what is difference between in ODATA and API
OData:

- A specific type of API protocol designed for querying and updating data.
- It standardizes how to access and manipulate data over the web.

Odata as gernate link as Hana server and SAP
API:

- A general term for any interface that allows software applications to communicate with each other.
- It can expose any kind of functionality or data and can follow various protocols (e.g., REST, SOAP, GraphQL).

Step 1 to deploy our app to SAP System

1.Vs Code à
2. System SAP à
3.ABAP repository Name (BSP à Business Service Page) Description
à
4.TR(Transport Request),Package
5.one
Sematic Object+ Action
Lunchpad designer
Sap System à Fiori Launchpad

Step 2 stage

/n/ui2/fg

You are using transaction code /UI2/FLP to open SAP Fiori Launchpad and transaction code /UI2/FLPD_CUST or /UI2/FLPD_CONF to open Launchpad Designer.

Fiori launchpad is the central point to access the Fiori Apps
Its consists of tiles and groups
Groups consists of groups
Catalogs holds a tiles
Tiles get a reference of our Application

UI 5 Build
Npm run deploy

Fiori Launchpad Design Catalog + Group = Tiles and target Mappings

Standalone App
1.index-Loads UI 5 framework
2. component.js
3. Manifest.js
4.View
5.controller
Embbeded System : - both front end and back end System in Single System

Central hub System : both front end and back end system are separated (connection (system alias))

Type of Binding

· Aggregation Binding: This allows binding of UI elements to collections of data, such as tables or lists, enabling dynamic rendering of lists or tables based on model data.

Main focus on rending in cild object base on Array data and Array model clid element in running base on number of element side in model

Example : table , list , comboBox , select..

· Expression Binding: This allows developers to define more complex bindings using expressions, which can combine values, literals, and functions to produce dynamic content in UI elements.

Express binding is inline data binding is binding base on two value true and false condition express binding only used single line

Example:Base On Requirement Condition In any element

Property Binding: Property binding in SAP UI5 is a method of linking the properties of user interface (UI) elements, such as text or visibility, directly to properties within a data model. This connection ensures that any changes made to the model's properties automatically update the corresponding UI elements, allowing for dynamic and synchronized displays in applications.

Example : Text , title , Date ….

Propety binding direct a property in element

Element Binding: Element binding allows binding a UI control directly to a single entry in a collection, enabling detailed views or forms.

Element binging bind the call the one ya single object

- Syntax: {modelName>/collection/0} (binds to the first item in the collection)

Example : VBox Simpleform , HBox …

Formatters: Use when you need reusable formatting logic across multiple UI elements or when the formatting rules are complex and need to be centralized.
Expression Binding: Use when you need to apply dynamic conditions or calculations directly within the XML view or when the logic is specific to the UI presentation and doesn't require reuse across different parts of the application.

OneWay : - It can give data

TwoWay : Give Data and Take the Data

OneTime : - it will one time

· One-Way Binding:

- Updates flow from the model to the UI element.
- Used for displaying data that should reflect changes in the model.

· Two-Way Binding:

- Updates flow bidirectionally between the model and the UI element.
- Used for interactive UI elements where changes should update both the model and the UI element.

· One-Time Binding:

In SAP UI5, "one-time binding" refers to a concept where data is bound from a model to a UI element only once during the initial rendering. Unlike regular bindings where changes in the model are automatically reflected in the UI, one-time binding does not update the UI if the model changes after the initial binding. This approach is useful for displaying static or immutable data that doesn't need to be dynamically updated based on changes in the model.

Manifest

In SAP UI5, the "manifest" refers to a configuration file named manifest.json that provides structured metadata and settings for an application. This file is central to SAP UI5 applications as it defines various aspects such as:

- **Application Resources**: Specifies the resources (like libraries, CSS, and JavaScript files) that the application depends on.
- **Application Metadata**: Includes information about the application itself, such as its title, description, and version.
- **Routing Configuration**: Defines navigation routes within the application, specifying which views or components should be displayed for different URLs or hash fragments.
- **Data Sources**: Describes the models and data services used within the application, including connections to backend systems or OData services.
- **Dependencies and Libraries**: Lists required UI5 libraries and versions necessary for the application to function correctly.

sap.app: Contains metadata about the application, such as its ID, type, title, and description.

- sap.ui5: Contains configuration settings related to the SAPUI5 framework, including the root view, dependencies, models, and routing.
- sap.ui5.models: Defines the models used in the application, such as the ResourceModel for internationalization (i18n).
- sap.ui5.routing: Defines the routing configuration, including the router class, views, routes, and targets.

Rounting

One View to another view
Step : Routes
Step 2 : target - view Details
Step 3 : call route name
onGoToDetailsPress: function() {

```
            this.getOwnerComponent().getRouter().navTo("details");
        }
onGoBackPress: function() {
    // Get the router associated with the owning component
    var oRouter = this.getOwnerComponent().getRouter();

    // Use the back() method to navigate back to the previous page
    oRouter.back();
}
onGoToDetailsPress: function() {
    var oRouter = this.getOwnerComponent().getRouter();

    // Navigate to the "details" route with the parameter {id: 123}
    oRouter.navTo("details", { id: 123 });
}
var oRouter = sap.ui.core.UIComponent.getRouterFor(this);
oRouter.attachRouteMatched(this.onRouteMatched, this);
```

Component

Component or Component.js is the first point of our application or we can say that it serves as index which encapsulates all our applications details i.e., view names, routing details, main view, applications type (Full Screen or Split App), application service configuration etc. An independent and reusable part that can be used to structure SAPUI5 applications and its code easier to understand and to maintain.
There are 2 types of components:
1. Faceless components: Mainly without UIs, Used for business logic and helper methods e.g. Formatters.
2. UI Component: Typically component.js file which is made by extending sap.ui.core.UI Component class.

Bootstrapping

SAPUI5 Bootstrapping means loading and initializing SAPUI5 in any HTML page. Themes and Libraries are defined in this.The most important library or resource loaded in SAPUI5 bootstrap is “sap-ui- core.js”. Apart from this theme for the application, SAPUI5 libraries etc are declared in the bootstrapping.

Fragments

Fragments are very light weight UI controls. Fragments are not completely like views but they act like a view. Fragments are defined similar like views and are names like “Fragment.fragment.xml“.

Own Controller?

Fragments are created without controllers, they don’t have their own controllers they share controller of the view which is invoking it.

model

A **model** in SAP UI5 is a component that holds and manages the data of an application, allowing you to connect (bind) this data to the user interface (UI) elements. It serves as a bridge between the data and the UI, ensuring that changes in the data are reflected in the UI and vice versa.

- **JSON Model**: For client-side JSON data.
- **XML Model**: For client-side XML data.
- **Resource Model**: For managing resource bundles and internationalization.
- **OData Model**: For interacting with OData services.

- A model in SAP UI5 for consuming OData services, supporting data binding and operations like filtering and sorting.

View
The view focuses on the presentation layer and specifies the arrangement of UI controls, such as buttons, tables, text fields, and other visual elements.
Controller
The controller handles user inputs, manipulates the data model, and updates the view accordingly. It contains event handlers, such as functions that are executed when a button is pressed or when data changes.
4 Views and One Controller
Yes, it is possible to have multiple views managed by a single controller. This setup is useful when you want to centralize the logic that needs to be shared across different views.
4 Controllers and One View
While less common, it is possible to have one view interact with multiple controllers. However, SAP UI5 MVC architecture typically associates one controller with one view. You can still achieve this by having multiple controller instances or by combining their functionalities.
4 Views and One Controller: Centralizes logic in one controller for multiple views, which is useful for shared functionality.
4 Controllers and One View: Less common, but achievable by delegating actions from a main controller to multiple controllers.

