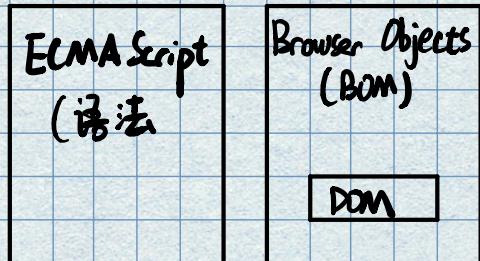


Java Script

基于对象和事件驱动的客户端脚本语言

组成



语法规则

// 单行注释

/* */ 多行

; 分号

语法

- ECMAScript 区分大小写

标识符

- 变量、函数、属性的名字、函数的参数

- 命名规则

- (1) 字母、数字、_、\$

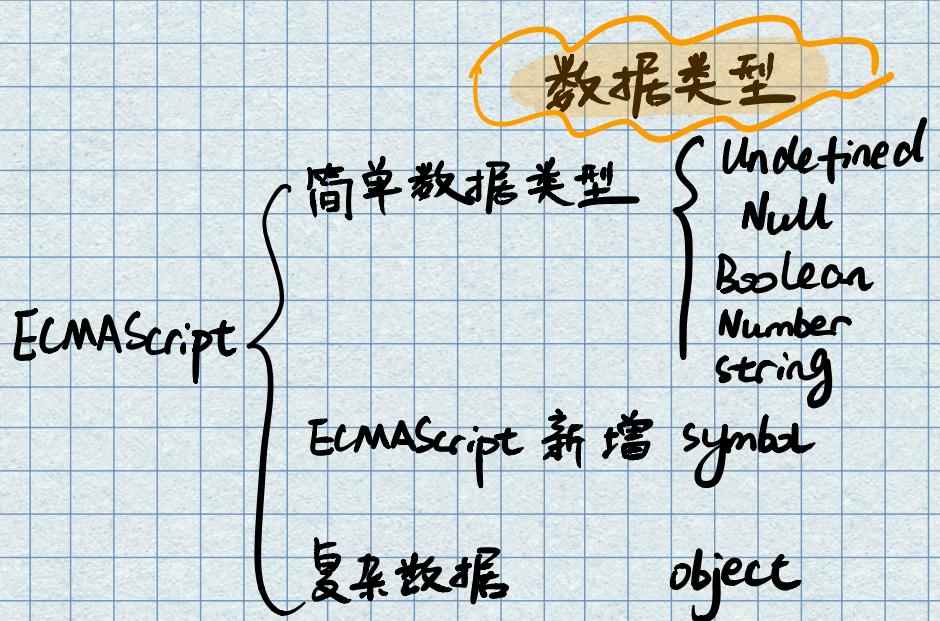
- (2) 不能数字开头

- (3) 不能使用关键字、保留字

变量

- 声明 var 变量名； var a,b,c;

- 赋值 = " ";



typeof 操作符

- 检测变量类型
- 语法： `typeof 变量` / `typeof(变量)`
- 返回值： `string` 类型
`string, number, boolean, object, undefined, function`

`console.log()` // 在控制台中打印

`console.log(typeof ...)`

undefined

- 未定义，没有赋值

null

- 空对象指针
- 变量在将来准备存放对象，初始化 `null`
- `undefined == null` true
- `undefined` 派生自 `null`

number

- 整数和浮点数

NaN

- Not A Number

特殊数值

- 任何 NaN 的操作
都会返回 NaN

- NaN 与任何值不相等
包括本身

isNaN()

- isNaN(n)
- 检测 n 是否是“非数值”
- 返回值: boolean
- var id = "16"
console.log(isNaN(id)) \Rightarrow false
(将 id 先转换成数)

数值转换

- 把 NaN 转换为数值

{ Number() id = Number(id);
 转不了，返回 NaN

parseInt()

数字开头

parseFloat()

第一个小数点有效

始终忽略前导的零

} 把字符串转成数值

String

- 字符串 " " ''
- `toString()` 与 `String()`
- `str.toString()`
- 将 str 转为字符串
- 返回：str 的一个副本
- str 是要转换的内容，可以是数值，Boolean，对象，string
- 不知道是不是 null 或 undefined
可以用 `String()`，它可以转换任何类型

Boolean

- 转换 `Boolean()`
- 除“空”以外，其它都是 true
 - 0 → false
 - null, undefined → false

算术操作符

递增和递减

- `++a`
返回递增之后的 a
- `a++`
先返回原 a，再返回递增之后的 a

$$5 + "5" = 55$$

其它操作符

- $a += 5 \Rightarrow a = a + 5$
 $b \% = 4 \Rightarrow b = b \% 4$

• 比较操作符

$==$ 全等 (比较值的同时, 类型是否一样)

$!=$ 不全等

• 三元操作符

语法：条件 ? 执行1 : 执行2

逻辑操作符

与

- $5 \&& 6$ 返回 6 (隐式类型)
 $0 \&& 5$ 返回 0

有一个是 null, NaN, undefined, 则返回它们

或

- $||$
- "hello" || 0 返回 hello
99 || 0 || "abc" 返回 99
- "" || 88 || true \Rightarrow 88
"" || 0 || "abc" \Rightarrow abc

非

• !

