## HTML5简介

HTML

1. HTML标签
2. <pre>这是 预格式文本。它保留了 空格和换行。</pre>

注：preact提前；超前。

（2）<abbr title="etcetera">etc.</abbr> 注：abbr 缩写；缩写词

HTML 5 简介

1. HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。HTML5 是 W3C 与 WHATWG 合作的结果。
2. HTML5新规则

①减少对外部插件的需求（比如 Flash）

②更优秀的错误处理

③更多取代脚本的标记

注释:减少对脚本和插件的依赖，优秀的错误处理。

1. HTML5新特性

①用于绘画的 canvas 元素

②用于媒介回放的 video 和 audio 元素

③对本地离线存储的更好的支持

④新的特殊内容元素，比如 article、footer、header、nav、section

⑤新的表单控件，比如 calendar、date、time、email、url、search

1. 浏览器支持

最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。Internet Explorer 9 将支持某些 HTML5 特性。

## 表单

1. 与表单(<form>)有关的知识点：提交信息、正则表达式、插件、去重复点击、对齐、事件冒泡、兼容问题、单选框复选框对齐问题。

①表单元素常用事件有 change、select、submit、reset。

在IE6 IE7 IE8 IE9中 change、select、submit、reset 事件均不产生事件冒泡。

造成的影响:如果将事件处理委托给产生这些事件的父元素或祖先元素处理，在 Chrome Safari Firefox 中均是没有问题的。但是由于 IE6 IE7 IE8 IE9中这些事件不产生事件冒泡，将会导致位于祖先元素的事件委托没有被执行，可能会导致错误数据提交或页面UI不正常等情况出现。

②表单元素对齐可以使用li或者table，推荐使用li与label。

1. 合理使用 display: inline-block/inline/block

例如:将<span>标签的display设置为inline-block，即可设置宽高以及内外边距。

1. HTML5新标签

结构性标签: <article>,<aside>,<footer>,<header>,<nav>,<section>

非结构性标签: <audio>,<video>,<canvas>,<command>,<datalist>,

<details>,<figure>,<mark>,<progress>,<source>,<time>

1. 拖放事件

拖放（drag 和 drop）是 HTML5 标准的组成部分。拖放是一种常见的特性，即抓取对象以后拖到另一个位置。在 HTML5 中，拖放是标准的一部分，任何元素都能够拖放。

浏览器支持Internet Explorer 9、Firefox、Opera 12、Chrome 以及 Safari 5 支持拖放。

ondragstart: 目标元素(被拖动元素)开始被拖动时触发事件函数。

ondrag: 目标元素(被拖动元素)被拖动过程中持续触发事件函 数。

ondragenter: 被拖动的元素进入目标元素(容器元素)时触发事件函数。

ondragleave: 被拖动的元素离开目标元素(容器元素)时触发事件函数。

ondragover: 被拖动元素完全停留在目标元素(容器元素)之内时持续 触发事件函数。

ondrop: 被拖动元素拖放落入到目标函数(容积元素)之内时触发 事件函数。

注释:与被拖动元素绑定的事件: ondrag、ondragstart

与容积元素绑定的事件: ondragenter、ondragmove、ondragleave、ondrop

注意: 实现拖放事件的关键点: ①设置被拖放元素可拖放(img元素的默认属性值)。②设置容积元素可放置(取消容积元素的ondragover事件的默认行为)。

例题1:<img id="img" src="img/201.jpg" />

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

<script>

div.ondragover=function(e){

e.preventDefault();

}

div.ondrop=function(e){

e.preventDefault();

e.target.appendChild(img);

}

</script>

第一步:取消容积元素div的ondragover事件的默认行为，使容积元素可放置(定义容积元素**可放置**)。第二步: 定义容积元素div在有被拖动元素img落入时，将被拖动元素img移动插入到容积元素div的尾部(定义被拖动元素img**移动插入的时机**)。

注意: ①img元素默认为可拖动(draggable="true"),如果设置被拖动元素不可拖动(draggable="false")，则拖放事件无法实现。②如果未定义容积元素可放置，则拖动事件无法实现。

例题2: <script type="text/javascript">

(function(){

var $=function(id){

return document.getElementById(id)

};

var div1=$('cla');

var div2=$('clb');

var img1=$('img');

//拖动什么 - ondragstart 和 setData()

img1.ondragstart=function(e){

//img1开始被拖动时，触发一次函数。

//**注意：ondragstart函数中不可取消事件默认行为，否 则不可拖动。**

var data=e.dataTransfer;

data.setData('Text/plain',**this.id**)

//e.dataTransfer.setData()方法设置被拖数据的数据类型和值,在这个例子中，数据类型是 "Text/plain"，值是可拖动元素的 id ("img")。

console.log('ondragstart')

}

div2.ondragenter=function(e){

//有拖动元素开始进入div2时，触发一次函数。

e.preventDefault();

console.log('ondragenter')

}

//放到何处 - ondragover

div2.ondragover=function(e){

//有拖动元素完全停留在div2时，持续触发函数。

e.preventDefault();

//ondragover 事件规定在何处放置被拖动的数据。默认无法将数据/元素放置到其他元素中。如果需要设置允许放置，我们必须阻止对元素的默认处理方式。

console.log('ondragover')

}

//进行放置 - ondrop

div2.ondrop=function(e){

//有拖动元素落入到div2时，触发函数。

e.preventDefault();

var data=e.dataTransfer;

var text=data.getData('Text/plain');

e.target.appendChild($(text));

console.log('ondrop')

}

}())

</script>  
注释: ①调用 preventDefault() 来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）②通过 dataTransfer.getData("Text") 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据。③**被拖数据是被拖元素的 id ("drag1")**④把被拖元素追加到放置元素（目标元素）中。

例题3:拖动li，当落入另一个li中时，互换li中文本内容。

<script>

var lis=document.getElementsByTagName('li');

for (var i=0;i<lis.length;i++) {

//与img标签不同，li默认不可拖动，故首先定义可拖动。

lis[i].draggable='draggable';

lis[i].ondragstart=function(e){

//ondragstart事件不可取消默认行为，否则不可拖动。

//声明一个变量，存储被拖动li元素。

drag=this;

};

lis[i].ondragover=function(e){

e.preventDefault();

}

lis[i].ondrop=function(e){

e.preventDefault();

//声明一个变量，存储被放置li元素的文本内容，实现互换效果。

droptext=this.innerHTML;

this.innerHTML=drag.innerHTML;

drag.innerHTML=droptext;

}

}

</script>

1. HTML5 Input 类型

HTML5 拥有多个新的表单输入类型。这些新特性提供了更好的输入控制和验证。email、url、number、range、Date pickers (date, month, week, time, datetime, datetime-local)、search、color。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input type** | **IE** | **Firefox** | **Opera** | **Chrome** | **Safari** |
| email | No | 4.0 | 9.0 | 10.0 | No |
| url | No | 4.0 | 9.0 | 10.0 | No |
| number | No | No | 9.0 | 7.0 | No |
| range | No | No | 9.0 | 4.0 | 4.0 |
| Date pickers | No | No | 9.0 | 10.0 | No |
| search | No | 4.0 | 11.0 | 10.0 | No |
| color | No | No | 11.0 | No | No |

注释：Opera 对新的输入类型的支持最好。不过您已经可以在所有主流的浏览器中使用它们了。即使不被支持，仍然可以显示为常规的文本域。

举例:<body>

<form action="1.html" method="get">

<ul>

<li><label for="ipt1">mail:</label><span><input

type="email" id="ipt1" name="user\_email" /></span></li>

<li><label for="ipt2">url:</label><span><input

type="url" id="ipt2" name="user\_url"/></span> </li>

<li><label for="ipt3">number:</label><span><input

type="number" id="ipt3" min="-10" max="10" value="2" step="1"

name="points"/></span></li>

<li><label for="ipt4">range:</label><span><input

type="range" id="ipt4" min="-9" max="9" value="3" step="3"

name="points"/></span></li>

<li><label for="ipt5">date:</label><span><input

type="date" id="ipt5" name="user\_date" /></span></li>

<li><label for="ipt6">month:</label><span><input

type="month" id="ipt6" name="user\_month" /></span></li>

<li><label for="ipt7">week:</label><span><input

type="week" id="ipt7" name="user\_week" /></span></li>

<li><label for="ipt8">time:</label><span><input

type="time" id="ipt8" name="user\_time" /></span></li>

<li><label for="ipt9">datetime:</label><span><input type="datetime"

id="ipt9" name="user\_datetime" /></span></li>

<li><label for="ipt10">datetime-local:</label><span><input type="datetime-local" id="ipt10" name="user\_datelocal"

/></span></li>

<li><label for="ipt11">search:</label><span><input type="search" id="ipt11" name="search" /></span></li>

<li><label for="ipt12">tel:</label><span><input

type="tel" id="ipt12" name="telephone" /></span></li>

<li><label for="ipt13">color:</label><span><input type="color" id="ipt13" name="color" /></span></li>

<li><label>submit:</label><span><input type="submit" value="提交"/></span></li>

</ul>

</form>

</body>

注释: ①label标签为行内元素，当定义display:inline-block时，可设置宽度，使表单元素对齐。②当用户选择label标签时，浏览器就会自动将焦点转到和该标签相关的表单控件上。

6. HTML5 表单元素

学历: <input type="text" list="ipt\_list"/>

<datalist id="ipt\_list">

<option label="初中" value="初中" />

<option label="高中" value="高中" />

<option label="本科" value="本科" />

</datalist>

注意: ①option 元素永远都要设置 value 属性。②为兼容各版本浏览器，使label值等于value值。③input、datalist组合既可以选择，又可以填写。而select、option组合只可以选择不可以填写。

7. HTML5 表单属性

新的 form 属性：autocomplete、novalidate

新的 input 属性：autocomplete、autofocus、list、min, max 和 step、multiple、pattern (regexp)、placeholder、required等。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input type** | **IE** | **Firefox** | **Opera** | **Chrome** | **Safari** |
| autocomplete | 8.0 | 3.5 | 9.5 | 3.0 | 4.0 |
| autofocus | No | No | 10.0 | 3.0 | 4.0 |
| form | No | No | 9.5 | No | No |
| form overrides | No | No | 10.5 | No | No |
| height and width | 8.0 | 3.5 | 9.5 | 3.0 | 4.0 |
| list | No | No | 9.5 | No | No |
| min, max and step | No | No | 9.5 | 3.0 | No |
| multiple | No | 3.5 | No | 3.0 | 4.0 |
| novalidate | No | No | No | No | No |
| pattern | No | No | 9.5 | 3.0 | No |
| placeholder | No | No | No | 3.0 | 3.0 |
| required | No | No | 9.5 | 3.0 | No |

1. **autocomplete** 属性

autocomplete 属性规定 form 或 input 域应该拥有自动完成功能。

注释：autocomplete 适用于 <form> 标签以及大部分<input> 标签。

注意: autocomplete="on"为默认属性值，当设置为"off"时，取消该功能。

1. autofocus 属性

autofocus 属性规定在页面加载时，域自动地获得焦点。

注释：autofocus 属性适用于所有 <input> 标签的类型。

举例:<input type="text" autofocus="autofocus" />

1. multiple 属性

multiple 属性规定输入域中可选择多个值。

注释：multiple 属性适用于以下类型的 <input> 标签：email 和 file。

举例:<input type="file" multiple="multiple" />

1. novalidate 属性

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。

注释：novalidate 属性适用于 <form> 以及大部分<input> 标签。

实例:<form action="form.asp" method="get" novalidate="true">

E-mail: <input type="email" name="user\_email" />

<input type="submit" />

</form>

1. **pattern**属性

pattern属性规定用于验证 input 域的模式（pattern）。

注释：pattern属性值是正则表达式。适用于以下类型的 <input> 标签： text, search, url, telephone, email 以及 password。

举例：<input type="text" pattern="[A-z]{3}" title="三个字母" />

(6)placeholder 属性

placeholder 属性提供一种提示（hint），描述输入域所期待的值。

注释：placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

(7)required 属性

required 属性规定必须在提交之前填写输入域（不能为空）。

注释：required 属性适用于大部分类型的 <input> 标签。

实例:<input type="text" required="required" />

(8)disabled 属性

disabled 属性规定应该禁用的input元素。。

注释：disabled 属性适用于大部分类型的 <input> 标签。

实例:<input type="text" disabled="disabled" />

8. 文件api

注意: FileList对象与FileReader接口

举例:<body>

<input id="ipt" type="file" multiple="multiple" />

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

<input class="btn" type="button" value="文件名称" />

<input class="btn" type="button" value="修改时间" />

<input class="btn" type="button" value="文件二进制" />

<input class="btn" type="button" value="文件文本" />

<input class="btn" type="button" value="显示图片" />

<script>

(function(){

var btn=document.getElementsByClassName('btn');

btn[0].onclick=function(){

var filelists=ipt.files;

div.innerHTML=filelists[0].name;

};

btn[1].onclick=function(){

var filelists=ipt.files;

div.innerHTML=filelists[0].lastModifiedDate;

};

btn[2].onclick=function(){

var filelists=ipt.files;

var reader=new FileReader();

reader.readAsBinaryString(filelists[0]);

reader.onload=function(){

div.innerHTML=this.result;

};

};

btn[3].onclick=function(){

var filelists=ipt.files;

var reader=new FileReader();

reader.readAsText(filelists[0]);

reader.onload=function(){

div.innerHTML=this.result;

}

};

btn[4].onclick=function(){

var filelists=ipt.files;

var reader=new FileReader();

reader.readAsDataURL(filelists[0]);

reader.onload=function(){

div.innerHTML="<img src= "+this.result+" />"

}

}

}())

</script>

</body>

注释:① multiple="multiple"为设置文本框可以一次上传多个文件。②上传文件文本框有一个files属性，该属性指向文本框已选择的文件对象集合。

9. <input type="image" /> 定义图像形式的提交按钮。

注释:src 属性和alt 属性 与 <input type="image"> 结合使用。

举例: <input type="image" src="submit.gif" alt="Submit" />

注意:alt属性是img标签的一个必需属性，规定在图像无法显示时的替代文本。

## canvas

1. HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。画布是一个矩形区域(容器)，您可以控制其每一像素。canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。(canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成)
2. 准备：创建canvas元素→寻找canvas元素→创建context对象

<canvas id='mycanvas' width=400 height=400>

Your browser does not support the canvas element.

