1. CSS盒子模型

Id选择器用“#”，class选择器用“.”

标签选择器：直接用标签表示

包含选择器：如div中的p标签，在css直接用div p{}实现

子选择器：div>p{} 也可利用class .A>B{}

元素居中:text-align:center

CSS外部样式：<link rel=”stylesheet” type=”text/css” href=”mycss.css”>

内部样式：<style></style>

内联样式：<p style=””></p>

样式优先级：内联样式>内部样式>外部样式

Background属性：body{background-color:blue} //background-image; background-repeat; background-attachment; background-position; //background-repeat:repeat-x; 设置竖直方向不平铺 //background-repeat:no-repeat; 设置不平铺 //background-position:right top; 设置位置 //简写属性 body{background:blue url(“ ”) no-repeat right top}

//background-attachment; 背景图像是否固定或者随页面的其余部分滚动。

文本格式：对齐方式text-align：center/right/justify 居中 右对齐 宽距相等左右对齐

文本修饰 a {text-decoration:none;} 主要用来删除连接的下划线。text-decoration：overline/line-through/underline 文本转换 text-transform: uppercase/lowercase/capitalize

全部转换为小写字母/大写字母/首字母大写 文本缩进 text-indet:50px;

元素垂直对齐 vertical-align

字体：font标签 font-family //设置字体类型 font-style //设置字体样式 normal italic oblique 正常 斜体 类似斜体 font-size //设置字体大小 可以以px为单位，也可以以em为单位 font-weight //设置字体粗细

链接样式：a:link //正常，未被访问 a:visited //已经访问 a:hover //鼠标放在链接上

a:active //链接被点击的那一刻

CSS列表样式：ul 有序列表 ol无序列表 list-style-type 用于设置标题列表样式

List-style-image:url(“ ”); 用于给列表项引入图像

CSS表格：table{border-collapse:collapse; //消除th，td各自边框的影响 }

⭐ CSS盒子模型： 

元素的总宽度=宽度（content）+2\*border+2\*margin+2\*padding

box-sizing:content-box(W3C)/border-box(IE) 这句可以将IE类盒子模型和标准的切换

可以说所有的html元素都是个盒子，我们的网页就是由一个又一个盒子构成的。

Javascript获取元素尺寸和大小：1.获取计算 后的样式 代码：**var** obj = document.getElementById("box");

**var** style = **null**;

**if** (window.getComputedStyle) {

style = window.getComputedStyle(obj, **null**); // 非IE

} **else** {

style = obj.currentStyle; // IE

}

alert("width=" + style.width + "\nheight=" + style.height);

在这里最主要的是windo.getComputedStyle方法和window.currentStyle方法，还需要注意的是，这段代码必须在body执行完后执行，因为如果body还未执行，那么所返回的id将是空值。

1. 获取<link>和<style>标签写入的样式

代码：**var** obj = document.styleSheets[0]; // [object StyleSheetList] 样式表的个数<link>var rule = null;// [object CSSRule]

**if** (obj.cssRules){

rule = obj.cssRules[0]; // 非IE [object CSSRuleList]

} **else** {

rule = obj.rules[0]; // IE [object CSSRuleList]

}

alert(rule.style.width);

这里主要是styleSheet[]方法和cssRules方法的理解。

1. 获取元素的实际大小

代码：window.onload = **function**(){

**var** obj = document.getElementById("box");

alert(obj.clientWidth + "," + obj.clientHeight);

};

在这里的clientWidth方法返回的是内边距padding和content的大小和，需要注意的是，在chrome中，如果加上了滚动条，当调用这个方法时，所返回的值要减去滚动条宽度。

