# js总结笔记

**js常识**

js是逐行进行解释执行的，加载顺序为html>css>js

写在html中js语句最好用window.onload包起来，等页面，css加载完后再执行js脚本

window.onload=function(){

js语句

 }

<a href='javascript:;'>删除</a>

html中推荐使用双引号，js中推荐使用单引号

es6中用let定义变量，不用var

js中没有块级作用域

retrun false会停止执行后面语句

**js预解析**

预解析后再按照代码顺序执行

变量预解析（变量提升）

只提升声明，不提升赋值

函数预解析（函数提升）

**引入方式**

1.外部js文件

<script type=**"text/javascript"** src=**"../resources/js/js-study.js"**></script>

2.内嵌js

<script type=**"text/javascript"**>

js语句**;**

</script>

**注释**

单行注释 //

多行注释 /\*\*/

**输出弹框**

alert() 弹出警示框

console.log() 控制台打印

prompt() 弹出输入框 , 输入的是字符串型的

**变量名：**

字符,数字,下划线,$ 组成, 不能以数字开头

**变量类型：**

简单数据类型：Number,Boolean,String,Undefined,Null

存放值，在栈中开辟空间

number:数字型

尽量避开使用浮点数运算,会存在精度问题，不能判断两个浮点数是否相等

string：字符串型

推荐使用单引号

使用规程：外双内单/外单内双

11**+**'11'**=**'1111'

boolean：布尔型

undefined：未定义数据类型

声明变量未赋值

null：空值

属于object类型

复杂数据类型（有属性和方法）

存放指向堆的地址，在堆中开辟空间，作为参数会改变自身值（引用类型）

object， Array， Date

**==、===判断**

判断值 18 == '18' true

判断值和类型 18 === '18' false

**分支语句**

if ()

else if ()

else ()

for(let i in arr){ //i为索引

}

for(let i in object){ //i为k

}

**function**

形参不用声明

实参个数>形参个数，多余的实参就不要了

形参个数<实参个数，没有接受到值的形参为undefined

若没有return 语句则返回undefined

arguments是函数的内置对象（存储参数）

有length属性，可以通过索引获得参数

函数关键字声明函数

**function** add1**(**num1**,** num2**)** **{** /\* 函数关键字声明函数 \*/

**let** sum **=** num1 **+** num2**;**

**return** sum**;**

**}**

函数表达式法声明函数

**var** add2 **=** **function(**num1**,** num2**)** **{** /\* 函数表达式声明函数 \*/

**let** sum **=** num1 **+** num2**;**

**return** sum**;**

**}**

创建数组

new Array();

new Array(size);

new Array(element0, element1, ..., elementn);

创建 object

创建object方法1 变量声明方式

var obj\_1 = {

uname: '小明',

age: 12,

sex: '男',

sayHi: function(){

console.log('hi');

}

}

console.log(obj\_1.uname); /\* 调用对象的属性 \*/

console.log(obj\_1['uname']);

obj\_1.sayHi();

遍历object对象

for(var i in obj\_1) {

console.log(i); //输出属性名

console.log(obj\_1[i]); //输出属性对应的变量名

}

创建object方法2 new方式

var obj\_2 = new Object(); //创建空对象

obj\_2.uname = '小红'; /\* 追加属性和方法 \*/

obj\_2.sex = '女';

obj\_2.sayHi = function(){

console.log('hi');

}

创建object方法3 构造函数方式

function Student(uname, sex) { //构造函数名首字母大写

this.uname = uname;

this.sex = sex;

this.sing = function() {

console.log('sing');

}

}

var obj\_3 = new Student('小白', 33);

内置对象

MDN文档地址：https://developer.mozilla.org/zh-CN/

w3c文档网址: https://www.w3.org/

数组去重

遍历旧数组，indexOf()查看新数组有无该元素，无则添加

基本包装类型

把简单数据类型包装成复杂数据类型

var str = 'qqwwee';

等同于

var temp = new String('qqwwee');

str = temp;

销毁临时变量 temp

字符串不可变

看上去变了，只不过是改地址而已

不要大量的拼接字符串，会反复的改变地址，开辟新的空间，花费时间大

javascript标准

ECMA

DOM(页面文档对象模型)：

针对元素操作，创建，增删改查，属性，事件操作

W3C标准

BOM(浏览器对象模型)：

提供独立于内容而与浏览器窗口交互的对象，核心是window(窗口)

缺乏标准，兼容性差

API(应用程序编程接口)

js文档页面从上往下加载，所以script要写到标签的下面

tag:标签

事件三要素

事件源(按钮)

var btn = document.getElementById('btn');

事件类型

鼠标点击（onclick）

鼠标经过

键盘按下

事件处理程序

函数赋值的方式

btn.onclick = function() {

alert('点秋香');

