and overconfidence.

“Yes, I can do this work is self-confidence. But only I can do this work is overconfidence.”

“Confidence is an internal belief that I am a right person for this job and overconfidence is thought that I am only the right person for this job.”

Just imagine that you have enough money to retire right now. Would you?

"No sir, I don't think so. I am a professional, and I love my work, so there is no question to leave my work. Yes, it may be that I would take a break to spend quality time with my family."

What makes you angry?

"Sir, I am not a short-tempered person, but I feel a bit of annoyance when someone disturbs me in my work without a genuine reason. Although I am an even-tempered person, when I get angry, I try to channel my negative feeling in my work."

What was the most difficult decision you have made in your past life?

“After completing my graduation, the toughest decision is whether to go for higher studies or do a job. Then I chose the job because getting trained is better than educated and it was also the demand of that time.”

“My toughest decision was to take admission in B.tech. I belong to a middle-class family, and my father was not in favor of taking admission, but I convinced him, and today he is very happy.”

“Before some time when I had to choose between joining a group of employees protesting some issue, and staying away from the issue. I ended up being a mediator between our immediate supervisor and employees, and I am glad I made that decision because it all ended well and without further conflicts in the company.”

Why did you apply to this job? (Why should we hire you for this position?)

In my next position, I'd like to continue building my project management and leadership skills. I reviewed the job description and saw an emphasis on team leadership and project management, so it seemed like a great fit. And since I've already spent 3 years managing 7-figure client projects and leading a team of five people in my most recent company, I'm confident that I could get up to speed very quickly and start contributing results for you in this role.

Where do you see yourself in 5 years

My career goal is to be a staff engineer in next 5 years by growing architect and management skills.

How do you handle conflicts with team members?

When I faced with a conflict, I like to ask questions and understand my coworker's perspective. This helps keep the situation calm, helps them feel like they're being heard, and after this, I've found it's much easier to come to an agreement or compromise while both staying a lot calmer.

Why did you leave your last job

It was a contract role and worked on last position for 2 years. The project has been completed and I had to spend up to 5 hours a week.

What motivates you most

Learning new things, coming up with creative ideas to improve something, or make something new, analyzing complex data in order to draw clear and simple conclusions, working well as part of a team.

In your opinion, what are the principles of good software engineering? What are some basic principles everyone should follow?

I think one of the main principles of software engineering, and one I try to live by, is to keep things as simple as possible.

You're often already dealing with complex algorithms and design concerns, so no need to make things even more difficult with overly complicated, resource-heavy code.

Your code should be simple, lean and easy to read.

If you start there, the rest will follow.

What do you do outside of work

A lot of my spare time lately has been dedicated to teaching myself how to keep work-life balance.

Besides, I love watching football matches as well as playing the game with friends. My favorite football team is Real Madrid.

How do you work with non-technical clients? (Tell me about a time you had to work with difficult client)

First, I would try to understand the big picture of his/her requirements. And going into details, relate his business requirements to technical field and let the client understand the approach that I am going to vet. It is very important to understand the difference between me and the client (technician vs non-technician) and always try to keep consistent and assertive communication. After discussing the requirements, it would be good to draw development roadmap to the client and upon mutual agreement, start the work.

Sir, it's a career move. I have learned a lot from my last job, but now I am looking for new challenges to broaden my horizons and to gain a new skill-set.

What do you value most?

I value happiness. By keeping this value at the center of my life, I am able to easily make decisions in the best interest of my family, my business and myself. A happy family, a happy home and a happy work environment add up to a happy life. I value making a difference in life and living with integrity.

Do you have any experience in mentoring others? or onboarding juniors to your project?

Yes, in my last position, I organized the onboarding process of junior developers in the company. Introduced the general development flow, projects we work on, version and task management, etc. Also mentored their work for 6 months to ensure they meet the company standard and produce efficiency work results accordingly.

What are the various types of software maintenance?

Maintenance types are corrective, adaptive, perfective and preventive.

Corrective: This type of maintenance is used to remove the errors spotted by business users.

Adaptive: This maintenance activity is performed to check the changes made in the hardware and software environment.

Perfective: This type of maintenance is used to implement changes in existing or new user requirements

Preventive: This maintenance activity is performed to avoid any issues in future implementations.

What is the worst job you ever had?

If I had to pick the worst job I've ever had, it was probably back to my very early job, working as a volunteer for an insurance company. They often scheduled me to work immediately after class, which cut into the time I had to study, and they often had very tight deadline.

Tell me about a time you made a mistake (failure)

In my first job, my manager asked me to develop several forms of tracking projects. I said "yes" immediately because I wanted to deliver result and demonstrate my skill asap. But as I started working on the project, I realized that I didn't understand the overall goal. And the project ended up creating templates that didn't meet my manager's expectations.

My manager was disappointed in me. She told me that if I had asked some clarifying questions, I would have delivered better result. I admitted to my manager that I had made a mistake and learned that it doesn't make me look stupid when I ask questions and it's better to speak up quickly. That's how I've handled those situations ever since.

How would you describe yourself with one word?

I am highly organized.

I always take notes, and I use a series of tools to help myself stay on top of deadlines.

I like to keep a clean workspace and create a logical filing method so I'm always able to find what I need.

I find this increases efficiency and helps the rest of the team stay on track as well.

What salary (expectation) are you looking for?

I'm more interested in the role itself than the pay.

That said, I'd expect to be paid the appropriate range for this role, based on my 8 years of experience.

I also think a fair salary would bear in mind the cost of living here in Hong Kong.

What are you passionate about?

One area that I'm passionate about is self-development and self-improvement in my personal life. I'm always looking to challenge myself and learn new things. That's one reason I enjoy working in software engineering; I've learned great communication skills, architecting skills, and problem-solving skills that help me in my day-to-day work but also in my personal life in terms of confidence, communication, etc.

What makes you unique from other candidates

Working at a startup gave me an opportunity to understand the ins-and-outs of the industry, and to take on tasks I might not have at a larger company. I think this experience gives me a slight edge over other applicants.

How long will you stay with us?

As long as there's a lot of work to do, then I'd be happy to stay for a long time. Since I'm a busybody, I like to being productive most of the time.

Are you a team player? or do you want to work alone?

It depends on the task.

When it comes to brainstorming, teams produce great ideas with multiple input, and teams can highlight people's strengths.

But I certainly enjoy working on my tasks as an individual as well, since in many ways it takes the ability to work alone for the team to fully succeed.

What is your workflow process for sticking to a project timeline?

Time-management is one of my strengths. I will divide the project requirements into smaller chunks and schedule the work process according to them. Practically, I am a fan of Agile methodology which handles it pretty nicely

What would you do if you encountered unexpected difficulties on a project?

Rather than trying to find the solution by myself and spending much time on it, I will share it with the team and find the solution together.

What do you do when you sense a project is going to take longer than expected?

I let the team-lead/product-owners know about the possible delay in the product delivery with proper reasons. And find the possible solution together.

Tell about how you work under pressure

This is a story when I worked as a lead developer at a US company. We were an agile team and worked on sprint by sprint, which usually took 3 weeks to complete. One day, Scrum Master said that the client wanted the sprint delivered in 2 weeks. I met with our team and reviewed the calendar. We eliminated team meetings and shifted lower-priority tasks until the end of the 2 weeks period to add extra hours to our workweeks.

I challenged my team to complete the project within 14 days or left and as a reward promised two days of extra PTO time. Our team got the job done in 12 days.

Do you have any questions for me?

“Thank you for giving me this opportunity. After my overall performance till now if I got selected what I need to improve and if I'm not selected how can I succeed further. Can you give any advice sir?”

Well prepared questions. Very important moment to move forward or not.

1. What's the makeup of the team as far as experience? Am I going to be a mentor or will I be mentored?
2. What does this company value the most and how do you think my work for you will further these values?
3. What kinds of processes are in place to help me work collaboratively?
4. What should be the most challenging part of this position?
6. What's the most important thing I can do to help within the first 90 days of my employment?
7. Do you have any questions or concerns about my ability to perform this job?
8. When top performers leave the company why do they leave and where do they usually go?
9. What do you see in me? What are my strongest assets and possible weaknesses? Do you have any concerns that I need to clear up in order to be the top candidate?
10. Who are the “coolest” people on my team? What makes him or her “cool? Can I meet them?

What is the most inventive or innovative thing you have done? Describe something that was your idea, e.g., a process change, a product idea, a new metric, or a novel customer interface. It does not need to be something that is patented. Do not write about anything your current or previous employer would deem confidential information. Provide relevant context for us to understand the invention/innovation. What problem were you seeking to solve, and

what was the result? Why was it an important problem to solve? How did it make a difference to the business or organization?

I was committed to one project that develops the web platform which provides the document encrypting service when I was at Ceridian.

After launching the production successfully, we should decide how to provide the APIs to the customers. Most team members agreed to provide the API documentation since we have well-structured swagger documentation about the APIs. But I had another idea that can provide a more user-experienced and user-friendly interface.

The idea was to develop the node and python packages that customers can easily integrate into their projects and access the APIs using wrapper functions. Of course, it requires more effort and time to develop new packages. But as you know, we should always care about customers when we release in production, so that we can get more rating in the market. After careful consideration, the company had decided to go with my idea.

Finally, we can deliver an easy-to-use and qualified product to the customers. And we can get a bunch of subscribers and 5-star feedback at the same time.

What was the biggest mistake of your life?

I think the worst mistake I ever made at work was in my first ever job - five years ago now. A more senior member of the team seemed to take an instant dislike to me from the start, and one day she was particularly unpleasant to me in front of several colleagues.

Later on, I was talking to one of those colleagues who was, I thought, attempting to console me. Angry and hurt, I foolishly vented my feelings and told her what I thought of the lady in question. I was naturally shocked to find out that she went on to tell everyone what I had said and this certainly didn't help my relationship with the team member who was causing me problems.

Rather than let the situation carry on, I chose to have a quiet word with this lady to find out what her problem was with me and to see if we could put it behind us. It turned out it was nothing personal; she just resented the fact that a friend of hers had also been interviewed for my position and had been turned down. Once we had got matters out into the air, her behavior changed, and we got on quite well after that. However, I certainly learned a lot from experience. I learned that careful communication is vital in managing interpersonal relationships and that if I have a problem with someone, it's always best to talk it over with them rather than with someone else.

Describe the three things that are most important for you in a job?

“According to me, Honesty, Loyalty, and determination to achieve my team's target are the three important things in a job.”

“According to me, Professionalism, growth and a healthy work-life balance are the three important things in a job.”

What are your expectations from the company?

I have always wanted to work with an organization which provides a very comfortable and home like work environment. I would like to work in the company where I can get the opportunity to learn and enhance my skill to become a better professional in the future.

What gets you up in the morning?

It is my promise that gets up me in the morning. My promise is to learn something new and someone in need. It provides me the satisfaction that I am making a difference in someone life.

What is your favorite book?

This question is asked to judge your taste about reading books. The interviewer wants to know what types of book you like. Would you fit for the company culture?

Answer this question according to your sense, your knowledge about the book. Only named the books you have really read. You should choose something from a reputable author that your interviewer has probably heard of.

As you said, internet surfing is your hobby. Which site do you surf mostly?

This is your choice that which sites you surf most, but while answering this question always refers to sites which are relevant to your field of job. Don't take the name of social networking sites or other irrelevant sites.

React Questions

<https://beta.reactjs.org/apis/react/useMemo>

What are new hooks in react 18?

useTransition

`useTransition` and `startTransition` let you mark some state updates as not urgent. Other state updates are considered urgent by default. React will allow urgent state updates (for example, updating a text input) to interrupt non-urgent state updates (for example, rendering a list of search results).

useDeferredValue

`useDeferredValue` lets you defer re-rendering a non-urgent part of the tree. It is similar to debouncing, but has a few advantages compared to it. There is no fixed time delay, so React will attempt the deferred render right after the first render is reflected on the screen. The deferred render is interruptible and doesn't block user input.

useId

`useId` is a new hook for generating unique IDs on both the client and server, while avoiding hydration mismatches. It is primarily useful for component libraries integrating with accessibility APIs that require unique IDs. This solves an issue that already exists in React 17 and below, but it's even more important in React 18 because of how the new streaming server renderer delivers HTML out-of-order.

`useId` is **not** for generating keys in a list. Keys should be generated from your data.

useSyncExternalStore

`useSyncExternalStore` is a new hook that allows external stores to support concurrent reads by forcing updates to the store to be synchronous. It removes the need for `useEffect` when implementing subscriptions to external data sources, and is recommended for any library that integrates with state external to React.

`useSyncExternalStore` is intended to be used by libraries, not application code.

useInsertionEffect

`useInsertionEffect` is a new hook that allows CSS-in-JS libraries to address performance issues of injecting styles in render. Unless you've already built a CSS-in-JS library we don't expect you to ever use this. This hook will run after the DOM is mutated, but before layout effects read the new layout. This solves an issue that already exists in React 17 and below, but is even more important in React 18 because React yields to the browser during concurrent rendering, giving it a chance to recalculate layout.

`useInsertionEffect` is intended to be used by libraries, not application code.

What is `useEvent()` hook ?

`useEvent()` is a React Hook that lets you extract non-reactive Effect logic into an [Event function](#).

The `useEvent` Hook is **called with each render of the component where it is used**. With each render, the handler function is passed to the `useEvent` Hook. The handler function always has the latest values of props and state because it's essentially a new function when a component is rendered.

Using `useEvent()`, we can maintain the same referential identity even if props or state used within the function change.

`useMemo` vs `useCallback`?

`useCallback` and `useMemo` both expect a function and an array of dependencies. The difference is that **`useCallback` returns its function when the dependencies change while `useMemo` calls its function and returns the result**.

`useMemo` is a React Hook that lets you cache the result of a calculation between re-

```
const cachedValue = useMemo(calculateValue, dependencies)
```

- Usage
 - Skipping expensive recalculations
 - Skipping re-rendering of components
 - Memoizing a dependency of another Hook
 - Memoizing a function
- Reference
 - `useMemo(calculateValue, dependencies)`
- Troubleshooting
 - My calculation runs twice on every re-render
 - My `useMemo` call is supposed to return an object, but returns undefined
 - Every time my component renders, the calculation in `useMemo` re-runs
 - I need to call `useMemo` for each list item in a loop, but it's not allowed

renders.

`useCallback` is a React Hook that lets you cache a function definition between re-renders.

```
const cachedFn = useCallback(fn, dependencies)
```

- Usage
 - Skipping re-rendering of components
 - Updating state from a memoized callback
 - Preventing an Effect from firing too often
 - Optimizing a custom Hook
- Reference
 - `useCallback(fn, dependencies)`

What is `useDebounce()` hook?

This React hook helps to limit that the component is re-rendered too many times.

To prevent this, the debounce uses an internal timer to execute the callback function every xx seconds (2nd parameter).

The debounced value will only reflect the latest value when the `useDebounce` hook has not been called for the specified time period.

UseMemo vs React Memo?

memo is for when you find an existing component that is expensive to render, and you don't have the option to optimize it internally. Just wrap it and let React do an extra check. React. `useMemo` is for internally optimizing a component by saving the return value of an expensive function call.

Why can't we use async func as first parameter in `useEffect`?

The issue here is that the first argument of `useEffect` is supposed to be a function that returns either nothing (undefined) or a function (to clean up side effects). But an async function returns a Promise, which can't be called as a function! **It's simply not what the `useEffect` hook expects for its first argument.**

When to use `useCallback`, `useMemo` and `useEffect`?

The `useCallback`, `useMemo`, and `useEffect` are a way to optimize the performance of React-based applications between rerendering of components. These functions provide some of the features of the class-based components like persistence of dedicated states through

render calls as well as the lifecycle functions to control how the components look over various stages of their lifecycle.

To answer when to use `useCallback`, `useMemo`, and `useEffect`, we should know what exactly they do and how they are different.

1. [useCallback](#): The `useCallback` is a react hook that returns a memoized callback when passed a function and a list of dependencies as parameters. It's very useful when a component is passing a callback to its child component to prevent the rendering of the child component. It only changes the callback when one of its dependencies gets changed.
2. [useMemo](#): The `useMemo` is similar to `useCallback` hook as it accepts a function and a list of dependencies but it returns the memoized value returned by the passed function. It recalculated the value only when one of its dependencies change. It is useful to avoid expensive calculations on every render when the returned value is not going to change.
3. [useEffect](#): A hook that helps us to perform mutations, subscriptions, timers, logging, and other side effects after all the components has been rendered. The `useEffect` accepts a function that is imperative in nature and a list of dependencies. When its dependencies change it executes the passed function.

`useCallback`

`useCallback` is a react hook that returns **a memorized callback** when passed a function and a list of dependencies that set the parameters. It's useful when a component is passing a callback to its child component in order to prevent rendering. It only changes the callback when one of its dependencies is changed.

`useMemo`

`useMemo` is very similar to `useCallback`. It accepts a function and a list of dependencies, but the difference between `useMemo` and `useCallback` is that `useMemo` returns the **memoized value** returned by the passed function. It only recalculates the value when one of the dependencies changes. It's very useful if you want to avoid expensive calculations on every render when the returned value isn't changing.

`useEffect`

We won't talk about `useEffect` in detail, but it's worth noting. The `useEffect` accepts a function as a first parameter and a list of dependencies as a second parameter. When the dependencies change, it executes the passed function. The hook helps developers perform mutations, subscriptions, logging, and other side effects once all the components have been rendered.

useMemo is Not Recommended to Call Other Hooks

```
function memoUsed() {  
  const [state, setState] = useState(null)  
  
  const _ = useMemo(() => {  
    // cause infinite re-render  
    setState('value')  
    return 'insert JSX here'  
  })  
  
  return _  
}
```

```
function callbackUsed() {
  const [state, setState] = useState(null)

  const _ = useCallback(() => {
    // regular thing
    setState('value')
    return 'insert JSX here'
  })

  return _()
}
```

The Function Takes No Parameters in useMemo

```
function memoUsed() {
  const _ = useMemo((arg1) => {
    // React ignores arguments
    return 'insert JSX here'
  }, [])

  return _

function callbackUsed() {
  const _ = useCallback((what, where) => {
    // can be used inside functions
    return `insert ${what} ${where}`
  })

  return _('JSX', 'here')
}
```

3.

Regular function vs arrow function?

Regular functions created using function declarations or expressions are constructible and callable. Since regular functions are constructible, they can be called using the new keyword. However, the arrow functions are only callable and not constructible, i.e arrow functions can never be used as constructor functions.

- “This” keyword

“This” keyword is one of the most used keywords in Javascript.

In a regular function, “this” keyword changes according to the way that function is invoked.

But, in arrow functions don’t have their own “this”, and they don’t redefine the value of “this” within the function.

Regardless of how you execute arrow functions, “this” inside an arrow function always refers to “this” from the outer context. This means that “this” keyword is lexically bound in arrow functions

.- Constructors

Regular functions can easily construct objects with “new” keywords.

Arrow function cannot be used as a constructor

.- Argument object

Inside the body of a regular function, arguments is a special array-like object containing the list of arguments with which the function has been invoked.

On the other side, no arguments special keyword is defined inside an arrow function.

- Implicit return

Regular function: return expression statement returns the result from a function. If the return statement is missing, or there's no expression after the return statement, the regular function implicitly returns undefined.

Arrow function: also can return values from the arrow function but with one useful exception.

If the arrow function contains one expression, and you omit the function's curly braces, then the expression is implicitly returned.

What about Progressive Rendering?

Progressive Rendering (aka Progressive Server Side Rendering) is a technique in which once you render the critical content on the server, you start streaming it to the client without waiting for non-critical content. You then stream the non-critical content later once it's rendered on the server.

What is prop drilling?

Sometimes while developing React applications, there is a need to pass data from a component that is higher in the hierarchy to a component that is deeply nested.

To pass data between such components, we pass props from a source component and keep passing the prop to the next component in the hierarchy till we reach the deeply nested component.

The **disadvantage** of using prop drilling is that the components that should otherwise be not aware of the data have access to the data.

What is shadow root ?

Shadow tree – a hidden DOM subtree, not reflected in HTML, hidden from prying eyes. The shadow root, returned by attachShadow, is like an element: we can use innerHTML or DOM methods, such as append, to populate it.

The element with a shadow root is called a “shadow tree host”, and is available as the shadow root host property.

What is closure in JavaScript?

Closures are created whenever a variable that is defined outside the current scope is accessed from within some inner scope.

It gives you access to an outer function's scope from an inner function.

In JavaScript, closures are created every time a function is created.

To use a closure, simply define a function inside another function and expose it.

```
function numberGenerator() {
  // Local “free” variable that ends up within the closure
  var num = 1;
  function checkNumber() {
    console.log(num);
  }
  num++;
  return checkNumber;
}
```

```
var number = numberGenerator();
number(); // 2
```

```
const add = (function () {
  let counter = 0;
  return function () {counter += 1; return counter}
})();

add();
```

What about polyfills in JavaScript?

A polyfill is a piece of code that implements the features that you expect the browser to support natively.

Polyfills usually emulate a newer API that provides fallback functionality to older browsers.

Polyfills can also make a feature that different browsers implement differently work the same way in each browser.

What is difference State vs global context.

Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language.

For example, in the code below we manually thread through a “theme” prop in order to style the Button component: class App extends React.

But we can use the state within only the components where it is defined.

How do we build a good React Architecture?

- Avoid using a lot of nested files & folders.
- Reuse your components.
- Put CSS in JavaScript.
- Comment only where necessary.
- Use capitals for Component Names.
- Use React's Bit tool.
- Use snippet libraries.

What about solid design principles?

SOLID is an acronym that stands for five key design principles: single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle.

All five are commonly used by software engineers and provide some important benefits for developers.

The Liskov Substitution Principle (LSP) states that objects of a superclass should be replaceable with objects of its subclasses without breaking the application. In other words, what we want is to have the objects of our subclasses behaving the same way as the objects of our superclass.

What about microservices?

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs.

These services are owned by small, self-contained teams.

The services communicate with clients, and often each other, using lightweight protocols, often over messaging or HTTP. Why microservices are used?

Microservices allow developers to become language and technology agnostic.

Different team members can use different programming languages and technologies when debugging and coding.

Monitoring microservices lets you check your architecture for service and performance and identify future debugging problems.

What is middleware in Express.js?

Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named next.

These functions are used to modify req and res objects for tasks like parsing request bodies, adding response headers, etc.- An Express application can use the following types of middleware:

Application-level middleware

Router-level middleware

Error-handling middleware

Built-in middleware

Third-party middleware-Most used Middleware

CORS, CSRF, Auth- Advantages of using middleware:

Middleware can process request objects multiple times before the server works for that request.

Middleware can be used to add logging and authentication functionality.

Middleware improves client-side rendering performance.

Middleware is used for setting some specific HTTP headers.

Middleware helps for Optimization and better performance.

Complex data type vs Primitive data type?

A complex data type is usually a composite of other existing data types. For example, you might create a complex data type whose components include built-in types, opaque types, distinct types, or other complex types. The primitive data types are types like string , integer , boolean , etc. that can take a single value. The complex types consist of multiple values grouped together .

What is Const used for?

The const declaration creates a read-only reference to a value. It does not mean the value it holds is immutable—just that the variable identifier cannot be reassigned. For instance, in the case where the content is an object, this means the object's

What are async and await?

An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains

.- When should I use async await JavaScript?

Inside an async function, you can use the await keyword before a call to a function that returns a promise. This makes the code wait at that point until the promise is settled, at which point the fulfilled value of the promise is treated as a return value, or the rejected value is thrown

- What is difference between async and await in JavaScript?

The async keyword is used to define an asynchronous function, which returns a

AsyncFunction object. The await keyword is used to pause async function execution until a Promise is fulfilled, that is resolved or rejected, and to resume execution of the async function after fulfillment.

- How does async await work?

The async keyword turns a method into an async method, which allows you to use the await keyword in its body. When the await keyword is applied, it suspends the calling method and yields control back to its caller until the awaited task is complete. await can only be used inside an async method.

What is event loop?

Call-stack, task- queue

Event Loop has pretty specific work. It has responsibility to see whether the call-stack is empty and does the task queue contains pending task to process. If the call-stack is empty, it will push the task to the call-stack from the queue and the task gets processed.

CallStack is a data structure which keeps track of function calls in our program. Whenever we call a function for its execution, we are pushing it to the stack. It is popped out of the stack when the execution is completed.

Task Queue (Micro task Queue) is a JavaScript runtime messaging queue which handles task that is allocated by different Web Apis. This queue is dedicated to handle the Web Apis callbacks. The messages are processed once the call stack is clear.

<https://mathsgee.com/qna/3009/what-event-loop-what-difference-between-call-stack-task-queue>

Advantages of class component and function component?

- The advantages of Class Component.

- Make use of ES6 class and extend the Component class in react.
- It tends to implement logic and state because of that its called "smart" or "stateful" component.
- React life cycle methods can be used within the class component.
- Can pass props and can be access props by using this. props.

- The advantages of Functional Component.

- Easier to test: You don't have to worry about hidden state and there aren't as many side effects when it comes to functional components, so for every input, the functions will have exactly one output.
- Easier to read/write: The syntax is less complex than that of class components, and it's easier to read due to knowing how much you can't do with functional components already. The use of prop destructuring makes it really beneficial to see what's going on and what's coming out of the component.

- Easier to debug: Again, functional components depend on the props they are given and don't rely on state. There isn't a reason to `console.log()` your state to constantly understand the changes that are going on.

useCallback & Rest API?

`useCallback` is a React Hook that lets you cache a function definition between re-renders.

The `useCallback` hook is used when you have a component in which the child is rerendering again and again without need.

It passes an inline callback and an array of dependencies. `useCallback` returns a memoized version of the callback that only changes if one of the dependencies has changed. This is useful when passing callbacks to optimized child components that rely on reference equality to prevent unnecessary renders.

What is SSR(server side rendering)?

Server-side rendering (SSR) is an application's ability to convert HTML files on the server into a fully rendered HTML page for the client. The web browser submits a request for information from the server, which instantly responds by sending a fully rendered page to the client. Search engines can crawl and index content prior to delivery, which is beneficial for Search Engine Optimization purposes.

Popular examples of server-side rendering JavaScript frameworks include: Angular server side rendering, ejs server side rendering, server side rendering Express, Gatsby server side rendering, Google server side rendering, NestJS server side rendering, Next server side rendering, Nuxt server side rendering, React server side rendering, and Vue server side rendering.

-What are the Benefits of Server-Side Rendering?

Some server-side rendering advantages include:

A server-side rendered application enables pages to load faster, improving the user experience.

When rendering server-side, search engines can easily index and crawl content because the content can be rendered before the page is loaded, which is ideal for SEO.

Webpages are correctly indexed because web browsers prioritize web pages with faster load times.

Rendering server-side helps efficiently load webpages for users with slow internet connection or outdated devices.

-What are the Risks of Server-Side Rendering?

Server-side rendering disadvantages may include:

Rendering server-side can be costly and resource-intensive as it is not the default for JavaScript websites, and the server takes on the full burden of rendering content for users and bots.

While rendering static HTML server-side is efficient, rendering bigger, more complex applications server-side can increase load times due to the bottleneck.

Server-side rendering may not be compatible with third-party JavaScript code.

Rendering server-side may be ideal for static site generation, but frequent server requests and full page reloads can result in overall slower page rendering in more complex applications.

What is Hoisting in JavaScript?

In JavaScript, Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution. Basically, it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local

Tree shaking in JavaScript?

Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code. It relies on the import and export statements to detect if code modules are exported and imported for use between JavaScript files.

Tree shaking, also known as dead code elimination, is the practice of removing unused code in your production build. It's important to ship as little code to your end-users as possible. By statically analyzing our source code, we can determine what's not being used and exclude it from our final bundle.

What is TDD?

Test Driven Development (TDD) is **a software development practice that focuses on creating unit test cases before developing the actual code.** It is an iterative approach that combines programming, the creation of unit tests, and refactoring.

Fewer bugs and errors are the primary benefit of the TDD approach. When the code has fewer bugs, you'll spend less time fixing them than other programming methodologies. TDD produces a higher overall test coverage and, therefore, a better quality of the final product.

What is BDD?

Behavior-driven development (BDD) is an Agile software development methodology in which an application is documented and designed around the behavior a user expects to experience when interacting with it.

The BDD process moves through three phases—**discovery, formulation, and automation**—where the acceptance criteria are transformed into acceptance tests that are later automated.

What is Web worker?

A web worker is a JavaScript that runs in the background, independently of other scripts that may also have been executed from the same HTML page, without affecting the performance of the page.

We can spawn as many workers as we wish. We can pass data to the script being executed in the worker threads and also return value to the main thread upon completion.

Why React is library not framework?

Because it is just a library for views and not complete, in the sense that if you need to add routes to your app, you need to install a package, if you need to make http requests without the native fetch API, you need to install another library, frameworks like Angular comes with all these internals,

Why React?

React is officially defined as a “Javascript library for building user interfaces.” Basically, the idea is to build web applications with rich UI elements. It facilitates smoother communication between developers and web page DOMs (Document Object Models), therefore eliminating the need for the DOM API.

Developers achieve this via composable UIs which are more efficient than traditional ideas of operating the backend of web applications. Using the Model-View-Controller (MVC) architectural model, React represents the ‘View’ section.

State vs Props in React?

Properties (props) are read-only. They determine the configuration of a component as received from the parent. If React components are functions, you may think of props as arguments for a function.

Props remain immutable throughout the lifecycle of a component.

On the other hand, the state is dynamic. Components don’t have a default state passed down from parents. Hence, the state of a component is mutable, representing a data value that can be updateable when certain conditions are fulfilled. Thus, the state is essentially what makes dynamic rendering possible.

What are components?

Components are the main building blocks.

When building an app, the entire app is divided into components and writing code instructs the components on interaction and so on. Each component is a functional UI element that forms a building block of an application. Further, the two main features of components; they are:

- **Composable** – can be reused and new components can be iterated from former ones.
- **Independent** – each component, though linked with others, still stands apart. So, when you fix the code in one section, you still preserve the rest...

What are the main features of React?

- **JSX**: JSX is a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code.
- **Components**: Components are the building blocks of any React application, and a single app usually consists of multiple components. It splits the user interface into independent, reusable parts that can be processed separately.
- **Virtual DOM**: React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM. When the state of an object changes, virtual DOM changes only that object in the real DOM, rather than updating all the objects.
- **One-way data-binding**: React's one-way data binding keeps everything modular and fast. A unidirectional data flow means that when designing a React app, you often nest child components within parent components.
- **High performance**: React updates only those components that have changed, rather than updating all the components at once. This results in much faster web applications.

What are the advantages of React?

- **Flexibility**: it allows developers to build upon any desired technology stack. Therefore, unlike other frameworks, developers are free of restrictions to rigid architectures and workflows.
- **Performance optimization**: declarative programming enhances performance by creating a virtual DOM by allowing the developer to write HTML directly into JavaScript files. The virtual DOM is much easier to manipulate than the real DOM and so that improves efficiency. Answers to ReactJS coding interview questions such as this should explain how it enables building rich user interfaces for mobile and web applications.
- **SEO Efficiency**: it is used for building web applications; therefore, SEO is a strong factor in considering it. Thankfully, it is SEO-friendly. This is despite the fact that search engines commonly have problems reading apps that are heavy on JavaScript. The virtual DOM feature

makes this less of a challenge. This may also feature in the React JS advanced interview questions and answers section.

- **Productivity Improvement:** developers commonly describe as the best feature the ability to reuse system components. This helps them to deal with issues concerning the library's complex logic. For companies, the ability to reuse code assets helps in setting internal coding standards and app consistency across the board.
- **Learn once – write anywhere:** while React is popular for web development, there is the React Native framework, which enables building web applications. Again, this improves cross-platform consistency and efficiency. It also improves the work of developers, since, for all platforms, they are working with similar tools. This is an important part of React frontend developer interview questions.

What is the difference between ReactJS and React Native?

In simple terms, ReactJS is a library, while React Native is a framework. Specifically, the latter is a mobile framework that enables cross-platform mobile development. It is commonly used to build apps meant for Android and iOS. Apps that use it include Skype, Pinterest, Uber Eats, etc.

On the other hand, ReactJS is a library for building the user interface (UI) of web applications. Keep in mind that when people simply use 'React', they are most often referring to ReactJS. The 'native' factor stems from the fact that it uses native APIs to render components on the app.

What are the differences between React and AngularJS?

First, while AngularJS is a JavaScript framework, React is a JavaScript library. Therefore, because of the concept of inversion of control, the latter enables developers to have a greater charge of the application flow.

AngularJS uses an actual DOM while React uses the more recent Virtual DOM technology. Therefore, the latter is more suitable for applications requiring dynamic content and regular updates. That's why some popular apps that use React include Facebook, Netflix, WhatsApp, Discord, etc. More so, a smaller library size makes its apps more lightweight than AngularJS.

AngularJS uses a two-way binding approach, which seems easier at first glance, but it is inefficient in the face of code-heavy applications. Its one-way binding allows developers to efficiently manage large code.

What are React Component life cycles?

The three main phases of a lifecycle: mounting, updating, and unmounting. Each features different methods, although they can be overridden by the programmer.

- **Mounting:** this happens when elements are entered into the DOM. That is when the component is initially created. The most common methods here are constructor(), render(), and componentDidMount().

- **Updating:** when one triggers a change to state, the component is re-rendered. Common methods are `render()` and `componentDidUpdate()`.
- **Unmounting:** this is removing a component. The `componentWillUnmount()` method applies here.

Class component VS functional component?

The preeminent difference between both types of components is syntax. A functional component accepts properties in plain JavaScript functions and returns HTML in the form of JSX elements. As such, they are also known as stateless components. More so, they do not use any render method.

However, class components, also called stateful components, create render functions that extend React's component classes. While the former is straightforward and easy to read, class components use complex UI logic. Moreover, functional components do not support lifecycle hooks though class components do.

Explain Controlled vs Uncontrolled Components in React?

One can have a form handled by a React component or alternatively by the DOM directly. Forms implemented in the former manner are called controlled components, while for the latter, they are uncontrolled components.

Uncontrolled components are similar to traditional HTML in that the state is stored internally, although you can still retrieve the current value from the DOM by using a ref. Moreover, controlled components maintain their internal state and allow for validation control, unlike uncontrolled components, which do neither. However, React mostly recommends controlled components for implementing forms.

How do you render elements in React?

In React, elements are the 'smallest building blocks' of apps. Elements are the plain objects returned from components and rendered into the DOM. Elements determine what you see on the browser screen but they themselves are not seen; rather they are stored in the app's memory.

Therefore, every component has an element that determines the properties, type, etc. of that component. Basically, one renders elements into the root DOM node by using `ReactDOM.render()`. In addition, since React elements cannot be changed neither can their attributes be updated, one has to use the same code function to create a new element in order to implement an update to the UI.

What is the difference between props and state?

Properties (props) are read-only. They determine the configuration of a component as received from the parent. If React components are functions, you may think of props as arguments for a function. Props remain immutable throughout the lifecycle of a component.

On the other hand, the state is dynamic. Components don't have a default state passed down from parents. Hence, the state of a component is mutable, representing a data value that can be updateable

when certain conditions are fulfilled. Thus, the state is essentially what makes dynamic rendering possible.

What are some lifecycle methods of React components?

- **render()** – the most commonly used lifecycle method and the only one required in a class component.
- **constructor()** – a method called before mounting a component. This method is not necessary if you didn't initialize state yet or bind methods.
- **componentDidMount()** – the first method called after a component is mounted.
- **componentDidUpdate()** – this method is implemented once a component is updated, though not at initial rendering.
- **componentWillUnmount()** – to be implemented before you unmount a component is unmounted and also used to perform due cleanups.

Those are the common lifecycle methods. Some are rarer, including `shouldComponentUpdate()`, `static getDerivedStateFromProps()`, `getSnapshotBeforeUpdate()`, `static getDerivedStateFromError()`, and so on. This question may also easily fall under ReactJS experienced interview questions for senior developers if you need to establish their understanding of components.

How does Virtual DOM Work?

React allows developers to write HTML into JavaScript files. This empowers developers to maintain control of their HTML code. The Document Object Model (DOM) is an API that works with HTML and XML files. Therefore, it is very useful in building web applications. However, React deviates from the traditional approach to using DOM.

With a virtual DOM, one can keep a representation of the DOM in memory. Thus, virtual DOM is lightweight than the real DOM and does not demand as many resources to maintain. However, through the process of reconciliation, the virtual DOM may be synced with the real DOM via a library.

Unlike a real DOM, one can use a virtual DOM to directly interact with the interface, even though it shares properties with a real DOM.

Many JavaScript framework operations require rapid updates to the DOM. This can slow down the application a lot. However, with virtual DOM, the time cost is greatly reduced.

What is Higher-Order Component (HOC)?

Higher-Order Components are used to add sophisticated functionalities to existing components. The purpose of the HOC technique, which is common in third-party React libraries, is to reuse component logic.

Therefore, the use of every Higher-Order component function returns a new component. HOCs in React are related to higher-order functions in JavaScript. Higher-order components find implementation in Redux connect (which is a higher-order function), state abstraction, and so on.

How does conditional rendering work in React?

Conditional rendering is the dynamic output of UI markups depending on the state of a condition. It works similarly to JavaScript conditions. With conditional rendering, one can toggle certain application functions, determine permission levels, show or hide elements, handle authentication, render API data, and so on.

There are various ways to implement conditional rendering in React. Some of the expected answers for this particular one of the React questions for senior developer candidates are:

- Using the **if-else conditional logic** (suitable for smaller and medium-sized apps)
- Using **element variables**, which actually enables writing cleaner code
- Using **ternary operators**, which take some complication away from if-else statements, etc.

What are the drawbacks of React?

Interview questions on ReactJS for experienced developers such as this help you to determine that a person knows what they are doing. For someone to claim expertise over a platform, they need to be well acquainted with its limitations and challenges, so as to know how to get around them.

JSX: although it is well-established that JSX has its benefits such as enhancing app performance, developers find it difficult to learn and use. Many developers don't have JSX foundation as a syntax extension. Hence, they find it complex.

The rapid pace of development in React can also be a disadvantage because it means developers have to continually relearn new concepts as updates are released. It also makes it difficult to keep up with documentation if updates always result in significant changes. However, there has been some improvement on this end recently, React's core API is now much stable than before so it's indeed getting better.

Also included is how the platform is highly flexible and the lack of conventional standards can make things complex particularly when onboarding new developers. Every React development team seems to have its own separate standards that developers have to learn to work with. In any case, this challenge can be mitigated through proper onboarding, as we do at ProCoders.

React is rich enough as a library, but developers coming from platforms such as Angular might miss some of the features associated with frameworks. For instance, on the Model-View-Controller structure algorithms, React represents only one part: the 'View' part. This gap can make React poorly structured and poorly managed.

What are React Hooks?

React Hooks are still a relatively new feature. That's why this question is in the mid-level section. Many people haven't gotten the hang of hooks completely. React Hooks were introduced in version 16.8, which was released in February 2019. For context, the latest version of React is version 17. So, it obviously has not been much longer than hooks were introduced. In any case, hooks are backward-compatible, so that's one potential problem avoided.

In simple terms, hooks allow a programmer to manipulate the state of stateless functional components without having to convert them into stateful class components. Essentially, with hooks, one no longer needs to write a class to use state and other React features. Some common hooks are:

- **useState** – return a function to update a stateful value
- **useEffect** – implementing side effects in function components
- **useContext** – return the current context value.

Users can also create their own custom hooks, for instance, by using higher-order components. The two rules of hooks are that one must only call hooks at the top level and that one can only call hooks from function components.

Business Benefits of React?

Because of reusable components, React developers don't have to write new code every time for different features that have similar functionalities. This improves the speed of the development process and reduces the time-to-market, essentially making the business more competitive in the industry.

React enables building SEO-friendly apps, and the benefits of SEO to marketing and business growth in this day cannot be overstated. By boosting page performance and lowering loading time, a business can improve the performance and user experience of its apps. This is a plus for SEO since website performance is a critical ranking factor of search engines.

