DOM CHEAT SHEET

github.com/Hengyu-DU

Document Object Model 元素 \ 节点 \ 事件

知识点	涉及语法	功能	备注
API 应用程序编程接口	（Application Programming Interface）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。 **Web API是浏览器提供的一套操作[浏览器功能]和[页面元素]的API（BOM和DOM）。**		
DOM	Document Object Model，W3C组织推进的处理可拓展标记语言（HTML\XML）的标准编程接口，可以改变网页的内容、结构、样式。		
DOM树	**文档 Document**	一个页面就是一个文档	
	元素 Element	页面里面所有的标签都是元素	
	节点 Node	网页中所有的内容都是节点（标签、属性、文本、注释）	
获取元素 Element	.getElementById('')	返回一个指定id的元素对象	调用者可以是document（全局查找），也可以是某个标签的id（局部查找）
	.getElementsByTagName('')	返回指定**标签名**的对象**集合**	
	.getElementsByClassName('')	返回指定**类名**的对象**集合**	
	.querySelector('')	**返回指定选择器查找到的第一个元素**	**[H5新增，推荐]**
	.querySelectorAll('')	**返回指定选择器查找到的所有元素对象**	
	document.body	获取body元素	获取特殊元素
	document.documentElement	获取html元素	
操作元素 Element	***获取、修改元素内容***		
	element.innerText 属性	**区别一： 对 HTML 标签的识别** '今天是：' innerText不识别html标签,非标准 innerHTML识别html标签，W3C推荐	// 【该两个属性是可以读写的，可以获取元素里面的内容】 **区别二： 对空格和换行的处理** innerText获取的内容不含空格 innerHTML获取的内容含空格
	element.innerHTML 属性		
	获取、修改元素样式		
	element.style.样式名(驼峰) = ''	修改样式	// 1. 如果样式少，可以用element.style.样式名 // 2. 如果样式多，可以修改元素的className **// 3. 如果想保留原先的类名，我们可以直接在后面跟加类名**
	element.className = ''	修改类名	
		for(var i=0;i<a.length;i++){ a[i].style.XXX = '' }	排他算法：如果有一组元素，想要某个元素实现某种样式，需要用到循环的排他算法，即首先排除其他人，然后再设置自己的样式。
	获取、修改元素属性		
	element.属性名 = ''	直接对获取到的元素进行属性赋值操作	
	表单属性：		
	input.value	输入框的内容	例如 myBtn.onclick = function(){ myInput.value = '已提交' **this.disabled = true** } // this指myBtn(调用者)
	input.disabled	表单禁用属性 true\false	
	input.type	表单明文\暗文切换'text'\'password'	
	自定义属性		
	element.setAttribute('属性名',value)	添加自定义属性	
	element.getAttribute('属性名')	获取自定义属性	
	element.removeAttribute('属性名')	移除自定义属性	
	[h5新增][ie10以上] data-开头的属性：		
	element.dataset	获取data-开头属性的一个合集	如果有-连接，获取时用驼峰命名
	element.dataset.属性名	获取data-自定义属性	
	element.dataset['属性名']	获取data-自定义属性	
节点 Node	HTML DOM树中的所有节点均可通过JavaScript进行访问，所有HTML元素(节点)均可被修改,也可以创建或删除。节点至少拥有nodeType(节点类型)、nodeName(节点名称)、nodeValue(节点值)这三个基本属性。 - 元素节点 nodeType为1 - 属性节点 nodeType为2 - 文本节点 nodeType为3（包含文字、空格、元素间的换行等） **- 我们在实际开发中,节点操作主要操作的是【元素节点】**		
获取节点 Node	**父节点**	**node.parentNode**	**最近的父节点**
	元素节点	**node.children**	**所有子元素节点 [非正式，IE不兼容]**
		.children[0]	*第一个*
		.children[node.children.length-1]	*最后一个*
		node.first**ElementChild**	**[IE9及以上]**
		node.last**ElementChild**	
		node.next**ElementSibling**	
		node.previous**ElementSibling**	
	元素节点+文本节点	**node.childNodes**	**所有子节点（元素+文本节点）**
		.childNodes[1]	
		node.first**Child**	
		node.last**Child**	
		node.next**Sibling**	
		node.previous**Sibling**	