</canvas>

<script type="text/javascript">

var c=document.getElementById('mycanvas');

var cxt=c.getContext('2d');

</script>

注释: ①canvas元素的width、height为html属性，绝对不可以设置为CSS样式。width=数值 height=数值 (无引号无px)②重新设置width或height，画布中任何已绘对象都将被清除。③getContext("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和属性。④每一个canvas元素仅有一个上下文(getContext("2d"))对象。

1. 路径

步骤: \*定义样式(颜色、线宽、端点、拐点、宽度)→定义路径(位置尺寸)→绘制

例题1:折线路径

cxt.beginPath();

// ①定义样式

cxt.strokeStyle='green'; //颜色

cxt.lineWidth=20; //线宽

cxt.lineCap='square'; //端点

cxt.lineJoin='round'; //拐点

//②定义路径

cxt.moveTo(50,50); //起点

cxt.lineTo(100,100); //拐点

cxt.lineTo(50,200); //终点

cxt.closePath(); //闭合

//③绘制

cxt.stroke();

注释:①定义样式和定义路径(位置尺寸)的相对位置可调整，但绘制必须在定义样式和定义路径(位置尺寸)之后，否则样式和路径无法生效。②第一步定义样式可省略。③若未定义样式，则为默认样式；若未修改样式，则为上次样式。④cxt.beginPath()为新创建路径的意思，cxt.beginPath()与cxt.closePath ()起到线段闭合的作用。⑤moveTo()为新创建起点的意思，若无moveTo()，则第一个lineTo()为起点，绘制一组连续不间断的折线段。

注意: ①beginPath()与moveTo()都有重新绘制的含义。②moveTo()包含折线段的新起点，而beginPath()不包含新起点。③beginPath()可应用于各种形状的绘制，起到重新绘制另一个图形的作用，moveTo()只能应用于折线段的重新绘制。

例题2:矩形路径

//②定义位置尺寸

//位置(50,20)，尺寸(200,200)

cxt.rect(50,20,200,200);

//③绘制

cxt.stroke();

例题3:圆形路径

//②定义位置尺寸

//中心点坐标(100,70),半径50,开始角180度，结束角0度，逆时针。

cxt.arc(100,70,50,Math.PI,0,true);

//③绘制

cxt.stroke();

1. 颜色填充

步骤: \*定义样式(填充颜色)→定义路径(位置尺寸)→填充

例题1:折线填充

//①定义样式

cxt.fillStyle='red'; //颜色

//②定义路径

cxt.moveTo(50,50); //起点

cxt.lineTo(100,100); //拐点

cxt.lineTo(50,200); //终点

//③填充

cxt.fill();

注意:①定义样式和定义路径(位置尺寸)的相对位置可调整，但填充必须在定义样式和定义路径(位置尺寸)之后，否则样式和路径无法生效。②第一步定义样式可省略。③若未定义样式，则为默认样式；若未修改样式，则为上次样式。

例题2-1:矩形填充

//①定义样式

cxt.fillStyle='red';

//②定义位置尺寸

cxt.rect(50,20,200,200);

//③填充

cxt.fill();

注意：当设置路径与填充的位置尺寸完全相同时，填充边界恰好在路径中心，路径可将填充范围包裹。

例题2-2:渐变填充、矩形的简化书写、清除矩形像素

//(1)矩形路径的简化书写(常用)

cxt.strokeRect(50,50,100,100)

//(2)渐变填充

var grd=cxt.createLinearGradient(0,200,400,200)

// 渐变起始**点**坐标(50,200),结束**点**坐标(400,200)

//定义渐变起始点颜色为红色。

grd.addColorStop(0,'#F00');

//定义渐变中间点颜色为绿色。

grd.addColorStop(0.5,'#0F0');

//定义渐变结束点颜色为蓝色。

grd.addColorStop(1,'#00F');

cxt.fillStyle=grd;

//(3)矩形填充的简化书写(常用)

cxt.fillRect(0,200,400,100)

//(4)清除矩形像素(常用)

cxt.clearRect(50,200,50,50)

例题3: 圆形填充

//①定义样式

cxt.fillStyle='blue';

//②定义位置尺寸

//中心点坐标(100,70),半径50,开始角0度，结束角90度，顺时针。

cxt.arc(100,70,50,0,Math.PI\*1.5,false);

//③填充

cxt.fill();

1. 图片填充

步骤: 创建img元素→选择填充时机→填充

例题1: //①创建img元素

var img=new Image();

//等价于var img=document.createElement('img');

img.src='./a1.jpg';

//②选择填充时机

img.onload=function(){

//③填充

//剪切位置(0,0),剪切尺寸(600,600),

//放置位置(0,0),放置尺寸(100,100);

cxt.drawImage(img,0,0,600,600,0,0,100,100);

}

注意:①img.onload不可省略，否则无法加载。②放置尺寸不可以省略。

例题2:走路动态效果

(function(){

var ctx=tc.getContext('2d');

var img=document.createElement('img');

img.src='mario.png'

img.onload=function(){

var x=0;

setInterval(function(){

ctx.clearRect(0,0,500,500);

ctx.drawImage(img,x,0,37,47,100,100,37,47);

x+=37;

if (x>=111) {x=0};

},100)

};

}())

注意:由于像素不一定恰好为整数，故判断条件(x>=111)不可以改为x=111。

1. canvas转化为图片路径

<canvas id='cvs'>

<p>当前浏览器不支持canvas</p>

</canvas>

<img id="img">

<script type="text/javascript">

(function(){

cvs.width=200;

cvs.height=200;

cxt=cvs.getContext('2d');

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

var url=cvs.toDataURL()

img.src=url;

}())

</script>

注释:canvas转化而成的图片包含canvas容器以及其内的图像，但不包含canvas边框。

## 音频/视频

HTML 5 音频

1. Web 上的音频：直到现在，仍然不存在一项旨在网页上播放音频的标准。今天，大多数音频是通过插件（比如 Flash）来播放的。然而，并非所有浏览器都拥有同样的插件。HTML5 规定了一种通过 audio 元素来包含音频的标准方法。audio 元素能够播放声音文件或者音频流。

2. audio 元素支持三种视频格式：①.ogg ②.mp3 ③.wav

注释: 不同浏览器对不同视频格式的支持情况不同。例如:IE9.0+浏览器只支持mp3格式，Safari3.0+浏览器支持mp3和wav格式，Firefox、Opera浏览器只支持ogg和wav格式，chrome浏览器支持ogg和mp3格式。

3. <audio src="song.ogg" controls="controls"></audio>

注释: control 属性供添加播放、暂停和音量控件。

1. <audio> 与 </audio> 之间插入的内容是供不支持 audio 元素的浏览器显示的。

<audio src="song.ogg" controls="controls">

Your browser does not support the audio tag.

</audio>

1. audio 元素允许多个 source 元素。source 元素可以链接不同的音频文件。浏览器将使用第一个可识别的格式。

<audio controls="controls">

<source src="song.ogg" type="audio/ogg">

<source src="song.mp3" type="audio/mpeg">

Your browser does not support the audio tag.

</audio>

1. <audio> 标签的属性

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| [autoplay](http://www.w3school.com.cn/tags/att_audio_autoplay.asp" \o "HTML5 <audio> autoplay 属性) | autoplay | 如果出现该属性，则音频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/tags/att_audio_controls.asp" \o "HTML5 <audio> controls 属性) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [loop](http://www.w3school.com.cn/tags/att_audio_loop.asp" \o "HTML5 <audio> loop 属性) | loop | 如果出现该属性，则每当音频结束时重新开始播放。 |
| [preload](http://www.w3school.com.cn/tags/att_audio_preload.asp" \o "HTML5 <audio> preload 属性) | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/tags/att_audio_src.asp" \o "HTML5 <audio> src 属性) | *url* | 要播放的音频的 URL。 |

1. Audio/Video 方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [addTextTrack()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/tags/av_met_addtexttrack.asp" \o "HTML5 Audio/Video DOM addTextTrack() 方法) | 向音频/视频添加新的文本轨道 |
| [canPlayType()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/tags/av_met_canplaytype.asp" \o "HTML5 Audio/Video DOM canPlayType() 方法) | 检测浏览器是否能播放指定的音频/视频类型 |
| [load()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/tags/av_met_load.asp" \o "HTML5 Audio/Video DOM load() 方法) | 重新加载音频/视频元素 |
| [play()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/tags/av_met_play.asp" \o "HTML5 Audio/Video DOM play() 方法) | 开始播放音频/视频 |
| [pause()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/tags/av_met_pause.asp" \o "HTML5 Audio/Video DOM pause() 方法) | 暂停当前播放的音频/视频 |

1. 例题:音乐播放器实现播放、暂停、静音、进度、音量功能

<body>

<audio id='audio' src='3.mp3'> </audio>

<progress id='prog' value='0'></progress>

<input type="button" value="播放" onclick="App.play()">

<input type="button" value="暂停" onclick="App.pause()">

<input type="button" value="静音" onclick="App.isMuted()">

<input id="range" type="range" min=1 max=100 onchange=

"App.setVolumn()" >

</body>

<script type="text/javascript">

(function(){

return App={

//return不可以省略

play:function(){

audio.play();

setInterval(function(){

//计时器不可以省略

prog.max=audio.duration;

prog.value=audio.currentTime;

},100)

},

pause:function(){

audio.pause();

},

isMuted:function(){

if (audio.muted==false) {

var val=range.value;

audio.muted=true;

//audio.muted=false不可以省略

range.value=0;

}else{

audio.muted=false

//audio.muted=false不可以省略

range.value=val;

};

},

setVolumn:function(){

audio.volume=range.value/100;

//可先实现音量功能，再实现静音功能。

}

}

}())

</script>

注释:<input type="range"/>标签的onchange事件只有当用户拖动滑动条时，才会执行onchange事件函数。JavaScript代码操作滑动条时不会执行onchange事件函数。因此实现静音功能按钮时不可以只设置<input type= "range"/>标签的value值。(可能出于安全考虑，只能手动触发onchange事件函数。)

1. **代码块(函数)的封装**

(function(){

//将不同功能模块块作为一个对象的不同方法，将这个对象作为闭包的返回值。

var Move=(function(){

var moveLeft=function(ele){功能模块1};

var moveRight=function(ele){功能模块2};

var moveTop=function(ele){功能模块3};

var moveBottom=function(ele){功能模块4};

return{

moveLeft:moveLeft,

moveRight:moveRight,

moveTop:moveTop,

moveBottom:moveBottom

}

}());

Move.moveLeft(test);

}())

HTML 5 视频

1. HTML5 规定了一种通过 video 元素来包含视频的标准方法。
2. video 元素支持三种视频格式：①.ogg ②.mpeg4 ③.webm

注释: 不同浏览器对不同视频格式的支持情况不同。例如:Safari3.0+、IE9.0+浏览器只支持mpeg4格式，Firefox、Opera浏览器只支持ogg和webm格式，chrome浏览器支持三种视频格式。

1. <video src="movie.ogg" controls="controls"> </video>

注释: ①src属性供添加视频地址。②control 属性供添加播放、暂停和音量控件。③视频窗体和播放等控件共同构成video元素，当独立设置video元素的width或height属性值时，video元素等比例缩放，当共同设置video的width和height属性值时，视频窗体和播放等控件可能发生视觉分离。

1. <video> 与 </video> 之间插入的内容是供不支持 video 元素的浏览器显示的。

<video src="movie.ogg" width="320" height="240"

controls="controls">

Your browser does not support the video tag.

</video>

1. video 元素允许多个 source 元素。source 元素可以链接不同的视频文件。浏览器将使用第一个可识别的格式。

<video width="320" height="240" controls="controls">

<source src="movie.ogg" type="video/ogg">

<source src="movie.mp4" type="video/mp4">

Your browser does not support the video tag.

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| [autoplay](http://www.w3school.com.cn/tags/att_video_autoplay.asp" \o "HTML5 <video> autoplay 属性) | autoplay | 如果出现该属性，则视频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/tags/att_video_controls.asp" \o "HTML5 <video> controls 属性) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [height](http://www.w3school.com.cn/tags/att_video_height.asp" \o "HTML5 <video> height 属性) | *pixels* | 设置视频播放器的高度。 |
| [loop](http://www.w3school.com.cn/tags/att_video_loop.asp" \o "HTML5 <video> loop 属性) | loop | 如果出现该属性，则当媒介文件完成播放后再次开始播放。 |
| [preload](http://www.w3school.com.cn/tags/att_video_preload.asp" \o "HTML5 <video> preload 属性) | preload | 如果出现该属性，则视频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/tags/att_video_src.asp" \o "HTML5 <video> src 属性) | *url* | 要播放的视频的 URL。 |
| [width](http://www.w3school.com.cn/tags/att_video_width.asp" \o "HTML5 <video> width 属性) | *pixels* | 设置视频播放器的宽度。 |

</video>

1. <video> 标签的属性
2. 视频播放器实现播放、暂停、全屏、静音、进度以及音量功能。

<video id='video' src='HTML5.mp4' width='600'>

</video>

<div class="controls">

<input type="button" value="播放" onclick="App.play()" />

<input type="button" value="暂停" onclick="App.pause()" />

<input type="button" value="静音" onclick="App.isMuted()" />

<input type="button" value="全屏" onclick="App. fullScreen()" />

<span>播放进度</span>

<progress id='prog' value='0' ></progress>

<span>音量</span>

<input type="range" id="range" min=0 max=100

onchange="App.setVolumn()" >

</div>

<script type="text/javascript">

var App=(function(){

var $=function(id){

return document.getElementById(id)

};

var video=$('video');

return {

play:function(){

video.play();

setInterval(function(){

prog.value=video.currentTime;

prog.max=video.duration;

},1000);

},

pause:function(){

video.pause()

},

isMuted:function(){

if (video.muted==false) {

var vol=range.value;

video.muted=true;

range.value=0;

}else{

video.muted=false;

range.value=vol;

}

},

fullScreen:function(){

if (video.webkitRequestFullScreen) {video.webkitRequestFullScreen();};

if (video.mozRequestFullScreen) {video.mozRequestFullScreen();};

//浏览器兼容，如果有此方法(无括号)，则**调用**该方法(有括号)。

},

setVolumn:function(){

video.volume=range.value/100;

}

}

}())

</script>

注释:<input type="range"/>标签的onchange事件只有当用户拖动滑动条时，才会执行onchange事件函数。JavaScript代码操作滑动条时不会执行onchange事件函数。因此实现静音功能按钮时不可以只设置<input type= "range"/>标签的value值。(可能出于安全考虑，只能手动触发onchange事件函数。)

HTML 5应用程序缓存(离线缓存)

1. 使用 HTML5，通过创建 cache manifest 文件，可以轻松地创建 web 应用的离线版本。HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。
2. 应用程序缓存为应用带来三个优势：①离线浏览 - 用户可在应用离线时使用它们。②速度 - 已缓存资源加载得更快。③减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。
3. Cache Manifest 操作步骤

(1)设置**<html>标签的manifest属性**

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（**除非在 manifest 文件中直接指定**了该页面）。manifest 文件的建议的文件扩展名是：".appcache"。

