包装对象

原始类型的值，数值、布尔值、字符串，在一定条件下也能转换为对象，也就是原始类型的包装对象。包装对象是指原始对象相对应的 `Number` `String` `Boolean` 三个原生对象。这三个原生对象可以把原始类型的值包装成对象。

```
let v1 = new Number(123);
let v2 = new String("abc");
let v3 = new Boolean(false);

typeof v1; // "object"
typeof v2; // "object"
typeof v3; // "object"
Object.prototype.toString.call(new Number(123));    // '[object Number]'
Object.prototype.toString.call(new String('123'));  // '[object String]'
Object.prototype.toString.call(new Boolean(false)); // '[object Boolean]'
```

包装对象的设计目的，首先是使得“对象”这种类型可以覆盖 JavaScript 所有的值，整门语言有一个通用的数据模型，其次是使得原始类型的值也有办法调用自己的方法。

`Number`、`String` 和 `Boolean` 这三个原生对象，作为构造函数使用（带有 `new`）时，可以将原始类型的值转为对象；作为普通函数使用时（不带有 `new`），可以将任意类型的值，转为原始类型的值。

1. 实例方法

包装对象共同具有从 `Object` 对象继承的方法：`valueOf()` 和 `toString()`。

1.1. valueOf()

`valueOf()` 方法返回包装对象实例对应的原始类型的值。

```
new Number(123).valueOf() === 123;     // true
new String("abc").valueOf() === "abc"; // true
new Boolean(true).valueOf() === true;  // true
```

1.2. toString()

`toString()` 方法返回对应的字符串形式。

```
new Number(123).toString();   // "123"
new String("abc").toString(); // "abc"
new Boolean(true).toString(); // "true"
```

2. 原始类型和实例对象的自动转换

某些场合，原始类型的值会自动当作包装对象调用，即调用包装对象的属性和方法。这时，JavaScript 引擎会自动将原始类型的值转为包装对象实例，并在使用后立刻销毁实例。比如，字符串可以调用 `length` 属性，返回字符串的长度。

```
"abc".length; // 3
```

上例中，`abc` 是一个字符串，本身不是对象，不能调用 `length` 属性。JavaScript 引擎自动将其转为包装对象，在这个对象上调用 `length` 属性。调用结束后，这个临时对象就会被销毁。这就叫原始类型与实例对象的自动转换。

```
let s = "abc";
s.length; // 3

// 等价于
let sObj = new String(s);
sObj; // {0: "a", 1: "b", 2: "c", length: 3, [[PrimitiveValue]]: "abc}
sObj.length; // 3
```

上例字符串自动转换成包装对象 `sObj` 后，包含了 `length`，故原字符串有 `length` 属性。

自动转换生成的包装对象是只读的，无法修改。所以，字符串无法添加新属性。

```
let s = "123456";
s.f = 2;
s.f; // undefined
```

3. 自定义方法

除了原生的实例方法，包装对象还可以自定义方法和属性，供原始类型的值直接调用。

```
Number.prototype.double = function () {
  return this.valueOf() * 2;
};
(123).double(); // 246

String.prototype.double = function () {
  return this.valueOf() + this.valueOf();
};
"abc".double(); // "abcabc"
```

上例在 `String` 和 `Number` 这两个对象的原型上面，分别自定义了一个方法，从而可以在所有实例对象上调用。最后一行的 `123` 外面必须要加上圆括号，否则后面的点运算符（`.`）会被解释成小数点。