React apps are easy to maintain. This lowers incidents of downtime, which may affect customer experience. In addition, React is a cost-optimized solution for building web apps.

Finally, to add to this, when answering these interview questions on ReactJS for experienced developers, they should be able to paint a picture of the future of React and its continued viability in the development stage.

What is JSX?

JSX is a syntax extension of JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code.

What is Redux and what are its components?

Redux is a popular library used for front-end development. It has no dependencies and is often used to help create JavaScript applications. Redux is made up of three components: actions, reducers and the store.

- **Store:** Holds the state of the application.
- **Action:** The source information for the store.
- **Reducer:** Specifies how the application's state changes in response to actions sent to the store.

How do you identify the difference between DOM and virtual DOM?

DOM is considered an abstraction of an HTML code that provides a visual representation of the written text displayed by the HTML code. Virtual DOM is related to HTML DOM in that it is considered to be derived from it, and is therefore considered an abstraction of an abstraction.

What are stateless components?

Stateless components are components that instead of possessing their own state, they take on whatever form that is applied to them. For this reason, they can also be called reusable components.

What are synthetic events in React?

- Synthetic events combine the response of different browser's native events into one API, ensuring that the events are consistent across different browsers.
- The application is consistent regardless of the browser it is running in. Here, `preventDefault` is a synthetic event.

Why is there a need for using keys in Lists?

Keys are very important in lists for the following reasons:

- A key is a unique identifier and it is used to identify which items have changed, been updated or deleted from the lists
- It also helps to determine which components need to be re-rendered instead of re-rendering all the components every time. Therefore, it increases performance, as only the updated components are re-rendered

How is React different from Angular?

	Angular	React
Author	Google	Facebook
Architecture	Complete MVC	View layer of MVC
DOM	Real DOM	Virtual DOM
Data-Binding	Bi-directional	Uni-directional
Rendering	Client-Side	Server-Side
Performance	Comparatively slow	Faster due to Virtual DOM

What is a state in React?

- The state is a built-in React object that is used to contain data or information about the component. The state in a component can change over time, and whenever it changes, the component re-renders.
- The change in state can happen as a response to user action or system-generated events. It determines the behavior of the component and how it will render.

What is React Router?

React Router is a routing library built on top of React, which is used to create routes in a React application.

Why do we need to React Router?

- It maintains consistent structure and behavior and is used to develop single-page web applications.
- Enables multiple views in a single application by defining multiple routes in the React application.

Differentiate between Real DOM and Virtual DOM.

<i>Real DOM vs Virtual DOM</i>	
Real DOM	Virtual DOM
1. It updates slow.	1. It updates faster.
2. Can directly update HTML.	2. Can't directly update HTML.
3. Creates a new DOM if element updates.	3. Updates the JSX if element updates.
4. DOM manipulation is very expensive.	4. DOM manipulation is very easy.
5. Too much of memory wastage.	5. No memory wastage.

What is React?

- React is a front-end JavaScript library developed by Facebook in 2011.
- It follows the component based approach which helps in building reusable UI components.
- It is used for developing complex and interactive web and mobile UI.
- Even though it was open-sourced only in 2015, it has one of the largest communities supporting it.

What are the features of React?

Major features of React are listed below:

- i. It uses the **virtual DOM** instead of the real DOM.
- ii. It uses **server-side rendering**.

- iii. It follows **uni-directional data flow** or data binding.

List some of the major advantages of React.

Some of the major advantages of React are:

- i. It increases the application's performance
- ii. It can be conveniently used on the client as well as server side
- iii. Because of JSX, code's readability increases
- iv. React is easy to integrate with other frameworks like Meteor, Angular, etc
- v. Using React, writing UI test cases become extremely easy

What are the limitations of React?

Limitations of React are listed below:

- i. React is just a library, not a full-blown framework
- ii. Its library is very large and takes time to understand
- iii. It can be little difficult for the novice programmers to understand
- iv. Coding gets complex as it uses inline templating and JSX

What is JSX?

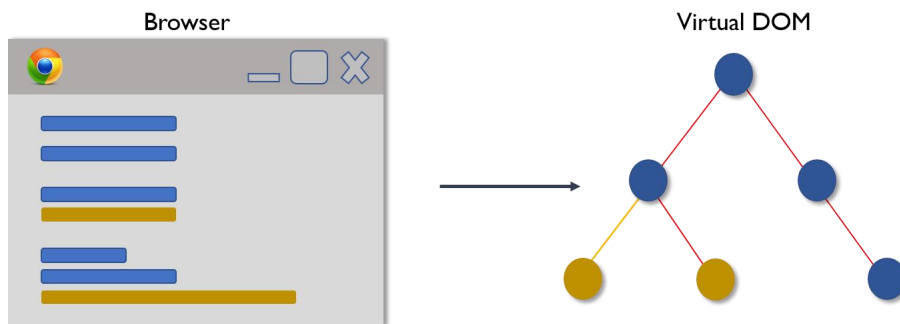
JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. This file makes applications robust and boosts its performance. Below is an example of JSX:

```
1  render(){
2    return(
3
4    <div>
5
6    <h1> Hello World from Edureka!!</h1>
7
8    </div>
9
10   );
11 }
```

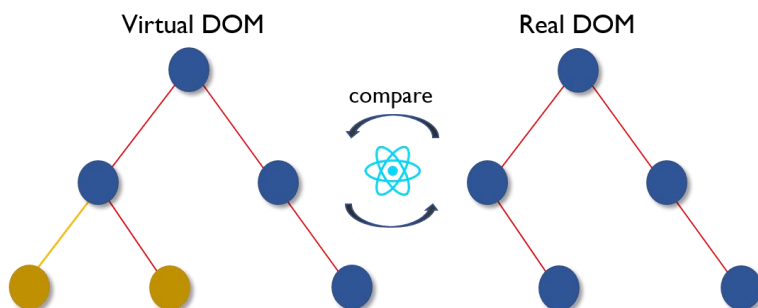
What do you understand by Virtual DOM? Explain its works.

A virtual DOM is a lightweight JavaScript object which originally is just a copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React's render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system. This Virtual DOM works in three simple steps.

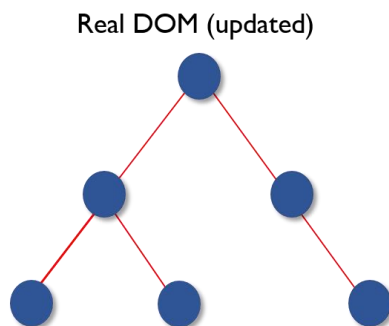
1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.



2. Then the difference between the previous DOM representation and the new one is calculated.



3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.



Why can't browsers read JSX?

Browsers can only read JavaScript objects but JSX is not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

How different is React's ES6 syntax when compared to ES5?

Syntax has changed from ES5 to ES6 in the following aspects:

i. require vs import

```
1 // ES5
2 var React = require('react');
3
4 // ES6
5 import React from 'react';
```

ii. export vs exports

```
1 // ES5
2 module.exports = Component;
3
4 // ES6
5 export default Component;
```

iii. component and function

```
1 // ES5
2 var MyComponent = React.createClass({
3   render: function() {
4     return
5
6     <h3>Hello Edureka!</h3>
7   ;
8   }
9 });
10
11 // ES6
12 class MyComponent extends React.Component {
13   render() {
14     return
15
16     <h3>Hello Edureka!</h3>
17   ;
18   }
19 }
```

iv. props


```

1 // ES5
2 var App = React.createClass({
3   propTypes: { name: React.PropTypes.string },
4   render: function() {
5     return
6
7     <h3>Hello, {this.props.name}!</h3>
8   ;
9   }
10 });
11
12 // ES6
13 class App extends React.Component {
14   render() {
15     return
16
17     <h3>Hello, {this.props.name}!</h3>
18   ;
19   }
20 }

```

v. state

```

1 // ES5
2 var App = React.createClass({
3   getInitialState: function() {
4     return { name: 'world' };
5   },
6   render: function() {
7     return
8
9     <h3>Hello, {this.state.name}!</h3>
10   ;
11   }
12 });
13
14 // ES6
15 class App extends React.Component {
16   constructor() {
17     super();
18     this.state = { name: 'world' };
19   }
20   render() {
21     return
22
23     <h3>Hello, {this.state.name}!</h3>

```

```
24 ;  
25 }  
26 }
```

How is React different from Angular?

<i>React vs Angular</i>		
TOPIC	REACT	ANGULAR
1. ARCHITECTURE	Only the View of MVC	Complete MVC
2. RENDERING	Server-side rendering	Client-side rendering
3. DOM	Uses virtual DOM	Uses real DOM
4. DATA BINDING	One-way data binding	Two-way data binding
5. DEBUGGING	Compile time debugging	Runtime debugging
6. AUTHOR	Facebook	Google

“In React, everything is a component.” Explain.

Components are the building blocks of a React application’s UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

What is the purpose of render() in React.

Each React component must have a **render()** mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>**, **<group>**, **<div>** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

How can you embed two or more components into one?

We can embed components into one in the following way:

```
1  class MyComponent extends React.Component{
2      render(){
3          return(
4
5      <div>
6
7      <h1>Hello</h1>
8
9          <Header/>
10         </div>
11
12         );
13     }
14 }
15 class Header extends React.Component{
16     render(){
17         return
18
19     <h1>Header Component</h1>
20
21     };
22 }
23 ReactDOM.render(
24     <MyComponent/>, document.getElementById('content')
25 );
```

What is Props?

Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This helps in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

What is a state in React and how is it used?

States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state()**.

Differentiate between states and props.

<i>States vs Props</i>		
Conditions	State	Props
1. Receive initial value from parent component	Yes	Yes
2. Parent component can change value	No	Yes
3. Set default values inside component	Yes	Yes
4. Changes inside component	Yes	No
5. Set initial value for child components	Yes	Yes
6. Changes inside child components	No	Yes

How can you update the state of a component?

State of a component can be updated using `this.setState()`.

```

1  class MyComponent extends React.Component {
2    constructor() {
3      super();
4      this.state = {
5        name: 'Maxx',
6        id: '101'
7      }
8    }
9    render()
10   {
11     setTimeout(()=>{this.setState({name:'Jaacha', id:'222'})},2000)
12     return (
13
14     <div>
15
16     <h1>Hello {this.state.name}</h1>
17
18     <h2>Your Id is {this.state.id}</h2>
19
20       </div>
21
22     );
23   }
24 }
25 ReactDOM.render(
26   <MyComponent/>, document.getElementById('content')
27 );

```

What is arrow function in React? How is it used?

Arrow functions are more of brief syntax for writing the function expression. They are also called *'fat arrow'* (`=>`) the functions. These functions allow to bind the context of the components properly since

in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

```
1 //General way
2 render() {
3   return(
4     <MyInput onChange={this.handleChange.bind(this)} />
5   );
6 }
7 //With Arrow Function
8 render() {
9   return(
10    <MyInput onChange={ (e) => this.handleChange(e) } />
11  );
12 }
```

Differentiate between stateful and stateless components.

<i>Stateful vs Stateless</i>	
Stateful Component	Stateless Component
1. Stores info about component's state change in memory	1. Calculates the internal state of the components
2. Have authority to change state	2. Do not have the authority to change state
3. Contains the knowledge of past, current and possible future changes in state	3. Contains no knowledge of past, current and possible future state changes
4. Stateless components notify them about the requirement of the state change, then they send down the props to them.	4. They receive the props from the Stateful components and treat them as callback functions.

What are the different phases of React component's lifecycle?

There are three different phases of React component's lifecycle:

- i. Initial Rendering Phase: This is the phase when the component is about to start its life journey and make its way to the DOM.
- ii. Updating Phase: Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
- iii. Unmounting Phase: This is the final phase of a component's life cycle in which the component is destroyed and removed from the DOM.

Explain the lifecycle methods of React components in detail.

Some of the most important lifecycle methods are:

- i. ***componentWillMount()*** – Executed just before rendering takes place both on the client as well as server-side.
- ii. ***componentDidMount()*** – Executed on the client side only after the first render.
- iii. ***componentWillReceiveProps()*** – Invoked as soon as the props are received from the parent class and before another render is called.
- iv. ***shouldComponentUpdate()*** – Returns true or false value based on certain conditions. If you want your component to update, return **true** else return **false**. By default, it returns false.
- v. ***componentWillUpdate()*** – Called just before rendering takes place in the DOM.
- vi. ***componentDidUpdate()*** – Called immediately after rendering takes place.
- vii. ***componentWillUnmount()*** – Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.

What is an event in React?

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

- i. Events are named using camel case instead of just using the lowercase.
- ii. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

How do you create an event in React?

```
1  class Display extends React.Component({
2    show(evt) {
3      // code
4    },
5    render() {
6      // Render the div with an onClick prop (value is a function)
7      return (
8
9        <div onClick={this.show}>Click Me!</div>
10
11      );
12    }
13  });
```

What are synthetic events in React?

Synthetic events are the objects which act as a cross-browser wrapper around the browser's native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

What do you understand by refs in React?

Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.

```
1  class ReferenceDemo extends React.Component{
2    display() {
3      const name = this.inputDemo.value;
4      document.getElementById('disp').innerHTML = name;
5    }
6    render() {
7      return(
8
9      <div>
10         Name: <input type="text" ref={input => this.inputDemo = input} />
11         <button name="Click" onClick={this.display}>Click</button>
12
13      <h2>Hello <span id="disp"></span> !!!</h2>
14
15      </div>
16    );
17  }
18 }
```

List some of the cases when you should use Refs.

Following are the cases when refs should be used:

- When you need to manage focus, select text or media playback
- To trigger imperative animations
- Integrate with third-party DOM libraries

How do you modularize code in React?

We can modularize code by using the export and import properties. They help in writing the components separately in different files.

```
1  //ChildComponent.jsx
2  export default class ChildComponent extends React.Component {
3    render() {
4      return(
5
6      <div>
7
8      <h1>This is a child component</h1>
9
10     </div>
11  )
```

```

12     );
13   }
14 }
15
16 //ParentComponent.jsx
17 import ChildComponent from './childcomponent.js';
18 class ParentComponent extends React.Component {
19   render() {
20     return(
21
22 <div>
23       <App />
24 </div>
25
26     );
27   }
28 }

```

How are forms created in React?

React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via `setState()`. Thus the elements can't directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

```

1  handleSubmit(event) {
2    alert('A name was submitted: ' + this.state.value);
3    event.preventDefault();
4  }
5
6  render() {
7    return (
8
9    <form onSubmit={this.handleSubmit}>
10      <label>
11        Name:
12        <input type="text" value={this.state.value} onChange={this.handleSubmit} />
13      </label>
14      <input type="submit" value="Submit" />
15    </form>
16
17    );
18  }

```

What do you know about controlled and uncontrolled components?

<i>Controlled vs Uncontrolled Components</i>	
Controlled Components	Uncontrolled Components
1. They do not maintain their own state	1. They maintain their own state

2. Data is controlled by the parent component	2. Data is controlled by the DOM
3. They take in the current values through props and then notify the changes via callbacks	3. Refs are used to get their current values

What are Higher Order Components(HOC)?

Higher Order Component is an advanced way of reusing the component logic. Basically, it's a pattern that is derived from React's compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won't modify or copy any behavior from their input components. You can say that HOC are 'pure' components.

What can you do with HOC?

HOC can be used for many tasks like:

- Code reuse, logic and bootstrap abstraction
- Render Hijacking
- State abstraction and manipulation
- Props manipulation

What are Pure Components?

Pure components are the simplest and fastest components which can be written. They can replace any component which only has a **render()**. These components enhance the simplicity of the code and performance of the application.

What is the significance of keys in React?

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React just reorders the elements instead of re-rendering them. This leads to increase in application's performance.

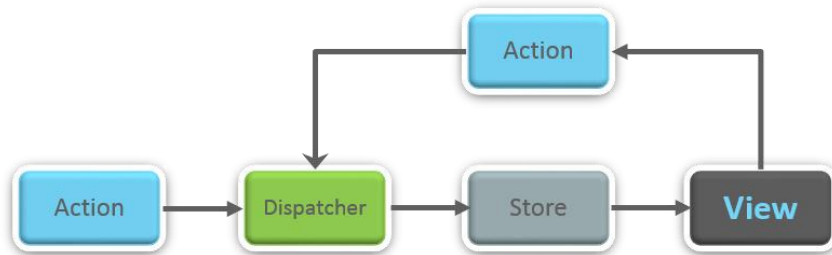
What were the major problems with MVC framework?

Following are some of the major problems with MVC framework:

- DOM manipulation was very expensive
- Applications were slow and inefficient
- There was huge memory wastage
- Because of circular dependencies, a complicated model was created around models and views

Explain Flux.

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for

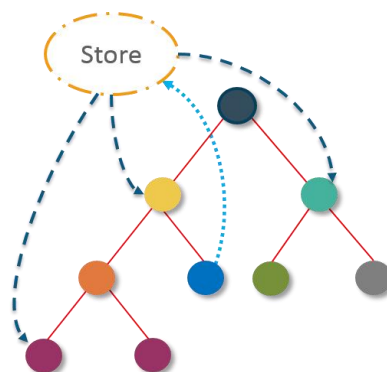


What is Redux?

Redux is one of the most trending libraries for front-end development in today's marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

What are the three principles that Redux follows?

- i. **Single source of truth:** The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
- ii. **State is read-only:** The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
- iii. **Changes are made with pure functions:** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.



What do you understand by “Single source of truth”?

Redux uses 'Store' for storing the application's entire state at one place. So all the component's state are stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

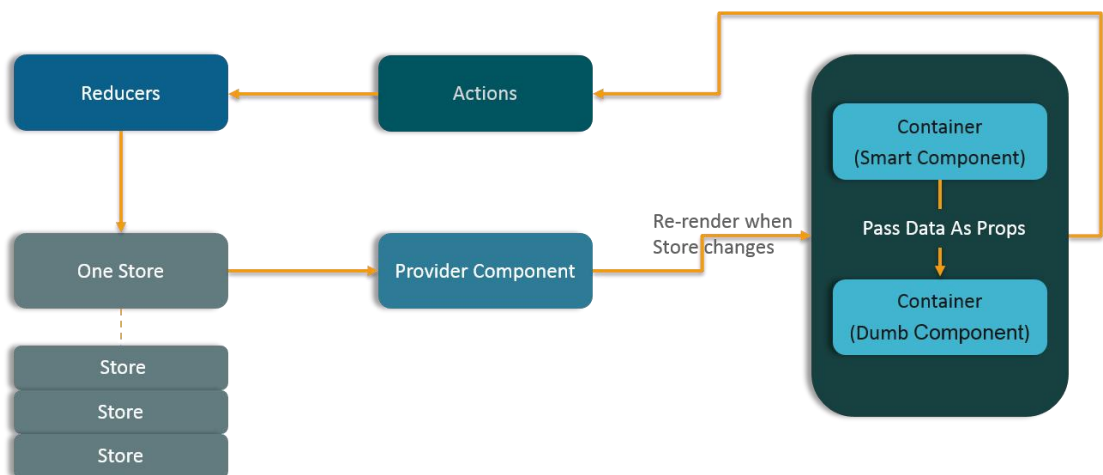
List down the components of Redux.

Redux is composed of the following components:

- i. **Action** – It's an object that describes what happened.
- ii. **Reducer** – It is a place to determine how the state will change.
- iii. **Store** – State/ Object tree of the entire application is saved in the Store.
- iv. **View** – Simply displays the data provided by the Store.

In case you are facing any challenges with these React interview questions, please comment on your problems in the section below.

Show how the data flows through Redux?



How are Actions defined in Redux?

Actions in Redux must have a `type` property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

```
1 function addTodo(text) {  
2   return {  
3     type: ADD_TODO,  
4     text  
5   }  
6 }
```

Explain the role of Reducer.

Reducers are pure functions which specify how the application's state changes in response to an ACTION. Reducers work by taking in the previous state and action, and then it returns a new state. It

determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

What is the significance of Store in Redux?

A store is a JavaScript object which can hold the application's state and provide a few helper methods to access the state, dispatch actions and register listeners. The entire state/ object tree of an application is saved in a single store. As a result of this, Redux is very simple and predictable. We can pass middleware to the store to handle the processing of data as well as to keep a log of various actions that change the state of stores. All the actions return a new state via reducers.

How is Redux different from Flux?

<i>Flux vs Redux</i>	
Flux	Redux
1. The Store contains state and change logic	1. Store and change logic are separate
2. There are multiple stores	2. There is only one store
3. All the stores are disconnected and flat	3. Single store with hierarchical reducers
4. Has singleton dispatcher	4. No concept of dispatcher
5. React components subscribe to the store	5. Container components utilize connect
6. State is mutable	6. State is immutable

In case you are facing any challenges with these React interview questions, please comment on your problems in the section below.

What are the advantages of Redux?

Advantages of Redux are listed below:

- **Predictability of outcome** – Since there is always one source of truth, i.e. the store, there is no confusion about how to sync the current state with actions and other parts of the application.
- **Maintainability** – The code becomes easier to maintain with a predictable outcome and strict structure.
- **Server-side rendering** – You just need to pass the store created on the server, to the client side. This is very useful for initial render and provides a better user experience as it optimizes the application performance.
- **Developer tools** – From actions to state changes, developers can track everything going on in the application in real time.
- **Community and ecosystem** – Redux has a huge community behind it which makes it even more captivating to use. A large community of talented individuals contribute to the betterment of the library and develop various applications with it.
- **Ease of testing** – Redux's code is mostly functions which are small, pure and isolated. This makes the code testable and independent.
- **Organization** – Redux is precise about how code should be organized, this makes the code more consistent and easier when a team works with it.

What is React Router?

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that's being displayed on the web

page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

Why is switch keyword used in React Router v4?

Although a `<div>` is used to encapsulate multiple routes inside the Router. The 'switch' keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The `<switch>` tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

Why do we need a Router in React?

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any 'route' defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

```
1 <switch>
2   <route exact path="/" component={Home}/>
3   <route path="/posts/:id" component={Newpost}/>
4   <route path="/posts" component={Post}/>
5 </switch>
```

List down the advantages of React Router.

Few advantages are:

- i. Just like how React is based on components, in React Router v4, the API is '*All About Components*'. A Router can be visualized as a single root component (`<BrowserRouter>`) in which we enclose the specific child routes (`<route>`).
- ii. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the `<BrowserRouter>` component.
- iii. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.

How is React Router different from conventional routing?

<i>Conventional Routing vs React Routing</i>		
Topic	Conventional Routing	React Routing
PAGES INVOLVED	Each view corresponds to a new file	Only single HTML page is involved
URL CHANGES	A HTTP request is sent to a server and corresponding HTML page is received	Only the History attribute is changed
FEEL	User actually navigates across different pages for each view	User is duped thinking he is navigating across different pages

React Lifecycle

Normally, React component has lifecycle with Mounting, Updating and Unmounting

Mounting means putting elements into the DOM

Constructor() - **getDerivedStateFromProps()** - **render()** - **componentDidMount()**

The render method is required and will always be called while the others are optional and only be called when they are defined.

Updating. A component is updated whenever there is a change in the component's state or props

getDerivedStateFromProps() - **shouldComponentUpdate()** - **render()** -
getSnapshotBeforeUpdate() - **componentDidUpdate()**

Unmounting. Component is removed from DOM

Only one built-in method with componentWillUnmount()

State vs Props - 2

Props are used to pass data from one component to another.

The state is a local data storage that is local to the component only and cannot be passed to other components.

While state is mutable which means it can be changed within the component by using `setState()` function or etc, props is immutable which means it's read-only.

When re-render()?

React component is normally re-rendered when the state or props are changed.

Also there is another case when the component is re-rendering when the parent component is re-rendering even though the props are changing or not.

Too many unnecessary re-renderings affect the app performance and cause loss of user's battery which surely no user would want. So we can use react hooks with `React.memo` to prevent unwanted re-renders. It doesn't re-render the child component if the props are not changed.

React hooks 3?

React hooks came across after React v16.8 and allows users to define state variables within the function component. There are both function component as well as class components. In the past, we were not able to add state variables within the function component, we can only render elements in it.

But after hooks came, we can also define state variables with `useState` and also other lifecycle functions like `componentDidMount`, `componentDidUpdate` using `useEffect`.

I normally love to use react hooks because it provides clean syntax not complicated codebase like class component, but the main reason is that it normally provides better performance for the application.

Callback vs promise vs observables?

A JavaScript callback is a function which is to be executed after another function has finished execution. We need callback functions because many JavaScript actions are **asynchronous**, which means they don't really stop the program (or a function) from running until they're completed, as you're probably used to. Instead, it will execute in the background while the rest of the code runs.

Meanwhile callback function is widely being used, it has some weakness. If the application gets bigger, we will have nested callbacks and this can cause callback hell with infinite loop where the app execution doesn't end.

That's why there are also promises with better performance than callbacks. A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason. This lets asynchronous methods return values like synchronous methods: instead of immediately returning the final value, the asynchronous method returns a promise to supply the value at some point in the future.

With modern javascript, there is also `async/await` promise which is better than callback.

Observable is a collection of asynchronous elements

While promises can handle only one event, observables handle 0, 1 or more events which callback function is called for each event. Also observable is cancellable. The last difference is that promise executes immediately while observable is called when subscribe. This means observable is lazier than promise but produce better performance

Catch error with `async/await`

Normally, we are using `try....catch` statement to catch the issue with synchronous code. But as you know `async/await` is a promise which is being used for modern Javascript.

If we are using this original `try...catch` statement to call `async` function, even though we throw an error in the function, it does not catch the exception. In this case, we can solve this problem with 2 ways.

If we want to use `try....catch`, we need to define `async` function and use `await` when call the function.

Second way is to remove `try....catch` statement and instead, we can use promise callback functions with `then()`, `catch()`, and `finally()`. Here at `catch()` callback function, we can catch exception similarly in `try....catch` statement, we can see the error message and do some error handling here.

So normally, there are two ways to catch error with async/await, but I normally use async/await (the first choice) within the application.

What is redux (store, action, reducer)-2?

Redux is the state management library containing a store with redux variables for the application.

As redux variables are read-only, we need to dispatch actions and the variables are changed in reducer.

Action – It's an object that describes what happened.

Reducer – It is a place to determine how the state will change.

Store – State/ Object tree of the entire application is saved in the Store.

Redux is widely being used with many applications as it's providing better performance with state management. Recently I have also experienced with Mobx for the state management, it has some differences with Redux as it has multiple stores etc.

createSlice function in Redux

A function that accepts an **initial state**, an **object of reducer functions**, and a "**slice name**", and automatically generates action creators and action types that correspond to the reducers and state. This API is the standard approach for writing Redux logic

Context API vs Redux

When we pass variables from parent to child, we normally use props. But if we have so many nested components, we need to pass the props value to all child components to reach to the last nested child. This makes code messy and this is called props drilling.

To remove props drilling, we use state management library like Redux or Context API which is the built-in tool for React.

Meanwhile both Redux and Context Api are the solutions to remove props drilling, there are some differences.

To use redux, we need to install another npm library react-redux, but Context API is built-in tool, so we can use this without installing a new dependency, and this will affect the bundle size.

The next one is that Context API is specifically designed for specific data but redux is for both static/dynamic data.

Also Context API has UI logic and state management logic within one component, but redux has better code organizations with separate UI logic and state management logic so redux provides easy debugging with redux dev tool, but it's a little hard for us to debug Context API if we have highly nested react components.

So context api is a good solution for a small application with small bundle size as well as easy to code, but for big applications, I recommend to use redux with well-organized structure and easy-to-debug.

Error boundaries in React

Error boundaries are react components that catch Javascript errors anywhere in their child component tree, log those errors, and display fallback UI instead of the component tree that crashed. Error boundaries catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them

Closure in JS

Before we say about closure in javascript, I wanna explain about the scope of variables. So we have global variables as well as local variables etc. Lexical scoping defines the scope of variables with the position where the variables are defined.

Normally a local variable only exists while the function executes but there is a closure. Closure is a function that preserves the outer scope in its inner scope

Check if the variable is array

There are three ways to check if the variable is array.

- We can use isArray function with `Array.isArray(variableName)`
- We can use instanceof operator `variable instanceof Array`
- We can use with constructor property. `Variable.constructor.name === 'Array'`

Arrow functions vs Regular functions

Arrow functions came across after ES6. It allows to create functions in a cleaner way compared to regular functions, we can use arrow operator, like `let functionName = () => {}`

In regular functions, **this** refers to the function where it is called, instead with the arrow functions, it does not have its own this. At this time, **this** refers to its parent scope.

There are some cases which we can't use this arrow function in the application

- We do not use arrow function for event handler when we need to use **this**.
- We do not use arrow function when add method to the object.
- We do not use arrow function for prototype functions when we need to use **this**.
- We can't use arrow function to define a constructor.

Object, Array == return false?

Normally when we compare two variables, we use == operator, but this operator returns false for two objects even though they have exactly the same properties or they are both empty.

The reason is that == operator compares the references of the variable, not the exact value. For the objects, it has different reference even though they have the same properties, so to compare two objects we need to loop and compare each property to compare the objects.

This is the same for Array also, it is always returning false when using == operator even though we have the same values in an array with the same order.

React testing, Why Jest - 2?

Jest, Jasmine, React testing library, enzyme, mocha, chai, Unit testing, integration testing, e2e testing, A/B testing

Jest is pretty simple, fast(parallelization, run slowest first, caching babel transforms), Typescript(ts-jest), Snapshot, Delta Testing with Watch

Simply install Jest using npm and yarn.

React Native (Cross-platform) vs native mobile apps

Take an alligator and a fish. An alligator can walk and swim, but will never swim as good as a fish. On the other hand, fish cannot walk at all. So cross-platform apps are like alligators. They can swim and walk, but will never be as good at swimming, as fishes (native apps).

Pros: All platforms, Easy to develop, Recycle components

Cons: Low performance than native, updating issues

FlatList vs ScrollView (React Native)

When we want to show list of items, there are two ways, with FlatList or ScrollView.

While ScrollView renders all the items at once, FlatList renders only items which are currently displayed.

That's why FlatList generally provides better performance and we recommend to use FlatList for the large list of data.

In the past, ListView was also being used to render list of items but it's deprecated for now.

React Native Navigators

There are 3 types of navigators in react native – createStackNavigator, tabNavigator and drawerNavigator

StackNavigator

The first one StackNavigator includes or holds screens in stack format [which means when a new screen is viewed, it is positioned on top of the existing one]

TabNavigator

The TabNavigator permits a user to use different screens by simply using the tab that one can use either at the top or bottom of the screen.

DrawerNavigator

DrawerNavigation is the preferred option for Android and is a type of navigation hint that offers users options and points to different screens.

React Native Native Modules

Native modules are set of javascript functions that are implemented natively for each platform. It is used when native capabilities are needed, react native does not have corresponding module or native performance is better

There is also a Native UI Component that creates a native view within inside JSX.

Normally we create a new module with react-native-create-library command, after you run this command, you can see we have android, ios and windows directories within our module

And we can define each native module for both android and ios and at last we deploy our native module.

React Native push notifications

Actually, we can implement the push notifications with React Native app on 3 ways

Apple push notification, Expo push notification, Firebase Cloud Messaging Push notification

-Apple push notification

You don't need to know client's ip address and you should have apple account for this.

-Expo push notification

It's free provided by expo, however the project should be implemented with expo.

-FCM push notification

This is provided by google and we can use firebase cloud messaging feature to implement the push notifications with react native apps.

Push Notification with FCM

First of all, you need to register app project on the firebase with your account

You can generate SSH key of the project and add that key to newly created project of the firebase

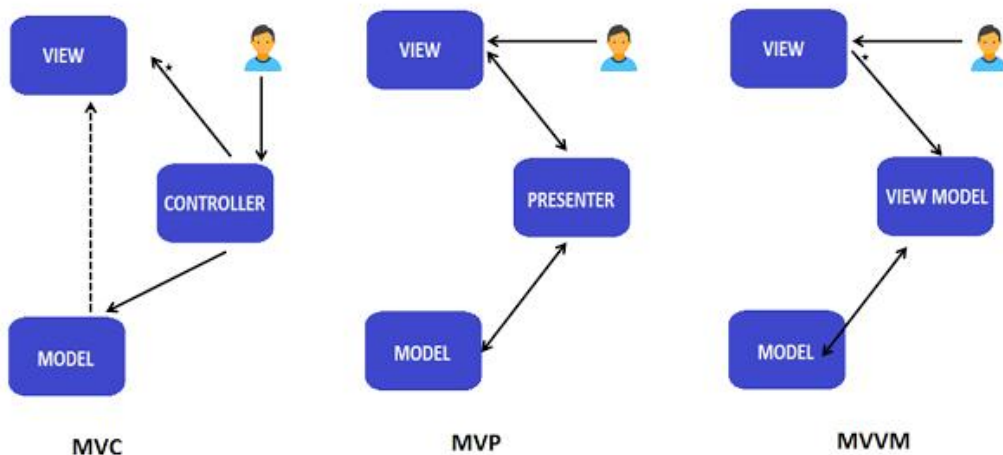
Next, You need to store push notification token in a project

At last, you need to add event listener of the push notification.

You can define a function within the event listener so that you can show text message on the top of the screen with the notification or etc.

MVC, MVP and MVVM Design Pattern

MVC, MVP, and MVVM are three popular design patterns in software development. Let's have a look on Model View Controller (MVC), Model View Presenter (MVP) and Model View View-model (MVVM) one by one. All these design patterns by and large help in developing applications that are loosely combined, easy to test and maintain. All discussion about the pattern is made in context of Android as a platform.



Model View Controller (MVC)

MVC design pattern divides an application into three major aspects: Model, View, and Controller.

- Model

Model means data that is required to display in the view. Model represents a collection of classes that describes the business logic (business model and the data model). It also defines the business rules for data means as how the data can be changed and manipulated.

- View

The View represents UI components like XML, HTML etc. View displays the data that is received from the controller as the outcome. In MVC pattern View monitors the model for any state change and displays updated model. Model and View interact with each other using the Observer pattern.

- Controller

The Controller is responsible to process incoming requests. It processes the user's data through the Model and passing back the results to View. It normally acts as a mediator between the View and the Model.

Model View Presenter (MVP)

The MVP pattern is similar to the MVC pattern. It is derived from MVC pattern, wherein the controller is replaced by the presenter. This pattern divides an application into three major aspects: Model, View, and Presenter.

- Model

The Model represents a set of classes that describes the business logic and data. It also defines business rules for data means how the data can be changed and manipulated.

- View

View is a component which is directly interacts with user like XML, Activity, fragments. It does not contain any logic implemented.

- Presenter

The Presenter receives the input from users via View, then process the user's data with the help of Model and passing the results back to the View. Presenter communicates with view through interface. Interface is defined in presenter class, to which it pass the required data. Activity/fragment or any other view component implement this interface and renders the data in a way they want.

In the MVP design pattern, the presenter manipulates the model and also updates the view. In MVP View and Presenter are completely decoupled from each other's and communicate to each other's by an interface. Because if decoupling mocking of the view is easier and unit testing of applications that leverage the MVP design pattern over the MVC design pattern are much easier.

Model View View-model (MVVM)

MVVM pattern supports two-way data binding between View and View-Model. This allows automatic propagation of changes, inside the state of View-Model to the View. Generally, the View-Model utilizes the observer pattern to inform changes in the View-Model to the Model.

- View-Model

It is responsible for exposing methods, commands, and other properties that help to maintain the state of the view, manipulate the model as the result of actions on the view, and trigger events in the view itself. View has a reference to View-Model but View-Model has no information about the View. There is many-to-one relationship between View and View-Model means many Views can be mapped to one View-Model. It is completely independent of Views.

The bi-directional data binding or the two way data binding between the view and the View-Model ensures that the models and properties in the View-Model is in sync with the view. The MVVM design pattern is well suited in applications that need support for bi-directional data binding.

Native Module Setup

There are two ways to write a native module for your React Native application:

1. Directly within your React Native application's iOS/Android projects
2. As a NPM package that can be installed as a dependency by your/other React Native applications

Android Native Modules

- Directly within your React Native application's Android project

Setup

To get started, open up the Android project within your React Native application in Android Studio. You can find your Android project in `project/android` directory.

Create custom native module file

The first step is to create the `[ModuleName].java` Java file inside `android/app/src/main/java/com/your-app-name/` folder. This Java file will contain your native module Java class.

Then add the following content:

```
package com.your-app-name; // replace com.your-app-name with your app's name
import com.facebook.react.bridge.NativeModule;
import com.facebook.react.bridge.ReactApplicationContext;
import com.facebook.react.bridge.ReactContext;
import com.facebook.react.bridge.ReactContextBaseJavaModule;
import com.facebook.react.bridge.ReactMethod;
import java.util.Map;
import java.util.HashMap;

public class CalendarModule extends ReactContextBaseJavaModule {
    CalendarModule(ReactApplicationContext context) {
        super(context);
    }
}
```

```
}
```

As you can see, your **CalendarModule** class extends the **ReactContextBaseJavaModule** class. For Android, Java native modules are written as classes that extend **ReactContextBaseJavaModule** and implement the functionality required by JavaScript.

Module Name

All Java native modules in Android need to implement the **getName()** method. This method returns a string, which represents the name of the native module. The native module can then be accessed in JavaScript using its name. For example, in the below code snippet, **getName()** returns **"CalendarModule"**.

```
// add to CalendarModule.java
@Override
public String getName() {
    return "CalendarModule";
}
```

The native module can then be accessed in JS like this:

```
const { CalendarModule } = ReactNative.NativeModules;
```

Export a Native Method to JavaScript

Next you will need to add a method to your native module that will create calendar events and can be invoked in JavaScript. All native module methods meant to be invoked from JavaScript must be annotated with **@ReactMethod**.

Set up a method **createCalendarEvent()** for **CalendarModule** that can be invoked in JS through **CalendarModule.createCalendarEvent()**. For now, the method will take in a name and location as strings. Argument type options will be covered shortly.

```
@ReactMethod
public void createCalendarEvent(String name, String location) {
}
```

Add a debug log in the method to confirm it has been invoked when you call it from your application. Below is an example of how you can import and use the Log class from the Android util package:

```
import android.util.Log;

@ReactMethod
public void createCalendarEvent(String name, String location) {
    Log.d("CalendarModule", "Create event called with name: " + name
        + " and location: " + location);
}
```

Once you finish implementing the native module and hook it up in JavaScript, you can follow these steps to view the logs from your app.

Register the Module (Android Specific)

Once a native module is written, it needs to be registered with React Native. In order to do so, you need to add your native module to a **ReactPackage** and register the **ReactPackage** with React Native. During initialization, React Native will loop over all packages, and for each **ReactPackage**, register each native module within.

React Native invokes the method **createNativeModules()** on a **ReactPackage** in order to get the list of native modules to register. For Android, if a module is not instantiated and returned in **createNativeModules** it will not be available from JavaScript.

To add your Native Module to **ReactPackage**, first create a new Java Class named `MyAppPackage.java` that implements **ReactPackage** inside the `android/app/src/main/java/com/your-app-name/` folder:

Then add the following content:

```
package com.your-app-name; // replace your-app-name with your app's name
import com.facebook.react.ReactPackage;
import com.facebook.react.bridge.NativeModule;
import com.facebook.react.bridge.ReactApplicationContext;
import com.facebook.react.uimanager.ViewManager;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class MyAppPackage implements ReactPackage {

    @Override
    public List<ViewManager> createViewManagers(ReactApplicationContext reactContext) {
        return Collections.emptyList();
    }

    @Override
    public List<NativeModule> createNativeModules(
        ReactApplicationContext reactContext) {
        List<NativeModule> modules = new ArrayList<>();

        modules.add(new CalendarModule(reactContext));

        return modules;
    }
}
```

This file imports the native module you created, **CalendarModule**. It then instantiates **CalendarModule** within the **createNativeModules()** function and returns it as a list of **NativeModules** to register. If you add more native modules down the line, you can also instantiate them and add them to the list returned here.