SAP UI5

SAP UI5 is a framework for building responsive web applications. It leverages HTML5, CSS, and JavaScript to create user-friendly, platform-independent applications.

1. **Application Development**: Creating and structuring applications using the SAP UI5 framework.
2. **Debugging UI5 Applications**: Techniques and tools to troubleshoot and resolve issues in UI5 applications.
3. **Navigation Between Views**: Implementing view-to-view navigation using routing and other methods.
4. **Routing**: Configuring and managing navigation and routing in UI5 applications.
5. **Applying Custom Styles for UI Elements**: Customizing the appearance of UI elements using CSS.
6. **Consuming Services**: Integrating and using external services, such as OData, in UI5 applications.
7. **Creating OData Services**: Developing OData services for data manipulation and integration.

Business Application Studio (BAS)

BAS is a development environment tailored for SAP applications.

1. **Development Environment**: Using BAS for developing and managing SAP applications.
2. **Project Management**: Organizing and maintaining projects within BAS.

MVC Architecture

MVC (Model-View-Controller) is a design pattern for separating concerns in application development.

1. **Model-View-Controller Design Pattern**: Understanding and implementing MVC to separate data (Model), user interface (View), and control logic (Controller).

2. **Implementing MVC in Applications**: Applying the MVC pattern to create structured and maintainable applications.

UI Elements and Layout

This involves designing and managing the user interface elements and their layout.

1. **UI Elements**: Using various UI elements provided by SAP UI5.
2. **Layout Management**: Organizing and arranging UI elements for optimal user experience.
3. **Applying Custom Styles for UI Elements**: Using CSS to customize the appearance of UI elements.

Data Binding

Data binding is a technique to bind UI elements to data sources.

1. **One-way and Two-way Binding**: Binding data from the model to the view (one-way) and vice versa (two-way).
2. **Binding Data to UI Elements**: Connecting data models to UI elements to display and manipulate data.

CRUD Operations

CRUD stands for Create, Read, Update, and Delete—basic operations for data manipulation.

1. **Create, Read, Update, and Delete Operations**: Implementing these operations to manage data within an application.

Formatter Functions

Formatter functions are used to transform data before displaying it in the UI.

1. **Custom Formatter Functions for Data Transformation**: Writing functions to format data as needed for display.

Fragments

Fragments are reusable UI components in SAP UI5.

1. **Reusable UI Components**: Creating and managing fragments to be reused across different parts of an application.
2. **Managing Fragments**: Incorporating fragments into views and controllers.

Split App and Shell

Split App and Shell provide structural and navigational elements for complex UI5 applications.

1. **Split App Architecture**: Using split app layout to manage complex applications with master-detail views.
2. **Using Shell for Navigation and Layout**: Implementing Shell for a consistent application layout and navigation.

Internationalization (i18n)

i18n refers to the process of designing applications that can be adapted to various languages and regions.

1. **Implementing i18n in Applications**: Techniques for making applications support multiple languages.
2. **Managing Translations and Locales**: Handling translations and locale-specific data.

Component.js

Component.js is used for application configuration and initialization in SAP UI5.

1. **Application Configuration and Initialization**: Setting up Component.js for initializing and configuring applications.

Debugging

Debugging is the process of identifying and fixing issues in an application.

1. **Debugging Techniques for UI5 Applications**: Tools and methods for debugging SAP UI5 applications.

Navigation Between Views

Effective navigation is essential for user experience in multi-view applications.

1. **View Management**: Techniques for managing different views in an application.
2. **Routing Techniques**: Implementing routes to handle navigation between views.

Reading Services Using Ajax

Ajax is used to make asynchronous HTTP requests to fetch data from services.

1. **Ajax Requests for Data Retrieval**: Using Ajax to read and retrieve data from external services.

Creating OData Services and Consuming Services

OData is a protocol for building and consuming RESTful APIs.

1. **OData Service Creation**: Developing OData services to expose data and operations.
2. **Consuming OData Services in Applications**: Integrating OData services into applications to consume data.

Web Technologies

These are fundamental technologies for web development.

1. **HTML5**: Markup language for structuring and presenting content on the web.
2. **CSS**: Style sheet language for describing the presentation of web pages.
3. **JavaScript**: Programming language for creating dynamic and interactive web content.
4. **Ajax**: Technique for asynchronous web requests.
5. **jQuery**: JavaScript library for simplifying HTML document traversal, event handling, and animation.

Introduction

No screen sharing, there is a problem with the system

Kapil dev Bidua(P)

From Jhansi Uttar Pradesh(P), curruntly living in noida

I have 4 years of experience in SAP UI 5, JavaScript, react native, react, and redux(P)

and my other skills are I am a Consultant, Problem-Solving person and can do Multi-Tasking with Strong Interpersonal Communication Skills

I also have 2.5 year of experience in SAP UI5 and Fiori

- Expertise in which im good includes SAP UI5, BAS, MVC Architecture, UI Elements, Layout, Binding, CRUD Operation, Formatter Function,

Fragments, Split App, Shell, i18n, Component.js, Debugging UI5 Application, Navigation Between Views, Routing, reading services using Ajax,

Applying Custom Style for UI Elements, Creating OData services and Consuming Services, Internationalization, HTML5, CSS, JavaScript, Ajax, jQuery.

Possess working exposure in both Agile and Waterfall Methodologies. I have knowledge of Application Development for web and mobile.

jQuery

(HTML document traversal and manipulation, event handling, animation, and Ajax)

ProjectsssSAP UI5

Smart Automotive Solutions

Service Request Management

Aims to streamline service request management by developing a SAP Fiori application for customers to submit requests, track status, and communicate with technicians.

Responsibilities:-

Creation of Fiori Launchpad and Creation of Tiles,Groups and Catalogs in Fiori Launchpad.

•Developed Custom Ui5 Fiori Application

•Configured standard Fiori Applications.

•Working on SAP DVH also.(SAP Data Volume History (DVH) is a tool used to analyze and manage data growth in an SAP system. It helps monitor data volume changes over time, allowing for better planning and management of database resources. DVH can track the size of tables and indexes, providing insights into data usage patterns and helping to identify potential areas for optimization.

Would you like more detailed information on a specific aspect of SAP DVH?

)

Request Submission:

    - Utilize SAPUI5 controls like sap.m.Input, sap.m.Select, or sap.m.DatePicker to collect user input for request details.
    - Implement event handlers for UI actions like pressing a submit button.
    - Use SAPUI5's model and binding features to bind form inputs to a data model.
    - Use AJAX requests or SAPUI5's sap.ui.model.odata.v2.ODataModel for sending data to the backend server for processing.
- **Status Tracking**:
    - Display the status of submitted requests using SAPUI5 controls like sap.m.ObjectStatus or sap.m.Label.
    - Retrieve the status information from the backend using data binding or AJAX requests.
    - Implement logic to dynamically update the UI based on changes in the status data.
- **Notifications**:
    - Use SAPUI5's built-in controls like sap.m.MessageToast or sap.m.MessageBox to display notifications to the user.
    - Show success messages upon successful request submission or completion.
    - Display error messages in case of failed requests or validation errors.
- **Messaging**:
    - Implement messaging functionalities using SAPUI5 controls like sap.m.FeedInput for direct messaging features within the application.
    - For more complex messaging features, integrate with SAP messaging services or backend systems.
    - Implement real-time messaging using WebSocket communication, if required.

