DECT-KS© + TOOLS

前端網絡開發人員課程
(二) 進階網絡程式設計

9. ES6 JS II: Syntax II

Presented by Krystal Institute

erb 僱員再培訓局 Employees Retraining Board

SCOPE School of Continuing and Professional Education 專業進修學院

KRYSTAL INSTITUTE DIGITAL ECONOMY CORE TECHNOLOGY

Learning Objective

- Understand what some of the new syntaxes of ES6 are, and how to use them effectively

Content

9.1
Revise on the previous lesson

9.2
ES6 Syntax pt.1

9.1 Revise on the previous lesson

ES6

- ES6 is a programming language standard for JavaScript
- Makes JS code more modern and readable
- There is many new features and additions to ES6 that tackles issues from ES5

let

- Let is a new keyword that is used to declare a new variable

```
let variable_name;
```

- Variables declared using let will be block-scoped, unlike var

- Variables declared using let is not added to the property list of the global object – window

let

- As let is block scoped, inside a for loop, it will declare a new temporary variable on every loop as if it's a separate block

- This could avoid issues with using asynchronous functions and methods like the setTimeout function

let vs. var

- Variables declared using var is global unless inside a function
- Variables declared using let is block-scoped

- Variables can be redeclared using var, however this has no benefits
- Variables cannot be redeclared using let

- Variables declared using var can hoist, however this has no benefits
- Variables declared using let cannot hoist

Constants

- The const keyword creates a read-only block-scoped constant that must be assigned a value upon declaring

- They are immutable, meaning you cannot reassign values to it

Constants

- Changing a const object's property is possible, but reassigning it to another object is not
- const arrays can use array methods like push and pop, but reassigning will not work
- It is recommended to use all upper cases to declare constants to differentiate them from other mutable variables

Default Parameters

- Default parameters allows you to set a default value of a parameter in a function
- Calling the function with less or no argument will cause the parameter to default to the preset value
- Calling the function with undefined arguments will also defaults the arguments to the default value

9.2 ES6 Syntax pt.1

Rest Parameter

- Before ES6, codes will work like this:

- See the example on right
- It adds up 2 numbers and console logs it

```
<script>
    function add(a, b) {
        console.log(a + b);
    };
    add(10, 20);
</script>
```

- But what if I want to add up 3, 4 or 5 numbers?

Rest Parameter

- The rest parameter in ES6 solves the issue by allowing any number of arguments to be passed into the function and used
- It has a prefix of ...
- It has to be put as the last parameter

```
function add(a, b ,...num) {
    //
};
```

Rest Parameter

- Put it at the end along with the name of the argument
- The rest parameter will represent any number of arguments after the three dots as an array
- In the example, add(10, 20, 30, 40, 50)
- Is equal to add([10, 20, 30, 40, 50])

```
function add(...num) {
    let sum = 0
    for (let i = 0; i < num.length; i++) {
        sum += num[i]
    }
    console.log(sum);
};
add(10, 20, 30, 40, 50);
// same as add([10, 20, 30, 40, 50])
```

Rest Parameter Exercise

- Create a function with 3 arguments using rest parameter:
- First name (e.g. James)
- Last name (e.g. Ng)
- A dynamic number of arguments (greater than 0) showing all the titles the person might have (e.g. Employee of company ABC, basketball hobbyist)

In the function, compile and display a sentence that shows the persons' name and all their titles.

(e.g. James Ng is a employee of company ABC, and a basketball hobbyist)

Rest Parameter Exercise Example

- The examples shows the function with 1 title, 2 titles and 3 titles

```
showPerson("Keith", "Fung", "Firefighter")
showPerson("Marvin", "Chan", "Highschool student", "Pet Owner")
showPerson("Justin", "Lau", "Consultant", "Hiker", "Camper")
```

```
Keith Fung is a Firefighter
Marvin Chan is a Highschool student, and a Pet Owner
Justin Lau is a Consultant, a Hiker, and a Camper
```

Rest Parameter Exercise Solution

- In the function, form the first part of the sentence using firstname and lastname arguments
- For the titles array formed using rest parameter
- If the title is not the last one or the only one, add it to the sentences string

```
function showPerson(firstname, lastname, ...titles) {
    sentence = firstname + " " + lastname + " is ";
    for (let i = 0; i < titles.length; i++) {
        if (i != titles.length - 1) {
            sentence += "a " + titles[i] + ", ";
        } else {
            if (titles.length == 1) {
                sentence += "a " + titles[i]
            } else {
                sentence += "and a " + titles[i];
            };
        };
    };
    console.log(sentence);
};
```

