Front-end Advanced

ES6 01 - New Syntax

Table of Contents

Section 1

History of JavaScript

History of JavaScript

1. Created in 1995 (by Brendan Eich in 10 days)
2. Standardizing in 1997 (officially name ECMAScript)
3. ES2 in 1998
4. ES3 in 1999
5. ES4 (abandoned 2008)
6. ES5 in 2009 (Major update)
7. **ES6 in 2016 (or ES2015)**
8. ES7 in 2017 (or ES2016)
9. And so on, JavaScript is now ”most used” language

Section 2

Transpiler

Transpiler

- What if user's browser is pretty old and not very friendly (like Internet Explorer), can it run new JavaScript code ?
- No, we have to do 1 of the following:
  1. Ask the user to update his/her browser (not possible)
  2. Downgrade the source code to support older browser

Transpiler

- Transpiler is a tool that downgrade the source code from new version to older version in-order to support many browsers

New version

```
[1, 2, 3, 4].map(e => e + 1);
(4) [2, 3, 4, 5]
```

input → Transpiler → output

Old version

```
[1, 2, 3, 4].map(function(e){
  return e + 1;
});
```

Transpiler

- Popular Transpiler:
  - Babel (https://babeljs.io/en/repl)
  - TypeScript (https://www.typescriptlang.org/play/)

Section 3

Arrow function

Arrow function

- Syntax:

added

Function expression

```
var add = function (a, b) {
  return a + b;
};
```

Arrow function

```
var add = (a, b) => {
  return a + b;
};
```

removed

Arrow function

- Variations:

```
// 1. If no parameter
var add = () => {
  return 10;
}
```

```
// 2. If only 1 parameter

var add = a => a + 10;
```

```
// 3. If you want to return an expression
// (remark no {} and return)

var add = (a, b) => a + b;
```

Arrow function

- Recall what is `this` and how to determine `this` ?

Arrow function

```
// HTML code
<div>
  <span id="span"></span>
  <button id="btn">Click</button>
</div>

// JS code
class App {
  constructor() {
    this.count = 0;
    this.btn = document.getElementById('btn');
    this.span = document.getElementById('span');

    this.btn.addEventListener('click', function() {
      this.count += 1; // this refer to window
      // same as window.count += 1 ~ undefined += 1 => NaN

      this.span.innerText = this.count; // this refer to window now
      // same as window.span.innerText ~ undefined.innerText => Error
    });
  }
}

new App();
```

Arrow function

```
// HTML code
<div>
  <span id="span"></span>
  <button id="btn">Click</button>
</div>

// JS code
class App {
  constructor() {
    this.count = 0;
    this.btn = document.getElementById('btn');
    this.span = document.getElementById('span');

    this.btn.addEventListener('click', () => {
      this.count += 1;

      this.span.innerText = this.count;
    });
  }
}

new App();
```

2

1

3

Arrow function

- Shorter syntax
- Implicit return when there is no block body
- lexcal `**this**` (can be dangerous since we can’t use arrow function as method in prototype anymore)

Section 4

Block Scope

Block Scope

- ES6 introduce another type of Scope (Block Scope)
- Syntax:

Block starts

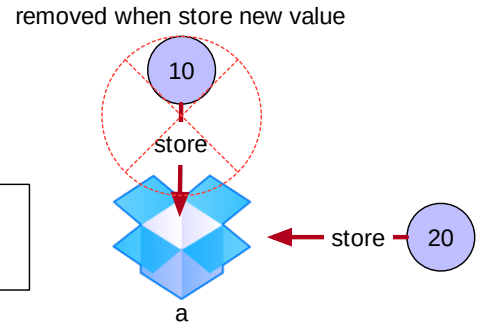
Block ends

```
{
    // must use const/let to declare block-scope variable
    let a = 10;

    var b = 10;
    // var is hoisted
}

console.log(b); // 10

console.log(a); // ReferenceError
```

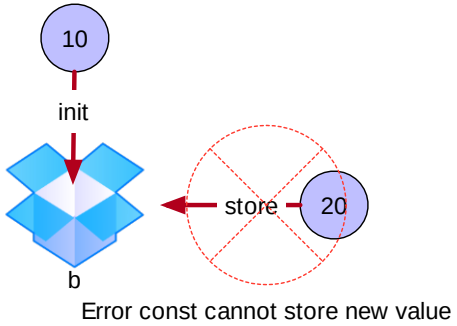
Block Scope

- let vs const

```
let a = 10;
a = 20; // OK
```




```
const b = 10;

b = 10; // Error
```



Section 5

Rest/Spread

Rest/Spread

- **Problem:** calculate **min** of an array using **Math.min**

```
var array = [1, 2, 3, 4];

Math.min(array); // NaN
Math.min(1, 2, 3, 4); // work
```

Rest/Spread

- **Spread:** allow us to convert an array to a list of parameter to support case like Math.min

```
var array = [1, 2, 3, 4];

Math.min(...array); // 1
// same as Math.min(1, 2, 3, 4);
```