其次，scrollWidth和scrollHeigt两个方法可以返回滚动内容的宽高，其实就是在上面所说的减去滚动条宽度后的大小。

还有offsetWidth和offsetHeight方法，这种方法会忽略滚动条宽度。

1. 计算边框大小

右边框的宽度：obj.offsetWidth-obj.clientWidth-obj.clientLeft  
底边框的宽度：obj.offsetHeight-obj.clientHeight-obj.clientTop

scrollTop和scrollLeft  
    这组属性可以获取滚动条被隐藏(滚动条上方区域)的区域大小，也可设置定位到该区域。如果要让滚动条滚动到最初始的位置，那么可以写一个函数：

**function** scrollStart (element) {

**if** ( element.scrollTop != 0 ) {

element.scrollTop = 0;

}

}

1. Position属性相关

1.所有主流浏览器都支持position属性。

2.position属性所有的值：absolute // 生成绝对定位元素，相对于static定位以外的第一个父元素进行定位，元素位置通过left，right，top，bottom属性规定。即页面展示没有依赖性，如left：20px 显示时元素就离页面最左20px。

fixed //生成绝对定位的元素，相对于浏览器窗口进行定位，元素通过left，top，right，bottom属性进行规定。

relative //生成相对定位的元素，相对于其正常位置进行定位，因此left：20会向元素的LEFT位置添加20像素。

static //默认值，没有定位，元素出现在正常流中。（忽略top，right，bottom，left或者z-index声明）

inherit // 规定应该从父元素继承position属性的值。

clip：rect(xxpx，xxpx，xxpx，xxpx)；可以将一张图片剪切。

z-index:默认值是0。用这个属性可以更改元素优先级，需要记住的是，z-index是和position配套使用的，必须在声明了position属性后使用，而且其作用在无浮动的情况下，然后当其发挥作用时，子元素的层叠优先级一般都是有父元素决定，即子元素的z-index无效。

javascript获取元素定位属性值：document.getElementById(“id”).style.position

javascript还可以直接修改定位属性如：

function setPosition(){

document.getElementById(“beSet”).style.position=”absolute”;

document.getElementById(“beSet”).style.top=”30px”;

}

<input type="button" id="beSet" onclick="setPositionAbsolute()"

value="Set button position to be absolute" />

1. 圣杯布局和双飞翼布局
2. 浮动：float属性，常规思维，即float：left块级元素靠左，float：right靠右，一行大小不够用就排到下一行，上一行有高度比较高的元素，那么排至下一行的元素则会被卡住，即被卡住元素的横向位置会在该高元素的后面。

clear属性：值有left（不允许左侧出现浮动元素），right，both，none（默认值）

1. margin的负属性值，negative margin，margin是存在负值的，这在W3C标准里标注过，它在存在浮动布局的情况和一般的元素定位中都有着很重要的作用，比如在圣杯布局中可以用它配合浮动来完成布局，在一般的元素定位时也可以利用它来调整位置，在界面效果上，float、negative margin、z-index三者配合可以完成很多的任务。
2. hack 对开放源码的直接修改，一般处于作者的爱好，这种开放的方式可以使得新的技术更加完善，也可以促进业内交流，但也不乏修改后出现重大bug的情况，总的来说利大于弊，但是早期由于第三方缺乏组织管理，所修改出来的优质代码容易因为没有及时配合版本更新而废弃，造成资源的浪费，现如今第三方团队的出现使得这种hack技术更加规范。
3. 文档流：文档，即页面，doctype，”流”就指的是输入输出的形式，其中输入应该是”布局”，输出应该是”显示”，综合起来就是：页面的布局和显示。
4. 圣杯布局和双飞翼布局基本上是一样的，都是中间自适应，两边固定宽度的三栏布局，其中，中间放到文档流前面，保证先行渲染，解决方案大体相同，都是三栏全部浮动，区别在于解决中间栏div内部不被遮挡上，圣杯布局时中间栏添加相对定位，并配合left和right属性，自傲过上变现是三栏独立分开的（如果可以看到空隙的话），而双飞翼则是在中间栏的布局上嵌套一盒div，内容写在div里，然后对嵌套的div设置margin-left和margin-right，效果上表现为左右两栏在中间栏的上面，中间栏的宽度还是100%，只不过中间栏的内容通过margin的值显示在中间。
5. 圣杯布局基本代码：

