You

TOTAL INTERVIEW QNAs FOR SPFX WITH TYPESCRIPT JS DEVELOPMENT, CUSTOMIZATION, WORKFLOWS, PLUGINS, HTML,CSS,XML JS & UNIFIED INTERFACE INTEGRATION IN A TABULAR REPRESENTATION WITH ALL FEATURES MENTIONED POINT WISE, BEST PRACTICES, HOW TO OPTIMISE IT, PROS & CONS, WHAT'S LATEST ADVANCEMENTS !!

Copilot

Certainly! Here's a concise tabular representation of SharePoint Framework (SPFx) interview questions and answers related to TypeScript, JavaScript development, customization, workflows, plugins, HTML, CSS, XML, and Unified Interface integration:

Question	Answer
1. New Delegate Controls in SharePoint	Three new delegate controls have been introduced for displaying the top suite bar (with links to SkyDrive, NewsFeed, Sync, and follow):1. SuiteBarBrandingDelegate2. SuiteLinksDelegate3. PromotedActionsDelegate
2. Changes in CSOM and REST-based APIs	Microsoft has improved both Client Side Object Model (CSOM) and Representational State Transfer (REST) based APIs by adding support for Search, User Profiles, Taxonomies, and Publishing Object Model. The `Client.svc` service now accepts HTTP GET, PUT, and POST requests.
3. App Permissions in SharePoint	An app for SharePoint requests permissions during installation from the user who installs it. Developers specify the required permissions in the app manifest file.
4. New SPSecurityEventReceiver	SharePoint introduces a Cloud App Model that allows creating self-contained apps. These apps can include SharePoint components (lists, workflows, site pages) as well as surface remote web applications and data in SharePoint.
5. Base Classes and Benefits	Two base classes are available:1. `Microsoft.SharePoint.WebPartPages.WebPart` (for backward compatibility)2. SharePoint WebPart base class (supports cross-page connections and client-side connections)

Remember that these answers provide a high-level overview. For more in-depth details, consider referring to relevant documentation or further resources . If you have any specific questions, feel free to ask! 😊

You

INCLUDE TYPESCRIPT TOP INTERVIEW QNAS EXPLAINED IN TABULAR BASED COMPARION WITH ALL FEATURES MENTIONED, PROS & CONS, WHAT'S LATEST ADVANCEMENTS, REAL TIME APPLICATIONS USED !!

Copilot

Certainly! Here's a concise tabular representation of TypeScript interview questions and answers, along with features, pros, cons, latest advancements, and real-time applications:

Question	Answer
1. What is TypeScript, and how does it differ from JavaScript?	TypeScript is a statically typed superset of JavaScript that compiles down to plain JavaScript. It adds static types, enhancing tooling (autocompletion, navigation, and refactoring), making code easier to read and debug. Key differences include static typing, support for classes and interfaces, compile-time errors, and powerful tooling.
2. What are the main components of TypeScript?	Three main components:1. **Language**: Includes types, annotations, classes, interfaces, and more.2. **TypeScript Compiler (tsc)**: Transpiles .ts files to .js files, catching errors at compile time and enforcing type-checking.3. **TypeScript Language Service**: Provides editor features like autocompletion and refactoring.
3. What are interfaces in TypeScript?	Interfaces define contracts for object shapes. They allow you to specify the structure of an object, including properties, methods, and their types. Interfaces are useful for achieving type-checking and ensuring consistency in code.
4. Explain the difference between `interface` and `type` in TypeScript.	Both define shapes, but:1. **Interface**: Can be extended and merged, ideal for defining object structures.2. **Type**: Supports union types, intersections, and mapped types, suitable for creating complex types. Use `interface` for objects and `type` for unions, intersections, and aliases.
5. What is a decorator in TypeScript?	Decorators are a TypeScript feature inspired by Python and Java annotations. They allow you to modify or annotate classes, methods, properties, or parameters. Common decorators include `@Component`, `@NgModule`, and `@ViewChild`.
6. How does TypeScript handle null and undefined values?	TypeScript has strict null checks. You can use union types (`string
7. What is a generic type in TypeScript?	Generics allow writing reusable code that works with different data types. They provide type safety while maintaining flexibility. Examples include `Array<T>`, `Promise<T>`, and custom generic functions.
8. Explain the `readonly` modifier in TypeScript.	`readonly` makes properties immutable after initialization. It prevents accidental modification and ensures data integrity. Use it for constants, configuration, or shared state.
9. How do you handle asynchronous operations in TypeScript?	TypeScript supports Promises, async/await, and Observables. Promises handle single asynchronous operations, async/await simplifies asynchronous code, and Observables manage streams of data.
10. What are namespaces in TypeScript?	Namespaces (previously known as internal modules) provide a way to organize code into logical containers. They prevent naming conflicts and improve code organization. However, consider using ES6 modules instead.

