博学谷
www.boxuegu.com

第七章 对象的扩展

一样的在线教育，不一样的教学品质

目录 Contents

- 属性与方法的简洁表示方式
- Object.assign( )方法
- Object.setPrototypeOf( )方法与Object.getPrototypeOf( )方法
- 对象扩展运算符

属性与方法的简洁表示方式

小节导学

采用传统方式创建对象，代码如下：

```
let userName = 'zhangsan';
let userAge = 18;
let person = {
    userName: userName,
    userAge: userAge
}
console.log(person);
```

说明：对象中的属性名和变量名一致，像这种情况，在ES6中可以简化。

属性与方法的简洁表示方式

采用ES6中简化的方式如下：

```
let userName = 'zhangsan’;

  let userAge = 18;

  let person = {
      userName,
      userAge
  }
  console.log(person);
```

说明：ES6中，如果**对象的属性名和变量名一致**，那么两者可以**合二为一，即属性简写**。

属性与方法的简洁表示方式

定义方法的传统形式，如下：

```
let userName = 'zhangsan';
        let userAge = 18;
        let person = {
            userName,
            userAge,
            sayHello: function() {
                console.log('你好')
            }
        }
        person.sayHello();
```

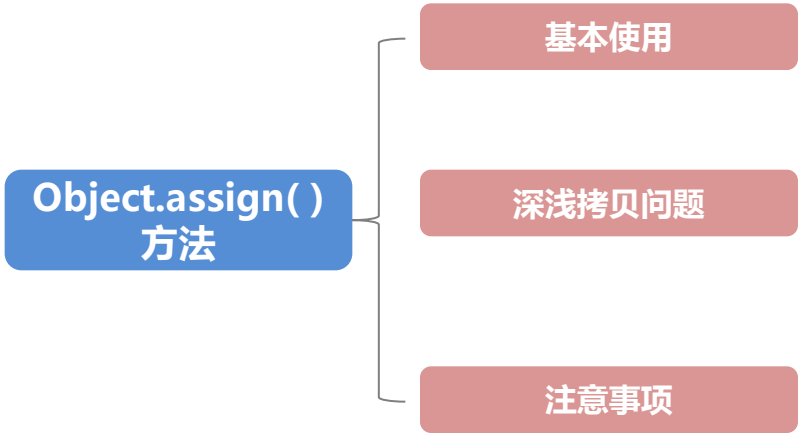
ES6中可方法可简化为：

```
let userName = 'zhangsan';
        let userAge = 18;
        let person = {
            userName,
            userAge,
            sayHello() {
                console.log('Hello');
            }
        }
        person.sayHello();
```

目录 Contents

- 属性与方法的简洁表示方式
- Object.assign( )方法
- Object.setPrototypeOf( )方法与Object.getPrototypeOf( )方法
- 对象扩展运算符

Object.assign( )方法



1.Object.assign( )方法基本使用

如何将一个对象的属性拷贝给另外一个对象？

传统方式，如下：

```
let obj1 = {
    name: 'zhangsan'
};
let obj2 = {
    age: 20
};
let obj3 = {};
for (let key in obj1) {
    obj3[key] = obj1[key];
}
for (let key in obj2) {
    obj3[key] = obj2[key];
}
console.log('obj3=', obj3);
```

优化：可使用Object.assign( )方法来实现；

1.Object.assign( )方法基本使用

◆ 将源对象的所有可枚举的属性复制到目标对象。

该方法至少需要两个对象作为参数，第一个参数是目标对象，后面的参数都是源对象。

只要有一个参数不是对象，就会抛出异常。

示例：

```
let target = {
    a: 1,
    b: 2
};
let source = {
    c: 3,
    d: 4
};
Object.assign(target, source);
console.log(target);
```

2.深浅拷贝问题

通过Object.assign( )方法，实现的拷贝，**只拷贝了属性的值，属于浅拷贝**。

```
let obj1 = {
        name: '张三',
        address: {
            city: '北京'
        }
    }
    let obj2 = {};
    Object.assign(obj2, obj1);
    obj2.address.city = "上海";
    console.log("obj1=", obj1);
    console.log("obj2=", obj2);
```

说明：对对象obj2的city属性值进行了修改，发现对应的对象obj1中的city属性值也发生了改变。

深拷贝：修改一个对象的属性值时，不会影响到另外一个对象的属性值。

3.注意事项

- 如果目标对象与源对象有同名属性，那么，后面的属性会覆盖前面的属性。
- 不可枚举的属性不会被复制。

目录 Contents

- 属性与方法的简洁表示方式
- Object.assign( )方法
- Object.setPrototypeOf( )方法与Object.getPrototypeOf( )方法
- 对象扩展运算符

setPrototypeOf( )方法与getPrototypeOf( )方法

Object.setPrototypeOf( )方法：

该方法的**作用**：**设置一个对象的prototype**，也就是用来**设置原型对象的方法**。

ES6写法，如下：

```
let obj = {
    name: 'zhangsan'
}
let obj1 = {};
Object.setPrototypeOf(obj1, obj)
console.log(obj1.name);
```

输出的值为：‘zhangsan’，也就是obj1对象继承了obj对象。

ES5写法，如下：

```
let obj = {
    name: 'lisi'
}
let obj1 = {};
obj1.__proto__ = obj;
console.log(obj1.name);
```

setPrototypeOf( )方法与getPrototypeOf( )方法

Object.getPrototypeOf( )方法：

该方法的作用：读取一个对象的prototype对象。

```
function Test() {

 }
 let test = new Test();
 console.log(Object.getPrototypeOf(test) === Test.prototype);
```

博学谷
www.boxuegu.com

目录 Contents

- 属性与方法的简洁表示方式
- Object.assign( )方法
- Object.setPrototypeOf( )方法与Object.getPrototypeOf( )方法
- 对象扩展运算符

对象扩展运算符

扩展运算符可以**取出对象的属性**，**复制到其它对象中**。

示例：

```
let obj = {
    a: 1,
    b: 2
}
let obj1 = {...obj
};
console.log("obj1=", obj1);
```

扩展运算符合并对象：

```
let obj = {
        a: 3,
        b: 2
    };
    let obj1 = {
        c: 1,
        d: 4
    }
    let obj2 = {...obj,
        ...obj1
    };
    console.log(obj2);
```

一样的在线教育，不一样的教学品质