# 复习

正则表达式：

/表达式/

数据类型：引用数据类型

（）：分组，用于除了多个重复出现的字符

分组反向引用：匹配组再次使用， 每一个分组按照顺序进行编码1,2,3,……

在正则表达式中再次使用： \编码

在正则表达式外再次使用： $编码

replace(a,b) 用b替换a

b还可以书写成match函数。

|  |
| --- |
| 1. replace(//, function (match, $1,$2……) { 2. return 返回值; 3. }) |

[]:字符集 将一些可能性字符都书写在中括号中，对应一个匹配

{}:量词

{n} n次

{n,m} n-m次

{n,} >=n次

+：>=1次

\*: 任意次

? : 0或者1次

. :除了回车和换行之外所有的字符

\w: 单词字符（字母，数字，下划线）

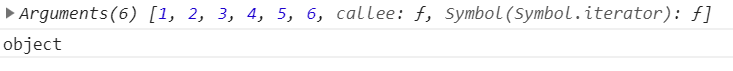
\d: 数字

\s:空白字符

# arguments

函数内部的类数组对象。

arguments由函数实际参数组成。



读取：通过索引值进行

|  |
| --- |
| 1. arguments[0] |

设置：使用=

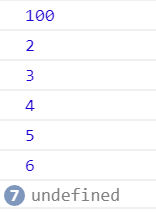
|  |
| --- |
| 1. arguments[0] = 100 |

length属性：

|  |
| --- |
| 1. arguments.length |

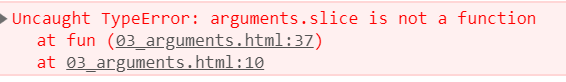
arguments只能通过length属性强制将类数组对象拉长。没有赋值undefined

|  |
| --- |
| 1. // 只能通过length强制将类数组拉长 2. arguments.length = 13; 3. console.log(arguments.length); 4. // 类数组也可以通过length进行遍历 5. for (var i = 0; i <= arguments.length - 1; i ++) { 6. console.log(arguments[i]); 7. } |



arguments并不能使用数组方法 slice(start,end) 从start开始截取到end(不含有end)

|  |
| --- |
| 1. console.log(arguments.slice(2,5)) |



函数重载：除了js其他语句都有重载现象（函数名相同，形参个数不同，表示不同函数）

js没有重载现象，只要函数名相同，后面函数层叠掉前面函数。

|  |
| --- |
| 1. // 函数调用 2. fun(); //2 3. function fun(a) { 4. console.log(1); 5. } 6. function fun(b,c) { 7. console.log(2); 8. } |

利用arguments模拟函数重载现象，根据不同参数，实现函数不同功能

如果用户传递2参数，实现参数相加

如果用户传递3参数，实现前两个数较大值和第三个相加

如果是其他参数，报错

|  |
| --- |
| 1. /\* 2. 如果用户传递2参数，实现参数相加 3. 如果用户传递3参数，实现前两个数较大值和第三个相加 4. 如果是其他参数，报错 5. 函数参数：3个数字 6. 返回值：加和结果 7. \*/ 8. function sum(a,b,c) { 9. // 条件分支语句 10. if (arguments.length === 2) { 11. return a + b; 12. }else if (arguments.length === 3) { 13. // 三元：二选一 三元运算符优先级低于数学运算符 提升优先级() 14. return (a > b ? a : b) + c; 15. }else { 16. return "参数有误"; 17. } 18. } 19. // 调用 20. console.log(sum(1,5)); 21. console.log(sum(1,5,10)); 22. console.log(sum()); |

