A.1 進入 Vue.js 前的 ES6 必備知識

延伸閱讀:

1. JavaScript ES6-重點紀錄 系列總集

A.1.1 var, let, const

const - 不變

let - 提高使用率

var - 應減少使用

推薦閱讀:

1. Day26 var 與 ES6 let const 差異
2. [JavaScript] 談var、let、const差異之var你這個矯情的賤人

A.1.2 ES Module & import, export

Node.js 不支援 ES6 匯出導入 -> 採用 html scripts 標籤 引入

Node.js 寫法:

- Node.js
  - 模組引入 `require`

```
const _ = require('lodash');
```

  - 模組導出 `module.exports` / `exports`

```
module.exports = {
  myFun: ()=>{}
}
```

推薦閱讀:

1. exports、module.exports和export、export default到底是咋回事
2. Day 7 - 一周目 - ES6 讓人欲罷不能的語法

A.1.3 箭頭函式 & this　　A.1.4 字串模板

```
const plus = function(numA, numB){
    return numA + numB;
};

const plus2 = (numA, numB) => {
    return numA + numB;
};

const plus3 = (numA, numB) => numA + numB;

const noParameter = () => console.log('noParameter');

const helloPerson = person => `Hello, ${person}`;

const helloPerson2 = function(name){
    console.log(`hello, ${name}`);
}

console.log(helloPerson("123"));
console.log(plus3(1, 2));
noParameter();
helloPerson2('test');
```

```
Hello, 123
3
noParameter
hello, test
```

參考文章:

1. 解構賦值

A.1.5 解構賦值

先解構, 再賦值。

陣列

```
let array = [1, 2, 3];
let one = array[0];
let two = array[1];
let three = array[2];
```

```
let arrTwo = [4, 5, 6]
let [num1, num2, num3] = arrTwo;
```

物件

```
let obj = {one: 1, two: 2};
let one = obj.one;
let two = obj.two;
```

```
let objTwo = {three: 3, four: 4};
let {three, four} = objTwo;

console.log(three);
console.log(four);
```

```
//type2 宣告與賦值分開
let arrTwo = [4, 5, 6];
let num1, num2, num3;
[num1, num2, num3] = arrTwo;

console.log(num1);
console.log(num2);
console.log(num3);
```

```
//type3 預設值
let arrTwo = [4]
let [num1, num2 = 7, num3 = 8] = arrTwo;

console.log(num1);
console.log(num2);
console.log(num3);
```

```
//type3 預設值
let arrTwo = [4, 5, 6]
let [num1, num2 = 7, num3 = 8] = arrTwo;

console.log(num1);
console.log(num2);
console.log(num3);
```

物件須按照物件名稱對應。

```
let obj = {one: 1, two: 2};
let test = obj.one;
let g = obj.two;

console.log(test);
console.log(g);

let objTwo = {three: 3, four: 4};
let {test2, four} = objTwo;

console.log(test2);   undefined
console.log(four);
```

```
//type2 宣告與賦值分開
let objTwo = {three: 3, four: 4};
let three, four;
({three, four} = objTwo);
```

```
//type3 預設值
let objTwo = {three: 3};
let {three, four = 5} = objTwo;
```

```
//type4 指定新的變數名稱

let objTwo = {three: 3, four: 4};
let {three:test2, four:test100} = objTwo;

console.log(test2);
console.log(test100);
```

A.1.6 ... 展開運算子 / 其餘運算子

其餘運算子是把許多的參數轉換成一個陣列, 而展開運算子則是可以把陣列中的元素取出。

```
let arr = [1,2,3,4,5];

let avg = function(arr){
  let sum = 0;
  for(let i = 0; i < arr.length; i++){
    sum += arr[i];
  }
  return sum / arr.length;
}

console.log(avg(arr)); // 3

console.log(avg(1, 2, 3, 4, 5)); // NaN

let avgs = function(...arr){
    let sum = 0;
    for(let i = 0; i < arr.length; i++){
      sum += arr[i];
    }
    return sum / arr.length;
  }

  console.log(avgs(1, 2, 3, 4, 5)); // 3
```

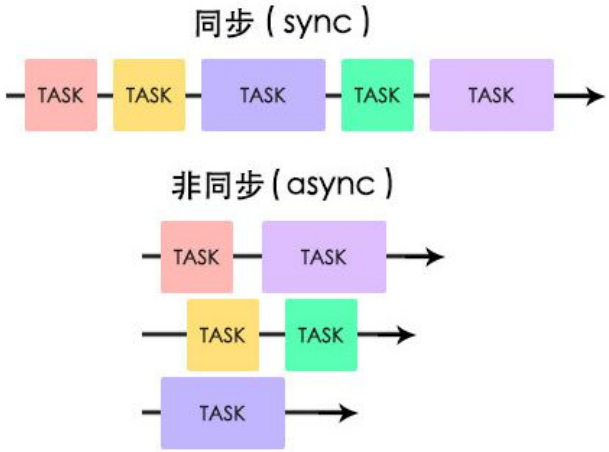
```
let number = [1, 2, 3, 4, 5];

console.log(Math.max(number)); // NaN

let numbers = [1, 2, 3, 4, 5];

console.log(Math.max(...numbers)); //  5

console.log(...numbers); // 1,2,3,4,5
```

參考文章:

1. [筆記] JavaScript ES6 中的展開運算子（spread operator）和其餘運算子（rest operator）

A.1.7 Promise 物件：async & await

- 同步：在「**同一個步道**」比賽「接力賽跑」，當棒子沒有交給我，我就得等你，不能跑。
- 非同步：在「**不 (非) 同步道**」比賽「賽跑」，誰都不等誰，只要輪到我跑，我就開始跑。



關於 await

此 await 表示法會暫停 async 函式執行，等待 Promise 物件的解析，並在 promise 物件的值被 resolve 時回復 async 函式的執行。await 接著回傳這個被 resolve 的值。如果回傳值不是一個 Promise 物件，則會被轉換為 resolved 狀態的 Promise 物件。

如果 Promise 物件被 rejected，則 await 會丟出 rejected 的值。

參考：

1. Async function / Await 深度介紹
2. Javascript Promises vs Async Await EXPLAINED (in 5 minutes)
3. 簡單理解 JavaScript Async 和 Await