- **Requirements Gathering**:
    - Understand the requirements of the Service Request Management application, including the types of service requests, user roles (customers, technicians), and desired functionality.
    - Identify key features such as request submission, status tracking, notifications, and messaging.
- **Design Phase**:
    - Design the user interface of the Fiori application using SAPUI5, including wireframes, mockups, and UI prototypes.
    - Define the application architecture, including views, controllers, models, and services required to implement the desired functionality.
    - Design the data model to represent service requests, customer information, technician assignments, and status updates.
- **Development Phase**:
    - Set up a new SAPUI5 project in SAP Web IDE or any other development environment.
    - Implement views and controllers for the various screens of the Fiori application, such as the service request submission form, request status overview, and messaging interface.
    - Implement data models to represent service request data and customer information using JSON models or OData models.

    - Implement data binding to connect UI elements with data models, ensuring that changes in the data are reflected in the UI.
    - Implement routing and navigation to enable seamless navigation between different screens of the application.
    - Implement features such as form validation, error handling, and user authentication to enhance the usability and security of the application.
- **Integration Phase**:
    - Integrate the Fiori application with backend systems, such as SAP ERP or a custom service management system, to fetch and update service request data.
    - Implement communication channels for customers to submit service requests and receive status updates, such as email notifications or push notifications.
- **Testing and Validation**:
    - Write unit tests and integration tests to validate the functionality, performance, and reliability of the Fiori application. QUnit
    - Conduct end-to-end testing to ensure seamless integration between the frontend SAPUI5 application and the backend systems.
    - Perform user acceptance testing (UAT) with real users to gather feedback and identify any usability issues or bugs.
- **Deployment and Monitoring**:
    - Deploy the Fiori application to a production environment, such as the SAP Fiori Launchpad or a standalone web server.
    - Monitor the performance, availability, and security of the application in production, and address any issues or incidents as they arise.

ParkSmart

Parksmart, Parksmart
The ParkSmart Driver App project involves the design and development of a Web application The app's main functionalities include appealing and paying parking charge notices, as well as making payments for parking at approved ParkSmart Permitted locations. Your role encompasses various responsibilities throughout the development process. Here's a breakdown:
Responsibilities:-
•Creation of Fiori Launchpad and Creation of Tiles, Groups and Catalogs in Fiori Launchpad.
•Developed Custom Ui5 Fiori Application using Smart Table on HANA.
•Developed new Launchpad Extension Application in Fiori Launchpad to add Custom Icon in Fiori launchpad and on click on that icon one Popup will open to show AFO Favorites.
•Worked on existing developed application and add the Calendar Legend functionality.
•integrate in Map and geolocation
Design and Build Applications (Web):
Designing and developing the user interface for the Web to ensure a seamless and user-friendly experience.
Collaboration:
Engaging in discussions to contribute ideas and solutions to enhance the

overall functionality and user experience.
Testing Libraries: - QUnit or OPA5
Front-End Frameworks :- SAP UI 5, SAP BTP
Programming Languages: - JavaScript , HTML, CSS, XML
Design Tools:- Adobe XD, Figma
UI/UX Design: - Responsive design, Material UI
Bug Fixing:
Identifying and addressing bottlenecks in the application's performance.
Fixing any bugs or issues that may arise during the development and usage of the applications.
Code Quality and Organization:
Contributing to maintaining high code quality and organization within the application's codebase.
Ensuring adherence to coding standards and best practices.
Automatization:
Contributing to the automation of development processes, testing, or deployment to improve efficiency and reliability.
Client Requirements Gathering:
Responsible for gathering client requirements to understand and incorporate specific needs and expectations into the development process.
Integration of ODATA with SAP UI 5 App (ODATA Functionality):
Integrating a REST API with the SAP UI 5 app, likely utilizing the ODATA functionality for state management.
Integration of Geolocation and Google Maps:
Responsible for integrating geolocation services and Google Maps to provide location-based services and features within the app.

Setting up push notification service in SAP Cloud Platform (SCP) involves several steps. Below is a general outline of the process:

- **Access SAP Cloud Platform Cockpit**:
  - Log in to the SAP Cloud Platform Cockpit using your credentials.
- **Navigate to Services**:
  - Once logged in, navigate to the "Services" section of the SAP Cloud Platform Cockpit.
- **Select Push Notification Service**:
  - Find and select the "Push Notification" service from the list of available services. If it's not already available, you may need to subscribe to the service.
- **Create a New Instance**:
  - Click on the "Create" button to create a new instance of the Push Notification service.
- **Configure Service Instance**:

    - Provide a name and description for the service instance.
    - Select the appropriate service plan based on your requirements (e.g., trial, standard).
    - Configure any additional settings or options as needed.
- **Obtain Credentials**:
    - Once the service instance is created, you'll typically receive credentials (such as API keys or tokens) that you'll use to authenticate your application when interacting with the push notification service.
- **Access Push Notification Dashboard**:
    - Navigate to the dashboard or administration console of the Push Notification service within the SAP Cloud Platform Cockpit.
    - Here, you can manage various aspects of the push notification service, including configuring push notification channels, defining notification templates, and monitoring push notification activity.
- **Integrate Push Notification Service with Your Application**:
    - In your application code (whether it's a SAPUI5 application or another type of application), integrate the push notification service by using the provided SDKs, APIs, or libraries.
    - Use the obtained credentials to authenticate your application when interacting with the push notification service.
    - Implement logic for registering devices for push notifications, handling push notification events, and sending notifications to registered devices.
- **Test and Deploy**:
    - Test the push notification functionality thoroughly to ensure that devices are successfully registered and notifications are delivered as expected.
    - Deploy your application with push notification integration to your development, testing, or production environment as appropriate.
- **Monitor and Maintain**:
    - Monitor the performance and reliability of push notifications in your application.
    - Handle any issues or errors that may arise during push notification delivery.
    - Keep the push notification functionality up-to-date with any changes or updates to the SAP Cloud Platform services or your application requirements.

BAS (Business Application Studio):

- A modern development environment tailored for building business applications, especially those using SAP technologies.

MVC Architecture:

- Model-View-Controller architecture used in SAP UI5 to separate the application's data, user interface, and control logic.

UI Elements:

- Various SAP UI5 controls such as buttons, tables, forms, and input fields used to build user interfaces.

Layout:

- Layout mechanisms in SAP UI5 like VBox, HBox, Grid, and FlexBox to arrange UI elements on the screen.

Binding:

- Data binding mechanisms in SAP UI5 to synchronize data between the model and the view.

CRUD Operation:

- Implementing Create, Read, Update, and Delete operations in SAP UI5 applications.

Formatter Function:

- Functions used to format data in the UI, such as converting dates, numbers, or strings to a specific format.

Fragments:

- Reusable UI parts in SAP UI5 that can be included in views without creating a new controller.

Split App:

- A layout for applications that typically display a master-detail relationship, useful for navigation on larger screens.

Shell:

- The SAP Fiori launchpad container providing services for application navigation, personalization, and configuration.

i18n (Internationalization):

- Techniques for making applications support multiple languages and regions.

