ES6 Notes :

1. When I declare a variable with the var keyword it is declared globally or locally if inside a function.  
   in order to separate a variable inside a function and the same one inside a “if” statement, I should use the “let” keyword.
2. Const variable is still vulnerable. If I have an array const myArray = [1, 2, 3]; and I will try to modify the array to myArray = [4, 5, 6] I will get an error. But if I will use the indexer of that Array I will be able to change the value.  
   myArray[0] = 2; will result that the Array is [2, 2, 3] now.
3. As said before about const, javascript enable the option to freeze the object therefore every attempt to change the object will result in error.  
   myArray = [1, 2, 3];  
   Object.freeze(MyArray);
4. Arrow function help to shorten the code, instead of declaring function, we can set the variable to function. (const myVar = function() …).   
   Here the arrow come in handy. Instead of writing function(), we can declare the variable with (const myVar = () => {code}).  
   furthermore, if the function only has a return method, we can shorten the code even more! (const myVar = () => “value”;) instead of (const myVar = () => return ..)
5. We can use arrow function to even pass parameters.   
   for example an “add” function of calculator could be like this :  
   const add = (num1, num2) => num1 + num2;  
   And if there is only one parameter it could be like this :  
   const showMeTheNumber = number => number;
6. We can concat an arrays using the concat function.  
   arr1.concat(arr2) or arr2.concat(arr1).
7. ES6 introduce a default parameters for function. So if a user didn’t put an value into a string or int parameters, we can use them as default.  
   For example if I had a calc function of add :  
   const add = (number1 = 1, number2) => number1 + number2.  
   if the user did not specify the first number with value, the result will be 1+number2.
8. Reduce function connect all numbers \ strings.

**Takes** an initial value.

**Sets** the result as the initial value to start with.

**Calls** a function on the result and the first element.

**Updates** the result to the return value of this function call.

**Calls** a function on the updated result and the second element.

**Repeats** until there are no values left in the array.

**Returns** the result.

for example :  
const message = [“I”, “Love”, “Pizza”];  
  
const conString = (firstValue, secondValue) => firstValue + secondValue;  
  
console.log(message.reduce(conString));

1. Rest parameter make life easier in terms he can take up variable number of parameters. For example:  
   const myFunc = (…args) = how many parameters Will I choose to pass!!  
   For example :  
   const sum = (...args) => args.reduce((a, b) => a + b, 0);  
   now calling sum(1,2,3,4,5,6,7,8,9,10) will be equal 55.   
   Lets go calculator!!!!
2. Spread operator can be declared as follow :  
   if I want to find the biggest number in an array, I will use the Math.max(…arr) , which will take all the values from the arr and spread.  
   another example is to copy the array values into a new array set.  
   const myArray = [1,2,3];  
   const myArray2 = […myArray];
3. Destructuring assignment is neatly assigning values taken directly from an object.  
   in es5 I would use :  
   const user = { name: ‘john’, age: 45}  
   const name = user.name;  
   const age = user.age;

but with Destructuring I can use as follow instead of declaring 2 variables :  
const user = {name: ‘john’, age: 45}  
const {name, age} = user;

furthermore I can use Destructuring to assign the values into new variables like so :  
const user = {name: “matan”, age: 24};  
const {name: newName, age: newAge} = user;  
this will create 2 variables (newName and newAge) and assign the values of name and age from the user JSON.  
  
Array Destructuring is very like spread (…) but using commas.  
for example :  
const [a,b,,,c] = [1,2,3,4,5,6] // a=1, b=2, c=6  
we can reach index by using commas.   
  
Another use of Destructuring Assignment is like slicing. Say I want to cut [1,2] from the array [1,2,3,4,5,6,7,8,9].   
const [a, b, …arr] = [1, 2, 3, ,4 ,5 ,6 ,7, 8, 9] will become  
a, b = [1, 2]  
arr = [3, 4, 5, 6, 7, 8, 9]  
I can also create a function and use it to return the array after cutting the a, b. like so :  
const source = [1,2,3,4,5,6,7,8,9,10];

function removeFirstTwo(list) {

const [a, b, ...arr] = list;

return arr;

}

const arr = removeFirstTwo(source);

1. To call or declare a variable : ${variable}
2. To use foreach : array.forEach(function => code).  
   With arrowing it would look like : array.forEach(item => array.push(`${variable}`)
3. Instead of creating new objects in this way :  
   const getMousePosition = (x, y) => ({

x: x,

y: y});

I can simplify this to:  
const getMousePosition = (x, y) => ({ x, y });

1. In ES5 when defining function within object we used the function keyword after the name of the function like so:  
   sayHello: function() {  
   in ES6 we don’t have to define the function keyword anymore. Like so :   
   sayHello() {
2. To create class in JS we should use the class keyword before the class name. like so :  
   class className {}
3. Classes should use constructor just like in C# and they are the first thing to run when calling the class. For example :  
   class Vegetable {  
   constructor(name) {  
    this.name = name;  
   }  
   }
4. Promise is a constructor and it's just like real-life. When you make a promise you eiter fulfill it or not.   
   promise have 3 states – pending, fulfilled and rejected.   
   const makeServerRequest = new Promise((resolve, reject) => {

});  
this is at pending state, because resolve or reject didn’t showed up.