String

- ASCII
- Unicode
- UCS
- GB
  - GB2312
  - GBK(GB13000)
  - GB18030
- ISO-8859
- BIG5

String———Encoding

- UTF
  - UTF-8 (8个比特位表示一个字符)
  - UTF-16 (16个比特位表示一个字符)

```
function stringToByte(str) {
    var bytes = new Array();
    var len, c;
    len = str.length;
    for(var i = 0; i < len; i++) {
        c = str.charCodeAt(i);
        if(c >= 0x010000 && c <= 0x10FFFF) {
            bytes.push(((c >> 18) & 0x07) | 0xF0);
            bytes.push(((c >> 12) & 0x3F) | 0x80);
            bytes.push(((c >> 6) & 0x3F) | 0x80);
            bytes.push((c & 0x3F) | 0x80);
        } else if(c >= 0x000800 && c <= 0x00FFFF) {
            bytes.push(((c >> 12) & 0x0F) | 0xE0);
            bytes.push(((c >> 6) & 0x3F) | 0x80);
            bytes.push((c & 0x3F) | 0x80);
        } else if(c >= 0x000080 && c <= 0x0007FF) {
            bytes.push(((c >> 6) & 0x1F) | 0xC0);
            bytes.push((c & 0x3F) | 0x80);
        } else {
            bytes.push(c & 0xFF);
```

```
        }
    }
    return bytes;
}


 function byteToString(arr) {
    if(typeof arr === 'string') {
        return arr;
    }
    var str = '',
        _arr = arr;
    for(var i = 0; i < _arr.length; i++) {
        var one = _arr[i].toString(2),
            v = one.match(/^1+?(?=0)/);
        if(v && one.length == 8) {
            var bytesLength = v[0].length;
            var store = _arr[i].toString(2).slice(7 - bytesLength);
            for(var st = 1; st < bytesLength; st++) {
                store += _arr[st + i].toString(2).slice(2);
            }
            str += String.fromCharCode(parseInt(store, 2));
            i += bytesLength - 1;
        } else {
            str += String.fromCharCode(_arr[i]);
        }
    }
    return str;
}
```

Object in JavaScript

在js运行时，原生对象的描述方法只需要关系原型（prototype）和属性

Object 唯一性用内存地址的唯一性来表示

原型链：如果对象不存在所寻找的对象就会从原型中去找，原型的原型不是空，会继续寻找该属性，从而产生的链式行为。一直找到原型顶端为null的空对象为止。

属性：（用属性来统一抽象对象状态和行为）

- JS属性是一个kv(key-value)对，根据key找到value。
- key值有两种类型：Symbol和String。Symbol在内存里创建后只能通过变量去引用它，不能创建两个一样的Symbol，很好是实现了属性方位的权限控制。
- 属性值：有两种形态，一种是数据属性，另一种是访问器属性。一般来说，数据属性用来描述状态，访问器属性用来描述行为。数据属性中如果存在函数，也可以描述行为。（Data Property: [[value]]、writable、enumerable、configurale; Accesser Property: get、set、enumerable、configurale）

Object Api/Grammar

- {},[] Object.defineProperty （通过语法去创建属性，定义属性和访问属性和定义行的属性，以及去改变属性的特征值）
- Object.create() / Object.setPrototypeOf / Object.getPrototypeOf (基于原型的描述对象的方法，通过Object.create()指定原型的情况下创建对象)
- new / class / extends （基于类的方式去描述对象）
- new / function / property

Function Object

function 是一个带[[call]]方法的对象（内置行为）

Special Object

- Array [[length]]
- Objcet.prototype [[setPrototypeOf]]
- ...

Host Object

Object [[call]]/[[construct]]

window 对象

特殊对象以及方法等

```
Array：Array 的 length 属性根据最大的下标自动发生变化，可以使用循环来遍历属性。
```

Object.prototype：作为所有正常对象的默认原型，不能再给它设置原型了。

String：为了支持下标运算，String 的正整数属性访问会去字符串里查找。

Arguments：arguments 的非负整数型下标属性跟对应的变量联动。

模块的 namespace 对象：特殊的地方非常多，跟一般对象完全不一样，尽量只用于 import 吧。
类型数组和数组缓冲区：跟内存块相关联，下标运算比较特殊。

bind 后的 function：跟原来的函数相关联。

window对象：在全局作用域中声明的变量、函数都是window对象的属性和方法。

this对象：this对象是在运行时基于函数的执行环境绑定的：在全局函数中，this等于window；当函数被作为某个对象的方法调用时，this等于那个对象。

Global对象：
  1)所有在全局作用域内定义的属性和方法，都是Global对象的属性。
  2)Global对象不能直接使用，也不能用new运算符创建。
  3)Global对象在JavaScript引擎被初始化时创建，并初始化其方法和属性。
  4)浏览器把Global对象作为window对象的一部分实现了，因此，所有的全局属性和函数都是window对象的属性和方法。
Global对象的属性；

Object	构造函数Object
Function	构造函数Function
Date	构造函数Date
Array	构造函数Array
RegExp	构造函数RegExp
Error	构造函数Error
String	构造函数String
Boolean	构造函数Boolean
Number	构造函数Number
undefined	特殊值undefined
NaN	特殊值NaN

Global对象的方法：

  encodeURI()：
    1)对整个URI进行编码。
    2)只对空格进行编码，不会对冒号、正斜杠、问号、井号等进行编码。
    3)编码后的结果：除了空格被替换成了%20之外，其它的都没有发生变化。
    4)对应的解码方法：decodeURI()

  encodeURIComponent()：
    1)对部分URI(指除去协议、主机地址、端口后的部分)进行编码。

2)对任何非标准字符进行编码。
3)对应的解码方法：decodeURIComponent()

URI编码方法说明：
1)有效的URI中不能包含某些字符(例如：空格等)。
2)用特殊的UTF-8编码替换所有无效的字符，从而让浏览器能够接受和理解。

eval()：当解析器发现代码中调用eval()方法时，它会将传入的参数当作实际的JavaScript语句来解析。