It is worth noting that this way of registering native modules eagerly initializes all native modules when the application starts, which adds to the startup time of an application. You can use **TurboReactPackage** as an alternative. Instead of **createNativeModules**, which return a list of instantiated native module objects, **TurboReactPackage** implements a **getModule(String name, ReactApplicationContext rac)** method that creates the native module object, when required. **TurboReactPackage** is a bit more complicated to implement at the moment. In addition to implementing a **getModule()** method, you have to implement

a **getReactModuleInfoProvider()** method, which returns a list of all the native modules the package can instantiate along with a function that instantiates them, example here. Again, using **TurboReactPackage** will allow your application to have a faster startup time, but it is currently a bit cumbersome to write. So proceed with caution if you choose to use **TurboReactPackages**.

To register the **CalendarModule** package, you must add **MyAppPackage** to the list of packages returned in **ReactNativeHost**'s **getPackages()** method. Open up your **MainApplication.java** file, which can be found in the following path: **android/app/src/main/java/com/your-app-name/MainApplication.java**

Locate **ReactNativeHost**'s **getPackages()** method and add your package to the packages list **getPackages()** returns:

```
@Override
protected List<ReactPackage> getPackages() {
    @SuppressWarnings("UnnecessaryLocalVariable")
    List<ReactPackage> packages = new PackageList(this).getPackages();
    // below MyAppPackage is added to the list of packages returned
    packages.add(new MyAppPackage());
    return packages;
}
```

You have now successfully registered your native module for Android!

Test What You Have Built

At this point, you have set up the basic scaffolding for your native module in Android. Test that out by accessing the native module and invoking its exported method in JavaScript.

Find a place in your application where you would like to add a call to the native module's **createCalendarEvent()** method. Below is an example of a component, **NewModuleButton** you can add in your app. You can invoke the native module inside **NewModuleButton**'s **onPress()** function.

```
import React from 'react';
import { NativeModules, Button } from 'react-native';

const NewModuleButton = () => {
  const onPress = () => {
    console.log('We will invoke the native module here!');
  };

  return (
    <Button
      title="Click to invoke your native module!"
      color="#841584"
      onPress={onPress}
    />
  );
};

export default NewModuleButton;
```

In order to access your native module from JavaScript you need to first import **NativeModules** from **React Native**:

```
import { NativeModules } from 'react-native';
```

You can then access the CalendarModule native module off of NativeModules.

```
const { CalendarModule } = NativeModules;
```

Now that you have the CalendarModule native module available, you can invoke your native method createCalendarEvent(). Below it is added to the onPress() method in NewModuleButton:

```
const onPress = () => {  
  CalendarModule.createCalendarEvent('testName', 'testLocation');  
};
```


Vue Questions

1. What is Vue?

Angular Questions

1. What is Angular? Why was it introduced?

Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability.

Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.

2. What is data binding? Which type of data binding does Angular deploy?

Data binding is a phenomenon that allows any internet user to manipulate Web page elements using a Web browser. It uses dynamic HTML and does not require complex scripting or programming. We use data binding in web pages that contain interactive components such as forms, calculators, tutorials, and games. Incremental display of a webpage makes data binding convenient when pages have an enormous amount of data.

Angular uses the two-way binding. Any changes made to the user interface are reflected in the corresponding model state. Conversely, any changes in the model state are reflected in the UI state. This allows the framework to connect the DOM to the Model data via the controller. However, this approach affects performance since every change in the DOM has to be tracked.

3. What are decorators in Angular?

Decorators are a design pattern or functions that define how Angular features work. They are used to make prior modifications to a class, service, or filter. Angular supports four types of decorators, they are:

1. Class Decorators
2. Property Decorators
3. Method Decorators
4. Parameter Decorator

4. Advantages of Angular.

1. MVC architecture - Angular is a full-fledged MVC framework. It provides a firm opinion on how the application should be structured. It also offers bi-directional data flow and updates the real DOM.

2. Modules: Angular consists of different design patterns like components, directives, pipes, and services, which help in the smooth creation of applications.
3. Dependency injection: Components dependent on other components can be easily worked around using this feature.
4. Other generic advantages include clean and maintainable code, unit testing, reusable components, data binding and excellent responsive experience.

5. What's New in Angular 14: Latest Features?

1. Stand-alone Components:

The Angular 14 modules are optional; however, the goal is to move away from the existing setup by building pipes, directives, and components.

In order to make Ng modules optional on stand-alone components, Angular has issued RFC (Request for Comments). These modules would not be obsolete in the Angular 14 update but will become provisional to keep up with the compatibility of the existing Angular libraries and applications.

It is worth noting that before Angular 14, each component was required to be associated with a module. The application would fail if the declarations array of a parent module is not linked with every component.

2. Strictly Typed Forms:

The new Angular version update completely rectifies the major Github issue, i.e., executing strict typing for Angular reactive forms package. The strictly typed forms will enhance a modern-driven approach for Angular to function seamlessly with the existing forms.

One of the most promising elements about what's new in Angular 14 is that they have provided a smooth transition experience from previous versions to v14. The Angular team has added auto migration to maintain the existing applications during the upgrade. Also, you can now work with FormControl which calls a particular value it carries, accepting generic type input.

3. Angular CLI Auto-Completion:

The best part about the Angular CLI auto-completion is that you can improve productivity by delivering the required commands to create modules, directives, and components for your new/existing project. However, the Angular 14 has plenty of commands handy for you.

4. Improved Template Diagnostics:

The new Angular 14 update comes with enhanced template diagnostics to shield the developers from generic errors through compiler reconciliation to typescript code.

In Angular 13 and the previous versions, there are no warning signs generated by the compiler, and it ceases to execute if any issue restricts it from doing so.

5. Streamlined Page Title Accessibility:

Generally, your page title distinctly shows the contents of your page during application development. Previously in Angular 13, the whole process of adding titles was aligned with the fresh Route.title property in the Angular router. However, Angular 14 doesn't provide additional imports that are required while adding a title to your page.

6. Latest Primitives in the Angular CDK:

The Angular Component Dev Kit (CDK) offers a comprehensive set of tools for Angular component development. In Angular 14, the CDK menu and Dialog have been driven to a stable Angular version. Nonetheless, the new CDK primitives allow the creation of custom components that are better accessible.

7. Angular DevTools is Now Present Online:

It is simple to employ the Angular DevTools debugging extension under the offline mode. This extension is available under the Mozilla's Add-ons for Firefox users.

8. Optional Injectors:

If you are developing an embedded view in Angular14, you can mention an optional injector through `TemplateRef.createEmbeddedView` and `ViewContainerRef.createEmbeddedView`.

9. Built-in Enhancements:

One of the interesting things about the Angular14 update is that it enables the CLI to deploy small code without reducing its value. The built-in enhancements assist you in connecting to protected component members straight from your templates. Overall, you get more control over the reusable components using the public API surface.

10. Extended Developer Diagnostics:

The extended developer diagnostics is an Angular14 feature delivering an extendable framework that aids better understanding into your template and shows suggestions facilitating potential boosts.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

7. What are Directives in Angular?

Directives are attributes that allow the user to write new HTML syntax specific to their applications. They execute whenever the Angular compiler finds them in the DOM. Angular supports three types of directives.

Component Directives

Structural Directives

Attribute Directives

8. What is an AOT compilation? What are its advantages?

The Ahead-of-time (AOT) compiler converts the Angular HTML and TypeScript code into JavaScript code during the build phase, i.e., before the browser downloads and runs the code.

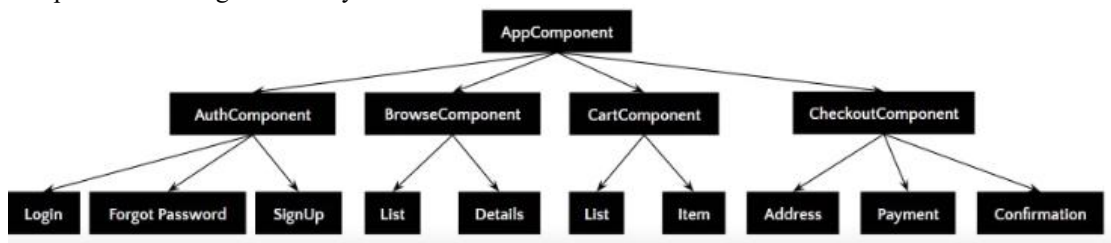
Some of its advantages are as follows.

- 1) Faster rendering

- 2) Fewer asynchronous requests
- 3) Smaller Angular framework download size
- 4) Quick detection of template errors
- 5) Better security

9. What are Components in Angular?

Components are the basic building blocks of the user interface in an Angular application. Every component is associated with a template and is a subset of directives. An Angular application typically consists of a root component, which is the AppComponent, that then branches out into other components creating a hierarchy.



10. What are Pure Pipelines?

A pure pipe doesn't use any internal state, and the output remains the same as long as the parameters passed stay the same. Angular calls the pipe only when it detects a change in the parameters being passed. A single instance of the pure pipe is used throughout all components.

11. What is an NgModule?

NgModules are containers that reserve a block of code to an application domain or a workflow. @NgModule takes a metadata object that generally describes the way to compile the template of a component and to generate an injector at runtime. In addition, it identifies the module's components, directives, and pipes, making some of them public, through the export property so that external components can use them.

12. What are filters in Angular? Name a few of them?

Filters are used to format an expression and present it to the user. They can be used in view templates, controllers, or services. Some inbuilt filters are as follows.

date - Format a date to a specified format.

filter - Select a subset of items from an array.

Json - Format an object to a JSON string.

limitTo - Limits an array/string, into a specified number of elements/characters.

lowercase - Format a string to lowercase.

13. What is view encapsulation in Angular?

View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa. Angular provides three encapsulation strategies:

- 1) Emulated - styles from the main HTML propagate to the component.
- 2) Native - styles from the main HTML do not propagate to the component.
- 3) None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page.

14. What are controllers in Angular?

AngularJS controllers control the data of AngularJS applications. They are regular JavaScript Objects. The ng-controller directive defines the application controller.

15. What do you understand by scope in Angular?

The scope in Angular binds the HTML, i.e., the view, and the JavaScript, i.e., the controller. It as expected is an object with the available methods and properties. The scope is available for both the view and the controller. When you make a controller in Angular, you pass the \$scope object as an argument.

16. Explain the lifecycle hooks in Angular?

In Angular, every component has a lifecycle. Angular creates and renders these components and also destroys them before removing them from the DOM. This is achieved with the help of lifecycle hooks. Here's the list of them -

ngOnChanges() - Responds when Angular sets/resets data-bound input properties.

ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/

ngDoCheck() - Detect and act upon changes that Angular can't or won't detect on its own.

ngAfterContentInit() - Responds after Angular projects external content into the component's view.

ngAfterContentChecked() - Respond after Angular checks the content projected into the component.

ngAfterViewInit() - Respond after Angular initializes the component's views and child views.

ngAfterViewChecked() - Respond after Angular checks the component's views and child views.

ngOnDestroy - Cleanup just before Angular destroys the directive/component.

17. What is String Interpolation in Angular?

String Interpolation is a one-way data-binding technique that outputs the data from TypeScript code to HTML view. It is denoted using double curly braces. This template expression helps display the data from the component to the view.

```
{{ data }}
```

18. What are Template statements in Angular?

Template statements are properties or methods used in HTML for responding to user events. With these template statements, the application that you create or are working on, can have the capability to engage users through actions such as submitting forms and displaying dynamic content.

For example,

```
<button (click)="deleteHero()">Delete hero</button>
```

The template here is deleteHero. The method is called when the user clicks on the button.

19. What is the difference between AOT and JIT, AOT vs JIT?

Ahead of Time (AOT) compilation converts your code during the build time before the browser downloads and runs that code. This ensures faster rendering to the browser. To specify AOT compilation, include the --aot option with the ng build or ng serve command.

The Just-in-Time (JIT) compilation process is a way of compiling computer code to machine code during execution or run time. It is also known as dynamic compilation. JIT compilation is the default when you run the ng build or ng serve CLI commands.

20. What is @Component Decorator in Angular?

TypeScript class is one that is used to create components. This genre of class is then decorated with the "@Component" decorator. The decorator's purpose is to accept a metadata object that provides relevant information about the component.:

21. What are Services in Angular?

Angular Services perform tasks that are used by multiple components. These tasks could be data and image fetching, network connections, and database management among others. They perform all the operational tasks for the components and avoid rewriting of code. A service can be written once and injected into all the components that use that service.

22. What are Promises and Observables in Angular?

While both the concepts deal with Asynchronous events in Angular, Promises handle one such event at a time while observables handle a sequence of events over some time.

Promises - They emit a single value at a time. They execute immediately after creation and are not cancellable. They are Push errors to the child promises.

Observables - They are only executed when subscribed to them using the subscribe() method. They emit multiple values over a period of time. They help perform operations like forEach, filter, and retry, among others. They deliver errors to the subscribers. When the unsubscribe() method is called, the listener stops receiving further values.

23. What is ngOnInit? How is it defined?

ngOnInit is a lifecycle hook and a callback method that is run by Angular to indicate that a component has been created. It takes no parameters and returns a void type.

```
export class MyComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
    //....
  }
}
```

24. How to use ngFor in a tag?

The ngFor directive is used to build lists and tables in the HTML templates. In simple terms, this directive is used to iterate over an array or an object and create a template for each element.

```
<ul>
  <li *ngFor = "let items in itemlist"> {{ item }} </li>
</ul>
```

“Let item” creates a local variable that will be available in the template

“Of items” indicates that we are iterating over the items iterable.

The * before ngFor creates a parent template.

25. What is Eager and Lazy loading?

Eager loading is the default module-loading strategy. Feature modules under Eager loading are loaded before the application starts. This is typically used for small size applications.

Lazy loading dynamically loads the feature modules when there's a demand. This makes the application faster. It is used for bigger applications where all the modules are not required at the start of the application.

26. What type of DOM does Angular implement?

DOM (Document Object Model) treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

Angular uses the regular DOM. This updates the entire tree structure of HTML tags until it reaches the data to be updated. However, to ensure that the speed and performance are not affected, Angular implements Change Detection.

27. Explain components, modules and services in Angular?

Components, modules and services are the three fundamental building blocks in Angular. Components are the smallest, self-contained units in an Angular application. They are typically used to represent a view or UI element, such as a button or a form input field.

Code example:

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

```
@Component({
```

```
  selector: 'app-test',
```

```
  templateUrl: './test.component.html',
```

```
  styleUrls: ['./test.component.css']
```

```
})
```

```
export class TestComponent implements OnInit {
```

```
  constructor() {}
```

```
  ngOnInit() {
```

```
  }
```

```
}
```

Modules are larger units that group together one or more related components. Services are singleton objects that provide specific functionality throughout an Angular application, such as data access or logging.

Code example:

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

```
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';
```

```
import { TestComponent } from './test/text.component';
```

```
@NgModule({
```

```
  declarations: [
```

```
    AppComponent,
```

```
    TestComponent
```

```
  ],
```

```
  imports: [
```

```
    BrowserModule
```

```
  ],
```

```
  providers: [],
```

```
  bootstrap: [AppComponent]
```

```
})
```

```
export class AppModule { }
```

Each component in Angular has its own isolated scope. This means that a component's dependencies (services, other components, etc.) are not accessible to any other component outside of its own scope. This isolation is important for ensuring modularity and flexibility in an Angular application.

Services, on the other hand, are not isolated and can be injected into any other unit in an Angular application (component, module, service, etc.). This makes them ideal for sharing data or functionality across the entire app.

Code example:

```
import { Injectable } from '@angular/core';
```

```
    @Injectable({
```

```
        providedIn: 'root'
```

```
    })
```

```
    export class TestServiceService {
```

```
        constructor() { }
```

```
    }
```

When designing an Angular application, it is important to keep these three building blocks in mind. Components should be small and self-contained, modules should group together related components, and services should provide shared functionality across the entire app. By following this design principle, you can create an Angular application that is modular, flexible, and easy to maintain.

28. Explain MVVM architecture?

MVVM architecture is an architectural pattern used mainly in software engineering. It stands for Model-View-ViewModel. MVVM is a variation of the traditional MVC (Model-View-Controller) software design pattern. The main difference between the two is that MVVM separates the user interface logic from the business logic, while MVC separates the data access logic from the business logic. This separation of concerns makes it easier to develop, test, and maintain software applications.

The Model layer in MVVM architecture is responsible for storing and managing data. It can be a database, a web service, or a local data source. The View layer is responsible for displaying data to the user. It can be a graphical user interface (GUI), a command-line interface (CLI), or a web page. The ViewModel layer is responsible for handling user input and updating the View layer accordingly. It contains the business logic of the application.

MVVM architecture is often used in conjunction with other software design patterns, such as Model-View-Presenter (MVP) and Model-View-Controller (MVC). These patterns can be used together to create a complete software application.

MVVM architecture is a popular choice for modern software applications. It allows developers to create applications that are more responsive and easier to maintain. Additionally, MVVM architecture can be used to create applications that can be easily ported to different platforms.

29. What are some disadvantages of using Angular??

Although Angular provides quite a lot of benefits, there are some disadvantages of using it as well. They are as follows:

- 1) Getting good SEO results on an Angular application can be a bit difficult and may need a bit of configuration.
- 2) Angular has a lot of features packed into it, so getting to know each of them and learning how to use them effectively together can be a little difficult.
- 3) Angular can add quite a lot of weight to your JavaScript bundle, so using it for smaller projects can be very inefficient and may significantly increase the load size.

30. What are the differences between Angular decorator and annotation??

In Angular, decorators are design patterns that help in the modification or decoration of the respective classes without making changes in the actual source code.

Annotations, on the other hand, are used in Angular to build an annotation array. They use the Reflective Metadata library and are a metadata set of the given class..

31. What are pipes in Angular?

When we are trying to output some dynamic data in our templates, we may sometimes need to manipulate or transform the data before it is put into our templates. Though there are several ways of doing that, in Angular, using pipes is the most preferred way. A pipe is just a simple function, which we can use with expressions in our templates.

Pipes are extremely useful as we can use them throughout our application after declaring them just once and registering them with the Angular framework. Some of the most common built-in pipes in Angular are UpperCasePipe, LowerCasePipe, CurrencyPipe, etc..

32. What is Angular Router?

Routing in a single-page frontend application is the task of responding to the changes in the URL made by adding and removing content from the application. This is a complicated task as we first need to

intercept a request that changes the browser's URL as we do not wish for the browser to reload. Then, we need to determine which content to remove and which content to add, and finally, we have to change the browser's URL as well to show the user the current page they are on.

As we can see, this can be very difficult to implement, especially in multiple applications. That is why Angular comes with a full routing solution for a single-page application. In this, we can define routes with matching components and let Angular handle the routing process.

33. What is server-side rendering in Angular?

In a normal Angular application, the browser executes our application, and JavaScript handles all the user interactions. However, because of this, sometimes, if we have a large application with a big bundle size, our page's load speed is slowed down quite a bit as it needs to download all the files, parse JavaScript, and then execute it. To overcome this slowness, we can use server-side rendering, which allows us to send a fully rendered page from the server that the browser can display and then let the JavaScript code take over any subsequent interactions from the user.

34. Differentiate between unit testing and end-to-end testing??

Unit testing ensures that isolated sections of code are working properly.

End-to-end testing entails inspecting complete sets of components to ensure that they operate properly together and that the application performs as expected. End-to-end tests imitate user interactions to ensure that a program is working properly.

35. What are new features in Angular 15?

- 1) Standalone components are stable!
- 2) HTTP with provideHttpClient
- 3) Directive composition API
- 4) NgOptimizedImage is stable
- 5) Dependency injection

The providedIn: NgModule syntax of the @Injectable() decorator is now deprecated.

- 6) Router

The router now auto-unwraps default exports from lazy-loaded modules, routes or components.

- 7) Forms

Some utility functions have been added to the forms package: isFormControl, isFormGroup, isFormRecord, isFormArray.

- 8) Common

Angular v13 introduced the token DATE_PIPE_DEFAULT_TIMEZONE to configure the default timezone of the DatePipe (see our blog post about v13).

This token has been deprecated in v15 and replaced with `DATE_PIPE_DEFAULT_OPTIONS` which accepts an object with a `timezone` property and a `dateFormat` property to specify the default date format that the pipe should use.

9) Devtools

The devtools now allow you to inspect the source code of a directive

10) Angular CLI v15

36. Can we make an Angular application render on the server-side?

Yes, we can, with Angular Universal. The benefits of using Angular Universal are:

Better User Experience: Allows users to see the view of the application instantly.

Better SEO: Universal ensures that the content is available on every search engine leading to better SEO.

Loads Faster: Render pages are available to the browsers sooner, so the server-side application loads faster.

37. Angular vs jQuery?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What is Angular loader?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

6. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

Redux Questions

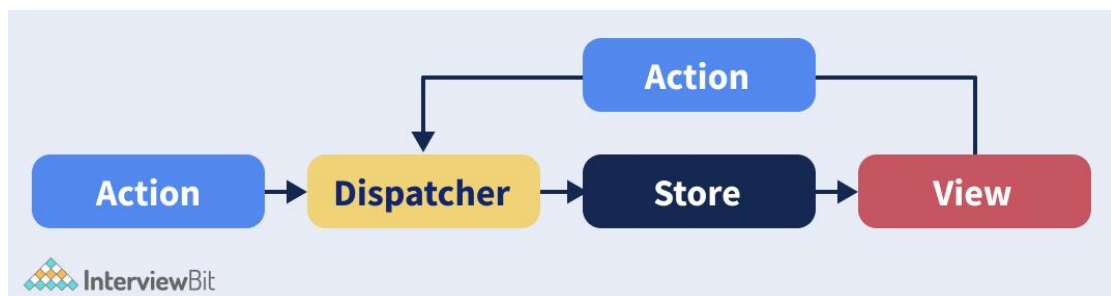
1. What is Redux?

Redux is an open-source library made using the scripting language JavaScript. Redux's primary use lies in managing and centralizing application state and it is usually used along with JavaScript libraries, for instance, React or Angular in order to build UIs (User Interfaces). It is a predictable state container for applications built using JavaScript. It is based on the Flux design pattern. Redux is very small in size (around 2 kilobytes) and has no dependencies.

2. What is Flux?

Flux is an application design paradigm just like the Model View Controller design pattern. Flux is nothing but a new kind of architecture that complements React and the concept of Unidirectional Data

Flow. The image given below shows how the workflow between dispatcher, stores and views components with distinct inputs and outputs are in Flux:



3. Difference between Redux and Flux.

The main differences in the comparison: Redux vs Flux are as follows:

COMPARISON PARAMETER	REDUX	FLUX
Number of stores per application	Redux includes a single Store per application. Rather than placing state information in multiple Stores across the app, Redux keeps everything in one region of the application	Flux includes multiple Stores per application.
Architecture	Redux is an open-source JavaScript library used for creating User Interfaces.	Flux's architecture is designed to build client-side web apps.
Place where Business Logic of the Application Resides	In Redux, the business logic of the application resides in the Reducer.	In Flux, the business logic of the application resides in the Store.

4. What is Redux in React js?

Redux in React is the official React binding for Redux which allows the components in React to read data from a Redux Store, and dispatch Actions to the Store for updating the data. The purpose of Redux is to help applications scale well by providing means to manage the state via a unidirectional data flow model.

5. State the core principles of Redux.

The core principles of Redux are as follows:

- **Single source of truth:** The global state of our application is always put away in an object tree inside one store.
- **The state is read-only:** The only way to change the state of our application is by emitting an action, an object explaining what has happened.
- **Changes are made with pure functions:** This principle means that in order to define how the state tree is being transformed by the actions, we have to write pure reducers.

6. What are the advantages of using Redux?

Some of the advantages of using Redux are as follows:

- Redux provides extremely easy state transfer between the components.
- The states are always predictable in Redux and its maintenance is relatively easy.
- Debugging and testing code in Redux is simple through logging behaviour and status.
- Redux provides great performance. It might occur to us that keeping the application's state global would result in bad performance. However, usually, that is not the case as React Redux implements a lot of performance optimizations internally so that our own connected component only re-renders when it actually needs to.
- Redux also offers state persistence by storing the application's state to local storage and restoring it after a refresh.

7. Is it true that Redux can only be used with React?

No, it is not true that Redux can only be used with React. Redux is being used as a data store for lots of UI layers. There are bindings available in Redux for React, Angular, Angular 2, Vue, etc.

8. What do you understand about Redux Toolkit?

Redux Toolkit is Redux's official, opinionated, batteries included toolset for efficient Redux development. It also consists of the most widely used Redux add-ons, for instance, Redux Thunk for asynchronous logic, Reselect for writing selector functions and many more for making development easy for developers and saving them time.

9. What are some of the major features of Redux DevTools?

Some of the major features of Redux DevTools are as follows:

- Redux DevTools is nothing but a time travel environment that makes it possible for us to live edit in Redux with a variety of functionalities like action replay, hot reloading, and customizable UI.
- Redux DevTools makes it possible for us to inspect all the states and action payload. We can go back into the time simply by cancelling the actions.
- Each stage action is re-evaluated in case we change the code of the reducer.
- We can also continue our debug sessions across page reloads with the help of `persistState()` store enhancer.

10. Is it necessary to keep all the component states in the Redux store?

No, it is not necessary to keep all the component states in the Redux store. We have to keep your application state as small as possible and therefore, we should do it only if it makes a difference for us to keep something there or maybe if it makes the use of Dev Tools easier.

11. Highlight the key differences between `mapStateToProps()` and `mapDispatchToProps()`?

The key differences between `mapStateToProps()` and `mapDispatchToProps()` are given below:

mapStateToProps()	mapDispatchToProps()
The mapStateToProps() method is used to render the stored data to the component.	The mapDispatchToProps() method is used to render the action creators with props to the component.
The entirety of the results of the mapStateToProps() method is a plain object which is later merged into the component's prop.	In the mapDispatchToProps() method, each action creator is wrapped in the dispatcher call so that they can be called upon directly and later merged into the component's prop.
This method's use is to connect the redux state to the props of the react component.	This method's use is to connect redux actions to the react props.

12. What do you understand by an action in Redux's architecture?

In the Redux architecture, actions are nothing but the plain JavaScript objects which contain a type field. They can be thought of as an event that is used to describe something which has happened in the application. Actions contain only a tiny bit of information that is required to mention what has happened.

13. Give an example of the usage of Actions in Redux's architecture.

An example of the usage of Actions in Redux's architecture is given below:

```
const addingTodoAction = {
  type: 'ADD',
  payload: 'Do-homework'
}
```

14. Where can we use Redux?

Redux is primarily being used along with React. However, we can also use it along with Angular, Vue, Meteor, etc. using the bindings it has to offer to us.

15. What are constants in Redux?

Constants in Redux help us in easily finding all the usages of a particular functionality across our entire project when we are using an Integrated Development Environment (IDE). Using constants, we can avoid silly bugs caused because of typing errors or typos as it shows a "ReferenceError" whenever a typo is made.

16. Show using code how constants can be used in Redux.

First of all, we can store all the constants in a single file in our project named constants.js or something else as follows:

```
export const ADDING_TODO = 'ADDING_TODO';
export const DELETING_TODO = 'DELETING_TODO';
export const EDITING_TODO = 'EDITING_TODO';
export const COMPLETING_TODO = 'COMPLETING_TODO';
export const COMPLETING_ALL = 'COMPLETING_ALL';
export const CLEARING_COMPLETED = 'CLEARING_COMPLETED';
```

After storing the constants in one place, we can use them in two ways in our project:

- During actions creation (in actions.js file of our project):

```
import { DELETING_TODO } from './constants';

export function deletingTodo(text) {
  return { type: DELETING_TODO, text };
}
```

- In Reducers (in reducer.js file of our project):

```
import { EDITING_TODO } from './constants';

export default (state = [], action) => {
  switch (action.type) {
    case EDITING_TODO:
      return [
        ...state,
        {
          text: action.text,
          completed: false
        }
      ];
    default:
      return state
  }
};
```

17. What are reducers in Redux's architecture?

Reducers in Redux's architecture are pure functions that are used to take the previous state and an action and return the next state. Its syntax is given below:

```
(previous_state, action) => new_state
```

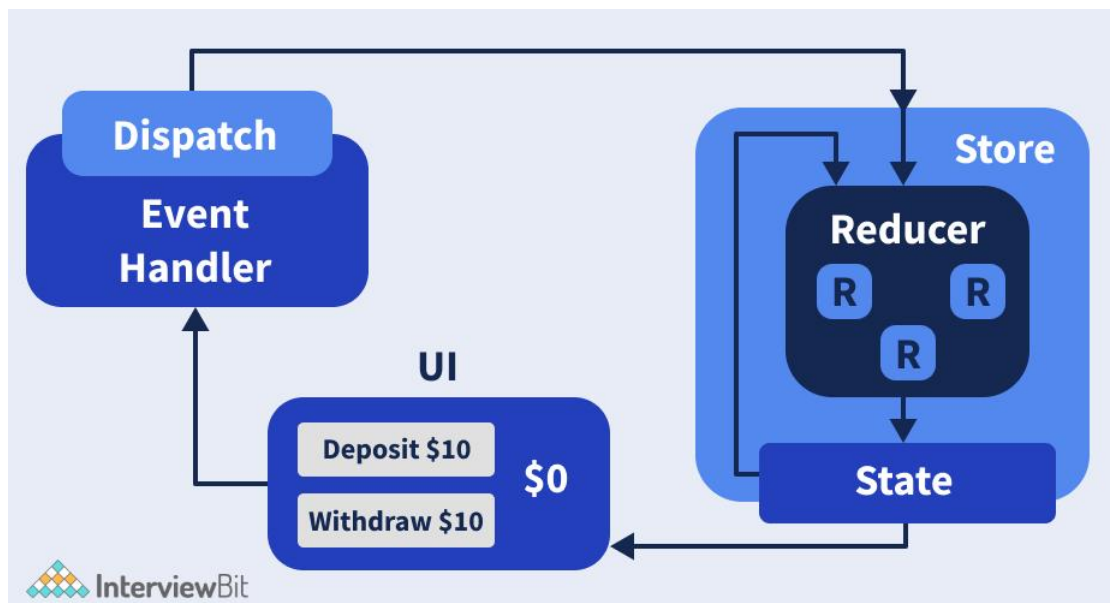
18. Show with an example how reducers are used in Redux.

An example of how reducers are used in Redux is given below:

```
const initial_state = { value: 0 }
function countReducer(state = initial_state, action) {
  // Checking to see if the reducer cares about this action
  if (action.type === 'counter/incremented') {
    // If the action is of type "counter" or "incremented", we make a copy of `state`
    return {
      ...state,
      // We also update the copy with the new value
      value: state.value + 1
    }
  }
  // If not, we return the original state unchanged
  return state
}
```

19. Explain the typical data flow in an application made using React and Redux (Redux Lifecycle for an application).

The typical data flow in Redux starts with a call back from the User Interface component which dispatches an action with a payload. After that, the reducers intercept and receive the dispatched actions, generating a new application state. After that, the actions are propagated down through a hierarchy of components from the Redux store.



20. What is the mental model of redux saga?

Redux Saga functions as a separate thread in our programme which is solely responsible for side effects. Redux Saga is a redux middleware. In other words, it means that it can be started, paused, and aborted from the main application using standard Redux actions, has access to the entire Redux application state, and can also dispatch Redux actions.

21. Describe what is meant by a "store" in Redux.

“Store” in Redux is used to carry together all the states, reducers, and actions which create the app. Some of the responsibilities of the store are as follows:

- The state of the current application from inside is held by the Redux Store.
- We can access the current state using `store.getState()`.
- We can update the state using `store.dispatch(action)`.
- We can also register listener callbacks using the `store.subscribe(listener)`.

22. Name all the Redux Store methods.

All the Redux Store Methods are as follows:

- `getState()`
- `subscribe(listener)`
- `dispatch(action)`
- `replaceReducer(nextReducer)`

23. Give an example depicting the usage of a Redux store.

An example depicting the usage of a Redux store is given below:

```
import { createStore } from 'redux'
const store = createStore(todos, ['Use Redux'])
function deletingTodo(text) {
  return {
    type: 'DELETING_TODO',
    text
  }
}
store.dispatch(deletingTodo('Do the homework'))
store.dispatch(deletingTodo('Buy coffee'))
```

24. Explain with an example how to set the initial state in Redux.

In order to set the initial state in Redux, we have to pass the initial state as the second argument to createStore as shown below:

```
const rootReducer = combineReducers({
  todos: todos,
  visibilityFilter: visibilityFilter
});

const initialState = {
  todos: [{id:100, name:'ritik', completed: true}]
};

const store = createStore(
  rootReducer,
  initialState
);
```

25. What do you understand about Redux Thunk?

Using Redux Thunk middleware, we can write action creators returning a function instead of an action. This thunk can postpone the dispatch of an action, or do conditional dispatchment. The arguments passed to the inner function are the store methods dispatch and getState(). In the event of an action creator returning a function, the function gets executed by the Redux Thunk middleware and it does not have to be pure. In other words, the function is allowed to have side effects, including executing asynchronous API calls. It can even dispatch actions. Redux thunk is used to delay the dispatch of an action, or to dispatch in the event of a certain condition being met. At the time of dispatch of a function instead of an action object, if Redux Thunk middleware is enabled, the middleware will call that function with the dispatch method itself as the first argument.

26. What are the workflow features in Redux?

The workflow features in Redux are as follows:

- **Reset:** The state of the store is allowed to be reset.
- **Revert:** Revert or Rollback to the last committed state is allowed.
- **Sweep:** Every disabled action which we have fired unintentionally will be removed.

- **Commit:** The current state is made the initial state.

27. Differentiate between Relay and Redux.

The key differences between Relay and Redux are given below:

Relay	Redux
The state originating from the server is taken care of by Relay.	All the states of the application are taken care of by Redux.
Relay can be used for caching and optimizing the data.	Redux is not responsible for handling data fetching (it can be done manually though).

28. How can we use connect from react redux?

In order to use connect from React Redux, we will have to follow a couple of steps to use our store in our container:

- Firstly, we use the `mapStateToProps()`: This would map the state variables from our store to the props which we specify.
- Secondly, we connect the above props to our container: The object returned by the `mapStateToProps` component is connected to the container.

A sample code for connect from react-redux is given below:

```
import React from 'react';
import { connect } from 'react-redux';

class App extends React.Component {
  render() {
    return <div>{this.props.containerData}</div>;
  }
}

function mapStateToProps(state) {
  return { containerData: state.appData };
}

export default connect(mapStateToProps)(App);

function mapStateToProps(state) {
  return { containerData: state.data };
}

export default connect(mapStateToProps)(App);
```

29. What are Redux forms? What are its major features?

Redux Forms present in React and Redux help in enabling a form in React to use Redux for storing all of its states. They can be used with raw inputs in HTML5. Redux forms work extremely well with User Interface (UI) frameworks, for instance, Material UI, React Widgets, React Bootstrap and many more.

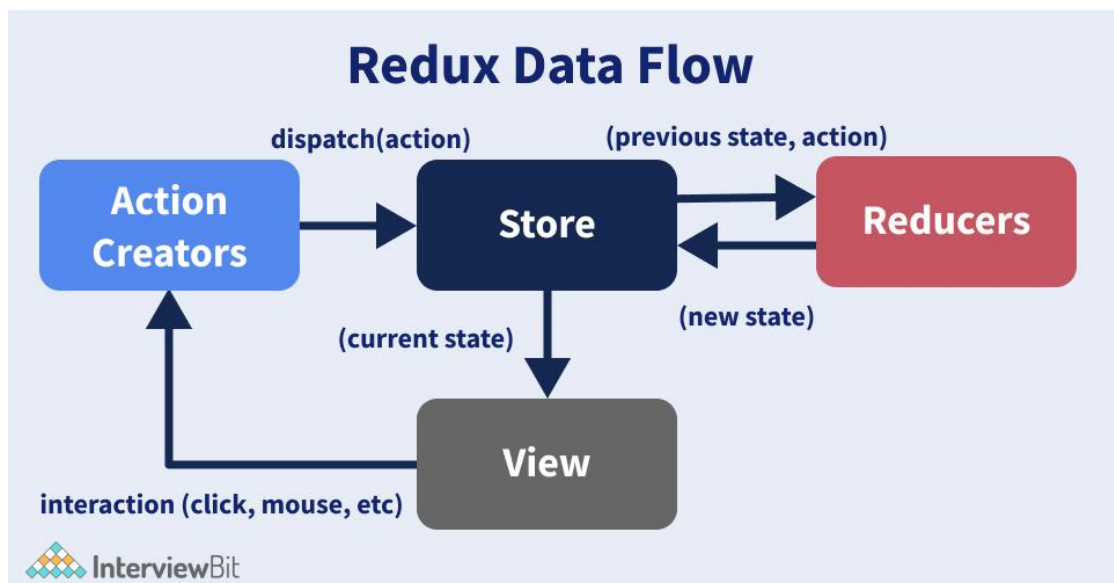
The **major features of Redux forms** are as follows:

- Field values persistence through the Redux store.
- Validation (synchronous/asynchronous) and submission.
- Formatting, parsing and normalization of field values.

30. How can we structure the top level directories in Redux?

Every Redux application has multiple top-level directories as given below:

- **Components:** Components are used for “dumb” React components unfamiliar with Redux.
- **Containers:** Containers are used for “smart” React components that are connected to Redux.
- **Actions:** Actions are used for all the action creators, where the file name should be corresponding to the part of the app.
- **Reducers:** Reducers are used for all the reducers where the file name is corresponding to the state key.
- **Store:** Stores are used for store initialization. This directory works best in small and mid-level size apps.



31. How can we access a redux store outside a react component?

For accessing a redux store outside a react component, we can export the store from the module where it has been created with createStore as done in the following example:

```
store = createStore(reducer);
export default store;
```

32. Differentiate between React Redux and React's Context API.

The key differences in the comparison: Context Api vs Redux are as follows:

React Redux	React Context API
In order to use React-Redux in our application, we need to code it separately and then merge it into the main project.	React Context can be used in the application directly.

The number of features provided by React-Redux is comparatively more than React Context.	The number of features provided by React Context is comparatively less than React-Redux.
--	--

33. What do you understand about the Redux Saga?

Redux Saga is a middleware library that can be useful for allowing a Redux store to interact with the resources outside of itself in an asynchronous manner, for example, making HTTP requests to external services, accessing browser storage, executing Input/Output operations and many more. These operations are also called side effects.

34. What are the things which we should never do inside a reducer?

The things which we should never do inside a reducer are as follows:

- Modify the argument of the reducer
- We should assure that we do not perform any side operations such as routing transitions, API calls, etc.
- We should not call non-pure functions, for instance `Date.now()`, `Math.random()`, etc.

35. How can the addition of multiple middlewares to Redux be done?

In order to add multiple middlewares to Redux, the usage of `applyMiddleware` can do the work. In `applyMiddleware`, we can pass every piece of middleware as a new argument. Therefore, our job is to just pass each piece of middleware that we want. In the example given below, we have added Redux Thunk and logger middlewares as an argument:

```
import { createStore, applyMiddleware } from 'redux'
const createStoreWithMiddleware = applyMiddleware(ReduxThunk, logger)(createStore);
```

36. Redux Saga vs Redux Thunk

Saga works like a separate thread or by a background process, So for that is only responsible for making side effects inside APIs or API calls, unlike `redux-thunk`, in some cases, which uses callbacks that lead to situations like 'callback hell'.

Redux Thunk is a middleware that lets you call action creators that return a function instead of an action object. That function receives the store's dispatch method, which is then used to dispatch regular synchronous actions inside the body of the function once the asynchronous operations have completed.

Redux Saga leverages an ES6 feature called Generators, allowing us to write asynchronous code that looks synchronous, and is very easy to test. In the saga, we can test our asynchronous flows easily and our actions stay pure. complicated, hard to learn and understand, 14kB volume but organized complicated asynchronous actions easily and make then very readable and the saga has many useful tools to deal with asynchronous actions

Next.js Questions

1) What is Next.js?

