3月2日报

1.今天内容  
讲的知识复习，安装软件

2.工作成果

Array.from方法用于将两类对象转为真正的数组：类似数组的对象（array-like object）和可遍历（iterable）的对象（包括 ES6 新增的数据结构 Set 和 Map）。

下面是一个类似数组的对象，Array.from将它转为真正的数组。

let arrayLike = {

'0': 'a',

'1': 'b',

'2': 'c',

length: 3};

// ES5的写法var arr1 = [].slice.call(arrayLike); // ['a', 'b', 'c']

// ES6的写法let arr2 = Array.from(arrayLike); // ['a', 'b', 'c']

实际应用中，常见的类似数组的对象是 DOM 操作返回的 NodeList 集合，以及函数内部的arguments对象。Array.from都可以将它们转为真正的数组。

// NodeList对象let ps = document.querySelectorAll('p');

Array.from(ps).filter(p => {

return p.textContent.length > 100;});

// arguments对象function foo() {

var args = Array.from(arguments);

// ...}

上面代码中，querySelectorAll方法返回的是一个类似数组的对象，可以将这个对象转为真正的数组，再使用forEach方法。

只要是部署了 Iterator 接口的数据结构，Array.from都能将其转为数组。

Array.from('hello')

// ['h', 'e', 'l', 'l', 'o']let namesSet = new Set(['a', 'b'])

Array.from(namesSet) // ['a', 'b']

上面代码中，字符串和 Set 结构都具有 Iterator 接口，因此可以被Array.from转为真正的数组。

如果参数是一个真正的数组，Array.from会返回一个一模一样的新数组。

Array.from([1, 2, 3])

// [1, 2, 3]

值得提醒的是，扩展运算符（...）也可以将某些数据结构转为数组。

// arguments对象function foo() {

const args = [...arguments];}

// NodeList对象[...document.querySelectorAll('div')]

扩展运算符背后调用的是遍历器接口（Symbol.iterator），如果一个对象没有部署这个接口，就无法转换。Array.from方法还支持类似数组的对象。所谓类似数组的对象，本质特征只有一点，即必须有length属性。因此，任何有length属性的对象，都可以通过Array.from方法转为数组，而此时扩展运算符就无法转换。

Array.from({ length: 3 });

// [ undefined, undefined, undefined ]

上面代码中，Array.from返回了一个具有三个成员的数组，每个位置的值都是undefined。扩展运算符转换不了这个对象。

对于还没有部署该方法的浏览器，可以用Array.prototype.slice方法替代。

const toArray = (() =>

Array.from ? Array.from : obj => [].slice.call(obj))();

Array.from还可以接受第二个参数，作用类似于数组的map

3. 未完成内容及原因

1. 体会与心得

**教师评语**