请注意，manifest 文件需要配置**正确的MIME-type**，即 "text/cache- manifest"。必须在 web 服务器上进行配置。

1. Manifest文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

①CACHE MANIFEST - 在此标题下列出的文件将在首次下载后进行缓存

②NETWORK - 在此标题下列出的文件需要与服务器的连接，且不会被缓存

③FALLBACK - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

**CACHE MANIFEST**

第一行，CACHE MANIFEST，是必需的：

CACHE MANIFEST

/theme.css

/logo.gif

/main.js

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

**NETWORK**

下面的 NETWORK 小节规定文件 "login.asp" 永远不会被缓存，且离线时是不可用的：

NETWORK:

login.asp

可以使用星号来指示所有其他资源/文件都需要因特网连接：

NETWORK:

\*

**FALLBACK**

下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 "offline. html" 替代 /html5/ 目录中的所有文件：

FALLBACK:

/html5/ /404.html

注释：第一个 URI 是资源，第二个是替补。

更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

用户清空浏览器缓存

**manifest 文件被修改**（一般为注释行被修改）

由程序来更新应用缓存

1. 实例——完整的Manifest文件

CACHE MANIFEST

**//注意:无冒号。**

# 2012-02-21 v1.0.0

//在manifest文件中，#号表示注释内容。

/theme.css

/logo.gif

/main.js

NETWORK:

login.asp

FALLBACK:

/html5/ /404.html

重要的提示：以 "#" 开头的是注释行，但也可满足其他用途。一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使您修改了服务器上的文件。为了确保浏览器更新缓存，您需要更新 manifest 文件(一般情况下更新注释行)。

注释：浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个**站点** 5MB）。

更新策略:改变时间戳后第一次刷新更新mainfest文件，但是未用新文件内容，仍然是上次的缓存，第二次刷新，这时才会用更新的文件缓存。

1. 使用nginx-1.11.0代理服务器实现web应用缓存
2. 将nginx-1.11.0安装包放在C盘根目录，双击nginx.exe开启服务器。

打开浏览器，进入http://localhost/页面，此时打开的是服务器上的文件(C:\nginx-1.11.0\html)，证明服务器已开启。

(2)配置manifest 文件的资源媒体类型

打开C:\nginx-1.11.0\conf\mime.types,添加text/cache-manifest mf manifest appcache;(引号不可省略)

1. 打开CMD重新加载服务器

cd.. → cd .. →cd nginx-1.11.0 →nginx -s reload

注释:nginx -s reload重新加载 nginx -s stop关闭服务器(关闭服务器后仍然可以访问已缓存的页面，但无法访问未缓存的页面)

(4)在C:\nginx-1.11.0\html路径下创建web项目(含.html .css .js文件)

1. 在web项目中创建manifest文件，定义manifest属性。

创建a.appcache文件，定义<html manifest='a.appcache'>

1. 配置manifest文件

CACHE MANIFEST

#20160609001

CACHE:

a.css

a.html

NETWORK:

a.js

FALLBACK:

(6)打开html文件

<http://localhost/manifest/a.html>

1. 修改服务器上的文件，更改注释行时间戳，刷新页面

注意: ①在web项目中必须引入a.js文件和a.css文件。②CACHE MANIFEST无冒号。③除了a.html外(已定义manifest属性，默认离线缓存)，其他文件(a.css与a.js)必须放在CACHE:或NETWORK:中。④CACHE:中的文件更改之后需要更新mainfest文件(一般更改注释行时间戳)，并两次刷新。NETWORK:中的文件更改之后不需要更新mainfest文件，且只需要一次刷新。

## 新的API

HTML 5 Web 存储(本地缓存)

1. 在客户端存储数据

HTML5 提供了两种在客户端存储数据的新方法：

(1)localStorage - 没有时间限制的数据存储

(2)sessionStorage - 针对一个 session 的数据存储

之前，这些都是由 cookie 完成的。但是 cookie 不适合大量数据的存储，因为它们由每个对服务器的请求来传递，这使得 cookie 速度很慢而且效率也不高。

在 HTML5 中，数据不是由每个服务器请求传递的，而是只有在请求时使用数据。它使在不影响网站性能的情况下存储大量数据成为可能。对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。HTML5 使用 JavaScript 来存储和访问数据。

1. localStorage 方法

localStorage 方法存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用。

localStorage.setItem('键',值); ——保存某个缓存键值对

localStorage.getItem('键'); ——读取某个缓存键值对

localStorage.removeItem('键'); ——删除某个缓存键值对

localStorage.clear(); ——删除所有缓存键值对

注释：①JSON.stringify(对象)——JSON转化为字符串

②JSON.parse(字符串)——字符串转化为JSON

③存储内存限制:5M。

④清除控制台: Ctrl+L

1. sessionStorage 方法

sessionStorage 方法针对一个 session 进行数据存储。当用户关闭浏览器窗口后，数据会被删除。

sessionStorage.setItem('键',值); ——保存某个缓存键值对

sessionStorage.getItem('键'); ——读取某个缓存键值对

sessionStorage.removeItem('键'); ——删除某个缓存键值对

sessionStorage.clear(); ——删除所有缓存键值对

注释：①key 的唯一性（重复设置，新值会覆盖旧值）

②localStorage与sessionStorage的key互不干扰、可以重复。

③存储内存限制:5M。

HTML5 地理定位

1. 定位用户的位置

HTML5 Geolocation API 用于获得用户的地理位置。鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

1. HTML5 - 使用地理定位并处理错误和拒绝

<p id="demo">点击这个按钮，获得您的坐标：</p>

<button onclick="getLocation()">试一下</button>

<script>

var x = document.getElementById("demo");

function getLocation() {

//如果浏览器支持

if (navigator.geolocation) {

//两个参数都为函数

navigator.geolocation.getCurrentPosition(showPosition, showError);} else {

x.innerHTML="Geolocation is not supported by this browser.";}

}

//第一个实参:showPosition() 函数获得并显示经度和纬度

function showPosition(position) {

x.innerHTML="Latitude:"+position.coords. latitude +

"<br />Longitude: " + position.coords.longitude;

}

//第二个实参:showError()函数用于处理错误

//该函数仅在获取用户位置失败时运行

function showError(error) {

switch (error.code) {

case error.PERMISSION\_DENIED:

x.innerHTML = "User denied the request for Geolocation."

break;

case error.POSITION\_UNAVAILABLE:

x.innerHTML = "Location information is unavailable."

break;

case error.TIMEOUT:

x.innerHTML = "The request to get user location timed out."

break;

case error.UNKNOWN\_ERROR:

x.innerHTML = "An unknown error occurred."

break;

}

}

</script>

注释:Permission denied - 用户不允许地理定位。Position unavailable - 无法获取当前位置。Timeout - 操作超时。

requestAnimationFrame

1. 动画的实现方式:

JavaScript: setTimout()/setInterval()

HTML5: canvas

HTML5: requestAnimationFrame()

CSS3: animation+@keyframes

CSS3: transition

JavaScript: jQuery Velocity.js

动画算子

1. requestAnimationFrame

requestAnimationFrame 是专门为实现高性能的帧动画而设计的一个API。该方法的原理与setTimeout/setInterval类型，通过递归调用同一方法来不断更新画面以达到动起来的效果。该API在运行时浏览器会自动优化方法的调用，并且如果页面不是激活状态下的话，动画会自动暂停，有效节省了CPU开销。

注意:requestAnimationFrame不是自己指定回调函数运行的时间，而是随着浏览器内建的刷新频率来执行回调，浏览器会基于当前页面是否可见、CPU的负荷情况等来自行决定最佳的帧速率，从而更合理地使用CPU，达到浏览器所能实现动画的最佳效果。

1. 基本语法

可以直接调用，也可以通过window来调用，接收一个函数作为回调，返回一个ID值，通过把这个ID值传给cancelAnimationFrame()可以取消该次动画。

4. 实例:模拟一个进度条动画，在step函数中将进度加1然后再更新到div宽度上，在进度达到100之前，一直重复这一过程。

<script>

window.requestAnimationFrame = window.requestAnimation Frame||window.webkitRequestAnimationFrame||window.mozRequestAnimationFrame||window.msRequestAnimationFrame;

btn1.onclick = function() {

var progress = 0;

requestAnimationFrame(step)

function step() {

div.innerHTML = div.style.width = progress + '%';

progress++;

if (progress<=100) {

move=requestAnimationFrame(step);

};

};

};

btn2.onclick=function(){

cancelAnimationFrame(move);

};

</script>

fullScreen

现在主流浏览器基本上实现了全屏效果，但是不同浏览器实现不一样：

【进入和退出全屏】

// Webkit (works in Safari5.1 and Chrome 15)

element.webkitRequestFullScreen();

document.webkitCancelFullScreen();

// Firefox 10+

element.mozRequestFullScreen();

document.mozCancelFullScreen();

// W3C 提议

element.requestFullscreen();

document.exitFullscreen();

注：可能出于安全考虑，只能手动触发才能实现全屏，浏览器自动执行无效果。

Web Worker

1. index.html文件

<script type="text/javascript">

var wk;

wk = new Worker('webworker.js');

wk.onmessage = function(data){

console.log(data);

};

</script>

2.webworker.js

var i=100;

postMessage(i);

注释:①JavaScript语言采用的是单线程模型，所有任务排成一个队列，一次只能做一件事情。Web Worker的目的就是为JavaScript创造多线程环境，在主线程运行的同时，子线程在后台运行，两者互不干扰。等到子线程完成计算任务，再把结果返回给主线程。因此，每一个子线程就好像一个“工人”(Worker)，默默地完成自己的工作。这样做的好处是，一些高计算量或高延迟的工作，被worker线程负担了，所以主线程就会很流程，不会被阻塞或拖慢。②同域限制:子线程加载的脚本文本，必须与主线程的脚本文件在同一个域。(C:\nginx-1.11.0\html\webworker或者使用Hbuild)③DOM限制:子线程所在的全局对象，与主线程不同，它无法读取网页的DOM对象。即:document、window、parent这些对象，子线程无法得到。(但是，navigator对象和location对象可以获取)。④脚本限制: 子线程无法读取网页的全局变量和函数，也不能执行alert和confirm方法，不过可以执行setInterval和setTimerOut，以及可以使用XMLHttpRequest对象发出的Ajax请求。⑤文件限制:子线程无法读取本地文件，即子线程无法打开本机的文件系统(file://)，它所加载的脚本，必须来自网络。

其他

1. 同源策略

协议相同、域名相同、端口相同。

页面地址组成：web协议+主机域名+端口+页面路径

页面地址： http://127.0.0.1:8020/20160408/new\_file2.html

协议不同 <http://a.>com <https://a.com>

域名不同 <http://a.com> <http://b.com>

端口不同 <http://a.com>:8080 <http://a.com>:8360

1. 假:NaN undefined null 0 空字符串 不能找到的元素 false

真: 非0数字 非空字符串 函数 true 能找到的元素 [] {}

注意:数字0为假。

## 视口

1. 混合式开发:混合式开发充分利用HTML5开发效率高、跨平台、能快速适应各种需求和Native App良好用户交互体验的优势。
2. 在CMD中查看本机IP地址: 输入ipconfig，查看IPv4地址
3. 设置子元素浮动之后，通过给父元素添加clear类名的方式清除浮动。

.clear:after{content:"";display:block;height:0;clear:both;visibility:hidden}

注意:①定义浮动，必清浮动。②inline-block尽可能替代float。

1. px、em与rem

px:像素;

em:相对于父级元素字体大小的倍数;

rem:相对于HTML根元素(HTML)字体大小的倍数;

1. viewport
2. viewport概念:移动设备上的viewport就是设备的屏幕上能用来显示我们的网页的那一块区域。
3. CSS的1像素与设备的1像素

设备像素：设备屏幕的物理像素，任何设备的物理像素的数量都是固定的。

CSS像素：为web开发者创造的，在CSS(JavaScript)中使用的一个抽象的层。

在css中我们一般使用px作为单位，在桌面浏览器中css的1个像素往往都是对应着电脑屏幕的1个物理像素，这可能会造成我们的一个错觉，那就是css中的像素就是设备的物理像素。但实际情况却并非如此，css中的像素只是一个抽象的单位，在不同的设备或不同的环境中，css中的1px所代表的设备物理像素是不同的。

如果移动设备的物理像素密度(分辨率)提高一倍，并且屏幕尺寸不变，此时同样大小的屏幕上物理像素点的数量多了一倍，一个CSS像素所代表的物理像素是之前的二倍。另外，当用户把页面放大一倍，那么css中1px所代表的物理像素也会增加一倍；反之把页面缩小一倍，css中1px所代表的物理像素也会减少一倍。

在移动端浏览器以及某些桌面浏览器中，window对象有一个devicePixelRatio属性，它的官方的定义为：设备物理像素和设备独立像素的比例，也就是 devicePixelRatio = 物理像素 / 独立像素。css中的px就可以看做是设备的独立像素，所以通过devicePixelRatio，我们可以知道该设备上一个css像素代表多少个物理像素。例如，在Retina屏的iphone上，devicePixelRatio的值为2，也就是说1个css像素相当于**4**个物理像素。

注释: 设备的分辨率越大，css中1px代表的物理像素就会越多，devicePixelRatio的值也越大。只有这样才能让1px的东西在屏幕上的大小与那些低分辨率的设备差不多，不然就会因为太小而看不清。

总结: **屏幕越大、分辨率越大、页面放大越多**，devicePixelRatio值越大。

1. 三个viewport(视口)的理论

布局视口: layout viewport

该视口与移动设备浏览器屏幕宽度不再相关联，而是完全独立，我们称之为布局视口。

视觉视口: visual viewport

指用户正在看到的网站区域。用户可以通过缩放来操作视觉视口，同时不会影响布局视口，布局视口仍保持原来的宽度。视觉视口与设备屏幕一样宽，并且他的CSS像素的数量会随着用户缩放而改变。

理想视口: ideal viewport

显示在理想视口中的网站拥有最理想的浏览和阅读的宽度，用户刚进入页面时也不再需要缩放，并且没有横向滚动条。只有当页面里添加meta视口标签时，理想视口才会生效。

注释:①布局视口等于视觉视口时产生完美视口。②定义理想视口是浏览器的工作，而不是设备或者操作系统的工作。因此，同一设备上的不同浏览器拥有不同的理想视口。但是也跟设备有关。

1. 利用meta标签对viewport进行控制

问题:移动设备默认的viewport是layout viewport，也就是那个比屏幕要宽的viewport(会出现横向滚动条)，但在进行移动设备网站的开发时，我们需要的是ideal viewport。要得到ideal viewport就必须把默认的layout viewport的宽度设为移动设备的屏幕宽度。

解决方法:<meta name="viewport" content="initial-scale=1.0, minimum-scale=1.0,maximum-scale=1.0,width=device-width,user-scalable=no" />