<body>

<header style="background:grey;text-align:center;">布局</header>

<div id="bd">

<div id="middle">middle</div>

<div id="left">left</div>

<div id="right">right</div>

</div>

<footer style="background:grey;">...</footer>

</body>

<style>

#bd{

/\*左右栏通过添加负的margin放到正确的位置了，此段代码是为了摆正中间栏的位置\*/

padding:0 200px 0 200px;

height:100px;

}

#middle{

float:left;

width:100%;/\*左栏上去到第一行\*/

height:100px;

background:blue;

}

#left{

float:left;

width:200px;

height:100px;

margin-left:-100%;

background:#0c9;

/\*中间栏的位置摆正之后，左栏的位置也相应右移，通过相对定位的left恢复到正确位置\*/

position:relative;

left:-200px;

}

#right{

float:left;

width:200px;

height:100px;

margin-left:-200px;

background:#0c9;

/\*中间栏的位置摆正之后，右栏的位置也相应左移，通过相对定位的right恢复到正确位置\*/

position:relative;

right:-200px;

}

</style>

双飞翼布局基本代码：

<body>

<header style="background:green;">双飞翼布局</header>

<div id="middle">

<div id="inside">middle</div>

</div>

<div id="left">left</div>

<div id="right">right</div>

<footer style="background:green;">双飞翼</footer>

</body>

<style>

#middle{

float:left;

width:100%;/\*左栏上去到第一行\*/

height:100px;

background:blue;

}

#left{

float:left;

width:180px;

height:100px;

margin-left:-100%;

background:#0c9;

}

#right{

float:left;

width:200px;

height:100px;

margin-left:-200px;

background:#0c9;

}

/\*给内部div添加margin，把内容放到中间栏，其实整个背景还是100%\*/

#inside{

margin:0 200px 0 180px;

height:100px;

}

</style>

三．flex

1.display属性

用于更改元素的显示情况

常见的值：block//此元素显示为块级元素

none//此元素不会被显示

inline//默认值，此元素会被显示为内联元素，元素前后没有换行符。

inline-block//此元素会被显示为内联的块级元素。

list-item//此元素会作为列表显示。

run-in//此元素会根据上下文作为块级元素或内联元素显示。

table//此元素会被作为块级表格显示。类似<table>，前后带有换行符。

inline-table//会被作为行内表格显示，类似<table>，前后没有换行符。

table-row//此元素会被作为表格行来显示，类似<tr>。

table-cell//此元素会被作为表格单元来显示，类似<td>和<th>。

table-caption//此元素会被作为表格标题来显示，类似<caption>。

table-column//此元素会被作为单元格来显示。类似<col>。

table-column-group//此元素会作为一个或多个列的分组来显示，类似<colgroup>。

table-row-group//此元素会作为一个或多个行的分组来显示。类似<tbody>。

table-header-group//此元素会作为一个或多个行的分组来显示。类似<theader>。

table-footer-group//此元素会作为一个或多个行的分组来显示。类似<tfoot>。

1. flex布局优点

布局的传统方案，基于盒子模型，依赖display属性+position属性+float属性。它对于一些特殊的布局非常不方便，例如垂直居中。2009年W3C提出flex布局方案，可以简便、完整、响应式的实现各种页面布局。目前，它已经得到了所有浏览器的支持。

1. flex布局是什么

flexible box的缩写，意为弹性布局，任何容器都可以指定为flex布局。

如 .box{ display:flex; }，行内元素也可以指定flex布局，如 .box{ display:inline-flex; }，webkit浏览器的内核，必须加上-webkit前缀，如 .box{ display:-webkit-flex;/ \*sqfari\*/ display:flex; }。