Next.js is an open-source, lightweight React.js framework that facilitates developers to build static and server-side rendering web applications. It was created by Zeit. Next.js framework is based on React, Webpack, and Babel and allows us to write server-rendered React apps easily. It doesn't require any webpack configuration and only needs `npm run dev` start building your next feature-filled web application.

2) Why is Next.js used for? / Why do world's leading companies prefer Next.js?

If you want to build a complete web application with React from scratch, you have to fulfill the following points:

- Your code has to be bundled using a bundler like webpack and transformed using a compiler like Babel.
- You have to do production optimizations such as code splitting.
- You have to pre-render some pages for performance and SEO statically. You might also want to use server-side rendering or client-side rendering.
- You might have to write some server-side code to connect your React app to your data store.

Next.js fulfills the above all requirements.

Reasons why the world's leading companies prefer Next.js:

Zero Setup: Next.js provides automatic code-splitting, filesystem-based routing, hot code reloading, and universal rendering; that's why the world's leading companies prefer it.

Fully Extensible: Next.js is fully extensible and has complete control over Babel and Webpack. It provides a customizable server, routing, and next plugins.

Ready for Production: Next.js is optimized for smaller build sizes, faster dev compilation, and many other improvements, making it a popular choice.

Next.js can Deploy Anywhere: Next.js is an open-source, lightweight React.js framework that facilitates developers to build static and server-side rendering web applications.

3) What is the process to install Next.js? / How to install Next.js?

Before installing Next.js, you must have installed Node.js on your system. Learn how to install Node.js on your system: <https://www.javatpoint.com/install-nodejs>

It requires NPM to start installing Next.js with all its dependencies.

Follow the steps given below to install Next.js:

- First, create a directory to keep the Next.js project and go into it:

```
1. mkdir my-portfolio-site  
2. cd my-portfolio-site
```

- Now, initialize this with a package.json file.
- Use the y flag by npm init -y
- Use the following syntax to install Next.js

```
1. npm install react react-dom next
```

- Update package.json with run script languages to start the initialization of the Next.js application.
- Now, find the package.json file on the root folder and add the below mentioned script

```
1. {  
2.   "scripts": {  
3.     "dev": "next",  
4.     "build": "next build",  
5.     "start": "next start"  
6.   }  
7. }
```

- After that, the filesystem is the main API. Every ".js" file becomes a route that gets automatically processed and rendered.
- Now, the process is completed, and Next.js is installed on your system.

4) What are the most prominent features of Next.js?

Following is a list of the most prominent features of Next.js that excite the developers most:

- js provides the by default and easy server rendering.
- js supports static exporting.

- It provides a Webpack-based dev environment which supports Hot Module Replacement (HMR)
- It supports automatic code-splitting for faster page loads.
- It supports simple client-side routing (page-based) or file system-based routing.
- It provides complete Webpack and Babel control.
- It provides a faster and optimized development compilation.
- It can be implemented with Express or any other Node.js HTTP server.
- You can easily customize it with your own Babel and Webpack configurations.
- It supports hot code reloading.

Besides this, Next.js also has some awesome features such as:

- Dynamic styles and themes support
- Built-in CSS vendor prefixing
- CSS Preprocessing via Plugins
- Full CSS support, no tradeoffs in power
- Its runtime size is very small. It is just 3kb (zipped from 12kb)
- It provides source maps support

5) Which types of websites most popularly use Next.js?

Next.js is a popular framework of React.js that is most popularly used for building the following types of apps and websites:

- Static Websites
- Desktop Websites
- SEO Friendly Websites
- Server Rendered Apps
- Progressive Web Apps (PWA) etc.

6) Is it possible to use Next.js with Redux?

Yes. You can easily use Next.js with Redux.

7) What is the recommended method to fetch data in Next.js?

There are multiple ways to fetch data in Next.js, but Next.js itself recommends **getInitialProps**, an async function to retrieve data from anywhere. When we use **getInitialProps** to retrieve data, it receives a context object which has the following properties:

- **pathname**- It specifies the path section of the URL.
- **query**- It is used to specify the query string section of URL parsed as an object.
- **asPath**- It specifies the string of the actual path (including the query) shows in the browser.
- **req**- It is used to specify the HTTP request object (server only).
- **res**- It is used to specify the HTTP response object (server only).
- **err**- It is used to specify the error object if any error is encountered during the rendering.

8) Give an example to demonstrate how do you set up CDN in Next.js?

To setup CDN in Next.js, the developers have to follow the steps given below:

To start, we have to first set up the "assetPrefix" setting and configure our CDN origin to support and resolve the domain that our Next.js is hosted on.

```
1. const isProd = process.env.NODE_ENV === 'production';
2. module.exports = {
3. // You may only need to add assetPrefix in the production.
4. assetPrefix: isProd ? 'https://cdn.mydomain.com' : "
5. };
```

If the CDN is present on a separate domain, we have to set a configuration option as following:

```
1. // next.config.js
2. module.exports = {
3. crossOrigin: 'anonymous'
4. ;
```

9) Are Create-React-App and Next.js used for the same thing?

The Create-React-App is basically React with an integrated build system. It acts like a good boilerplate, so we don't need to set up Webpack, Babel, and other dependent packages to run React. Other than that, if you require extra functionalities such as routing, server-side rendering, and so on, you just need to

add packages on top of Create-React-App. On the other hand, The Next.js is an open-source, lightweight full-stack React framework that comes bundled with an efficient build system, server-side rendering, routing, API routing, and many other awesome features that make the production environment easy.

10) How can you install and use Next.js?

There are mainly two ways to install and run Next.js on your system. If you're new to Next.js, we recommend that you make sure that your development environment is ready. Next.js is a React framework, and it requires Node.js to be installed on your system. If you don't have Node.js installed, you can install it from here: <https://www.javatpoint.com/install-nodejs>. Next.js requires Node.js version 10.13 or later. You should also have your text editor and terminal.

System Requirement for Next.js

- You must have installed Node.js version 12.0 or later.
- js supports Windows, MacOS, and Linux also.

The simplest way to install Next.js

- The simplest way to install and run Next.js on your system is by creating a new Next.js app using create-next-app. It sets up everything automatically for you. To create a project, run the following command on the Node.js console.

```
1. npx create-next-app  
2. # or  
3. yarn create next-app
```

- If you want to start with a TypeScript project, you can use the following command:

```
1. npx create-next-app --typescript  
2. # or  
3. yarn create next-app -typescript
```

- After completing the installation, follow the instructions to start the development server. You can also try to edit the pages/index.js and see the result on your browser, which by default runs on `http://localhost:3000`

Manual Installation and Setup of Next.js

- First, install the next react and react-dom in your project by running the following command.

```
1. npm install next react react-dom
2. or
3. yarn add next react react-dom
```

- Now, open the package.json file and add the following scripts:

```
1. "scripts": {
2.   "dev": "next dev",
3.   "build": "next build",
4.   "start": "next start",
5.   "lint": "next lint"
6. }
```

The above script specifies the different stages of developing an application:

- **dev** - It runs the next dev, which starts Next.js in development mode.
- **build** - It runs "next build", which builds the application for production usage.
- **start** - It runs "next start", which starts a Next.js production server.
- **lint** - It runs "next lint", which sets up Next.js' built-in ESLint configuration.

Next.js is built around the concept of pages. A page is a React Component exported from a .js, .jsx, .ts, or .tsx file in the pages directory. Pages are associated with a route based on their filename. For example, pages/about.js is mapped to /about. You can even add dynamic route parameters with the filename.

11) How can you disable the etag generation in Next.js?

Generally, we use the app.disable('etag') syntax to disable the etag generation in Next.js. But, this may not work for all static contents. So, we should use the following syntax to disable the etag for all static contents.

Syntax:

```
1. app.use(express.static(path.join(__dirname, 'public'), {
2.   etag: false
3. }));
```


12) How can you configure the build-id in Next.js?

To configure the build-id in Next.js, we must configure a static ID between our builds. So, we have to provide the "generateBuildId" function with the following configuration.

Syntax:

```
1. // next.config.js
2. module.exports = {
3.   generateBuildId: async () => {
4.     // For example get the latest git commit hash here
5.     return 'my-build-id';
6.   }
7. };
```

13) How can you create a page directory inside your project?

To create a page directory inside our project we have to populate the ./pages/index.js with the following contents:

```
1. function HomePage() {
2.   return <div>Welcome to Next.js!</div>
3. }
```

To start developing our application, we have to run the npm run dev or yarn dev command. This will start the development server on <http://localhost:3000>. Now we can visit the localhost: <http://localhost:3000> to view our application.

14) Give an example to demonstrate how to create a custom error page in Next.js?

We can create our custom error page by defining a _error.js in the pages folder. See the following example:

```
1. import React from "react";
2. class Error extends React.Component {
3.   static getInitialProps({ res, err }) {
4.     const statusCode = res ? res.statusCode : err ? err.statusCode : null;
```

```
5.     return { statusCode };
6.   }
7.   render() {
8.     return (
9.
10.      {this.props.statusCode
11.        ? `An error ${this.props.statusCode} has occurred on the server`
12.        : "An error occurred on client-side"}
13.
14.    );
15.  }
16. }
17. export default Error;
```

15) What do you understand by code splitting in Next.js?

Generally, code splitting is one of the most compelling features of webpack. This feature facilitates us to split our code into various bundles, which can be loaded only on-demand or in parallel. This is mainly used to achieve the smaller bundles and facilitates us to control resource load prioritization which finally has a great impact on the load time.

There are mainly three approaches to code splitting:

- **Entry Points:** It is used to split code using entry configuration manually.
- **Prevent Duplication:** It uses Entry dependencies or SplitChunksPlugin to dedupe and split chunks.
- **Dynamic Imports:** It splits the code via inline function calls within modules

It is mainly used to enable pages that can never load unnecessary code.

16) How can you enable AMP in Next.js?

This is an important question and is asked in many Next.js interview questions. There are two ways to enable AMP in Next.js.

- AMP-First Pages

- Hybrid AMP Pages

AMP-First Pages: The AMP-First Pages are served to the primary traffic of the website as well as traffic generated from the search engine. Use the following syntax to implement AMP-first pages.

Example:

```
1. // pages/index.js
2. import { withAmp } from 'next/amp'
3. function HomePage() {
4.   return <p> Welcome to AMP + Next.js.</p>
5. }
6. export default withAmp(HomePage)
```

Hybrid AMP Pages: The Hybrid AMP pages allow the users to have coexisted AMP version of a traditional page so that the search engines can display the AMP version or the page in different mobile search results.

See the following example to understand how to implement the Hybrid AMP to pages:

Example:

```
1. // pages/index.js
2. import { withAmp } from 'next/amp'
3. function HomePage() {
4.   return <p> Welcome to AMP + Next.js.</p>
5. }
6. export default withAmp(HomePage)
```

17) Is it possible to host Next.js in a web server like Nginx?

Next.js is not as simple as static html files. It requires an application server that runs Node.js to deploy and run a Next.js application. Here, we get requests that have to be processed on the server.

Typescript Questions

Disadvantage of TypeScript?

- TypeScript sets aside a long effort to incorporate the code.
- TypeScript doesn't uphold theoretical classes.
- When utilizing an outsider library, there should be a definition document, and now and again it's not generally accessible.
- Nature of type definition documents is a worry.
- At whatever point TypeScript needs to run in a program, there should even now be a gathering step to change TypeScript to JavaScript.
- Enrolling TypeScript engineers may be a genuine annoyance, since a portion of the designers are not keen on learning TypeScript, being entirely alright with utilizing JavaScript. Groups that as of now use JavaScript, become profitable with TypeScript after around 2-3 months, and conversant in TypeScript simply after about a large portion of a year. So instructing staff can likewise be an obstacle.
- TypeScript isn't completely coexpressive with JavaScript. Missing highlights include: HOFs, Composition, Generics with Higher key.

What are the primitive types in TypeScript?

TypeScript has three primitive types that are frequently used: string, number, and boolean. These correspond to the similarly named types in JavaScript.

- **string:** represents text values such as "javascript", "typescript", etc.
- **number:** represents numeric values like 1, 2, 32, 43, etc.
- **boolean:** represents a variable that can have either a 'true' or 'false' value.

When creating a variable, there are two main ways TypeScript assigns a type:

- Explicit
- Implicit

Explain how the arrays work in TypeScript.

We use arrays to store values of the same type. Arrays are ordered and indexed collections of values. The indexing starts at 0, i.e., the first element has index 0, the second has index 1, and so on.

Here is the syntax to declare and initialize an array in TypeScript.

```
let values: number[] = [];  
values[0] = 10;  
values[1] = 20;  
values[2] = 30;
```

You can also create an array using the short-hand syntax as follows:

```
let values: number[] = [15, 20, 25, 30];
```

TypeScript provides an alternate syntax to specify the Array type.

```
let values: Array<number> = [15, 20, 25, 30];
```

Why we use TypeScript?

The reason is that Typescript has some of these advantages.

- TypeScript always points out the compilation errors at the time of development (pre-compilation). Because of this getting runtime errors is less likely, whereas JavaScript is an interpreted language.
- TypeScript supports static/strong typing. This means that type correctness can be checked at compile time. This feature is not available in JavaScript.
- TypeScript is nothing but JavaScript and some additional features i.e. ES6 features. It may not be supported in your target browser but the TypeScript compiler can compile the .ts files into ES3, ES4, and ES5 also.

What is any type, and when to use it?

There are times when you want to store a value in a variable but don't know the type of that variable in advance. For example, the value is coming from an API call or the user input. The 'any' type allows you to assign a value of any type to the variable of type any.

TypeScript assumes a variable is of type **any** when you don't explicitly provide the type, and the compiler cannot infer the type from the surrounding context.

What is void, and when to use the void type?

The void indicates the absence of type on a variable. It acts as the opposite type to any. It is especially useful in functions that don't return a value.

If a variable is of type void, you can only assign the null or undefined values to that variable.

What is an unknown type, and when to use it in TypeScript?

The unknown type is the type-safe counterpart of any type. You can assign anything to the unknown, but the unknown isn't assignable to anything but itself and any, without performing a type assertion or a control-flow-based narrowing. You cannot perform any operations on a variable of an unknown type without first asserting or narrowing it to a more specific type.

Consider the following example. We create the foo variable of unknown type and assign a string value to it. If we try to assign that unknown variable to a string variable bar, the compiler gives an error.

You can narrow down a variable of an unknown type to something specific by doing typeof checks or comparison checks or using type guards.

What are the different keywords to declare variables in TypeScript?

var: Declares a function-scoped or global variable. You can optionally set its value during the declaration. Its behavior and scoping rules are similar to the var keyword in JavaScript.

let: Declares a block-scoped local variable. Similar to var, you can optionally set the value of a variable during the declaration.

const: Declares a block-scoped constant value that cannot be changed after it's initialized.

Provide the syntax of a function with the type annotations.

Functions are blocks of code to perform a specific code. Functions can optionally take one or more arguments, process them, and optionally return a value.

How to create objects in TypeScript?

Objects are dictionary-like collections of keys and values. The keys have to be unique. They are similar to arrays and are also sometimes called associative arrays. However, an array uses numbers to index the values, whereas an object allows you to use any other type as the key.

In TypeScript, an Object type refers to any value with properties. It can be defined by simply listing the properties and their types.

For example,

```
let pt: { x: number; y: number } = {  
  x: 10,  
  y: 20  
};
```

How to specify optional properties in TypeScript?

An object type can have zero or more optional properties by adding a '?' after the property name.

```
let pt: { x: number; y: number; z?: number } = {  
  x: 10,  
  y: 20  
};  
console.log(pt);
```

In the example above, because the property 'z' is marked as optional, the compiler won't complain if we don't provide it during the initialization.

Explain the concept of null and its use in TypeScript.

In programming, a null value indicates an absence of value. A null variable doesn't point to any object. Hence you cannot access any properties on the variable or call a method on it.

In TypeScript, the null value is indicated by the 'null' keyword.

What is undefined in TypeScript?

When a variable is declared without initialization, it's assigned the undefined value. It's not very useful on its own. A variable is undefined if it's declared, but no value has been assigned to it. In contrast, null is assigned to a variable, and it represents no value.

Explain the purpose of the never type in TypeScript.

As the name suggests, the never type represents the type of values that never occur. For example, a function that never returns a value or that always throws an exception can mark its return type as never.

```
function error(message: string): never {  
  throw new Error(message);  
}
```

You might wonder why we need a 'never' type when we already have 'void'. Though both types look similar, they represent two very different concepts.

A function that doesn't return a value implicitly returns the value undefined in JavaScript. Hence, even though we are saying it's not returning anything, it's returning 'undefined'. We usually ignore the return value in these cases. Such a function is inferred to have a void return type in TypeScript.

```
// This function returns undefined  
function greet(name: string) {  
  console.log(`Hello, ${name}`);  
}  
  
let greeting = greet("David");  
console.log(greeting); // undefined
```

In contrast, a function that has a never return type never returns. It doesn't return undefined, either. There are 2 cases where functions should return never type:

1. In an unending loop e.g a while(true){} type loop.
2. A function that throws an error e.g function foo(){throw new Exception('Error message')}

What is the typeof operator? How is it used in TypeScript?

Similar to JavaScript, the typeof operator in TypeScript returns the type of the operand as a string.

```
console.log(typeof 10); // "number"  
  
console.log(typeof 'foo'); // "string"  
  
console.log(typeof false); // "boolean"
```

```
console.log(typeof bar); // "undefined"
```

In TypeScript, you can use the `typeof` operator in a type context to refer to the type of a property or a variable.

```
let greeting = "hello";  
let typeOfGreeting: typeof greeting; // similar to let typeOfGreeting: string
```

What are the rest parameters and arguments in TypeScript?

A rest parameter allows a function to accept an indefinite number of arguments as an array. It is denoted by the `'...'` syntax and indicates that the function can accept one or more arguments.

```
function add(...values: number[]) {  
  let sum = 0;  
  values.forEach(val => sum += val);  
  return sum;  
}  
const sum = add(5, 10, 15, 20);  
console.log(sum); // 50
```

In contrast, the rest arguments allow a function caller to provide a variable number of arguments from an array. Consider the following example.

```
const first = [1, 2, 3];  
const second = [4, 5, 6];  
  
first.push(...second);  
console.log(first); // [1, 2, 3, 4, 5, 6]
```

What is parameter destructuring?

Parameter destructuring allows a function to unpack the object provided as an argument into one or more local variables.

```
function multiply({ a, b, c }: { a: number; b: number; c: number }) {  
  console.log(a * b * c);  
}  
  
multiply({ a: 1, b: 2, c: 3 });
```

You can simplify the above code by using an interface or a named type, as follows:

```
type ABC = { a: number; b: number; c: number };  
  
function multiply({ a, b, c }: ABC) {  
  console.log(a * b * c);  
}  
  
multiply({ a: 1, b: 2, c: 3 });
```

Explain the TypeScript class syntax.

TypeScript fully supports classes. The TypeScript syntax for class declaration is similar to that of JavaScript, with the added type support for the member declarations.

Here is a simple class that defines an Employee type.


```
class Employee {
  name: string;
  salary: number;

  constructor(name: string, salary: number) {
    this.name = name;
    this.salary = salary;
  }
  promote(): void {
    this.salary += 10000;
  }
}
```

You can create an instance (or object) of a class by using the new keyword.

```
// Create a new employee
let john = new Employee("John", 60000);

console.log(john.salary); // 60000
john.promote();
console.log(john.salary); // 70000
```

Explain the arrow function syntax in TypeScript.

Arrow functions provide a short and convenient syntax to declare functions. They are also called lambdas in other programming languages.

Consider a regular function that adds two numbers and returns a number.

```
function add(x: number, y: number): number {
  let sum = x + y;
  return sum;
}
```

Using arrow functions syntax, the same function can be defined as:

```
let add = (x: number, y: number): number => {
  let sum = x + y;
  return sum;
}
```

You can further simplify the syntax by getting rid of the brackets and the return statement. This is allowed when the function body consists of only one statement. For example, if we remove the temporary sum variable, we can rewrite the above function as:

```
let add = (x: number, y: number): number => x + y;
```

Arrow functions are often used to create anonymous callback functions in TypeScript. Consider the example below that loops over and filters an array of numbers and returns an array containing multiples of five. The filter function takes an arrow function.

```
let numbers = [3, 5, 9, 15, 34, 35];

let fiveMultiples = numbers.filter(num => (num % 5) === 0);

console.log(fiveMultiples); // [5, 15, 35]
```

Provide the syntax for optional parameters in TypeScript.

A function can mark one or more of its parameters as optional by suffixing its name with '?'. In the example below, the parameter `greeting` is marked optional.

```
function greet(name: string, greeting?: string) {
  if (!greeting)
    greeting = "Hello";

  console.log(`${greeting}, ${name}`);
}

greet("John", "Hi"); // Hi, John
greet("Mary", "Hola"); // Hola, Mary
greet("Jane"); // Hello, Jane
```

What is the purpose of the `tsconfig.json` file?

A `tsconfig.json` file in a directory marks that directory as the root of a TypeScript project. It provides the compiler options to compile the project.

Here is a sample `tsconfig.json` file:

```
{
  "compilerOptions": {
    "module": "system",
    "noImplicitAny": true,
    "removeComments": true,
    "outFile": "../built/local/tsc.js",
    "sourceMap": true
  },
  "include": ["src/**/*.ts"],
  "exclude": ["node_modules", "**/*.spec.ts"]
}
```

Explain the different variants of the `for` loop in TypeScript.

TypeScript provides the following three ways to loop over collections.

- 'for' loop

```
let values = [10, "foo", true];

for(let i=0; i<values.length; i++) {
  console.log(values[i]); // 10, "foo", true
}
```

- 'forEach' function

```
let values = [10, "foo", true];
values.forEach(val => {
  console.log(val); // 10, "foo", true
})
```

- 'for..of' statement

```
let values = [10, "foo", true];
for (let val of values) {
```

```
console.log(val); // 10, "foo", true
}
```

Explain the symbol type in TypeScript.

Symbols were introduced in ES6 and are supported by TypeScript. Similar to numbers and strings, symbols are primitive types. You can use Symbols to create unique properties for objects.

You can create symbol values by calling the Symbol() constructor, optionally providing a string key.

```
let foo = Symbol();
let bar = Symbol("bar"); // optional string key
```

A key characteristic of symbols is that they are unique and immutable.

```
let foo = Symbol("foo");
let newFoo = Symbol("foo");

let areEqual = foo === newFoo;
console.log(areEqual); // false, symbols are unique
```

Explain how optional chaining works in TypeScript.

Optional chaining allows you to access properties and call methods on them in a chain-like fashion. You can do this using the ‘?’ operator.

TypeScript immediately stops running some expression if it runs into a ‘null’ or ‘undefined’ value and returns ‘undefined’ for the entire expression chain.

Using optional chaining, the following expression

```
let x = foo === null || foo === undefined ? undefined : foo.bar.baz();
```

can be expressed as:

```
let x = foo?.bar.baz();
```

Provide the TypeScript syntax to create function overloads.

Function overloading allows us to define multiple functions with the same name, as long as their number of parameters or the types of parameters are different.

The following example defines two overloads for the function buildDate. The first overload takes a number as a parameter, whereas the second takes three numbers as parameters. These are called overload signatures.

The body of the function also called an implementation signature, follows the overload signatures. You can't call this signature directly, as it's not visible from the outside. It should be compatible with the overload signatures.

```
function buildDate(timestamp: number): Date;
function buildDate(m: number, d: number, y: number): Date;
function buildDate(mOrTimestamp: number, d?: number, y?: number): Date {
  if (d !== undefined && y !== undefined) {
    return new Date(y, mOrTimestamp, d);
  } else {
    return new Date(mOrTimestamp);
  }
}
```

```
}  
}  
  
const d1 = buildDate(87654321);  
const d2 = buildDate(2, 2, 2);
```

What is meant by type inference?

TypeScript can infer the type of a variable when you don't provide an explicit type. This is known as type inference. This is usually done when the variables or parameters are initialized during the declaration.

For example, TypeScript knows that the variable `foo` is a string, even though we don't mention string as a type.

```
let foo = "this is a string";  
console.log(typeof foo); // "string"
```

What is meant by contextual typing?

When the TypeScript compiler uses the location (or context) of a variable to infer its type, it's called contextual typing.

In the following example, TypeScript uses the `Window.onmousedown` function type information to infer the type of the function expression on the right-hand side of the assignment. This allows it to infer the type of the `e` parameter, which does have a `button` property but not a property named `foo`.

```
window.onmousedown = function (e) {  
  console.log(e.button); //<- OK  
  console.log(e.foo); //<- Error!  
};
```

What is the purpose of `noImplicitAny`?

Usually, when we don't provide any type on a variable, TypeScript assumes 'any' type. For example, TypeScript compiles the following code, assuming the parameter '`s`' is of any type. It works as long as the caller passes a string.

```
function parse(s) {  
  console.log(s.split(' '));  
}  
parse("Hello world"); // ["Hello", "world"]
```

However, the code breaks down as soon as we pass a number or other type than a string that doesn't have a `split()` method on it. For example,

```
function parse(s) {  
  console.log(s.split(' ')); // [ERR]: s.split is not a function  
}  
parse(10);
```

`noImplicitAny` is a compiler option that you set in the `tsconfig.json` file. It forces the TypeScript compiler to raise an error whenever it infers a variable is of any type. This prevents us from accidentally causing similar errors.

Parameter 's' implicitly has an 'any' type.(7006)

```
function parse(s) {
```

```
console.log(s.split(' ')); // [ERR]: s.split is not a function
}
```

What is an interface?

An interface defines a contract by specifying the type of data an object can have and its operations. In TypeScript, you can specify an object's shape by creating an interface and using it as its type. It's also called "duck typing".

In TypeScript, you can create and use an interface as follows:

```
interface Employee {
  name: string;
  salary: number;
}

function process(employee: Employee) {
  console.log(`${employee.name}'s salary = ${employee.salary}`);
}

let john: Employee = {
  name: "John Doe",
  salary: 150000
}

process(john); // "John Doe's salary = 150000"
```

Interfaces are an effective way to specify contracts within your code as well as outside your code.

Explain the various ways to control member visibility in TypeScript.

TypeScript provides three keywords to control the visibility of class members, such as properties or methods.

- **public:** You can access a public member anywhere outside the class. All class members are public by default.
- **protected:** A protected member is visible only to the subclasses of the class containing that member. Outside code that doesn't extend the container class can't access a protected member.
- **private:** A private member is only visible inside the class. No outside code can access the private members of a class.

Does TypeScript support static classes? If not, why?

TypeScript doesn't support static classes, unlike the popular object-oriented programming languages like C# and Java.

These languages need static classes because all code, i.e., data and functions, need to be inside a class and cannot exist independently. Static classes provide a way to allow these functions without associating them with any objects.

In TypeScript, you can create any data and functions as simple objects without creating a containing class. Hence TypeScript doesn't need static classes. A singleton class is just a simple object in TypeScript.

What are abstract classes? When should you use one?

Abstract classes are similar to interfaces in that they specify a contract for the objects, and you cannot instantiate them directly. However, unlike interfaces, an abstract class may provide implementation details for one or more of its members.

An abstract class marks one or more of its members as abstract. Any classes that extend an abstract class have to provide an implementation for the abstract members of the superclass.

Here is an example of an abstract class `Writer` with two member functions. The `write()` method is marked as abstract, whereas the `greet()` method has an implementation. Both the `FictionWriter` and `RomanceWriter` classes that extend from `Writer` have to provide their specific implementation for the `write` method.

```
abstract class Writer {
  abstract write(): void;

  greet(): void {
    console.log("Hello, there. I am a writer.");
  }
}

class FictionWriter extends Writer {
  write(): void {
    console.log("Writing a fiction.");
  }
}

class RomanceWriter extends Writer {
  write(): void {
    console.log("Writing a romance novel.");
  }
}

const john = new FictionWriter();
john.greet(); // "Hello, there. I am a writer."
john.write(); // "Writing a fiction."

const mary = new RomanceWriter();
mary.greet(); // "Hello, there. I am a writer."
mary.write(); // "Writing a romance novel."
```

What are anonymous functions? Provide their syntax in TypeScript.

An anonymous function is a function without a name. Anonymous functions are typically used as callback functions, i.e., they are passed around to other functions, only to be invoked by the other function at a later point in time. For example,

```
setTimeout(function () {
  console.log('Run after 2 seconds')
}, 2000);
```

You can invoke an anonymous function as soon as it's created. It's called 'immediately invoked function execution (IIFE)', For example:

```
(function() {  
  console.log('Invoked immediately after creation');  
})();
```

What are union types in TypeScript?

A union type is a special construct in TypeScript that indicates that a value can be one of several types. A vertical bar (|) separates these types.

Consider the following example where the variable value belongs to a union type consisting of strings and numbers. The value is initialized to string "Foo". Because it can only be a string or a number, we can change it to a number later, and the TypeScript compiler doesn't complain.

```
let value: string | number = "Foo";  
value = 10; // Okay
```

However, if we try to set the value to a type not included in the union types, we get the following error.

```
value = true; // Type 'boolean' is not assignable to type 'string | number'.(2322)
```

Union types allow you to create new types out of existing types. This removes a lot of boilerplate code as you don't have to create new classes and type hierarchies.

What are intersection types?

Intersection types let you combine the members of two or more types by using the '&' operator. This allows you to combine existing types to get a single type with all the features you need.

The following example creates a new type Supervisor that has the members of types Employee and Manager.

```
interface Employee {  
  work: () => string;  
}  
  
interface Manager {  
  manage: () => string;  
}  
  
type Supervisor = Employee & Manager;  
  
// john can both work and manage  
let john: Supervisor;
```

What are type aliases? How do you create one?

Type aliases give a new, meaningful name for a type. They don't create new types but create new names that refer to that type.

For example, you can alias a union type to avoid typing all the types everywhere that value is being used.

```
type alphanumeric = string | number;
let value: alphanumeric = "";
value = 10;
```

Explain the tuple types in TypeScript.

Tuples are a special type in TypeScript. They are similar to arrays with a fixed number of elements with a known type. However, the types need not be the same.

```
// Declare a tuple type and initialize it
let values: [string, number] = ["Foo", 15];

// Type 'boolean' is not assignable to type 'string'.(2322)
// Type 'string' is not assignable to type 'number'.(2322)
let wrongValues: [string, number] = [true, "hello"]; // Error
```

Since TypeScript 3.0, a tuple can specify one or more optional types using the ? as shown below.

```
let values: [string, number, boolean?] = ["Foo", 15];
```

Explain how tuple destructuring works in TypeScript.

You can destructure tuple elements by using the assignment operator (=). The destructuring variables get the types of the corresponding tuple elements.

```
let employeeRecord: [string, number] = ["John Doe", 50000];
let [emp_name, emp_salary] = employeeRecord;
console.log(`Name: ${emp_name}`); // "Name: John Doe"
console.log(`Salary: ${emp_salary}`); // "Salary: 50000"
```

After destructuring, you can't assign a value of a different type to the destructured variable. For example,

```
emp_name = true; // Type 'boolean' is not assignable to type 'string'.(2322)
```

What are type assertions in TypeScript?

Sometimes, you as a programmer might know more about the type of a variable than TypeScript can infer. Usually, this happens when you know the type of an object is more specific than its current type. In such cases, you can tell the TypeScript compiler not to infer the type of the variable by using type assertions.

TypeScript provides two forms to assert the types.

- as syntax:

```
let value: unknown = "Foo";
let len: number = (value as string).length;
```

- <> syntax:

```
let value: unknown = "Foo";
let len: number = (<string>value).length;
```


Type assertions are similar to typecasting in other programming languages such as C# or Java. However, unlike those languages, there's no runtime penalty of boxing and unboxing variables to fit the types. Type assertions simply let the TypeScript compiler know the type of the variable.

How to enforce strict null checks in TypeScript?

Null pointers are one of the most common sources of unexpected runtime errors in programming. TypeScript helps you avoid them to a large degree by enforcing strict null checks.

You can enforce strict null checks in two ways:

- providing the `--strictNullChecks` flag to the TypeScript (tsc) compiler
- setting the `strictNullChecks` property to `true` in the `tsconfig.json` configuration file.

When the flag is false, TypeScript ignores null and undefined values in the code. When it is true, null and undefined have their distinct types. The compiler throws a type error if you try to use them where a concrete value is expected.

How to make object properties immutable in TypeScript? (hint: readonly)

You can mark object properties as immutable by using the `readonly` keyword before the property name. For example:

```
interface Coordinate {  
  readonly x: number;  
  readonly y: number;  
}
```

When you mark a property as `readonly`, it can only be set when you initialize the object. Once the object is created, you cannot change it.

```
let c: Coordinate = { x: 5, y: 15 };  
c.x = 20; // Cannot assign to 'x' because it is a read-only property.(2540)
```

What is a type declaration file?

A typical TypeScript project references other third-party TypeScript libraries such as JQuery to perform routine tasks. Having type information for the library code helps you in coding by providing detailed information about the types, method signatures, etc., and provides IntelliSense.

A type declaration file is a text file ending with a `.d.ts` extension providing a way to declare the existence of some types or values without actually providing implementations for those values. It contains the type declarations but doesn't have any source code. It doesn't produce a `.js` file after compilation.

What are triple-slash directives?

Triple-slash directives are single-line comments that contain a single XML tag. TypeScript uses this XML tag as a compiler directive.

You can only place a triple-slash directive at the top of the containing file. Only single or multi-line comments can come before a triple-slash directive. TypeScript treats them as regular comments if it occurs in the middle of a code block, after a statement.

The primary use of triple-slash directives is to include other files in the compilation process. For example, the following directive instructs the compiler to include a file specified by the path in the containing TypeScript file.

```
/// <reference path="..." />
```

Triple-slash directives also order the output when using `--out` or `--outFile`. The output files are produced to the output file location in the same order as the input files.

Explain the purpose of the ‘in’ operator.

The `in` operator is used to find if a property is in the specified object. It returns `true` if the property belongs to the object. Otherwise, it returns `false`.

```
const car = { make: 'Hyundai', model: 'Elantra', year: 2017 };
console.log('model' in car); // true
console.log('test' in car); // false
```

What are the ‘implements’ clauses in TypeScript?

An `implements` clause is used to check that a class satisfies the contract specified by an interface. If a class implements an interface and doesn't implement that interface, the TypeScript compiler issues an error.

```
interface Runnable {
  run(): void;
}

class Job implements Runnable {
  run() {
    console.log("running the scheduled job!");
  }
}

// Class 'Task' incorrectly implements interface 'Runnable'.
// Property 'run' is missing in type 'Task' but required in type 'Runnable'.(2420)
class Task implements Runnable {
  perform() {
    console.log("pong!");
  }
}
```

A class can implement more than one interface. In this case, the class has to specify all the contracts of those interfaces.

What are string literal types?

In TypeScript, you can refer to specific strings and numbers as types.

```
let foo: "bar" = "bar";
```

```
// OK
foo = "bar";

// Error: Type '"baz"' is not assignable to type '"bar"'.(2322)
foo = "baz";
```

String literal types on their own are not that useful. However, you can combine them into unions. This allows you to specify all the string values that a variable can take, in turn acting like enums. This can be useful for function parameters.

```
function greet(name: string, greeting: "hi" | "hello" | "hola") {
// ...
}

greet("John", "hello");

// Error: Argument of type '"Howdy?"' is not assignable to parameter of type '"hi" | "hello" | "hola"'.(2345)
greet("Mary", "Howdy?");
```

String literal types can help us spell-check the string values.

What are template literal types?

Template literal types are similar to the string literal types. You can combine them with concrete, literal types to produce a new string literal type. Template literal types allow us to use the string literal types as building blocks to create new string literal types.

```
type Point = "GraphPoint";

// type Shape = "Grid GraphPoint"
type Shape = `Grid ${Point}`;
```

Template literal types can also expand into multiple strings via unions. It helps us create the set of every possible string literal that each union member can represent.

```
type Color = "green" | "yellow";
type Quantity = "five" | "six";

// type ItemTwo = "five item" | "six item" | "green item" | "yellow item"
type ItemOne = `${Quantity | Color} item`;
```

Explain the concept of inheritance in TypeScript.

Inheritance allows a class to extend another class and reuse and modify the behavior defined in the other class. The class which inherits another class is called the derived class, and the class getting inherited is called the base class.

In TypeScript, a class can only extend one class. TypeScript uses the keyword 'extends' to specify the relationship between the base class and the derived classes.

```
class Rectangle {
  length: number;
  breadth: number

  constructor(length: number, breadth: number) {
```

```

    this.length = length;
    this.breadth = breadth
  }

  area(): number {
    return this.length * this.breadth;
  }
}

class Square extends Rectangle {
  constructor(side: number) {
    super(side, side);
  }

  volume() {
    return "Square doesn't have a volume!"
  }
}

const sq = new Square(10);

console.log(sq.area()); // 100
console.log(sq.volume()); // "Square doesn't have a volume!"

```

In the above example, because the class Square extends functionality from Rectangle, we can create an instance of square and call both the area() and volume() methods.

What are conditional types? How do you create them?

A conditional type allows you to dynamically select one of two possible types based on a condition. The condition is expressed as a type relationship test.

```
C extends B ? TypeX : TypeY
```

Here, if type C extends B, the value of the above type is TypeX. Otherwise, it is TypeY.

What is the Function type in TypeScript?

Function is a global type in TypeScript. It has properties like bind, call, and apply, along with the other properties present on all function values.

```
function perform(fn: Function) {
  fn(10);
}
```

You can always call a value of the Function type, and it returns a value of 'any' type.

List some of the utility types provided by TypeScript and explain their usage.

TypeScript provides various utility types that make common type transformations easy. These utility types are available globally. Here are some of the essential utility types included in TypeScript.

Utility Type	Description
Partial<Type>	Constructs a type with all properties of Type set to optional.

Required<Type>	Constructs a type consisting of all properties of Type set to required.
Readonly<Type>	Constructs a type with all properties of Type set to readonly.
Record<Keys, Type>	Constructs an object type with property keys are of type Keys, and values are Type.

Node.js Questions

1. What is a first class function in Javascript?

When functions can be treated like any other variable then those functions are first-class functions. There are many other programming languages, for example, scala, Haskell, etc which follow this including JS. Now because of this function can be passed as a param to another function(callback) or a function can return another function(higher-order function). `map()` and `filter()` are higher-order functions that are popularly used.

2. What is Node.js and how it works?

Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome's V8 JavaScript engine. Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient. It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

3. How do you manage packages in your node.js project?

It can be managed by a number of package installers and their configuration file accordingly. Out of them mostly use npm or yarn. Both provide almost all libraries of javascript with extended features of controlling environment-specific configurations. To maintain versions of libs being installed in a project we use `package.json` and `package-lock.json` so that there is no issue in porting that app to a different environment.

4. How is Node.js better than other frameworks most popularly used?

- Node.js provides simplicity in development because of its non-blocking I/O and event-based model results in short response time and concurrent processing, unlike other frameworks where developers have to use thread management.
- It runs on a chrome v8 engine which is written in c++ and is highly performant with constant improvement.
- Also since we will use Javascript in both the frontend and backend the development will be much faster.
- And at last, there are ample libraries so that we don't need to reinvent the wheel.

5. Explain the steps how “Control Flow” controls the functions calls?

- Control the order of execution
- Collect data
- Limit concurrency
- Call the following step in the program.

6. What are some commonly used timing features of Node.js?

- **setTimeout/clearTimeout** – This is used to implement delays in code execution.
- **setInterval/clearInterval** – This is used to run a code block multiple times.

- **setImmediate/clearImmediate** – Any function passed as the setImmediate() argument is a callback that's executed in the next iteration of the event loop.
- **process.nextTick** – Any function passed as the setImmediate() argument is a callback that's executed in the next iteration of the event loop.

7. What are the advantages of using promises instead of callbacks?

The main advantage of using promise is you get an object to decide the action that needs to be taken after the async task completes. This gives more manageable code and avoids callback hell.