**该meta标签的作用是让当前layout viewport的宽度等于设备的宽度**，同时不允许用户手动缩放。**（即：与设备尺寸数值相等的CSS像素即会覆盖整个屏幕宽度。将物理像素、CSS像素与屏幕尺寸三者进行连接）**

**注意:在不同的屏幕上(普通屏幕 vs retina屏幕)，css像素所呈现的大小(物理尺寸)是一致的，不同的是1个css像素所对应的物理像素个数是不一致的。**

|  |  |
| --- | --- |
| width | 设置**layout viewport** 的宽度，为一个正整数，或字符串"width-device"(含义为布局视口的宽度等于视觉视口的宽度) |
| initial-scale | 设置页面的初始缩放值，为一个数字，可以带小数 |
| minimum-scale | 允许用户的最小缩放值，为一个数字，可以带小数 |
| maximum-scale | 允许用户的最大缩放值，为一个数字，可以带小数 |
| height | 设置**layout viewport**  的高度，这个属性对我们并不重要，很少使用 |
| user-scalable | 是否允许用户进行缩放，值为"no"或"yes", no 代表不允许，yes代表允许 |

注释: ①这些属性可以同时使用，也可以单独使用或混合使用，同时使用时用逗号隔开。②缩放是相对于 ideal viewport来进行缩放的，当对ideal viewport进行100%的缩放，也就是缩放值为1。

6. 移动端自适应:水平方向(**width**、margin-left、padding-left、margin- right、padding-right)不出现像素，百分比替代像素。

7.打电话、发短信、发邮件

<a href="tel:13567153653">电话</a>

<a href="sms:13567153653">短信</a>

<a href="Mailto:254060001@qq.com">邮件</a>

## 触屏事件

1. Event 对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态。

标准 **Event** 属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [bubbles](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_bubbles.asp) | 返回布尔值，指示事件是否是冒泡事件类型。 |
| [cancelable](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_cancelable.asp) | 返回布尔值，指示事件是否可拥可取消的默认动作。 |
| **[currentTarget](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_currenttarget.asp)** | 返回其事件监听器触发该事件的元素。 |
| [eventPhase](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_eventphase.asp) | 返回事件传播的当前阶段。 |
| **[target](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_target.asp)** | 返回触发此事件的元素（事件的目标节点）。 |
| [timeStamp](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_timestamp.asp) | 返回事件生成的日期和时间。 |
| [type](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_type.asp) | 返回当前 Event 对象表示的事件的名称。 |

标准 **Event** 方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [initEvent()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_initevent.asp) | 初始化新创建的 Event 对象的属性。 |
| [preventDefault()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_preventdefault.asp) | 通知浏览器不要执行与事件关联的默认动作。 |
| [stopPropagation()](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_stoppropagation.asp) | 不再派发事件。 |

鼠标/键盘**Event**属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [altKey](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_altkey.asp) | 返回当事件被触发时，"ALT" 是否被按下。 |
| [button](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_button.asp) | 返回当事件被触发时，哪个鼠标按钮被点击。 |
| [clientX](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_clientx.asp) | 返回当事件被触发时，鼠标指针的水平坐标。 |
| [clientY](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_clienty.asp) | 返回当事件被触发时，鼠标指针的垂直坐标。 |
| [ctrlKey](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_ctrlkey.asp) | 返回当事件被触发时，"CTRL" 键是否被按下。 |
| [metaKey](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_metakey.asp) | 返回当事件被触发时，"meta" 键是否被按下。 |
| [relatedTarget](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_relatedtarget.asp) | 返回与事件的目标节点相关的节点。 |
| [screenX](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_screenx.asp) | 返回当某个事件被触发时，鼠标指针的水平坐标。 |
| [screenY](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_screeny.asp) | 返回当某个事件被触发时，鼠标指针的垂直坐标。 |
| [shiftKey](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/event_shiftkey.asp) | 返回当事件被触发时，"SHIFT" 键是否被按下。 |

1. 触屏事件

步骤:点击 → 滑动 → 抬起

touchstart事件:当手指触摸屏幕的时候触发。

touchmove事件:当手指在屏幕上滑动的时候连续触发。

touchend事件:当手指从屏幕上离开的时候触发。

touchcancel事件:当系统停止跟踪触摸的时候触发。

注释: ①touchstart事件——在绑定元素上点击时触发。②touchmove事件——在绑定元素上点击，滑动过程持续触发(即使已离开绑定元素范围)。③touchend事件——在绑定元素上点击，抬起时触发(即使已离开绑定元素范围)。④touchcancel事件——触发的两种情况: (1)点击与滑动间隔时间过长。(2)点击鼠标右键。

注意:touchmove与onmousemove的对比：

onmousemove:鼠标在被绑定元素上移动，就会持续触发 onmousemove 事件(无论鼠标是否按下)。

touchmove:鼠标在被绑定元素上按下，在抬起之前移动，就会持续触发touchmove事件(无论鼠标是否移出绑定元素范围)。

即:onmousemove需要鼠标在绑定元素上，但不管是否点击。

ouchmove需要鼠标在绑定元素上点击，但不管移动范围。

1. 跟踪触屏的特性

touches:当前屏幕上所有触摸点的集合列表(touch对象数组)。

targetTouches: 绑定事件的DOM元素上触摸点的集合列表(touch对象数组)。

changedTouches: 触发事件时改变的触摸点的集合列表(touch对象数组)。

单指操作时，数组中的对象最多有一个，在点击→滑动→抬起的全过程：changedTouches数组存储touchstart、touchmove、touchend、touchcancel四个事件对象。touches与targetTouches数组仅存储touchstart与touchmove事件对象。

1. Touch对象的属性

clientX：触摸目标在视口中的x坐标。

clientY：触摸目标在视口中的y坐标。

identifier：标识触摸的唯一ID。

pageX：触摸目标在页面中的x坐标。

pageY：触摸目标在页面中的y坐标。

screenX：触摸目标在屏幕中的x坐标。

screenY：触摸目标在屏幕中的y坐标。

target：触摸的DOM节点目标。

1. 在代码中书写debugger;可以起到在当前行设置断点的效果。
2. 手势(不是事件):

tap点击;

doubletap双击;

longtap长按;

swipeleft/right/top/bottom;滑动

多指操作(了解)

注释: 双击事件不能超过500ms，长按事件超过700ms。

1. 触屏事件实现手势效果代码模板

var $=function(id){

return document.getElementById(id);

};

var Swip=function(ele,**callback**){

**var startX, startY, endX, endY, moveX, moveY, startTime, endTime, changeTime;**

var element=$(ele);

function touchStart(event){

event.preventDefault();

var touch=event.touches[0];

startX=touch.clientX;

startY=touch.clientY;

startTime=new Date()-0;

};

function touchMove(event){

event.preventDefault();

var touch=event.touches[0];

};

function touchEnd(event){

event.preventDefault();

var touch=event.changedTouches[0];

endX=touch.clientX;

endY=touch.clientY;

moveX=endX-startX;

moveY=endY-startY;

endTime=new Date()-0;

changeTime=endTime-startTime;

//callback回调函数写在每一个判断语句尾部

};

function touchCancel(event){

event.preventDefault();

var touch=event.changedtouches[0];

};

element.addEventListener('touchstart',touchStart,false);

element.addEventListener('touchmove',touchMove,false);

element.addEventListener('touchend',touchEnd,false);

element.addEventListener('touchcancel',touchCancel,false);

};

注释:①通过jQuery库的on()方法绑定事件，则无法通过event.touches获取触摸事件对象。

1. 滑动手势

<script>

(function(){

var $=function(id){

return document.getElementById(id);

};

var Swip=function(ele,callback){

var startX,startY,endX,endY,moveX,moveY;

var element=$(ele);

function touchStart(event){

event.preventDefault();

var touch=event.touches[0];

startX=touch.clientX;

startY=touch.clientY;

};

function touchMove(event){

event.preventDefault();

var touch=event.touches[0];

};

function touchEnd(event){

event.preventDefault();

var touch=event.changedTouches[0];

endX=touch.clientX;

endY=touch.clientY;

moveX=endX-startX;

moveY=endY-startY;

if (Math.abs(moveX)>=Math.max(Math.abs(moveY),20)) {

if (moveX>0) {

console.log("Swip Right!");

callback&&callback();

}else{

console.log("Swip Left!");

callback&&callback();

};

};

if (Math.abs(moveY)>=Math.max(Math.abs(moveX),20)) {

if (moveY>0) {

console.log("Swip Dowm!");

callback&&callback();

}else{

console.log("Swip Up!");

callback&&callback();

};

};

};

function touchCancel(event){

event.preventDefault();

var touch=event.changedtouches[0];

};

element.addEventListener('touchstart',touchStart,false);

element.addEventListener('touchmove',touchMove,false);

element.addEventListener('touchend',touchEnd,false);

element.addEventListener('touchcancel',touchCancel,false);

};

Swip('test',function(){

console.log('完成');

});

}())

</script>

1. 纯数字符串与数字相加减

<script>

alert('1'+5) //弹出15

alert('1'-(-5)) //弹出6

</script>

## CSS3

1. **CSS选择器**

|  |  |  |  |
| --- | --- | --- | --- |
| **分类** | **选择器** | **例子** | **例子描述** |
| **属性选择器** | [[attribute]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attribute.asp" \o "CSS [attribute] 选择器) | p[target] | 选择带有 target 属性的所有p元素。  (注1:与jQuery类似，属性过滤筛选无冒号)  (注2:与jQuery类似，p可省略，选择带有target属性的所有元素。下同) |
| [[attribute=value]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器) | a[class="abc"] | 选择属性 class="abc" 的所有a元素。 |
| [[attribute\*=value]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器) | a[class\*="abc"] | 选择其 class 属性值中包含 "abc" 子串的所有 <a> 元素。 |
| [[attribute^=value]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器) | a[class^="abc"] | 选择其 class 属性值以 "abc" 开头的所有<a> 元素。 |
| [[attribute$=value]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器) | a[class$="abc"] | 选择其class属性值以 "abc" 结尾的所有<a> 元素。 |
| **[[]](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_attribute.asp" \o "CSS [attribute] 选择器)[]** | **P[][]** | **选择经过多次属性筛选的p元素** |
| **伪元素选择器** | [:first-line](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_first-line.asp" \o "CSS :first-line 选择器) | p:first-line | 选择每个 <p> 元素的首行。 |
| [:first-letter](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_first-letter.asp" \o "CSS :first-letter 选择器) | p:first-letter | 选择每个 <p> 元素的首字母(汉子)。 |
| [:before](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_before.asp" \o "CSS :before 选择器) | p:before | 在每个 <p> 元素的内容之前插入内容。  (注:插入的内容不可被选中) |
| [:after](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_after.asp" \o "CSS :after 选择器) | p:after | 在每个 <p> 元素的内容之后插入内容。 |
| **结构性伪类选择器** | [:root](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_root.asp" \o "CSS :root 选择器) | :root | 选择文档的根元素。  (注:根元素为HTML元素) |
| [:not(selector)](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_not.asp" \o "CSS :not(selector) 选择器) | **:not(p)** | 排除某结构元素下的<p>元素 |
| [:empty](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_empty.asp" \o "CSS :empty 选择器) | p:empty | 选择没有子元素的每个 <p> 元素  （包括文本节点）。 |
| **[:target](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_target.asp" \o "CSS :target 选择器)** | **#news:target** | **选择当前活动的 #news 元素。** |
| [element>element](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_element_gt.asp" \o "CSS element>element 选择器) | div>p | 选择父元素为 <div> 元素的所有 <p> 元素。 |
| [element+element](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_element_plus.asp" \o "CSS element+element 选择器) | **div+p** | 选择紧接在 <div> 元素之后的 <p> 元素。(注:下一个同辈元素) |
| [element1~element2](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_gen_sibling.asp" \o "CSS element1~element2 选择器) | **p~ul** | 选择前面有 <p> 元素的每个 <ul> 元素。  (注:选择p元素之后的所有ul兄弟元素) |
| [:first-child](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_first-child.asp" \o "CSS :first-child 选择器) | p:first-child | **选择属于父元素的第一个子元素的每个<p> 元素。**(注1:与jQuery中含义不同,在jQuery中为加空格筛选子元素，CSS3为不加空格筛选自己)  (注2:筛选出**兄弟元素**中排序第一的p元素) |
| [:last-child](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_last-child.asp" \o "CSS :last-child 选择器) | p:last-child | 选择属于其父元素最后一个子元素每个<p> 元素。 |
| [:only-child](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_only-child.asp" \o "CSS :only-child 选择器) | p:only-child | 选择属于其父元素的唯一子元素的每个 <p> 元素。 |
| **[:nth-child(n)](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_nth-child.asp" \o "CSS :nth-child(n) 选择器)** | **p:nth-child(2)** | **选择属于其父元素的第二个子元素的每个 <p> 元素。(注1:所有子元素都参与排序)**  **(注2:两个条件 ①排序第二；②是p元素。)** |
| [:nth-last-child(n)](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_nth-last-child.asp" \o "CSS :nth-last-child(n) 选择器) | p:nth-last-child  (2) | 同上，从最后一个子元素开始计数。 |
| [:first-of-type](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_first-of-type.asp" \o "CSS :first-of-type 选择器) | p:first-of-type | **选择属于其父元素的首个 <p> 元素的每个 <p> 元素。**  (注2:筛选出**兄弟p元素**中首个p元素) |
| [:last-of-type](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_last-of-type.asp" \o "CSS :last-of-type 选择器) | p:last-of-type | 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。  (注2:筛选出**兄弟p元素**中最后的p元素) |
| [:only-of-type](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_only-of-type.asp" \o "CSS :only-of-type 选择器) | p:only-of-type | 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。 |
| **[:nth-of-type(n)](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_nth-of-type.asp" \o "CSS :nth-of-type(n) 选择器)** | **p:nth-of-type**  **(2)** | **选择属于其父元素第二个 <p> 元素的每个 <p> 元素。****(注:只有p元素参与排序)** |
| [:nth-last-of-type(n)](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_nth-last-of-type.asp" \o "CSS :nth-last-of-type(n) 选择器) | p:nth-last-of-type(2) | 同上，但是从最后一个子元素开始计数。 |
| **UI元素状态伪类选择器** | [:focus](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_focus.asp" \o "CSS :focus 选择器) | input:focus | 选择获得焦点的 input 元素。 |
| [:enabled](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_enabled.asp" \o "CSS :enabled 选择器) | input:enabled | 选择每个启用的 <input> 元素。 |
| [:disabled](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_disabled.asp" \o "CSS :disabled 选择器) | input:disabled | 选择每个禁用的 <input> 元素 |
| [:checked](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_checked.asp" \o "CSS :checked 选择器) | input:checked | 选择每个被选中的 <input> 元素。 |
| **[::selection](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/cssref/selector_selection.asp" \o "CSS ::selection 选择器)** | **::selection** | **选择被用户选取的元素部分。**  **(注:两个冒号)** |