Rest Parameter Exercise Solution

- If the title is the only one in the array, add it to the sentence without the comma

- If the title is the last one in the array, add it to the sentence with “and”

```
function showPerson(firstname, lastname, ...titles) {
    sentence = firstname + " " + lastname + " is ";
    for (let i = 0; i < titles.length; i++) {
        if (i != titles.length - 1) {
            sentence += "a " + titles[i] + ", ";
        } else {
            if (titles.length == 1) {
                sentence += "a " + titles[i]
            } else {
                sentence += "and a " + titles[i];
            };
        };
    };
    console.log(sentence);
};
```

Spread Operator

- The spread parameter is very useful in spreading out elements inside an array

- It uses the same prefix like the rest parameter - ...

- It unpacks elements inside an array

```
let numbers = [1, 2, 3, ...[4, 5, 6]]
```

Spread Operator

- The spread operator can be used in any position of an array or iterable objects
- This makes many operations simpler

```
let num = [4, 5, 6]
let numbers = [1, ...num, 2, 3]
console.log(numbers) // [1, 4, 5, 6, 2, 3]
```

Spread Operator Activity

- Finish and display the following using the spread operator:
- Add an array [“A”, “B”] into another array [“C”, “D”] and display [“A”, “B”, “C”, “D”]
- Unpack strings inside an array using only “A”, “BC” and “D” to form [“A”, “B”, “C”, “D”]
- Use one array.push() method to add [“C”, “D”] into [“A”, “B”]

Spread Operator Activity

- Add an array [“A”, “B”] into another array [“C”, “D”] and display [“A”, “B”, “C”, “D”]

```
let a = ["A", "B"];
let b = [...a, "C", "D"];
console.log(b) // ["A", "B", "C", "D"]
```

- Unpack strings inside an array using only “A”, “BC” and “D” to form [“A”, “B”, “C”, “D”]

```
let c = ["A", ..."BC", "D"];
console.log(c) // ["A", "B", "C", "D"]
```

- Use one array.push() method to add [“C”, “D”] into [“A”, “B”]

```
let d = ["A", "B"];
d.push(...["C", "D"])
console.log(d) // ["A", "B", "C", "D"]
```

Object Literal Extensions: Property Duplication

- Before ES6, an object literal is a collection of name-value pairs

- The example on right creates an object with name and age properties

```
function Person(name, age) {
    return {
        name: name,
        age: age
    };
};
console.log(Person("Chris", 21))
```

```
▶ {name: "Chris", age: 21}
```

Object Literal Extensions: Property Duplication

- The name and age properties take the value of parameters name and age

- This might be redundant as name and age is mentioned twice inside the object

```
function Person(name, age) {
    return {
        name: name,
        age: age
    };
};
console.log(Person("Chris", 21))
```

```
▶ {name: "Chris", age: 21}
```

Object Literal Extensions: Property Duplication

- In ES6, you can remove the duplication when the property of the object is the same as the parameter name

```
function Person(name, age) {
    return {
        name,
        age
    };
};
console.log(Person("Chris", 21))
```

```
▶ {name: "Chris", age: 21}
```

Object Literal Extensions: Property Naming

- Before ES6, using square brackets ([ ]) will let you use string literals and variables as property names

```
let name = "person name"
let Person = {
    [name]: "Chris",
    "person age": 21
};
console.log(Person[name]) // Chris
console.log(Person["person name"]) // Chris
```

Object Literal Extensions: Property Naming

- In ES6, when a property name is placed inside the square brackets, it will be counted as a string
- String expressions can be used for object literal properties

```
let prefix = "person ";
let Person = {
    [prefix + "name"]: "Chris",
    "person age": 21
};
console.log(Person["person name"]) // Chris
```

Object Literal Extensions: Functions

- Before ES6, property names need to be specified before the value, that happens even for functions inside an object literal

```
let Person = {
    name: "Chris",
    age: 21,
    birthdate: function() {
        // Code
    }
};
```