8. What is fork in node JS?

A fork in general is used to spawn child processes. In node it is used to create a new instance of v8 engine to run multiple workers to execute the code.

9. Why is Node.js single-threaded?

Node.js was created explicitly as an experiment in async processing. This was to try a new theory of doing async processing on a single thread over the existing thread-based implementation of scaling via different frameworks.

10. How do you create a simple server in Node.js that returns Hello World?

```
var http = require("http");
http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(3000);
```

11. How many types of API functions are there in Node.js?

There are two types of API functions:

- **Asynchronous, non-blocking functions** - mostly I/O operations which can be fork out of the main loop.
- **Synchronous, blocking functions** - mostly operations that influence the process running in the main loop.

12. What is REPL?

PL in Node.js stands for Read, Eval, Print, and Loop, which further means evaluating code on the go.

13. List down the two arguments that async.queue takes as input?

- Task Function
- Concurrency Value

14. What is the purpose of module.exports?

This is used to expose functions of a particular module or file to be used elsewhere in the project. This can be used to encapsulate all similar functions in a file which further improves the project structure.

For example, you have a file for all utils functions with util to get solutions in a different programming language of a problem statement.

```
const getSolutionInJavaScript = async ({
  problem_id
}) => {
  ...
};
const getSolutionInPython = async ({
  problem_id
}) => {
  ...
};
module.exports = { getSolutionInJavaScript, getSolutionInPython }
```

Thus using module.exports we can use these functions in some other file:

```
const { getSolutionInJavaScript, getSolutionInPython } = require("../utils")
```

15. What tools can be used to assure consistent code style?

ESLint can be used with any IDE to ensure a consistent coding style which further helps in maintaining the codebase.

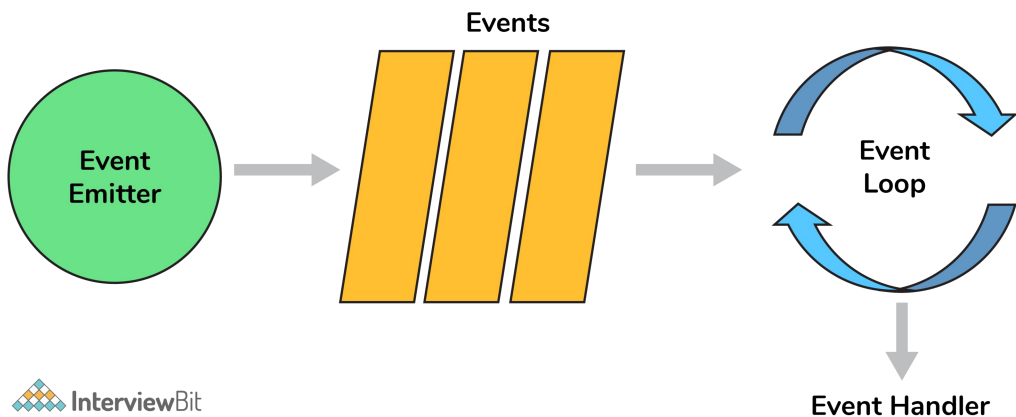
16. What do you understand by callback hell?

```
async_A(function(){
  async_B(function(){
    async_C(function(){
      async_D(function(){
        ....
      });
    });
  });
});
```

For the above example, we are passing callback functions and it makes the code unreadable and not maintainable, thus we should change the async logic to avoid this.

17. What is an event-loop in Node JS?

Whatever that is async is managed by event-loop using a queue and listener. We can get the idea using the following diagram:



Node.js Event Loop

So when an async function needs to be executed (or I/O) the main thread sends it to a different thread allowing v8 to keep executing the main code. Event loop involves different phases with specific tasks such as timers, pending callbacks, idle or prepare, poll, check, close callbacks with different FIFO queues. Also in between iterations it checks for async I/O or timers and shuts down cleanly if there aren't any.

18. If Node.js is single threaded then how does it handle concurrency?

The main loop is single-threaded and all async calls are managed by libuv library.

For example:

```
const crypto = require("crypto");
const start = Date.now();
function logHashTime() {
  crypto.pbkdf2("a", "b", 100000, 512, "sha512", () => {
    console.log("Hash: ", Date.now() - start);
  });
}
logHashTime();
logHashTime();
logHashTime();
logHashTime();
```

This gives the output:

```
Hash: 1213
Hash: 1225
Hash: 1212
Hash: 1222
```

This is because libuv sets up a thread pool to handle such concurrency. How many threads will be there in the thread pool depends upon the number of cores but you can override this.

19. Differentiate between `process.nextTick()` and `setImmediate()`?

Both can be used to switch to an asynchronous mode of operation by listener functions.

`process.nextTick()` sets the callback to execute but `setImmediate` pushes the callback in the queue to be

executed. So the event loop runs in the following manner

timers→pending callbacks→idle,prepare→connections(poll,data,etc)→check→close callbacks

In this process, `nextTick()` method adds the callback function to the start of the next event queue and `setImmediate()` method to place the function in the check phase of the next event queue.

20. How does Node.js overcome the problem of blocking of I/O operations?

Since the node has an event loop that can be used to handle all the I/O operations in an asynchronous manner without blocking the main function.

So for example, if some network call needs to happen it will be scheduled in the event loop instead of the main thread(single thread). And if there are multiple such I/O calls each one will be queued accordingly to be executed separately(other than the main thread).

Thus even though we have single-threaded JS, I/O ops are handled in a nonblocking way.

21. How can we use async await in node.js?

Here is an example of using async-await pattern:

```
// this code is to retry with exponential backoff
function wait (timeout) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve()
    }, timeout);
  });
}

async function requestWithRetry (url) {
  const MAX_RETRIES = 10;
  for (let i = 0; i <= MAX_RETRIES; i++) {
    try {
      return await request(url);
    } catch (err) {
      const timeout = Math.pow(2, i);
      console.log('Waiting', timeout, 'ms');
      await wait(timeout);
      console.log('Retrying', err.message, i);
    }
  }
}
```

22. What is node.js streams?

Streams are instances of `EventEmitter` which can be used to work with streaming data in Node.js. They can be used for handling and manipulating streaming large files(videos, mp3, etc) over the network. They use buffers as their temporary storage.

There are mainly four types of the stream:

- **Writable:** streams to which data can be written (for example, `fs.createWriteStream()`).

- **Readable:** streams from which data can be read (for example, `fs.createReadStream()`).
- **Duplex:** streams that are both Readable and Writable (for example, `net.Socket`).
- **Transform:** Duplex streams that can modify or transform the data as it is written and read (for example, `zlib.createDeflate()`).

23. What are node.js buffers?

In general, buffers is a temporary memory that is mainly used by stream to hold on to some data until consumed. Buffers are introduced with additional use cases than JavaScript's `Unit8Array` and are mainly used to represent a fixed-length sequence of bytes. This also supports legacy encodings like ASCII, utf-8, etc. It is a fixed(non-resizable) allocated memory outside the v8.

24. What is middleware?

Middleware comes in between your request and business logic. It is mainly used to capture logs and enable rate limit, routing, authentication, basically whatever that is not a part of business logic. There are third-party middleware also such as body-parser and you can write your own middleware for a specific use case.

25. Explain what a Reactor Pattern in Node.js?

Reactor pattern again a pattern for nonblocking I/O operations. But in general, this is used in any event-driven architecture.

There are two components in this: 1. Reactor 2. Handler.

Reactor: Its job is to dispatch the I/O event to appropriate handlers

Handler: Its job is to actually work on those events

26. Why should you separate Express app and server?

The server is responsible for initializing the routes, middleware, and other application logic whereas the app has all the business logic which will be served by the routes initiated by the server. This ensures that the business logic is encapsulated and decoupled from the application logic which makes the project more readable and maintainable.

27. For Node.js, why Google uses V8 engine?

Well, are there any other options available? Yes, of course, we have Spidermonkey from Firefox, Chakra from Edge but Google's v8 is the most evolved(since it's open-source so there's a huge community helping in developing features and fixing bugs) and fastest(since it's written in c++) we got till now as a JavaScript and WebAssembly engine. And it is portable to almost every machine known.

28. Describe the exit codes of Node.js?

Exit codes give us an idea of how a process got terminated/the reason behind termination.

A few of them are:

- Uncaught fatal exception - (code - 1) - There has been an exception that is not handled
- Unused - (code - 2) - This is reserved by bash
- Fatal Error - (code - 5) - There has been an error in V8 with stderr output of the description
- Internal Exception handler Run-time failure - (code - 7) - There has been an exception when bootstrapping function was called
- Internal JavaScript Evaluation Failure - (code - 4) - There has been an exception when the bootstrapping process failed to return function value when evaluated.

29. Explain the concept of stub in Node.js?

Stubs are used in writing tests which are an important part of development. It replaces the whole function which is getting tested.

This helps in scenarios where we need to test:

- External calls which make tests slow and difficult to write (e.g HTTP calls/ DB calls)
- Triggering different outcomes for a piece of code (e.g. what happens if an error is thrown/ if it passes)

For example, this is the function:

```
const request = require('request');
const getPhotosByAlbumId = (id) => {
const requestUrl = `https://jsonplaceholder.typicode.com/albums/${id}/photos?_limit=3`;
return new Promise((resolve, reject) => {
  request.get(requestUrl, (err, res, body) => {
    if (err) {
      return reject(err);
    }
    resolve(JSON.parse(body));
  });
});
};
module.exports = getPhotosByAlbumId;
```

To test this function this is the stub

```
const expect = require('chai').expect;
const request = require('request');
const sinon = require('sinon');
const getPhotosByAlbumId = require('./index');
describe('with Stub: getPhotosByAlbumId', () => {
  before(() => {
    sinon.stub(request, 'get')
      .yields(null, null, JSON.stringify([
        {
          "albumId": 1,
          "id": 1,
          "title": "A real photo 1",
          "url": "https://via.placeholder.com/600/92c952",
          "thumbnailUrl": "https://via.placeholder.com/150/92c952"
        },
        {
          "albumId": 1,
```

```

        "id": 2,
        "title": "A real photo 2",
        "url": "https://via.placeholder.com/600/771796",
        "thumbnailUrl": "https://via.placeholder.com/150/771796"
      },
      {
        "albumId": 1,
        "id": 3,
        "title": "A real photo 3",
        "url": "https://via.placeholder.com/600/24f355",
        "thumbnailUrl": "https://via.placeholder.com/150/24f355"
      }
    ]
  });
});
after(() => {
  request.get.restore();
});
it('should getPhotosByAlbumId', (done) => {
  getPhotosByAlbumId(1).then((photos) => {
    expect(photos.length).toEqual(3);
    photos.forEach(photo => {
      expect(photo).toHaveProperty('id');
      expect(photo).toHaveProperty('title');
      expect(photo).toHaveProperty('url');
    });
    done();
  });
});
});
});

```

30. What is an Event Emitter in Node.js?

EventEmitter is a Node.js class that includes all the objects that are basically capable of emitting events. This can be done by attaching named events that are emitted by the object using an eventEmitter.on() function. Thus whenever this object throws an even the attached functions are invoked synchronously.

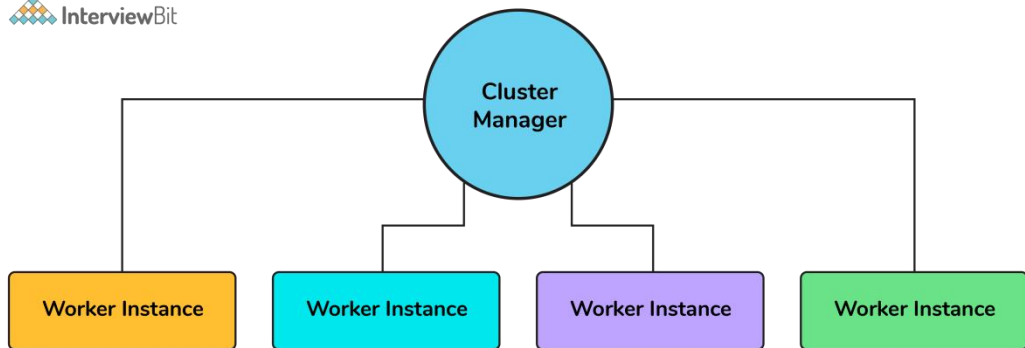
```

const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();
myEmitter.on('event', () => {
  console.log('an event occurred!');
});
myEmitter.emit('event');

```

31. Enhancing Node.js performance through clustering.

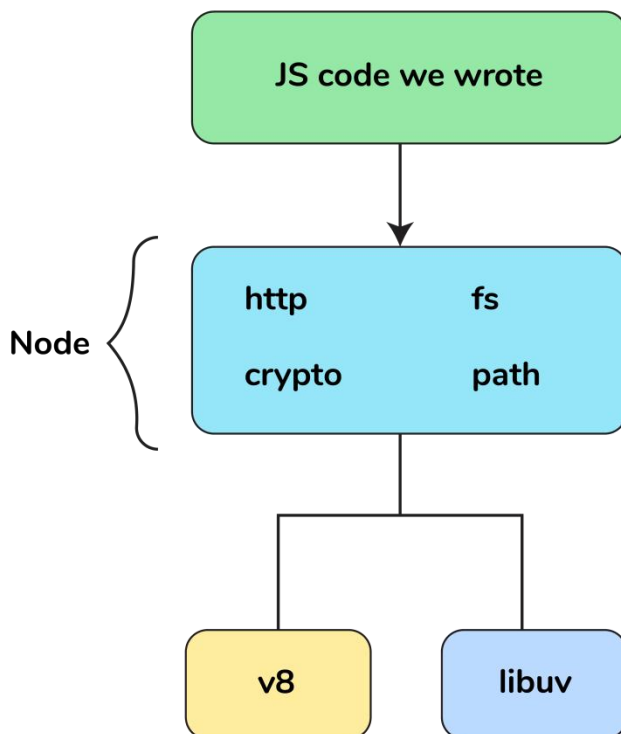
Node.js applications run on a single processor, which means that by default they don't take advantage of a multiple-core system. Cluster mode is used to start up multiple node.js processes thereby having multiple instances of the event loop. When we start using cluster in a nodejs app behind the scene multiple node.js processes are created but there is also a parent process called the **cluster manager** which is responsible for monitoring the health of the individual instances of our application.



Clustering in Node.js

32. What is a thread pool and which library handles it in Node.js

The Thread pool is handled by the libuv library. libuv is a multi-platform C library that provides support for asynchronous I/O-based operations such as file systems, networking, and concurrency.



Thread Pool

33. What is WASI and why is it being introduced?

Web assembly provides an implementation of WebAssembly System Interface specification through WASI API in node.js implemented using WASI class. The introduction of WASI was done by keeping in mind its possible to use the underlying operating system via a collection of POSIX-like functions thus further enabling the application to use resources more efficiently and features that require system-level access.

34. How are worker threads different from clusters?

Cluster:

- There is one process on each CPU with an IPC to communicate.
- In case we want to have multiple servers accepting HTTP requests via a single port, clusters can be helpful.
- The processes are spawned in each CPU thus will have separate memory and node instance which further will lead to memory issues.

Worker threads:

- There is only one process in total with multiple threads.
- Each thread has one Node instance (one event loop, one JS engine) with most of the APIs accessible.
- Shares memory with other threads (e.g. SharedArrayBuffer)
- This can be used for CPU-intensive tasks like processing data or accessing the file system since NodeJS is single-threaded, synchronous tasks can be made more efficient leveraging the worker's threads.

35. How to measure the duration of async operations?

Performance API provides us with tools to figure out the necessary performance metrics. A simple example would be using `async_hooks` and `perf_hooks`

```
'use strict';
const async_hooks = require('async_hooks');
const {
  performance,
  PerformanceObserver
} = require('perf_hooks');
const set = new Set();
const hook = async_hooks.createHook({
  init(id, type) {
    if (type === 'Timeout') {
      performance.mark(`Timeout-${id}-Init`);
      set.add(id);
    }
  },
  destroy(id) {
    if (set.has(id)) {
      set.delete(id);
      performance.mark(`Timeout-${id}-Destroy`);
      performance.measure(`Timeout-${id}`,
        `Timeout-${id}-Init`,
```

```

        `Timeout-${id}-Destroy`);
    }
  }
});
hook.enable();
const obs = new PerformanceObserver((list, observer) => {
  console.log(list.getEntries()[0]);
  performance.clearMarks();
  observer.disconnect();
});
obs.observe({ entryTypes: ['measure'], buffered: true });
setTimeout(() => {}, 1000);

```

This would give us the exact time it took to execute the callback.

36. How to measure the performance of async operations?

Performance API provides us with tools to figure out the necessary performance metrics.

A simple example would be:

```

const { PerformanceObserver, performance } = require('perf_hooks');
const obs = new PerformanceObserver((items) => {
  console.log(items.getEntries()[0].duration);
  performance.clearMarks();
});
obs.observe({ entryTypes: ['measure'] });
performance.measure('Start to Now');
performance.mark('A');
doSomeLongRunningProcess(() => {
  performance.measure('A to Now', 'A');
  performance.mark('B');
  performance.measure('A to B', 'A', 'B');
});

```

<https://www.geeksforgeeks.org/node-js-examples/>

Laravel & PHP Questions

Laravel 10 new features

Domain Driven Design Aggregates in Laravel

Behavioral job interview questions about time management

Behavioral job interview questions about time management

Ruby on Rails Questions

encapsulation

polymorphic -> multiple names

example of polymorphism

authentication in ruby on rails

structure how did you authentication

what are the migrations?

which of the migrations are up and down?

there are five migrations which are up. -> schema

different associations in rails. -> belongs_to, has_one, has_many

has belongs vs has many

when would you use like has many and has belongs

what are the callbacks in ruby on rails?

what;s the swequence like?

If you have after save after create and after image, so what would be executed telling me the sequence?

before_validation

after_validation

what about commit after_commit in Ruby on Rails?

after_destroy on ruby on rails?

delete wont allow yo to raise backs right?

what are the closures in ruby?

lambda vs block in ruby on rails?

different access in ruby?

public protected vs predicted private in ruby?

initialize method, is it public private or connect?

what are the collections in ruby?

what can you do to prevent security attacks CSRF attacks in ruby?

how is it used? how remakes it?

What are all featrure having used in metaprogramming in ruby on rails?

metaphor programming funtions in metaprogramming? define_method in ruby on rails

arr.flatten method in ruby.

```
arr=[1,2,3,[3,[4,1,2]],6,6]
```

```
def recursive_function(array, result = [])
```

```
  array.each do |element|
```

```
    if elelment.calass == Array
```

```
      recursive_flatten(element, results)
```

```
    else
```

```
      results << element
```

```
    end
```

```
  end
```

```
  results
```

```
end
```

```
print recursive_function(arr)
```

validate brackets in ruby?

```
def check_parences(str)
  stacks = []
  symbols = {'{' => '}', '[' => ']', '(' => ')'}
  str.each_char do |c|
    stack << c if symbols.key?(c)
    return false if symbols.key(c) && symbols.key(c) != stack.pop
  end
  stack.empty?
end

puts check_parences("")
```

background job processing

join table-> left join, right join, inner join

teacher table

id name age address

1 Erik 23 LA

2 Nick 30 LB

3 David 32 LC

student table

id name age teacher_id

1 Vlad 10 1

2 Frank 11 1

3 Nichol 12 2

```
select t.name as teacher_name, s.name as student_name from teacher t as join student as s
on t.id = s.teacher_id
```

Behavioral Questions

Behavioral job interview questions about time management

Software teams can follow these nine ways of fixing bugs in production:

1. Establish a standardized process.
2. Make plans to quickly fix defects.
3. Practice time management.
4. Implement benchmarks.
5. Prioritize test code.
6. Perform chaos engineering.
7. Move fast and break things.
8. Adopt a mission-critical mentality.
9. Mature the product.

What To Do When There's a Bug in Production

Step One: Remain Calm

Because we are the ones who are testing the product and signing off on the release, it's easy to panic when a bug is found in Production. We ask ourselves "How could this have happened?" and we can be tempted to thrash around looking for answers. But this is not productive. Our top priority should be to make sure that the bug is fixed, and if we don't stay calm, we may not investigate the issue properly or test the fix properly.

Step Two: Reproduce the Issue

If the issue came from a customer, or from another person in your company, the first thing to do is to see if you can reproduce the issue. It's possible that the issue is simply user error or a configuration problem. But don't jump to those conclusions too quickly! Make sure to follow any steps described by the user as carefully as you can, and wherever possible, make sure you are using the same software and hardware as the user: for example, use the same type of mobile device and the same build; or the same type of operating system and the same browser. If you are unable to reproduce the issue, ask the user for more information and keep trying. See [How to Reproduce A Bug](#) for more tips.

Step Three: Gather More Information

Now that you have reproduced the issue, gather more information about it. Is the issue happening in your test environment as well? Is the issue present in the previous build? If so, can you find a build where the issue is not present? Does it only happen with one specific operating system or browser? Are there certain configuration settings that need to be in place to see the issue? The more information you can gather, the faster your developer will be able to fix the problem.

Step Four: Understand the Root Cause

At this point, your developer will be working to figure out what is causing the bug. When he or she figures it out, make sure they tell you what the problem was, and make sure that you understand it. This will help you figure out how to test the fix, and also to determine what regression tests should be done.

Step Five: Decide When to Fix the Issue

When there's a bug in Production, you will want to fix it immediately, but that is not always the best course of action. I'm sure you've encountered situations where fixing a bug created new ones. You will want to take some time to test any areas that might have been impacted by the fix.

When trying to decide when to fix a bug, think about these two things:

- how many users are affected by the issue?
- how severe is the issue?

You may have an issue where less than one percent of your users are affected. But if the bug is so severe that the users can't use the application, you may want to fix the bug right away.

Or, you may have an issue that affects all of your users, but the issue is so minor that their experience won't be impacted. For example, a misaligned button might not look nice, but it's not stopping your users from using the application. In this case, you might want to wait until your next scheduled release to fix the issue.

Step Six: Test the Fix

When you test the bug fix, don't check it just once. Be sure to check the fix on all supported browsers and devices. Then run regression tests in any areas affected by the code change. If you followed Step Four, you'll know which areas to test. Finally, do a quick smoke test to make sure no important functionality is broken.

Step Seven: Analyze What Went Wrong

It's tempting to breathe a big sigh of relief and then move on to other things when a Production bug is fixed. But it's very important to take the time to figure out exactly how the bug got into Production in the first place. This is not about finger-pointing and blame; everybody makes mistakes, whether they are developers, testers, product owners, managers, or release engineers. This is about finding out what happened so that you can make changes to avoid the problem next time.

Perhaps your developers made a code change and forgot to tell you about it, so it wasn't tested. Perhaps you tested a feature, but forgot to test it in all browsers, and one browser had an issue. Maybe your product owner forgot to tell you about an important use case, so you left it out of your test plan.

Whatever the issue, be sure to communicate it clearly to your team. Don't be afraid to take responsibility for whatever your part was in the issue. See my post on [Extreme Ownership](#) for more details.

Step Eight: Brainstorm Ways to Prevent Similar Issues in the Future

Now that you know what went wrong, how can you prevent it from happening again? Discuss the issue with your team and see if you can come up with some strategies.

You may need to change a process: for example, having your product owner sign off on any new features in order to make sure that nothing is missing. Or you could make sure that your developers let you know about any code refactoring so you can run a regression test, even if they are sure they haven't changed anything.

You may need to change your strategy: you could have two testers on your team look at each feature so it's less likely that something will be overlooked. Or you could create test plans which automatically include a step to test in every browser.

You may need to change both your process and your strategy! Whatever the situation, you can now view the bug that was found in Production as a blessing, because it has resulted in you being a wiser tester, and your team being stronger.

Question #1 - How do you accomplish tasks when under a tight deadline? Give me an example.

Sample Answer:

Situation: Well, typically, I try to never commit to a deadline I don't think I can make. But sometimes, unexpected things happen and you're forced to think on your feet. For example, at my last job, my coworker had to take some time off work because of an emergency, and his project was left without a manager.

Task: My supervisor then instructed me to take over his project and complete what work was left. Suddenly, I had a new project on my hands, and I wasn't really sure how to handle it, as the deadline was in 1 week.

Action: First, I requested a reduction on my own daily sales goals - which I was granted. This way, I could pay more attention to the project, and only a few hours per day to my original tasks. Once I had a consistent schedule and hours set for each of my tasks, it was mostly easy from there.

Results: Thanks to my teammates and my good time management skills, I managed to finish up 2 days early before the deadline. And once my coworker came back to work, I was able to review the whole thing with him before submitting it. For what it's worth, he was thoroughly impressed. And a few months later, I even got promoted based on my performance.

Question #2 - Describe a long-term project you managed. How did you make sure everything was running smoothly?

Sample Answer:

Situation: When I was at Company X, I was managing the web development team in charge of setting up a new website for one of our biggest clients at the time. With most projects, we had a process set up and we would get most sites done in up to 2 months. This project, however, was a bit different, as the website was supposed to be more detailed, with a lot of unique pages. So, we had to be a lot more careful with our time-management.

Task: We had a strict deadline of 15 weeks, and I had to make sure that we used up our time as efficiently as possible.

Action: Before getting to actual work, I decided that we should plan everything out by the week. After some research and consulting with our team of developers, we decided to split the workload between different stages. We would devote around 1 week to the discovery phase, 5 weeks to design, 3 weeks to initial development and the rest to any modifications and updates.

Results: In the end, we actually finished the website with all the promised functionalities in just under 3 months. The client was very satisfied with the result and eventually ended up recommending partners to our firm.

Question #3 - Sometimes, it's almost impossible to get everything done on your to-do list. What do you do when your list of responsibilities becomes overwhelming?

Situation: As a senior at University X, there were times when I just couldn't physically get everything done on time. For example, towards the end of my final semester, I was the Student Council President and I was also writing my University thesis. I had to submit my thesis the next day, and I was also working with my fellow student council members to organize the end-of-the-year ceremony for the University.

Task: If I had tried to multitask both, I would just have done a poor job. Now, for me, the University thesis was clearly higher up in my list of priorities. After all, this was what my studies were building up to for so long. But I couldn't just abandon my council members either. With 24 hours until my thesis deadline, I had to think fast.

Action: I decided that the best approach was to send all of my notes and outlines for the event to the Student Council VP, who was also a close friend of mine. Luckily, he understood my situation and took over my event-management responsibilities. Then, I had just enough time to edit and finalize my paper.

Results: Thanks to the VP, I was able to fix and finalize my Thesis. And fortunately, the event went without a hitch too. In the end, I learned a valuable lesson on time-management, and the importance of having the right team around you who you can rely on.

Question #4 - Tell me about a time you set a personal goal for yourself. How did you ensure you would meet your objectives and what steps did you take?

Situation: I think the most recent, and important, personal goal that comes to mind is that I managed to teach myself web development from scratch. You see, I wasn't very satisfied as a sales rep at Company X. My coworkers were nice, and the pay was decent too, but I just didn't see myself growing there.

Task: So, I decided that I wanted a career change in a field I've always been interested in - web development. Now, because I was working full-time, I had to be very efficient with my time-management skills.

Action: I did some research, and all that was left to do was just follow my routine and stay committed. I set up a personal calendar and made sure to study HTML, CSS, and JavaScript for at least 2 hours every day. I gathered a list of beginner-friendly books to start with, and once I was done with those, I paid for some advanced online courses to improve my React and Vue.js skills. When I felt comfortable, I started working on some personal projects for my portfolio and did some freelance work part-time while I was still working at Company X.

Results: In the end, I'm glad I stuck to my plan and continued with my set curriculum. If I did not have my calendar planned out with specific objectives, I surely would have been overwhelmed. Sure, at times, it felt like I was basically working 2 jobs and that a lot of the material wasn't making sense. But I just kept moving forward, and then, I got my first real break as a junior web dev at Company Y.

Question #5 - Can you describe an instance where your supervisor or manager just gave you too much work with not enough time? What did you do?

Situation: I had a pretty rocky start with my manager at Agency X, as we had different expectations for my workload. Normally, I don't have a problem with a fast-paced working environment, and I tend to thrive when I'm thinking on my feet. But at the agency, I had just finished onboarding, and I was already bombarded with tasks and weekly reports. For the most part, I was managing to get everything done on time, but I realized the quality would suffer if my list of tasks kept getting longer.

Task: So, I had to take up my work schedule issue with my manager and let him know about my concern. I decided that being direct, and also respectful was the best approach, and booked the meeting.

Action: During the meeting, I remained calm, and just went straight to the point. I explained how I liked my work, but the heavy workload was really impacting the quality of the work.

Results: Luckily, he was understanding. I was the first in-house designer they'd hired, and they weren't 100% sure what was a lot of work, and what wasn't. We ended up working together to better define my responsibilities. From then on, I was, for the most part, only getting the workload I could handle without diminishing the quality of my work.

Question #6 - How do you deal with unexpected changes to deadlines?

Interviewers use this question to assess how you react to time-sensitive projects and how well you work under pressure. In your response, highlight your ability to react quickly without sacrificing quality, and emphasize strategies that have helped you adapt to changes in your previous work experience.

Example: *"Last April, my manager assigned me to increase a software's efficiency by 3% in 30 days. However, the deadline changed a few days later, and I only had two weeks to complete the task. While I was a little nervous about my ability to complete the task in time, I asked fellow team members for their advice and learned a new strategy that allowed me to increase software efficiency by 4% and exceed my goal in two weeks."*

Behavioral job interview questions about communication skills

Question #1 - How do you handle a disagreement with your colleagues? Give me an example of when you successfully persuaded someone to see things your way at work.

Situation: When I was working as a recruiter at Company X, I noticed that one of the candidates who had sent in their application was perfect for the role. Though he didn't have a university diploma and his resume wasn't too polished, reading his cover letter, it was obvious he knew the industry and had delivered clear results.

Task: I thought it was worth giving him a shot, but my supervisor didn't see it that way. She skimmed through the resume and told me not to waste time, and just discard the candidate.

Action: I was, however, still pretty confident in the candidate, so I talked to the supervisor over lunch. I took a bit of an indirect approach, though. Instead of trying to directly pitch the candidate, I asked her to clarify the job description a bit more. We went a bit in-depth on what, exactly, we were looking for in the candidate, and once we were done discussing it, I told her that we happened to have a candidate that possessed all the relevant experience, but his resume was a bit weak.

Results: Convinced, the supervisor decided to give the candidate's application a more in-depth look and realized that they were, in fact, very qualified. She thanked me for bringing it up and agreed with me that the candidate was worth calling in for an interview.

Question #2 - What would you do if you misunderstood an important task on the job? Give me an example.

Situation: At my previous internship at Company X, I underestimated the amount of time it would take me to finish a presentation for a team meeting. The deadline my boss gave me was around a week, which was completely fair and I didn't think it would be a problem. However, apparently, we had some miscommunication with what he'd meant with the deadline. I thought it was the date where we would go through the presentation, edit it together, and submit it like that. Apparently what he'd meant, though, was to have the presentation 100% ready on that date.

Task: So, I had to submit a draft presentation first, edit it based on my manager's comments, and then present the report, all within 2 days.

Action: I booked a meeting with the manager for the following day, and spent 4 extra hours at the office to make sure that the first draft of the presentation was spotless. We held the meeting the next day, and went through the presentation together to make sure it's spotless.

Results: The manager loved the work, and it only took us around 30 minutes to finalize the whole thing.

Question #3 - Have you ever had to work under someone who wasn't very good at communicating? What happened?

Situation: Yes, at my last job as a tech recruiter the hiring manager I was working directly with was somewhat more difficult to communicate with. He had very strict and precise requirements on the type of candidates he wanted to invite for interviews. He wasn't open to much communication on the matter or trying new things even when the company desperately needed new hires. This one time, I got a candidate that was a pretty good fit for the job, but was lacking in some aspects.

Task: I wanted to make sure that we got the person in for an interview, but I was 100% sure that my hiring manager would shut me down.

Action: So, before running the candidate through him, I called them and collected his biggest strengths to present to the hiring manager.

Results: The hiring manager did, indeed, end up liking the candidate and calling them in for an interview.

Question #4 - Tell me about a time when you successfully explained a technical problem to a colleague or a customer who didn't have a tech background?

Situation: I've worked as a tech support specialist before, so I really excel at this. I've had to explain complex concepts to customers on a regular basis, but to give you one single example, I've had to explain to clients with next to no understanding of computers how to delete a virus on their computer.

Task: After trying to give basic instructions to the client, they still didn't really understand much, so I had to come up with a smarter solution.

Action: So what I did was, I walked them through the entire thing step by step while explaining it simply but in no condescending terms. Instead of making them do most of the work, I walked them through the process of getting me to connect with their computer, and then I explained to them what, exactly, I did.

Results: The customer was very happy with my work, and we managed to fix the issue with their computer.

Question #5 - Can you tell me about a time you gave a presentation that was particularly successful? Why do you think it went well?

Situation: Sure thing. As the business development manager at Firm X, there were quite a few opportunities when I had to speak in front of a crowd. The most recent, and successful, one was for the new project we were launching.

Task: I was called on to speak for a 2 department-wide meeting, of up to 50 people. Now, I had never delivered a presentation to this many people, but luckily, I knew most of them quite well after years of working with them.

Action: Working with 2 other members of my team, I decided to take a more creative approach, and create a short video (a skit) to hook the audience. That was the intro, and then we used PowerPoint and hands-on examples to show what to expect from the new project launch. And finally, we dedicated the last 5 minutes to a Q&A session.

Results: It felt longer, but the whole speech took about 15 minutes in total. We got great feedback from the audience, and I was later asked to present at the all-hands meeting the next month. I knew my colleagues well enough and I tried to make the speech as if I was having a one-on-one conversation with a friend - with a few jokes in-between.

Behavioral job interview questions about teamwork

Question #1 - Tell me about a time when you had to work with someone completely different from you. How did you adapt to collaborate better?

Situation: Sure, I always enjoy working with new and different people. Usually, because they bring something new to the table. At Company X, there was a particularly young developer who was assigned to work with me on a new software development project, and I was to run him through what our typical coding process was like.

Task: It was also my job to get to know him, and find common ground so that we could effectively work together. The fact that he was younger wasn't an issue for me, but because he was completely self-taught, he didn't know a lot about the industry methodologies we used.

Action: Teaching him everything from scratch would take too much time. So, instead, I briefly explained the development process (waterfall model) we were using for that specific project, and taught him how to write tests for our code-base. Writing tests is the number 1 way to learn what code does. After all, that's how I got started with development.

Results: I also sat down and helped him go through the material at times, but in the end, he surprised me by how much of a fast-learner he was. He just needed a bit of encouragement and guidance. Through this approach, he learned our whole routine in less than a week, while most of our new hires needed at least up to 2 weeks. In return, I learned a lot about multitasking and time-management from him. The whole thing was a win-win situation, and it was all smooth sailing the next time we worked together (which was quite often).

Question #2 - What do you do when your team member refuses to, or just can't complete their part of the work? Give me an example.

Situation: There was one co-worker at Company X who was notorious for being bad at deadlines. But she would always end up delivering exceptional work, just a few hours (or worse - days) late. For some reason, the company was ok with this as her work was just too good. So, this one time, the management put us together to work on a time-sensitive project.

Task: Our task was to turn in a sales presentation together and have our manager go over it before sending the client the final version. Because of how important the project was, I didn't want to risk going over the deadline - as this would also directly impact other people. Either way, for everyone's sake, I had to somehow get her to hurry up with the project. So, I decided to try and push her a little and see what would happen.

Action: I started regularly checking in on her to see where she was with work. I would bring it up at times over lunch, send a quick Slack message, and so on. She wasn't taking this quite well, but it DID get her to work faster and more efficiently.

Results: At the end, the constant check-ins and pushing did have a positive effect, even though the co-worker didn't particularly like me too much once we were finished with the work. We even managed to submit the final version of the presentation 2 days before the deadline.

Behavioral job interview questions about working with clients

Question #1 - Clients can be difficult to work with sometimes. Can you describe a situation when a client was wrong and you had to correct them?

Situation: Absolutely. One of our past clients at Agency X came to us because his Facebook advertising strategy wasn't working. He was driving traffic but wasn't getting any conversions, so they thought that it was because they weren't reaching the right audience. We realized, though, that it was actually because their product homepage wasn't really that convincing. The client, however, was adamant about "not fixing what wasn't broken."

Task: I had to somehow communicate with the client that the service he wanted wasn't what he wanted - there was no way for us to fix his Facebook ads if his homepage wasn't selling the product.

Action: We had to give the client an ultimatum - they either go with our approach, or we wouldn't be able to get the results (and hence, work with them).

Results: After some back and forth, the client grudgingly agreed to do an A/B test between the existing landing page, and one that we'd propose. So, we tested the two landing pages with the same ads he'd been running, and ended up getting 2.5x better results. From then on, the client was a lot more willing to allow us to experiment with whatever we proposed.

Question #2 - How do you handle irate customers? Give me an example.

Situation: Working in customer support, you really get to talk with many different kinds of people. I remember I had one angry customer that called the helpdesk once to complain. He kept repeating the product he bought was faulty and demanded me to resolve the situation then and there.

Task: Customers calling for refunds happen all the time, but this one was different as he just kept shouting over the phone the whole time. I had to get him to calm down if I wanted the call to go anywhere.

Action: Fortunately, I had experience dealing with loud customers, and knew the first thing I had to do was listen to his story. Halfway through telling his story, he calmed down once he realized I was trying to help. He explained that the product was supposed to be a gift, and that's why he was so frustrated. Then, I offered 2 solutions: a refund or a replacement for his product with express delivery.

Results: The customer opted for the replacement option. I called him back once they received the order just to check-in if he was happy with the product. He turned out to be happy both with the product and our service, and thanked me for the help.

Question #3 - We all make mistakes sometimes we wish we could take back. Is there a time that comes to mind where you wish you had handled a situation with a client or colleague differently?

Situation: This one client we worked with was particularly difficult. They were extremely unpleasant to work with and treated our staff pretty badly. The management, however, insisted on sticking with them, since they made up for a good chunk of our income. At one point, though, the client just barged into our office and started yelling at their account manager for a small mistake on their end.

Task: At this point, I realized that working with the client was really affecting our staff negatively, and we'd be losing some good employees if we kept working with them.

Action: So, I set up a meeting with the management team, and gave them concrete facts and figures about the client. Sure, they were paying us good money, but they were really hurting the workplace morale.

Results: After hearing me out, the management agreed and fired the client. They decided that overall, the impact such clients had on the company wasn't worth it, and started doing stricter vetting during discovery calls.

Behavioral job interview questions about adaptability

Question #1 - Tell me about your first job in the industry. What did you do to learn the ropes?

Situation: Well, my first job in the field was as a junior dev ops engineer. While I did have extensive knowledge of the field, I didn't have too much experience doing it.

Task: This made it very hard for me to get started with the job. While I was working almost all the time, I wasn't getting too much done.

Action: So, what I did was, taking a lot of my personal time to really work and learn the ins and outs of dev ops. I also made sure to talk to my team members and get their input on daily tasks.

Results: A few months into the job, I managed to learn the ropes and ended up being a lot more productive.

Question #2 - Can you give me an example of when you had to adapt to a new and sudden change in the workplace? What happened?

Situation: Sure thing. In my previous position as an account manager at Company X, we had to suddenly change all of our CRM software and move all the data to a new tool. The CRM tool we'd been using till now wasn't fit for a growing team, and on top of that, they were upping their pricing, so it wasn't really worthwhile for us.

Task: I was put in charge of finding the replacement CRM, as I was the one who knew the previous one inside-out. And this was also an opportunity for me to clean up our outdated info and start fresh. All the while, I still had to handle my daily responsibilities and as usual.

Action: So, the first thing I did was ask our sales associates and lead generation teams what they thought of the old CRM, and if there were any new features they were lacking. After doing a bit of research and asking around, I found the perfect tool that had it all - sales analytics, email integration, and more. And because I typically have no problem with learning new tools, I stayed in one evening, transferred our data to the new tool, and wiped the old account. Finally, I sent a new announcement to the entire team about the new software, as well as a video on how to use it.