备注:(1)与jQuery选择器对比记忆。

(2) 在:nth-child(n)选择器中，n的取值为自然数(0和正整数集合)，含义为第n个元素。

注释:2n+1等价于odd; 2n等价于even; -3n+10含义为第1,4,7,10 ;

注意:不可以写成x+yn的形式。

(3) 在CSS1版本中的四个伪类选择器

①a:link有链接属性时; ②a:visited 链接地址已访问;

③a:hover鼠标悬停; ④a:active被用户激活;

未点击时，a处于link状态；点击过后，a处于link和visited状态；悬停时，a处于link和hover状态；点击时，a处于link、hover和active状态。考虑用户体验和CSS选择器优先级，在CSS中的设置超链接的伪类选择器属性时，伪类选择器的定义顺序不能改变。

(4) 锚点与:target选择器

锚点是网页制作中页面内的超级链接，可以起到页面内跳转功能。

:target 选择器可用于选取当前活动的目标元素。使用target选择器来对页面中某个target元素指定样式，该样式只在用户点击了页面中的超链接，并且跳转到target元素后起作用。

举例1: <a href="**#**tips">跳转</a>

<div id="tips"></div>

当目标元素(div)不是a元素时，需要将目标元素(div)的**id**属性值与锚元素(a)元素的href属性值进行绑定。

举例2-1:<a href="**#**tips">跳转</a>

<a name="tips"></a>

举例2-1:<a href="**#**tips">跳转</a>

<a id="tips"></a>

当目标元素是a元素时，既可以将目标元素(a)的**name**(推荐)属性值与锚元素(a)的href属性值进行绑定，又可以将目标元素(a)的**id**属性值与锚元素(a)的href属性值进行绑定。

注意:id绑定优先级高于name绑定；位置靠前绑定优先级高于位置靠后绑定。

1. **CSS3边框——圆角边框、盒子阴影、图片边框**
2. 圆角边框

border-radius 属性用于创建圆角。

1. 盒子阴影

盒子阴影效果 {box-shadow:投影方式 x轴偏移 y轴偏移 模糊半径 阴影扩展半径 颜色;}

其中：投影方式默认向外投影，可省略；如需设置向内投影，写inset。

注释:①支持多个阴影效果。②RGBA颜色表示backgroung-color:rgba (255,0,123,.5) alpha:通道。

(3)图片边框

border-image-source 用在边框的图片的路径。

border-image-slice 图片边框向内偏移，用来分解引入的图片，将之拆解成9份，形成一个九宫格造型。 border-image-slice:10 10 10 10。

border-image-width 图片边框的宽度。

border-image-outset 边框图像区域超出边框的量。

border-image-repeat 图像边框是否应平铺(repeat)、铺满(round)或拉伸(stretch)。

举例: border-image:**url**(border.png) 30 30 round;

**3. CSS3 背景——尺寸、定位、区域**

(1)background-**size**: length|percentage|**cover**|contain;

length: 设置背景图像的高度和宽度。

percentage: 以百分比来设置背景图像的宽高。

cover: 背景图片等比例缩放/扩展至全覆盖盒子尺寸。(常用)

contain: 背景图片等比例缩放/扩展到盒子尺寸。

(2)background-**origin** 属性规定背景图片的**定位区域**。

背景图片可以放置于 content-box、padding-box 或 border-box 区域。

padding-box 背景图像相对于内边距框来定位。

border-box 背景图像相对于边框盒来定位。

content-box 背景图像相对于内容框来定位。

(3)background-**clip** 属性规定背景的**绘制区域**。

background-clip: border-box|padding-box|content-box;

border-box 背景被裁剪到边框盒。

padding-box 背景被裁剪到内边距框。

content-box 背景被裁剪到内容框。

注释:CSS3 多重背景图片

举例:background-image:url(bg\_flower.gif),url(bg\_flower\_2.gif);

**4. CSS3 文本效果——文本阴影、自动换行**

(1)文本阴影效果 {text-shadow: x轴偏移 y轴偏移 模糊半径 颜色;}

注释:支持多个阴影效果。

1. 文本溢出处理

div{

width: 50px;

overflow: hidden;

text-overflow:ellipsis;

white-space: nowrap;

};

**5. CSS3 颜色渐变——**线性渐变、径向渐变、重复渐变

1. 线性渐变

语法:linear-gradient(渐变方向/角度,颜色1,颜色2,颜色3...);

方向:left、right、top、bottom、left top、right top、

left bottom、right bottom

例子1: background: linear-gradient(red,yellow);

例子2:background: linear-gradient(**to** bottom,red,yellow);

例子3:background: linear-gradient(45deg,red,yellow,blue);

注意:线性渐变是背景的属性值，而不是颜色的属性值。

1. 径向渐变

语法:radial-gradient(形状 大小 at 位置,颜色1,颜色2,颜色3...);

形状:circle(圆形)、ellipse(椭圆形、默认值)

大小:①像素表示，如10px。②百分比表示，如50%。

位置:center(默认值)、left、right、top、bottom、left top、

right top、left bottom、right bottom。

例子1:background: radial-gradient(red,black);

例子2:background: radial-gradient(circle,red,black);

例子3:background: radial-gradient(10px,red,black);

例子4:background: radial-gradient(80% 20%,red,black);

例子5:background: radial-gradient(**at** left,red,black);

例子6:background: radial-gradient(at left top,red,black);

注意:径向渐变时背景的属性值，而不是颜色的属性值。

(3)重复渐变

重复一个渐变，可以使用：

-moz-repeating-linear-gradient

-moz-repeating-radial-gradient

-webkit-repeating-linear-gradient

-webkit-repeating-radial-gradient

例子:background: repeating-radial-gradient(aqua 10%,blue 20%);

## 响应式布局

1. PC端传统布局方案:基于盒状模型，依赖 display属性 + position属性 + float属性。传统布局方案的灵活性很差。

Flex布局

1. Flex布局简介

http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html?utm\_source=tuicool

2009年，W3C提出了一种新的方案——Flex布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，成为未来布局的首选方案。

1. Flex布局的基本概念

Flex是Flexible Box的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。任何一个容器都可以指定为Flex布局。设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

用Flex布局的元素，称为Flex容器（flex container），简称"容器"。它的所有子元素自动成为容器成员，称为Flex项目（flex item），简称"项目"。

容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）。项目默认沿主轴排列。

注意:(1)Flex布局的常用设置——①容器元素和项目元素都为块级元素。②容器元素定义display: flex;项目元素定义display:inline-block。

(2)Flex布局的三个关键因素:**①主轴与交叉轴方向;②两个方向对齐方式;③两个方向伸缩情况。**

1. 容器的属性

以下7个属性设置在容器上。

flex 定义flex布局

flex-direction 定义主轴与交叉轴方向

flex-wrap 定义主轴方向是否换行

flex-flow 定义主轴与交叉轴方向、是否换行

justify-content 定义项目在主轴方向对齐方式

align-items 定义项目在交叉轴方向对齐方式及伸缩情况

align-content 定义多根轴线在交叉轴方向的对齐方式

注意:①最常用的三个属性为: flex-direction、justify-content、align-items。②在Web中使用flex布局，容器需要设置display:flex;ReactNative默认布局方式为flex布局，容器无需设置display:flex。

1. flex-direction属性——定义主轴与交叉轴方向

flex-direction: row | row-reverse | column | column-reverse;

①row（默认值） 主轴方向水平向右，交叉轴方向竖直向下。

②row-reverse 主轴方向水平向左，交叉轴方向竖直向下。

③column 主轴方向竖直向下，交叉轴方向水平向右。

④column-reverse 主轴方向竖直向上，交叉轴方向水平向右。

**注释:在ReactNative中——①flex-direction属性以flexDirection驼峰形式存在。②属性值只有row和column两种。③column属性值为默认值。**

1. flex-wrap属性——定义主轴方向是否换行

flex-wrap: nowrap | wrap | wrap-reverse;

①nowrap（默认） 不换行。

②wrap： 换行，第一行在上方。

③wrap-reverse 换行，第一行在下方。

**注释:在ReactNative中，wrap-reverse属性值不生效。**

1. flex-flow属性——定义主轴与交叉轴方向、是否换行

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

1. justify-content属性——定义项目在主轴方向对齐方式

justify-content: flex-start | flex-end | center | space-between | space-around;

①flex-start（默认值） 向主轴起点方向对齐

②flex-end 向主轴终点方向对齐

③center 向主轴中点方向对齐

④space-between 两端对齐，间隔相等

⑤space-around 两侧间隔相等，项目间隔是项目与边框间隔的2倍

**注释:ReactNative中使用方式与此相同。**

1. **align-items属性——定义项目在交叉轴方向对齐方式及伸缩情况**

align-items: flex-start | flex-end | center | baseline | stretch;

①flex-start 向交叉轴起点方向对齐

②flex-end 向交叉轴终点方向对齐

③center 向交叉轴中点方向对齐

④baseline 项目的第一行文字的基线对齐

⑤stretch（默认值） 如果项目未设置高度或设为auto，将占满整个容器的高度

注释: stretch为默认属性值，即项目在交叉轴方向若未设置尺寸，则占满容器尺寸；若已设置尺寸，则等于设置尺寸，既不可被拉伸又不可被压缩。

当align-items属性为其他属性值时，无论项目在交叉轴方向是否设置尺寸，都以自身尺寸为准，既不可被压缩，又不可被拉伸。

**注意:在ReactNative中baseline不生效。**

1. align-content属性——定义多根轴线在交叉轴方向的对齐方式

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

①flex-start 与交叉轴的起点对齐。

②flex-end 与交叉轴的终点对齐。

③center 与交叉轴的中点对齐。

④space-between 与交叉轴两端对齐，轴线之间的间隔平均分布

⑤space-around 每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍

⑥**stretch（默认值）** 轴线占满整个交叉轴。

1. 项目的属性

以下6个属性设置在项目上。

order 定义项目在主轴方向的排列顺序

flex-grow 定义项目在主轴方向的放大比例

flex-shrink 定义项目在主轴方向的缩小比例

flex-basis 定义在分配多余空间之前，项目占据的主轴空间

flex 定义项目在主轴方向的伸缩情况

align-self 定义单个项目在交叉轴方向的对齐方式及伸缩情况

1. order属性——定义项目在主轴方向的排列顺序

order: <integer>;

注释:数值越小，排列越靠前，默认为0。

1. flex-grow属性——定义项目在主轴方向的放大比例

flex-grow: <number>;

注释:①默认为0,即如果存在剩余空间，也**不放大**。②如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。即: 宽度=弹性宽度 \* (flex-grow/sum(flex-grow))

1. flex-shrink属性——定义项目在主轴方向的缩小比例

flex-shrink: <number>;

注释: 默认为1，即如果空间不足，该项目**将缩小**。②如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

(4)flex-basis属性——定义在分配多余空间之前，项目占据的主轴空间

flex-basis: <length> | auto;

注释: ①浏览器根据这个属性，计算主轴是否有多余空间。②默认值为auto，即项目的本来大小。③可以设置为与width或height属性一样的值。

1. flex属性

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

该属性有两个快捷值：**auto** (1 1 auto) 和 none (0 0 auto)。

注释:①auto为常用属性值，即项目在主轴方向可伸缩。②**外边距不可伸缩。**

②可设置**某个**项目的宽度为**固定**百分比，**其余**项目平**均分**配剩余空间。

.item{flex:auto} **.item1{flex:0 0 60%}**

③可设置**某个**项目的宽度为**固定**像素，**其余**项目平**均分**配剩余空间。

.item{flex:auto} **.item1{flex:0 0 100px}**

**注释:ReactNative中使用方式与此相同。**

1. align-self属性——定义单个项目在交叉轴方向的对齐方式及伸缩情况

align-self: auto | flex-start | flex-end | center | baseline | stretch;

注释: ①默认值为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。②该属性可能取6个值，除了auto，其他都与align-items属性完全一致。

**注释:ReactNative中使用方式与此相同。**

注意:子设置align-self属性，可以**覆盖**父的align-items属性。

1. Flex常用属性汇总

|  |  |  |
| --- | --- | --- |
| Flex布局 | | display:flex |
| 主轴与交叉轴方向 | | flex-direction;flex-flow |
| 主轴 | 单轴对齐方式 | justify-content |
| 单轴伸缩情况 | flex-grow ;flex-shrink;flex-basis;flex |
| 单轴是否换行 | flex-wrap;flex-flow |
| 单轴排列顺序 | order |
| 交叉轴 | 单轴对齐方式 | align-items |
| 单轴伸缩情况 | align-items |
| 多轴对齐方式 | align-content |
| 多轴伸缩情况 | align-content(与单轴伸缩情况对比) |
| 单项目对齐方式 | align-self |
| 单项目伸缩情况 | align-self(与单轴伸缩情况对比) |

注释: Flex布局的三个关键因素:①主轴与交叉轴方向;②两个方向对齐方式;③两个方向伸缩情况。

1. Flex布局实例篇

<http://www.ruanyifeng.com/blog/2015/07/flex-examples.html?bsh_bid=683103006>

注意:Flex布局可以**多层次嵌套**。

1. Flex布局兼容版本实例

.flexbox {

display: -webkit-flex;

display: flex;

-webkit-flex-flow: row nowrap;

flex-flow: row nowrap;

-webkit-justify-content: center;

justify-content: center;

align-items: center;

}

.flexbox > .item {

display: -webkit-flex;

display: flex;

-webkit-align-self: center;

align-self: center;

}

百分比布局(流式布局)

1. 目标元素宽度 / 父元素宽度 = 百分比宽度
2. box-sizing: border-box;

设置div盒子模型的width和height为border的宽高。

应用场景1:解决边框宽度及内边距对块级元素宽占据尺寸的影响，可应用于百分比布局。

实例: .inner{

float: left;

width: 25%;

padding: 5px;

border: 5px solid red;

box-sizing:border-box;

};

注释:在百分比布局中实现自适应。

媒体查询(Media Queries)

1. 媒体查询: 在不改变内容的情况下在样式中选择一种页面的布局以精确地适应不同的设备，从而改善用户体验。网页制作者针对不同的浏览器窗口尺寸来编写不同的样式，浏览器根据不同的窗口尺寸来选择使用不同的样式。
2. 格式：@media 设备类型 and (设备特性)｛样式代码｝

举例:<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 400px)" href=“tinyScreen.css” />

即:如果屏幕宽度小于400像素(max-device-width: 400px)，则加载tinyScreen.css文件。

注释:使用媒体查询实现自适应布局——①百分比宽度替代绝对宽度。②相对字体大小(em)替代绝对字体大小(px)。

rem布局

1. <script>

var fn=function(){ document.documentElement.style.fontSize=window.innerWidth/16+'px';};

fn();

window.**onresize**=fn;

