**html**

html

(1)<section></section>

(2) <blockquote></blockquote>

(3)<div></div> 块级元素（自动换行）

(4)<span></span> 块级元素（不换行）

(5)<pre></pre> 以原格式显示文本

(6)<ol> 定义有序列表

<li></li>

</ol>

**CSS**

CSS：层叠样式表

<p align="center"><font color="red">test</font></p> 居中加色

**1.**引入CSS

(1)<p style="text-align: center;color: red;">test</p>

(2)在html头信息中定义CSS文件

|  |
| --- |
| <head>  <style type="text/css">  p {  text-align:center;  color:red;  }  </style>  </head> |

(3)方式1：<head> <link href="style.css" type="text/css" rel="stylesheet"></head>

方式2：

|  |
| --- |
| <head>  <style type=*"text/css"*>  @import *url("style.css")*;  </style>  </head> |

样式优先级由低到高：由上到下、由外到内

**2.**CSS的选择器

基本选择器：（优先级：元素 < class < id）

(1)html选择器（元素类型选择器）

(2)class选择器，把html标签的class属性值作为选择器，加“.”做前缀

|  |
| --- |
| <p class="sao">test</p> |
| *.sao*{  text-align:*center*;  color:*green*;  position:*absolute*;  } |

(3)id选择器，加“#”做前缀

|  |
| --- |
| <p id="sao2">test</p> |
| *#sao2*{  text-align:*center*;  font-size: *30px*;  color:*red*;  } |

**2.**选择器扩展

(1)关联选择器：标签的嵌套使用（p d）

(2)组合选择器：不同的标签使用相同的样式（p,b：逗号两侧元素使用当前同一个样式）

(3)伪元素选择器

格式：标签名：伪元素，类名

<a href="http://www.baidu.com">www.baidu.com</a>

**style.css**

|  |
| --- |
| @charset *"UTF-8"*;  **a***:link*{  color:*green*;  text-decoration: *none*;  }  **a***:visited*{  color: *red*;  text-decoration: *none*;  }  **a***:hover*{  color: *gray*;  text-decoration: *underline*;  }  **a***:active* {  color: *navy*;  text-decoration: *underline*;  font-size: *30px*;  } |

**3.**div

|  |
| --- |
| <div style="border-style: solid;">五哥常打开</div>  （none无/solid实线边框/dotted虚线原点边框/dashed虚线方框/double双边框） |
| **div**{  font-family: *楷体*;  font-size: *20px*;  text-align:*center*;  font-weight:*bold*;  width: *200px*;  height: *200px*;  padding:*10px 20px 30px 40px*; /\*内边距：从上边距顺时针\*/  border:*1px solid black*;  background-color: *#fffcd3*;  margin-left:*30px*;  } |

div浮动

|  |
| --- |
| float:*left*;  /\*向左漂浮后，宽度为0  div1移到父元素左边  div1的原始位置空出，由后边的元素补全\*/ |

**4.**插入背景图片

|  |
| --- |
| <style type="text/css">  body{  background-image:url(tu.png);  background-size:100% 100%;  height:100%;  }  </style> |

**5.**动画

|  |
| --- |
| @**keyframes** **myfirst**{  from {background:*red*; opacity:*0*;}  **to** {background:*green*;opacity:*100*;}  }  @**keyframes** **c**{  0%{opacity:*0*}  20%{**opacity***:*100}  40%{**opacity***:*0}  100%{**opacity***:*0}  }  **h1**{  animation:*myfirst 2s infinite*;  } |

**6.**音频

|  |
| --- |
| <audio src=*"wav/bgm.mp3"* controls=*"controls"* autoplay=*"autoplay"* loop=*"loop"*>  </audio> |

**js**

**1.** 弹出窗口

|  |
| --- |
| <script type=*"text/javascript"*>  alert("an error");  </script> |

**2.** 输入框：**var** price = prompt("input");

**3.** 判断是否为数字：isNaN(var v);（不是数字返回true）

**4.** 判断是否为整数的方法：Number.isInteger(x);

|  |
| --- |
| **function** isInteger(obj){  **return** obj%1 === 0;  } |

5. 输出到页面：document.write("Not a Integer");

