JAVASCRIPT CHEAT SHEET

github.com/Hengyu-DU

ECMAScript5

知识点	涉及语法	功能	备注
变量	var abc = *	声明变量	var abc　不赋值时变量为undefined
	var aaa = * , bbb = * , ccc = *	多个变量用逗号隔开	
	var a = b = c = 9	相当于 var a = 9; b = 9;(无声明) c = 9;(无声明)	
	命名规则: 1. a-z , 0-9 , _ , $　　2. aAbB 区分大小写　　3. 不以数字开头 4. 不用关键字 var for 等　　5. 名称有意义　　6. 驼峰命名法		
数据类型	**Number 数字型**		
	var num = 010 数字前加0	八进制	
	var num = 0x1f 数字前加0x	十六进制	
	Number.MAX_VALUE	最大数值	
	Number.MIN_VALUE	最小数值	
	Number.MAX_VALUE * 2	Infinity	
	- Number.MAX_VALUE * 2	-Infinity	
	isNaN()函数	判断是否NaN(not a number)	null 37 true/false '' 都不属于NaN {} undefined NaN '37,5'属于NaN
	String 字符串型		
	\n	换行符	转义字符
	\\	\	
	\'	‘	
	\"	“	
	\t	tab缩进	
	.length	字符串长度	
	.charAt(index)	根据位置返回字符串中的单个字符	str[index] ie8以下不兼容
	.charCodeAt(index)	根据位置返回字符串中的单个字符ASCII值	
	.indexOf('')	查找字符串中某个字符的位置	.indexOf('', N) 从第N个开始找
	.concat(string)	**拼接字符串**	**返回一个新的字符串，不改变原来的**
	.substr(start,length)	**截取字符串**	
	.replace(target,newstring)	**单次替换字符**	
	.split('分隔符')	**拆分字符串为一个数组**	
	.toUpperCase()	大写	
	.toLowerCase()	小写	
	.trim()	删除空格	
	.trimStart()	删除空格	
	.trimEnd()	删除空格	
	Boolean 布尔值型		
	true + 1	=2　ture加法运算中当1	
	false + 1	=1　false加法运算中当0	
	Undefined 未定义		
	undefined + 'str'	'undefinedstr'	字符串的组合
	undefined + 1	NaN	
	Null 空值		
	null + 1	1	
	typeof 变量	判断基本数据类型	null的输出结果为object 一个历史bug Object下的[],{},Data等均为object
数据类型转换	**转为数字型**		
	parseInt()	去单位、取整	
	parseFloat()	去单位、保留小数	
	Number(要转换的内容)	**如遇单位则为NaN**	
	'12' - 0	**利用运算隐式转换**	
	转为字符串型		
	.toString()		
	String(要转换的内容) **new String()**		
	12 + ''	**利用加法隐式转换**	
	转为布尔值型		
	Boolean(要转换的内容)	"" 0 NaN null undefined 均为false	
运算符	+ - * / %	算术运算符	
	++e	**前置递增运算符**	单独使用无区别，运算中即时自加1，参与运算
	e++	**后置递增运算符**	单独使用无区别，运算中以原值计算，而后自加1
	+= -= *= /= %=	赋值运算符（自加自减）	
	> >= < <= == !=	只比较值	
	=== !==	全等，比较值和数据类型	
	&&	逻辑与	
	||	逻辑或	
	!	逻辑非	
短路运算 (逻辑中断)	A && B && C	从A开始，只要是0 null false '' NaN，就返回该false值并停止后面的运算，否则往后继续看下去，直到最后一个值，就返回最后一个值	有一个false就停止并返回false
	A || B || C	从A开始看，当遇到属于true的对象，就返回该true值并停止后面的运算，否则一直到最后还没有ture的话，就返回最后一个非true的值	有一个true就停止并返回true

JAVASCRIPT CHEAT SHEET

github.com/Hengyu-DU

ECMAScript5

知识点	涉及语法	功能	备注
运算符优先级	1. 小括号 () 2. 一元运算符 ++ -- ! 3. 算术运算符 先*%/ 后+- 4. 关系运算符 > >= < <= 5. 相等运算符 == != === !== **6. 逻辑运算符 先&& 后||** 7. 赋值运算符 = 9. 逗号运算符 ,		
流程控制	if(){ }	if语句	
	if(){ } else{ }	双分支语句	
	if(){ } else if{ } else{ }	多分支语句	
	条件表达式 ? A : B	三元表达式	
	switch(表达式){ case A: case B: default:}	switch语句 适合有固定值的情况	**注意:** 1. 开发时经常将表达式写成变量 2. 表达式和case值必须是【全等】才可以匹配 3. 每一个case后都有一个break，如果没有break会往下运行
循环	1 for(var i=1; i<10; i++){}		continue; 直接结束本次循环，直接进入i++ break; 立刻结束所有循环
	2 while(条件表达式){B}	A为true时，循环执行B	while里一定要有计数器、初始化变量、操作表达式，防止死循环
	3 do{B}while(条件表达式)	先执行B，然后判断，循环，直到条件为false	
作用域	ES6 时， 新增了块级作用域，ES5没有	1. 全局作用域 整个script标签 或者是一个单独的js文件 2. 局部作用域（函数作用域） 该变量名只在函数内部起效果和作用	1. 全局变量：在全局作用域下的变量，在全局下都能使用 **// 注意 如果在函数内部，没有声明就直接赋值的变量，也属于全局变量** 2. 局部变量：在局部作用域下的变量，在函数内部的变量就是局部变量
预解析	js引擎运行js分两步： **预解析 代码执行** 1） 预解析 js引擎会把js里面所有var、function提升到**当前作用域的最前面** 2） 代码执行 按照代码书写顺序从上往下执行	**1. 变量预解析（变量提升）**	把所有的变量声明提升到当前的作用域最前面，不提升赋值操作；把所有的函数声明提升到当前的作用域最前面，不调用函数。（因此所有提升的变量初始都为undefined）
		2. 函数内部预解析	执行某个函数时，会先对函数内部的变量进行一次提升
对象	var obj = { attribute : value , attribute : value , method : function(){} }	创建对象（一） 属性和方法用逗号隔开，方法后接一个匿名函数	
	var obj = new Object()	创建对象（二）new	
	obj.attribute	调用属性	
	obj['attribute']	调用属性(放入字符串)	
	obj.method()	调用方法	
	obj.attribute = ''	添加属性	
	obj.method() = function(){}	添加方法	
	for(var key in obj){ key 属性名 obj[key] 属性值 }	遍历对象	
	检测对象 *instanceof* Object	**检测变量是否为对象**	**可以用来检测Function，Array，Date，等复杂对象**
函数对象 Function	fucntion name(形参){}	函数关键字(命名函数)	
	var name = function(形参){}	函数表达式(匿名函数)	1) name 是变量名，不是函数名 2) 函数表达式声明方式跟声明变量差不多，只不过变量里面存的是值，而函数表达式里存的是函数
	name(实参)	调用函数	如果实参少于形参，没有提供的实参会被当作undefined
	return	返回值，并立即退出函数	return 退出函数， 返回return中的值
	arguments	传递过来的实数(伪数组形式)，可索引	

