目录

[1. 课程笔记 3](#_Toc1293)

[1.1. Html模板 3](#_Toc8605)

[1.2. getElementById、getElementsByTagName 3](#_Toc4181)

[1.3. 事件onmouseover、onmouseout 4](#_Toc32615)

[1.4. 标签id 4](#_Toc10034)

[1.5. className 4](#_Toc7355)

[1.6. 函数传参 4](#_Toc20884)

[1.7. 操作元素属性的方法 4](#_Toc7710)

[1.8. 行间 与 样式表 4](#_Toc13556)

[1.9. style 与 className 5](#_Toc514)

[1.10. 提取行间事件 与 匿名函数 5](#_Toc4436)

[1.11. window.onload 5](#_Toc15960)

[1.12. document.getElementsByTagName 5](#_Toc11023)

[1.13. innerHTML 5](#_Toc15247)

[1.14. Alert 6](#_Toc12369)

[1.15. —————————————华丽分割线———————————————— 6](#_Toc8376)

[1.16. Js的组成 6](#_Toc24370)

[1.17. 变量类型 6](#_Toc16808)

[1.17.1. 显式类型转换（强制转换）： 6](#_Toc6600)

[1.17.2. 隐式转换 6](#_Toc18255)

[1.18. 闭包 6](#_Toc16304)

[1.19. 程序流程控制 7](#_Toc11166)

[1.20. 函数不定参 7](#_Toc8402)

[1.21. 取非行间样式 7](#_Toc11691)

[1.21.1. 兼容性 7](#_Toc15224)

[1.21.2. 样式分类 7](#_Toc18424)

[1.22. 数组 8](#_Toc18416)

[1.22.1. 基本操作 8](#_Toc16912)

[1.22.2. 万能的splice(起点，长度，元素) 8](#_Toc26183)

[1.22.3. 数组连接 concat() join() 8](#_Toc32717)

[1.22.4. 数组排序 8](#_Toc9959)

[1.22.5. charAt() 9](#_Toc1029)

[1.23. 定时器 9](#_Toc11548)

[1.23.1. 开启关闭 9](#_Toc2418)

[1.23.2. Date() 9](#_Toc25294)

[1.24. 获取边距offsetXxx 9](#_Toc27666)

[1.25. Dom节点 10](#_Toc12013)

[1.25.1. 子节点 10](#_Toc11862)

[1.25.2. 父节点 10](#_Toc27749)

[1.25.3. 首尾子节点 10](#_Toc201)

[1.25.4. 兄弟节点 11](#_Toc3480)

[1.26. className 11](#_Toc10281)

[1.27. DOM操作应用 11](#_Toc30888)

[1.27.1. 创建、插入DOM元素 11](#_Toc15619)

[1.27.2. 删除DOM元素 12](#_Toc19228)

[1.27.3. 文档碎片 creatDocumentFragment(元素) 12](#_Toc22852)

[2. 小技巧 13](#_Toc26532)

[3. 随记 13](#_Toc11116)

[3.1. 组件间通信 14](#_Toc1801)

# 课程笔记

## Html模板

<!DOCTYPE HTML>

<html>

<head>

<metacharset="utf-8">

<title>无标题文档</title>

<style>

</style>

<script>

</script>

</head>

<body>

</body>

## getElementById、getElementsByTagName

Document.getElementById 、Document.getElementsByTagName

Id和name都是元素中的属性，id相当于人的身份证，是唯一的，name则可以重复，所以Document.getElementById获取到一个元素，而Document.getElementsByTagName可以获取多个元素

在FF等一些浏览器下直接使用ID存在问题，Document.getElementId引入在任何浏览器下均可使用。

Document.getElementsByTagName获取到的元素存在于数组中，eg.

Var aDiv=Document.getElementsByTagName(‘Div’);

aDiv[0].xxx = xxx;

....