注意：设为flex布局后，子元素的float、clear、vertical-align将失效。

采用flex布局的元素，叫flex容器，简称容器，它的所有子元素自动成为容器成员，称为flex项目，简称项目。以下为flex容器的基本构成。



容器默认存在两根轴，一条水平的主轴(main axis)和垂直(cross axis)的交叉轴，水平主轴的开始位置叫main start，结束位置叫main end，垂直交叉轴的起始位置叫cross start，结束位置叫cross end。

单个项目占据的主轴空间叫main size，交叉轴空间叫cross size。

1. 容器的属性

六个属性

1. flex-direction

决定主轴方向，即项目的排列方向

.box{flex-direction:row(主轴为水平方向，起点在左端)|row-reverse(主轴为水平方向，起点在右端)|column(主轴为垂直方向，起点在上沿)|column-reverse(主轴为垂直方向，起点在下沿)}

1. flex-wrap属性

一般情况下，项目都排在一条线上，即轴线上，flex-wrap属性规定如果一条轴线排不下，如何换行。

.box{ flex-wrap:nowrap(默认值，不换行)|wrap(换行，第一行在上方)|wrap-reverse(换行，第一行在下方);}

1. flex-flow

flex-flow是flex-direction属性和flex-wrap的简写形式。默认值为row nowrap。

.box{ flex-flow:<flex-direction值> <flex-wrap值>}

1. justify-content属性

定义了项目在主轴上对齐的方式。

.box{ justify-content:flex-start(默认值，左对齐)|flex-end(右对齐)|center(居中)|space-between(两端对齐，项目之间的间隔相等)|space-around(每个项目两侧的间隔相等);}

1. align-items属性

定义项目在交叉轴上如何对齐。

.box{ align-items:flex-start(交叉轴起点对齐)|flex-end(交叉轴终点对齐)|center(居中)|baseline(项目的第一行文字的基线对齐)|stretch(默认值，如果项目未设置高度或设置为auto，将占满整个容器的高度);}

1. align-content属性

定义了多根轴线的对齐方式。

.box{ align-content:flex-start(交叉轴起点对齐)|flex-end(交叉轴终点对齐)|center(交叉轴中点对齐)|space-between(交叉轴两端对齐，轴线中间的间隔平均分布)|space-around(每根轴线两侧间隔相等)|stretch(轴线占满整个交叉轴);}

1. 项目的属性

六个属性

1. order属性

定义项目的排列顺序，数值越小，排列越靠前，默认为0。

items{ order:<integer>;}

1. flex-grow属性

定义项目的放大比例，默认为0，即如果存在剩余的空间，也不放大。

.item{ flex-grow:<number>;}

如果所有的项目flex-grow属性值都为一，那么它们将等分剩余的空间，如果某个项目为2，其余的为1，那么这个项目分得得剩余空间是其它项目的两倍。

1. flex-shirnk属性

定义了项目的缩小比例，即如果空间不足，项目将缩小。

.item{ flex-shirnk:<number>;}

如果所有项目的flex-shirnk属性值都为1，那么当空间不足时，所有项目等比例缩小，如果其中有某个项目的值为0，那么该项目不缩小，其它项目等比例缩小，该属性负值无效。

1. flex-basis属性

定义了在分配多余的空间之前，项目占据的主轴空间，浏览器根据这个属性，计算是否有多余的空间，它的默认值时auto，即项目本身的大小。

.item{ flex-basis:<length>|auto;}

它可以设为跟width和height一样的属性值，如设置值为350px，则项目将占据固定空间。

1. flex属性

flex属性是flex-grow、flex-shirnk、flex-basis的简写。默认值为0 1 auto，后面两个属性可选。

.item{ flex:none|[<’flex-grow’> <’flex-shirnk’>||<’flex-basis’>]}

该属性有两个快捷属性值，none（0 0 auto）和auto（1，1，auto）

1. align-self属性

align-self允许单个项目和其它项目有不一样的对齐方式，可覆盖align-items属性，默认为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