构造函数（类）	function Name(a,b,c){
	this.attribute = a
	this.attribute = b
	this.attribute = c
	this.method = function(){}
	}
		把对象里一些相同的属性和方法抽象出来封装的函数，用来初始化对象(为成员变量赋初始值)	构造函数名首字母大写，不需要return

	var obj = new Name(d,e,f)	对象的实例化	利用构造函数创建对象的过程，也称为【对象的实例化】，必须用new来调用构造函数
	obj.attribute	调用属性	
	obj.method()	调用方法	

JAVASCRIPT CHEAT SHEET

github.com/Hengyu-DU

ECMAScript5

知识点	涉及语法	功能	备注
内置对象	JS语言自带的一些对象，提供了一些常用和基本的功能，帮我们快速开发。		
Math对象	Math.PI	π	
	Math.max()	最大值	
	Math.abs()	绝对值	
	Math.floor()	向下取整	
	Math.ceil()	向上取整	
	Math.round()	四舍五入	
	Math.random()	返回随机小数[0,1)	
Date对象	var name = new Date(年,月,日,时,分，秒)	自定义时间	monthIndex从0开始 var birthday = new Date('Jan 17, 1995 03:24:00') var birthday = new Date('1995-12-17 04:24:00') var birthday = new Date(1995, 11, 17) var birthday = new Date(1995, 11, 17, 3, 24, 0)
	var name = Date()	**获得当前时间**	
	var name = new Date()	**获得当前时间**	**任意name**
	.getFullYear()	格式化年	
	.getMonth()	格式化月	monthIndex从0开始
	.getDate()	格式化日	
	.getDay()	格式化星期	Sunday为0
	.getHours()		
	.getMinutes()		
	.getSeconds()		
	var name = +new Date()	**获取时间戳一(ms) （也可用来自定义时间戳）**	获得Date总的毫秒数（时间戳） 即距离1970年1月1日过去了多少毫秒
	Date.now()	**获取时间戳二(ms)**	**IE9以上**
	.valueOf()	将一个Date对象转换为时间戳(ms)	
	.getTime()	将一个Date对象转换为时间戳(ms)	
Array对象	**创建数组**	var a = new Array()	创建数组（一） new
		var a = []	创建数组（二） 字面量
		a[0] a[1] a[2]	数组元素
		a.length	数组元素个数
		a[3] = 'hello'	修改\新增数组元素
		var a = new Array(2)	**长度为2的数组，元素为undefined**
		var a = new Array(2,3,4)	元素为2，3，4
	检测数组	**name *instanceof* Array**	**检测变量是否为数组**
		Array.isArray(检测对象)	**检测变量是否为数组【ie9以上】**
	操作数组	.push(*)	在末尾添加一个或多个元素
		.unshift(*)	在开头添加一个或多个元素
		.pop()	删除末位元素，并返回该元素
		.shift()	删除首位元素，并返回该元素
		a1 = a2.concat(a3)	合并a2和a3
		.slice(start,end)	**截取数组元素，返回一个由这些元素组成的新数组** 1. 填写两个索引号，结果包含start，不包含end) 2. 填写一个索引号，正值表示从第N个起，往后的所有 3. 负值表示从倒数的某个起，往后的所有
		.splice(start, deleteCount, item1, item2, itemN)	**对原数组的删除，替换，新增** 1. 只写start，删除从start起往后的所有元素 2. start，数量 删除从start起往后的N个元素 3. 后面继续写元素的话，就是在删除的位置新增，因此可以借此往中间添加新元素
	数组排序	**.reverse()**	**翻转数组**
		.sort()	**按元素首字母排序**

		.sort(function(a,b){
		return a - b
		}
			升序排列（a - b），降序排列（b - a）
			ES2015:
			.sort((a,b) => a - b)

	元素查找	数组名.indexOf(元素名)	正序查找元素，只返回找到的第一个元素的索引号，找不到返回-1
		数组名.lastIndexOf(元素名)	逆序查找元素，返回索引号
	类型转换	.toString()	转换为字符串，逗号分隔
		.join()	**用分隔符拼接成字符串，不写的话，默认逗号分隔**
		.join('')	没有分隔
		.join('&')	&分隔，可换成任意符号
立即执行函数	function fn(){*} fn()		
	(function(){*})()		
	(function fn(a,b){*})(1,3)		