=================================================  
Module 6: 50 Interview Question on Web Back-End Development  
=================================================  
#1. What is object-oriented programming?  
  
Object-oriented programming (OOP) is a programming language structure where the data and its associated processing, also known as methods, are defined as self-contained entities. These entities are called "objects." OOP involves modeling a system as a collection of various objects, with each object representing some specific aspect of the system.  
  
OOP languages are the norm today and provide developers with a formal set of rules and concepts for creating and managing objects. OOP has four fundamental concepts: inheritance, polymorphism, encapsulation, and data abstraction.  
  
#2. What is the difference between Wrapper Classes and Primitive Data Types?  
  
Wrapper classes offer a simple way to use various primitive data types, such as int and boolean, as objects. In other words, the wrapper class converts a primitive type into an object. On the other hand, a primitive type is a predefined data type that the Java programming language provides.  
  
#3. Define and explain these nine server response error codes: 200, 201, 204, 301, 400, 401, 404, 409 and 500.  
  
“200 means “OK” and everything went well. A 201 “Created”' message means the system created a resource at the client's request. A 204 “No Content” code means the server didn't send back a status. A 301 “Moved Permanently” message means a client-triggered action changed the resource URI. A 400 “Bad Request” error refers to a client-side error.   
  
If the client doesn't provide the correct authentication, you see a 402 “Unauthorized” code. A 404 “Not Found” return means it didn’t find a mapped resource. An inconsistent or impossible state returns code 409, “Conflict.” Server-side errors generate code 500, described as, ‘“Internal Server Error."  
  
#4. What is scope in JavaScript?   
  
In JavaScript every function has its own scope. It is basically a collection of rules for how variables are accessed and variables itself.   
  
#5. What is Content Security Policy?   
  
Content security Policy also known as CSP, is a header in HTML which lets the soite operators gain the whole control over the resources which are loading on the site.   
  
#6. What is Cross-Site Scripting (XSS)?  
  
XSS, Cross-Site Scripting is an attack which takes place when any attacker uses a web application to send any malicious code, in the form of browser side script, to another user.   
  
#7. What is Strict Mode?  
  
A new feature of ECMAScript 5 is Strict mode which lets you place a function, or a program in a “strict” context of operating.   
  
#8. What does SOLID stand for?  
  
S.O.L.I.D is an acronym of object-oriented design principles   
  
S- single responsibility principle   
  
O- open-closed principle   
  
L- Liskov Substitution principle   
  
I- interface segregation principle   
  
#8. What is callback hell?  
  
JavaScript which uses callbacks and is hard to get the right intuitively. The callbacks which are stacked in the form of pyramid structure.   
  
#9. What is Polymorphism ?  
  
The word Polymorphism means having many different forms. If we talk in object-oriented form, polymorphism refers to one interface, multiple functions.   
  
#10. What is a web server?  
  
A web server is a computer that stores and delivers web pages. When you type a URL into your browser, the browser contacts the web server and requests the page. The web server then sends the page back to the browser, which displays it on your screen. Apache and NGINX are some of the most popular web servers used by backend applications.  
  
Web servers can also host other resources, like images or videos.  
  
#11. What is the difference between a GET and a POST request?  
  
A GET request retrieves data from a server, whereas a POST sends data to a server. With a GET request, parameters get passed in the URL. With a POST request, parameters get passed in the request's body.   
  
#12. When should you use asynchronous programming?  
  
Asynchronous programming is often used when there is a need to improve the performance of an application. For example, if an application needs to make many database queries, it may be beneficial to use asynchronous programming to avoid blocking the main thread.  
  
#13. What is the difference between promises and callbacks?  
  
A promise is an object that represents the result of an asynchronous operation. A callback is a function that is invoked when an asynchronous operation completes.  
  
14# When should you use asynchronous programming?  
  
Asynchronous programming is often used when there is a need to improve the performance of an application. For example, if an application needs to make many database queries, it may be beneficial to use asynchronous programming to avoid blocking the main thread.  
  
#15. What are the tradeoffs of client-side rendering vs. server-side rendering?  
  
There are a few tradeoffs when deciding between client-side rendering and server-side rendering.  
  
Client-side rendering can be more complex to set up because the application needs to be able to run in a browser. Meaning the code must be transpiled from a higher-level language (such as JavaScript) to a lower-level language (such as HTML). In addition, client-side rendering can be slower because the application needs to download all of the necessary resources before it can start rendering.  
  
On the other hand, server-side rendering is typically easier to set up because developers can use any language capable of generating HTML. In addition, server-side rendering can be faster because the HTML can be generated on the server and then sent to the client.  
  
Please note that these tradeoffs often depend on the site's complexity and function.  
  
#16. What are high-order functions? Why are they useful?  
  
High-order functions are functions that take other functions as arguments. These functions help abstract common patterns of code. For example, a high-order function could create a function that logs the arguments it is called with. This would be useful for debugging purposes.  
  
  
#17. What is a microservice?  
A microservice is a small, independent component of a more extensive application. Each microservice has its own functionality and can be deployed independently.  
  