6. Math.random(); 获取[0,1)中的随机数

Math.sqrt(x); 开平方根

Math.cell(x); 向上取整

Math.round(x)； 四舍五入

Math.floor(x); 向下取整

Math.abs(x); 取绝对值

|  |
| --- |
| <script type=*"text/javascript"*>  **var** a = Math.random();  a = Math.floor(a\*6)+1;  document.write(a);  document.write("<img src=\"images/"+(a%3+1)+".jpg\">");  </script> |

7. 控制台输出信息：console.info(a);

8. 数字转ASCII码：String.fromCharCode(x);

50. 数组

* 对象：内建对象、宿主对象、自定义对象
* 数组（Array）也是一个对象，属于内建对象
* 数组的储存性能比普通对象性能好
* 如果读取不存在的索引不会报错，而是返回undefined
* 对于连续的数组，使用length属性可以获取数组的长度；对于非连续的数组，会返回最大索引加1。
* 修改length时，如果修改的length大于原长，则多出的部分会被空出来；若小于原长，则多出的元素会被删除。
* 使用字面量来创建数组 var arr = [1,3,5,10];

var arr = new Array(1,3,5,10);

* 数组中可以放对象（包括函数、数组）

|  |
| --- |
| var arr = [function(){alert(0);},function(){alert(1);}];  arr[0](); |

* 常用方法

|  |
| --- |
| var arr = ["孙悟空","猪八戒","沙和尚"];  //push()向数组末尾添加一个或多个元素，返回数组的长度  arr.push("唐僧","白龙马");  //pop()删除数组的最后一个元素，并返回被删除的元素  arr.pop();  //unshift()向数组的开头添加一个或多个元素，并返回数组的长度  //向前边插入元素后，其他元素的索引会依次调整  arr.unshift("白龙马");  //shift()删除数组的第一个元素，并返回被删除的元素  arr.shift();  var arr = ["孙悟空","猪八戒","沙和尚"];  //forEach不兼容IE8及以下的浏览器  //forEach需要一个函数作为参数，每次执行时浏览器会将遍历到的元素以实参的形式传入  //这种函数由我们创建，但不由我们调用称为回调函数  //浏览器在回调函数中传入三个参数，第一个参数是当前正在遍历的元素，第二个参数是当前的索引值，第三个参数就是我们正在遍历的数组  arr.forEach(function(value,index,obj){  console.log(value);  }); |

**1.** DOM（Ducument Object Model）文档对象模型

节点（Node）：构成HTML文档最基本的单元

|  |  |  |  |
| --- | --- | --- | --- |
|  | nodeName | nodeType | nodeValue |
| 文档节点 | #document | 9 | null |
| 元素节点 | 标签名 | 1 | null |
| 属性节点 | 属性名 | 2 | 属性值 |
| 文本节点 | #text | 3 | 文本内容 |

范例代码：

|  |
| --- |
| <body>  <button id="btn">我是一个按钮</button>  <script>  //浏览器已经为我们提供了文档节点 这个对象是window的属性document  //我们可以直接在页面中使用，文档节点代表整个网页  //获取button对象  var btn = document.getElementById("btn");  //绑定一个单击响应函数  btn.onclick = function onClick(){  //innerHTML能获取到元素内部的html代码，对于自结束标签没有意义  btn.innerHTML = "I'm a button";  };  </body> |

2. 事件：用户和浏览器之间的交互行为。

点击两下触发：ondblclick="onClick();"

鼠标接触事件：onmousemove="onClick();"

onload事件会在整个页面加载完成后才触发：window.onload = function(){}

3. 获取元素节点：(通过document对象调用)

通过id属性获取一个元素节点对象：getElementById()

通过标签名获取一组元素节点对象[返回一个类数组对象]：getElementsByTagName();

通过name属性获取一组元素节点对象：getElementsByName();

|  |
| --- |
| <script>  var btn = document.getElementById("btn");  btn.onclick = function(){  var names = document.getElementsByName("gender");  for(var i = 0;i < names.length;i++){  //可以通过“.”来读取元素属性  //但注意：class属性不能采取这种方式  //读取class时应该用“.className”  alert(names[i].className);  }  }  </script> |

4. 获取元素节点的子节点：（通过具体的元素节点调用）

返回当前节点的指定标签名的子节点：getElementsTagName();