都会返回 true · false

分支语句

if语句

- if () {
 ...;
}
- if () {
 ①
}
else {
 ②
}
- if () {}
else if () {}
else {}

switch

```
switch ( ) {  
    case ... : ....  
    break ;  
    case ... : ...  
    break ;  
    default : statement }
```

alert()

弹出警告对话框

alert (...)

prompt()

弹出输入框

① 点击确定，返回值

② 取消，返回 null

string.length

获取 string 的长度

返回 number

获取星期

new Date().getDay()

返回：number (0 - 6)

循环语句

for

- 语法

```
for ( 1 ; 2 ; 3 ) {  
    ...  
}
```

while

- while () {

```
}
```

do...while

- do {

```
    ...  
} while ( )
```

break

- 立即退出

continue

- 退出本次循环

继续下一次

函数

定义

```
function functionName( [arg0, arg1, ...] ) {  
    statements  
}
```

- 命名规范

函数参数
[] 可有可无

调用

```
functionName( [arg0 ...] )
```

返回值

```
return;  
return ...;
```

调用以及要用 变量接收

Arguments

- 在函数体内用 arguments 对象来访问这个数组参数
⇒ arguments 对象只是与数组类似，并不是 array 实例
⇒ 参数传到函数内部用 arguments 对象来管理
类似一个数组
⇒ arguments.length

$\Rightarrow \text{arguments} []$ // 从0开始的正整数

内置对象

Array 数组

创建数组

1. Array 构造函数

`new Array()`

- ① 数组长度
- ② 具体值

2. 数组字面量表示

`var nums = [1, 3, 6, 9];`

可以是不同类型

读写

[] 从0 开始

长度

`array.length`

* 可以移除末尾的值

数组方法

`push()`

`anyobject.push(1, 2, 3...)`

顺序添加到尾部

返回：数组新长度

unshift()

添加到开头

pop()

删除最后一个

shift()

删除第一个

返回：被删除的元素

join()

arrayobject.join(separator)

把数组中所有元素放入一个字符串

分隔符（默认，）
("")

返回：字符串

reverse()

颠倒元素顺序

sort() 排序

arrayobject.sort(sortby)

• 按照字符串（第一位）

• 接收一个比较函数比较大小

arr.sort(function(a,b){return b-a}); 降序

concat()

连接多个数组

arr1.concat(arr2) 连接 1 和 2

b = [].concat(a)

slice() 从数组中返回选定的元素

arrayobject.slice(start, end)

↓
从何处开始
参数：长度 + 负
(位置)

↓
截取至 end - 1

splice

删除

arrayobject.splice(index, count)

→ 删的数量
0: 不删

删除从 index 开始的 0 个或多个元素

不设：从 index 开始
全删

返回：被删除的数组

插入

arrayobject.splice(index, 0, item...)

在指定位置插入

↓
删除 0 ↓
插入 index 前面

替换

arrayobject.splice(index, count, item...)

返回：从原数组中删除的项

indexOf()

arrayObject.indexOf(searchValue, startIndex)

从(开始)向后查找

返回 { 查找的项在数组的位置 , number
没找到 : -1 }

lastIndexOf

从(末尾)查找

自己一个方法实现 indexOf

```
function ArrayIndexOf(arr, value) {
    for (var i=0; i<arr.length; i++) {
        if (arr[i] == value) {
            return i;
        }
    }
    return -1;
}
```

String

charAt()

stringObject.charAt(index)

返回：index 位置的字符本身

indexOf()

stringObject.indexOf('O')

搜索 给定的子字符串，返回位置

没找到：-1

lastIndexOf()

末尾开始

slice() 截取一段

.slice(start, end)

substring() 与 slice 一样

• 区别：参数为负，自动转为0

(2, -5) 自动
转换
(0, 2)

substr()

.substr(start, len)

截取字符总段，为负，返回空

split()

stringObject . split (separator)

把字符串分割成数组

⇒ split ('')

空，每一个字母都拆分

返回： Array

2016/05/05 ⇒ ["2016", "05", "05"] 按 "/" 分开

replace()

RegExp 对象

stringObject.replace (regexp / substr, replacement)

在字符串中用一些字符替换另一些字符，
或替换一个与正则表达式匹配的子串。
一个字符串值

返回： string

toUpperCase(), toLowerCase()

字符串转换大小写

Math 对象

Math.min (num1, 2, 3...) 求最小值

Math.max (num1, 2, 3...) 求最大值

Math.ceil (num) 向上取整，大于 num

Math.floor (num) 向下取整

Math. round (num) 四舍五入

Math. abs (num) 求绝对值

Math. random () $0 \leq o < 1$ 的一个随机数

n到m之间的随机整数

random = Math. floor (Math. random () * (m-n+1) + n);