.item{ align-self:auto()|flex-start()|flex-end()|center()|baseline()|stretch();}

除auto外，其它与align-items一致。

1. 选择器优先级

1.一般优先级公式

内联>id>class>标签

1. 优先级关系是如何确定的？

《CSS REFACTORING》中提到了具体过程

优先级是由a，b，c，d的值来决定的

1. 如果存在内联样式，那么a=1，否则a=0；
2. b的值等于id选择器出现的次数。
3. c的值等于类选择器、属性选择器、伪类出现的总次数。
4. d等于标签选择器和伪元素出现的次数。

伪类（pseudo-classes）是什么？？

伪类用于向某些选择器添加特殊的效果

语法：选择器：伪类{相应的属性和值}

选择器：类 伪类{相应的属性和值}

锚伪类：用于a标签

a.link{ } //未访问的链接

a.visited{ } //已访问的链接

a.hover{ } //鼠标移动到链接上

a.active{ } //选定的链接

a:hover必须被置于a:link和a:visited后才是有效的

a:active必须被置于a:hover之后才是有效的

伪类与css类：

<a class=”red”href=”xxxxxx”>css class and pseudo class</a>

a.red :visited{color:red}

点击链接的时候a标签内字体变为红色。

css first-child伪类 例如：li：first-child{}

注意：这并不是选择了页面中的第一li元素，而是选择了以li为子元素的元素里的第一个li元素。

匹配所有p元素中的第一个i元素：p>i:first-child{}

匹配所有作为第一个子元素的p元素的第一个i元素：

p:first-child i{}

lang伪类：如下例子：

<style>

q:lang(no){ quotes:”~””~” }

</style>

<body>

<p>文字<q lang=”no”>段落</q>文字</p>

</body>

回到优先级的话题上：

我们说优先级主要由abcd四个值决定，但是是怎么决定的呢？

首先，针对每一种选择器，我们计算他们的abcd，组成（a，b，c，d）这样的形式，先比较它们的a，若相等，往下比较b，以此类推，可以知道的是，内联样式的a为1，那么内联样式无论如何优先级都最高。

经过上面的思考，我们已经知道内联样式优先级最高，那么有没有可能让外部样式的优先级高于内联样式呢？

当然有可能，如果在外部样式中声明了！important（在相应的属性：值对后加上！important），那么外部样式优先级就比内联样式高了。

那么再思考，如果我在内联样式中加入!important,那内联样式岂不是无敌？虽然有点多此一举，但是事实上确实内联样式的优先级已经无法反驳了。

但还需要注意一点，我们的优先级有些是针对某些属性的，不同属性之间的优先级比较有时确实能超过内联样式。

1. CSS单位
2. 相对长度

分别有em（描述相对于应用在当前元素的字体尺寸，一般浏览器字体大小默认是16px，则2em=32px）、ex（依赖于英文字母x的高度）、ch（数字0的宽度）、rem（根元素html的font-size）、vw（viewpint wide视窗宽度，1vw=视窗宽度的1%）、vh（视窗高度，1vh=视窗高度的1%）、vmin（vw和vh中较小的那个）、vmax（vw和vh中较大的那个）、%

1. 绝对长度

分别有cm（厘米）、mm（毫米）、in（英寸，1in=96px=2.54cm）、px（像素，1px=1/96 in）、pt（point，大约1/72 in）、pc（pica 大约12pt，1/6 in）

1. 新特性
2. HTML5新特性（九大新特性）
3. 添加了video，audio标签

视频播放<video src=””></video>

查看视频所有属性、方法、事件：console.log(videobirds);

音频播放<audio src=””></audio>

查看音频所有属性、方法、事件：console.log(bgMusic);