当前节点的所有子节点：childNodes

当前节点的第一个子节点（包括空白文本节点）：firstChild

区别于firstElementChild获得当前元素的第一个子元素，但在IE8及以下不兼容

当前节点的最后一个子节点：lastChild

|  |
| --- |
| //childNodes属性会获得包括文本节点内的所有节点  //根据DOM标签，空白也会被当成文本节点  //注意：在IE8及以下版本，不会将空白文档当做子节点  var btn02 = document.getElementById("btn02");  btn02.onclick = function(){  var city = document.getElementById("city");  alert(city.childNodes.length);  //children属性可以获取当前元素的所有子元素  alert(city.children.length);  } |

5. 获取父节点和兄弟节点：

当前节点的父节点：parentNode

当前节点的前一个兄弟节点（包括空白文本节点）：previousSibling

当前节点的后一个兄弟节点：nextSibling

6. 获取body标签：var body = document.body;

或 var body = document.getElementsByTagName(“body”)[0];

获取html根标签：var html = document.documentElement;

获取页面中的所有元素：var all = document.all;

或var all = document.getElementsByTagName(“\*”);

根据元素class属性获得元素节点对象：

var box = document.getElementsByClassName(“box”); //该方法不兼容IE8及以下

document.querySelector(); 需要一个选择器的字符串作为参数，可以根据一个CSS选择器来查询一个元素节点对象，但该方法只会返回唯一的一个元素。

document.querySelectorAll(); 该方法返回一个数组

7. 元素的增删改：

document.creatElement()； 创建一个元素节点对象，需要一个标签名作为参数

document.createTextNode(); 创建一个文本节点，需要一个文本内容作为参数

appendChild(); 向一个父节点中添加一个子节点

insertBefore(); 在指定的子节点前插入新的子节点，由父节点调用，第一个参数为新节点，第二个参数为旧节点

replaceChild(); 父节点.replaceChild(新节点，旧节点);

removeChild(); 删除子节点

**使用innerHTML也可以实现dom的增删改，不过有时风险较大，一般是两者结合使用。**

8. 使用DOM操作CSS：

修改样式语法：元素.style.样式名 = 样式值；（只能操作内联样式）

|  |
| --- |
| box1.style.height = "300px";  //注意：如果css样式含有减号，这种名称在css中是不合法的  //需要将这种名称改为驼峰命名法  //我们通过style属性读取和修改的样式都是内联样式 (内联样式的优先级较高，故通过js改的样式会立刻显现)  //但是如果在样式中写了 !important ,则此时的样式即使通过js也不能修改 (尽量少用)  box1.style.backgroundColor = "yellow"; |

获取元素当前显示的样式：元素.currentStyle.样式名

|  |
| --- |
| <script type="text/javascript">  window.onload = function(){  var box1 = document.getElementById("box1");  var btn01 = document.getElementById("btn01");  btn01.onclick = function(){  //获取元素当前显示的样式  //currentStyle只兼容IE  //alert(box1.currentStyle.backgroundColor);  //其他浏览器使用getComputedStyle()函数  //该方法需要两个参数，第一个为要获取样式的元素，第二个传递一个伪元素，一般传null，该方法返回一个封装了当前元素样式的对象,但不支持IE8及以下  var w = getStyle(box1,"width");  alert(w);  };  };  //自定义一个函数，获取指定元素当前的样式，处理兼容性问题常用方法  function getStyle(obj,name){  //这两种方式都是只读的，不能修改修改的话，必须通过style属性  //window不能省略，加上代表属性，不加代表变量（变量不存在则报错）  if(window.getComputedStyle){  return getComputedStyle(obj,null)[name];  }else{  return obj.currentStyle[name];  }  }  </script> |

其他样式操作

|  |
| --- |
| btn01.onclick = function(){  //element.clientWidth获取元素可见宽度，返回不带px,可直接计算  //获取到的值包括内容区和内边距,但不包括边框,这些属性是只读的  alert(box1.clientWidth);  //element.offsetWidth包括内容区，内边距和边框  alert(box1.offsetWidth);  //offsetParent获取离当前元素最近的开启了定位的父元素  //即position不是static,若所有祖先元素都未开启定位，则返回body  alert(box1.offsetParent);  //offsetLeft、offsetTop当前元素相对于定位父元素的水平、垂直偏移量  alert(box1.offsetLeft);  //scrollHeight获取滚动区域的高度  alert(box4.scrollHeight);  //scrollTop、scrollLeft获取滚动条移动的距离  //scrollHeight - scrollTop == clientHeight 说明垂直滚动条滚动到底  alert(box4.scrollTop);  } |