Object Literal Extensions: Functions

- In ES6, property name don't have to be included and the property can be treated as the name and the function at the same time

```
let Person = {
    name: "Chris",
    age: 21,
    birthdate() {
        // Code
    }
};
```

For...of

- For…of is a new constructor that is a new way of iterating over iterable object

- It works similar to a for loop, but only for iterating through an iterable object

```
for (item of array) {
    // code
};
```

For...of: Before and After

- Before ES6, iterating through an array using for loop will look like this:
- This function uses the array and add 1 to each number inside

```
let numlist = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for (let i = 0; i < numlist.length; i++) {
    numlist[i] += 1;
};
console.log(numlist);
```

```
▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

For...of: Before and After

- In ES6, iterating through an array using for...of loop will look like this instead:

```
let numlist = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for (let i of numlist) {
    numlist[i] += 1;
};
console.log(numlist);
```

```
▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

For...of: Entries

- Before ES6, to get the value of an array element, you will need to use the index of the array to get the value

```
let animals = ["Seagull", "Lizard", "Cat", "Bear"]
for (let i = 0; i < animals.length; i++) {
    console.log(animals[i] + " is at index " + i);
}
```

- The entries method makes the code simpler

```
Seagull is at index 0
Lizard is at index 1
Cat is at index 2
Bear is at index 3
```

For...of: Entries

- array.entries can also be used in the for...of statement

```
let animals = ["Seagull", "Lizard", "Cat", "Bear"]
for (const [index, animal] of animals.entries()) {
    console.log(animal + " is at index " + index);
};
```

- The example on right displays animals inside an array and their index

```
Seagull is at index 0
Lizard is at index 1
Cat is at index 2
Bear is at index 3
```

Octal Literals

- Octal numbers are numbers with a base of 8
- That means it only uses the number 0 to 7, and will add a digit when it goes over 7
- One of ES6's new additions was the support for Octal literals

Octal Numbers	Decimal Numbers
1	1
2	2
3	3
4	4
5	5
6	6
7	7
10	8
11	9
12	10

Octal Literals

- To represent octal literals, prefix 0o (zero followed by the letter o) is used followed by the octal number

```
let a = 0o23;
console.log(a)
```

- Displaying it will result in it converting into decimal numbers

```
19
```

Octal Literals

- If the number is invalid (e.g. using 8 in a octal literal)

```
let a = 0o28;
console.log(a)
```

- SyntaxError will be thrown

```
Uncaught SyntaxError: Invalid or unexpected token
```

- Back in ES5, using the prefix 0 for octal literal will not cause any errors if there is invalid numbers, it will see it as decimal instead

Binary Literals

- In ES5, there is no literal form of binary numbers, to create one

```
let a = parseInt('10110', 2);
console.log(a)
```

- you have to use the parseInt function

```
22
```

Binary Literals

- In ES6, it is possible to declare binary literals by using the 0b prefix followed by binary numbers

```
let a = 0b10110;
console.log(a)
```

- Octal and Binary Literals may not seem useful at first, but it can be used as bit data that could act as encryption tools!

```
22
```

For...of Exercise

- Create a function that finds and displays the number of a specified animal in an array of animals
- The function accepts 2 variables: the specified animal, and the animal array

- Create a function that multiplies a array of numbers by a specified number
- The function accepts 2 variables: the specified multiplier, and the number array
- Finish this exercise by the end of this lesson

For...of Exercise Example

- Create a function that finds and displays the number of a specified animal in a array of animals

```
let alist = ["Owl", "Bat", "Parrot", "Parrot", "Bat", "HummingBird"]
let a = AnimalCount(“Bat", alist)
console.log(a)
```

```
2
```

```
let alist = ["Owl", "Bat", "Parrot", "Parrot", "Bat", "HummingBird"]
let a = AnimalCount("Dog", alist)
console.log(a)
```

```
0
```

For...of Exercise Example

- Create a function that multiplies a array of numbers by a specified number

```
let numlist = [1, 2, 3, 4, 5, 6]
let b = MultiplyList(2, numlist);
console.log(b)
```

```
▶ (6) [2, 4, 6, 8, 10, 12]
```

References

- Use these if you need more explanations!
- https://www.javascripttutorial.net/es6/
- https://javascript.info/

- Use this if you need more specific answers!
- https://stackoverflow.com/