Remember that these answers provide a high-level overview. For more in-depth details, consider referring to relevant documentation . If you have any specific questions or need further clarification, feel free to ask! 😊

You

What are some real-world applications of TypeScript?

Copilot

Certainly! TypeScript is widely used in various real-world scenarios due to its benefits over plain JavaScript. Here are some common applications:

1. **Web Development**:
    - **Angular**: Angular, a popular front-end framework, is built using TypeScript. It provides strong typing, dependency injection, and component-based architecture.
    - **React with TypeScript**: Many React projects adopt TypeScript for better type safety and improved development experience.
    - **Vue.js with TypeScript**: Vue.js developers can leverage TypeScript for enhanced code quality and maintainability.
2. **Node.js Applications**:
    - TypeScript is increasingly used for server-side development with Node.js. It allows developers to write scalable, maintainable code using modules, classes, and interfaces.
3. **Cross-Platform Mobile Apps**:
    - **Ionic**: The Ionic framework, which builds cross-platform mobile apps, supports TypeScript. Developers benefit from type checking and better tooling.
    - **NativeScript**: NativeScript, another cross-platform framework, integrates well with TypeScript for building native mobile apps.
4. **Desktop Applications**:
    - **Electron**: Electron, used for creating desktop applications, supports TypeScript. Developers can build cross-platform apps with a consistent codebase.
5. **Game Development**:
    - **Phaser**: Phaser, a popular HTML5 game framework, allows game developers to use TypeScript for better organization and type safety.
6. **Backend Services and APIs**:
    - Many RESTful APIs and microservices are written in TypeScript. It ensures robustness and maintainability.
7. **Tooling and Build Systems**:
    - **Webpack**: TypeScript integrates seamlessly with Webpack for bundling and optimizing code.
    - **Gulp and Grunt**: TypeScript can be part of build pipelines using Gulp or Grunt.
8. **Enterprise Applications**:
    - Large-scale enterprise applications benefit from TypeScript's static typing, interfaces, and better code organization.
9. **Testing and Test Automation**:
    - TypeScript is used for writing unit tests, end-to-end tests, and test automation scripts.
10. **Migration from JavaScript**:
    - Existing JavaScript projects can gradually adopt TypeScript by rewriting parts of the codebase. This migration path is common in real-world scenarios.

Remember that TypeScript’s adoption continues to grow, and its applications extend beyond these examples. Developers appreciate its developer experience, tooling, and safety features. 😊

You

What are the pros and cons of using TypeScript?