1. 添加了canvas画布和svg，渲染矢量图片

web中可用的绘图技术：canvas绘图：H5原生绘图技术，基于网页画布绘制2D位图绘图技术，善于表现细腻颜色

SVG绘图：H5借鉴技术，基于SVG绘图空间绘制2D矢量图绘图技术，缩放不会失真

webGL绘图：尚不是H5标准技术 ，基于HTML5 canvas提供硬件3D加速渲染；有一个非常强大的3D扩展库，three.js

详细介绍：

canvas绘图技术：H5原生绘图技术，基于网页画布绘制2D位图，善于表现细腻颜色，可用于统计表、页面游戏、地图应用、网页特效等。

使用步骤：HTML5的canvas元素使用Javascript再网页上绘制图像，画布是一个矩形区域，可以控制其每一个像素，canvas拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

创建canvas元素，向HTML5页面添加canvas元素，规定元素的id，宽度和高度：

<canvas id=”myCanvas” width=”200”height=”100”></canvas>

通过Javascript进行绘制，canvas本身是没有绘制能力的所有的绘制工作必须在Javascript中完成：

<script type=”text/javascript”>

var c=document.getElementById(“myCanvas”);//使用id元素查找canvas对象

var cxt=c.getContext(“2d”);//创建context对象，getContext(“2d”)对象是内建的HTML5对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法

cxt.fillStyle=”#FF0000”;//fillStyle方法将其染成红色

cxt.fillRect(0,0,150,75);//这两行绘制了一个红色的矩形，fillRect方法规定形状、位置和尺寸

</script>

fillRect拥有四个参数(0,0,150,75)，其中前两个是矩形在画布中x轴y轴的位置，后两个规定了长和宽。

要注意：这里图形坐标是以图形左上角的点为准。

用canvas画直线：同样是建立canvas标签，然后是getContext

之后与画矩形不同点如下：在canvas标签要设置border样式

具体的划线代码：cxt.moveTo(10,10);//一个开始点

cxt.lineTo(150,50);//结束点，又一个开始点

cxt.lineTo(10,50);//又一个结束点

cxt.storke();//基于现有路径进行描边

用canvas画圆：前期操作与画线相同

具体的画圆代码：cxt,fillStyle=”#FF0000”;

cxt.beginPath();//开始

cxt.arc(70,18,15,0,Math.PI\*2,true);//画圆

cxt.closePath();//结束

cxt.fill();//基于现有路径进行填充

cxt.arc中前两个参数是圆的坐标（以圆心为准），第三个是圆的半径

用canvas实现渐变效果：前期操作同上

具体的渐变实现代码：

var grd=cxt.createLinearGradient(0,0,175,50);

grd.addColorStop(0,”#FF0000”);

gra.addColorStop(1,”#00FF00”);

cxt.fillStyle=grd;

cxt.fillRect(0,0,175,50);

把一幅图像放在画布上：前期操作同上

具体实现代码：var img=new image();

img.src=”xxxx”;

cxt.drawImage(img,0,0);

SVG绘图技术：在HTML5之前，SVG标签不能直接写在网页上，只能编写在独立的XML文档中，在网页中使用<img src=”x.svg”>进行嵌入。

在HTML5之后，可以直接将SVG标签写在网页中

与canvas不同之处1：canvas是位图，SVG是矢量图

2：canvas是javascript绘图技术（不是DOM元素）；SVG是标签绘图技术（是DOM元素）

3：canvas内容不能使用CSS；SVG内容可以使用CSS。

4：canvas内容不方便绑定事件处理；SVG内容方便进行事件绑定

常用的SVG图形：

1：矩形<rect width=”100”height=”100” x=”400”y=”400”fill=”#f0f”fill-opacity=”0.3” stroke=”#00f”stroke-width=”6”stroke-opacity=”.3”></rect>