Microservices often build into large, complex applications that are easy to maintain and scale. One of the benefits of using microservices is that they can be written in different programming languages and deployed on different servers.  
  
Common examples of microservices include user authentication, payment processing, and image manipulation.  
  
API Questions  
18. How would you design an API?  
  
When designing an API, it's helpful to consider the needs of the developers who use the API. The API should be easy to use and well-documented. It's also critical to consider the API's security and ensure that only authorized users can access the data. Additionally, the API should be able to handle a large number of requests without overloading the server.   
  
#19. What is the difference between a RESTful and a SOAP API?  
  
RESTful APIs are designed to be easy to use and well-documented. They use a standard set of rules, which makes them easy to learn and use. On the other hand, SOAP APIs are designed to be more secure and can handle a larger number of requests. However, they are more complex to learn and use.   
  
#20. How do you handle errors when making API calls?  
  
When making API calls, it's industry standard to handle errors in a way consistent with the rest of the application. For example, if the API returns a 404 error, you might want to display a message to the user saying the data could not be found.  
  
Database Question:   
=================  
#21. How would you handle optimizing an existing database?  
  
Once you've selected a database, it's vital to keep it optimized. This process can be done by periodically running maintenance tasks, such as indexing the data or purging old data that is no longer needed. Additionally, monitoring the database's performance and ensuring it can handle the application's load is crucial.  
  
#22. How would you query data from a MongoDB database?  
  
MongoDB uses a query language called MongoDB Query Language (MQL). You would use the find() method to query data from a MongoDB database. This method takes a set of parameters that specify the criteria for the query. For example, to find all documents in the "users" collection that have a "firstName" of "John," you would use the following query:  
  
db.users.find({"firstName": "John"})   
  
#23. What are some benefits of using a NoSQL database?  
  
NoSQL databases have a few advantages over relational databases. They are generally more scalable and easier to manage. Additionally, they can be more flexible because they do not require a schema. However, NoSQL databases can be more difficult to query, and they often do not provide the same level of data consistency as relational databases.  
  
Security Questions  
==================  
#29. What are some common security risks when building a web application?  
  
Some common security risks when building a web application include SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).  
  
SQL injection is a type of attack where malicious code is injected into an SQL statement, resulting in the execution of unintended actions. XSS attacks occur when malicious code is injected into a web page, resulting in the execution of unintentional actions. CSRF attacks happen when a malicious user tricks a victim into submitting a request that performs an unwanted activity, such as changing their password or transferring funds.  
  
#30. How would you implement authentication and authorization on a new project?  
  
There are many ways to implement authentication and authorization on a new project. One way would be to use an existing third-party service, such as Auth0 or Okta. Another way would be to roll out your solution using JSON Web Tokens (JWTs) or similar technology.  
  
Regardless, you need to create a login page where users can enter their credentials. Once the user's credentials were verified, you would generate a JWT and send it back to the user. The user would then need to send the JWT with each authentication request.  
  
You would also need to implement an authorization system to check if the user has the correct permissions to access a particular resource. One way to do this would be to create roles and assign users to those roles. Then, you would use that to check the user's permissions when handling each request.  
  
#31. What is the difference between a cookie and a session?  
  
A cookie is a small piece of data stored on the user's browser. A session is a server-side data structure that holds information about the user's current session.  
  
Cookies store information such as the user's ID, language preference, or any other preferences. Sessions store information from a series of requests, such as the user's shopping cart or other information that needs to persist across multiple requests.  
  
Scalability Questions  
====================  
#27. How would you design a software system for scalability?  
  
When designing a software system for scalability, it's essential to consider the application's needs. For example, if the application needs to handle many concurrent users, you might want to design the system using a microservices architecture. This architecture allows each component of the system to be scaled independently. Additionally, you might want to use a message queue to decouple the components of the system. This will allow each component to scale independently without affecting the performance of the other components.   
  
#28. What are some common scalability issues? How can they be addressed?  
  
Some common scalability issues include performance degradation, data loss, and downtime. You can address these issues using various techniques, such as caching, sharding, and replication. Additionally, it is crucial to have a well-designed architecture that can handle increased loads.  
  
Scalability is an essential consideration for any application. While a few common scalability issues can occur, you can solve them with proper infrastructure. By understanding these issues and how to solve them, you can ensure that your application will be able to scale as needed.  
  
Testing Questions  
==================  
#32. How would you unit test a new feature?  
  
When testing a new feature, you would first need to write a test covering the new feature's functionality. You should isolate these tests from any other code in the system. Once you write the test, you need to run it and verify that it passes.  
  
If the test fails, you need to debug the code and find the cause of the failure. Once you find the cause, you need to fix the code and re-run the test. If the test passes, you can then commit the code and move on to testing other features.  
  
#33. How would you integrate tests into your workflow?  
  
When integrating tests into your workflow, you need to create a testing environment that mirrors the production environment as closely as possible. You need to set up this environment with all the necessary dependencies and data.  
  
Once you have set up the testing environment, you need to write tests covering the system's functionality. These tests should be run automatically whenever new code gets pushed to the repository. If any of the tests fail, the build should be marked as failed, and the developers notified.  
  
