ES6 was the addition of new features in Javascript back in 2015. Although ES6 came with a lot of new features, some of the features easily stood out.

1. Promises
2. Let and Const
3. Arrow function
4. Default Parameters
5. Destructuring (Objects and Arrays)
6. Template literals
7. Modules
8. Spread and rest operator
9. Array in-built methods like map, reduce, filter, sort etc
10. Classes

The spread operator allows us to expand an array or object into its individual elements, while the rest operators allow us to condense multiple elements into a single array or object.

```
const numbers = [1, 2, 3, 4];
const newNumbers = [...numbers];
```

This would create a new array called newNumbers that contains the same values as numbers.

The rest operator is written using three consecutive dots (...) followed by the name of the array that will contain the elements. For example, to condense multiple elements into a single array, you would write:

```
function sum(...numbers) {
  return numbers.reduce((a, b) => a + b, 0);
}
```

This would create a function called sum that takes any number of arguments and returns the sum of those arguments.

Destructuring assignment is a JavaScript expression that allows us to unpack values from arrays, or properties from objects, into distinct variables.

```
// Array destructuring
const numbers = [1, 2, 3, 4, 5];
const [first, second, ...rest] = numbers;

console.log(first); // 1
console.log(second); // 2
console.log(rest); // [3, 4, 5]

// Object destructuring
const user = {
  name: "John Doe",
  age: 30,
  occupation: "Software Engineer"
};

const { name, age, occupation } = user;

console.log(name); // "John Doe"
console.log(age); // 30
console.log(occupation); // "Software Engineer"
```

Destructuring can also be used to assign values to nested properties.

```
const nestedObject = {
  user: {
    name: "John Doe",
    age: 30,
    occupation: "Software Engineer"
  }
};

const { user: { name, age, occupation } } = nestedObject;

console.log(name); // "John Doe"
console.log(age); // 30
console.log(occupation); // "Software Engineer"
```

Default parameters in JavaScript are a way to set default values for function parameters if no value is passed in. This can be useful for preventing errors when a function is called without all of the required parameters.

```
function greet(name = "World") {
  console.log(`Hello, ${name}!`);
}
```

Template literals are string literals that can contain embedded expressions. This allows you to create more complex and dynamic strings. Template literals are enclosed in backticks (`) instead of double or single quotes. Template literals can also contain multi-line strings.

```
// Regular string
const message = "This is a multi-line string.\nThis is the second line.";
console.log(message);

// Template literal (using template literals)
const message = `This is a multi-line string.
This is the second line.`;
console.log(message);
```

Template literals can only compute normal JS expressions inside of a string. Don't try to run full blown JS code inside of these template literals like for loop, if-else, switch-case etc.