</script>

注释: ①在打开页面和页面窗口尺寸改变的时机：html字体大小等于窗口宽度 的16分之一，页面内所有尺寸以html字体大小为基准(rem)。

②获取窗口(视口)的高度与宽度(不包含工具条与滚动条)。

innerWidth;innerHeight;

③浏览器的最小字体单位是12px，设置字体小于12px，仍显示12px。

④document.documentElement为文档根元素(<html>)。等价于document.getElementsByTagName('html')[0]。

注意: **innWidth为当前屏幕横向在不产生滚动条的情况下所能够容纳的最大CSS像素数。**①PC端屏幕的innerWidth为屏幕宽度。②移动端屏幕的innWidth在无meta适配标签时一般为980。③移动端屏幕的innWidth在有meta适配标签但不设置缩放比例时为移动端屏幕宽度。④移动端屏幕的innerWidth在meta适配标签并设置缩放比例时为移动端屏幕宽度除以缩放比例。

注意:不同浏览器的解析情况不同，小米自带浏览器符合以上规则(innerWidth=屏幕宽度/scale)，而安卓版UC浏览器不符合此规则(innerWidth=屏幕宽度)。

其他

1. 当元素的margin/padding设置为百分比时，以父元素的**宽度**作为基准。
2. 图片的自适应： img { max-width: 100%;}
3. **CSS代码花括号后无分号**。
4. px、rem、% 三者的选择:当尺寸较小时(margin、padding等)，可使用px，保证尺寸的精度。当尺寸较大时，应使用rem或%，保证布局的自适应。

## 个性化布局

1. 一个CSS像素对应一个物理像素，页面才能得到完美清晰的展示。如果一个CSS像素对应多个物理像素(devicePixelRatio>1)，则导致页面模糊(border: 1px问题)。如果一个物理像素对应多个CSS像素，则产生色差。

<style>

[data-dpr='2'] .tc{

font-size: .8rem;

} //针对不同像素比设备的CSS设置

</style>

<div class="tc">今天天气很好</div>

<script>

var fn=function(){

var dpr=window.devicePixelRatio;

var scale=1/dpr;

var html=document.getElementsByTagName('html')[0]

var meta=document.getElementById('vp');

html.style.fontSize=innerWidth/10+'px';

//文档根节点字体大小为屏幕宽度的十分之一

html.setAttribute('data-dpr',dpr);

meta.setAttribute('content','initial-scale='+scale+',minimum-scale='+scale+',maximum-scale='+scale+',user-scalable=no')

//针对像素比设置屏幕缩小比例

};

fn();

window.onresize=fn;

</script>

引P38:innWidth为当前屏幕横向**在不产生滚动条的情况下所能够容纳的最大CSS像素数**。①PC端屏幕的innerWidth为屏幕宽度。②移动端屏幕的innWidth在无meta适配标签时一般为980。③移动端屏幕的innWidth在有meta适配标签但未设置缩放比例时为移动端屏幕宽度。④移动端屏幕的innerWidth在有meta适配标签并设置缩放比例时为移动端屏幕宽度**除以**缩放比例。**(不同浏览器的解析情况不同)**

引P24: 移动端新添加meta标签的作用是让当前viewport的宽度等于设备的宽度，同时不允许用户手动缩放。（即：**与设备尺寸数值相等的CSS像素即会覆盖整个屏幕宽度**。）当scale等于1时，innerWidth与屏幕尺寸数值相等。

**注意:在不同的屏幕上(普通屏幕 vs retina屏幕)，css像素所呈现的大小(物理尺寸)是一致的，不同的是1个css像素所对应的物理像素个数是不一致的。**

备注: ①文档根节点字体大小为innerWidth(innerWidth=屏幕宽度/scale)的十分之一(rem布局)。②针对像素比设置缩小比例，既解决border: 1px问题，又个性化设置页面尺寸基准，保证页面布局。③在当前设置下，物理像素与CSS像素一比一对应，**并充满屏幕**。

注意: ①当像素比大于1时，**设置屏幕缩小比例为像素比的倒数**，使一个CSS像素对应一个物理像素，保证页面得到完美清晰的展示。②**当屏幕缩小时(scale<1)，屏幕视口宽度(innerWidth)增加**，文档根元素的字体大小(页面尺寸基准的像素数)增加(innerWidth/10)，页面像素数增加。缩小的同时增加CSS像素数，不会影响布局。

注释:meta标签内设置缩放比例会增加innerWidth，进而增加页面CSS像素数，保持页面整体布局。

总结:页面内所有尺寸以html字体大小为基准(rem)，当像素比大于1时，设置页面缩小比例，既可以保证一个CSS像素对应一个物理像素，又可以保证页面布局(rem基准像素数增加)。

注意:设置页面缩小比例，不会影响页面布局，只会影响清晰度与模糊度；**设置页面放大比例，会影响页面布局(浏览器最小字体显示单位为12px)。**

注意:影响页面清晰程度的只有两个因素，像素比与缩放比例。当像素比乘以缩放比例等于一时，页面最清晰，大于一时页面模糊，小于1时产生色差。文档根元素的字体大小不会影响页面清晰程度，文档根元素的字体大小有两个作用，统一页面尺寸基准和**弥补页面缩小比例**。

注意:不同浏览器对innerWidth的解析不同。

关键点:dpr(像素比)、scale(缩放比例)、1rem(尺寸基准)与**innerWidth**

1. 一像素边框模糊问题与innWidth浏览器兼容问题的解决办法(待验证)

<meta id='vp' name="viewport" content="width=device- width,initial-scale=1.0,maximum-scale=1.0,minimum-scale=1.0,user-scalable=no" />

<script>

(function(){

**var rem=innerWidth/16;**

var fn=function(){

var dpr=window.devicePixelRatio;

var scale=1/dpr;

var html=document.getElementsByTagName('html')[0]

var meta=document.getElementById('vp');

html.style.fontSize=**rem\*dpr**+'px';

html.setAttribute('data-dpr',dpr);

meta.setAttribute('content','initial-scale='+scale+',minimum-scale='+scale+',maximum-scale='+scale+',user-scalable=no');

};

fn();

window.onresize=fn;

}())

</script>

1. **移动端京东页面布局特点**

(1)结合百分比与px布局

(2)1px边框

/\*右边框\*/

.bd-r{

position: relative;

}

.bd-r:after{

height: 100%;

content: '';

width: 1px;

border-right: 1px solid black;

position: absolute;

top: 0;

right: -1px;

transform: scaleX(.5);

-webkit-transform: scaleX(.5);

z-index: 10;

}

/\*上下边框\*/

.bd-tb{

position: relative;

}

.bd-tb:before {

height: 1px;

content: '';

width: 100%;

border-bottom: 1px solid red;

position: absolute;

top: -1px;

right: 0;

transform:scaleY(.5);

-webkit-transform:scaleY(.5);

z-index: 10;

}

.bd-tb:after{

height: 1px;

content: '';

width: 100%;

border-top: 1px solid red;

position: absolute;

bottom: -1px;

right: 0;

transform:scaleY(.5);

-webkit-transform:scaleY(.5);

z-index: 10;

}

/\*下边框\*/

.bd-b{

position: relative;

}

.bd-b:after {

height: 1px;

content: '';

width: 100%;

border-top: 1px solid blue;

position: absolute;

bottom: -1px;

right: 0;

transform:scaleY(.5);

-webkit-transform:scaleY(.5);

z-index: 10;

}

1. 不同浏览器对innerWidth的解析与对Flex布局的支持情况

|  |  |  |
| --- | --- | --- |
| 浏览器 | InnerWidth解析 | Flex |
| 谷歌浏览器手机模拟器 | innerWidth=屏幕宽度/scale | 支持最新写法 |
| 小米自带浏览器 | innerWidth=屏幕宽度/scale | 支持最新写法 |
| 安卓UC浏览器 | innerWidth=屏幕宽度 | 不支持最新写法 |
| 苹果UC浏览器 | innerWidth=屏幕宽度 | 支持最新写法 |

1. 将图片设置为背景，并保持原始高宽比

div{

border: 1px solid red;

background-image:url(border.png);

}

div:before{

content: '';

padding-top: 81.63%;

display: block;

/\*图片原始尺寸:width:245px;height:200px。200/245=81.63%\*/

}

</style>

<div></div>

注释：在div中插入内容，并满足三个条件——①内容为空。②padding-top为背景图片高宽比。③display设置为block。

1. 原生JavaScript操作类名

元素.classList.add('类名') ——添加类名

元素.classList.remove('类名') ——移除类名

元素.classList.contains('类名') ——返回值为布尔值

元素.classList.toggle('类名') ——切换类名

## 动画

CSS3 2D转换-transform

1. 通过CSS3转换，我们能够对元素进行移动(translate)、旋转(rotate)、伸缩(scale)、翻转(skew)。转换是使元素改变形状、尺寸和位置的一种效果。
2. 浏览器支持

Internet Explorer 10、Firefox 以及 Opera 支持 transform 属性。Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 需要前缀 -ms-。

1. 2D转换方法: translate(); rotate(); scale(); skew(); matrix();

实例:div{

transform: rotate(30deg);

-ms-transform: rotate(30deg); /\* IE 9 \*/

-webkit-transform: rotate(30deg); /\* Safari and Chrome \*/

-o-transform: rotate(30deg); /\* Opera \*/

-moz-transform: rotate(30deg); /\* Firefox \*/

}

1. translate()——移动

transform: translate(50px,100px); //右移50px，下移100px

transform: translate(50px); //右移50px

translateX(100px); //右移100px

translateY(100px); //下移100px

注释:使用translate移动元素无需设置绝对定位。

1. rotate()——旋转

transform: rotate(30deg); //顺时针旋转30度

注释:2D旋转即为3D旋转中的绕Z轴旋转 transform:rotateZ(30deg)。

1. scale()——伸缩

transform:scale(2); //宽度和高度都放大2倍

transform:scale(1,2); //高度放大2倍

transform:scaleX(2); //宽度放大2倍

transform:scaleY(2); //高度放大2被

1. skew()——翻转

transform:skew(30deg); //以X轴为轴翻转30度

transform:skew(30deg,20deg); //以X轴为轴翻转30度

//以Y轴为轴翻转20度

transform:skewX(30deg); //以X轴为轴翻转30度

transform:skewY(30deg); //以Y轴为轴翻转30度

1. transform-origin——被转换元素的基准位置

transform-origin:center; //中心点作为基准位置(默认)

transform-origin:left; //左边中点作为基准位置

transform-origin:top; //上边中点作为基准位置

transform-origin:right; //右边中点作为基准位置

transform-origin:bottom; //底边中点作为基准位置

transform-origin:left top; //左上端点作为基准位置

transform-origin:right top; //右上端点作为基准位置

transform-origin:right bottom; //右下端点作为基准位置

transform-origin:left bottom; //左下端点作为基准位置

transform-origin:200px 100px;

//(200px,100px)相对坐标作为基准位置

1. matrix(n,n,n,n,n,n)

定义 2D 转换，使用六个值的矩阵。

http://www.zhangxinxu.com/wordpress/2012/06/css3-transform-matrix-矩阵/comment-page-2/

CSS3 3D 转换-transform

http://www.zhangxinxu.com/wordpress/2012/09/css3-3d-transform-perspective-animate-transition/

1. 3D转换可以实现元素在X轴、Y轴、Z轴方向上的移动、旋转、伸缩、以及翻转处理。
2. 转换属性

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [transform](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_transform.asp" \o "CSS3 transform 属性) | 向元素应用 2D 或 3D 转换。 | 3 |
| [transform-origin](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_transform-origin.asp" \o "CSS3 transform-origin 属性) | 允许你改变被转换元素的位置。 | 3 |
| [transform-style](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_transform-style.asp" \o "CSS3 transform-style 属性) | 规定被嵌套元素如何在 3D 空间中显示。 | 3 |
| [perspective](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_perspective.asp" \o "CSS3 perspective 属性) | 规定 3D 元素的透视效果。 | 3 |
| [perspective-origin](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_perspective-origin.asp" \o "CSS3 perspective-origin 属性) | 规定 3D 元素的底部位置。 | 3 |
| [backface-visibility](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/css3/../cssref/pr_backface-visibility.asp" \o "CSS3 backface-visibility 属性) | 定义元素在不面对屏幕时是否可见。 | 3 |

1. 2D Transform 方法

|  |  |
| --- | --- |
| **函数** | **描述** |
| matrix3d(n,n,n,n,n,n, n,n,n,n,n,n,n,n,n,n) | 定义 3D 转换，使用 16 个值的 4x4 矩阵。 |
| translate3d(x,y,z) | 定义 3D 转化。 |
| translateX(x) | 定义 3D 转化，仅使用用于 X 轴的值。 |
| translateY(y) | 定义 3D 转化，仅使用用于 Y 轴的值。 |
| translateZ(z) | 定义 3D 转化，仅使用用于 Z 轴的值。 |
| scale3d(x,y,z) | 定义 3D 缩放转换。 |
| scaleX(x) | 定义 3D 缩放转换，通过给定一个 X 轴的值。 |
| scaleY(y) | 定义 3D 缩放转换，通过给定一个 Y 轴的值。 |
| scaleZ(z) | 定义 3D 缩放转换，通过给定一个 Z 轴的值。 |
| rotate3d(x,y,z,angle) | 定义 3D 旋转。 |
| rotateX(angle) | 定义沿 X 轴的 3D 旋转。 |
| rotateY(angle) | 定义沿 Y 轴的 3D 旋转。 |
| rotateZ(angle) | 定义沿 Z 轴的 3D 旋转。 |
| perspective(n) | 定义 3D 转换元素的透视视图。 |

1. translate()——移动

transform:translateX(50px);

transform:translateY(100px);

transform:translateZ(20px);

或者：

transform:translateX(50px) translateY(100px) translateZ(20px);

1. rotate()——旋转

transform:rotateX(45deg);

transform:rotateY(45deg);

transform:rotateZ(45deg);

或者：

transform:rotateX(45deg) rotateY(45deg) rotateZ(45deg);

1. scale()——伸缩

transform:scaleX(0.5);

transform:scaleY(1);

transform:scaleZ(2);

或者：

transform:scaleX(0.5) scaleY(1) scaleZ(2);

1. skew()——翻转

transform:skewX(45deg);

transform:skewY(45deg);