练习：moveWithMouse

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style type="text/css">  #box1{  width:100px;  height:100px;  background-color:red;  /\*开启绝对定位\*/  position:absolute;  }  </style>  <script type="text/javascript">  window.onload = function(){  /\*  \* 使div随鼠标移动  \*/  var box1 = document.getElementById("box1");  document.onmousemove = function(event){  //事件对象解决兼容性问题  event = event||window.event;    //获取鼠标的滚动条滚动的距离  /\*chrome认为浏览器的滚动条是body的，可通过body.scrollTop获取  \* 其他浏览器认为浏览器的滚动条是html的  \*/  //var st = document.body.scrollTop;  var st = document.documentElement.scrollTop || document.body.scrollTop;  var sl = document.documentElement.scrollLeft || document.body.scrollLeft;  //获取鼠标坐标，获得到的为数值  //clientX、clientY获取鼠标在当前可见窗口的坐标  //pageX、pageY获取鼠标相对于当前页面的坐标  //这两个属性在IE8中不兼容  var left = event.clientX;  var top = event.clientY;  //设置偏移量，必须开启定位  box1.style.left = left + sl + "px";  box1.style.top = top + st +"px";  }  }  </script>  </head>  <body style="height:1000px;width:2000px;">  <div id="box1"></div>  </body>  </html> |

9. 事件对象：

当事件的相应函数被触发时，浏览器每次都会将一个事件对象作为实参传递给相应函数。

在事件对象中，封装了当前事件相关的一切信息。

|  |
| --- |
| //事件对象  //在IE8及以下浏览器中相应函数被触发时，浏览器不会传递事件对象  //在IE8及以下浏览器中，是将事件对象作为window对象的属性保存的  areaDiv.onmousemove = function(event){  //解决事件对象的兼容性问题  event = event || window.event;  var x = event.clientX;  var y = event.clientY;  showMsg.innerHTML = "x = "+x+", y = "+y;  } |

10. 事件的冒泡（Bubble）

事件的冒泡指事件的向上传导，当后代元素的事件被触发时，其祖先元素的相同事件也会被触发，在开发中大部分冒泡是有好处的；若不希望发生事件冒泡，则可以通过事件对象来取消冒泡。

|  |
| --- |
| s1.onclick = function(event){  event = event||window.event;  alert("我是span");  //取消事件冒泡  event.cancelBubble = true;  }; |

11. 事件的委派：将事件统一绑定给元素的共同的祖先元素，当后代元素上的事件触发时，会一直冒泡到祖先元素，从而通过祖先元素的相应函数来处理事件。

|  |
| --- |
| u1.onclick = function(event){  //如果触发事件的对象是我们希望的元素，则执行；否则不执行  event = event || window.event;  //event.target表示触发事件的对象  if(event.target.className == "link"){  alert("我是ul");  }  }  ……  <ul id="u1"><li><a href="javascript:;" class="link">超链接1</a></li></ul> |

12. 事件的绑定

|  |
| --- |
| window.onload = function(){  var btn01 = document.getElementById("btn01");  //addEventListener()也能为元素绑定相应函数,而且可绑定多个响应函数  //第一个参数为事件的字符串（但不要on），第二个参数为一个回调函数  //第三个参数为一个布尔值，指是否在捕获阶段触发事件，一般传false  //但该方法不兼容IE8及以下，在IE8中使用attachEvent，第一个参数要on，共两个参数，不同的是attachEvent是从后执行  /\*btn01.addEventListener("click",function(){  alert(1);  },false);\*/    bind(btn01,"click",function(){  alert(this);  })  }  /\*自定义函数来解决为指定元素绑定响应函数的不兼容问题  \* addEventListener()中的this是绑定事件的对象  \* attachEvent()中的this是window  \* 参数：obj 要绑定事件的对象  \* eventStr 事件的字符串(不要on)  \* callback 回调函数  \*/  function bind(obj,eventStr,callback){  if(obj.addEventListener){  //大部分浏览器  obj.addEventListener(eventStr,callback,false);  }else{  //this是由调用方决定，若是可控，则可使用callback.call(obj)来修改this  //IE8及以下  obj.attachEvent("on"+eventStr,function(){  //在匿名函数里调用回调函数,解决this的问题  callback.call(obj);  });  }  } |

13. 事件的传播

* 对于事件的传播网景公司和微软公司有不同的设计：

微软公司：事件应该是由内向外传播的，当事件触发时，应先触发当前元素的事件，然后再向祖先元素上传(冒泡阶段)；

网景公司：事件应该是由外向内传播，即先触发该元素的最外层祖先元素(捕获阶段)

