**DOM的尺寸**

获取非行内样式（兼容问题）

@param \_\_\_\_ <DOM Object>

@param \_\_\_\_ <string>

@return \_\_\_\_ <string>

**事件**

事件触发三要素：通过谁触发？ 通过什么触发？触发后要做什么？

事件总结：事件是给浏览器定义一个预处理函数，当事件触发的时候，执行函数，这就是事件。

当事件被触发的时候奥特曼会得到一个信息（事件对象），包含了跟事件相关的一些属性和方法的封装（如：事件发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态等），只有事件在触发的时候才会得到。

e=e || window.event;

alert (e.buttons)观察.buttons的返回值；

**鼠标事件及方法**

|  |  |
| --- | --- |
| 属性名 | 含义 |
| e.buttons | 返回鼠标点击按键（1左键，2右键，4中键滚轮） |
| e.offsetX / offsetY | 获取事件触发最近的盒子（事件源）的坐标 |
| e.clientX / clientY | 获取可视区的坐标（根据浏览器的定位） |
| e.screenX / screenY | 获取整个屏幕的坐标 |

键盘事件：keydown,keyup,keypress键盘上每一个键都有一个唯一的编码，用来识别当前用户正在操作的是键盘上哪一个键

兼容问题：e.keyCode || e.which

特殊键码：是否按下alt ctrl 和 shift（e.altKey e.ctrlKey e.shiftKey ：返回值是布尔值）

if（e.keyCode==13&&e.altKey）{

     alert('同时按下了enter和alt')；

}

**事件冒泡**

事件流：事件执行顺序document->elementhtml->elementbody->elementdiv

子元素的事件被触发时，父级也会被触发（冒泡）

一个完整事件包含（捕获阶段->目标阶段->冒泡阶段）

**事件监听**

DOM0级事件处理，是一种赋值方式，是被所有浏览器所支持的，简单易懂容易操作。

DOM2级事件处理是室友DOM节点中的方法，可以重复监听，但是浏览器兼容存在问题。

**事件绑定**

同一个元素的同一个事件只能绑定一个处理函数，前面的会被覆盖。

**添加事件监听**

obj <DOM object> 添加监听的DOM对象

type <string> 事件句柄（不带on）

fn <function> 事件处理函数

isCapture <boolean> 是否捕获，默认为false （IE8 + 有效）

**给某个元素绑定鼠标滚轮事件**

obj <DOM object> 添加监听的DOM对象

fn <function> 事件处理函数 这个函数有一个参数（<boolean> true 向下 false代表向上）

/\* @return <boolean> true 向下 false代表向上\*/

**正则表达式**

定义正则表达式：

var reg = /a/;

var reg = new RegExp(“a”);

var reg = /[abcd]/ abcd任意一个有都为true;

var reg = /(ab)|(cd)/ ab 或者 cd;

/^q/ 以q开头 /3[^q]/ 3后面不能直接跟q

/$q/ 以q结尾 /^q$/ 单个字符q

元字符

\d 0-9 数字 \w 数字字母下划线 \s 空白字符

\D 非数字 \W 非数字字母下划线 \S非空白字符

**正则对象的方法**

test方法：该方法用来测试某个字符串是否与正则匹配，匹配就返回true，否则就返回false。

compile方法：该方法的作用是能够对正则表达式进行编译，被编译过的正则在使用的时候效率会更高，适合于对一个正则多次调用的情况下，如果对一个正则只使用一次，那么该方法没有特别显著的效应。

exec方法：返回的是一个数值，数值元素为匹配的子字符串。

**支持正则的字符串方法**

search：查找第一次匹配的字符串的位置，如果找到就返回一个number类型的index值，否则返回-1.

replace：该方法用来将字符串中的某些子串替换为需要的内容，接受两个参数，第一个参数可以为正则或者子字符串，表示匹配需要被替换的内容，第二个参数为被替换的新的子字符串。

split：将一个字符串拆分成一个数组，它接受一个正则或者子字符（串）作为参数，返回一个数组。

match：接受一个正则作为参数，用来匹配一个字符串，返回一个数组。

严格模式行为变更：

1. 全局变量声明时，必须加var
2. this 无法指向全局对象
3. 函数内重名属性
4. arguments 对象
   1. arguments 对象不允许被动态改变
   2. arguments 对象不允许被自调用
5. 新增保留字 implements，interface，let，package，private，protected，public，stalic，yield。

**字符串**

trim();去掉字符串前后空格

trimLeft();

trimRight();

JSON.parse(str);//json序列化，将符合json格式的字符串转换为json

JSON.stringify();//json转换为字符串

Date.now();//日期对象得到当前日期的毫秒数

Object.defineProperties(obj,props);//给obj设置属性

Object.leys(obj);//获取obj的所有属性名称，返回数组

Object.values(obj);//获取obj的所有属性值，返回数组

**let/const**

块级作用域：一种普遍存在于各个语言中的作用域范围；

用let声明的变量在外部不可访问，每个大括号都是独立的作用域

**暂时性锁区**

ES6规定在某个区块中， 一旦用let或const声明一个变量，那么这个区块就变成块级作用域，用let 或者const声明的变量即为该区块绑定， 该变量不受任何变量影响。 在该变量使用let声明前不可以用。在语法上，我们叫这种情况为：暂时性死区 (temporal dead zone，简称 TDZ)

字符串的新增方法

1. repeat()重复功能

‘x’.repeat(3) //xxx; 重复字符串

1. includes() startsWith() endsWith(); 判定字符串中是否存在某个字符串;

var s = ‘Hello world!’;

s.startsWith(‘Hello’) //true 以参数开头

s.endsWith(‘!’) //true 以参数结尾

s.includes(‘o’) //true 包括参数;

3. for of 一种新的遍历方式;

for of 可以用于遍历字符串;

var s = “abc”; for(let b of s){ console.log(b)}

//”a” “b” “c”

=>箭头函数

使用箭头函数：var 函数名 = 参数 => 运算规则;