First of all, thank you so much for giving me this opportunity. I would like to describe myself as a person who is versatile, determined and someone who loves learning.  I have 12 years of experience in Web development and for the last 2 years I’ have been working in project called My State Street which is custodian bank in US. The project executed in Agile fixed bid model. We use React for View and Java as backend primarily. We also use Redux for state management of the application. I’m managing a team where we have 15 UI developers in the team. I worked closely with clients and I build some good relationships. The project has been pretty successful and I like to think that I have contributed to that success. That’s pretty much it.

**Closure** : a simple function defined in another function which has access of outer function’s variables and parameters. We can use this to enforce the private functions. Javascript doesn’t have this in build.

**preventDefault**: Cancels the event if it is cancelable, without stopping further propagation of the event.

**stopPropagation**: Prevents further propagation of the current event. stops the move upwards, but on the current element all other handlers will run.

**stopImmediatePropagation:**Prevents other listeners of the same event from being called. After it no other handlers execute.

**ev.target** -  identifies the element on which the event occurred.

**ev.currentTarget** - Identifies the current target for the event, as the event traverses the DOM

**Chaining in js - event chaining :** Method chaining is a technique that can be used to simplify code in scenarios that involve calling multiple functions on the same object consecutively.

**Promise** : Is an object that may give a value sometime in the future; either a resolved or reason for not resolved. *resolve ()* – on resolve, *reject ()* – on reject

**Self-Invoking Functions**: is a function which called immediately right after it’s defined. It also called immediately invoked function(IIFE). You can use it if you want to execute sth while page loads

**Prototypical Inheritance**: Every object has property called prototype where we will add property and methods to it. When you create other object from this object, this newly created object would inherit the property from parent but bot by including by own property instead it uses from parent.

**Arrow Function**: No binding required.

**Let Vs Const** : Both are used for defining the variables. Let=> can reassign const => can’t reassign

|  |  |
| --- | --- |
| **Var** | **Let** |
| Function scope | Block scope |
| Gets hoisted | Doesn’t get hoisted |
| From beginning | Introduced in ES6 |

|  |  |
| --- | --- |
| **Const** | **Let** |
| Cant reassign. But object’s property can be changes | Can re assign |

|  |  |
| --- | --- |
| **Function Declaration** | **Function Expression** |
| If we call it before declare it will give you result | If we call it before declare it will throw a error |
| We can’t pass this as parameter | Since it stored in a variable, it behaves like a variable. So that we can pass this as parameter |

**Hoisted** - is a JavaScript default behavior to move declaration on the top. a variable can be accessed before declarations.

**Strict Mode** - using strict mode can help you to avoid simple mistakes like accidental variable, variable without declarations

**Call** - expects parameter individually - Ex : var myfun = myFunction.call(myfun, 0, 1);

**Apply** - expects a parameter as array - Ex: var myfun = myFunction.apply(myfun,$arr);

**Bind** – function can be invoked later.

**Serialize and Unserialize in js** - json.stringify() - seriolize, json.parse() – unserialize.

**Redux** :

Redux is a library which controls the state of your JavaScript app. It helps to manage and organize data better

**3 Principles**: Redux can be described in three principles:

1.     Single source of truth – single store object tree

2.     State cannot be modified - only way to change the state is to emit an action

3.     Changes made by pure functions – pure reducers

**Action**:  are the payloads which sends data to the store from your app.

**Reducer**: that update the state according to those actions.

**Store**: is the object that brings them together. The store has following responsibilities.

-     Holds application state

-     Allows access to state via getState()

-     Allows access to update state via dispatch ()

-     Registers listeners via subscribe(listener)

**Middleware**: It requires when you make async call, this can be used to delay the dispatch an action, or to dispatch only if certain condition is met. The inner function receives the store method dispatch() and getState() as parameter. People use it for logging, crash reporting and routing also.

**Agile Methodology:**agile methodology isa practice that promotes continues iteration of the development and testing throughout the software development lifecycle of the project.

**It has 3 Roles:**

**Scrum Master:**responsible forsetting up the team, sprint meeting and removes obstacles to progress

**Product Owner**: creates product backlog, prioritizes the backlog and is responsible for delivery of the functionality at each iteration.

**Scrum team**: Team manages its own work and organizes the work to complete the sprint

**Process/ Activities/ Ceremonies**/ **Events:**

1.     Planning meeting

2.     Execution of Sprint Tasks

3.     Daily Standup

4.     Review Meeting(Demo)

5.     Retrospective Meeting

6.     Product backlog refinement or grooming

**Definition of Done**: 1. All dev, QA tasks are marked completed, 2. All bugs are fixed, 3. The code review and unit testing is completed. 4. Demo is completed

**Definition of Ready**: It’s a working agreement between team and product owner on what readiness means. It’s an input criteria to plan a story in a sprint.

**Challenge** :

I was the one who has done all the cross-cutting concerns in this project. React 15.x doesn’t have error handling in built. So, I have implemented a HOC where I have done that by wrapping the Try-Catch method with all the lifecycle method of each components. So that any error occurs while component rendering, we can capture those errors and send it for logging. While I implement it, I faced many challenges, now its pretty successful and its up and running in production.