## 事件onmouseover、onmouseout

鼠标移入，鼠标移出

## 标签id

任何标签都可以加id

可通过给标签加id，通过js更改其标签属性

## className

标签中的class在js中在用className，因为js中也有关键字class，用于区别

## 函数传参

定义形参时不需要制定参数类型，eg. Function setColor(color){ }

## 操作元素属性的方法

第一种方法：用 ’ . ’，eg. document.getElementById('div1').style.display

第二种方法：用 “ [‘ ’] ” eg. document.getElementById('div1').style[‘display’]

一切用‘. ’的地方都能用” [ ] “

再eg. Document.getElementById(‘txt1’) => document[‘getElementById’](‘txt1’);

第三种：Dom方式：

获取： getAttribute(名称)

设置： setAttribute(名称, 值)

删除： removeAttribute(名称)

## 行间 与 样式表

行间：html中的行代码 < xxx >

样式表: <style></style>中的的样式

## style 与 className

Style加样式和取样式值都在行间中处理，如果style放在样式表中，则取不出来

--元素.style.属性=xxx 是修改行间样式

--修改完行间样式之后再修改className不会有效果

## 提取行间事件 与 匿名函数

提取行间事件，把行间的事件放在js中处理

匿名函数，不取名字的函数，eg.

<sricpt>

oBtn.onclick = function()

{

.....

}

</script>

## window.onload

Window.onload也是一个事件，在整个页面加载完成的时候才会执行

可以把js代码都放在该函数中处理eg.

window.onload=function() {

...

}

## document.getElementsByTagName

## innerHTML

表示便签中属于HTML的东西, 可以往里面写HTML也可以读出来

## Alert

## —————————————华丽分割线————————————————

## Js的组成

ECMAScript 翻译解析器

DOM （Document Object Model） Html Document

Bom （Browser Bbject Model） 浏览器 window

## 变量类型

常见类型： number string boolean undefined object function

### 显式类型转换（强制转换）：

**parseInt(string)** 转换成整数它从左到右扫描字符串，直到遇到非数字即停下来。

Eg. parseInt(“123px456”) => 123

Nan 与 Nan不相等，即Nan == Nan为false，判断用 isNan() 判断

**ParseFloat(string)** 转换成小数

### 隐式转换

== 先自动转换类型后比较 !=

=== 比较类型和数值 !==

- （减号） 先转换后相减，+加号则不会，加号作用：1.字符串连接 2.相加

## 闭包

通俗理解就是：子函数可以使用父函数中的局部变量

## 程序流程控制

判断： if switch...case:break; ?:

循环：while for

Eg. for(var i in xxx)

跳出：break continue

## 函数不定参

arguments 它是系统提供的一个数组，函数中可以传任意参数，形参中没有名字的参数可以通过arguments来获取，按顺序排列

## 取非行间样式

### 兼容性

Var oDiv = document.getElementById(‘div1’);

IE: oDiv.currentStyle.width

Chrome FF: getComputedStyle(oDiv, xxx).width ps.第二个参数xxx是个垃圾不用管

为兼容两种不同浏览器，可作以下处理：

If(oDiv.currentStyle){

//IE

oDiv.currentStyle.width

}else{

//Chrome FF

getComputedStyle(oDiv, xxx).width

}

只能取单一样式

### 样式分类

复合样式：background、border

单一样式：width height position

## 数组

### 基本操作

创建：

Var a=[1,2,3];

Var a= new Array(1,2,3);

属性：

Length: 既可以获取也可以设置

Eg. 快速清空数组 a.length=0;

添加：

push a.push(xx)

unshift a.unshift(xx)

删除：

pop a.pop() 尾部删除

shift a.shift() 头部删除

### 万能的splice(起点，长度，元素)

Splice(起点，长度) 中间删除

splice(起点，长度，元素)，先从起点删除n个元素，再添加

### 数组连接 concat() join()

b.concat(a), 把a连接到b后面

b.join(‘-’) ,分隔符

### 数组排序

sort()

a.Sort() 按首字母assic码从小到大排序

如何排列数字？

A.sort(fuction(n1,n2){

Return n1-n2;})

### charAt()

解决坚决兼容性问题，相当于数组的[]

var a=[2,2]

a.charAt(0) => a[0]