Date 对象

new Date();

创建一个日期时间对象

返回：日期时间，不传参 \Rightarrow 当前

.getFullYear() ; .getMonth() 0~11 ; .getDate() ; .getDay() 0~6 ;

.getHours() ; .getMinutes() ; .getSeconds() ; .getTime();
毫秒

设置 get \Rightarrow set

DOM

DOM Document Object Model 文档对象模型

DOM 查找方法

`document.getElementById("id")`

返回对拥有指定 ID 的第一个对象的引用

返回：DOM 对象

* id 为 DOM 元素上 id 属性的值

`document.getElementsByTagName("tag")`

返回一个对所有 tag 标签引用的集合

返回：数组

设置元素样式

`ele.style.styleName = styleValue`

Dom 对象
设置 ele 的 CSS 样式
要设置的样式名

font-size
少
驼峰 fontSize

`innerHTML`

`ele.innerHTML`

返回 ele 开始和结束标签之间的 HTML

`ele.innerHTML = "html"`

设置 ele 开始和结束标签之间的 HTML 为 ⇒ html

className

ele.className 返回 class 属性

ele.className = "cls" 设置，替换本身 class

获取属性

ele.getAttribute ("attribute")

要获取的 html 属性 (id, type)

获取 ele 元素的 attribute 属性

自带 : ele.id, type, value ...

设置属性

ele.setAttribute ("attribute", value)

属性名 属性值

删除属性

ele.removeAttribute ("attribute")

DOM 事件

事件是 html 文档或浏览器窗口中发生的一些特定的交互瞬间。

HTML 事件

直接在元素标签内添加

<tag 事件 = "执行脚本" > </tag>

this 对 DOM
对象的引用

onclick : 鼠标点击触发

onmouseover : 鼠标滑过

onmouseout : 鼠标离开

onload : 页面加载时

DOM 0 级事件

(获取 Html 元素). 事件 = 执行脚本

ele. 事件 = 执行脚本

* function abc() {}

调用时 btn.onclick = abc (不要加括号)

事件类型

onload

window. onload 页面加载时

等所有内容加载完 触发 script

onfocus 和 onblur

用于 input 标签, type 为 text, password, textarea

this.value ⇒ 用于获取表单元素的值

onchange

域的内容改变时发生 , 单选框, 复选框, 下拉菜单 ...

获取 select 下拉菜单的值 : ① this.value

② menu. options [menu.selectedIndex]. value

document. body. ...

onsubmit 表单中确认按钮被点击时发生

* 不是加在按钮上，而是表单上绑定

拖动事件

onmousedown + onmousemove + onmouseup
按下 移动 松开

onresize

调整浏览器窗口大小时触发

onscroll

拖动滚动条滚动时

} window. --

键盘事件

onkeydown



onkeypress (只响应字母与数字)



onkeyup

keyCode : 返回事件触发的键的值的字符代码

event : 事件状态，触发 event 对象的元素、鼠标位置等

event. keycode

BOM

browser object model 浏览器对象模型

window 对象

① js 访问浏览器窗口的一个接口

② global 对象 → 全局对象

声明一个全局变量，方法

window.name = ...

window.name = function() ...

- window.alert("...")
- window.confirm("...") {
 - 确定 → 返回 true
 - 取消 → 返回 false
- window.prompt("...", "...") {
 - 取消 → null
 - 确认 → 输入文本
- window.open(page URL, name, parameters)
 - 窗口参数
- window.close

(方法)

定时器 setTimeout

超时调用：`setTimeout (code, millisec)`
在指定的毫秒数后调用函数或表达式

⇒ 返回一个 ID，用来取消超时调用
`clearTimeout (id)`

定时器 setInterval

间歇调用 `setInterval (code, millisec)`
每隔指定时间执行一次代码
`clearInterval (id)`

location 对象

`location.href` 返回当前页面完整 URL

`location.hash` 返回 URL 中 hash (# 后零或多个字符)
不包含则返回空字符串
`id="top"` `location.hash = "# top"`

`location.host` 返回服务器名称和端口号

`location.hostname` 返回不带端口号的服务器名称

`location.pathname` 返回 URL 中的目录或文件名

`location.port` 返回 URL 中端口号

`location.protocol` 返回页面使用的协议

`location.search` 返回 URL 的查询字符串。以问号开头

方法

改变 URL、跳转页面 `location.href = ...`, 有历史记录

`location.replace(url)`

不能后退，没有历史记录

`location.reload()`

重新加载当前页面，从缓存中加载

`location.reload(true)` 从服务器加载

history 对象

历史记录

`history.back()` 回到历史记录的上一步
相当于 `history.go(-1)`

`history.forward()` 下一步

相当于 `history.go(1)`

screen 对象

`screen.availWidth` 返回可用的屏幕宽度

`screen.availHeight` ----- 高度

`window.innerWidth` } 窗口
`innerHeight` }

Navigator

userAgent : 识别浏览器名称、版本、引擎，操作系统