* W3C综合了两个公司的方案，将事件传播分为3个阶段：

(1). 捕获阶段；在捕获阶段从最外层的祖先元素，向目标元素进行事件的捕获，但是默认此时不会触发事件

(2). 目标阶段：事件捕获到了目标元素，事件捕获结束后开始在目标元素上触发事件

(3). 冒泡阶段：事件从目标元素向祖先元素传递，依次触发祖先元素上的事件

* 如果希望在捕获阶段便触发事件，可以将addEventListener()的第三个参数设置为true，（IE8及以下浏览器没有捕获阶段）一般情况下不会用。

14. 鼠标拖拽

|  |
| --- |
| //用于拖拽的函数  function drag(obj){  obj.onmousedown = function(event){  //当调用一个元素的setCapture()方法后，这个元素会把下一次所有的鼠标按下事件捕获到自身  //设置box1捕获所有鼠标按下事件  //setCapture()方法只有IE支持，但在火狐中不会报错，但chrome会报错  if(obj.setCapture){  obj.setCapture();  }    event = event || window.event;  //求出鼠标相对于div的偏移量  var ol = event.clientX - obj.offsetLeft;  var ot = event.clientY - obj.offsetTop;  document.onmousemove = function(event){  //解决兼容问题  event = event || window.event;  //获取鼠标坐标  var left = event.clientX - ol;  var top = event.clientY - ot;  //修改box1的位置  obj.style.left = left + "px";  obj.style.top = top + "px";  };  //鼠标松开时，被拖拽元素固定在当前位置  document.onmouseup = function(){  document.onmousemove = null;  //取消document的onmouseup事件  document.onmouseup = null;  //当鼠标松开时，取消对box1的捕获  obj.releaseCapture && obj.releaseCapture();  };    /\*当我们去拖拽一个网页的内容时，浏览器去会默认去搜索引擎中搜素内容  \* 此时，会导致拖拽功能的异常，这是浏览器提供的默认行为  \* 如果不希望发生这个行为，可以通过return false去取消该行为  \* 但是该方法不兼容IE8  \*/  return false;  };  } |

15. 鼠标滚轮事件：

|  |
| --- |
| /\*当鼠标滚轮向下滚动时，box1变长；向上时，box1变短\*/  window.onload = function(){  var box1 = document.getElementById("box1");  //为box1绑定鼠标滚轮滚动事件，不兼容火狐浏览器  //在火狐中需要使用DOMMouseScroll来绑定滚轮滚动事件  //而且该事件需要通过addEventListener()函数来绑定  box1.onmousewheel = function(event){  //判断鼠标滚轮滚动的方向  event = event || window.event;  //event.wheelDelta可获得滚轮滚动的方向，向上滚为120，向下滚为-120  //但火狐不支持该属性  //火狐中event.detail可获得滚轮滚动的方向，向下滚为3，向上滚为-3  if(event.wheelDelta > 0 || event.detail < 0){  box1.style.height = box1.clientHeight - 10 + "px";  }else{  box1.style.height = box1.clientHeight + 10 + "px";  }  /\*当滚轮滚动时，如果浏览器有滚动条，滚动条会随之滚动，这是浏览器的默认行为\*/  //但使用addEventListener()方法绑定响应函数时，取消默认行为不能使用return false，需要使用event.preventDefault来取消默认行为  //但IE8不兼容该方法  event.preventDefault && event.preventDefault();  return false;  };  bind(box1,"DOMMouseScroll",box1.onmousewheel);  }; |

16.

**jQuery**

**1.** 入口函数：$(document).ready(function(){});

**2.** 原生js和jquery区别：

(1)原生js和jquery入口函数的加载模式不同：

原生js会等到DOM元素加载完毕，并且图片也加载完毕才会执行；jquery会等到DOM元素加载完毕，但不会等到图片加载完毕就会执行。

(2)原生js如果有多个入口函数，后面的函数会覆盖前边的函数；jquery如果有多个入口函数，后边的不会覆盖前边的函数。

**3.** jquery其他的入口函数写法：

(1) $(document).ready(function(){alert("h1")});

(2) jQuery(document).ready(function(){alert("h2")});

(3) $(function(){alert("h3")}); (推荐使用)

(4) jQuery(function(){alert("h4")});

4.