You should also manually run tests before each release. This will ensure that all of the functionality is working as expected and that there are no regressions.  
  
#34. What are some performance testing steps?  
  
1. Identify the critical areas of the application that need testing.  
2. Create test cases that exercise those areas of the application.  
3. Run the test cases and collect data on the application's performance.  
4. Analyze the data and identify any areas of improvement.  
  
#35. Why are TDD tests written before code?  
  
TDD (Test-Driven Development) is a development methodology in which you write before code. The idea behind TDD is that by writing tests first, developers can ensure that their code meets the requirements. In addition, TDD can help to find bugs early and prevent them from being introduced into the code. However, TDD can be time-consuming and requires a good understanding of testing principles.  
  
#36. How would you manage Web Services API versioning?  
  
Versioning is a critical part of API design, as it gives developers the ability to improve their API without stopping the clientʼs applications whenever new updates are rolled out. The three types of API versioning are:  
  
URL Versioning or Route Versioning: This solution uses URI routing to point to a specific version of the API.  
Versioning using a custom header: REST APIs are versioned by providing custom headers with the version number included as an attribute.  
Query String Parameter: Considered to be the worst method, the version number is included as a query parameter.  
  
#37. When would you apply asynchronous communication between two systems?  
  
In asynchronous communications, the client sends a request to the server (typically requiring lengthy processing), while receiving a delivery acknowledgment immediately.  
After the client receives the acknowledgment, it carries on with other tasks and will be notified eventually when the server finishes processing the request. The main benefit of asynchronous communications is improved performance.  
Asynchronous communications can be applied in situations where the response is not required immediately, and the current process can continue without the response. Real-world examples can include email, Slack, and other messaging platforms.  
  
#38. Explain the architectural style used to create the web API.   
  
The architectural style for creating web API are:  
  
HTTP for client-server communication  
XML / JSON as formatting language  
Simple URI as the address for the services  
Stateless communication  
  
#39. What does REST stand for and what is a RESTful API?   
REST states for REpresentational State Transfer and is a software architectural style that is meant to guide the design and development of web services.  
A RESTful API is an architectural style for an application program interface that utilizes HTTP requests to access and process data. The API can be used for getting, putting, posting, or deleting various data types.  
  
#40. Explain the essence of DRY and DIE principles?   
Do not Repeat Yourself (DRY) or Duplication is Evil (DIE) is a principle of software development that states that every piece of information must have only one, unambiguous and authoritative representation within a system. The principle is aimed at reducing the repetition of software patterns and replacing it with abstractions in order to avoid redundancy.  
  
#41. If we gave you a design and development challenge, how would you approach it?  
  
This question is a great way to test your problem-solving skills and how you would apply them in the workplace. When answering this question, it can be helpful to describe the steps you would take to solve the challenge and what resources you would use to complete the task.  
  
Example: “If I were given a design and development challenge, I would first look at the requirements of the project and create a plan for how I would approach the work. I would then break down the project into smaller tasks that I could complete one by one. For example, if I was tasked with creating a website, I would start by researching different frameworks and languages that are compatible with the client’s needs. Then, I would begin coding the website using those frameworks and languages.”  
  
#42. What is the difference between Express.js and Node.js?  
Node.js is an open-source, cross-platform run-time environment used for executing JavaScript code outside of a browser. Node.js is not a framework or a programming language; it is a platform that acts as a web server. Many big companies such as Paypal, Uber, Netflix, Wallmart, etc., are using this. On the other hand, Express is a small framework based on the functionality of Node.js.  
  
#43. What ways are provided by Express JS to configure the properties?  
Express JS provides us with two ways for configuring the properties, which are given below:  
  
With process.ENV  
With require.JS.  
  
#44. When does a Cross-Origin resource get failed in Express JS?  
A cross-Origin can fail in the following scenarios-  
  
If it's to a different domain  
If it's to a different port  
If it's to a different subdomain  
If it's to a different protocol.  
  
#45. What do you mean by the sanitizing input process?  
People can always enter weird things via the client-side code. They use the tools to POST things directly to our endpoints. For this, the Express provides various sanitizing methods to prevent these happenings.  
  
#46. Consider the following sanitizing methods:  
  
Trim() will trim the characters at the beginning as well as the ending of a string.  
Escape() will replace ‘, “, <, >, &, / with the corresponding HTML entities.  
NormalizeEmail() will canonicalize an email address.   
Blacklist() will remove the characters appearing on the blacklist.  
  
#47. Tell us about Router-level Middleware and Built-in Middleware.  
Router-level Middleware - We use the router-level Middleware for binding with a particular instance of Express.Router().  
  
Built-in Middleware - The version 4.x of Express introduced the built-in Middleware. The dependency on connecting gets removed by use of this Middleware.  
  
#48. Mention the arguments that are available in an Express JS route handler function.  
The arguments that are available in the route handler function of Express JS are given below:  
Res - It is the response object.  
Req - It is the request object  
Next (optional) - This argument is used for passing the management to any of the above-given route handlers.