Component.js:

- The entry point for defining the metadata and configuration of an SAP UI5 application.

Debugging UI5 Application:

- Techniques for troubleshooting and fixing issues in SAP UI5 applications, using tools like the browser's developer console.

Navigation Between Views:

- Methods for transitioning between different views within an SAP UI5 application.

Routing:

- Defining routes and navigation patterns in SAP UI5 to manage URL-based navigation.

Reading Services using Ajax:

- Using AJAX to make asynchronous HTTP requests to fetch data from services.

Applying Custom Style for UI Elements:

- Using CSS to apply custom styles to SAP UI5 controls and components.

Creating OData Services and Consuming Services:

- Building and consuming OData services for data exchange in SAP UI5 applications.

General Web Development

HTML5:

- The latest version of Hypertext Markup Language, used for structuring and presenting content on the web.

CSS:

- Cascading Style Sheets used for describing the presentation of a document written in HTML or XML.

JavaScript:

- A programming language used to create dynamic and interactive effects within web browsers.

Ajax:

- Asynchronous JavaScript and XML used for creating asynchronous web applications, allowing web pages to be updated asynchronously by exchanging small amounts of data with the server.

jQuery:

- A fast, small, and feature-rich JavaScript library that simplifies things like HTML document traversal and manipulation, event handling, and animation.

Fiori

What is Fiori?

It's a design aprroach which developed by SAP to build Responsive web Application using 5 Fiori PriciplAdaptive,Simple, Coherent,Roled Based,Delightfull

sap fiori design principles

ROLE-BASED - >

role base in application basee on the other roles .

SAP Fiori applications are designed to be role-based, focusing on the needs and tasks of specific user roles. This principle ensures that users see only the information and functions relevant to their roles, simplifying their experience and increasing productivity. By tailoring the UI to specific roles, users can perform their tasks more effectively, with a clear and intuitive interface that supports their daily workflows.

DELIGHTFUL ->

delightful which is used on user friendly application .

Applications should be delightful, offering a user experience that is enjoyable and satisfying. This involves designing user-friendly interfaces that are visually appealing and engaging, enhancing the overall user experience.

COHERENT ->

The design should be coherent, ensuring consistency across the application.

The design should be coherent, ensuring consistency across the application. This means providing a unified and consistent UI, avoiding complexity, and ensuring that the user interface is predictable and easy to understand.

SIMPLE ->

The design should be simple, focusing on making operations quick and fast.

The design should be simple, focusing on making operations quick and fast. Simplicity in design helps users accomplish their tasks efficiently without unnecessary steps or complications, streamlining the user experience.

ADAPTIVE ->

SAP Fiori is adaptive, meaning the design is flexible and responsive.

The design should be simple, focusing on making operations quick and fast. Simplicity in design helps users accomplish their tasks efficiently without unnecessary steps or complications, streamlining the user experience.

SAP Fiori is adaptive, meaning the design is flexible and responsive. It adjusts seamlessly to different devices and screen sizes, providing an optimal user experience regardless of the device being used.

What is UI5 & Fiori?

No, UI5 is a Framework which is used to build a Fiori Applications. Fiori is design approach to develop a web Application.

One App to Another App

SAP Fiori employs a concept called "Cross-App Navigation." This is part of the Fiori Launchpad services and is typically managed using Semantic Object and Action mappings. Here's a brief overview of how it works:

Cross-App Navigation (Sematic Object + Action)

Cross-App Navigation Syntax: var oCrossAppNavigator = sap.ushell.Container.getService("CrossApplicationNavigation"); // get a handle on the global XAppNav service var hash = (oCrossAppNavigator && oCrossAppNavigator.hrefForExternal({ target: { semanticObject: "Supplier", action: "display" }, params: { "supplierID": supplier } })) || ""; // generate the Hash to display a Supplier oCrossAppNavigator.toExternal({ target: { shellHash: hash } }); // navigate to Supplier application }

Semantic Objects: These are abstract representations of business entities (e.g., "Customer," "SalesOrder"). Semantic objects are used to decouple the technical implementation from the navigation intent.

Actions: Actions define what can be done with a semantic object (e.g., "display," "create," "edit"). Each action is associated with a specific application and view.

Intent-Based Navigation: This combines semantic objects and actions into a navigation intent (e.g., #SalesOrder-display). This intent is mapped to a target application and view.

Configuration in SAP Fiori Launchpad: The navigation targets are configured in the Fiori Launchpad using target mappings. This involves defining which application should be launched for a given intent.

Cross-App Navigation API: SAP provides an API (sap.ushell.Container.getService("CrossApplicationNavigation")) that enables applications to trigger navigation to another Fiori app programmatically.

Define Semantic Object and Action:

- Go to the SAP Fiori Launchpad Designer.

- Create or use an existing semantic object.
- Define actions for the semantic object and map them to the target application and view.

Configure Target Mapping:

- In the Fiori Launchpad Designer, configure target mappings to define the relationship between the semantic object, action, and the target application.

Use Cross-App Navigation in Your Code:

```
// Example code to navigate from one app to another
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/ushell/services/CrossApplicationNavigation"
], function (Controller, CrossApplicationNavigation) {
    "use strict";
    return Controller.extend("myApp.controller.MyController", {
        onNavigate: function () {
            var oCrossAppNavigator = sap.ushell.Container.getService("CrossApplicationNavigation");
            // Define the target app's semantic object and action
            var sSemanticObject = "SalesOrder";
            var sAction = "display";
            // Define parameters to pass to the target app
            var oParams = {
                "SalesOrderId": "12345"
            };
            // Navigate to the target app
            oCrossAppNavigator.toExternal({
                target: {
                    semanticObject: sSemanticObject,
                    action: sAction
                },
                params: oParams
            });
        }
    });
});
```

Testing the Navigation:

- Deploy both apps to the Fiori Launchpad.
- Ensure that the target mapping is correctly configured.
- Test the navigation to verify that it works as expected.

To Create Sematic Object - ui2/semobj
TO Create package - SE80
To Create TR(Transport Request) - SE09
1. Fiori LaunchPad Designer => Sematic Object - Transaction App - VA01
2.SICF Node (SICF) => GUI Transaction - VA01
S4HANA 2022
Standard Application Activation
1.Fiori Ref Lib - https://fioriappslibrary.hana.ondemand.com/sap/fix/externalViewer/#/detail/Apps('F1

2.Select a Version of Our System
3.Status-Magnifier Symbol
4. SICF => FIN_BANK=>AcϴvateFron- End)
50%
5.iwfnd/maint_service = > FCLM_BM_SRV => Package and System Alias -> Activate 6.ui2/flpd_cust (Fiori Launch Pad Designer) - > i.create Catalog ii.Create Group
7.-> Catalog - SAP_TC_FIN_CM_COMMON -> (Bank+manage) -> Custom Catalog
8. - Group - Create Group and Inside Assign it.
90%
9. Activate Application (Green Color Tick) ZGROUP AND ZCATALOG ZROLE_KAPIL - SPACE , CATALOG , GROUP 100% KAPIL - PFCG
Rapid Activation ->
Activation of multiple app in a single shot based roles(BR) 1.STC01 -> TaskList Tcode
2.SAP_FIORI_CONTENT_ACTIVATION - SAP_BR_CASH_MANAGER
3…....Execute Output -> ZSAP_BR_CASH_MANAGER
Space and Pages It was came in S4 HANA 2020 Groups was deprecated (Spaces and Pages) two Fiori Apps =>
1. Manage LaunchPad Pages ,
2. Manage Launchpad Spaces
Spaces and Pages in SAP Fiori provide a modern way to organize and navigate applications within the SAP Fiori Launchpad. They offer a more structured and user-friendly experience compared to the traditional tile-based approach.
Rapid Activation in the context of SAP Fiori refers to a streamlined process to quickly enable and deploy Fiori apps. This process allows businesses to accelerate the adoption of Fiori apps by minimizing the complexity and time required for activation. The main goal of Rapid Activation is to make the deployment of Fiori apps as fast and efficient as possible.