Copilot

Certainly! Let’s explore the pros and cons of using TypeScript:

Pros (Advantages):

1. **Static Typing**:
    - TypeScript adds static typing to JavaScript, catching type-related errors at compile time. This improves code quality and reduces runtime surprises.
2. **Enhanced Tooling and IDE Support**:
    - TypeScript provides better tooling, including autocompletion, navigation, and refactoring features in modern IDEs (such as Visual Studio Code).
    - Developers benefit from a more productive development experience.
3. **Code Maintainability**:
    - Strongly typed code is easier to understand and maintain.
    - Interfaces and type annotations improve code documentation.
4. **Type Safety**:
    - TypeScript helps prevent common runtime errors by enforcing type checks.
    - It catches issues like null/undefined values and incorrect function calls.
5. **ECMAScript Compatibility**:
    - TypeScript follows ECMAScript standards and can transpile to different ECMAScript versions (ES5, ES6, etc.).
    - Developers can use modern language features while targeting older browsers.
6. **Gradual Adoption**:
    - Existing JavaScript projects can gradually adopt TypeScript by rewriting parts of the codebase.
    - TypeScript files (.ts) can coexist with JavaScript files (.js).

Cons (Challenges):

1. **Learning Curve**:
    - Developers familiar with JavaScript may need time to learn TypeScript’s additional features (such as interfaces, decorators, and type annotations).
2. **Build Process Overhead**:
    - TypeScript requires a build step to transpile .ts files to .js files.
    - This adds complexity to the development workflow.
3. **Increased File Size**:
    - Transpiled TypeScript code can be larger than handwritten JavaScript due to type annotations.

  - However, this impact is usually negligible in practice.

4. **Compatibility with Third-Party Libraries**:
   - Some third-party libraries lack TypeScript typings.
   - Developers may need to write custom type definitions or use “any” type for such libraries.

5. **Tooling and Configuration**:
   - Setting up TypeScript configuration (tsconfig.json) correctly can be challenging.
   - Choosing the right compiler options is crucial for optimal development.

6. **Not a Silver Bullet**:
   - TypeScript doesn’t solve all problems. It’s essential to understand its limitations and use it judiciously.

In summary, TypeScript offers significant benefits in terms of type safety, tooling, and maintainability. However, developers should weigh these advantages against the learning curve and build process overhead. 😊

You

Tell me more about TypeScript’s latest advancements.

Copilot

Certainly! Here are some of the recent advancements in TypeScript:

1. **TypeScript 5.5** (June 2024):
   - **Inferred Type Predicates**: TypeScript now automatically infers type predicates for functions that return boolean values. This helps narrow types based on the function’s return type.
   - **Control Flow Narrowing for Indexed Accesses**: Improved type narrowing for constant indexed property accesses enhances type checking precision.
   - **JSDoc @import Tag**: A new JSDoc tag simplifies importing types in JavaScript projects, making it easier to integrate TypeScript’s type checking capabilities.
   - **Regular Expression Syntax Checking**: TypeScript now validates regular expressions for correct syntax, reducing runtime errors.
   - **New ECMAScript Set Methods**: TypeScript supports new ECMAScript methods (union, intersection, and difference) for Set objects, aligning with the latest standards.
   - **Isolated Declarations**: Developers can use the `--isolatedDeclarations` option to generate declaration files without a full type check, speeding up builds.
   - **${configDir} Template Variable**: Provides flexibility in specifying paths within configuration files.
   - **Editor and Watch-Mode Improvements**: Enhancements for reliability and performance in editor integrations and watch mode.
   - **Performance and Size Optimizations**: Various optimizations improve compiler performance and reduce generated output size.

2. **TypeScript 5.4** (March 2024):
   - **Preserved Narrowing in Closures**: Ensures type narrowing in closures following last assignments.
   - **NoInfer Utility Type**: Helps prevent inference of certain types.
   - **Object.groupBy and Map.groupBy**: New utility types for grouping objects and maps.
   - **Support for require() calls in bundler and --module preserve**: Enhances module resolution.
   - **Checked Import Attributes and Assertions**: Improves type checking.
   - **Quick Fix for Adding Missing Parameters**: Helps catch missing function parameters.

3. **TypeScript 5.1** (June 2023):
   - Introduced various enhancements, including improved type checking and better tooling.

Remember that TypeScript continues to evolve, addressing JavaScript limitations while maintaining cross-platform compatibility. 😊