# 二、IIFE

IIFE是一个缩写，immediately-invoked function expression。即时调用函数表达式。

IIFE表示在函数定义的时候，就立即执行。

|  |
| --- |
| 1. function fun() { 2. console.log(1); 3. }(); //报错 |

():表示调用只能用在函数表达式后面或者函数名后面。

|  |
| --- |
| 1. var fun = function () { 2. console.log(1); 3. }(); |

我们还可以将function声明函数通过数学运算符强制转为函数表达式， + - ，！ ,()

|  |
| --- |
| 1. (function fun3() { 2. console.log(3); 3. })(); |

第一个小括号表示将函数转换为函数表达式 ，第二个小括号表示调用。

IIFE使用：习惯将一个匿名函数转化为函数表达式，再次调用。

|  |
| --- |
| 1. (function () {})() |

IIFE性质: IIFE可以关住函数作用域

|  |
| --- |
| 1. // IIFE 能够关住函数作用域，不能在IIFE外面调用 2. (function fun4() { 3. console.log(4); 4. })(); 5. fun4(); |

IIFE实际参数书写在调用小括号中。

|  |
| --- |
| 1. // IIFE实际参数书写在调用小括号中 2. (function (a) { 3. a ++; 4. console.log(a); 5. })(4); |

IIFE内部有return可以作为实际参数参与运算。

|  |
| --- |
| 1. sum(5,(function (m) { 2. return ++ m; 3. })(9)); |

# 三、通过数组观察闭包

数组：是引用数据类型，保存的是地址

闭包：函数能够记住定义时外部环境和内部语句，不管不任何地方再次使用都会时刻参考

|  |
| --- |
| 1. // 声明数组 2. var arr = []; 3. for (var i = 0 ; i <= 9 ; i ++) { 4. // 当i在循环时，立即让函数调用 5. (function (a) { 6. // 数组第a项 赋值一个函数，函数返回值就是对应的索引值a 7. arr[a] = function () { 8. return a; 9. }; 10. })(i); 11. } 12. console.log(arr); 13. console.log(arr[0]); 14. console.log(arr[0]()); 15. console.log(arr[1]()); 16. console.log(arr[9]()); 17. console.log(arr[10]()) |

# DOM

## 4.1 ＤＯＭ概述

我们前面JS学习都是语言核心部分，也就是ECMAscript。一般都是在

控制台、输出语句里操作，JS还包括DOM和BOM。

DOM（Document Object Model，文档对象模型）描绘了一个层次

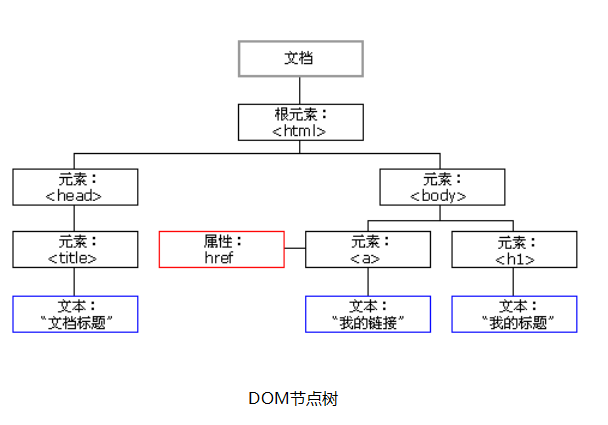
化的节点树，允许开发人员添加、移除和修改页面的某一部分。

这使得JavaScript操作HTML，不是在操作字符串，而是在操作节点，

极大地降低了编程难度。

DOM对很多东西做了抽象，提供了丰富的API：取得元素、css样式、

事件、运动、元素尺寸位置、节点操作。



## 4.2 html操作

document: 表示整个文档对象

数据类型：object

document.title 获取网页标题

设置： =

|  |
| --- |
| 1. document.title = "html操作"; |

document.body 获取body对象

获取html元素对象：

getElementById():通过id名获取js元素对象

调用对象：只能是document,表示从整个文档进行全局查找

参数：id名

返回值：元素对象

|  |
| --- |
| 1. var oBox = document.getElementById("box"); //Id具有唯一性 |

innerHTML: 获取元素内部文本。

|  |
| --- |
| 1. oBox.innerHTML |

设置内部文本： =

|  |
| --- |
| 1. oBox.innerHTML = "div内部文本设置"; |

元素对象可以通过点语法读取属性值

|  |
| --- |
| 1. // 读取元素对象属性值 2. console.log(oPic.src); 3. // 设置 4. oPic.src = "images/xiaoming.png"; |

