JS.5

Interview Questions (Elliot)

▼ Can you name two programming paradigms important for JavaScript app developers?

JavaScript is a multi-paradigm language, supporting imperative/procedural programming along with OOP (Object-Oriented Programming) and functional programming.

JavaScript supports OOP with prototypal inheritance.

▼ What is functional programming?

Functional programming produces programs by composing mathematical functions and avoids shared state & mutable data.

▼ What are some functional languages?

Lisp and Haskell

▼ What does functional programming encompass?

the use of pure functions, avoiding side effects, and simple function composition

▼ How are side effects avoided?

through the use of deterministic algorithms or pure functions which given a particular input will always produce the same output

▼ What features support functional programming?

first-class functions, higher order functions, functions as arguments/values

▼ What is classical inheritance?

instances inherit from classes and create sub-class relationships: hierarchical class taxonomies

Classes create tight coupling or hierarchies/taxonomies

▼ What is prototypal inheritance?

instances inherit directly from other objects

every object has a link to a prototype for delegation

an example of this is concatenative inheritance which allows features inherit directly from one object to another by copying the source objects properties

I prefer prototypal inheritance & composition over class inheritance because it's not brittle

▼ What are object-oriented programming pros?

It’s easy to understand the basic concept of objects and easy to interpret the meaning of method calls.

▼ What are object-oriented programming cons?

OOP Typically depends on shared state.

Objects and behaviors may be accessed with non-deterministic order (unpredictable outcome), which may lead to undesirable behavior such as race conditions

a highly OOP codebase can be extremely resistant to change and very brittle

▼ What is a deterministic algorithm?

an algorithm that, given a particular input, will always produce the same output

▼ What are functional programming pros?

avoiding any shared state or side-effects, which eliminates bugs caused by multiple functions competing for the same resources

functions tend to be radically simplified and easily recomposed for more generally reusable code

also tends to favor declarative styles which concentrates on what to do not how to do something

▼ What are functional programming cons?

FP has a much steeper learning curve than OOP and the language of FP tends to be much more academic and formal

▼ When is classical inheritance an appropriate choice?

Never.

▼ When is prototypal inheritance an appropriate choice?

Any time you need inheritance.

When you need to compose objects from multiple sources through Object.assign() for instance.

▼ Prototypal inheritance creates what kind of relationship?

it creates has-a or uses-a or can-do relationships as opposed to the is-a relationship created with class inheritance

▼ What does “favor object composition over class inheritance” mean?

It means that code reuse should be achieved by assembling smaller units of functionality into new objects instead of inheriting from classes and creating object taxonomies.

It means avoiding:

- the brittle base class problem
- tight coupling
- the gorilla banana problem (“what you wanted was a banana, what you got was a gorilla holding the banana, and the entire jungle”)

▼ What are two-way data binding and one-way data flow, and how are they different?

Two way data binding means that UI fields are bound to model data dynamically such that when a UI field changes, the model data changes with it and vice-versa.

One way data flow means that the model is the single source of truth. Only the model has the access to change the app's state.

React uses one way data flow.

Angular uses two way binding.

▼ What is a monolithic architecture?

an app that is written as one cohesive unit of code whose components are designed to work together, sharing the same memory space and resources

▼ What is a microservice architecture?

it means that your app is made up of lots of smaller, independent applications capable of running in their own memory space and scaling independently from each other across potentially many separate machines

▼ What are monolithic pros?

most apps typically have a large number of cross-cutting concerns, such as logging, rate limiting, and security features such audit trails and DOS protection.

When everything is running through the same app, it's easy to hook up components to those cross-cutting concerns.

There can also be performance advantages, since shared-memory access is faster than inter-process communication (IPC).

▼ What are monolithic cons?

app services tend to get tightly coupled and entangled as the application evolves, making it difficult to isolate services for purposes such as independent scaling or code maintainability

harder to understand, because there may be dependencies and side-effects

▼ What are microservice pros?

Microservice architectures are typically better organized, since each microservice has a very specific job, and is not concerned with the jobs of other components.

Decoupled services are also easier to recompose and reconfigure to serve the purposes of different apps

▼ What are microservice cons?

In a microservice architecture, you'll either need to incur the overhead of separate modules for each cross-cutting concern, or encapsulate cross-cutting concerns in another service layer that all traffic gets routed through.

▼ What is synchronous programming?

Synchronous programming means that, barring conditionals and function calls, code is executed sequentially from top-to-bottom, blocking on long-running tasks such as network requests and disk I/O.

▼ What is asynchronous programming?

Asynchronous programming means that the engine runs in an event loop.

When a blocking operation is needed, the request is started, and the code keeps running without blocking for the result.

When the response is ready, an interrupt is fired, which causes an event handler to be run, where the control flow continues. In this way, a single program thread can handle many concurrent operations.

▼ Why is asynchronous programming important in JavaScript?

This is important in JavaScript, because it is a very natural fit for user interface code, and very beneficial to performance on the server.