Key Features of Rapid Activation:

1. **Simplified Process**: Rapid Activation simplifies the process of enabling Fiori apps by reducing the steps and configurations traditionally needed.
2. **Predefined Content**: It uses predefined technical content, such as roles, authorizations, and OData services, which are bundled and ready for activation.
3. **Reduced Manual Effort**: Automation tools and predefined settings reduce the manual effort required to configure and deploy apps.
4. **Best Practices**: The process follows SAP best practices, ensuring that the Fiori apps are deployed correctly and optimally.
5. **Comprehensive Coverage**: Rapid Activation covers a wide range of Fiori apps across various lines of business and industries, making it applicable to many different use cases.

Steps in Rapid Activation:

1. **Preparation**:
    - Ensure your SAP system meets the necessary prerequisites.
    - Install the relevant SAP Fiori front-end and back-end components.
2. **Check Predefined Content**:
    - SAP provides predefined roles, catalogs, and groups that are needed for the Fiori apps. These are checked and validated during the process.
3. **Run Task Lists**:

    - SAP provides specific task lists for activating Fiori apps. These task lists automate many of the steps required for activation. Examples include:
        - SAP_FIORI_CONTENT_ACTIVATION
        - SAP_GATEWAY_ACTIVATION
4. **Activate OData Services**:
    - The necessary OData services for the Fiori apps are activated automatically through the task lists.
5. **Assign Roles**:
    - Predefined roles are assigned to users to give them access to the activated Fiori apps.
6. **Test and Validate**:
    - Once the activation process is complete, test the Fiori apps to ensure they are functioning as expected.

Benefits of Rapid Activation:

- **Time Efficiency**: Significantly reduces the time needed to deploy Fiori apps.
- **Cost Savings**: Lower implementation costs due to reduced manual configuration and faster deployment.
- **Consistency**: Ensures a consistent and standardized approach to activating Fiori apps.
- **Quick Adoption**: Enables quicker adoption of Fiori apps, helping businesses to leverage modern UI/UX capabilities sooner.

Example Use Case:

A company wants to quickly enable a suite of SAP Fiori apps for their sales team. By using Rapid Activation, they can:

- Prepare their system with the necessary components.
- Use the predefined roles and content provided by SAP.
- Run the task lists to automate the activation of the relevant OData services and Fiori apps.
- Assign the appropriate roles to the sales team members.
- Test the apps to ensure they are working correctly and the sales team can start using them with minimal delay.

SAP Fiori Elements is a framework that allows developers to create SAP Fiori applications quickly and efficiently using standardized templates. These templates follow SAP Fiori design principles, ensuring a consistent and high-quality user experience. Fiori Elements leverages metadata-driven development, which means that a significant portion of the application logic is generated based on OData annotations and metadata.

Key Features of SAP Fiori Elements:

1. **Metadata-Driven Development**:
    - Applications are generated based on metadata provided by OData services.
    - This approach minimizes the need for custom UI coding, speeding up development.
2. **Standardized Templates**:
    - Predefined templates for common application patterns ensure a consistent user experience.
    - Examples include List Report, Object Page, Analytical List Page, Overview Page, and Worklist.
3. **OData Annotations**:

- Annotations in the OData service define the data and behavior of the UI elements.
- These annotations provide instructions for the Fiori Elements framework on how to render and interact with data.

4. **Extensibility**:
   - While Fiori Elements provide a lot of out-of-the-box functionality, it also allows for custom extensions to meet specific business requirements.
5. **SAPUI5 Integration**:
   - Fiori Elements are built on top of SAPUI5, ensuring compatibility with other SAP Fiori technologies and components.

Common Fiori Elements Templates:

1. **List Report**:
   - Displays a list of items based on OData service data.
   - Supports filtering, sorting, and navigation to detailed item views.
2. **Object Page**:
   - Displays detailed information about a single item from the list.
   - Contains sections and subsections to organize related data and actions.
3. **Analytical List Page**:
   - Combines analytical capabilities with transactional data.
   - Includes visualizations like charts and graphs along with a list report.
4. **Overview Page**:
   - Provides a dashboard-like view with multiple cards showing key information and actions.
5. **Worklist**:
   - Focuses on tasks and work items that need attention or action.
   - Supports filtering and sorting to help users prioritize their tasks.

Example Scenario: Creating a List Report and Object Page

1. **Prepare OData Service**:
   - Ensure your OData service is ready and includes necessary annotations. For example, @UI.selectionFields for filter fields, @UI.lineItem for list items, and @UI.headerInfo for object header information.
2. **Create the Application**:
   - Use the SAP Business Application Studio or SAP Web IDE to create a new Fiori Elements application.
   - Select the List Report and Object Page template.
   - Connect the application to your OData service.
3. **Configure Annotations**:
   - Define annotations in the OData service to describe the UI elements.
   - Example annotation for a list report:xmlCopy code

     <Annotations Target="YourService.YourEntityType">
   - <Annotation Term="UI.LineItem">
   - <Collection>
   - <Record Type="UI.DataField">
   - <PropertyValue Property="Label" String="ID"/>
   - <PropertyValue Property="Value" Path="ID"/>
   - </Record>

```
        <Record Type="UI.DataField">
          <PropertyValue Property="Label" String="Name"/>
          <PropertyValue Property="Value" Path="Name"/>
        </Record>
      </Collection>
    </Annotation>
  </Annotations>
```

4. **Deploy and Test**:
    - Deploy the application to your SAP Fiori Launchpad.
    - Test the application to ensure it displays the list report and allows navigation to the object page correctly.

Benefits of Using SAP Fiori Elements:

- **Rapid Development**: Accelerates the development process by reducing the need for custom coding.
- **Consistency**: Ensures a consistent look and feel across different applications.
- **Quality**: Leverages SAP's best practices and design guidelines, improving the overall quality of the applications.
- **Maintenance**: Simplifies maintenance and updates due to the standardized framework.

JavaScript

1.What is the difference between let and var

The difference between let and var is in the scope of the variables they create: Variables declared by let are only available inside the block where they're defined. Variables declared by var are available throughout the function in which they're declared.

2.What is IIFE immediately invoked function expression in JavaScript?

An IIFE (Immediately Invoked Function Expression) is a function that runs the moment it is invoked or called in the JavaScript event loop. Having a function that behaves that way can be useful in certain situations. IIFEs prevent pollution of the global JS scope.

Ex:- example, "Hello" is returned because the greeting() function is invoked immediately after it is defined, as a result of having the second set of parentheses () at the end. As with the other IIFEs above, once the function is defined, it is immediately invoked.

3.What are closures in JavaScript?

Closures are functions that have access to the parent scope, even after the parent function has closed. This is due to lexical scoping which allows a function to use variables declared outside its immediate lexical scope.