getAttribute():读取元素属性值

参数：属性名（自带属性，自定义属性）

返回值：属性值

|  |
| --- |
| 1. oBox.getAttribute("data-xi") |

setAttribute():设置元素属性值

参数：第一个参数属性名，第二个参数属性值

返回值：没有返回值undefined

getAttribute()和点读取属性区别：

* 点语法只能读取或者设置元素自带属性。getAttribute()：可以读取自带属性或者自定义属性

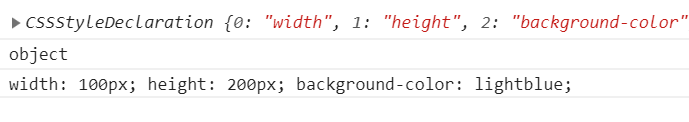
|  |
| --- |
| 1. oPic.getAttribute("alt") |

* 点语法可能需要改名字， getAttribute()不用改名字

class改名className

|  |
| --- |
| 1. console.log(oBox.className); 2. console.log(oBox.getAttribute("class")); |

* 点语法读取style获取到的是样式对象，get读取的是字符串



* 点语法读取到style样式对象可以继续操作，get读取到的是字符串不能再次操作

|  |
| --- |
| 1. oBox.style.width = "300px"; |

总结：除了自定义属性使用getAttribute()其他都使用点语法。

## 4.3 css操作

通过点语法读取style属性获得的是样式对象。包含了所有css样式

读取css属性值 读取的是行内样式属性值，不能读取计算后的样式

|  |
| --- |
| 1. 元素对象.style.属性 2. oBox.style.width |

// 设置 = 将要设置的属性值书写在=右侧（属性值写法和css相同）

|  |
| --- |
| oBox.style.border = "1px solid red"; |



单一属性需要改为驼峰命名

|  |
| --- |
| 1. oBox.style.backgroundColor |

## 4.4 事件

事件监听：我们计算机在解析我们JS代码的时候，会去看某一些元素身上是否添加了事件。随时监听这些事件有没有被触发，如果触发就立即执行相应的行为。

onclick 单击

ondblclick 双击

onmouseenter 鼠标进入

onmouseleave 鼠标离开

onmousedown 鼠标按下

onmouseup 鼠标弹起

onfocus 获取焦点

onblur 失去焦点

onload 加载完毕之后

绑定事件：元素.事件名然后进行赋值。（可以赋值一个函数或者一个函数名）。

这个函数会在事件被触发的时候立即执行。

|  |
| --- |
| 1. // 将匿名函数赋值给事件 2. oBox.onclick = function () { 3. alert(oBox.innerHTML); 4. }; 5. // 绑定事件方式 将函数名赋值事件 6. oBox2.ondblclick = change; 7. // 声明change函数 8. function change() { 9. oBox.style.backgroundColor = "pink"; 10. } |

onload 加载完毕之后

绑定onload一般使用window对象。

|  |
| --- |
| 1. window.onload = function () { 2. 只要当页面加载完毕之后，函数内部语句才会执行。 3. } |

## 4.5 信号量

多个函数同时控制一个变量，这个变量称为信号量（全局变量）

信号量不会无限变大或者变小，需要验证信号量边界

|  |
| --- |
| 1. // 信号量设置 2. var nowWd = 100; 3. // 点击btn1盒子变长15 4. oBtn1.onclick = function () { 5. // 1信号量改变 6. nowWd += 15; 7. // 2验证信号量边界 8. if (nowWd >= 300) { 9. nowWd = 300; 10. } 11. console.log(nowWd); 12. // 3体现在元素身上 13. oBox.style.width = nowWd + "px"; 14. }; 15. // 点击btn2盒子变短10 16. oBtn2.onclick = function () { 17. // 1信号量改变 18. nowWd -= 10; 19. // 2信号量验证 20. if (nowWd <= 100) { 21. nowWd = 100; 22. } 23. console.log(nowWd); 24. // 3体现在元素身上 25. oBox.style.width = nowWd + "px"; 26. }; |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |