基本引用类型

单例内置对象

ECMA-262对内置对象的定义：
任何由 ECMAScript 实现提供、与宿主环境无关，并在 ECMAScript程序开始执行时就存在的对象
开发者不必显式实例化内置对象，因为已经实例化好了

Global

Global式ES最特别的对象，代码不会显式访问它
ECMA-262 规定 Global 对象为一种兜底对象
所针对的是不属于任何对象的属性和方法

事实上不存在全局变量或全局函数，全局作用域中定义的变量和函数，都会变成Global对象的属性

isNaN()、isFinite()、parseInt()和 parseFloat()，实际上都是 Global 对象的方法

- URL编码方法
  encodeURL()
  encodeURIComponent()
  decodeURI()
  decodeURIComponent()

```
let uri = "http://www.wrox.com/illegal value.js#start";
// "http://www.wrox.com/illegal%20value.js#start"
console.log(encodeURI(uri));
// "http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.js%23start"
console.log(encodeURIComponent(uri));
```

  encodeURL不会编码属于URL组件的特殊字符，encodeURIComponent编码所有非标准字符

```
let uri = "http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.js%23start";
// http%3A%2F%2Fwww.wrox.com%2Fillegal value.js%23start
console.log(decodeURI(uri));
// http:// www.wrox.com/illegal value.js#start
console.log(decodeURIComponent(uri));
```

- eval()方法
  ES最强大的方法。
  一个完整的ES解释器

```
eval("console.log('hi')");
上面这行代码的功能与下一行等价：
```

```
console.log("hi");
```

解释完，把解释结果插入代码中
eval中的变量不会提升，严格模式不能被外部访问

- Globe对象属性

属　性	说　明
undefined	特殊值 undefined
NaN	特殊值 NaN
Infinity	特殊值 Infinity
Object	Object 的构造函数
Array	Array 的构造函数
Function	Function 的构造函数
Boolean	Boolean 的构造函数
String	String 的构造函数
Number	Number 的构造函数
Date	Date 的构造函数
RegExp	RegExp 的构造函数
Symbol	Symbol 的伪构造函数
Error	Error 的构造函数
EvalError	EvalError 的构造函数
RangeError	RangeError 的构造函数
ReferenceError	ReferenceError 的构造函数
SyntaxError	SyntaxError 的构造函数
TypeError	TypeError 的构造函数
URIError	URIError 的构造函数

- window对象
  虽然ECMA-262没有规定直接访问Global对象的方法，但是浏览器将window对象实现为Global
  所以全局作用域声明的变量和函数都变成了window属性

  另外一种获取Global的方法

```
let global = function() {
 return this;
}();
```

Math

- math对象属性

属　性	说　明
Math.E	自然对数的基数 e 的值
Math.LN10	10 为底的自然对数
Math.LN2	2 为底的自然对数
Math.LOG2E	以 2 为底 e 的对数
Math.LOG10E	以 10 为底 e 的对数
Math.PI	π 的值
Math.SQRT1_2	1/2 的平方根
Math.SQRT2	2 的平方根

- min() max()
- 舍入方法
  - Math.ceil() 始终向上舍入为最接近的整数
  - Math.floor() 方法始终向下舍入为最接近的整数
  - Math.round() 方法执行四舍五入
  - Math.fround() 方法返回数值最接近的单精度（32 位）浮点值表示

```
console.log(Math.ceil(25.9)); // 26
console.log(Math.ceil(25.5)); // 26
console.log(Math.ceil(25.1)); // 26
console.log(Math.round(25.9)); // 26
console.log(Math.round(25.5)); // 26
console.log(Math.round(25.1)); // 25
console.log(Math.fround(0.4)); // 0.4000000059604645
console.log(Math.fround(0.5)); // 0.5
console.log(Math.fround(25.9)); // 25.899999618530273
console.log(Math.floor(25.9)); // 25
console.log(Math.floor(25.5)); // 25
console.log(Math.floor(25.1)); // 25
```

- random()
  Math.random()不接受传参，返回[0,1)
  因为一直返回小数，需要Math.floor

```
let num = Math.floor(Math.random() * 10 + 1);
// [1,10]

let num = Math.floor(Math.random() * 9 + 2);
// [2,10]
```

- 其他方法

`Math.abs(x)`	返回 x 的绝对值
`Math.exp(x)`	返回 `Math.E` 的 x 次幂
`Math.expm1(x)`	等于 `Math.exp(x) - 1`
`Math.log(x)`	返回 x 的自然对数
`Math.log1p(x)`	等于 `1 + Math.log(x)`
`Math.pow(x, power)`	返回 x 的 *power* 次幂
`Math.hypot(...nums)`	返回 *nums* 中每个数平方和的平方根
`Math.clz32(x)`	返回 32 位整数 x 的前置零的数量
`Math.sign(x)`	返回表示 x 符号的 `1`、`0`、`-0` 或`-1`
`Math.trunc(x)`	返回 x 的整数部分，删除所有小数
`Math.sqrt(x)`	返回 x 的平方根
`Math.cbrt(x)`	返回 x 的立方根
`Math.acos(x)`	返回 x 的反余弦
`Math.acosh(x)`	返回 x 的反双曲余弦
`Math.asin(x)`	返回 x 的反正弦
`Math.asinh(x)`	返回 x 的反双曲正弦
`Math.atan(x)`	返回 x 的反正切
`Math.atanh(x)`	返回 x 的反双曲正切
`Math.atan2(y, x)`	返回 y/x 的反正切
`Math.cos(x)`	返回 x 的余弦
`Math.sin(x)`	返回 x 的正弦
`Math.tan(x)`	返回 x 的正切

总结

JavaScript 中的对象称为引用值，几种内置的引用类型可用于创建特定类型的对象。

- 引用值与传统面向对象编程语言中的类相似，但实现不同
- Date 类型提供关于日期和时间的信息，包括当前日期、时间及相关计算。
- RegExp 类型是 ECMAScript 支持正则表达式的接口，提供了大多数基础的和部分高级的正则表达式功能。

JavaScript 比较独特的一点是，函数实际上是 Function 类型的实例，也就是说函数也是对象。因为函数也是对象，所以函数也有方法，可以用于增强其能力。

由于原始值包装类型的存在，JavaScript 中的原始值可以被当成对象来使用。有 3 种原始值包装类型：Boolean、Number 和 String。它们都具备如下特点。

- 每种包装类型都映射到同名的原始类型。
- 以读模式访问原始值时，后台会实例化一个原始值包装类型的对象，借助这个对象可以操作相应的数据。
- 涉及原始值的语句执行完毕后，包装对象就会被销毁。
  当代码开始执行时，全局上下文中会存在两个内置对象：Global 和 Math。其中，Global 对象在大多数 ECMAScript 实现中无法直接访问。不过，浏览器将其实现为 window 对象。所有全局变量和函数都是 Global 对象的属性。Math 对象包含辅助完成复杂计算的属性和方法。