爱创课堂前端培训

# javascript基础

第8天课堂笔记（本课程共8天）

班级：北京前端训练营22期

讲师：冯楠娜

日期：2019年3月31日

冯楠娜老师

E-mail : fnn@icketang.com

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# 复习

arguments: 函数内部类数组对象

IIFE: 即时调用函数

|  |
| --- |
| 1. (function () { 2. })() |

# 一、信号量

|  |
| --- |
| 1. // 获取元素 2. var oPic = document.getElementById("pic"); 3. var oBtn1 = document.getElementById("btn1"); 4. var oBtn2 = document.getElementById("btn2"); 5. // 信号量 不同事件同时控制图片改变 6. // 信号量有效值0,1,2,3,4 7. var idx = 0; 8. // 点击右按钮，图片下一张 9. oBtn2.onclick = function () { 10. // 1信号量改变 11. idx ++; 12. // 2验证信号量边界 13. if (idx > 4) { 14. idx = 0; 15. } 16. console.log(idx); 17. // 3体现在元素身上 18. oPic.src = "images/" + idx + ".png"; 19. }; 20. // 点击左按钮，图片上一张 21. oBtn1.onclick = function () { 22. // 1信号量改变 23. idx --; 24. // 2信号量边界检验 25. if (idx < 0) { 26. idx = 4; 27. } 28. console.log(idx); 29. // 3体现在元素身上 30. oPic.src = "images/" + idx + ".png"; 31. }; |

# 定时器

设置定时器：setInterval() 每隔一段时间函数自动执行一次

参数：一个参数：函数 二个参数：时间间隔，单位是ms 1s=1000ms（省略不写）

返回值：id数值

清除定时器：clearInterval()

参数：id数值

返回值：没有

* 到达终点，清除定时器。

|  |
| --- |
| 1. // 加载完毕盒子每隔1s自动变长25px，一旦到达300或者超过300盒子不在改变 2. var timer = setInterval(function () { 3. // 1信号量改变 4. nowWd += 25; 5. // 2边界检测 6. if (nowWd >= 300) { 7. nowWd = 300; 8. // 拉终停表 9. clearInterval(timer); 10. } 11. console.log(nowWd); 12. // 3体现在元素身上 13. oBox.style.width = nowWd + "px"; 14. }, 1000); |

* 设表先关

|  |
| --- |
| 1. // 鼠标离开盒子，重新开启定时器 2. oBox.onmouseleave = function () { 3. // 设表先关 4. clearInterval(timer); 5. timer = setInterval(function () { 6. // 信号量改变 7. nowWd += 25; 8. // 验证边界 9. if (nowWd >= 300) { 10. nowWd = 300; 11. // 关闭定时器。拉终停表 12. clearInterval(timer); 13. } 14. // 体现在元素身上 15. oBox.style.width = nowWd + "px"; 16. }, 1000); 17. }; |

# getElementsByTagName()

## 3.1 概述

调用对象：可以是document，表示从整个文档内进行标签查找。不管标签嵌套多深，都可以查找到

参数：标签名

返回值：类数组对象

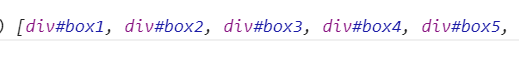
该类数组对象中保存的数据是元素对象。可以通过索引值读取该元素对象

|  |
| --- |
| 1. ps[0] |

还可以使用length属性，读取元素对象个数

该类数组对象保存元素时，只和标签首次出现的位置有关和标签嵌套没有关系。

|  |
| --- |
| 1. <div id="box1"> 2. <div id="box2"> 3. <div id="box3"></div> 4. </div> 5. <div id="box4"></div> 6. </div> 7. <div id="box5"></div> |



## 3.2 连读打点

getElementsByTagName()：

调用对象:元素对象，表示在该元素对象范围内进行查找

|  |
| --- |
| 1. // 先获取box1 2. var oBox1 = document.getElementById("box1"); 3. // 在box1范围内查找p元素 4. var ps1 = oBox1.getElementsByTagName("p"); 5. // 获取2号p元素对象 6. ps1[1].style.backgroundColor = "orange"; |

还可以连续打点

|  |
| --- |
| 1. // 连续打点 2. document.getElementById("box1").getElementsByTagName("p")[1].style.backgroundColor = "orange"; |

## 3.3 批量添加事件

通过得到的类数组对象，批量给数组中每一项添加相同的事件。

|  |
| --- |
| 1. // 给box1中4个p添加点击事件 显示该p元素索引值 2. // 获取ps类数组对象 3. var ps = document.getElementsByTagName("div")[0].getElementsByTagName("p"); 4. // 通过length批量添加点击事件 5. for (var i = 0; i <= ps.length - 1; i ++) { 6. // 给p元素添加点击事件 打印索引值 7. ps[i].onclick = function () { 8. console.log(i); 9. }; 10. } |

由于闭包影响：i=4

解决：IIFE

|  |
| --- |
| 1. // 通过length批量添加点击事件 2. for (var i = 0; i <= ps.length - 1; i ++) { 3. (function (idx) { 4. ps[idx].onclick = function () { 5. console.log(idx); 6. }; 7. })(i) 8. } |

解决：this 表示触发事件的该对象

原来该对象具有属性和方法，this同样具有。

|  |
| --- |
| 1. // 2this 触发事件的该对象 2. for (var i = 0; i <= ps.length - 1; i ++) { 3. ps[i].onclick = function () { 4. // 触发事件的对象 具有方法属性，同样this具有 5. console.log(this.innerHTML - 1); 6. }; 7. } |