Ex : - example we'll demonstrate that a closure contains any and all local variables that were declared inside the outer enclosing function

4.What is a promise in javascript

Promise is an object that will produce a single value some time in the future. If the promise is successful, it will produce a resolved value, but if something goes wrong then it will produce a reason why the promise failed. The possible outcomes here are similar to that of promises in real life.

A promise is an object that may produce a single value some time in the future with either a resolved value or a reason that it's not resolved(for example, network error). It will be in one of the 3 possible states: fulfilled, rejected, or pending.
A JavaScript Promise object can be:

- **Pending**
- **Fulfilled**
- **Rejected**

The Promise object supports two properties: state and result.
While a Promise object is "pending" (working), the result is undefined. When a Promise object is "fulfilled", the result is a value.
When a Promise object is "rejected", the result is an error object.

5.What is callback function in JavaScript interview questions?

A callback is a JavaScript function that is passed to another function as an argument or a parameter. This function is to be executed whenever the function that it is passed to gets executed.
Ex - A callback's purpose is to execute code in response to an event. These events can be like mouse clicks; with callback, we can add text at the end of each statement like"execute this code every time the user clicks a key on the

6.What is a polyfill in javascript

Polyfills are pieces of code that provide modern functionality to older browsers that lack native support for those features. They bridge the gap between the JavaScript language features and APIs that are available in modern browsers and the limited capabilities of older browser versions.
Ex - Polyfills are pieces of code that provide modern functionality to older browsers that lack native support for those features. They bridge the gap between the JavaScript language features and APIs that are available in modern browsers and the limited capabilities of older browser versions.

7.What is a Async /await in javascript

The async keyword transforms a regular JavaScript function into an asynchronous function, causing it to return a Promise. The await keyword is used inside an async function to pause its execution and wait for a Promise to resolve before continuing.

8.What is event loop in javascript

An event loop is something that pulls stuff out of the queue and places it onto the function execution stack whenever the function stack becomes empty. The event loop is the secret by which JavaScript gives us an illusion of being multithreaded even though it is single-threaded
Ex - something that pulls stuff out of the queue and places it onto the function execution stack whenever the function stack becomes empty

1

Shallow Copy

A ***shallow copy*** is a copy that only goes one level deep. In other words, it copies the object and all its properties, but any nested objects or arrays will still reference the same memory location as the original object. It means that if you make changes to the nested object, it will also affect the original object, as well as the copied object.

```
1. const originalObject = { a: 1, b: { c: 2 } };
2. const shallowCopy = { ...originalObject };
1. shallowCopy.a = 3; // Changes shallowCopy, but not originalObject
2. shallowCopy.b.c = 4; // Changes both shallowCopy and originalObject
```

Deep Copy

A ***deep copy*** is a copy that creates a new object with new memory locations for all of its properties and nested objects or arrays. It means that if you make changes to the copied object or any of its nested objects or arrays, it will not affect the original object.

1. const originalObject = { a: 1, b: { c: 2 } };
2. const deepCopy = JSON.parse(JSON.stringify(originalObject));
1. deepCopy.a = 3; // Changes deepCopy, but not originalObject
2. deepCopy.b.c = 4; // Changes deepCopy, but not originalObject

What is a Recursive Function?

A recursive function is a function that calls itself somewhere within the body of the function. Below is a basic example of a recursive function.

```
// program to count down numbers to 1
function countDown(number) {
    // display the number
    console.log(number);
    // decrease the number value
    const newNumber = number - 1;
    // base case
    if (newNumber > 0) {
        countDown(newNumber);
    }
}
countDown(4);
```

Higher-Order Functions In JavaScript

functions that take one or more functions as arguments, or return a function as their result.

Higher-order functions are a functional programming pattern when functions are being passed as arguments to other functions or returned as a result.
The example below illustrates the HoF pattern when one function takes another function as an argument and returns composed data:

```
function log(item) {
    return console.log(item);
}
function process(data, callback) {
    for (let i = 0; i < data.length; i += 1) {
        callback(data[i]);
    }
}
process([1, 2, 3], log); // prints 1; 2; 3;
///
function greet(name){
            return `Hi!! ${name} `;
}
function greet_name(greeting,message,name){
            console.log(`${greeting(name)} ${message}`);
}
greet_name(greet,'Welcome To GeeksForGeeks','JavaScript');
```

What is a prototype in JavaScript?

Function prototype tells the number of arguments passed to the function. Function prototype tells the data types of each of the passed arguments. Also, the function prototype tells the order in which the arguments are passed to the function.

Every object in JavaScript has a built-in property, which is called its prototype. The prototype is itself an object, so the prototype will have its own prototype, making what's called a prototype chain. The chain ends when we reach a prototype that has null for its own prototype

What is the difference between a set and a weak set in JavaScript?

The main differences to the Set object are: WeakSet s are collections of objects and symbols only. They cannot contain arbitrary values of any type, as Set s can. The WeakSet is weak, meaning references to objects in a WeakSet are held weakly.

Differences Between Set and WeakSet

Now that we've discussed Set and WeakSet individually, let's highlight the key differences between them:

Allowed Values:

- Set: Set can store any type of values, including primitive types (numbers, strings, booleans) and objects. It ensures that only unique values are stored.
- WeakSet: WeakSet can only store objects. Attempting to add non-object values like numbers or strings will result in an error. This restriction is because WeakSet is designed to work with object references.

Garbage Collection:

- Set: Objects stored in a Set are held by strong references, meaning they won't be garbage collected as long as the Set retains a reference to them. This can lead to memory leaks if you unintentionally retain objects.
- WeakSet: Objects stored in a WeakSet are held by weak references, allowing them to be garbage collected as soon as no other strong references to those objects exist. This behavior is useful for scenarios where you want to avoid memory leaks and allow automatic cleanup of objects.

Iteration Order:

- Set: Set maintains the insert ion order of values. When you iterate over a Set, values are returned in the order they were added.
- WeakSet: WeakSet does not provide a way to iterate over its elements. This is because it doesn't guarantee any specific order, and it's primarily used for managing object relationships rather than value retrieval.

Size Property:

- Set: Set has a size property that allows you to determine the number of elements it contains.
- WeakSet: WeakSet does not have a size property, so you cannot directly obtain the number of elements it contains. You typically use WeakSet for membership checks and object management rather than counting its elements.

Use Cases:

- Set: Use Set when you need to manage collections of unique values and order matters, or when you want to use non-primitive values as keys.

- WeakSet: Use WeakSet when you want to store objects but do not want to prevent them from being garbage collected, or when you need to create relationships between objects without strong references.

What is this?

In JavaScript, the this keyword refers to an **object**.
Which object depends on how this is being invoked (used or called).
The this keyword refers to different objects depending on how it is used:

In an object method, this refers to the **object**.
Alone, this refers to the **global object**.
In a function, this refers to the **global object**.
In a function, in strict mode, this is undefined.
In an event, this refers to the **element** that received the event.
Methods like call(), apply(), and bind() can refer this to **any object**.

9.what is call apply and bind in javascript
call : binds the this value, invokes the function, and allows you to pass a list of arguments.
apply : binds the this value, invokes the function, and allows you to pass arguments as an array.
bind : binds the this value, returns a new function, and allows you to pass in a list of arguments.