2:圆形

<circle r=”100”cx=”400”cy=”400”fill=”#f0f”

fill-opacity=”0.3”stroke=”#00f”stroke-width=”6”stroke-opacity=”.4”></circle>

3:椭圆<ellipse rx=”100”ry=”50”cx=”400”cy=”350”fill=”#f0f”fill-opacity=”.4”>

1. 添加了一些列语义化标签header、footer、main、section、aside、nav等

语义标签：从标签名可以直接看出其功能的，如<table><img>等，非语义标签如<div><span>。

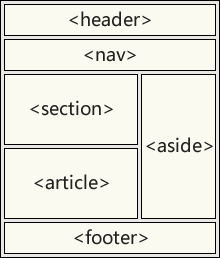
video，radio就是html5新定义的语义标签。

许多网站包含了指示导航、页眉、以及页脚的HTML代码，为了方便，HTML5

提供了定义页面不同部分的语义元素。

分别有<article><aside><details><figcaption><figure><footer><header><main> <mark><nav><section><summary><time>

具体功能如下图



<section>通常是语句中的有主题的内容组，一般带有标题。可以将网站划分为简介、内容、联系信息等

如：<section>

<h1>标题</h1>

<p>内容</p>

</section>

<article>规定独立的自包含内容，文档有其自身意义，并且可以独立于网站其他内容进行阅读。

如： <article>

<h1>标题</h1>

<p>内容</p>

</article>

<header>为文档或节规定页眉，一般被用作介绍性内容的容器。

<footer>为文档或节规定页脚，一般提供有关其包含元素的信息，通常包含文档作者、版权信息、使用条款连接、联系信息等，与header一样，一个页面中允许多次出现。

<nav>定义导航链接集合

如： <nav>

<a></a>

<a></a>

<a></a>

</nav>

<aside>主内容以外的某些内容，如侧边栏

<figure>和<figcaption> 与图片搭配的标题，caption的作用是为图片添加可见的解释。

如：<figure>

<img src=””alt=””></img>

<figcaption>标题内容</figcaption>

</figure>

1. input的type值添加了很多属性（email、url、number、range、Data pickers（date，month，week，time，datetime，datetime-local）、search、color）

增强型表单：新的input type

新的表单元素<input><textarea><select><option>...

<detailist>：数据列表，为input提供输入建议列表

                  <progress>：进度条，展示链接/下载进度

<meter>：刻度尺/度量衡，描述数据所处的阶段，红色（危险）=》黄色（警告）=》绿色（优秀）

<output>：输出内容，语义上表示此处的数据是经过计算而输出得到的

                  表单元素的新属性

通用属性：placeholder：占位提示文字

mutiple：是否允许多个输入

autofocus：自动获得输入焦点

form：制定输入元素所从属的表单，可以实现输入框放在表单外部并能被提交的效果

验证属性：required：输入框内容不能为空

min：允许输入的数字最小值

max：允许输入的数字最大值

minlength：允许输入字符串的最小长度

maxlength：允许输入字符串的最大长度

pattern：输入框内容必须符合的正则表达式

1. 添加了地理位置定位功能Geolocation API

通过浏览器获取用户当前所在地理坐标，以实现LBS服务，如实时导航，周边推荐等。

情形1：用户使用手机浏览器，可以根据内置GPS芯片读取数据。

情形2：用户使用pc浏览器，可以根据电脑的ip地址进行反向查询（需要很大的ip分配库）

 window.navigator.geolocation : {

                             watchPosition(){},

                             clearWatch(){},

                             getCurrentPosition(function(pos){

                                     '定位成功'

                                     定位时间：pos.timestamp

                                     维度：pos.coords.latitude

                                     经度：pos.coords.longitude

                                     海拔：pos.coords.altitude

                                     速度：pos.coods.speed

                             }, function(err){

                                     '定位失败'

