JavaScript 进阶 - 第4天

了解函数中 this 在不同场景下的默认值，动态指定函数 this 的值，提升代码封装的灵活度。

- 能够区分不同场景下函数中 this 的默认值
- 知道箭头函数的普通函数的区别，掌握箭头函数的使用
- 能够动指定函数中 this 的值
- 了解基于类的面向对象的实现语法

this

了解函数中 this 在不同场景下的默认值，知道动态指定函数 this 值的方法。

默认值

普通函数

箭头函数

普通函数

普通函数的调用方式决定了 this 的值，即【谁调用 this 的值指向谁】

注： 普通函数没有明确调用者时 this 值为 window，严格模式下没有调用者时 this 的值为 undefined。

箭头函数

箭头函数中的 this 与普通函数完全不同，也不受调用方式的影响，事实上箭头函数中并不存在 this ！箭头函数中访问的 this 不过是箭头函数所在作用域的 this 变量。

在开发中【使用箭头函数前需要考虑函数中 this 的值】，**事件回调函数**使用箭头函数时，this 为全局的 window，因此DOM事件回调函数不推荐使用箭头函数

同样由于箭头函数 this 的原因，**基于原型的面向对象也不推荐采用箭头函数**

定义值

以上归纳了普通函数和箭头函数中关于 this 默认值的情形，不仅如此 JavaScript 中还允许指定函数中 this 的指向，有 3 个方法可以动态指定普通函数中 this 的指向：

- **Call**
- **apply**
- **Bind**

call

使用 call 方法调用函数，同时指定函数中 this 的值

- 总结：
- call 方法能够在调用函数的同时指定 this 的值
- 使用 call 方法调用函数时，第1个参数为 this 指定的值
- call 方法的其余参数会依次自动传入函数做为函数的参数

apply

使用 call 方法**调用函数**，同时指定函数中 this 的值

- 总结：
- apply 方法能够在调用函数的同时指定 this 的值
- 使用 apply 方法调用函数时，第1个参数为 this 指定的值
- apply 方法第2个参数为数组，数组的单元值依次自动传入函数做为函数的参数

bind

bind 方法并**不会调用函数**，而是创建一个指定了 this 值的新函数

注：bind 方法创建新的函数，与原函数的唯一的变化是改变了 this 的值。

改变this三个方法总结

call：fun.call(this，arg1, arg2,……)

apply：fun.apply(this, [arg1, arg2,……])

bind：fun.bind(this, arg1, arg2,……)

相同点：

都可以用来改变this指向，第一个参数都是this指向的对象

区别：

call和apply：都会使函数执行，但是参数不同

bind：不会使函数执行，参数同call

class

了解 JavaScript 中基于 class 语法的面向对象编程，为后续课程中的应用做好铺垫。

传统面向对象的编程序语言都是【类】的概念，对象都是由类创建出来，.

然而早期 JavaScript 中是没有类的，面向对象大多都是基于构造函数和原型实现的，

但是 ECMAScript 6 规范开始增加了【类】相关的语法，使得 JavaScript 中的面向对象实现方式更加标准。

封装

class（类）是 ECMAScript 6 中新增的关键字，专门用于创建类的，类可被用于实现逻辑的封装。

```
<script>
 // 创建类
 class Person {
     // 此处编写封装逻辑
 }

 // 实例化
 let p1 = new Person();
 console.log(p1);
</script>
```

实例成员

属性 = 值;

- 总结：
- 关键字 class 封装了所有的实例属性和方法
- 类中封装的并不是变量和函数，因此不能使用关键字 let、const 或 var

静态成员

static 属性 = 值;

- 总结：
- static 关键字用于声明静态属性和方法
- 静态属性和方法直接通过类名进行访问

构造函数

创建类时在类的内部有一个特定的方法 constructor ，该方法会在类被实例化时自动被调用，常被用于处理一些初始化的操作。

- 总结：
- constructor 是类中固定的方法名
- constructor 方法在实例化时立即执行
- constructor 方法接收实例化时传入的参数
- constructor 并非是类中必须要存在的方法

extends

- extends 是专门用于实现继承的语法关键字
- 如果想要实现继承，用extends定义子类，去继承父类

super

在继承的过程中子类中 constructor 中必须调 super 函数，否则会有语法错误

- 子类构造函数中的 super 函数的作用是可以将子类实例化时获得的参数传入父类的构造函数之中。
- Super用于调用父类的方法

类的本质就是构造函数

ECMAScript 6 中基于类的面向对象相较于构造函数和原型对象的面向对象本质上是一样的，基于类的语法更为简洁，

未来的 JavaScript 中也都会是基于类的语法实现，当前阶段 先熟悉基于类的语法，

后面课程中会加强基于类语法的实践。

对象拷贝

拷贝不是直接赋值

- 深拷贝
- 浅拷贝

浅拷贝

只拷贝最外面层

Object.assign：方法可以实现

全部层拷贝

利用递归实现，后续还会学习方法

传智教育旗下高端IT教育品牌