Difference between UI and UX

The UI stands for User Interface and UX stands for User Experience.
UI is designed by the use of frameworks, but to get a user experience we need to follow guidelines and design principles
UI is conceived as a technical object and UX is an emotional object
We can change the UI but we can never change the UX

- **User Interface (UI)**:
  - The UI refers to the visual elements of the app, such as buttons, menus, icons, colors, typography, etc.
  - In the banking app, the UI includes the layout of the screens, the design of the buttons, the placement of menus, the color scheme used for different sections, and the overall visual aesthetics.
  - For example, the UI might have a "Transfer Money" button prominently displayed on the home screen, and when you tap it, it leads you to a screen with fields to enter the recipient's account number, the amount to transfer, etc.
  - UI focuses on how the app looks and feels to the user.
- **User Experience (UX)**:
  - The UX refers to the overall experience a user has while interacting with the app, including how easy or difficult it is to accomplish tasks, how intuitive the app is to navigate, how engaging the content is, etc.
  - In the banking app, good UX design ensures that users can easily and securely perform banking transactions without confusion or frustration.
  - For example, a good UX design might include clear and concise instructions on each screen, error messages that provide helpful guidance when something goes wrong, and a smooth and intuitive navigation flow that guides users through the app's features.

- UX focuses on the overall satisfaction and enjoyment users derive from using the app.

Array Iteration in javascript
9.What is Array Iteration in javascript

forEach:-
The forEach() method is an iterative method. It calls a provided callbackFn function once for each element in an array in ascending-index order. Unlike map() , forEach() always returns undefined and is not chainable. The typical use case is to execute side effects at the end of a chain.

map:-
map() creates a new array from calling a function for every array element. map() does not execute the function for empty elements. map() does not change the original array.
filter:-
Description. The filter() method is an iterative method. It calls a provided callbackFn function once for each element in an array, and constructs a new array of all the values for which callbackFn returns a truthy value. Array elements which do not pass the callbackFn test are not included in the new array.
reduce:-
The reduce() method of Array instances executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value
some and every :-
Summary. The every() array method in JavaScript is used to check if all the elements of the array satisfy the callback function condition or not.
The some() array method in JavaScript is used to check if at least one of the elements passes the callback test or not.
find and findIndex:-
The findIndex() method of Array instances returns the index of the first element in an array that satisfies the provided testing function. If no elements satisfy the testing function, -1 is returned.
See also the find() method, which returns the first element that satisfies the testing function (rather than its index).

JavaScript Array Methods

concat:-
The concat() method concatenates (joins) two or more arrays. The concat() method returns a new array, containing the joined arrays.
The join:-
The join() method of Array instances creates and returns a new string by concatenating all of the elements in this array, separated by commas or a specified separator string. If the array has only one item, then that item will be returned without using the separator.
indexOf and lastIndexOf
Description. The indexOf() and lastIndexOf() function return a numeric index that indicates the starting position of a given substring in the specified metadata string:
indexOf() returns the index for the first occurrence of the substring.
lastIndexOf() returns the index for the last occurrence of the substring.

slice:-
The slice() method extracts a part of a string. The slice() method returns the extracted part in a new string. The slice() method does not change the original string. The start and end parameters specifies the part of the string to extract.
splice:-
The splice() method is used to add or remove elements of an existing array and the return value will be the removed items from the array. If nothing was removed from the array, then the return value will just be an empty array. Here is the basic syntax: splice(start, optional delete count, optional items to add)
pop and push :-
push() and pop() are two Array methods in JavaScript. push() is used to add an element to the end of an array, and pop() is used to remove the last element from an array.
shift and unshift
shift() removes the first element from an array and returns it, while unshift() adds one or more elements to the beginning of an array and returns the new length.
reverse:-
The reverse() method reverses the order of the elements in an array. The reverse() method overwrites the original array.
filter:-
The filter() method creates a new array filled with elements that pass a test provided by a function. The filter() method does not execute the function for empty elements. The filter() method does not change the original array.
Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.

isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

What is the difference between find and filter in array?

1.find() returns the first matching element, while . filter() returns an array of all matching elements.
2.filter() method creates a new array with all elements that pass a given condition, while the find() method returns the first element that satisfies a condition or is undefined if no match is found.
3.filter() returns an array containing the element that satisfies the condition, but find() returns the element itself that satisfies the condition.

What is the difference between forEach and map?

- The forEach() method does not returns a new array based on the given array. The map() method returns an entirely new array. The forEach() method returns “undefined“. The map() method returns the newly created array according to the provided callback function.
- map returns a new array with the results of the function, while forEach does not return anything and only modifies the original array.

What is difference between map and filter in JavaScript?

Map: returns an array of pieces of information from the original array. In the callback function, return the data you wish to be part of the new array. Filter: returns a subset of the original array based on custom criteria.

What is the difference between map and forEach in streams?

map() is similar to . forEach() because they are both array iterators that execute a provided function on every element within the given array. Now the big difference between the two, is that with . map() we don't need to tell our function to add every element to a new array like we do with

Differences between forEach() and map() methods

forEach()	map()
The forEach() method does not returns a new array based on the given array.	The map() method returns an entirely new array.
The forEach() method returns "*undefined*".	The map() method returns the newly created array according to the provided callback function.
The forEach() method doesn't return anything hence the method chaining technique cannot be applied here.	With the map() method, we can chain other methods like, reduce(),sort() etc.
It is not executed for empty elements.	It does not change the original array.

Property	map()	filter()	reduce()
Return type	Returns a new array	Returns a new array	Returns a single element
Action	Modifies each element of the array	Filter out the element which passes the condition	Reduces array to a single value
Parameters	value, index, array	value, index, array	accumulator, value, index, array

Difference between Map and WeakMap:

Map	WeakMap
A Map is an unordered list of key-value pairs where the key and the value can be of any type like string, boolean, number, etc.	In a Weak Map, every key can only be an object and function. It used to store weak object references.
Maps are iterable.	WeakMaps are not iterable.
Maps will keep everything even if you don't use them.	WeakMaps holds the reference to the key, not the key itself.
The garbage collector doesn't remove a key pointer from "Map" and also doesn't remove the key from memory.	The garbage collector goes ahead and removes the key pointer from "WeakMap" and also removes the key from memory. WeakMap allows the garbage collector to do its task but not the Map.
Maps have some properties : .set, .get, .delete, .size, .has, .forEach, Iterators.	WeakMaps have some properties : .set, .get, .delete, .has.
You can create a new map by using a **new Map()**.	You can create a new WeakMap by using a **new WeakMap()**.

Bootstrap

3
4

javascript object

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.
6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.
8	Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.
15	Object.isSealed()	This method determines if an object is sealed.
16	Object.keys()	This method returns an array of a given object's own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.
18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.

20	Object.values()	This method returns an array of values.

Project

1. The Smartworks project focuses on creating an iOT enabled app to improve office experiences for businesses in India. The app aims to keep users connected to their office community wherever they are. It makes office smart. Used RFID card reading technology too.
- The FR KIOSK project is all about a smart system which is iOT Enabled app that lets users book office seats using facial recognition technology. Here's a simpler breakdown:
- **Facial Recognition**: Users can book seats by scanning their faces.
- **Automatic Power Control**: Once a user is identified, the power to their seat turns on automatically.
- RFID Reader gate:- confirm user by verified by RFID cards and sends user details to admin. And also provides the access to the user for lifts too, that enhance the efficiency and security of the system.
- The JoyScore app is all about helping users feel happier by doing Custom build activities and exercises(अभ्यास). twilio

The Talk2MeMore app helps speech and language therapists monitor a child's language skills. And also monitor the health of the child by tracking the heartbeat

- **Automating Tasks**: Finding ways to make building, testing, and updating the app faster and more reliable.
- **Listening to Clients**: Talking to therapists to understand what they need from the app and making sure those needs are met.
- **Connecting to Other Apps**: Making the app work with other services like Firebase for sending messages to therapists.