                             }){},

                    }

1. 添加了web存储功能，localStorage和sessionStorage

web项目存储数据常用方案

1：服务器端存储

数据库存储，如商品、用户等核心数据

Session/内存存储，如用户的登陆信息

2：客户端存储

cookie存储，如用户偏好，访问历史，浏览器兼容性好，但是处理麻烦，且有容量限制

H5 WebStorage存储，如用户偏好，访问历史等安全要求的数据，老IE不兼容，但易使用且容量大。H5 WebStorage存储具体涉及到两个对象： 1：window.sessionStorage:类数组对象，通过键值对存储字符串数据------会话级存储

添加数据：sessionStorage[‘key’]=’value’

修改数据：sessionStorage[‘key’]=’newValue’

删除数据：delete sessionStorage[‘key’]

获取数据：var v=sessionStorage[‘key’]

2：window.localStorage:类数组对象，通过键值对存储字符串数据-----本地/跨会话级/永久存储

添加数据：localStorage[‘key’]=’value’

修改数据：localStorage[‘key’]=’newValue’

删除数据：delete localStorage[‘key’]

获得数据：var v=localStorage[‘key’]

1. 使用html5，通过创建cache manifest文件，可以轻松创建web应用的离线版本
2. web worker是运行在后台的Javascript，独立于其它脚本，不会影响页面性能；您可以继续做任意愿意做的事：点击，选取内容等，而此时web worker在后台运行

在HTML页面执行脚本时，页面是不可以操作的，不可响应的，直到脚本完

成。

web worker是运行在后台的javascript，独立于其他脚本，不会影响页面

性能，可以同时继续操作，选取、点击等。

web worker使用实例：

<script>

var w;

function startWorker(){

if(typeOf(worker)!==”undefined”){

if(w==”undefined”){

w=new Worker(“外部js路径”)；

}

w.onmessage=function(event){

document.getElementById(“id”).innerHTML=event.data;

};

}

else{

document.getElementById(“id”).innerHTML=”Sorry your browser does not support web worker”;

}

}

function stopWorker(){

w.terminate();

}

</script>

1. 服务端事件推送（EnventSources对象用于接收服务器发送事件通知），所有主流浏览器均支持服务器发送事件，除了Internet Explorer

1. 拖放API

H5之前没有拖放API，可以使用鼠标放下+鼠标移动两个事件来模拟用户拖动事件。

H5之后专门提供了七个鼠标拖动相关事件句柄：

拖动源对象(source)可能触发的事件：dragstart：拖动开始

drag：拖动中

dragend：拖动结束

拖动目标对象可能触发的事件：dragenter：拖动进入

dragover：拖动悬停

drop：松手释放

dragleave：拖动离开

注意：拖放API事件句柄中所有的事件对象都有一个dataTransfer属性（数据运输对象），用于在源对象和目标对象之间传递数据。

源对象：event.dataTransfer.setData(key,value)

目标对象：var value=event.dataTransfer.getData(key)

拖动实例：

<div id=“div1”ondrop=”drop(event)”ondragover=”allowDrop(event)”>

</div>

<img id=”drag1”src=”xxxx”draggable=”true”ondragstart=”drag(event)”width=”336”height=”69”>

<script>

function allowDrop(e){

e.preventDefult();

}

function drag(e){

e.dataTransfer.setData(“text”,e.target.id);

}

function drop(e){

e.preventDefult();

var data=e.dataTransfer.getData(“text”);

e.target.appendChild(document.getElementById(data));

}

</script>

1. CSS3新特性（九大新特性）
2. 媒体查询（可以查询设备的物理像素然后进行自适应操作）
3. transform，transition，translate，scale，skelw，rotate等相关动画效果
4. box-shadow，text-shadow等特效
5. CSS3 @font-face规则，设计师可以引入任意字体了
6. CSS3 @keyframes规则，可以自己创建一些动画等
7. 2D、3D转换
8. 添加了border-radius，border-image等属性
9. CSS3创建多列（column-count规定文本可以以几列的方式布局）
10. CSS3用户界面（resize，bo-sizing，outline-offset）