**W09 Prepare – Q/A AI**

**JavaScript Objects**

**Q: What are objects used for in JavaScript?**

A: Objects serve two main purposes:

1. Organizing data with properties (key-value pairs).
2. Encapsulating data and related functionality using methods (functions attached to the object).

**Q: How do I create objects in JavaScript?**

A: There are two main ways:

1. Object literals (curly braces {} for key-value pairs).
2. Constructor functions (blueprints for creating objects with specific properties and methods).

**Q: What are examples of well-made objects?**

A: Well-made objects are:

* Clear and concise with descriptive property names.
* Reusable across different parts of your code.
* Encapsulated, keeping data and functionality together.
* Modular, promoting code organization and maintainability.

**Q: What are examples of poorly made objects?**

A: Poorly made objects include:

* God objects (trying to do too much).
* Objects with magic numbers or strings (unclear variable names).
* Objects mixing concerns (e.g., data storage and presentation logic).
* Reliance on global variables instead of using objects for data encapsulation.
* Mutable objects by default (consider immutability for data integrity).

**Q: How can I access properties from an object?**

A: There are two main ways:

1. Dot notation (. after object name followed by property name) - preferred for known valid property names.
2. Bracket notation ([] after object name with property name as a string) - useful for dynamic property access or names with special characters.

**ES Modules**

**Q: What are ES modules?**

A: ES modules, also known as ECMAScript modules, are the standardized way for structuring JavaScript code into reusable modules. Here's a breakdown of their key features:

* Organize code into self-contained units that can be imported and exported
* Promote code reusability and modularity
* Modules are defined using files (.js or .mjs extensions)

Overall, ES modules are the recommended approach for structuring JavaScript code, offering a clean, standardized, and well-supported way to manage your project's modularity.

**Q: What are the advantages of ES modules for organizing code?**

A: ES modules offer several benefits:

* Improved modularity with clear separation and boundaries between code units.
* Enhanced reusability by making modules easily importable and usable in different parts of your application.
* Efficient loading and bundling through lazy loading and tree shaking techniques.
* Standardized approach with modern browser and Node.js compatibility.

**Q: What are the drawbacks of ES modules compared to a single JavaScript file?**

A: ES modules have some downsides:

* Increased complexity due to managing multiple files and potentially using build tools.
* Potential request overhead from fetching separate modules, impacting initial page load times.
* Learning curve for understanding the import/export syntax and potentially configuring bundlers.
* Might be overkill for very small projects where a single file suffices.

**Q: On average, how many ES modules are used in a JavaScript game?**

A: There's no single answer as it depends on factors like:

* Game complexity (larger games use more modules).
* Developer preference (some prefer many small modules, others use fewer larger ones).
* Use of libraries/frameworks (pre-built modules might be used).

A rough estimate suggests:

* Dozens to a few hundred modules for smaller games.
* A few hundred to a thousand modules for medium-sized games.
* Thousands or more modules for large and complex games.

The key isn't the exact number, but how well the modules are organized for reusability and maintainability.

**Q: Can you summarize the key points about ECMAScript?**

A: ECMAScript (ES) is the standardization body for scripting languages, not a specific language itself.

Key points:

* Ensures compatibility of web pages using scripting languages across browsers.
* Maintained by Ecma International (ECMA) through the ECMA-262 specification.
* JavaScript is the most well-known implementation following the ES standard for syntax, core features, and functionality.

Benefits of standardization:

* Consistent behavior of websites using scripting languages across browsers.
* Consistent development experience using standard features.
* Ability for languages to benefit from advancements in the ES standard.

In essence, ECMAScript provides the foundation upon which JavaScript and other languages build their functionalities.