

You

Let's Explore All Real Time Projects based QnA for all kinds of SPFX APPS & SERVICES ON 2016,2019 MODERN ENVIRONMENT at all Levels: Admin, Development, Designing, Migration, Branding level for Super Senior Architect Consultant level IN A TABULAR REPRESENTATION MENTIONING ALL KINDS OF POINTS SUMMARY !!

Copilot

Certainly! Let's delve into the world of SharePoint Framework (SPFx) and explore its applications in various environments. Here's a concise summary of SPFx development considerations for SharePoint Server 2016, SharePoint Server 2019, and SharePoint Server Subscription Edition (SE):

1. SharePoint Framework (SPFx) Overview:

- SPFx is a client-side development framework for building custom solutions in SharePoint.
- It supports client-side web parts and extensions (such as application customizers and field customizers).

2. Environment-Specific Considerations:

- **SharePoint Online (SPO):** Always uses the latest version of SPFx.
- **SharePoint Server 2016:** Supports SPFx v1.1.0.
- **SharePoint Server 2019 and Subscription Edition (SE):** Supports the version that matches server-side dependencies (currently SPFx v1.4.1).

3. Development Environment Setup:

- Ensure you have Node.js installed (SPFx v1.4.1 supports Node.js v6 and v8).
- Use Yeoman and the SharePoint Framework Yeoman generator to scaffold solutions.
- Internet connectivity is required for npm package access during development.

4. Key Points for Different Levels:

- **Admins:** Configure App Management Service and App Catalog sites.
- **Developers:** Leverage SPFx for custom web parts and extensions.
- **Designers:** Customize SPFx components to match branding.
- **Migration:** Consider SPFx compatibility during SharePoint migrations.
- **Branding:** Use SPFx to enhance user experience.

5. Tabular Representation (Summary):

Level	Considerations
Admin	- Configure App Management Service and App Catalog - Set up environment for apps
Development	- Use SPFx for web parts and extensions
Designing	- Customize SPFx components for branding
Migration	- Ensure SPFx compatibility during migrations
Branding	- Enhance user experience with SPFx
Architect	- Align SPFx solutions with overall architecture

Remember, SPFx evolves, so staying informed about updates and best practices is essential. If you have specific questions or need further details, feel free to ask! 😊🚀

You

Can you provide an example of a real-world SPFx project?

Copilot

Certainly! Here's an example of a real-world SharePoint Framework (SPFx) project:

- **CRUD Web Part with ReactJS:**

- GitHub Repository: [vccampbell/spfx-reactjs](https://github.com/vccampbell/spfx-reactjs)
- Description: This project demonstrates a complete CRUD (Create, Read, Update, Delete) functionality using ReactJS and the SharePoint Framework. It's based on a real-world implementation, making it a practical example for SPFx development.

Additionally, you can explore more community-contributed SPFx samples on the [SharePoint Framework Samples](#) page. These samples cover various scenarios and frameworks, providing valuable insights for your own projects. Feel free to dive in and explore! 😊🚀

You

How does SharePoint Framework compare to classic solutions?

Copilot

Certainly! Let's explore the key differences between **SharePoint Framework (SPFx)** and classic SharePoint solutions:

1. **SharePoint Framework (SPFx):**

- **Model:** SPFx is a **page and web part model** designed for **client-side SharePoint development**.
- **Integration:** It seamlessly integrates with **modern web technologies**, making it accessible to developers.
- **Customization Depth:** While SPFx offers customization, it doesn't provide the same **depth of customization** as classic sites.
- **Responsive and Mobile-Ready:** SPFx solutions are **responsive and mobile-ready** by default.
- **Recommended Model:** SPFx is the **recommended customization and extensibility model** for developers in SharePoint Online, Microsoft Teams, and Microsoft Viva Connections.

2. **Classic SharePoint:**

- **Customization Options:** Classic sites support **basic page construction** using wiki-style text editing and built-in web parts.
- **Sophisticated Functionality:** For more sophisticated CMS-style functionality, classic sites rely on the **publishing framework** with field-based controls and structured page editing.

In summary, SPFx is the modern approach, while classic SharePoint provides extensive customization capabilities but requires deeper technical knowledge. Choose based on your project requirements! 😊🚀

You

Can you provide an example of a real-world SPFx project?

Copilot

Certainly! Here's a real-world example of a SharePoint Framework (SPFx) project:

- **CRUD Web Part with ReactJS:**

- GitHub Repository: [vccampbell/spfx-reactjs](https://github.com/vccampbell/spfx-reactjs)
- Description: This project demonstrates a complete **CRUD (Create, Read, Update, Delete)** functionality using **ReactJS** and the SharePoint Framework. It's based on a real-world implementation, making it a practical example for SPFx development.

Additionally, you can explore more community-contributed SPFx samples on the [SharePoint Framework Samples](#) page. These samples cover various scenarios and frameworks, providing valuable insights for your own projects.

Feel free to dive in and explore! 😊🚀

You

Tell me more about ReactJS in SPFx.

Copilot

Certainly! **SharePoint Framework (SPFx)** and **React** can be seamlessly integrated, allowing you to create responsive and dynamic web elements within SharePoint. Here's how to get started:

1. Set Up Your Development Environment:

- Ensure you have **Node.js** and **Yeoman** installed.
- Run the following commands in your terminal:

```
npm install -g yo gulp  
npm install -g @microsoft/generator-sharepoint
```

- Scaffold a new SPFx project using:

```
yo @microsoft/sharepoint
```

- Choose the **React framework** when prompted.

2. Trust the Self-Signed Developer Certificate:

- Execute this command from the project root folder:

```
gulp trust-dev-cert
```

3. Build Your React Components:

- Create React components within the “**src**” directory.
- Organize them in a “**web parts**” folder based on the specific web parts they belong to.
- Customize or create new components to suit your project's requirements.
- Start with a default class component and consider converting it into a functional component for simplicity and efficiency.

Remember, SPFx with React empowers developers to build efficient, maintainable web parts that seamlessly integrate with SharePoint Online. Explore further by diving into React hooks like **useState** and **useEffect** for enhanced functionality! 😊🚀

For more details, you can refer to the [SPFx with React guide](#). 📖