Results: We completed the transfer with 4 days to spare, the team was satisfied with the new CRM, and my daily responsibilities as an account manager didn't suffer.

Question #3 - Give me an example of when you had to suddenly perform under pressure. What happened and how did you handle it?

Situation: As a seasonal worker, there have been a lot of times where I had to juggle extra responsibilities. My last position as a line cook at Restaurant X comes to mind. During summer, we were pretty much always full, and sometimes, even understaffed to handle all the customers. To make things worse, we didn't have the best shift system at the time either. So, if someone were to unexpectedly not show up for their shift, we'd have to put out the fires as they came up.

Task: Which is exactly what happened when one of our waitresses had to cancel her shift due to an emergency.

Action: So, I stepped up and took her shift as soon as I had clocked out of mine as one of the line cooks. Luckily, I had previous experience working as a waiter.

Results: I was tired and a bit uncoordinated at the beginning, but at the end of the day, everything worked out just fine.

Question #4 - Tell me about a time you were new to a situation or environment. How did you adapt?

When I started my last job, I had never worked as a full-time software engineer and knew that I had a lot to learn. However, I made sure to ask many questions and take notes about what I learned, reviewing them at the end of each workday. Eventually, I became familiar with the systems and protocol and exceeded my goals within the first six months of employment.

Question #5 - As a software engineer, you must be both predictable and innovative. How can these traits coexist in your work?

By asking this question, the interviewer is likely trying to assess your ability to balance consistency with creativity and gain an understanding of your philosophy toward your work. Try to provide an answer that's true to your philosophy but also reflects the values of the company with which you're interviewing.

Example: *"I believe that the balance of predictability and innovation is the foundation of my work as a software engineer. It's important that my team delivers high-quality software within a predictable time frame, but our everyday work requires us to be innovative and develop alternative systems and processes. From my experience, it is an engineer's ability to balance predictability and innovation that leads to their success and potential to be an effective team member."*

Behavioral job interview questions about leadership

Question #1 - Tell me about a time when you successfully delegated tasks to your team.

Situation: Well, at my first job as a team lead, I had to really get to know most of my team in order to delegate tasks appropriately.

Task: Most team members were new to the company, so I didn't have much to go with.

Action: So, I sat down with each team member individually, and really got to know them and their strengths and weaknesses, and distributed tasks based on their personality.

Results: Team members were pretty happy with the tasks they got, and started off their relationship with our company on a positive note.

Question #2 - Can you tell me about a time when you had to perform a task or work on a project you had no previous experience before? How did you approach this situation and what did you learn?

Situation: In my previous position at Company X, my manager had to leave unexpectedly for about a month due to a medical condition. Fortunately, she was able to give us a week's notice.

Task: Because of that, our director asked me to fill in as the interim manager. I was familiar with the basics of management on a theoretical level, and I had worked with my manager closely before, but I certainly wasn't trained to be a manager yet. Though, I wasn't going to say no, and I, more or less, felt confident about my ability to take on the new challenge.

Action: So, I accepted the position. The first thing I did was gather the team and let them know about the situation. I was very open about my lack of experience, and asked them to be open about giving feedback when possible. I also asked a manager for an hour of their time to pick their brain and make sure I'm doing everything right.

Results: In the end, we managed to get through the month without any problems, and delivered all the projects on time. When my manager returned, she was very pleased with the work, and I even got compliments from our director. Because of my success with the role, I was then promoted to team manager at the end of that year.

Question #3 - What are your organizational strategies?

When answering this question, be honest and reflect on what strategies have worked for you in the past. Discuss which methods were the most successful and why, and show that you understand your own work style and best practices. You can explain your time management strategies and give past examples.

Example: *"Over the years, I've found that I perform the best at work when I'm the most organized. For this reason, I have developed a variety of strategies to make sure that I'm managing my time and projects accordingly. For example, I use time management applications on my phone that help me stay focused, and I limit multitasking when possible so that I remain concentrated on one task until I fully execute it."*

Blockchain Questions

Our Team Tech interview Questions

1. Disadvantage of Typescript

- TypeScript sets aside a long effort to incorporate the code.
- TypeScript doesn't uphold theoretical classes.
- When utilizing an outsider library, there should be a definition document, and now and again it's not generally accessible.
- Nature of type definition documents is a worry.
- At whatever point TypeScript needs to run in a program, there should even now be a gathering step to change TypeScript to JavaScript.
- Enrolling TypeScript engineers may be a genuine annoyance, since a portion of the designers are not keen on learning TypeScript, being entirely alright with utilizing JavaScript. Groups that as of now use JavaScript, become profitable with TypeScript after around 2-3 months, and conversant in TypeScript simply after about a large portion of a year. So instructing staff can likewise be an obstacle.
- TypeScript isn't completely coexpressive with JavaScript. Missing highlights include: HOFs, Composition, Generics with Higher key.

2. What is optional type in Typescript?

The optional chaining operator (?.) accesses an object's property or calls a function. If the object is undefined or null, it returns undefined instead of throwing an error.

```
const adventurer = {
  name: 'Alice',
  cat: {
    name: 'Dinah'
  }
};const dogName = adventurer.dog?.name;
console.log(dogName);
// expected output: undefinedconsole.log(adventurer.someNonExistentMethod?.());
// expected output: undefined
```

3. Why can't you use async as first parameter in useEffect?

The issue here is that the first argument of useEffect is supposed to be a function that returns either nothing (undefined) or a function (to clean up side effects). But an async function returns a Promise, which can't be called as a function! **It's simply not what the useEffect hook expects for its first argument.**

4. What is encapsulation and abstraction in OOP and diff of them?

Abstraction is an OOP concept that focuses only on relevant data of an object. It hides the background details and emphasizes the essential data points for reducing the complexity and increase efficiency. It generally retains only information which is most relevant for that specific process. [Abstraction](#) method mainly focusses on the idea instead of actual functioning.

Encapsulation is a method of making a complex system easier to handle for end users. The user need not worry about internal details and complexities of the system. [Encapsulation](#) is a process of wrapping the data and the code, that operate on the data into a single entity. You can assume it as a protective wrapper that stops random access of code defined outside that wrapper.

- Abstraction is hiding the details and implementation of the code.
- Encapsulation is hiding the data and controlling the visibility of the code.

Phase

- Abstraction is a design level process.
- Encapsulation is an implementation level process.

Pivotal Ability

- Abstraction is concerned about what a class instance can do, instead of the implementation of the class.
- Encapsulation helps in data binding and control over maintaining the transparency of the data.

Use Case

- Abstraction is a design level process and it is used to reduce the complexity at the designing stage of a project.
- Encapsulation is an implementation level process, and it is used to provide privacy and maintain control over the transparency of data at the implementation stage of a project.

How can you optimize image for web? How can you use it in Angular and react?

Image optimization is about reducing the file size of your images as much as possible, without sacrificing quality, so your page load times remain low.

It's also about image SEO—that is, getting your banner and product images to rank highly on

Google and other image search engines. The goal of image optimization is to create high-quality images with the lowest file size. Three main elements play a role here: Image file size (JPEG vs. PNG vs. WEBP images)

Image compression level

Image height and width By finding the balance between the three, you can reduce image size and improve website performance. We can optimize images in React using [react-optimized-image](#) module. We can optimize images in Angular using [NgOptimizedImage](#) directive.

The NgOptimizedImage directive ensures that the loading of the Largest Contentful Paint image is prioritized by:

- Automatically setting the fetch priority attribute on the tag
- Lazy loading other images by default
- Asserting that there is a corresponding preconnect link tag in the document head

SPA vs regular web

Traditional web applications perform most of the application logic on the server side whereas single page applications (SPAs) perform most of the user interface logic in a web browser. Single Page Application communicate with the web server primarily using web APIs. The SPA only sends what you need with each click, and your browser renders that information. This is different than a traditional page load where the server rerenders a full page with every click you make and sends it to your browser. SPAs are faster than traditional web applications because they execute the logic in the web browser itself rather than on the server. And after the initial page load, only data is sent back and forth instead of the entire HTML that reduces the bandwidth.

Difference between regular function and arrow function

<https://dmitripavlutin.com/differences-between-arrow-and-regular-functions/>

- “This” keyword

“This” keyword is one of the most used keywords in Javascript.

In a regular function, “this” keyword changes according to the way that function is invoked.

But, in arrow functions don’t have their own “this”, and they don’t redefine the value of “this” within the function.

Regardless of how you execute arrow functions, “this” inside an arrow function always refers to “this” from the outer context. This means that “this” keyword is lexically bound in arrow functions

.- Constructors

Regular functions can easily construct objects with “new” keywords.

Arrow function cannot be used as a constructor

.- Argument object

Inside the body of a regular function, arguments is a special array-like object containing the list of arguments with which the function has been invoked.

On the other side, no arguments special keyword is defined inside an arrow function.

- Implicit return

Regular function: return expression statement returns the result from a function. If the return statement is missing, or there's no expression after the return statement, the regular function implicitly returns undefined.

Arrow function: also can return values from the arrow function but with one useful exception.

If the arrow function contains one expression, and you omit the function's curly braces, then the expression is implicitly returned.

Have you setup CI/CD process for your product and what tools did you use?

<https://www.bmc.com/blogs/ci-cd-pipeline-setup/>

Yes, I have experience in building a four-stage CI/CD pipeline.

The main stages were Source, Build, Test, and Deployment

Our team developed the code and committed the changes to a centralized code repository using Git.

And we used Jenkins to carry out the automated build and testing. If any issues were detected while building or testing, Jenkins automatically informed via Slack.

And then, the final package was uploaded to AWS Elastic Beanstalk.

The elastic beanstalk managed the provisioning of infrastructure, load balancing, and scaling of the required resource type, such as EC2, RDS, or others.

What did you do next to push your build into the server?

https://www.phusionpassenger.com/library/walkthroughs/deploy/nodejs/ownserver/nginx/oss/trusty/deploy_app.html

- Transferring the app code to the server

Push your code to a Git repository

Login to your server, create a user for the app

Install Git on the server

Pull code

- Preparing the app's environment

Login as the app's user

Install app dependencies

Difference between SQL and NoSQL databases

<https://www.mongodb.com/nosql-explained/nosql-vs-sql>

- Data Storage Model

SQL: Tables with fixed rows and columns

NoSQL: Document(JSON documents), Key-value(key-value pairs), Wide-column(tables with rows and dynamic columns), Graph(nodes and edges)

- Development History

SQL: Developed in the 1970s with a focus on reducing data duplication

NoSQL: Developed in the late 2000s with a focus on scaling and allowing for rapid application change driven by agile and DevOps practices.

- Examples

SQL: Oracle, MySQL, Microsoft SQL Server, and PostgreSQL

NoSQL: Document(MongoDB and CouchDB), Key-value(Redis and DynamoDB), Wide-column(Cassandra and HBase), Graph(Neo4j and Amazon Neptune)

- Primary Purpose

SQL: General purpose

NoSQL: Document(general purpose), Key-value(large amounts of data with simple lookup queries), Wide-column(large amounts of data with predictable query patterns), Graph(analyzing and traversing relationships between connected data)

- Schemas

SQL: Rigid

NoSQL: Flexible

- Scaling

SQL: Vertical (scale-up with a larger server)

NoSQL: Horizontal (scale-out across commodity servers)

- Multi-Record ACID Transactions

SQL: Supported

NoSQL: Most do not support multi-record ACID transactions. However, some — like MongoDB — do.

- Joins

SQL: Typically required

NoSQL: Typically not required

- Data to Object Mapping

SQL: Requires ORM (object-relational mapping)

NoSQL: Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages.

What about Sharding and cluster.

<https://www.mongodb.com/basics/clusters> A sharded cluster in MongoDB is a collection of datasets distributed across many shards (servers) in order to achieve horizontal scalability and better performance in read and write operations. Sharding is very useful for collections that have a lot of data and high query rates.

What do you know about Progressive rendering?

[https://medium.com/the-thinkmill/progressive-rendering-the-key-to-faster-web-ebfbbece41a4#:~:text=Progressive%20Rendering%20\(aka%20Progressive%20Server,it's%20rendered%20on%20the%20server.](https://medium.com/the-thinkmill/progressive-rendering-the-key-to-faster-web-ebfbbece41a4#:~:text=Progressive%20Rendering%20(aka%20Progressive%20Server,it's%20rendered%20on%20the%20server.) Progressive Rendering (aka Progressive Server Side Rendering) is a technique in which once you render the critical content on the server, you start streaming it to the client without waiting for non-critical content. You then stream the non-critical content later once it's rendered on the server.

What about HTML semantic principles?

<https://medium.com/adalab/the-importance-of-semantic-html-78e74fb75ff0>
<https://www.thoughtco.com/why-use-semantic-html-3468271>
<https://localhost.dev/blog/the-right-tag-for-the-job-why-you-should-use-semantic-html/> Semantic HTML introduces meaning to the web page rather than just presentation. Most important advantages of semantic markup:

- Clearer code and easier to maintain
- Help the site be accessible
- Improve SEO positioning

Why is it so important to use correct tags in HTML?

<https://dev.to/polgarj/a-short-guide-to-help-you-pick-the-correct-html-tag-5619>

A semantically correct HTML tag informs the browser and the developer about its content. It also helps keyboards and assistive technologies to navigate through a website.

What about shadow root element?

<https://www.geeksforgeeks.org/what-is-shadow-root-and-how-to-use-it/>
<https://javascript.info/shadow-dom>

Shadow tree – a hidden DOM subtree, not reflected in HTML, hidden from prying eyes. The shadow root, returned by `attachShadow`, is like an element: we can use `innerHTML` or DOM methods, such as `append`, to populate it. The element with a shadow root is called a “shadow tree host”, and is available as the `shadowRoot` property.

What methodologies or techniques do you use for styling css and the main purpose?

<https://blog.logrocket.com/the-most-in-demand-css-technologies-in-2020/>

styled-components

styled-components is a CSS-in-JS library that helps you use visual primitives and the best of the modern browser technologies to style your React components.

Sass

Sass reduces repetition of CSS and therefore saves time.

Less

converts your Less styles to CSS styles

Bootstrap

Bootstrap is an open-source, mobile-first toolkit for developing with HTML, CSS, and JS. You can quickly prototype your ideas or build your entire app with Sass variables and mixins, the responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Tailwind CSS

Tailwind CSS is a utility-first CSS framework for rapidly building custom designs.

What is the closure in Javascript?

Closures are created whenever a variable that is defined outside the current scope is accessed from within some inner scope.

It gives you access to an outer function's scope from an inner function.

In JavaScript, closures are created every time a function is created.

To use a closure, simply define a function inside another function and expose it.

Difference between private vs protected?

Private: These members are only accessible within the class that instantiated the object.

Protected: This keyword allows a little more access than private members but a lot less than the public.

A protected member is accessible within the class (similar to private) and any object that inherits from it.

What about polyfills in Javascript?

A polyfill is a piece of code that implements the features that you expect the browser to support natively.

Polyfills usually emulate a newer API that provides fallback functionality to older browsers.

Polyfills can also make a feature that different browsers implement differently work the same way in each browser.

What is state management library in React?

State management is one of the most important aspects of every app. The app's state dictates what users see, how the app looks, what data is stored, and so on. Redux is one of the most used libraries in state management for React applications. Redux has a lot of middleware libraries available in React such as Redux-Saga and Redux thunk, it's especially used in handling API calls and side effects. We can write a standard way to redux logic using ReactJS/toolkit.

State vs global context.

Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language. For example, in the code below we manually thread through a “theme” prop in order to style the Button component: class App extends React. But we can use the state within only the components where it is defined.

How do you build a good React architecture?

Here are the best practices to follow using React architecture:

- Avoid using a lot of nested files & folders.
- Reuse your components.
- Put CSS in JavaScript.
- Comment only where necessary.
- Use capitals for Component Names.
- Use React's Bit tool.
- Use snippet libraries.

What about solid design principles?

SOLID is an acronym that stands for five key design principles: single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle. All five are commonly used by software engineers and provide some important benefits for developers.

The Liskov Substitution Principle (LSP) states that objects of a superclass should be replaceable with objects of its subclasses without breaking the application. In other words,

what we want is to have the objects of our subclasses behaving the same way as the objects of our superclass.

What about open-closed principle?

Open-closed Principle (OCP) states: Objects or entities should be open for extension but closed for modification.

This means that a class should be extendable without modifying the class itself. The Open Close Principle encourages software developers to design and write code in a fashion that adding new functionality would involve minimal changes to existing code.

Most changes will be handled as new methods and new classes.

What do you know about microservices?

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs.

These services are owned by small, self-contained teams.

The services communicate with clients, and often each other, using lightweight protocols, often over messaging or HTTP.

Why microservices are used?

Microservices allow developers to become language and technology agnostic.

Different team members can use different programming languages and technologies when debugging and coding.

Monitoring microservices lets you check your architecture for service and performance and identify future debugging problems.

The ways of interactions between microservices.

The most common type is single-receiver communication with a synchronous protocol like HTTP/HTTPS when invoking a regular Web API HTTP service.

Microservices also typically use messaging protocols for asynchronous communication between microservices. Microservices are distributed and microservices communicate with each other by inter-service communication on network level. Each microservice has its own instance and process. Therefore, services must interact using an inter-service communication protocols like HTTP, gRPC or message brokers AMQP protocol.

What is middleware in Express?

Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-

response cycle. The next middleware function is commonly denoted by a variable named next.

These functions are used to modify req and res objects for tasks like parsing request bodies, adding response headers, etc.- An Express application can use the following types of middleware:

Application-level middleware

Router-level middleware

Error-handling middleware

Built-in middleware

Third-party middleware-Most used Middleware

CORS, CSRF, Auth- Advantages of using middleware:

Middleware can process request objects multiple times before the server works for that request.

Middleware can be used to add logging and authentication functionality.

Middleware improves client-side rendering performance.

Middleware is used for setting some specific HTTP headers.

Middleware helps for Optimization and better performance.

Is it possible to access to filesystem in Node.js?

We can access the file system in NodeJS using some inbuilt modules. File System: A file is a collection of related information stored in secondary storage Or a file is a collection of similar types of entities that work with the file and manage it also called a File System.Ex]

File System Module (fs module): One of the popular inbuilt modules for working with the file system in NodeJS is the file system module which is also called the “fs” module in short. fs module is very powerful for doing any task in NodeJS related to File systems.

Access the File System in Node JS means performing some basic operations on files. This is also known as CRUD operations.

Why the password should be hashed?

Password hashing is defined as putting a password through a hashing algorithm (bcrypt, SHA, etc) to turn plaintext into an unintelligible series of numbers and letters. This is important for basic security hygiene because, in the event of a security breach, any compromised passwords are unintelligible to the bad actor.- What is the main purpose of hashing?

Hashing is a cryptographic process that can be used to validate the authenticity and integrity of various types of input. It is widely used in authentication systems to avoid storing plaintext passwords in databases, but is also used to validate files, documents and other types of data.

Is it possible to decrypt password?

Encryption means the data (such as the password) can be decrypted if you have the right key. Most passwords, however, cannot be decrypted since they weren't encrypted in the first place. Instead, one might be able to recover them by running a lengthy attack.

Tell me about JavaScript data types.

Primitive values (immutable datum represented directly at the lowest level of the language) Boolean type. Null type. Undefined type. Number type. BigInt type. String type. Symbol type.
Objects (collections of properties)

What is complex data type? and complex data type vs primitive data type?

A complex data type is usually a composite of other existing data types. For example, you might create a complex data type whose components include built-in types, opaque types, distinct types, or other complex types.
The primitive data types are types like string , integer , boolean , etc. that can take a single value. The complex types consist of multiple values grouped together.

What is Const used for?

The const declaration creates a read-only reference to a value. It does not mean the value it holds is immutable—just that the variable identifier cannot be reassigned. For instance, in the case where the content is an object, this means the object's contents (e.g., its properties) can be altered.

What is difference between arrow function and general function?

Regular functions created using function declarations or expressions are constructible and callable. Since regular functions are constructible, they can be called using the new keyword. However, the arrow functions are only callable and not constructible, i.e arrow functions can never be used as constructor functions.

Tell me about async await.

An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains

.- When should I use async await JavaScript?

Inside an async function, you can use the await keyword before a call to a function that returns a promise. This makes the code wait at that point until the promise is

settled, at which point the fulfilled value of the promise is treated as a return value, or the rejected value is thrown

- What is difference between async and await in JavaScript?

The async keyword is used to define an asynchronous function, which returns a AsyncFunction object. The await keyword is used to pause async function execution until a Promise is fulfilled, that is resolved or rejected, and to resume execution of the async function after fulfillment.

- How does async await work?

The async keyword turns a method into an async method, which allows you to use the await keyword in its body. When the await keyword is applied, it suspends the calling method and yields control back to its caller until the awaited task is complete. await can only be used inside an async method.

Why do we use TypeScript?

The reason is that Typescript has some of these advantages.

- TypeScript always points out the compilation errors at the time of development (pre-compilation). Because of this getting runtime errors is less likely, whereas JavaScript is an interpreted language.
- TypeScript supports static/strong typing. This means that type correctness can be checked at compile time. This feature is not available in JavaScript.
- TypeScript is nothing but JavaScript and some additional features i.e. ES6 features. It may not be supported in your target browser but the TypeScript compiler can compile the .ts files into ES3, ES4, and ES5 also.

The advantages of class component and functional component?

- The advantages of Class Component.
- Make use of ES6 class and extend the Component class in react.
- Its tend to implement logic and state because of that its called "smart" or "stateful" component.
- React life cycle methods can be used within the class component.
- Can pass props and can be access props by using this. props.
- The advantages of Functional Component.
- Easier to test: You don't have to worry about hidden state and there aren't as many side effects when it comes to functional components, so for every input, the functions will have exactly one output.
- Easier to read/write: The syntax is less complex than that of class components, and it's easier to read due to knowing how much you can't do with functional components

already. The use of prop destructuring makes it really beneficial to see what's going on and what's coming out of the component.

- Easier to debug: Again, functional components depend on the props they are given and don't rely on state. There isn't a reason to `console.log()` your state to constantly understand the changes that are going on.

Which tools do you use to optimize component?

Basically, we use five tools to optimize our components.

Profiler, React Developer Tools, Bit, WellDone Software and Performance Timeline.

About useCallback hook and rest API?

The useCallback hook is used when you have a component in which the child is re-rendering again and again without need.

It passes an inline callback and an array of dependencies. useCallback returns a memoized version of the callback that only changes if one of the dependencies has changed. This is useful when passing callbacks to optimized child components that rely on reference equality to prevent unnecessary renders.

How to integrate Cypress test to AWS?

End-to-end testing is a type of functional testing that focuses on testing the entire system from start to finish to ensure that all components work together as expected.

This type of testing is usually done by simulating real-world scenarios and testing how the system responds. Unit, integration and system testing, on the other hand, focus on testing individual components or groups of components in isolation from the rest of the system.

Explain storefront in Vue.

Vue Storefront is a headless and backend-agnostic eCommerce Progressive Web App written in Vue.js. The fact that it uses headless architecture allows Vue Storefront to be connected with any eCommerce platform, so it can be a frontend PWA for Magento, Shopify, BigCommerce, WooCommerce, etc.

It's a very popular Open Source project with a strong and growing community.

Difference between composition API and option API.

The difference between composition API and option API is syntax. The Composition API lets us create components without the need of a large single exportable object, like in the Options API. For example, if we wanted to make a simple counter component with the Options API, it would look like the code below.

Explain slot in Vue.

We use slots to display content passed down from one component to another. There are two types of slot in vuejs namely: named slot and unnamed(default) slot.

Explain about mixin in Vue.

Mixins are a flexible way to distribute reusable functionalities for Vue components. A mixin object can contain any component options. When a component uses a mixin, all options in the mixin will be “mixed” into the component's own options.

Explain about Reactivity in Vue.

In Vue 3, the reactive API has been rewritten to solve many of the shortcomings of Vue 2. The reactive system was rewritten to utilize JavaScript’s proxies. A proxy acts as a wrapper around an object or function that intercept operations, like getting and setting a value.

With the new reactive system in Vue 3, there is now better support for observing changes in data using proxy objects. Proxy objects enable you to create a proxy for another object, which can intercept and redefine fundamental operations for that object. With proxies, we can intercept operations like get and set and immediately see when data is accessed or changed.

About server side rendering , How can you solve issue for implement SSR?

Server-side rendering (SSR) is an application’s ability to convert HTML files on the server into a fully rendered HTML page for the client. The web browser submits a request for information from the server, which instantly responds by sending a fully rendered page to the client. Search engines can crawl and index content prior to delivery, which is beneficial for Search Engine Optimization purposes.

Popular examples of server-side rendering JavaScript frameworks include: Angular server side rendering, ejs server side rendering, server side rendering Express, Gatsby server side rendering, Google server side rendering, NestJS server side rendering, Next server side rendering, Nuxt server side rendering, React server side rendering, and Vue server side rendering.

What are the Benefits of Server-Side Rendering?

Some server-side rendering advantages include:

A server-side rendered application enables pages to load faster, improving the user experience.

When rendering server-side, search engines can easily index and crawl content because the content can be rendered before the page is loaded, which is ideal for SEO.

Webpages are correctly indexed because web browsers prioritize web pages with faster load times.

Rendering server-side helps efficiently load webpages for users with slow internet connection or outdated devices.

What are the Risks of Server-Side Rendering?

Server-side rendering disadvantages may include:

Rendering server-side can be costly and resource-intensive as it is not the default for JavaScript websites, and the server takes on the full burden of rendering content for users and bots.

While rendering static HTML server-side is efficient, rendering bigger, more complex applications server-side can increase load times due to the bottleneck.

Server-side rendering may not be compatible with third-party JavaScript code.

Rendering server-side may be ideal for static site generation, but frequent server requests and full page reloads can result in overall slower page rendering in more complex applications.

Explain about Hoisting in javascript.

In JavaScript, Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution. Basically, it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local

Explain about tree shaking in javascript.

Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code. It relies on the import and export statements to detect if code modules are exported and imported for use between JavaScript files.

Tree shaking, also known as dead code elimination, is the practice of removing unused code in your production build. It's important to ship as little code to your end-users as possible. By statically analyzing our source code, we can determine what's not being used and exclude it from our final bundle.

how can you implement security authentication?

One-Time Password (OTP)

An OTP and its sibling, time-based one-time passwords (TOTP), are unique temporary passwords. Using an OTP means that hackers won't be able to use your stolen credentials since only your username will be valid. This is a way to significantly protect sensitive data, such as banking credentials.

Biometrics Authentication

Biometric scans are a common authentication method in large companies. Your fingerprint, face pattern, hand geometry, and eyes are all unique to you and stealing them is almost impossible. You don't even need those ominous machines you see in old sci-fi films – with the right calibration, a smartphone will do the job. Biometric authentication is bullet-proof since stealing your physical traits is much harder than hacking a password, text message, or smartphone.

Continuous Authentication

Continuous authentication means what its name suggests: it regularly identifies you during a session. This is likely familiar to those who often use online banking services, as most require you to enter your authentication code when signing in and then again to validate a transfer. When used with other online accounts, this form of authentication monitors your behavior and regularly verifies your identification by asking for your password, generating a unique password again, or requesting a biometric scan. While it offers increased security due to the repetitive nature of its authentication, it also faces the same problems as the methods previously mentioned.

The Three Factors of Authentication

There are three authentication factors to talk about when you use any of these methods: knowledge, possession, and inherence. The knowledge factor is the most self-explanatory, as it involves authentication based on information you already know. This can be anything: usernames, passwords, the name of your favorite childhood action hero, the ultimate question of life, etc. The more information you provide – that is, answering numerous personalized questions – the harder this factor is to crack, making it a great first line of defense. The possession factor refers to a physical item, such as the device you use for work, your personal smartphone, or a security key. The inherence factor is closely connected to biometric authentication, as it's something specific to you. It can involve any physical trait, such as your fingerprint, retina, face, or even voice.

How can you approach to tesing API and Timer?

Gathering API requirements

It is necessary to gather the API's requirements before testing. You may use API to API comparison or database verification to evaluate input and output data, depending on where and how an API fits into the overall application process.

Discovery Testing

API integration and API performance tests are impossible without API discovery. When it comes to API discovery, QA engineers need to find the tiniest of elements that make everything work as it should.

At this stage of API discovery, you should figure out the best way to get to API testing in the first place.

Choosing the right API testing methodology can help reduce and eliminate the amount of API noise in various applications, software, and websites. To verify API calls defined in R&D and client requirements, the discovery step requires human execution of API calls.

Define the API requirements

Writing good requirements is very hard. Therefore, it needs open communication to define proper requirements as every stakeholder can have useful input on the requirements. Also, they may pose some tough questions that need answering

This is why teamwork matters — developers, testers, system architects, analysts, product owners... they all know their domain best and should be included in creating requirements.

Data Input → Output → Outcome Validation and Verification

Compared to managing the quantity of test data created and subsequently confirmed, each of these activities is relatively easy. Methods for verifying test data may be challenging and make or break an API and application's testing process. API testing automation can be useful here.

Advantage of TDD

Fewer bugs and errors are the primary benefit of the TDD approach. When the code has fewer bugs, you'll spend less time fixing them than other programming methodologies. TDD produces a higher overall test coverage and, therefore to a better quality of the final product.

Explain Javascript function context. If run function without context, what will execute?

In JavaScript, "context" refers to an object. Within an object, the keyword "this" refers

About automatically memory clean?

The Set. clear() method in JavaScript is used for the removal of all the elements from a set and make it empty. No arguments are required to be sent as parameters to the Set. clear() method and it returns an undefined return value.to that object (i.e. “self”), and provides an interface to the properties and methods that are members of that object. When a function is executed, the keyword “this” refers to the object that the function is executed in.

```
mySet.clear()
```

Difference between Javascript code in browser and node.js.

JavaScript is a simple programming language that runs in any browser JavaScript Engine. Whereas Nods.js is an interpreter or running environment for a JavaScript programming language that holds many excesses, it requires libraries that can easily be accessed from JavaScript programming for better use.

About DOM. How can you interact with it?

DOM stands for Document Object Model, which is a programming interface for web documents. It represents the page so that program can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

The easiest way to access a single element in the DOM is **by its unique ID**. We can get an element by ID with the getElementById() method of the document object.

There are five methods for accessing elements in a document.

1. Method one: getElementById()
2. Method two: getElementByClassnName()
3. Method three: getElementByTagName
4. Method four: querySelector()
5. Method five: querySelectorAll()

About Webworker.

A web worker is a JavaScript that runs in the background, independently of other scripts that may also have been executed from the same HTML page, without affecting the performance of the page.

We can spawn as many workers as we wish. We can pass data to the script being executed in the worker threads and also return value to the main thread upon completion.

Difference between local storage, session storage and cookies.

For most cases, we use the local Storage object if we want some data to be on the browser. If we want it on the server, then we use cookies, and the session storage is used when we want to destroy the data whenever that specific tab gets closed or the session is closed by the user.

Feature	Cookies	localStorage	sessionStorage
Maximum data size – more info	4 kB	5 MB	5 MB
Blockable by users – more info	yes	yes	yes
Auto-expire option – more info	yes	no	yes
Supported data types – more info	string only	string only	string only
Browser support – more info	very high	very high	very high
Accessible server-side – more info	yes	no	no
Data transferred on every HTTP request – more info	yes	no	no
Editable by users – more info	yes	yes	yes
Supported on SSL – more info	yes	n/a	n/a
Can be accessed on – more info	server-side & client-side	client-side only	client-side only
Clearing / deleting – more info	PHP, JS & automatic	JS only	JS & automatic
Lifetime – more info	as specified	till deleted	till tab is closed
Secure data storage – more info	no	no	no

About memory leak issue. How can you solve it?

? :: In simple words, a memory leak is an allocated piece of memory that the JavaScript engine is unable to reclaim. The JavaScript engine allocates memory when you create objects and variables in your application, and it is smart enough to clear out the memory when you no longer need the objects. Memory leaks are caused due to flaws in your logic, and they make way for poor performance in your application.

Types of Memory leak.

- Undeclared or accidental global variables
- More great articles from LogRocket:
- Closures
- Forgotten timers
- Out of DOM reference
- Identify memory leaks using Chrome DevTools
- Visualize memory consumption using the performance profiler
- Identify detached DOM nodes

The way to solve.

- Garbage collector

The main cause of memory leaks in an application is due to unwanted references. The garbage collector finds the memory that is no longer in use by the program and releases it back to the operating system for further allocation.

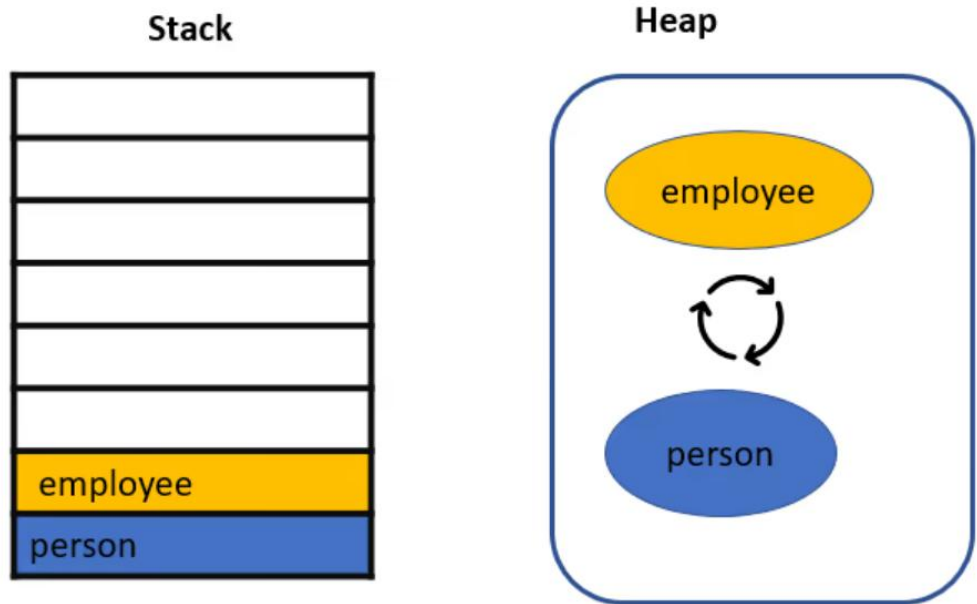
To know what is an unwanted reference, first, we need to get an idea of how garbage collection determines that a piece of memory is unreachable. Garbage collection uses two main algorithms to find unwanted references and unreachable code, reference count and mark-and-sweep.

- Mark-and-sweep algorithm

The mark-and-sweep algorithm follows two steps. First, in JavaScript, the root is the global object. The garbage collector periodically starts from the root and finds all objects that are referenced from the root. It will mark all the reachable objects **active**. Then, garbage collection frees the memory for all objects that are not marked as **active**, returning the memory to the operating system.

- Reference count

The reference count algorithm looks for objects that have no references. An object can be released if it has zero references pointing to it.



Difference between event bubbling and propagation.

Event propagation describes the order in which the events travel the DOM tree when the web browser fires them.

Assume there is a form tag inside a div tag, and both have onclick event listeners. Event Propagation describes how one event listener may fire after the other.

There are two types of propagation:

1. Event bubbling, by which events bubble upwards, from child to parent.
2. Event capturing, by which events travel downwards, from parent to child.

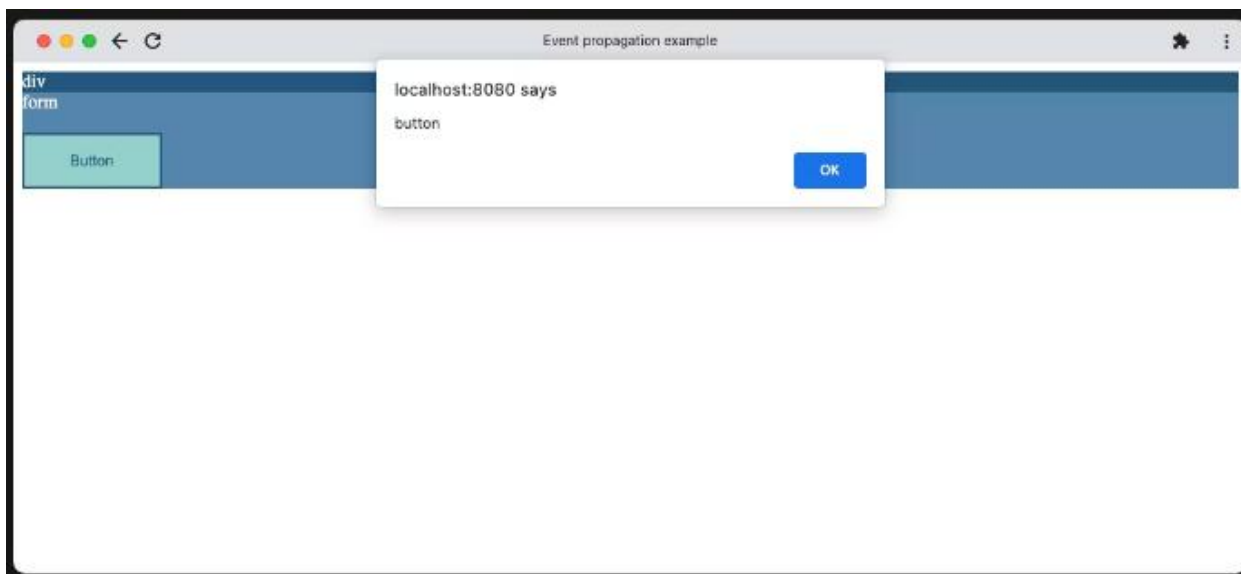
Event bubbling means the direction of event propagation will be from the child element to its parent element.

Consider the following example. It has three nested elements: div, form, and button. Each element has a click event listener. The browser displays an alert with a message when you click on each element.