Rest/Spread

- **Problem:** how too support function that have variable of parameter? (like Math.min)

```
function test(a) {} // 1 parameter

function test(a, b) {} // 2 parameters

function test(a, b, c) {} /// 3 parameters
```

Rest/Spread

- **Problem:** how too support function that have variable of parameter? (like Math.min)

```
function test(a, b, c) {
  console.log(arguments);
}


test(1);
test(1, 2, 3);
// Arguments(3) [1, 2, 3, callee: f, Symbol(Symbol.iterator): f]
```

Rest/Spread

- **Problem:** how too support function that have variable of parameter? (like Math.min)

```
function test(a, b, c) {
  console.log(arguments); // look like array but it NOT an array;

  console.log(arguments.map);// undefined
  console.log(arguments.filter); // undefined
  console.log(arguments.reduce); // undefined
}

test(1, 2, 3, 4, 5);
```

Rest/Spread

- **Rest** allow use to have **fixed** number of parameter and **variable** number of paremeter **combine**

Fixed 2 parameters

```
function test(a, b, ...rest) {
  console.log(a, b, rest);
}

test(1, 2, 3, 4, 5);
test(1, 2, 3, 4, 5, 6);
```

```
1 2 ▶ (3) [3, 4, 5]
1 2 ▶ (4) [3, 4, 5, 6]
```

Variable parameters

a and b

rest (array type now)

Rest/Spread

- **Rest vs Spread:** how to differentiate ?

```
function test(a, b, ...rest) {
  console.log(a, b, rest);
}

var array = [1, 2, 3, 4, 5, 6];

test(...array);
```

Same keyword

Rest/Spread

- **Rest vs Spread:** how to differentiate ?

```
function test(a, b, ...rest) {
  console.log(a, b, rest);
}

var array = [1, 2, 3, 4, 5, 6];

test(...array);
```

assignment context

expression context

Section 6

Destructuring

Destructuring

- Problem with below code ?

```
var data = getDataFromBackend(); // line 1
// ...
getUser(data[0].id); // line 10
// ...
getUserPosts(data[0].postId); // line 100
// ...
getMoreData(data[0].name); // line 1000
```

Destructuring

- Problem with below code ?

```
var data = getDataFromBackend(); // line 1
// ...
getUser(data[0].id); // line 10
// ...
getUserPosts(data[0].postId); // line 100
// ...
getMoreData(data[0].name); // line 1000
```

Destructuring

- Destructuring allow **developer** to describe the data structure at a single place and extract value to variable to use later.

```
var [ { id: id, postId: postId, name: name } ] = getDataFromBackend(); // line
1
// ...
getUser(id); // line 10
// ...
getUserPosts(postId); // line 100
// ...
getMoreData(name); // line 1000
```

Array Destructuring

- 1st type of Destructuring: Array Destructuring

```
var array = [1, 2, 3, 4];

var [first, second] = array;

console.log(first, second);
```

Array Destructuring

- Array Destructuring Usage:

assign

```
// swap 2 number;

var a = 1;
var b = 2;

[a, b] = [b, a];
```

assign

Object Destructuring

- 2nd type of Destructuring: Object Destructuring

```
var a = { name: 'Anh' };

var { name: name } = a;
// same as var { name } = a;

console.log(name); // Anh
```

assign

Object Destructuring

- **Problem:** How to differentiate Destructuring and Object/Array literal?

Assignment context

```
var [ { id: id, postId: postId, name: name } ] = getDataFromBackend(); // line
1
// ...
getUser(id); // line 10
// ...
getUserPosts(postId); // line 100
// ...
getMoreData(name); // line 1000
```

Expression context

Object Destructuring

- **Usage:** Use Destructuring to describe input of function

Destructuring

```
function checkUser({ name, age, clazz }) {
  console.log(name, age, clazz);
}


checkUser({ name: 'Anh', age: 20, clazz: 'ReactJS'});
```

Section 7

Template String

Template String

- **Problem:** concatenate string in ES5

```
var name = 'AnhNV'
var s = 'Hello ' + name; // easy


// become a mess when working with HTML string
// prone to error
var span = '<span id="span" title="' + name +' ">' + name +  '</span>'
```

Template String

- **Template String**: make it easier to work with string

Backtick

expression

```
var name = 'AnhNV'
var s = `Hello ${name}`; // easy

var span = `<span id="span" title="${name}">${name}</span>`;
// "<span id="span" title="AnhNV">AnhNV</span>"
```

Template String

- **Tag function**: any function prefix template string

```
function tag(strings, ...values) {

   console.log(strings, values);
}


var name = 'AnhNV';

tag`Hello ${name}`;
```

```
▶ (2) ["Hello ", "", raw: Array(2)] ▶ ["AnhNV"]
```

Summary

- Understand brief history of JavaScript and its official name
- Understand how to support old browsers with transpiler
- Understand new features in ES6: arrow function, block-scope, destructuring, rest/spread, template string

References

- https://github.com/getify/You-Dont-Know-JS/blob/1st-ed/es6%20%26%20beyond/ch2.md#arrow-functions

Thank you