DOM CHEAT SHEET

github.com/Hengyu-DU

Document Object Model 元素 \ 节点 \ 事件

知识点	涉及语法	功能	备注
操作节点 Node	document.createElement('标签名')	动态**创建**元素节点	
	父节点.appendChild(新节点)	**追加**子节点	
	父节点.insertBefore(新节点,某个子节点)	**插入**到指定子节点之前	如ul.children[0](向前追加)
	父节点.removeChild(子节点)	**删除**子节点	如ul.children[0]（删除首位）
	node.cloneNode(是否深拷贝)	**克隆**当前节点	true 深拷贝 （标签+内容） false或空为浅拷贝 （只拷贝标签）
事件三要素	// (1) 事件源 - 事件被触发的对象 (谁 对象) // (2) 事件类型 - 如何触发 (什么事件, 比如鼠标点击(onclick)\鼠标经过\键盘按下) // (3) 事件处理程序 - 通过一个匿名函数赋值的方式完成 **执行事件三步骤：1. 获取事件源 2. 注册事件（绑定事件）3. 添加事件处理程序（采取函数赋值形式）**		
注册事件	1 onlick = function(){}	传统注册方式	特点:注册事件的【唯一性】 同一个元素同一个事件只能设置一个处理函数,最后注册的处理函数将会覆盖前面注册的处理函数
	2 addEventListener('事件类型',fn,[是否捕获])	方法监听注册方式 **（第三个参数默认或false为冒泡，如果是填true则捕获）**	[IE9之前的IE]不支持此方法，可使用 *attachEvent('',fn)* 代替。 特点：同一个元素同一个事件可以注册多个监听器，按注册顺序依次执行。
删除事件	1 onclick = null	传统注册方式	
	2 removeEventListener('事件类型',fn)		[IE9之前的IE] *detachEvent('',fn)*
鼠标事件	.onclick	鼠标点击	
	.onmousedown / onmouseup	鼠标按下 松开弹起	
	.onfocus / onblur	获得焦点 / 失去焦点	无冒泡
	.onmouse**enter** / onmouse**leave**	鼠标进入 / 离开	无冒泡
	.onmouse**over**/onmouse**move**/onmouse**out**	鼠标经过(进入范围) / 移动 / 离开	
键盘事件	keydown	按下触发	不区分大小写，识别功能键，
	keyup	弹起触发	
	keypress	按动按键	区分字母大小写，不能识别功能键
事件冒泡	IE最早提出，事件开始时由最具体的元素接收，然后逐级向上传播到DOM最顶层节点的过程。		
事件捕获	网景最早提出，由DOM最顶层开始，然后逐级向下传播到最具体的元素的接收的过程。		
事件委托	原理：不是每个子节点单独设置事件监听器，而是事件监听器设置在其父节点上，然后利用**冒泡原理影响设置在每个子节点。**		
事件对象 event/evt/e	div.onclick = function(e){ e = e || ***window.event*** var target = e.target || ***e.srcElement*** }	【官方解释】事件对象代表事件的状态,比如键盘按键的状态、鼠标按钮的状态、位置。【简单理解】事件发生后,跟事件相关的一系列信息数据的集合都放到这个对象里面,这个对象就是事件对象event,它有很多属性和方法。 [IE9之前的IE] ***window.event***	
常见事件对象 event/evt/e	e.target	返回的是触发事件的对象（元素）	比如点击的对象（目标） [IE9之前的IE] *e.srcElement*
	e.currentTarget	绑定事件的对象，和this一样	[IE9及以上]
	e.type	返回事件类型的名字	
鼠标事件对象 event/evt/e	e.clientX / clientY	鼠标在**浏览器可视区**的x和y坐标	
	e.pageX / pageY	鼠标在**页面文档**的x和y坐标(受滚动条影响)	
	e.screenX / screenY	鼠标在**电脑屏幕**的x和y坐标	
键盘事件对象 event/evt/e	e.key		
	e.keyCode	ASCII码值	
事件对象行为	e.preventDefault()	阻止默认行为，比如让a标签的链接不跳转，表单无法提交等	[IE9之前的IE] 1 *e.returnValue;* 2 *return false* 也能阻止默认行为，没有兼容问题,因为回调函数中return之后的代码都不会执行了（只限于传统的注册方式。）
	document.addEventListener('**contextmenu**',()=>{ e.preventDefault() })	禁用右键菜单	
	document.addEventListener('**selectstart**',()=>{ e.preventDefault() })	禁止选中文字	
	e.stopPropagation()	阻止冒泡	[IE9之前的IE] *e.cancelBubble = true* **兼容写法:** if(e && e.stopPropagation){ e.stopPropagation() } else { ***window.event.cancelBubble*** = true }