By default, the order in which the web browser displays alerts will be button, form, then div. The events bubble up from child to parent.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Event propagation example</title>
</head>
<body>
<div id="div" style="background:#22577E; color:white">
  div
  <form id="form" style="background:#5584AC; color:white">
    form <br/><br/>
    <button id="button" style="background:#95D1CC; color:#22577E; border: solid 2px #22577E;
padding: 15px 40px;">
      Button
    </button>
  </form>
</div>
<script>
const div = document.getElementById("div");
const form = document.getElementById("form");
const button = document.getElementById("button");

div.addEventListener("click", ()=>{
  alert("div");
});
form.addEventListener("click", ()=>{
  alert("form")
});
button.addEventListener("click", ()=>{
  alert("button")
});
</script>
</body>
</html>
```



Why is react library, not framework?

Because it is just a library for views and not complete, in the sense that if you need to add routes to your app, you need to install a package, if you need to make http requests without the native fetch API, you need to install another library, frameworks like Angular comes with all these internals,

About advantage of react.

- **Flexibility:** it allows developers to build upon any desired technology stack. Therefore, unlike other frameworks, developers are free of restrictions to rigid architectures and workflows.
- **Performance optimization:** declarative programming enhances performance by creating a virtual DOM by allowing the developer to write HTML directly into JavaScript files. The virtual DOM is much easier to manipulate than the real DOM and so that improves efficiency. Answers to ReactJS coding interview questions such as this should explain how it enables building rich user interfaces for mobile and web applications.
- **SEO Efficiency:** it is used for building web applications; therefore, SEO is a strong factor in considering it. Thankfully, it is SEO-friendly. This is despite the fact that search engines commonly have problems reading apps that are heavy on JavaScript. The virtual DOM feature makes this less of a challenge. This may also feature in the React JS advanced interview questions and answers section.
- **Productivity Improvement:** developers commonly describe as the best feature the ability to reuse system components. This helps them to deal with issues concerning the library's

complex logic. For companies, the ability to reuse code assets helps in setting internal coding standards and app consistency across the board.

- **Learn once – write anywhere:** while React is popular for web development, there is the React Native framework, which enables building web applications. Again, this improves cross-platform consistency and efficiency. It also improves the work of developers, since, for all platforms, they are working with similar tools. This is an important part of React frontend developer interview questions.

Difference between state and props.

Properties (props) are read-only. They determine the configuration of a component as received from the parent. If React components are functions, you may think of props as arguments for a function.

Props remain immutable throughout the lifecycle of a component.

On the other hand, the state is dynamic. Components don't have a default state passed down from parents. Hence, the state of a component is mutable, representing a data value that can be updateable

when certain conditions are fulfilled. Thus, the state is essentially what makes dynamic rendering possible.

What is high order component?

Higher-Order Components are used to add sophisticated functionalities to existing components. The purpose of the HOC technique, which is common in third-party React libraries, is to reuse component logic.

Therefore, the use of every Higher-Order component function returns a new component. HOCs in

React are related to higher-order functions in JavaScript. Higher-order components find implementation in Redux connect (which is a higher-order function), state abstraction, and so on.

Why is it better to use fragment instead of div tag?

The main difference between `React.Fragment` vs `div` is that **using a `React.Fragment` will not add any additional elements into the DOM tree, whereas, using a `div` will add a `div` to the DOM tree.**

With `React.Fragment`, you can render multiple elements of a component without adding extra `div` tags. We can write cleaner, more readable code with `React.Fragments`. It takes up less memory and renders components faster. Each component is rendered as expected.

About reactivity data.

Reactive Programming in JavaScript is like **a sequence of events that occur in time**. It is an Asynchronous Programming concept around data streams. Reactive programming is a programming paradigm for writing code, mainly concerned with asynchronous data streams. It is a programming of event streams that happens in time.

Reactivity is **the ability for a variable (array, string, number, object, etc) to update when its value or any other variable that it makes reference to is changed after declaration**.

About Http protocol. What is main part of it?

The Hypertext Transfer Protocol is [an application protocol](#) for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web.

As a request-response protocol, HTTP gives users a way to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers. HTTP clients generally use [Transmission Control Protocol \(TCP\)](#) connections to communicate with servers.

HTTP utilizes [specific request methods](#) in order to perform various tasks. All HTTP servers use the GET and HEAD methods, but not all support the rest of these request methods:

- GET requests a specific resource in its entirety
- HEAD requests a specific resource without the body content
- POST adds content, messages, or data to a new page under an existing web resource
- PUT directly modifies an existing web resource or creates a new URI if need be
- DELETE gets rid of a specified resource
- TRACE shows users any changes or additions made to a web resource
- OPTIONS shows users which HTTP methods are available for a specific URL
- CONNECT converts the request connection to a transparent TCP/IP tunnel
- PATCH partially modifies a web resource

Difference between Http and Https.

HTTPS is HTTP with encryption and verification. The only difference between the two protocols is that HTTPS uses TLS (SSL) to encrypt normal HTTP requests and responses, and to digitally sign those requests and responses. As a result, HTTPS is far more secure than HTTP.

About content security policy.

Content Security Policy (CSP) is an added layer of security that **helps to detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks**. These attacks are used for everything from data theft, to site defacement, to malware distribution.

The primary benefit of CSP is **preventing the exploitation of cross-site scripting vulnerabilities**. When an application uses a strict policy, an attacker who finds an XSS bug will no longer be able to force the browser to execute malicious scripts on the page.

About eval method in Javascript.

The `eval()` function evaluates JavaScript code represented as a string and returns its completion value.

```
console.log(eval('2 + 2'));  
  
// expected output: 4  
  
console.log(eval(new String('2 + 2')));  
  
// expected output: 2 + 2  
  
console.log(eval('2 + 2') === eval('4'));  
  
// expected output: true  
  
console.log(eval('2 + 2') === eval(new String('2 + 2')));  
  
// expected output: false
```

What is good and bad unit test?

Advantages of unit testing are that it reduces or prevents production bugs, increases developer productivity, encourages modular programming.

- **It helps you write better code.**
- **It helps you catch bugs earlier.**
- **It helps you detect regression bugs.**
- **It makes your code easy to refactor.**
- **It makes you more efficient at writing code.**

Disadvantages are that it is time-consuming, can't be challenging to cover all the code, and won't catch all bugs.

- **It takes time to write test cases.**
- **It's difficult to write tests for legacy code.**
- **Tests require a lot of time for maintenance.**
- **It can be challenging to test GUI code.**
- **Unit testing can't catch all errors.**

What is purpose of webpack?

This is why webpack exists. It's a tool that **lets you bundle your JavaScript applications** (supporting both ESM and CommonJS), and it can be extended to support many different assets such as images, fonts and stylesheets.

It helps to use different javascript files without having tension that will load first. It makes code shorter. It helps in converting many files other than javascript into modules. It compiles different javascript module.

About Babel.

Babel is a **JavaScript compiler**

Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments.

JSX is a syntactical extension to JavaScript. It comes with the full power of JavaScript. JSX produces React elements and can embed any JavaScript expression. Ultimately, the JSX is converted to JavaScript using compiler/transformers. Babel can convert JSX syntax!

What is bad side of react?

React contains a myriad of tools used to develop and create user interfaces. However, unlike frameworks, React JS is not an 'all-in-one-tool' for app development. Therefore, if you happen to use a model-view-controller (MVC), React development is only responsible for the UI. The developers have to create the model and controller part by using other tools. To create a functional application, you will have to apply more tools that will cover other crucial parts like backend and data storage.

How can you deal with accessibility?

Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them. More specifically, people can:

- perceive, understand, navigate, and interact with the Web
- contribute to the Web

How to Optimize Your Website for Disabled Users

1. 0

2. Allow Users to Enlarge Font Sizes

3. Keep Contrast Sensitivity in Mind

4. Add Keyboard Navigation

5. Make Video and Multimedia Accessible

6. Use Descriptive URLs

7. Use ARIA Roles

8. Avoid Using Placeholder Text in Forms

9. Minimize the Use of Tables

About progressive web Apps.

Progressive Web Apps (PWAs) are web apps that use service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience on par with native apps.

PWAs provide a number of [advantages](#) to users — including being [installable](#), [progressively enhanced](#), [responsively designed](#), [reengageable](#), [linkable](#), [discoverable](#), [network independent](#), and [secure](#).

What about factory design pattern

The factory method pattern provides an interface for creating objects that can be modified after creation. The cool thing about this is that the logic for creating our objects is centralized in a single place, simplifying and better organizing our code.

What is cross-site scripting?

Cross site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it. If the app or website lacks proper data sanitization, the malicious link executes the attacker's chosen code on the user's system. As a result, the attacker can steal the user's active session cookie.

<https://www.synopsys.com/glossary/what-is-cross-site-scripting.html>

What does CORS mean?

Cross-Origin Resource Sharing (CORS) is an [HTTP](#)-header based mechanism that allows a server to indicate any [origins](#) (domain, scheme, or port) other than its own from which a browser should permit loading resources. CORS also relies on a mechanism by which browsers make a "preflight" request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

An example of a cross-origin request: the front-end JavaScript code served from <https://domain-a.com> uses [XMLHttpRequest](#) to make a request for <https://domain-b.com/data.json>.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the [Fetch API](#) follow the [same-origin policy](#). This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

What is git stash command?

Use `git stash` when you want to record the current state of the working directory and the index, but want to go back to a clean working directory. The command saves your local modifications away and reverts the working directory to match the `HEAD` commit.

The modifications stashed away by this command can be listed with `git stash list`, inspected with `git stash show`, and restored (potentially on top of a different commit) with `git stash apply`. Calling `git stash` without any arguments is equivalent to `git stash push`. A stash is by default listed as "WIP on **branchname** ... ", but you can give a more descriptive message on the command line when you create one.

The latest stash you created is stored in `refs/stash`; older stashes are found in the reflog of this reference and can be named using the usual reflog syntax (e.g. `stash@{0}` is the most recently created stash, `stash@{1}` is the one before it, `stash@{2.hours.ago}` is also possible). Stashes may also be referenced by specifying just the stash index (e.g. the integer `n` is equivalent to `stash@{n}`).

<https://git-scm.com/docs/git-stash>

84. What is git merge command?

Incorporates changes from the named commits (since the time their histories diverged from the current branch) into the current branch. This command is used by `git pull` to incorporate changes from another repository and can be used by hand to merge changes from one branch into another.

Assume the following history exists and the current branch is "`master`":

```
A---B---C topic
/
D---E---F---G master
```

Then "`git merge topic`" will replay the changes made on the `topic` branch since it diverged from `master` (i.e., `E`) until its current commit (`C`) on top of `master`, and record the result in a new commit along with the names of the two parent commits and a log message from the user describing the changes.

```
A---B---C topic
/      \
         
```

D---E---F---G---H master

The second syntax ("`git merge --abort`") can only be run after the merge has resulted in conflicts. `git merge --abort` will abort the merge process and try to reconstruct the pre-merge state. However, if there were uncommitted changes when the merge started (and especially if those changes were further modified after the merge was started), `git merge --abort` will in some cases be unable to reconstruct the original (pre-merge) changes. Therefore:

Warning: Running `git merge` with non-trivial uncommitted changes is discouraged: while possible, it may leave you in a state that is hard to back out of in the case of a conflict.

The third syntax ("`git merge --continue`") can only be run after the merge has resulted in conflicts.

<https://git-scm.com/docs/git-merge>

85. git merge vs rebase

Git Merge Vs Git Rebase:

Merge	Rebase
Git merge is a command that allows you to merge branches from Git.	Git rebase is a command that allows developers to integrate changes from one branch to another.
In Git Merge logs will be showing the complete history of the merging of commits.	Logs are linear in Git rebase as the commits are rebased
All the commits on the feature branch will be combined as a single commit in the master branch.	All the commits will be rebased and the same number of commits will be added to the master branch.
Git Merge is used when the target branch is shared branch	Git Rebase should be used when the target branch is private branch

<https://www.edureka.co/blog/git-rebase-vs-merge/#Git-Merge-vs-Git-Rebase>.

86. What about strict mode?

Strict mode was a new feature in ECMAScript 5 that allows you to place a program, or a function, in a "strict" operating context.

This strict context prevents certain actions from being taken and throws more exceptions. The statement "use strict": instructs the browser to use the Strict mode, which is a reduced and safer feature set of JavaScript.

Benefits of using ‘use strict’: Strict mode makes several changes to normal JavaScript semantics.

- Strict mode eliminates some JavaScript silent errors by changing them to throw errors.
- Strict mode fixes mistakes that make it difficult for JavaScript engines to perform optimizations: strict mode code can sometimes be made to run faster than identical code that’s not strict mode.
- Strict mode prohibits some syntax likely to be defined in future versions of ECMAScript.
- It prevents, or throws errors, when relatively “unsafe” actions are taken (such as gaining access to the global object).
- It disables features that are confusing or poorly thought out.
- Strict mode makes it easier to write “secure” JavaScript.

87. What about garbage collector in JS?

The main cause of memory leaks in an application is due to unwanted references. The garbage collector finds the memory that is no longer in use by the program and releases it back to the operating system for further allocation.

To know what is an unwanted reference, first, we need to get an idea of how garbage collection determines that a piece of memory is unreachable. Garbage collection uses two main algorithms to find unwanted references and unreachable code, reference count and mark-and-sweep.

88. What about mark and sweep algorithm?

The mark-and-sweep algorithm follows two steps. First, in JavaScript, the root is the global object. The garbage collector periodically starts from the root and finds all objects that are referenced from the root. It will mark all the reachable objects **active**. Then, garbage collection frees the memory for all objects that are not marked as **active**, returning the memory to the operating system.

89. proto vs prototype in JS?

Proto and prototype both are objects that help in whether creating an array, object, or function and provide access to those specific methods or objects directly without taking memory and even it will give access to its constructor and all the array methods like push, pop, etc.

Proto: It is an actual object that provides a way inherit to inherit properties from JavaScript with the help of an object which is created with new. Every object with behavior associated has internal property

Syntax: Object.__proto__ = value

Prototype: It is a special object which means it holds shared attributes and behaviors of instances. It is a way to inherit properties from javascript as it is available in every function declaration.

Syntax : objectTypeName.prototype.SharedPropertyName=value;

90. What is microtask and macrotask? example?

A microtask is a tiny function that is run after the function or programme that produced it has exited and only if the JavaScript execution stack is empty, but before the user agent returns control to the event loop that controls the script's execution environment.

This event loop could be the browser's main event loop or the event loop that controls a web worker. This allows the specified function to run without the risk of interfering with the execution of another script, while also ensuring that the microtask runs before the user agent has a chance to react to the microtask's actions.

Examples: process.nextTick, Promises, queueMicrotask, MutationObserver

A **macrotask** is a tiny function that runs after the JavaScript execution stack and microtask have both been cleared.

A macro-task represents a separate and independent piece of task. The JavaScript code is always executed here in macrotask queue, and the **microtask queue** is always empty. The macrotask queue is frequently confused with the task queue or the event queue. The macrotask queue, on the other hand, works in the same way as the task queue. The task queue is used for synchronous statements, whereas the **macrotask queue** is utilised for asynchronous statements.

In a single macrotask execution cycle, all logged microtasks are processed in one fell swoop. The macrotask queue, on the other hand, has a lower priority. Parsing HTML, producing DOM, executing main thread JavaScript code, and other events like page loading, input, network events, timer events, and so on are all macrotasks.

Example: setTimeout, setImmediate, requestAnimationFrame, setInterval, requestAnimationFrame, etc

91. What about Map in ES6

Map is a collection of elements where each element is stored as a Key, value pair. Map object can hold both objects and primitive values as either key or value. When we iterate over the map object it returns the key, value pair in the same order as inserted.

Syntax: new Map([it])

Parameter:

it - It is any iterable object whose values are stored as key, value pair,

If the parameter is not specified then a new map is created is Empty

Returns: A new Map object

92. Promise.all() vs Promise.allSettled()

Promise.all will reject as soon as one of the Promises in the array rejects.

Promise.allSettled will never reject - it will resolve once all Promises in the array have either rejected or resolved.

Their resolve values are different as well. Promise.all will resolve to an array of each of the values that the Promises resolve to - eg [Promise.resolve(1), Promise.resolve(2)] will turn into [1, 2].

Promise.allSettled will instead give you [{ status : 'fulfilled', value: 1 }, { status : 'fulfilled', value: 2 }]

```
Promise.all([Promise.resolve(1), Promise.resolve(2)])
  .then(console.log);
Promise.allSettled([Promise.resolve(1), Promise.resolve(2)])
  .then(console.log);
```

93. What about Promise.race()

The Promise.race() method returns a promise that fulfills or rejects as soon as one of the promises in an iterable fulfills or rejects, with the value or reason from that promise. const promise1 = new Promise((resolve, reject) => {
 setTimeout(resolve, 500, 'one');
});

```
const promise2 = new Promise((resolve, reject) => {
  setTimeout(resolve, 100, 'two');
});
```

```
Promise.race([promise1, promise2]).then((value) => {
  console.log(value);
  // Both resolve, but promise2 is faster
});
// expected output: "two"
```

94. What about Shadow DOM

Shadow DOM allows hidden DOM trees to be attached to elements in the regular DOM tree — this shadow DOM tree starts with a shadow root, underneath which you can attach any element, in the same way as the normal DOM.

In short, the shadow DOM is a browser technology whose main objective is to provide encapsulation when creating elements. On the other hand, the virtual DOM is managed by JavaScript libraries—e.g., React—and it's mostly a strategy to optimize performance.

95. How to optimize site with gzip

Gzip is a fast and easy way to improve page speed performance while still delivering a high-quality experience to your users.

1. Browser send request to server
2. Server found file, zipping it and sending it over.
3. Browser unzip it and show the user.

But in old browser it doesn't support gzip and it only work if browser support it.

You can see if it is possible or not in request header.

Accept-Encoding: gzip, deflate, br

And you can check response file with Content-Encoding in response headers.

Content-Encoding: gzip

96. useEffect vs useLayoutEffect in React

The main difference between the useEffect hook and the useLayoutEffect hook is that the useEffect hook serves asynchronously, whereas the useLayoutEffect hook works synchronously.

In simple words, we can say that the difference between useEffect and useLayoutEffect is in the time at which these functions are invoked.

97. useCallback vs useMemo in React

The useCallback hook is used when you have a component in which the child is rerendering again and again without need. Pass an inline callback and an array of dependencies.

useCallback will return a memoized version of the callback that only changes if one of the dependencies has changed.

useMemo is a React Hook that lets you cache the result of a calculation between re-renders.

useCallback and useMemo both expect a function and an array of dependencies.

The difference is that useCallback returns its function when the dependencies change while useMemo calls its function and returns the result.

98. What do you know about webpack?

Webpack is a static module bundler used for JavaScript applications. Since webpack understands only JavaScript and JSON files, it transforms front-end assets such as HTML, CSS, and images into valid modules if the corresponding loaders are included. While Processing application webpack, internally builds a dependency graph that maps every module your project needs and produces one or more output bundles.

Some core concepts of webpack are:

- Entry
- Output
- Loaders
- Plugins
- Mode

<https://www.geeksforgeeks.org/introduction-to-webpack-module-bundler/>

99. Difference between unit & e2e testing?

Unit Testing is the type of software testing level in which each individual component of software are tested separately.

It's generally performed by a developer.

It can't be used for those systems which have a lot of interdependence between different modules.

It does not allow for parallel testing.

End-to-end testing is the type of software testing used to test entire software from starting to the end along with its integration with the external interfaces.

The main purpose of end-to-end testing is to identify the system dependencies and to make sure that the data integrity and communication with other systems, interfaces, and databases to exercise complete productions.

<https://www.geeksforgeeks.org/difference-between-end-to-end-testing-and-unit-testing/#:~:text=In%20end%2Dto%2Dend%20testing,the%20connected%20systems%20both%20combine.>

100. Which tools & system do you prefer for operating system and IDE?

AWS Cloud9.

Eclipse.

Zend Studio.

PhpStorm.

Arduino.

NetBeans.

RubyMine.

WebStorm.

101. What is Vue.\$nextTick() in Vue?

Vue allows to use `this.$nextTick(callback)` right on the component instance.

In the following example `handleClick()` method changes `this.show` data, and right away `this.$nextTick()` is setup to catch the moment when this update reaches DOM:

```
<template>
  <div>
    <button @click="handleClick">Insert/Remove</button>
    <div v-if="show" ref="content">I am an element</div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      show: true,
    };
  },
  methods: {
    handleClick() {
      this.show = !this.show;
      this.$nextTick(() => {
        console.log(this.show, this.$refs.content);
      });
    },
  },
};
</script>
```

`this.$nextTick()` is more convenient if want to access the updates of the current component instance.

<https://dmitripavlutin.com/vue-next-tick/>

102. What does mean undefined in JS?

undefined is a property of the global object.

It indicates that a variable has not been declared at all.

It is an ECMAScript1 (ES1) feature.

In all non-legacy browsers, undefined is a non-configurable, non-writable property. (Even when this is not the case, avoid overriding it.)

For example :

```
//demonstrating usage of undefined in javascript
var n;console.log(n);//undefined
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined
<https://www.geeksforgeeks.org/undefined-vs-null-in-javascript/>

103. Why does class component move to functional component in react?

That's because Functional component is

- Easier to test: You don't have to worry about hidden state and there aren't as many side effects when it comes to functional components, so for every input, the functions will have exactly one output.
- Easier to read/write: The syntax is less complex than that of class components, and it's easier to read due to knowing how much you can't do with functional components already. The use of prop destructuring makes it really beneficial to see what's going on and what's coming out of the component.
- Easier to debug: Again, functional components depend on the props they are given and don't rely on state. There isn't a reason to `console.log()` your state to constantly understand the changes that are going on.

<https://betterprogramming.pub/pros-cons-of-functional-components-in-react-f52bde98db0>

104. What reason is for rendering in react?

Render means renew only an appropriate part of information on user's screen when the element properties (props) are replaced by new ones or a component state (as set of props) changes in application. Thanks to the render method, we avoid reloading the whole web page, save time, and increase productivity.

Purpose of `render()`:

- React renders HTML to the web page by using a function called `render()`.
- The purpose of the function is to display the specified HTML code inside the specified HTML element.
- In the `render()` method, we can read props and state and return our JSX code to the root component of our app.
- In the `render()` method, we cannot change the state, and we cannot cause side effects(such as making an HTTP request to the webserver).

105. `useState` vs `useRef`

`useState` is a built in React hook that store information as states in a variable.

It lets adding React states to functional components.

It declares the state variable while the value is stored in the count variable.

```
//import useState from react
import React, { useState } from 'react';
function Count() {
  // Declare a new state variable called count
  const [count, setCount] = useState(0);
```

useRef is a built in React hook that takes one argument or parameter as its initial value and returns a reference or persisted mutable values.

Ref contains the value which can be retrieved using the current property.

We can also store user inputs in refs and display the collected data.

Different parts

- Data or values stored in a reference or ref remains the same, even after component re-rendering, unlike states. So, References do not affect component rendering but states do.

- useState returns 2 properties or an array. One is the value or state and the other is the function to update the state. In contrast, useRef returns only one value which is the actual data stored.

- When the reference value is changed, it is updated without the need to refresh or re-render. However in useState, the component must render again to update the state or its value.

<https://dev.to/salehmubashar/useref-or-usestate-which-is-better-258j>

106. Should I use index as key in React?

Using an index as key in React is probably a bad idea.

The reason is because you are not passing an unique key for each child element in the list. Without a unique key, React couldn't differentiate if the element was removed or just the content is changed. In particular, using indexes as keys in reactive lists is an anti-pattern and should be avoided. Using an array index is not recommended because it does not uniquely identify an element.

When an array is sorted or an element is added to the beginning of the array, the index changes, even if the element representing that index is the same. This causes unnecessary rendering.

107. What is SSR and how does it works?

Server-side rendering (SSR) is an application's ability to convert HTML files on the server into a fully rendered HTML page for the client.

Server-side sends a fully rendered page to the client; the client's JavaScript bundle takes over and allows the SPA framework to operate. Page load behaviour in Server Side Rendering :-

1. Browser makes a request to the server for HTML. Server makes API requests (usually co-located) and renders the content in the server.
2. Once the page is ready, the server sends it to the browser. The browser loads and parses the HTML and paints the content on the screen without waiting for the JavaScript bundle(s) to load.
3. Once the JavaScript bundle(s) are loaded, the browser hydrates interactivity to DOM elements, which is usually attaching event handlers and other interactive behaviours.

Some Pros of SSR are :

1. It enables pages to load faster which provides a better user experience. Since the server serves HTML with content, neither the browser nor search engines have to depend on

JavaScript for the content.

2. It plays an important role in SEO and correctly indexes webpages. (Google favours web pages with faster load time.)
3. It provides body to the HTML pages for all server ships and Content is not blocked by the time taken to load JavaScript bundle(s).
4. Page loads a lot faster than CSR.

108. What is Core Web Vital and how can you improve Core Web Vital with SSR?

Core Web Vitals is a set of metrics introduced by Google that aim at helping you measure the quality of user experience on a website.

Core Web Vitals are made up of three important measurements of a user's experience on a website. Loading time, page interactivity, and visual stability.

-Largest Contentful Paint (LCP): measures loading performance and reports the render time of the largest image or text block visible within the viewport.

-First Input Delay (FID): measures interactivity and reports the time from when a user first interacts with a page (e.g. clicks a button) to the time when the browser is able to begin processing event handlers related to this interaction. It reflects how fast a user can perform an action on the page and see a result.

-Cumulative Layout Shift (CLS): measures visual stability and is an aggregate of unexpected layout shifts. Having a good CLS value means a user won't experience any sudden movements of the content they're interacting with (text they're reading, button they're trying to click). Core Web Vitals are important for SEO, as they can help give your website more recognition and keep it organized and clean.

-How to improve with SSR?

Code-splitting (LCP, FID improvement)

Optimizing images (LCP enhancement)

<https://www.elpassion.com/blog/how-to-improve-core-web-vitals-with-nextjs>

109. What is purpose of mentoring?

Mentoring is a process through which an individual offers professional expertise as well as support to a less experienced colleague.

The purpose of mentoring is to grow by tapping into the knowledge and experience of someone further along than yourself. It's the best way to accelerate your development.

110. What is scrum poker?

Planning Poker, sometimes referred to as "Scrum Poker", bridges the gap between estimating the requirements of your project and the logistics of how everything is going to get done in time.

Scrum poker or planning poker is a consensus based, gamified technique to estimate the complexity and effort of a software feature. After presenting the feature, each member of the team picks a vote from different ranges of card sets. The vote remains hidden until all members have voted to avoid influence from other team members. After everyone has voted the highest and lowest estimates explain their choice and the process is repeated until the team agrees on a value.

<https://www.makeuseof.com/what-is-scrum-poker/>

33 Unit em vs rem in CSS

EM is relative to the parent element's font size, so if you wish to scale the element's size based on its parent's size, use EM.

REM is relative to the root (HTML) font size, so if you wish to scale the element's size based on the root size, no matter what the parent size is, use REM.

By default 1 rem, 1em = **16 pixels**.

111. Web worker vs service worker

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a **JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page.**

Unlike Web Workers, **Service Workers have a dedicated role of being a proxy between the network and the browser and/or cache.**

Similar to Web Workers, they are registered in the primary JavaScript file as a dedicated worker.

Web Workers are handy to run expensive scripts without causing the user interface to freeze, while Service Workers are useful to modify the response from network requests.

112. How do you handle the side-effect in Redux?

In the application, We need to handle side-effects(asynchronous) like API calls or setting timeout and intervals, because these are required when building features of a “complex web application.”

Redux has a lot of middleware libraries available in React such as Redux-Saga and Redux thunk, it's especially used in handling API calls and side effects.

We can handle simple side effects within action creators by leveraging redux-thunk.

We can use redux-thunk which allows us to dispatch functions/promises and write async logic in our action creators.

Redux-thunk is the most used redux middleware for handling side-effects, and it works for most use-cases, but it doesn't work in complex applications where you might want to cancel an API request after an action has been dispatched, or you want to debounce a function call.

This complex situation is where a middleware like `redux-saga` thrive, it helps you handle these easily using special function called effects provided by `redux-saga`.

`Redux-saga` is built on the concept of generators which was introduced as part of the ECMAScript 2015 module.

Generators are a powerful concept and its strengths were leveraged to create `Redux-saga`, which then gives users the ability to write asynchronous code in a synchronous manner, thereby eliminating problems with callbacks(callback hell), as well as helps to run side-effects in parallel, cancel them, etc.

Both of which are difficult or impossible using `redux-thunk`.

113. `const` vs `Object.freeze`

ES6 has brought several new features and methods into JavaScript since its release.

Amongst these new features are `Object.freeze()` method and `const`.

Sometimes people get confused between `Object.freeze()` method and `const` but the `Object.freeze()` and `const` are completely different.

`const` applies to bindings ("variables"). It creates an immutable binding, i.e. you cannot assign a new value to the binding.

`Object.freeze` works on values, and more specifically, object values. It makes an object immutable, i.e. you cannot change its properties.

114. What is main advantage of using React rather than Angular?

Because of its virtual DOM implementation and rendering optimizations, React outperforms Angular.

It's also simple to switch between React versions; unlike Angular, you don't have to install updates one by one.

Finally, using React, developers have access to a wide range of pre-built solutions.

115. How to handle too quiet quality code.

For the code to be easy enough to change, it needs to be **readable, testable, and reusable**. It also needs to be reliable, maintainable, and gracefully handle failures without any user effort.

1. Embrace coding conventions

Development teams usually create a list of guidelines known as coding conventions. They include the recommended styles and practices for a certain programming language.

Coding conventions dictate a wide range of things, such as:

- naming conventions
- spaces
- operators
- declarations
- comments
- indentation
- file organization
- architectural patterns

2. Use a code linter

A code linter is a tool that automatically checks your code to see if it meets your project's or company's coding convention.

If it doesn't match the convention, linters will show these errors as warnings.

3. Adopt continuous integration

Developers merge their changes to the main branch several times a day – this process is called continuous integration (CI). Each merge launches an automated code build and test sequence, which take a moment to run. But if there's an error, the code build doesn't run, and the CI system prevents it from advancing to the next step. The team then gets a report and can fix the error immediately.

4. Leave helpful comments

Developers sometimes make comments that aren't necessary to understand the code. This kind of clutter can result in developers taking extra time to read through the code.

Add comments only when you feel they have some merit and can provide value to your codebase.

116. Disadvantage of Redux

1. Lack of Encapsulation

There is no possibility of encapsulating any data in the Redux library. So, the chances of security issues grow as an application becomes heavier with new components. Since there is no data encapsulation, any component can access it.

2. Restricted Design

Most developers have complained about the rigidity of Redux. It is very restricted with minimal alternatives. So, you can use it for a trivial application or anything of that nature as state management will become more complex than it is without Redux.

3. Excessive Memory Use

In Redux, every state is immutable. In order to update a state, a reducer has to return a new state every time. This process creates compilation in the long run leading to excessive memory usage.

4. Increased Complexity

Even though the benefits are plenty, Redux has an additional complexity in layering when it comes to using manipulation logic actions or reducers.

5. Time-Consuming

For small or medium-level local state management, Redux can be highly time-consuming as it requires more inputs of the boilerplate code to begin writing.

117. What issues of using Cypress

<https://blog.autify.com/limitations-of-cypress>

1. Write Tests in JavaScript

You would be required to have some advanced JavaScript knowledge which includes Promises, jQuery, asynchronous JS, and other advanced JavaScript methodologies.

JavaScript Promise is an async programming model in JavaScript which can be complex for anyone without the basic knowledge of JS. It's been known to cause confusion for automated testing.

This limitation to JavaScript also brings about the compulsory installation of all the NPM packages in the package.json file as is known with all NPM frameworks. This manages the dependencies associated with the Cypress framework, and even though this is the behavior for applications, it makes no sense for automated testing. There can sometimes be errors when running a Mocha test if the correct dependencies are not installed.

The node modules folder is quite large too, which makes it a limitation to its speed when running automated tests.

2. No multi-tab support

Because Cypress runs inside the browser, you wouldn't be able to support multiple tabs when running tests.

3. *Cypress is slower

For a partly small test suite, it's been found that Cypress is about the same speed, or slightly slower than Selenium WebDriver. This is also noticeable on a medium or large test suite. It is inefficient and very difficult to maintain test automation with Cypress.

4. Maintenance

In the quality assurance and test automation industry, as with most other professional industries, the view of an inexperienced test automation engineer would be quite different from that of an expert. There are laid down rules and specifications when it involves industry-standard UI testing which an expert should know.

Tests happen rapidly as the application grows, and this is most often time large scale of automated UI tests. It has been found that Cypress does a bad job of keeping up with maintenance. It is quite easy to create tests than to maintain with Cypress. Industry-standard automation testing means keeping up with one's automated tests in real-time as the application changes which happens very frequently.

It's the ease to use that matters when it comes to automated testing, as it's not just a programmer-facing technology, but also open to the general public, and if it's so difficult to maintain tests than to create them, then there's an unbalance somewhere.

This makes it difficult to rely solely on Cypress for automated testing as much more effort is geared towards enhancing and stabilizing it than running the automated tests itself.

5. Complex and Inconsistent Syntax

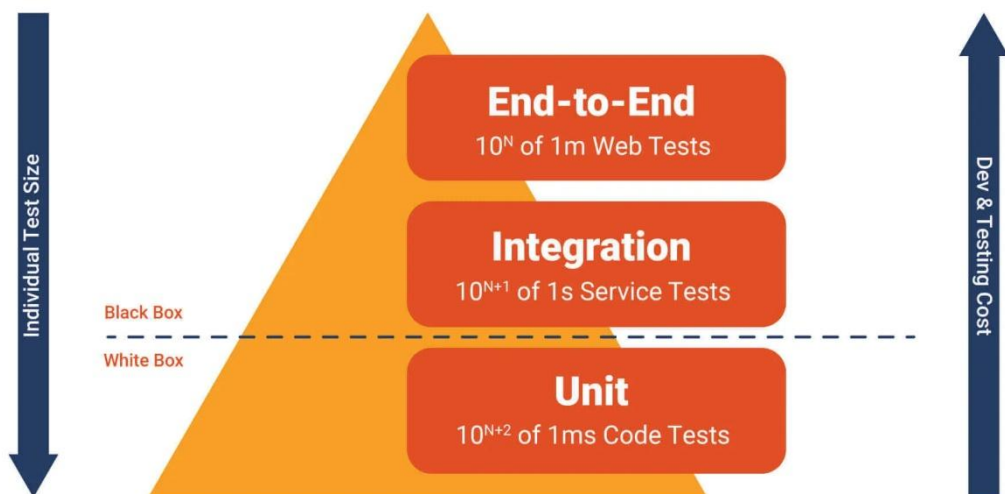
There is a level of syntactic ambiguity in Cypress which makes it difficult to use and understand. As compared with some other testing frameworks, the level of complexity experienced with Cypress is multiplied because of the sole reliance on JavaScript. Asynchronous programming can be a challenging process and when integrated with automated testing, there usually is a lot of trade-offs. Non-engineers or any stakeholder asides from the engineers in a software development life cycle (SDLC) cannot be a part of the testing process because of this fact.

118. What about Testing Pyramid

<https://www.headspin.io/blog/the-testing-pyramid-simplified-for-one-and-all>

Testing Pyramid is a framework that can facilitate both developers and QAs to build high-quality software.

fə'silə,tāt



Test automation pyramid functions at three levels:

Unit tests

Unit testing serves as the foundation of the test pyramid. Unit tests have a limited scope and ensure that isolated code units function as expected. Unit tests should assess a single variable and not rely on external dependencies.

The team has to execute pre- and post-commit tests in this pyramid layer, and tests happen to be dev-triggered.

Using virtual devices like simulators and emulators for virtual devices for these tests is highly recommended. It is sufficient to execute these tests on virtual devices as it provides the required feedback with log/debugging information. Moreover, testing with virtual devices is both faster and more cost-effective.

Integration tests

The integration test is the second step of the software testing pyramid. Unit tests check small portions of a codebase. The Dev team should perform integration tests to verify the interaction of this code with other codes in the application, which generally comprise the overall software. These tests check how a code interacts with external components. These components may include external services(APIs), databases, web services, and similar elements. Integration tests ensure that the software communicates efficiently and retrieves accurate data to perform as expected.

Integration tests are generally slower than unit tests due to the necessity of interaction with external sources. They also require a pre-production environment to run.

It is essential to have the right balance of virtual and real devices for running tests in integration testing.

Some testing types, such as acceptance testing, can be performed on virtual devices. Other types of testing, including security testing, performance testing, and certain accessibility features, cannot be performed on virtual devices; therefore, it is ideal to have a combination of actual and virtual devices at this stage. You will also need to add additional real OS/device settings that correspond with usage metrics and market trends.

End-to-end tests

End-to-end testing checks the enormous bulk of code (the entire application). It uses a test environment and data to simulate real-world functionality. It is the most expensive to maintain and has the slowest operating pace. As this test checks assembled applications, it is also the most difficult to identify issues.

The ideal method to approach end-to-end testing is from the end-users perspective. Testers might consider facts like different ways an end-user might engage with the app, what they would do to achieve their objectives with the app, and what might go wrong. The aim is to develop a test based on such actions.

End-to-end testing is the apex of the test automation pyramid as it can be the most time-consuming phase of the testing process. End-to-end testing is typically fragile and may have unreliable external dependencies like integration tests.

In addition, it is crucial to simulate actual user behavior during pyramid testing. You can determine how your app will perform in the real world by testing it under various settings, including app conflicts, call/text interruptions, network latency, and throttling.

119. Min heap vs Max heap

<https://www.educative.io/answers/min-heap-vs-max-heap>

<https://www.section.io/engineering-education/understanding-min-heap-vs-max-heap/>

<https://www.geeksforgeeks.org/difference-between-min-heap-and-max-heap/>

Min and Max heaps are complete binary trees with some unique properties.

Min-Heap

The root node has the minimum value.

The value of each node is equal to or greater than the value of its parent node.

Max-Heap

The root node has the maximum value.

The value of each node is equal to or less than the value of its parent node.

120. pick vs omit vs record in TypeScript

<https://hackernoon.com/utility-types-in-typescript-part-2-pick-omit-record>

Pick<T, Keys>

The utility type Pick creates a new type with only the specified Keys of the input type T.

```
interface Point {  
  x: number;  
  y: number;  
  z: number;  
}  
  
type PointXY = Pick<Point, "x" | "y">;  
// {  
//   x: number;  
//   y: number;  
// }
```



```
interface User {  
  name: string;  
  age: number;  
  email: string;  
  password: string;  
  address: string;  
  city: string;  
}
```

```
type UserAddress = Pick<User, "address" | "city">;
```

```
const address: UserAddress = {  
  address: "No 1 ABC Street",  
  city: "Port Harcourt",  
};
```

Omit<T, Keys>

The utility type Omit creates a new type by picking all properties from the input type T and then removing the specified Keys. It does the opposite of Pick.

```
interface Point {  
  x: number;  
  y: number;  
  z: number;  
}
```

```
type PointXY = Omit<Point, "z">;  
// {  
//   x: number;  
//   y: number;  
// }
```

```
interface User {  
  name: string;  
  age: number;
```

```

email: string;
password: string;
address: string;
city: string;
}

type UserProfile = Omit<User, "address" | "city" | "password">;

const profile: UserProfile = {
  name: "Anderson",
  age: 27,
  email: "test@test.com",
};

```

Record<K, T>

The utility type Record creates a new object type. The keys in this new object type are the values passed as K while the values are what is passed in as T.

```

type Axes = "x" | "y" | "z";

type Point = Record<Axes, number>;

// {
//   x: number;
//   y: number;
//   z: number;
// }

```

```

interface UserInfo {
  id: number;
  name: string;
  age: number;
}

const data: UserInfo[] = [
  {

```

```
    id: 1,  
    name: "Ande",  
    age: 28,  
  },  
  {  
    id: 2,  
    name: "Mosa",  
    age: 25,  
  },  
];  
  
const userList: Record<string, UserInfo> = {  
  "1": {  
    id: 1,  
    name: "Ande",  
    age: 28,  
  },  
  "2": {  
    id: 2,  
    name: "Mosa",  
    age: 25,  
  },  
};
```

1. Restful api vs GraphQL, pros of GraphQL

Rest vs GraphQL

- GraphQL only supports the POST method while REST support GET, POST, DELETE, PATCH, etc.

- Unlike REST, GraphQL returns result only in JSON representation
- In the case of data retrieval from the server: Unlike GraphQL, REST API can end up making requests to multiple endpoints, whereas, GraphQL, only has a single endpoint with which one has to access at the time data fetching on a server.

Advantage of GraphQL

- One endpoint: With traditional REST APIs, you have to create specific endpoints based on the data you want to request. This makes scaling your API difficult because soon, you might find yourself having to manage tens, maybe hundreds, of routes that you will have to remember.
- Fewer server requests: GraphQL allows you to make multiple queries and mutations with only one server request. This can be useful when your server only allows a limited number of requests a day.
- Declarative data fetching: Unlike a REST API, GraphQL only fetches what you actually need. All you have to do is specify what fields to return.
- Type system: GraphQL uses a type system to describe your data, which makes developing much easier. If you are a TypeScript fan, this is a win-win.
- Self-documenting: GraphQL is self-documenting, meaning that all of your queries and mutations will automatically be documented by GraphQL.

2. Restful API vs Restless API

POD	RESTful	RESTless
Meaning	An application that conforms to the REST architectural style that provides interoperability between computer systems on the internet.	An application that is not use on the principles of REST.
Example	REST	SOAP
Data Format Supported	Support various data formats such as HTML, JSON, XML, Text etc	Support XML format only
Expose Business Logic	Use URL to expose business logic.	Use the service interface to expose business logic.
Easier and Flexible	Yes	Not as easy
Security	Inherits security measures from the underlying transport protocols; therefore it is less secure	Defines its own security layer and is more secure
Bandwidth and resources	deploys multiple standards, so it takes fewer resources and bandwidth as compared to SOAP that uses XML for the creation of Payload and results in the large sized file.	more bandwidth as it needs to convert the data in XML which increases its payload and results in the large sized file.