}

点击 叉 图片后，将 display:none; 就能隐藏

焦点就是光标 获得焦点就是光标在文本框内

H5自定义属性

目的：页面中保存并使用数据，有些数据无需保存到数据库中

H5规定: 自定义属性 data- 开头

例子：data-index="2";

H5新增获取属性的方法： dataset.index;

节点获取元素

节点的属性

nodeType(节点类型)

元素节点：1

属性节点：2

文本节点：3（包括文字，空格，换行）

nodeName（节点名称）

nodeValue（节点值）

javascript:; 可以代替a链接中的#,地址栏不会变化

创建元素

document.write();会使页面重绘

innerHTML;拼接字符串创建元素 ，耗时，数组形式的话会很快

createElements 直接创建元素 快

事件注册

传统注册方式

on开头是事件onclick

btn.onclick = function() {}

唯一性

同一元素的同一事件只能有一个处理函数,后注册的会覆盖前面的

监听注册方式

IE9之后

w3c标准推荐方式

addEventListener() 是一个方法

同一个元素同一个事件可以注册多个监听器，按顺序依次执行

btn\_17.addEventListener('click', function(){

alert(22);

})

querySelector 查询选择器

stopPropagation 停止传播

DOM事件流

捕获阶段（外到内）

冒泡阶段 （内到外）

js代码中只能执行捕获/冒泡的一个阶段

onclick只能得到冒泡阶段

addEventListener(type, listener,[useCapture])

第三个参数是true，表示事件捕获阶段调用事件处理程序

是false,表示事件冒泡阶段调用事件处理程序

很少捕获，更多关注冒泡

onblur,onfocus,onmouseenter,onmouseleave没有冒泡

事件对象

div\_26.onclick = function(event) { 事件对象在函数()里面

console.log(event);

}

鼠标相关信息

键盘相关信息

事件对象可以自己命名 event,e

IE6,7,8 通过window.event获得事件对象

事件委托

不给每个子节点单独设置监听器，设置在父节点上，利用冒泡原理设置每个子节点

鼠标事件对象

MouseEvent

键盘事件对象

KeyboardEvent

keydown 谈起触发

keydown 按下触发

比如按下s光标定到文本框，但是按得久了，就把s输入到文本框，

所以应该用keyup

keydown/keypress特点（触发太早），所以用keyup

这两个键盘事件触发的时候，文字还没落入文本框

BOM(浏览器对象模型)

window对象是浏览器的顶级对象

全局对象，全局域中的变量，函数都会变成window对象的属性和方法

window.document.querySelector();

this

全局作用下/普通函数/计时器中指向的是window

方法调用中，谁调用指向谁

构造函数中this指向构造函数的实例

js是单线程(同步)

HTML5允许js多线程(异步)

异步任务

普通事件：click,resize

资源加载：load,error

定时器：setInterval,setTimeout

同步任务执行完，再执行异步任务

URL:

http://www.itcast.cn/index.html？naem=andy&age=18

**普通方式绑定事件 和 用addEventListener添加事件的区别**

传统方式绑定多个事件，后面的会覆盖前面的

btn\_1.onclick = function(){}

事件监听绑定多个事件， 会按照顺序执行

btn\_1.addEventListener('click', function() {})

事件

resize 窗口大小发生改变

load窗口加载完毕

**定时器（异步编程）**

setTimeout:一次性

PC端

offset 元素偏移量

获得元素距离带有定位的父元素的位置

获得元素自身的 宽高

只能读，每单位，任意样式表的值 ，用来获得元素大小位置

style只能得到行内样式表，修改的也是行内样式

style 可读写，不包含边框和padding，带有单位

用来修改元素

client 元素大小

立即执行函数：不需要调用，立马能执行的函数

所有变量都是局部变量，避免了命名冲突问题

scroll 滚动：获得滚动距离

得到元素的大小，滚动距离

头部卷进去多少

IE9才支持 window.pageYOffset

声明了DTD,使用 document.documentElement.scrollTop;

未声明DTD,使用 document.body.scrollTop

mouseenter

只会经过自身盒子时触发

mouseover

经过自身盒子会触发，经过子盒子也会触发

js移动端

无需考虑js在移动端的兼容性问题

触碰事件touch

touchstart 手指触摸到DOM元素触发

touchmove 手指在DOM元素上滑动时触发

touchend 手指从DOM元素上离开时触发

触摸事件对象touchEvent

touches 正在触摸屏幕所有手指的列表

targetTouches 触摸当前DOM元素上的手指列表

changedTouches 手指状态发生了改变的列表，从无到有，从有到无变化

尽量用CSS3做动画