ES6 Expanded Interview Questions (Highly Detailed)

Q1. What is the difference between var, let, and const?

(Already included previously — keeping for continuity)

Q2. What are Arrow Functions?

(Already included previously)

Q3. Explain Promises with detailed real-world example.

(Already included previously)

Q4. What is async/await?

(Already included previously)

NEW INTERVIEW QUESTIONS (Highly Detailed)

Q5. What are Symbols in ES6? Why are they used?

Symbols are a new primitive datatype introduced in ES6.

They create unique values that cannot clash with other object keys.

Why use Symbols?

- Avoid key collisions in objects
- Create private-like object properties
- Used internally by JavaScript for built-in behaviors (e.g., iterators)

Example:

```
const id1 = Symbol("id");

const id2 = Symbol("id");

console.log(id1 === id2); // false (always unique)
```

Practical Use:

```
const user = {

name: "Pushpa",

[Symbol("secret")]: "This is hidden"

};
```

Symbols do not appear in normal loops like for...in.

Q6. What is the purpose of the Spread (...) operator? Explain in detail.

The spread operator EXPANDS arrays/objects into individual elements.

Uses:

1. Copy arrays/objects
2. Merge arrays/objects
3. Pass array as arguments

Examples:

Example 1: Copy array

```
const nums = [1,2,3];

const copy = [...nums];
```

Example 2: Merge arrays

```
const a = [1,2];

const b = [3,4];

const c = [...a, ...b];
```

Example 3: Pass array to function

```
function sum(a,b,c){ return a+b+c; }
let arr = [1,2,3];
sum(...arr);
```

Q7. What is Rest Parameter? How is it different from Spread Operator?

Rest parameter COLLECTS arguments into an array.

Spread operator EXPANDS arrays.

Example:

```
function total(...nums){
return nums.reduce((a,b)=>a+b);
}
total(1,2,3,4); // 10
```

Rest = collect

Spread = expand

Q8. What are ES6 Modules? Why are they better?

Modules allow JavaScript code to be split into files.

Benefits:

- Reusability
- Cleaner code
- Maintains scope
- Supports import/export

Example:

file1.js

```
export const x = 10;
```

file2.js

```
import { x } from "./file1.js";
```

Q9. What is a Generator function? Explain with detailed example.

Generators are functions that can PAUSE and RESUME using `yield`.

Syntax:

```
function* generator(){}
```

Example:

```
function* counter(){
yield 1;
yield 2;
yield 3;
}

const g = counter();
g.next(); // { value:1, done:false }
g.next(); // { value:2, done:false }
```

Real-World Use:

Useful for:

- Lazy loading
- Infinite sequences
- Handling async tasks before promises existed

Q10. What is the difference between Map and Object?

Map:

- Stores key-value pairs
- Keys can be ANY type
- Maintains insertion order
- Easier to get size

Object:

- Keys are strings or symbols
- Prototype chain overhead

Example:

```
const map = new Map();
map.set("name", "Pushpa");
map.get("name");
```

Q11. What is Set? How is it different from Array?

Set stores UNIQUE values.

Example:

```
const s = new Set([1,2,2,3]);
// Set => {1,2,3}
```

Benefits over array:

- No duplicates
- Fast search
- Useful for uniqueness checking

Example:

```
[...new Set([1,2,3,2,1])] // [1,2,3]
```

Q12. Explain the 'this' keyword behavior in ES6 and how Arrow Functions changed it.

In normal functions:

`this` depends on the caller.

In arrow functions:

`this` is lexically scoped — it uses the surrounding scope.

Example:

```
const obj = {
value: 20,
show() {
setTimeout(function(){
console.log(this.value); // undefined
}, 1000);
}
};
```

Arrow version:

```
setTimeout(()=>{
console.log(this.value); // works
}, 1000);
```

Q13. What is the difference between for...of and for...in?

for...of → loops through values

for...in → loops through keys

Example:

```
let arr = ["a", "b"];

for (let x of arr) console.log(x); // values
for (let y in arr) console.log(y); // indexes
```

Q14. Explain Default Parameters with detailed examples.

Example 1:

```
function greet(name="Guest"){
console.log("Hello " + name);
}
```

Example 2:

```
function sum(a=5, b=a*2){
return a + b;
}
sum(); // 5 + 10 = 15
```

Q15. What are Enhanced Object Literals in ES6?

They allow:

1. Property shorthand
2. Method shorthand
3. Computed keys

Example:

```
const name = "Pushpa";
const obj = {
name,
```

```
greet(){ return "Hi"; },
["id_" + 1]: 101
};
```