DOM CHEAT SHEET

github.com/Hengyu-DU

Document Object Model 元素 \ 节点 \ 事件

知识点	涉及语法	功能	备注

PC端网页特效

offset系列	offset就是偏移量,我们使用 offset系列相关属性可以动态的得到**该元素的的位置**等。 **- 注意:** **1 返回的数值都不带单位** 2 offsetwidth等属性是只读属性,只能获取不能赋值,所以我们如果只想要【获取元素大小位置】,用offset更合适。如果要赋值,用style或className。	.offsetParent	返回作为该元素带有定位的父级元素,如果父级都没有定位则返回body
		.offsetTop	返回元素相对于**带有定位父元素**上方的偏移
		.offsetLeft	返回元素相对于**带有定位父元素**左边框的偏移
		.offsetWidth	返回自身包括width + padding + border的宽度
		.offsetHeight	返回自身包括height + padding + border的高度
client系列	client就是客户端,使用它的相关属性可以获取元素可视区的相关信息,如**边框宽度、元素大小**。 **注意,返回数值不带单位。**	.clientTop	返回元素上边框宽度
		.clientLeft	返回元素左边框宽度
		.clientWidth	包括自身 width + padding 的宽度,不含边框
		.clientHeight	包括自身 height + padding 的宽度,不含边框
scroll系列	scroll,滚动,用于获取**滚动距离** **注意,返回数值不带单位。**	.scrollTop	返回元素内容被卷去的上侧距离
		.scrollLeft	返回元素内容被卷去的左侧距离
		.scrollWidth	返回元素实际宽度,包括超出overflow的全部宽度
		.scrollHeight	返回元素实际高度,包括超出overflow的全部高度
page页面偏移量	window.pageXOffset	页面头部卷去距离	**[IE9及以后支持]**
	window.pageYOffset	页面左侧卷去距离	

获取页面头部及左侧卷去距离的兼容性写法:

```
function getScroll() {
  return {
    left : window.pageXOffset || document.documentElement.scrollLeft || document.body.scrollLeft || 0,
    top : window.pageYOffset || document.documentElement.scrollTop || document.body.scrollTop || 0,
  }
}
```

使用的时候 getScroll().left

移动端网页特效

移动端	移动端浏览器兼容性较好,我们不需要考虑以前JS的兼容性问题,可以放心的使用原生JS书写效果,但是移动端也有自己独特的地方。比如触屏事件touch(也称触摸事件), Android和 IOS都有。 touch对象代表一个触摸点。触摸点可能是一根手指,也可能是一根触摸笔。触屏事件可响应用户手指(或触控笔)对屏幕或者触控板操作。		
触屏事件	touchstart	手指触摸到一个DOM元素时触发	
	touchmove	手指在一个DOM元素上滑动时触发	
	touchend	手指从一个DoM元素上移开时触发	
触屏事件对象 event	e.touches	当前正在触摸屏幕的所有**手指列表**	
	e.targetTouches	**正在触摸当前DOM元素的手指列表**	**重要**
	.targetTouches[0]	用于得到正在触摸dom元素的第一个手指的信息	
	.targetTouches[0].pageX / pageY	获得触摸点的坐标	
	e.changedTouches	**触摸动作变化记录列表**	**手指状态发生了改变的列表 从无到有 或者 从有到无**
classList	.classList	返回元素的类名列表	**[H5新增,IE10及以上支持,移动端随便用]**
	.classList.add('name')	添加类名	
	.classList.remove('name')	删除类名	
	.classList.toggle('name')	切换类(没有->有,或有->没有)	
移动端默认缩放行为	移动端 click事件会有300ms的延时,原因是移动端屏幕双击会缩放(double tap to zoom) 页面。 解决方案:		
	1.禁用缩放:浏览器禁用默认的双击缩放行为并且去掉300ms的点击延迟 <meta name='viewport' content='user-scalable=no'>		
	2.利用touch事件自己封装这个事件解决300ms延迟。		
	3.fastclick.js : *https://github.com/ftlabs/fastclick/*		
本地存储	随着互联网的快速发展,基于网页的应用越来越普遍,同时也变的越来越复杂,为了满足各种各样的需求,会经常性在本地存储大量的数据,HTML5规范提出了相关解决方案。**本地存储特性:** **1、数据存储在用户浏览器中** **2、设置、读取方便、甚至页面新不丢失数据** **3、容量较大, sessionStorage约5M、 lotalStorage约20M** **4、只能存储字符串,可以将对象 JSON.stringify()编码后存储**		
	window.sessionStorage 1、生命周期为**关闭浏览器窗口** 2、在同一个窗口(页面)下数据可以共享 3.以键值对的形式存储使用	存储 sessionStorage.setItem('name',value)	
		获取 sessionStorage.getItem('name',value)	
		删除 sessionStorage.removeItem('name')	
		清除所有 sessionStorage.clear()	
	window.localStorage 1、生命周期**永久生效,除非手动删除否则关闭页面也会存在** 2、可以多窗口(页面)共享,同浏览器可以共享 3、以键值对的形式存储使用	localStorage.setItem('name',value)	
		localStorage.getItem('name',value)	
		localStorage.removeItem('name')	
		localStorage.clear()	