## 定时器

### 开启关闭

setInterval(x1,x2); x1是一个函数参数，x2是间隔时间，表示每隔x2毫秒执行一次x1函数。

setTimeout(x1,x2); 同上，但是setInterval一直执行，而setTimeout执行一次。

它们是有返回值的，可以通过 var timer=setInterval(xx,xx)

关闭定时器：

clearInterval(timer) clearTimeout

### Date()

var oDate = new Date()

oDate.getHours() 时

oDate.getMinutes() 分

oDate.getSeconds() 秒

oData.getYear() 年

oData.getMonth()+1 月

oData.getDate() 日

oData.getDay() 星期

## 获取边距offsetXxx

offsetLeft

offsetTop

offsetWidth

offsetHeight

## Dom节点

### 子节点

childNodes:

Eg. oUl.childNodes.length 获取子节点数量

oUl.childNodes[i].style.background=’red’ 调用子节点

nodeType节点类型:

oUl.childNodes[i].nodeType

3 ==> 文本类型 1 ==> 元素类型  
可用于解决兼容性问题eg.

If(oUl.childNodes.nodeType == 1){ xxx }

Children:

只获取元素不获取节点，相当于childNodes+nodeType

Eg. oUl.children[i]

### 父节点

parentNode：

Eg. oUl.parentNode

offsetParent:

获取用于参照定位的父节点元素 eg.oUl.offsetParent

### 首尾子节点

firstChild eg.oUl.firstChild //IE6-8

firstElementChild eg.oUl.firstElementChild //高级IE

判断兼容问题：

If(oUl.firstElementChild){

oUl.firstElementChild.xxx;

}else{

oUl.firstChild.xxx;

}

lastChild lastElementChild

### 兄弟节点

nextSibling nextElementSibling

previousSibling previousElementSibling

## className

例程 className.js，通过className筛选元素

## DOM操作应用

### 创建、插入DOM元素

createElement(标签名)+ 【appendChild(节点)】 或【insertBefore(节点, 原有节点)】

Eg.

var oLi = document.createElement('li');

if(aLi.length>0){

oUl.insertBefore(oLi,aLi[0]);

}else{

oUl.appendChild(oLi);

}

### 删除DOM元素

使用removeChild([子]节点)删除子节点元素

<ul >

<li> <a > </a></li>

</ul>

aA=xxx;

oUl=xxx;

for(var i=0;i<aA.length;i++){

aA[i].onclick=function(){

oUl.removeChild(this.parentNode);

};

}

本例子中为了删除<li>，则需要通过其父节点<ul>删除，通过获取<a>对应的父节即<ul>的子节点删除，这里的this.parentNode就是<a>的父节点。

### 文档碎片 creatDocumentFragment(元素)

var oUl=document.getElementById(‘ul1’);

var oFrag=document.createDocumentFragment();

for(var i=0;i<1000;i++){

var oLi=document.createElement(‘li’);

oFrag.appendChild(oLi);

}

oUl.appendChild(oFrag);

先把元素存到文档碎片中，再一次性渲染，在低级浏览器上性能可以提升，高级浏览器作用不大，基本上不用了

### 表格应用

table tBodies tHead tFoot rows cells

eg.

var oTab=document.getElementById('tab1');

oTab.getElementByTagName('tbody')[0].getElementByTagName('tr')[1].getElementByTagName('td')[1].innerHTML ==> oTab.tBodies[0].rows[1].cells[1].innerHTML

即：

oTab.getElementByTagName('tbody')[0] ==》 oTab.tBodies[0]

getElementByTagName('tr')[1] ==》 rows[1]

getElementByTagName('td')[1] ==》 cells[1]

# 小技巧

1. 字符串处理中 ‘’ + num 可成为字符串
2. setInterval(fun1(),1000 ); 会有1s延时，解决这个问题可以在后面先执行一次fun1()

即setInterval(fun1(), 1000);

fun1()

# 随记

## 组件间通信

设信号变量名：single

发送信号

组件a ——————————> 组件b

↓ ↓

在函数中使用 在mounted中监听

$emit(“single”, [xxx, ...] ); $on( “single”, (变量 or 函数) )；