The School Appz project aims to create user-friendly apps for different people involved in schools, like principals, teachers, parents, and visitors.
The Cellway Online Platform project is about creating an app where people can buy used smartphones, tablets, and accessories.
The ParkSmart Driver App project is about creating a web application where users can view and pay for parking charges.
I have started my career from Tech pathway that was the startup, I worked for 1 and half year there, due to fund crunches tech pathway got ishut down,
And after that I started working on Hash Studio, I worked there for half year and they pressurised me to come to office when I was covid positive, when I said so, they asked to resign,
After that I joined QToss, and they have provided me to Smart works and because of my hard work, after one year, Smartwork hired me in its payroll. So overall I worked for 2 and half years in smartwork.
If you see my resume it’s looks like I worked for 4 companies but technically I worked for 3 companies only within 4 and half years of my experience.
On 15 December my father got second heart attack, when we admitted father, doctors checked and suggested for heart valve change and also suggested 1 month bed rest, so I was there to take care of my father, my company didn’t provided me that much time for leaves and also not agreed with Work from home due to having iOT projects., that’s the reason I left my company.

Whenever I start to works or implement any new work for first time, it always like the tough work for me, but when I done with work or used to do the same work, it becomes easy for me. New learning is always tough but it also excite me to improve my self and give me lots of learning.
If I will have designs and APIs, it will take 1 month, if project is big then only it will take 2 months Max to build

ProjectsssSAP UI5

Smart Automotive Solutions
Service Request Management
Aims to streamline service request management by developing a SAP Fiori application
for customers to submit requests, track status, and communicate with technicians. explain project in SAP UI5 and implentation

Implementing a Smart Automotive Solutions Service Request Management application using SAPUI5 involves designing and developing a SAP Fiori application that enables customers to submit service requests, track their status, and communicate with technicians. Here's a detailed breakdown of the project:

- **Requirements Gathering**:
  - Understand the requirements of the Service Request Management application, including the types of service requests, user roles (customers, technicians), and desired functionality.
  - Identify key features such as request submission, status tracking, notifications, and messaging.
- **Design Phase**:
  - Design the user interface of the Fiori application using SAPUI5, including wireframes, mockups, and UI prototypes.
  - Define the application architecture, including views, controllers, models, and services required to implement the desired functionality.
  - Design the data model to represent service requests, customer information, technician assignments, and status updates.
- **Development Phase**:
  - Set up a new SAPUI5 project in SAP Web IDE or any other development environment.
  - Implement views and controllers for the various screens of the Fiori application, such as the service request submission form, request status overview, and messaging interface.
  - Implement data models to represent service request data and customer information using JSON models or OData models.
  - Implement data binding to connect UI elements with data models, ensuring that changes in the data are reflected in the UI.
  - Implement routing and navigation to enable seamless navigation between different screens of the application.
  - Implement features such as form validation, error handling, and user authentication to enhance the usability and security of the application.
- **Integration Phase**:
  - Integrate the Fiori application with backend systems, such as SAP ERP or a custom service management system, to fetch and update service request data.
  - Implement communication channels for customers to submit service requests and receive status updates, such as email notifications or push notifications.
- **Testing and Validation**:

- Write unit tests and integration tests to validate the functionality, performance, and reliability of the Fiori application.
- Conduct end-to-end testing to ensure seamless integration between the frontend SAPUI5 application and the backend systems.
- Perform user acceptance testing (UAT) with real users to gather feedback and identify any usability issues or bugs.

- **Deployment and Monitoring**:
  - Deploy the Fiori application to a production environment, such as the SAP Fiori Launchpad or a standalone web server.
  - Monitor the performance, availability, and security of the application in production, and address any issues or incidents as they arise.
- **Documentation and Training**:
  - Document the architecture, design decisions, implementation details, and user guides for the Service Request Management application.
  - Provide training sessions to customers and technicians to familiarize them with the functionality and usage of the Fiori application.
- **Request Submission**:
  - Utilize SAPUI5 controls like sap.m.Input, sap.m.Select, or sap.m.DatePicker to collect user input for request details.
  - Implement event handlers for UI actions like pressing a submit button.
  - Use SAPUI5's model and binding features to bind form inputs to a data model.
  - Use AJAX requests or SAPUI5's sap.ui.model.odata.v2.ODataModel for sending data to the backend server for processing.
- **Status Tracking**:
  - Display the status of submitted requests using SAPUI5 controls like sap.m.ObjectStatus or sap.m.Label.
  - Retrieve the status information from the backend using data binding or AJAX requests.
  - Implement logic to dynamically update the UI based on changes in the status data.
- **Notifications**:
  - Use SAPUI5's built-in controls like sap.m.MessageToast or sap.m.MessageBox to display notifications to the user.
  - Show success messages upon successful request submission or completion.
  - Display error messages in case of failed requests or validation errors.
- **Messaging**:
  - Implement messaging functionalities using SAPUI5 controls like sap.m.FeedInput for direct messaging features within the application.
  - For more complex messaging features, integrate with SAP messaging services or backend systems.
  - Implement real-time messaging using WebSocket communication, if required.

Setting up push notification service in SAP Cloud Platform (SCP) involves several steps. Below is a general outline of the process:

- **Access SAP Cloud Platform Cockpit**:
  - Log in to the SAP Cloud Platform Cockpit using your credentials.
- **Navigate to Services**:
  - Once logged in, navigate to the "Services" section of the SAP Cloud Platform Cockpit.
- **Select Push Notification Service**:

- Find and select the "Push Notification" service from the list of available services. If it's not already available, you may need to subscribe to the service.
- **Create a New Instance**:
- Click on the "Create" button to create a new instance of the Push Notification service.
- **Configure Service Instance**:
- Provide a name and description for the service instance.
- Select the appropriate service plan based on your requirements (e.g., trial, standard).
- Configure any additional settings or options as needed.
- **Obtain Credentials**:
- Once the service instance is created, you'll typically receive credentials (such as API keys or tokens) that you'll use to authenticate your application when interacting with the push notification service.
- **Access Push Notification Dashboard**:
- Navigate to the dashboard or administration console of the Push Notification service within the SAP Cloud Platform Cockpit.
- Here, you can manage various aspects of the push notification service, including configuring push notification channels, defining notification templates, and monitoring push notification activity.
- **Integrate Push Notification Service with Your Application**:
- In your application code (whether it's a SAPUI5 application or another type of application), integrate the push notification service by using the provided SDKs, APIs, or libraries.
- Use the obtained credentials to authenticate your application when interacting with the push notification service.
- Implement logic for registering devices for push notifications, handling push notification events, and sending notifications to registered devices.
- **Test and Deploy**:
- Test the push notification functionality thoroughly to ensure that devices are successfully registered and notifications are delivered as expected.
- Deploy your application with push notification integration to your development, testing, or production environment as appropriate.
- **Monitor and Maintain**:
- Monitor the performance and reliability of push notifications in your application.
- Handle any issues or errors that may arise during push notification delivery.
- Keep the push notification functionality up-to-date with any changes or updates to the SAP Cloud Platform services or your application requirements.