解决：提前将索引值保存在对象的属性中

|  |
| --- |
| 1. // 3提前将索引值保存在对象的属性中 2. for (var i = 0; i <= ps.length - 1; i ++) { 3. // 设置对象的一个属性i 4. ps[i].index = i; 5. ps[i].onclick = function () { 6. console.log(this.index); 7. }; 8. } |

## 3.4 对应思想

通过索引值相同进行设置

|  |
| --- |
| 1. // 分别获取box1中p box2中p 2. var ps1 = document.getElementById("box1").getElementsByTagName("p"); 3. var ps2 = document.getElementById("box2").getElementsByTagName("p"); 4. // 利用索引值相同 设置 5. // 点击box1中的p 对应的box2中的p变色 6. for (var i = 0; i < ps1.length; i ++) { 7. // 提前将i保存在属性中 8. ps1[i].index = i; 9. ps1[i].onclick = function () { 10. ps2[this.index].style.backgroundColor = "orange"; 11. }; 12. } |

## 3.5 排他思想

选中的元素改变其他元素都是原状。

|  |
| --- |
| 1. for (var i = 0; i < ps1.length; i ++) { 2. // 提前将i保存 3. ps1[i].index = i; 4. // 绑定事件 5. ps1[i].onclick = function () { 6. // 1先排他 将所有box2中的p都恢复原样 7. for (var j = 0; j < ps2.length; j ++) { 8. ps2[j].style.backgroundColor = "#eee"; 9. } 10. // 2对应 11. ps2[this.index].style.backgroundColor = "orange"; 12. } 13. } |

# 四、计算后的样式

## 4.1 高级浏览器

计算后的样式：指的是HTML元素在css各种选择器综合作用下，得到的最终样式。

getComputedStyle():

调用：window

参数：元素对象

返回值：的到该元素对象计算后的样式集合对象

|  |
| --- |
| 1. window.getComputedStyle(oPic); |



1可以通过样式对象属性名读取属性值

|  |
| --- |
| 1. window.getComputedStyle(oPic)["border-color"] |

2 getPropertyValue():

调用对象：样式对象

参数：属性名（如果是单一属性，使用短横命名法）

返回值：计算后的属性值

|  |
| --- |
| 1. window.getComputedStyle(oPic).getPropertyValue("border-width") |

## 4.2 IE6,7,8

currentStyle:

调用对象：元素对象

结果：该对象计算后的样式对象。

的到属性值：需要继续打点调用属性名

|  |
| --- |
| 1. oPic.currentStyle.borderStyle |

也可以通过中括号读取属性值（驼峰命名法）

|  |
| --- |
| 1. op.innerHTML = oPic.currentStyle["borderColor"]; |

## 4.3 能力检测

通过浏览器是否认识getComputedStyle，如果认识返回值true; 不认识反之false

|  |
| --- |
| 1. // 能力检测 2. if (window.getComputedStyle) { 3. // 高级浏览器 4. op.innerHTML = window.getComputedStyle(oPic).getPropertyValue("border-width"); 5. }else { 6. // IE6,7,8 7. op.innerHTML = oPic.currentStyle["borderWidth"]; 8. } |

# 五、对象基础知识

对象：具有一些属性和方法集合

创建对象字面量：{

k: v,

k: v,

k: v

}

属性：和对象相关的一些值

方法：对象可以执行的一些行为

|  |
| --- |
| 1. var person = { 2. // 属性 和对象相关一些值 3. name: "小明", 4. height: 180, 5. sex: "男性", 6. // 方法 7. getStudy: function () { 8. console.log("小明你好"); 9. } 10. }; |

读取属性值：对象.属性 对象[“属性”]

|  |
| --- |
| 1. person.name 2. person[“height”] |

设置： 通过=

|  |
| --- |
| 1. person.age = 18; |



# 六、json

## 6.1 概述

json(JavaScript Object Notation) javascript对象表示法。

字面量： {}

k： v 属性名必须书写双引号

每一项之间使用逗号隔开，最后一项不用写逗号

|  |
| --- |
| 1. var json = { 2. // 属性名必须设置双引号 3. "name": "小明", 4. "height": 180, 5. "age": 18 6. }; |

读取：通过点语法，中括号

|  |
| --- |
| 1. // 读取 2. // 点语法 3. console.log(json.name); 4. // 中括号 5. console.log(json["height"]); 6. // 设置： = 7. json.sex = "男性"; 8. console.log(json); |

json嵌套json

|  |
| --- |
| 1. // json嵌套json 2. json.aihao = { 3. "aihao1": "学习", 4. "aihao2": "吃" 5. }; 6. console.log(json); 7. // 读取aihao属性值 8. console.log(json.aihao); 9. // 读取aihao2可以使用连续打点操作 10. console.log(json.aihao.aihao2); |

## 6.2 删除

delete json.属性（用空格隔开）

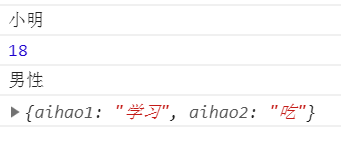
|  |
| --- |
| 1. delete json.height; |

## 6.3 遍历

for in: 依次输出json每一项属性值

声明变量k,依次从第一项开始接收json中属性名直到遍历到最后一个属性名，跳出循环

|  |
| --- |
| 1. // 遍历 2. for (var k in json) { 3. // k依次接收属性名 4. console.log(json[k]); 5. } |



## 6.4 json复制

json是object引用数据类型。保存地址

将json中每一项复制给另外一个新json

|  |
| --- |
| 1. // 创建json 2. var json = { 3. // 属性名必须设置双引号 4. "name": "小明", 5. "height": 180, 6. "age": 18 7. }; 8. var newJson = {}; 9. for (var k in json) { 10. newJson[k] = json[k]; 11. } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |