Symbol

Symbol用于防止属性名冲突而产生的，比如向第三方对象中添加属性时。

Symbol 的值是唯一的，独一无二的不会重复的

#基础知识

#Symbol

```
let hd = Symbol();
let edu = Symbol();
console.log(hd); //symbol
console.log(hd == edu); //false

```

Symbol 不可以添加属性

```
let hd = Symbol();
hd.name = "后盾人";
console.log(hd.name);

```

#描述参数

可传入字符串用于描述Symbol，方便在控制台分辨Symbol

```
let hd = Symbol("is name");
let edu = Symbol("这是一个测试");

console.log(hd); //Symbol(is name)
console.log(edu.()); //Symbol(这是一个测试)

```

传入相同参数Symbol也是独立唯一的，因为参数只是描述而已，但使用 `Symbol.for`则不会

```
let hd = Symbol("后盾人");
let edu = Symbol("后盾人");
```

```
console.log(hd == edu); //false

```

使用`description`可以获取传入的描述参数

```
let hd = Symbol("后盾人");
console.log(hd.description); //后盾人

```

#Symbol.for

根据描述获取Symbol，如果不存在则新建一个Symbol

- 使用Symbol.for会在系统中将Symbol登记
- 使用Symbol则不会登记

```
let hd = Symbol.for("后盾人");
let edu = Symbol.for("后盾人");
console.log(hd == edu); //true

```

#Symbol.keyFor

`Symbol.keyFor` 根据使用`Symbol.for`登记的Symbol返回描述，如果找不到返回undefined 。

```
let hd = Symbol.for("后盾人");
console.log(Symbol.keyFor(hd)); //后盾人

let edu = Symbol("houdunren");
console.log(Symbol.keyFor(edu)); //undefined

```

#对象属性

Symbol 是独一无二的所以可以保证对象属性的唯一。

- Symbol 声明和访问使用

`[]`（变量）形式操作

- 也不能使用

`.` 语法因为 `.`语法是操作字符串属性的。

下面写法是错误的，会将`symbol` 当成字符串`symbol`处理

```
let symbol = Symbol("后盾人");
let obj = {
  symbol: "hdcms.com"
};
console.log(obj);

```

正确写法是以`[]` 变量形式声明和访问

```
let symbol = Symbol("后盾人");
let obj = {
  [symbol]: "houdunren.com"
};
console.log(obj[symbol]); //houdunren.com

```

#实例操作

#缓存操作

使用`Symbol`可以解决在保存数据时由于名称相同造成的耦合覆盖问题。

```
class Cache {
  static data = {};
  static set(name, value) {
    this.data[name] = value;
  }
  static get(name) {
    return this.data[name];
  }
}

let user = {
  name: "后盾人",
  key: Symbol("缓存")
};
```

```
15
16 let cart = {
17   name: "购物车",
18   key: Symbol("购物车")
19 };
20
21 Cache.set(user.key, user);
22 Cache.set(cart.key, cart);
23 console.log(Cache.get(user.key));
24
```

#遍历属性

Symbol 不能使用 `for/in`、`for/of` 遍历操作

```
1 let symbol = Symbol("后盾人");
2 let obj = {
3   name: "hdcms.com",
4   [symbol]: "houdunren.com"
5 };
6
7 for (const key in obj) {
8   console.log(key); //name
9 }
10
11 for (const key of Object.keys(obj)) {
12   console.log(key); //name
13 }
14
```

可以使用 `Object.getOwnPropertySymbols` 获取所有`Symbol`属性

```
1 ...
2 for (const key of Object.getOwnPropertySymbols(obj)) {
3   console.log(key);
4 }
5
```

也可以使用 `Reflect.ownKeys(obj)` 获取所有属性包括`Symbol`

```
...
for (const key of Reflect.ownKeys(obj)) {
  console.log(key);
}
...

```

如果对象属性不想被遍历，可以使用`Symbol`保护

```
const site = Symbol("网站名称");
class User {
  (name) {
    this[site] = "后盾人";
    this.name = name;
  }
  getName() {
    return `${this[site]}-${this.name}`;
  }
}
const hd = new User("向军大叔");
console.log(hd.getName());
for (const key in hd) {
  console.log(key);
}

```