3. nextjs pros cons

Pros

- Building super-fast static websites that behaves like dynamic
- Great for User Experience
- Great for SEO
- Fast time-to-market
- Great community support
- Flexibility in building UX and UI
- Many development advantages

Cons

- Requires development in order to make changes
- Poor plugin ecosystem
- No built-in state manager

- Considered as a more expensive option

4. SSR vs SSG, diff

The major difference between SSR and SSG is that in the latter's case, rather than during runtime, your HTML is generated during build time. Such websites are extremely fast since the HTML content is served even before you make a request. On the other hand, the website needs to be rebuilt and reloaded entirely every time a change is made. Consequently, SSG-based websites are far less interactive and native-like than those that rely on SSR. They are largely static sites with little to no dynamic content.

5. lifecycle methods in class component, not in functional component

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases. The three phases are: **Mounting**, **Updating**, and **Unmounting**.

Mounting

Mounting means putting elements into the DOM. React has four built-in methods that gets called, in this order, when mounting a component:

- `constructor()`
- `getDerivedStateFromProps()`
- `render()`
- `componentDidMount()`

Updating

The next phase in the lifecycle is when a component is *updated*. A component is updated whenever there is a change in the component's **state** or **props**. React has five built-in methods that gets called, in this order, when a component is updated:

- `getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`
- `componentDidUpdate()`

Unmounting

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it. React has only one built-in method that gets called when a component is unmounted:

- `componentWillUnmount()`

https://www.w3schools.com/react/react_class.asp

6. Difference between div and span?

Span and div are both generic HTML elements that group together related parts of a web page. However, they serve different functions. A div element is used for block-level organization and styling of page elements, whereas a span element is used for inline organization and styling.

div

The [div \(division\) element](#) is a generic block-level element, most often used to divide page content into blocks. A block element is a page element that starts a new line and has a width equal to the entire page or the parent container.

You'll very often see divs used to group related paragraphs, images, headings, and links.

span

The span element is a generic inline element, typically used to apply styling to a portion of inline content. An inline element does not start a new line and only takes up as much space on the page as its content. Span tags are used on small segments of text, links, images, and other HTML elements that appear inline with the surrounding content.

To summarize, a div tag creates a block-level element while a `` tag wraps around an inline element. Despite their differences, both are used to group like elements for structural and/or styling purposes.

<https://blog.hubspot.com/website/span-vs-div>

7. How many positioning values existing in CSS?

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the **position** property is set first. They also work differently depending on the position value.

https://www.w3schools.com/css/css_positioning.asp

8. Difference between Post vs Put method in rest API?

<https://byjus.com/gate/difference-between-put-and-post-http-request/>

S.No.	PUT	POST
1	The PUT method is considered idempotent. In this method, if we send a request several times, it will be counted as a single request modification.	The POST method is considered idempotent. In this method, if we send a request multiple times, then multiple URLs will get created on the servers.
2	The PUT method is used to modify a single resource.	The POST method is used to add a child resource.
3	It can be cached.	It cannot be cached.
4	This method acts as specific.	This method acts as an abstract.
5	In the PUT method, we use the UPDATE query.	In the POST method, we use a CREATE query.
6	When it comes to the PUT method, the client finalizes which URL resource should be updated.	When it comes to the POST method, the server finalizes which URL resource should be added.

9. What are keyframes and the purpose?

The @keyframes rule specifies the animation code.

The animation is created by gradually changing from one set of CSS styles to another.

During the animation, you can change the set of CSS styles many times.

Keyframes are, in essence, an animator's signposts.

They direct the animation software, telling it what images will be moving, how they will be moving, what speed they will be moving at, and when the movement will stop.

Animators use a keyframe to mark the start and end of an action.

10. How many ways of converting string to a number in Javascript?

using the Number() function.

using the parseInt() function.

using the parseFloat() function.

using the unary plus operator (+)

multiplying the string by the number 1.

dividing the string by the number 1.

subtracting the number 0 from the string.

11. How to delete property from object in Javascript?

The delete operator deletes both the value of the property and the property itself.

After deletion, the property cannot be used before it is added back again.

The delete operator is designed to be used on object properties.

It has no effect on variables or functions.

12. What is temporal dead zone in ES6

Temporal Dead Zone is the period of time during which the let and const declarations cannot be accessed.

Temporal Dead Zone starts when the code execution enters the block which contains the let or const declaration and continues until the declaration has executed.

Later when the let & const were introduced they have some restrictions for accessing just to help javascript developers write good code and make debugging easy and so the new concept of temporal dead zone came along, it helps us to catch errors because accessing some data before providing is wrong.

13. Difference between interface and type in typescript?

One major difference between type aliases vs interfaces are that interfaces are open and type aliases are closed. This means you can extend an interface by declaring it a second time.

In the other case a type cannot be changed outside of its declaration.

Type in TypeScript: The Type System in TypeScript portrays the different data types that are supported by the language. It is divided into three major sections that are Any Type, Built-In Type, and User-Defined Type.

Interface in TypeScript: An Interface in TypeScript is a syntactical obligation that all entities must follow. It can only contain the declaration of the members and is responsible for defining the properties, methods, and events.

Unlike an interface declaration, which always introduces a named object type, a type alias declaration can introduce a name for any kind of type, including primitive, union, and intersection types.

Interface types have many similarities to type aliases for object type literals, but since interface types offer more capabilities they are generally preferred to type aliases. For example, the interface type

```
interface Point {  
    x: number;  
    y: number;  
}
```

could be written as the type alias

```
type Point = {  
    x: number;
```

```
y: number;  
};
```

However, doing so means the following capabilities are lost:

- An interface can have multiple merged declarations, but a type alias for an object type literal cannot.

14. what is unknown, never type in typescript?

- Use never in positions where there will not or should not be a value.
- Use unknown where there will be a value, but it might have any type.

never is the most specific type because there is no set smaller than the empty set. unknown is the least specific type because it contains all possible values.

15. For building app with react, what library is needed?

<https://www.monocubed.com/blog/best-react-libraries/#:~:text=With%20React%2C%20there%20is%20one,Create%20React%20App'%20CLI%20library.>

- React CLIs
- Redux
- Mobx
- Material UI
- React Bootstrap
- Ant Design
- Framer Motion
- Formik
- React Hook Form
- React Router

16. Redux vs context

Redux is a complete state manager capable of allowing an app to undo/redo actions and provides advanced developer tooling for debugging. Context API is designed as a dependency injection mechanism that allows making data available through the component tree without being manually passed.

17. how can you implement SOLID principle in react component?

Single responsibility principle (SRP)

- Break large components that do too much into smaller components
- Extract code unrelated to the main component functionality into separate utility functions
- Encapsulate connected functionality into custom hooks

Open-closed principle (OCP)

OCP states that “software entities should be open for extension, but closed for modification”.

The right way to start a extensible component is to use composition from the beginning.

This way, the Input component would be closed for modifications and open for extension.

```
<InputContainer ...props>
  <InputLabel ...props />
  <InputLeftAddon ...props />
  <InputControl ...props />
  <InputRightAddon ...props />
</InputContainer>
```

Liskov substitution principle

LSP recommends designing objects in such a way that “subtype objects should be substitutable for supertype objects”.

A very basic example of a subtype/supertype relationship could be demonstrated with a component built with styled-components library

```
import styled from 'styled-components'
```

```
const Button = (props) => { /* ... */ }
```

```
const StyledButton = styled(Button)`
```

border: 1px solid black;

border-radius: 5px;

```
const App = () => {  
  return <StyledButton onClick={handleClick} />  
}
```

Interface segregation principle (ISP)

According to ISP, “clients should not depend upon interfaces that they don’t use.”

It’s not worth reusing the interface from your data mapper just because its easier. A component interface should only have the properties that are relevant for it.

This also helps for your IntelliSense. When you have a huge dropdown on your IDE and it can’t autocomplete properly, you are probably not segregating your interfaces.

Dependency inversion principle

One component shouldn’t directly depend on another component, but rather they both should depend on some common abstraction.

18. deep copy vs shallow copy in javascript.

A mechanism in which an object is bit-wise copied to a specified object is known as Shallow Copy.

- The shallow copy method pastes an exact copy of a particular object into another object.
- It is primarily utilized for copying One Dimensional array elements, where it only copies the elements present at the first level.
- It only copies the reference addresses to another object.

Object.assign() method

spread operator [...]

Assignment operator “=”

“Deep Copy” is a recursive procedure of copying object. This mechanism creates a new object and then clones the specified object’s key-value pair to it.

This statement signifies that, while Deep Copying, a JavaScript object is completely cloned into another object. After that, the copied object gets disconnected from the original object.

JSON.stringify()

JSON.parse(),

In deep copying value of the new variable is disconnected from the original variable while a shallow copied values are still connected to the original variable.

19. How implement deep copy in Javascript.

Answer is in 18.

20. reflow vs repaint

The Repaint occurs when changes are made to the appearance of the elements that change the visibility, but doesn't affect the layout

Eg: Visibility, background color, outline

Reflow means re-calculating the positions and geometries of elements in the document. The Reflow happens when changes are made to the elements, that affect the layout of the partial or whole page. The Reflow of the element will cause the subsequent reflow of all the child and ancestor elements in the DOM

21. how can you avoid reflow

- 1) Use Best-Practice Layout Techniques
- 2) Minimize the Number of CSS Rules
- 3) Minimize DOM depths
- 4) Update Classes Low in the DOM Tree
- 5) Remove Complex Animations From the Flow
- 6) Modify Hidden Elements
- 7) Update Elements in Batch
- 8) Limit the Affected Elements
- 9) Recognize that Smoothness Compromises Performance
- 10) Analyze Repaint Issues with Browser Tools

<https://www.sitepoint.com/10-ways-minimize-reflows-improve-performance/>

22. debance vs throttling

Debouncing and throttling are two techniques that limit when and how often a function is called.

debouncing a function means we wait a certain time, doing nothing, until we call the

function.

throttling a function means we wait a certain time, doing nothing, after we call the function.

23. Explain pipe and interceptor in angular

Pipes are simply a function that we can directly apply on any expression/value in a template to transform it into some other value.

1) Pure Pipes

A pure pipe is only called when Angular detects a change in the value or the parameters passed to a pipe.

2) Impure Pipes

An impure pipe is called for every change detection cycle no matter whether the value or parameter(s) changes. The angular interceptor is a medium connecting the backend and front-end applications. Whenever a request is made, the interceptors handle it in between. They can also identify the response by performing Rxjs operators. The interceptors do not initiate the handle method and handle the requests at their level.

24. What is cross browser?

Cross-browser compatibility can be defined as the ability of a website, application or script to support various web browsers identically. This issue is important as now-a-days numerous web browsers are used for web surfing and all of them have different standards.

25. How can you implement CSP(content security policy).

1 – Define your CSP

There are many different directives that can be used in a CSP, making it easy for you to customize your CSP to fit the needs of your website or application.

To get started, make a list of policies or directives and source values to define which resources your site will allow or restrict. We've provided a list of common CSP Directives and source values for you to mix and match.

2 – Add the CSP to your HTTP Response Header

Where do I set a CSP?

Most modifications to your HTTP Response header will happen in the configuration file. Follow these steps to identify your server and set your CSP for the appropriate environment.

Find out what server your website is running on. You can also log into your cPanel and check under the Server Information interface to find out. This will show the server that hosts your account.

Once you know your server, set your HTTP response header within the corresponding configuration file as detailed below. We have also linked additional resources for greater detail.

3 – Apply your Content Security Policy

CSPs are typically implemented using a special HTTP header that is sent with the response from the server. The header contains the CSP rules, which are then enforced by the browser.

We are going to outline two ways to configure your system to start enforcing a CSP: The first option is to add your CSP via Meta tags, which works on all browsers, but is less popular. The second option is to set your CSP using the HTTP Response Header.

26. what is prototypical inheritance

Prototype inheritance in javascript is the linking of prototypes of a parent object to a child object to share and utilize the properties of a parent class using a child class.

Prototypes are hidden objects that are used to share the properties and methods of a parent class to child classes.

Syntax

The syntax used for prototype inheritance has the `__proto__` property which is used to access the prototype of the child. The syntax to perform a prototype inheritance is as follows :

```
child.__proto__ = parent;
```

EX)

```
let company = {
```

```
  name: "A",
```

```
  pay: function () {
```

```
    console.log("Paying");
```

```
  },
```

```
}; //company object
```

```
let worker = {
```

```
  id: 1,
```

```
  work: function () {
```

```
    console.log("Working");
```



```
},  
}; //worker object  
worker.__proto__ = company; //worker object inherits company object  
console.log(worker);  
worker.pay(); // calling method from company object using worker object.
```

Output:

```
id: 1  
work: f ()  
[[Prototype]]: Object  
name: "A"  
pay: f ()  
[[Prototype]]: Object
```

Paying

27. prototype design pattern

The Prototype design pattern relies on the JavaScript prototypical inheritance. The prototype model is used mainly for creating objects in performance-intensive situations. The objects created are clones (shallow clones) of the original object that are passed around.

What is the prototype design pattern example?

The mitotic division of a cell — resulting in two identical cells — is an example of a prototype that plays an active role in copying itself and thus, demonstrates the Prototype pattern. When a cell splits, two cells of identical genotype result. In other words, the cell clones itself.

Form Submit Question

What do you think about documentation in projects? How do you approach documentation in projects?

Documenting projects is important because it ensures that you have the materials to review your work throughout the project

It saves time for you from struggling with taking the required actions and making decisions.

Define the purpose and scope of the project;

Identify results and key points;

Document the technical parameters and the technologies to be used;

Address the way deliverable will be built or distributed;

Evaluate elements such as quality, scope, resources, risks, training and costs;

How to do documentation for a project?

Best Practices for Documenting Your Project

Include a README file that contains. ...

Allow issue tracker for others.

Write an API documentation. ...

Document your code.

Apply coding conventions, such as file organization, comments, naming conventions, programming practices, etc.

Include information for contributors.

Should we write tests on the frontend? Why?

Advantages of unit testing are that it reduces or prevents production bugs, increases developer productivity, encourages modular programming.

- It helps you write better code.
- It helps you catch bugs earlier.
- It helps you detect regression bugs.
- It makes your code easy to refactor.
- It makes you more efficient at writing code.

The main purpose of end-to-end testing is to identify the system dependencies and to make sure that the data integrity and communication with other systems, interfaces, and databases to exercise complete productions.

Do you measure performance of your applications? How would you do it?

Application performance indicates how the app is functioning and how responsive the app is to the end-user.

Metrics & Measure

Perceived load speed: how quickly a page can load and render all of its visual elements to the screen.

Load & Runtime responsiveness: how quickly a page can load and execute any required JavaScript code in order for components to respond quickly to user interaction

Visual stability: do elements on the page shift in ways that users don't expect and potentially interfere with their interactions?

Smoothness: do transitions and animations render at a consistent frame rate and flow fluidly from one state to the next?

First contentful paint (FCP): measures the time from when the page starts loading to when any part of the page's content is rendered on the screen.

Largest contentful paint (LCP): measures the time from when the page starts loading to when the largest text block or image element is rendered on the screen.

First input delay (FID): measures the time from when a user first interacts with your site (i.e. when they click a link, tap a button, or use a custom, JavaScript-powered control) to the time when the browser is actually able to respond to that interaction.

Time to Interactive (TTI): measures the time from when the page starts loading to when it's visually rendered, its initial scripts (if any) have loaded, and it's capable of reliably responding to user input quickly.

Total blocking time (TBT): measures the total amount of time between FCP and TTI where the main thread was blocked for long enough to prevent input responsiveness.

Cumulative layout shift (CLS): measures the cumulative score of all unexpected layout shifts that occur between when the page starts loading and when its lifecycle state changes to hidden.

How do you make the decision about adding a new dependency to your project?

How do you determine project dependencies?

In order to identify project dependencies, you must first create a map of project tasks. Next, look for tasks that the team cannot perform until they receive information or deliverables from a previous task. Those tasks are dependent. Think of your whole project as a series of workflows or a flowchart.

6 Things to Consider when Adding a New Project Dependency

Is this library worth it?

Is this part of your project's core domain?

What's your exit strategy?

What does it do, and why?

What's its outlook?

What are the second-order effects?

What are the solutions for handling state management in the application, which one is your favourite?

Redux is one of the most trending libraries for front-end development in today's marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior

How do you share the state between multiple components?

We can share the state using context, redux, mobx, atomic state management.

What Browser API / Web API have you used recently?

Browser API

-Fetch API

-Canvas API

-Fullscreen API

-Clipboard API

-Payment Request API

-Geolocation API

Web API

BOM API

DOM API

Client Storage

FormData API

Drag-and-Drop API

Notification API

History API

Serverless hosting

AWS Lambda.

Microsoft Azure Functions.

Google Cloud Functions.

Cloudflare Workers.

Netlify Functions.

Vercel Functions.

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by AWS.

As an AWS cloud engineer with 4+ years practical experience, I have used Lambda service on the various purposes of my previous projects.

Mostly, I built serverless API endpoint for my web & mobile apps using Lambda (Node.js, Python or .NET Core) with API Gateway.

I am very familiar with its core concepts, advanced usages including concurrency management, throttling, lambda layer, efficient use of memory/cpu resource, diagnosing metrics, etc.

Also, extensively used it in connection with other services like SQS/SNS and Kinesis for data streaming.

Angular vs Angular.js

One vital distinction between Angular vs AngularJS is AngularJS is JavaScript-based while Angular is TypeScript based.

We can see the difference in their Components. Angular JS, based on JavaScript, uses terms of scope and controllers while Angular uses a hierarchy of components. Angular is component-based while AngularJS uses directives.

While both Angular and AngularJS use Directives, how they use them makes the difference. Angular has standard Directives while AngularJS has a pack of them. One has to be cautious while using the Directives. For instance, when you want to create a two-way binding in AngularJS, you will have to use the ng-model. For one-way binding, you will use ng-bind.

In Angular, you will find only the ngModel. You can use the symbols for one-way binding as: "[," and two-way binding as: "[()]" "[()]" is for property binding, while "[()]" is for event binding.

Angular JS

This framework has a model-view-controller (MVC) that acts as the central component as it manages data, logic, rules, and expresses how the applications behave.

Angular

Angular uses components that are directives with templates. There are two kinds of directives in Angular.

Structural directives: they alter the DOM's layout by replacing its elements.

Attributive directives: it changes how the DOM behave and the appearance of the element.

Mobile support

AngularJS does not provide mobile support while Angular supports mobile.

Performance and Speed

Angular JS; The two-way binding feature ultimately reduces the efforts and time of construction.

Angular; The framework upgraded to a better structure, which provides improved performance and speed.

Tool support

Angular JS; The framework relies on third-party tools such as IDE and WebStorm.

Angular; It uses the Command Line Interface (CLI) to reduce the time when creating applications.

Role of RxJS

RxJS provides an implementation of the Observable type, which is needed until the type becomes part of the language and until browsers support it. The library also provides utility functions for creating and working with observables.

What is the purpose of RxJS?

The full form of RxJS is Reactive Extension for Javascript. It is a javascript library that uses observables to work with reactive programming that deals with asynchronous data calls, callbacks and event-based programs. RxJS can be used with other Javascript libraries and frameworks.

What is benefit of RxJS?

RxJS provides a graceful API that makes it easy to describe a complex stream of data compact and precise. RxJS comes with a comprehensive kit of standard entities such as Subjects, Observables, and operators which undertakes all the hard work and makes it possible.

Where does Angular use RxJS?

Using RxJS to reduce the state of our components. Everything in Angular revolves around the state of the components (and the app in general) and how it gets projected to the UI. And there are lots of times when we can use streams to represent volatile pieces of our data inside the view.

Promise vs Observable in RxJS

The Promise is always asynchronous, while the Observable can be either asynchronous or synchronous, the Promise can provide a single value, whereas the Observable is a stream of values (from 0 to multiple values), you can apply RxJS operators to the Observable to get a new tailored stream.

Promise

A Promise handles a single event when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

Observable

An Observable is like a Stream (in many languages) and allows to pass zero or more events where the callback is called for each event.

Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case.

Observable also has the advantage over Promise to be cancellable. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore.

While a Promise starts immediately, an Observable only starts if you subscribe to it. This is why Observables are called lazy.

Observable provides operators like `map`, `forEach`, `reduce`, ... similar to an array

There are also powerful operators like `retry()`, or `replay()`, ... that are often quite handy. A list of operators shipped with `rxjs`.

Lazy execution allows to build up a chain of operators before the observable is executed by subscribing, to do a more declarative kind of programming.

Two-way data binding in Angular

The two-way data binding in Angular enables data to flow from the component to the view and the other way round. It is used to display information to the end-user and allows them to make changes to the underlying data using the UI. This makes a two-way connection between the view (the template) and the component class that we already mentioned.

How does data binding work in Angular?

Angular data binding works by synchronizing the data in the components with the UI. This way it can always show the current value of the data. In terms of the two-way binding, the automatic data synchronization happens between the Model and the View, keeping both synced all the time. Because of this, any change made in the Model is immediately reflected in the View as well. And vice versa – changes made in the View are updated in the Model too. The two-way data binding in Angular is used to display information to the end user and allows the end user to make changes to the underlying data using the UI. This makes a two-way connection between the view (the template) and the component class.

Dependency Injection in Angular

Dependency injection, or DI, is one of the fundamental concepts in Angular. DI is wired into the Angular framework and allows classes with Angular decorators, such as Components, Directives, Pipes, and Injectables, to configure dependencies that they need.

How to use dependency injection in AngularJS?

To inject a value into AngularJS controller function, add a parameter with the same when the value is defined.

1. `var myModule = angular.module("myModule", []);`
2. `myModule.value("numberValue", 100);`
3. `myModule.controller("MyController", function($scope, numberValue) {`

4. `console.log(numberValue);`
5. `});`

Authentication vs Authorization

We can use Auth0 and JWT

It is the process of recognizing the user's identity. Authentication is a very important process in the system with respect to security. Authorization is the process of giving permission to the user to access certain resource in the system.

What is the difference between authentication and authorization?

Authentication and authorization are two vital information security processes that administrators use to protect systems and information. Authentication verifies the identity of a user or service, and authorization determines their access rights.

How to detect bugs in Angular?

Name folders and files to demonstrate their uses, without a second guess.

Use a consistent naming approach throughout your Angular app.

Avoid Using Modules Incorrectly

What is universal Angular?

Angular Universal **executes on the server, generating static application pages that later get bootstrapped on the client**. This means that the application generally renders more quickly, giving users a chance to view the application layout before it becomes fully interactive.

The Angular Universal project is a community-driven project to expand on the core APIs from Angular (platform-server) to enable developers to do server-side rendering of Angular applications in a variety of scenarios.

Angular Unit Testing

Angular Unit testing is the process of testing small and isolated pieces of code in your Angular application.

This provides an added advantage to the users in the sense that they can add any new features without breaking any other part of their application.

Popular Angular unit testing options are **Jasmine, Karma, and Jest**.

Angular vs React

Angular is a full-fledged framework, while React is a JavaScript library.

Therefore, React has to be paired with a framework to build fast, beautiful, and compatible UI. Here are some of the React-based frameworks used by developers: Material UI.

Why choose Angular over React?

React is a library, but Angular is a full-fledged framework. The virtual DOM and one-way data binding are used by React.js, but the real DOM and two-way data binding are used by Angular. There's also a speed difference (React's is faster) and a difference in bundle size (React's is smaller) (React works a bit faster)

Datastructure and algorithm

-data structure
tree,

Suffix tree,

Splay tree.

Segment tree

Fenwick tree

Etc

-algorithm

DP

Greedy

Binary Search

Maximum network flow

Minimum cost max flow

KMP

Z-index

DPS

BPS

Greedy algorithm

A greedy algorithm is an algorithm used to find an optimal solution for the given problem. greedy algorithm works by finding locally optimal solutions (optimal solution for a part of the problem) of each part so show the Global optimal solution could be found.

heuristic algorithm

In mathematical programming, a heuristic algorithm is a procedure that determines near-optimal solutions to an optimization problem. However, this is achieved by trading optimality, completeness, accuracy, or precision for speed.

The heuristics at each stage would be to choose a locally optimal solution that will lead to a globally best fit solution. A greedy algorithm doesn't guarantee to produce an optimal solution all the time, but it can give us a globally approximate solution of a given problem in efficient time.

Saturday Test

2-18

1. Regular functions vs Arrow function about this keyword

In Arrow function this indicates one step out side object.

In regular function they are different where its invoked.

2. What is Liskov substitution principle

This principle states that objects of a superclass should be replaceable with objects of its subclasses without breaking the application.

3. Explain about event+loop

There are two concepts "Call-Stack" and "Task-Queue".

There are normal tasks and micro and macro tasks.

Normal tasks go into call stack and then micro and macro tasks go into callback queue.

At first call stack tasks will be execute and once call stack empty callback queue tasks will be execute.

In micro tasks there are `async await` and `promise` and in macro task there are `setTimeout` and `setInterval`.

Micro tasks will run first and the macro tasks.

`setTimeout(). setInterval()`

4. What is useEffect hook? explain about parameters

`useEffect` is one of React hooks that use to render the components when the component was mounted and updated.

The component will be rerender when at least one of any dependencis changes.

It works as `componentDidMount()`, `componentDidUpdate`, `componentWillUnmounted()` of life cycle methods in react class components.

5. useCallback vs useMemo

useCallback and useMemo use Memorize technical in React.

They are used to prevent expensive of the component rerendering many times.

useCallback returns the memorized function but useMemo returns the memorized value of the function.

6. Explain about hoisting

Hoisting is a new JavaScript technique which means that new variable and functions are seems to declare at the top of script.

So we can use them anywhere in the script.

7. event bubbling vs propagation

Event bubbling and propagation are used all events order in the nested components.

Event propagation is used to find all events from parent to child elements and Event bubbling is to decide the orders of these events, it works from child to parent elements.

There are two methods like event bubbling and event capturing.

8. What is HOC? example

HOC stands for higher-order-components that it aims to make the reusable components.

Ex] React.memo(), React.connect()

9. mark-and-sweep algorithm

mark and sweep algorithm is used to find the memory leaks in the whole code base.

First it makes mark used variables and functions to find from the root in the whole codebase.

Then it releases variables and functions which unmarked.

10. What is generic type in typescript

generic type is just like type template.

```

function genericType<Type>(arg: Type): Type {
    return arg;
}

```

1. Implement Person class that has 'firstname' and 'lastname' as properties and 'show_name' method that show full name.

ex) p1 = Person("YG", "S")

p1.show_name() =====> output: 'Full name is YG S.'

```

class Person:
    def __init__(self, f_name, l_name):
        self.firstname = f_name;
        self.lastname = l_name;
    def show_name(self):
        return self.firstname + ' ' + self.lastname;

```

```

p1 = Person("YG", "S");
print(p1.show_name());

```

2. Implement Student class inherited above Person class. This class has more properties named 'age' and 'school_name'. And represent this class as string.

ex) s1 = Student("YG:", "S", 23, "PYM#1")

print(s1) =====> output: 'YG S is 23 years old and studying in PYM#1'

```

class Person:
    def __init__(self, f_name, l_name):
        self.firstname = f_name;
        self.lastname = l_name;

```



```
def show_name(self):  
    return self.firstname + ' ' + self.lastname;
```

class Student:

```
def __init__(self, f_name, l_name, age, school):  
    self.firstname = f_name;  
    self.lastname = l_name;  
    self.age = age;  
    self.school = school;
```

3. List vs tuple vs Set vs dict

List is a collection which is ordered and changeable. Allows duplicate members.

Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.

Dictionary is a collection which is ordered and changeable. No duplicate members.

4. Implement reverse function using Generator.

```
def reverse(tmp):  
    for index in range(len(tmp)-1, -1, -1):  
        yield tmp[index]
```

```
str = "python";  
result = ""  
    for char in reverse(str):  
        result = result + char;  
  
print(result);
```

2-25

1. What is an Immediately Invoked Function in Javascript.

An IIFE (Immediately Invoked Function Expression) can be used for avoiding the variable hoisting from within the blocks.

It allows the public access to methods while retaining the privacy of variables defined in the function.

2. What is Single Responsibility principle explain with example

- Break large components that do too much into smaller components
- Extract code unrelated to the main component functionality into separate utility functions
- Encapsulate connected functionality into custom hooks

3. Explain call(), apply(), bind() methods

call() invokes the function and allows you to pass in arguments one by one using commas.

apply() invokes the function and allows you to pass in arguments as an array.

bind() returns a new function, allowing you to pass in this array and any number of arguments.

4. difference between `exec()` and `test()`

`exec()` will return specified match of given string or null if string is not exist.

`test()` return a boolean.

5. What is recursion

recursion is invoked itself inside itself function.

6. what is object destructuring

Object destructuring is an expression that access the data from objects like arrays, objects, maps and assign it to new variables.

7. What is temporal dead zone

A temporal dead zone (TDZ) is the area of a block where a variable is inaccessible until the moment the computer completely initializes it with a value.

A block is a pair of braces `{...}` used to group multiple statements.

Initialization occurs when you assign an initial value to a variable.

Temporal Dead Zone is the period of time during which the `let` and `const` declarations cannot be accessed.

Temporal Dead Zone starts when the code execution enters the block which contains the `let` or `const` declaration and continues until the declaration has executed.

8. explain about this keyword in Regular function vs Arrow function

In Arrow function this keyword indicates one step out side object.

In regular function they are different where its invoked.

9. What is mark-and-sweep algorithm

10. what is the output of the following code?

```
const b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

```
for (let i = 0; i < 10; i++) {  
  setTimeout(() => console.log(b[i]), 1000);  
}
```

```
// 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

```
for (var i = 0; i < 10; i++) {  
  setTimeout(() => console.log(b[i]), 1000);  
}
```

```
undefined repeat(10)
```

1. how can you check wether one real number is qual to another one, or not?

2. what is pip and how can you use it? (explain with example)

pip is used to install, upgrade, remove packages in Python.

exam]

```
$ python -m pip list
```

3. what is bisect package used for?

bisect module is used to manipulate sorted lists.

ex]

```
import bisect
```

```
num = [1, 3, 7, 11, 18, 23]
```

```
bisect.insort(num, 9)
```

```
print(num)
```

4. how can you choose 10 random numbers in range of 0 - 99?

```
import random
```

```
random.sample(range(100),10)
```

5. how can you get today date?

```
from datetime import date
```

```
now = date.today()
```

```
print(now)
```

```
["1", "1", "1"].map(parseInt)
```

3-4

1. Projects vs. apps in Django project.

In simple words Project is the entire Django application and an app is a module inside the project that deals with one specific use case.

For eg, payment system(app) in the eCommerce app(Project).

2. About parameters of path() in Django framework.

The path() function is passed four arguments, two required: route and view, and two optional: kwargs, and name.

path() argument: route

path() argument: view

path() argument: kwargs

path() argument: name

3. Create a project named "mysite" and create a poll named "polls" and view a "Hello world!" when you request via "http://localhost:8000/polls/"

1) - \$ django-admin startproject mysite

2) - \$ python manage.py runserver

3) - \$ python manage.py startapp polls

4) - //We put the following python code in polls/view.py

```
from django.http import HttpResponse
```

```
def index(request):
```

```
    return HttpResponse("Hello, world!")
```

5) - // We create urls.py in the polls directory and write code like this

```
from django.contrib import admin
```

```
from django.urls import include, path
```

```
urlpatterns = [
```

```
    path('polls/', include('polls.urls')),
```

```
    path('admin/', admin.site.urls),
```

```
]
```

6) - \$ python manage.py runserver

4. Regarding to above question, how to show html template?

```
templates > home.html
```

```
<h3>Hello world!</h3>
```

```
// update urls.py
```

```
from django.urls import path
```

```
from .views import view
```

```
urlpatterns = [  
    path("", home_view ),  
]
```

```
//We update view.py
```

```
from django.shortcuts import render
```

```
def home_view(request):  
    return render(request, "home.html")
```

5. how can you choose 10 random numbers in range of 0 - 99? (python)

```
import random
```

```
random.sample(range(100),10)
```

6. how can you get today date? (python)

```
from datetime import date
```

```
now = date.today()
```



```
print(now)
```

7. how can you check whether one real number is equal to another one, or not? (python)

```
def compareFloatNumbers(a, b):  
    if (abs (a - b) < 1e-9):  
        print("The numbers are equal")  
    else:  
        print("The numbers are not equal")  
  
a = (0.3 * 3) + 0.1  
b = 1  
compareFloatNumbers(a, b)
```

8. List vs tuple vs Set vs dict (python)

List is a collection which is ordered and changeable. Allows duplicate members.

Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.

Dictionary is a collection which is ordered and changeable. No duplicate members.

9. How to get all keys or values of Dictionary using Loop (python)

```
d = {'key1': 1, 'key2': 2, 'key3': 3}
```

```
for k in d.keys():
```

```
    print(k)
```

```
for v in d.values():
```

```
    print(v)
```

```
for k, v in d.items():
```

```
    print(k, v)
```

10. Loop over two or more sequences at the same time. (python)

```
questions = ['name', 'quest', 'favorite color']
```

```
answers = ['lancelot', 'the holy grail', 'blue', 'asdfasdf']
```

```
tmp = [1,2]
```

```
for q, a, v in zip(questions, answers, tmp):
```

```
    print('What is your {0}? It is {1} = {2}'.format(q, a, v))
```

1. Explain call(), apply(), bind() methods

call() invokes the function and allows you to pass in arguments one by one using commas.

apply() invokes the function and allows you to pass in arguments as an array.

bind() returns a new function, allowing you to pass in this array and any number of arguments.

2. What is temporal dead zone

A temporal dead zone (TDZ) is the area of a block where a variable is inaccessible until the moment the computer completely initializes it with a value.

A block is a pair of braces ({...}) used to group multiple statements.

Initialization occurs when you assign an initial value to a variable.

3. What is mark-and-sweep algorithm

mark and sweep algorithm is used to find the memory leaks.

First it makes mark used variables and functions from the root in the whole codebase.

Then it releases variables and functions which unmarked.

4. What is Liskov substitution principle

This principle states that objects of a superclass should be replaceable with objects of its subclasses without breaking the application.

5. Regarding to below react code, is it good practice? Why? How can you avoid it?

```
....  
{  
    students.map(item => <div>Name:{item.name} Age: {item.age}</div>)  
}
```

It is not best practice.

We have to add unique id value for the each student object.

The best practice is just use the specific key value to make possible for create, update, delete features.

```
{  
    students.map(item => <div key={item.id}>Name:{item.name} Age: {item.age}</div>  
}
```

3-10

1. Create a project named "mysite" and create a poll named "polls" and view a "Hello world!" when you request via "http://localhost:8000/polls/"

- \$ django-admin startproject mysite
- \$ python manage.py runserver
- \$ python manage.py startapp polls
- write code in polls/view.py

```
from django.http import HttpResponse
```

```
def index(request):
```

```
    return HttpResponse("Hello, world!")
```

- Create urls.py

```
from django.contrib import admin
```

```
from django.urls import include, path
```

```
urlpatterns = [  
    path('polls/', include('polls.urls')),  
    path('admin/', admin.site.urls),  
]
```

- \$ python manage.py runserver

2. what is generic views and advantage of it?

generic views are built-in classes in Django.

Django's generic views take certain common idioms and patterns found in view development and abstract them.

So we can write common views of data without having to write too much code.

3. what module do you use for auto test?

we can use unittest module in python

also we can use like django.test.utils

exam] from django.test import TestCase

4. what is good rules to arrange the tests?

1. All TestCase subclasses are run first.

2. Then, all other Django-based tests are run with no particular ordering guaranteed nor enforced among them.

5. how can you style the html template in django?

we can import global css and customize css files in base.html file

exam]

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"/>
//customize style
```

```
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

1. const vs Object.freeze

const applies to bindings ("variables"). It creates an immutable binding and we can't assign a new value to the binding.

Object.freeze works on values, and more specifically, object values. It makes an object immutable and we can't change its properties.

2. what is web worker

A web worker is a JavaScript that runs in the background, independently of other scripts that may also have been executed from the same HTML page, without affecting the performance of the page.

3. Pick vs Omit vs Record in Typescript

Pick<T, Keys>

The utility type Pick creates a new type with only the specified Keys of the input type T.

Omit<T, Keys>

The utility type Omit creates a new type by picking all properties from the input type T and then removing the specified Keys. It does the opposite of Pick.

Record<K, T>

The utility type Record creates a new object type. The keys in this new object type are the values passed as K while the values are what is passed in as T.

4. em vs rem in CSS

em is a css unit relative to the font size of the parent element, while rem is a css unit relative to the font size of an html element.

5. promise.all vs promise.allsettled vs promise.race

Promise.allSettled returns a promise that is fulfilled with an array of objects that represents the outcome of each input promise. The Promise.race method is a way to wait for the first of multiple promises to complete.

6. why is it better to use fragment instead of div in react

with react fragment, we can render multiple elements without adding extra div tags.

7. state vs props in React

state variables are mutable, while props are immutable.

8. explain HOC

HOC are used to add sophisticated functionalities to existing components.

The purpose of the HOC technique, which is common in third-party React libraries, is to reuse component logic.

9. about this keyword in regular vs arrow function

in arrow function, this keyword indicates one step outside of itself, while in regular function this keyword indicates itself.

10. Get all subsequence indexes whose sum of all elements is equal to targetSum.

For example:

array = [1, 1, 2, 3] targetSum = 5;

answer = [0, 1, 4] of [1, 1, 3], [3, 4] of [2, 3]

```
function findIndexsArrayEqualsTargetSum(num, sum) {
```

```
  let result = [];
```

```
  let len = num.length;
```

```
  for(let i = (1 << len) - 1 ; i >= 0; i --) {
```

```
    let arr = [];
```

```
    let s = 0;
```

```
    for(let j = 0; j < len; j ++){
```

```
      if(i & (1<<j)) {
```

```
        s += num[j];
```

```
        arr.push(j);
```

```
      }
```

```
    }
```

```
    if(s === sum) {
```

```
      result.push(arr);
```

```
    }
```

```
  }
```

```
  return result;
```

```
}
```

```
let array = [1,1,2,3];
```



```
let targetSum = 5;
```

```
console.log(findIndexsArrayEqualsTargetSum(array, targetSum));
```

3-18

1. what is model in Django?

A model is a python class that represents a database table.

It defines the fields and behaviour of the data that is stored in the database.

A model refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

Example]

```
from django.db import models
```

```
class SampleModel(models.Model):  
    field1 = models.CharField(max_length = 50)  
    field2 = models.IntegerField()
```

```
class Meta:  
    db_table = "sample_model"
```

The metaclass set things like available permissions, singular and plural versions of the name, associated database table name, whether the model is abstract or not, etc.

2. what is view in Django?

A view function, or "view" for short, is simply a python function that takes a web request and returns a web response.

This response can be HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image, etc.

Exam]

```
from django.http import HttpResponse
```

```
def sample(request):  
    return HttpResponse("Hello World")
```

There are 2 types of views:

- 1 Function-Based Views: we import our view as a function.
- 2 Class-based Views: It's an object-oriented approach.

3. what is template in Django?

Templates are an integral part of the Django MVT architecture.

They generally comprise HTML, CSS, and js in which dynamic variables and information are embedded with the help of views.

Some constructs are recognized and interpreted by the template engine.

A template is rendered with a context.

Rendering just replaces variables with their values, present in the context, and processes tags.

The syntax of the Django template language includes the following four constructs.

- 1) Variables
- 2) Tags
- 3) Filters
- 4) Comments

4. how to update model after table is already created?

1) open models.py file.

2) add fields like joined_date to this model.

3) make a migration to tell Django that it has to update the database:

Exam] py manage.py makemigrations members

4) run the migrate command

Exam] py manage.py migrate

5. what is SQLite?

An SQLite database is a file-based database system that is stored on disk and managed by the SQLite library.

It is a relational database that stores data in tables, and allows developers to query, insert, update, and delete data using SQL statements.