JavaScript ES6/ ES2015

I. Introduction

ES6/ ES2015 is a significant update to the version of ECMAScript/JavaScript, and the first major update to the language since ES5 was standardized in 2009. ES6 added class and modules and partially supported in all modern browser.

II. Features

1. Arrow and Lexical This: Arrows are a function shorthand using the **=>** syntax. Arrows share the same lexical **this** as their surrounding code. If an arrow is inside another function, it shares the "arguments" variable of its parent function.

```
// Expression bodies
var odds = evens.map(v => v + 1);
var nums = evens.map((v, i) => v + i);

// Statement bodies
nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v);
});

// Lexical this
var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
      console.log(this._name + " knows " + f));
  }
};

// Lexical arguments
function square() {
  let example = () => {
    let numbers = [];
    for (let number of arguments) {
      numbers.push(number * number);
    }

    return numbers;
  };

  return example();
}

square(2, 4, 7.5, 8, 11.5, 21); // returns: [4, 16, 56.25, 64, 132.25, 441]
```

2. Classes: Classes support prototype-based inheritance, super calls, instance and static methods and constructors.

3. Enhanced Object Literals: Object literals are extended to support setting the prototype at construction, shorthand for foo: foo assignments, defining methods and making super calls.
4. Template Strings: Template strings provide syntactic sugar for constructing strings.
5. Destruction: Destructuring allows binding using pattern matching, with support for matching arrays and objects. Destructuring is fail-soft, similar to standard object lookup foo["bar"], producing undefined values when not found.

6. Default +Rest+ Spread: Callee-evaluated default parameter values. Turn an array into consecutive arguments in a function call. Bind trailing parameters to an array. Rest replaces the need for arguments and addresses common cases more directly.

7. Let+ Const: Block-scoped binding constructs. let is the new var. const is single-assignment. Static restrictions prevent use before assignment.

8. Iterators + For..Of: Iterator objects enable custom iteration like CLR IEnumerable or Java Iterable. Generalize for..in to custom iterator-based iteration with for..of.

9. Generator: Generators simplify iterator-authoring using function* and yield.
10. Unicode: Non-breaking additions to support full Unicode, including new unicode literal form in strings and new RegExp u mode to handle code points, as well as new APIs to process strings at the 21bit code points level.
11. Modules:

```
// lib/math.js
export function sum(x, y) {
  return x + y;
}
export var pi = 3.141593;
```

12. Map+ Set+ WeakMap+ WeakSet

```
// Sets
var s = new Set();
s.add("hello").add("goodbye").add("hello");
s.size === 2;
s.has("hello") === true;

// Maps
var m = new Map();
m.set("hello", 42);
m.set(s, 34);
m.get(s) == 34;

// Weak Maps
var wm = new WeakMap();
wm.set(s, { extra: 42 });
wm.size === undefined

// Weak Sets
var ws = new WeakSet();
ws.add({ data: 42 });
// Because the added object has no other references, it will not be held in the set
```

13. Proxies: Proxies enable creation of objects with the full range of behaviors available to host objects.

```
// Proxying a normal object
var target = {};
var handler = {
  get: function (receiver, name) {
    return `Hello, ${name}!`;
  }
};

var p = new Proxy(target, handler);
p.world === "Hello, world!";
```

14. Symbols: Symbols enable access control for object state.
15. Subclassable Built-ins: In ES2015, built-ins like Array, Date and DOM Elements can be subclassed.
16. Math+ Number+ String+ Object APIs: Many new library additions, including core Math libraries, Array conversion helpers, and Object.assign for copying.