Introduction to ES6

ES6 (officially ECMAScript 2015) represents a major upgrade to the JavaScript language. It adds several important features and makes it simpler to write error-free code.
As of January 2018, ES6 is extremely well-supported by modern web browsers. The latest versions of Chrome, Safari, Firefox, and Edge all support nearly every feature of ES6 (as well as the 2 new features of ES7).

Keep in mind, the set of specifications created by ECMA is called "ECMAScript"— and JavaScript is an implementation of it. This is why you may hear people refer to JavaScript as "ECMAScript".

Common post-ES5 JavaScript features you should know.

- Variable Assignment
- Destructuring Assignment
- Rest & Spread
- Default Parameter Values
- Block Scope
- Template Literals
- Arrow Functions
- Classes
- Modules

- Variable Assignment

Before ES6 we mainly made use of the var keyword whenever we wanted to declare a variable. But it had some serious issues, also it was not developers favorite so in the ES6 version we were introduced to const and let keywords which allows us to store variables. They both have their own way to storing variables.

1). const:

The const keyword is mainly used to store that variable who's value is not going to be changed.Consider an example where you are creating a web application where you want to store a variable whose value is not going to change, then const is definitely the best choice to store that variable. In javascript, const is considered to be more powerful than var. Once used to store a variable it can't be reassigned. In simple words, it is an immutable variable except when used with objects.

Example:

```
const course = "Introduction to ReactJS";
console.log(course); /* Outputs:  Introduction to ReactJS  */

course = "Introduction to JavaScript";
console.log(course);  /* Will give a TypeError */
```

Note: In the above code we declared a variable course with const keyword and then console.log it which works fine, but reassigning it with some other value will give an error.

2. let

In case of let keyword, the variables declared will be mutable i.e there values can be changed. It works similar to the var keyword with some key differences like scoping which makes it a better option as when compared to var.

```
let course = "Introduction to ReactJS";
console.log(course); /* Outputs:  Introduction to ReactJS  */

course = "Introduction to JavaScript";
console.log(course);   /* Introduction to JavaScript */
```

Find out more about where we declare an object using const and let keyword.

- Object and Array Destructing

Destructing in javascript basically means the breaking down of a complex structure(Objects or arrays) into simpler parts. With the destructing assignment, we can 'unpack' array objects into a bunch of variables.

Example of object Destructing:

```
const college = {
    name : 'DTU',
    established : '1941',
    isPrivate : false
};
let{name,established,isPrivate} = college;
console.log(name); // DTU
console.log(established); // 1941
console.log(isPrivate); // false
```

Try to run the above code and see the output.

- Array destruction

In case of destruction of arrays we can simply replace the curly brackets with square brackets.

Example:

```
const arr = ['lionel','messi','barcelona'];

let[value1,value2,value3] = arr;

console.log(value1); // lionel
console.log(value2); // messi
console.log(value3); // barcelona
```

Try to run the above code and see the output

Template literal

Template literals is a feature of ES6 which allows us to work with strings in a better way as when compared to ES5 and below. By simply using template literals, we can improve the code readability. Before ES6 we made use of '+' operator whenever we wanted to concatenate strings and also when we wanted to use a variable inside a string which is definitely not a recommended method. Template literals use back-ticks(``) rather than the single or double quotes we've used to with regular strings.

Example:

```
const course = 'Introduction to Reactjs';

console.log(`This is  ${course} course`);
```

- Arrow functions

Arrow functions (also known as 'fat arrow functions') are a more concise syntax for writing function expressions. Introduced in ES6, arrow functions are definitely one of the most impactful changes in javascript.

These function expressions makes your code more readable, more modern.

Example:

```
const printName = name =>{
    return `Hi ${name}`; // using template literals
}
console.log(printName('Wanjiku Marine')); // Hi Wanjiku Marine

// We can also write the above code without
// the return statement
const printName1 = name => `Hi ${name}`;

console.log(printName1('Marine')); // Hi Marine
```

- Spread Operator

The Spread operator is another operator provided by ES6 that allows us the privilege to obtain a list of parameters from an array. Consider an example where we have a list of numbers and we return the minimum number from that list and write a program making use of spread operator to implement it.

- Rest Parameter

The rest parameter syntax allows us to represent an indefinite number of arguments as an array. With the help of a rest parameter a function can be called with any number of arguments, no matter how it was defined.

Write three program that makes use of spread and rest operators. Submit via email for review.

Note:

Both the spread and the rest operator makes use of triple dots (...), and sometimes it's hard to differentiate which one is rest or spread. Simply remember that :

- When … is at the end of function parameter, it's rest parameter.
- When … occurs in function call or alike, its called a spread operator.

Default Parameters.

Default parameters allows you to define a parameter in advance, which can be helpful in certain scenarios. In JavaScript, function parameter default to undefined. However it is useful to set a different default value. Before ES6 the way we used to define default parameters is by testing parameters value in the default function body and assign a value if they are undefined.

In ES6 we can implement it as simple as shown in the example below :

Example:

```
function fun(a,b=1){
    return a + b;
}
console.log(fun(2,1)); // 3
console.log(fun(2)); // 3
```

- Classes:

Classes are the core of Object Oriented programming(OOPs). ES6 introduced classes in JavaScript. Classes in JavaScript can be used to create new Objects with the help of constructor, each class can only have one constructor inside it.

```
class Vehicle{
   constructor(name,engine){
      this.name = name;
      this.engine = engine;
   }
}

const bike1 = new Vehicle('Ninja ZX-10R','998cc');
const bike2 = new Vehicle('Duke','390cc');

console.log(bike1.name); // Ninja ZX-10R
console.log(bike2.name); // Duke
```

Where you can go to learn more

There are many places you can go to learn more about ES6 and the newer specifications. If you feel like reading through the actual documentation of ECMAScript, follow the link below:

https://www.ecma-international.org/ecma-262/9.0/

Also below are the links to the MDN pages of the features mentioned above. You can learn a lot more about them in these pages:

- Keyword 'const'
- Arrow Functions
- Template Literals
- Iterator 'for of'
- Default Parameters
- Internationalization (i18n)