和2D效果类似

注意:翻转只有两个方向。

CSS3 过渡-transition

1. CSS3中的动画分为Animation功能与Transition功能，这两种功能都可以通过改变CSS中的属性值来产生动画效果。

Animation功能支持通过关键帧的指定在页面上产生复杂的动画效果。

Transition功能支持从一个属性值平滑过渡到另一个属性值。

注释：Animations——①支持关键帧(中间状态)的设定。

②支持动画暂停效果。

③支持延迟时间与动画结束后的状态设定。

Transition——①不支持中间状态的设定。

②不支持动画暂停效果。

③不支持延迟时间与动画结束后的状态设定。

1. CSS3 过渡:通过 CSS3，我们可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种CSS样式变换为另一种CSS样式时为元素添加动态效果。

注释:①规定CSS样式。②规定动态效果时长。

1. 语法:transition:属性名称 过渡时间 速度曲线 延迟时间

,属性名称 过渡时间 速度曲线 延迟时间

属性名称——①默认值(all) ②width

过渡时间——①默认值(0) ②3s ③2000ms

速度曲线——①默认值(ease)(慢-快-慢) ②linear(匀速)

③ease-in(慢-块) ④ease-out(快-慢)

延迟时间——①默认值(0) ②3s ③2000ms

注释:过渡时间不可省略，其余都可省略。

1. 实例1——应用于单项属性的过渡效果

div{transition: width 2s;

-moz-transition: width 2s; /\* Firefox 4 \*/

-webkit-transition: width 2s; /\* Safari 和 Chrome \*/

-o-transition: width 2s; /\* Opera \*/}

div:hover{width:300px;}

1. 实例2——应用于多项属性的过渡效果

如需添加多个样式的过渡效果，则用逗号隔开

div{transition: width 2s, height 3s, transform 5s;}

1. 实例3——应用于所有属性的过渡效果

div{transition:2s;}

注释:默认值为all。

1. 多步动画

div{

transition: width 2s, height 2s, transform 2s;

-moz-transition: width 2s, height 2s, -moz-transform 2s;

-webkit-transition: width 2s, height 2s, -webkit-transform 2s;

-o-transition: width 2s, height 2s,-o-transform 2s;}

CSS3 动画-animation/@keyframes

1. transition实现动画时只能指定要改变的属性的开始值和结束值，然后在这两个值之间进行平滑过渡的方式来实现动画效果，不能实现比较复杂的动画效果；而animation通过定义多个关键帧以及定义每个关键帧中元素的属性值来实现更为复杂的动画效果。
2. @keyframes 规则用于创建动画。在 @keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新定义样式的动画效果。

语法:@keyframe 关键帧集合的名称{创建关键帧的代码}

1. animation功能的使用方法

语法:animation:动画名称 动画时间 速度曲线 延迟时间

执行次数 执行方向

动画名称——关键帧集合的名称

动画时间——①默认值(0) ②3s ③2000ms

速度曲线——①默认值(ease)(慢-快-慢) ②linear(匀速)

③ease-in(慢-块) ④ease-out(快-慢)

延迟时间——①默认值(0) ②3s ③2000ms ④-2s(立即执行，跳过2秒)

执行次数——①默认值(1) ②5 ③infinite(无限次播放)

执行方向——①默认值(normal)(每次动画执行完毕后返回初始状态)

②alternate(正反交替轮流播放)

③reverse(反向执行动画)

④alternate-reverse(反向开始交替轮流播放)

注释: 动画名称与动画时间不可省略，其余都可省略。

1. 实例1

#tc{animation:zh 4s;}

@keyframes zh{

0%{height: 150px;width: 150px;background-color:pink;}

25%{height: 200px;width: 200px;background-color:green;}

50%{height: 100px;width: 100px;background-color:yellow;}

70%{height: 400px;width: 400px;background-color:blue;}

100%{height: 500px;width: 500px;background-color:pink;}

}

注释:为了得到最佳的浏览器支持，应始终定义 0% 和 100% 选择器。

1. 暂停动画

animation-play-state: paused|running;

plused ——定义动画暂停

running——定义动画播放

举例:

tc.onmouseover=function(){this.style.animationPlayState='paused';};

tc.onmouseout=function(){this.style.animationPlayState='running';}

1. 动画起始终止状态

animation-fill-mode : none | forwards | backwards | both;

none ——默认值(以初始状态作为起始终止状态)

forwards ——以100%时的属性值作为动画终止状态

backwards ——以0%时的属性值作为延迟时间内的状态

both ——forwards与backwards模式都将生效

注释:animation-fill-mode属性必须定义在animation属性之后才会生效。

总结

1. transform

CSS3中的变形处理，实现文字和图像的移动、旋转、伸缩、翻转等。

1. transition

CSS3中的动画功能，通过一个属性值平滑过渡到另一个属性值来实现。

1. animation

CSS3中的动画功能，通过在样式中创建多个关键帧，在这些关键帧中编写样式，并且能够在页面上综合运行这些关键帧来实现较为复杂的动画。

1. 页面性能:重绘与回流

重绘: 操作影响了页面的可见性，但是没有影响布局，浏览器发生重绘。

改变透明度、背景颜色、字体颜色、改变同等尺寸img会发生重绘。

注释:重绘——布局不变、重新绘制。

回流: 浏览器重新计算所有元素的尺寸和位置。

CSS样式没有放在head头部、删减Dom节点、改变Dom节点尺寸等会引起回流。

注释:回流——布局改变、重新绘制。

为避免回流，使用CSS3执行动画的元素应绝对定位(脱离文档流)。

## Git

微博密码: nimengwei1990

新浪云账号:254060001@qq.com 密码:299\*\*\*\*\*\*

微信公众平台账号:13567153653@163.com 密码: nmw299\*\*\*\*\*\*

新浪云操作:

(1)登录新浪云—控制台—云应用SAE—创建新应用(PHP5.6)...—进入应用—应用—代码管理—Git代码部署—获取仓库地址(https://git. sinacloud.com/nmwei)

微信公众平台操作:

(2)进入微信公众平台—公众号设置—功能设置—JS接口安全域名nmwei.applinzi.com

(3)进入微信公众平台—基本配置—成为开发者—开发者ID—获取AppID(应用ID)和AppSelect(应用秘钥)

本地准备工作:

(4)修改index.php文件，将AppID和AppSelect覆盖$jssdk = new jssdk("wx4b1e51e9e80932ee", "a509eea97c87a7a4318c452db6b22573");的两个参数

(5)将index.html中代码复制到index.php文件中，将index.php文件中相关html代码注释，修改index.html文件名(index.php成为web主页)

(6)创建新文件夹—将所有PHP文件和本地web文件拷入该文件夹

git操作指令:

**创建版本库：**

1. 创建本地文件夹(工作区) mkdir 文件夹名称
2. 显示当前本地文件(工作区)目录 pwd
3. 显示下一级本地文件(工作区)目录 ls
4. 初始化本地版本库 git init

备注:本地分支属于本地版本库(即本地仓库)，远程分支属于远程版本库(即远程仓库)。

**修改和提交：**

1. 已修改工作区文件同步到暂存区 git add readme.txt
2. **所有已修改工作区文件同步****到暂存区 git add .**
3. 暂存区已修改文件同步到本地分支 git commit -m "注释"
4. 工作区已修改文件内容到本地分支 git commit -am "注释"
5. 查看文件版本修改提交状态 git status

注释:红色为已修改工作区内容，绿色为已修改暂存区内容，不出现已提交到本地分支的文件。

1. 查看变更内容 git diff

注释: (查看变更)→跟踪文件(git add .)→提交已更新文件(git commit -m "注释")

**查看提交历史:**

1. 查看向版本库(本地分支)提交历史 git log
2. 撤销未提交文件的修改内容 git reset --hard

**分支操作：**

(1)建立新的本地分支 git branch yupeng

(2)查看所有本地分支 git branch

(3)切换到指定本地分支 git checkout yupeng

注释: 如果修改了工作区，commit操作之后，才可以切换到指定分支。

**注意: 切换到指定分支之后，本地文件夹中的内容会相应改变。**

(4)合并指定本地分支到当前本地分支 git merge origin/yupeng

(5)删除指定本地分支 git branch -D 1(本地分支名)

**远程操作:**

1. 克隆远程版本库: git clone 远程仓库地址

注释:只允许向无子文件夹的本地文件夹克隆远程版本库。

1. 查看所有本地分支与远程分支 git branch -a
2. **推送本地分支到远程分支： git push origin 1(分支名)**

注释:路径为index.php所在目录。**更新版本后应首先清理浏览器缓存**。

**注意:第一次上传本地主干上的内容，则命令为git push origin master:1**

**如果本地有主干，则从主干向远程分支上传文件。**

**(4)删除指定远程分支 git push origin --delete 2**

注释:不可以删除已设置为默认版本的远程分支。

前端模板

1. script标签添加类型type="text/html"，将script标签内所有JavaScript代码使用<% %>包裹。
2. 赋值 <%=字段名称%>

图片异步加载

1. 设置img的data-src=图片的地址，而真实的地址src设为空。
2. 设置img 的class=lazy-img
3. lazyload.init()

## 继承

1. 以空函数作为桥梁实现继承

例子1:

<script type="text/javascript">

function Person(age){

this.age = age;

};

Person.prototype = {

constructor:Person,

sayage:function(){

alert(this.age);

}

};

function Student(age,name){

Person.call(this,age);

this.name = name;

};

function inherit(proto){

function f(){};

f.prototype =proto;

return new f();

};

Student.prototype = inherit(Person.prototype);

Student.prototype.constructor=Student;

Student.prototype.sayname = function(){

alert(this.name);

};

var p=new Person('18','jage');

var s=new Student('20','Jake');

</script>

## AngularJS

地址:http://www.runoob.com/angularjs/angularjs-tutorial.html

AngularJs简介

1. AngularJS是一个JavaScript 框架，通过新的属性和表达式扩展了 HTML。
2. AngularJS 是比较新的技术，**Google** 的员工 Miško Hevery 从 2009 年开始着手开发,1.0版本是在 2012 年发布的。该项目目前已由 Google 正式支持，有一个全职的开发团队继续开发和维护这个库。
3. AngularJS通过指令扩展了 HTML，且通过表达式绑定数据到 HTML。

ng-app指令定义一个AngularJS应用程序。

ng-model指令把元素值(比如输入域的值)绑定到应用程序。

ng-blind指令把应用程序数据绑定到HTML视图。

举例: 第一个AngularJs程序

<div ng-app=''>

<p>姓名：<input type="text" ng-model="name"></p>

<h1>Hello{{name}}</h1>

//第二种写法<h1 ng-bind="name"></h1>

</div>

ng-app指令定义<div>元素是AngularJS应用程序的所有者。

ng-model指令把输入域的值绑定到应用程序变量name。

ng-bind指令把应用程序变量绑定到某个段落的innerHTML。

一个网页可以包含多个运行在不同元素中的 AngularJS 应用程序。

注释: (1)ng-app 定义应用程序的执行环境。ng-model将输入域的value值与应用程序中的变量绑定。ng-bind将应用程序中的变量值向DOM元素的innerHTML赋值。(ng-model为DOM元素与应用程序变量绑定的过程，ng-bind是应用程序变量向DOM元素赋值的过程。)

(2){{name}}表达式与ng-blind="name"的对比，前者是对innerHTML插入片段的过程，后者是对innerHTML重新赋值的过程。

1. AngularJS的功能: 单页应用、元素赋值、克隆复制元素、隐藏显示元素、元素添加代码、输入验证等。
2. AngularJS指令

AngularJS指令是**以ng作为前缀的HTML属性**。

ng-init指令初始化AngularJS应用程序变量。

<div ng-app="" ng-init="arr=[1,'aaa',43]">

<h1>{{arr[1]}}</h1>

</div>

注释:AngularJS指令是以ng为前缀的DOM元素的属性。使用ng-init初始化数据的情况并不常见。

1. AngularJS 表达式

AngularJS 表达式书写在双大括号内，与ng-bind 指令类似，把数据与DOM元素的innerHTML绑定。AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 JavaScript 表达式：它们可以包含文字、运算符和**变量**。

实例: <div ng-app="" ng-init="arr=['头']">

<input type="text" ng-model='name'>

<h1>{{arr[0]+name+'尾'}}</h1>

</div>

1. AngularJS 应用

AngularJS 模块(Module)定义了 AngularJS 应用。

AngularJS 控制器(Controller) 用于控制 AngularJS 应用。

ng-app指令定义了应用程序, ng-controller 定义了控制器。

实例: <div ng-app="myApp" ng-controller="myCtrl">

//ng-controller控制器可以代替ng-init

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{firstName + " " + lastName}}

</div>

<script>

//定义应用

var app = angular.module('myApp', []);

//定义控制器

app.controller('myCtrl', function($scope) {

$scope.firstName= "John";

$scope.lastName= "Doe";

});

</script>

AngularJs表达式

1. AngularJS 使用表达式把数据绑定到 HTML。
2. 实例: <div ng-app='' ng-init='num=1;price=4'>

<h1>总价为:{{num\*price}}</h1>

<h1 ng-bind='num\*price'></h1>

</div>

注释: ①使用ng-init进行初始化数据的情况并不常见。②使用ng-init可以一次性初始化多个数据。③ng-init初始化的数据类型有number、string、object、array等。

1. AngularJS 表达式 与 JavaScript 表达式的对比

①类似于JavaScript表达式，AngularJS表达式可以包含字母，操作符，变量。②与JavaScript 表达式不同，AngularJS 表达式可以写在 HTML 中。③与 JavaScript 表达式不同，AngularJS 表达式不支持条件判断，循环及异常。④与 JavaScript 表达式不同，AngularJS 表达式支持过滤器。

AngularJs指令

1. AngularJS 通过被称为指令的新属性来扩展 HTML。AngularJS 通过内置的指令来为应用添加功能。AngularJS 允许你自定义指令。
2. AngularJS **指令是扩展的HTML属性**，带有前缀 ng-。

ng-app 指令初始化一个 AngularJS 应用程序。

ng-init 指令初始化应用程序数据。

ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

1. 实例: <div ng-app ng-init='name="jake"'>

<input type="text" ng-model='name'>

<h1>{{name}}</h1>

</div>

1. {{}}数据绑定

上面实例中的 {{name}} 表达式是一个 AngularJS 数据绑定表达式。

数据绑定同步了 AngularJS 表达式与 AngularJS 数据。

{{name}} 是通过 ng-model="name" 进行同步。

实例:<div ng-app='' ng-init='quantity=1;price=5'>

数量:<input type="number" ng-model='quantity'>

价格:<input type="number" ng-model='price'>

<p>总价{{quantity\*price}}</p>

</div>

注释: 一般ng-init定义变量的默认值，ng-model定义变量的动态值，{{}}表达式进行变量赋值。

1. ng-repeat重复HTML元素

ng-repeat 指令对于集合中（数组中）的每个项会克隆一次 HTML 元素。

实例1:<div ng-app="" ng-init="arr=[1,2,3]">

<li ng-repeat='item in arr'>

{{item}}

</li>

</div>

实例2: <div ng-app="" ng-init="names=[

{'name':'jake','age':20},

{'name':'json','age':18},

{'name':'kai','age':22}

]">

<li ng-repeat='item in names'>

{{item.name+';'+item['age']}}

</li>

</div>

注释:①遍历数组与复制元素共同进行，并通过表达式一一对应进行赋值。②ng-init中的变量名(names)不加引号。

1. ng-app 指令

ng-app 指令定义了 AngularJS 应用程序的根元素。

ng-app 指令在网页加载完毕时会自动引导（自动初始化）应用程序。

ng-app可以通过一个值（比如 ng-app="myModule"）连接到代码模块。

1. ng-init 指令

ng-init 指令为 AngularJS 应用程序定义了初始值。

通常情况下，不使用 ng-init。您将使用一个控制器或模块来代替它。

1. ng-model 指令

ng-model 指令绑定HTML元素到应用程序数据。

ng-model 指令也可以:

为应用程序数据提供类型验证（number、email、required）。

为应用程序数据提供状态（invalid、dirty、touched、error）。

为HTML元素提供CSS 类。

绑定HTML元素到 HTML 表单。

AngularJS ng-model模型

1. ng-model 指令用于绑定应用程序数据到HTML控制器(input, select, textarea)的值。
2. ng-model 指令

ng-model 指令可以将输入域的值与 AngularJS 创建的变量绑定。

实例: <div ng-app='myApp' ng-controller='myCtrl'>

名字:<input ng-model='name'>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl',function($scope){

$scope.name='jake'

});

</script>

注释: function($scope){}应写成["$scope",function($scope){}]

1. 双向绑定

在修改输入域的值时， AngularJS 属性的值也将修改。

1. 验证用户输入

实例: <form ng-app="" name="myForm">

Email:

<input type="email" name="myAddress" ng-model="text">

<span ng-show="myForm.myAddress.$error.email">不是一个合法的邮箱地址</span>

</form>

注释: 以上实例中，提示信息会在 ng-show 属性返回 true 的情况下显示。

1. 应用状态

ng-model 指令可以为应用数据提供状态值(invalid, dirty, touched, error):

实例:

<form ng-app="" name="myForm" ng-init="myText =

'test@runoob.com'">

Email:

<input type="email" name="myAddress"

ng-model="myText" required>

<h1>状态</h1>

{{myForm.myAddress.$valid}}

{{myForm.myAddress.$dirty}}

{{myForm.myAddress.$touched}}

</form>

1. CSS 类

<style>

input.ng-invalid {

background-color: lightblue;

}

</style>

<body>

<form ng-app="" name="myForm">

输入你的名字:

<input name="myAddress" ng-model="text" required>

</form>

AngularJS Scope(作用域)

1. Scope(作用域) 是应用在 HTML (视图) 和 JavaScript (控制器)之间的纽带。Scope 是一个对象，有可用的方法和属性。Scope 可应用在视图和控制器上。

<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{carname}}</h1>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.carname = "Volvo";

});

</script>

注释:①当在控制器中添加 $scope 对象时，视图 (HTML) 可以获取了这些属性。视图中，不需要添加 $scope 前缀, 只需要添加属性名即可，如： {{carname}}。②作用域通过属性的方式存储变量。

1. Scope 概述

AngularJS 应用组成如下：

**View(视图), 即 HTML。**

**Model(模型), 当前视图中可用的数据。**

**Controller(控制器), 即 JavaScript 函数，可以添加或修改属性。**

scope 是模型。

scope 是一个 JavaScript 对象，带有属性和方法，这些属性和方法可以在视图和控制器中使用。

1. 如果你修改了视图，模型和控制器也会相应更新：

实例: <div ng-app="myApp" ng-controller="myCtrl">

<input ng-model="name">

<h1>我的名字是 {{name}}</h1>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.name = "John Dow";

});

</script>

1. Scope 作用范围

当我们使用 ng-repeat 指令时，每个重复项都访问了当前的重复对象。

实例: <div ng-app="myApp" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in names">{{x}}</li>

</ul>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.names = ["Emil", "Tobias", "Linus"];

});

</script>

每个 <li> 元素可以访问当前的重复对象，这里对应的是一个字符串， 并使用变量 x 表示。

1. 根作用域

所有的应用都有一个 $rootScope，它可以作用在 ng-app 指令包含的所有 HTML元素中。

$rootScope 可作用于整个应用中。是各个 controller 中 scope 的桥梁。用 rootscope 定义的值，可以在各个 controller 中使用。

实例: 创建控制器时，将 $rootScope 作为参数传递，可在应用中使用：

<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{lastname}} 家族成员:</h1>

<ul>

<li ng-repeat="x in names">{{x}} {{lastname}}</li>

</ul>

</div>

<script>

var app = angular.module('myApp', []);

//注释:每一个应用程序是一个angular模块(module)

app.controller('myCtrl',function($scope,$rootScope) {$scope.names = ["Emil", "Tobias", "Linus"];

$rootScope.lastname = "Refsnes";

});

</script>

实例: <div ng-app='myApp'>

<div ng-controller='myCtrl1'>

<p ng-repeat='x in arr1'>

{{x}}{{age1}}{{age2}}

</p>

</div>

<div ng-controller='myCtrl2'>

<p ng-repeat='x in arr1'>

{{x}}{{age2}}

</p>

<p ng-repeat='x in arr2'>

{{x}}

</p>

</div>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl1',function($scope,$rootScope){

$scope.age1=18;

$rootScope.arr1=[1,2,3];

})

app.controller('myCtrl2',function($scope,$rootScope){

$scope.arr2=[4,5,6];

$rootScope.age2=20;

})

</script>  
注释: ①$scope定义局部作用域变量，只能在当前控制器对应的视图中使用。$rootScope定义全局作用域变量，可以在应用程序内部所有视图中使用。②一个html页面只可以定义一个应用程序，但该应用程序可以对应多个控制器。

AngularJS 控制器

1. AngularJS 控制器可以控制 AngularJS 应用程序的数据。AngularJS 控制器是常规的 JavaScript 对象。AngularJS 应用程序被控制器控制。ng-controller 指令定义了应用程序控制器。控制器是 JavaScript 对象，由标准的 JavaScript 对象的构造函数创建。

注意: ng-app定义了应用程序;

view(视图), 即 HTML;

ng-model定义模型(当前视图中的可用数据);

ng-controller 定义控制器;

应用程序由视图、模型和控制器组成。

1. 实例:

<div ng-app="myApp" ng-controller="myCtrl">

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{firstName + " " + lastName}}

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.firstName = "John";

$scope.lastName = "Doe";

});

</script>

应用解析：

AngularJS 应用程序由 ng-app 定义。应用程序在 <div> 内运行。

ng-controller="myCtrl" 属性是一个 AngularJS 指令。用于定义一个控制器。

myCtrl 函数是一个 JavaScript 函数。

AngularJS 使用$scope 对象来调用控制器。

在 AngularJS 中， $scope 是一个应用象(属于应用变量和函数)。

控制器的 $scope （相当于作用域、控制范围）用来保存AngularJS Model(模型)的对象。

控制器在作用域中创建了两个属性 (firstName 和 lastName)。

ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。

1. 控制器方法

上面的实例演示了一个带有 lastName 和 firstName 这两个属性的控制器对象。控制器也可以有方法（变量和函数）：

实例: <div ng-app='myApp' ng-controller='myCtrl'>

<h3>{{fullName()}}</h3>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl',function($scope,$rootScope){

$scope.name='Jake';

$scope.age=18;

$scope.fullName=function(){

return $scope.name+' '+$scope.age;

}

})

</script>

注释: 执行控制器的方法时，小括号不可以省略。

1. 外部文件中的控制器

在大型的应用程序中，通常是把控制器存储在外部文件中。只需要把 <script> 标签中的代码复制到外部js文件中即可。

实例: <div ng-app='myApp' ng-controller='myCtrl'>

<h3>{{fullName()}}</h3>

</div>

<script src="1.js"></script>

angular.module('myApp',[]).controller('myCtrl',function($scope,$rootScope){

$scope.name='Jake';

$scope.age=18;

$scope.fullName=function(){

return $scope.name+' '+$scope.age;

}

})

注释: 应用程序与控制器的链式写法。

AngularJS 过滤器

1. 过滤器可以使用一个管道字符（|）添加到表达式和指令中。AngularJS 过滤器可用于转换数据。
2. currency 格式化数字为货币格式。 (转化数据)
3. filter 从数组项中选择一个子集。 (筛选数据)
4. lowercase 格式化字符串为小写。 (转化数据)
5. uppercase 格式化字符串为大写。 (转化数据)
6. orderBy 根据某个表达式排列数组。 (排列数据)

注释: 过滤器可以起到转化数据、排列数据和筛选数据的效果。其中，转化数据过滤器书写在表达式中，筛选数据和排列数据书写在ng-repeat指令中。

1. **表达式中**添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到表达式中。

1. 举例: {{ lastName | uppercase }}

{{ lastName | lowercase }}

{{ (quantity \* price) | currency }}

1. 指令中添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中。

实例:<div ng-app='' ng-init="names=[{'name':'jake','age':20}, {'name':'ki','age':18},{'name':'Gedon','age':22}]">

<li ng-repeat='x in names|orderBy:"name"'>

{{x.name+","+x.age}}

</li>

</div>

注释: ①ng-init中的names不可以用引号包裹。②orderBy中的name必须用引号包裹。③以数组中每一项(对象)的某个属性名进行排序。

1. 过滤输入

输入过滤器可以通过一个管道字符(|)和一个过滤器添加到指令中，该过滤器后跟一个冒号和一个模型名称。

filter 过滤器从数组中选择一个子集。

实例: <div ng-app='' ng-init="names=[{

'name':'a','age':16

},{

'name':'b','age':22

},{

'name':'c','age':24

},{

'name':'d','age':18

}]">

<input type="text" ng-model='test'>

<ul>

<li ng-repeat='x in names|orderBy:"age"|filter: test'>

{{x.name+','+x.age|uppercase}}

</li>

</ul>

</div>

注释: ①ng-init中的变量名names不加引号。②过滤器中的对象属性名加引号。③currency、lowercase、uppercase三个过滤器在表达式中添加。③orderBy过滤器以及filter在ng-repeat指令中添加，且以键值对的形式存在。④orderBy过滤器的值为数组每一项元素(对象)的属性名，filter过滤器的值为模型(ng-model)的属性值。

AngularJS 服务(Service)

1. 在 AngularJS 中，服务是一个函数或对象，可在你的 AngularJS 应用中使用。AngularJS 内建了30 多个服务。AngularJS 中可以创建自己的服务，或使用内建服务。
2. 实例: $location 服务可以返回当前页面的 URL 地址。

<div ng-app='myApp' ng-controller='myCtrl'>

<h2>{{myUrl}}</h2>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl',function($scope,$location){

$scope.myUrl= $location.absUrl();

})

</script>

注释: $location 服务是作为一个参数传递到 controller 中。如果要使用它，需要在 controller 中定义。

1. **AngularJS $http 是一个用于读取web服务器上数据的服务。**

$http.get(url) 是用于读取服务器数据的函数。

<div ng-app="myApp" ng-controller="customersCtrl">

<ul>

<li ng-repeat="x in names">

{{ x.Name + ', ' + x.Country }}

</li>

</ul>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('customersCtrl', function($scope, $http) { $http.get("http://www.runoob.com/try/angularjs/data/Customers\_JSON.php")

.success(function (response) {$scope.names = response.records;});

});

</script>

AngularJS HTML DOM

1. ng-disabled 指令绑定应用程序数据 "mySwitch" 到 HTML 的 disabled 属性。

ng-model 指令绑定 "mySwitch" 到 HTML input checkbox 元素的内容（value）。

ng-show 指令隐藏或显示一个 HTML 元素。

1. 实例:

<div ng-app='' ng-init='mySwitch=true'>

<input type="checkbox" ng-model='mySwitch'>

<h1>{{mySwitch}}</h1>

<input type="text" ng-disabled='mySwitch'>

<p ng-show='true'>可见</p>

<p ng-show='3<2'>不可见</p>

<!-- <p ng-show='mySwitch'>是否可见</p> -->

<p ng-hide='mySwitch'>是否可见</p>

</div>

注释:在一个控制器中ng-show与ng-hide不可以同时出现。

AngularJS 事件

1. AngularJS 有自己的 HTML 事件指令。ng-click 指令定义了 AngularJS 点击事件。  
    <div ng-app="" ng-init="count=0">

<button ng-click="count = count + 1">点我！</button>

<p>{{ count }}</p>

</div>

注释:两种动态改变应用程序中变量值的指令：①ng-model；②ng-click。

1. 实现点击隐藏与显示的效果

<div ng-app='myApp' ng-controller='myCtrl'>

<button ng-click="change()">点击</button>

<p ng-show='bool'>隐藏与显示</p>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl',function($scope){

$scope.bool=true;

$scope.change=function(){

$scope.bool=!$scope.bool;

}

})

</script>

AngularJS 模块

1. 模块定义了一个应用程序。模块是应用控制器的容器。控制器属于一个模块。
2. 创建模块

<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>

1. 添加控制器

可以使用 ng-controller 指令来添加应用的控制器:

1. 添加指令

AngularJS 提供了很多内置的指令，你可以使用它们来为你的应用添加功能。http://www.runoob.com/angularjs/angularjs-reference.html

指令以ng为前缀，作为HTML元素的属性而存在。

基本指令

ng-app 定义应用程序的根元素。

ng-init 定义应用的初始化值

ng-controller 定义应用的控制器对象。

ng-repeat 定义集合中每项数据的模板

定义HTML属性方面的指令

ng-checked 规定元素是否被选中

ng-href 为 the <a> 元素指定链接

ng-class 指定 HTML 元素使用的 CSS 类

ng-disabled 规定一个元素是否被禁用

ng-model 绑定 HTML 控制器的值到应用数据

ng-selected 指定元素的 selected 属性

ng-src 指定 <img> 元素的 src 属性

ng-value 规定 input 元素的值

定义CSS样式方面的指令

ng-style 指定元素的 style 属性

定义事件方面的指令

ng-click 定义元素被点击时的行为

ng-dblclick 规定双击事件的行为

ng-mousedown 规定按下鼠标按键时的行为

ng-mouseenter 规定鼠标指针穿过元素时的行为

ng-mouseleave 规定鼠标指针离开元素时的行为

ng-mousemove 规定鼠标指针在指定的元素中移动时的行为

ng-mouseup 规定当在元素上松开鼠标按钮时的行为

ng-keydown 规定按下按键事件的行为

ng-keypress 规定按下按键事件的行为

ng-keyup 规定松开按键事件的行为

ng-blur 规定 blur 事件的行为

ng-change 规定在内容改变时要执行的表达式

ng-focus 规定聚焦事件的行为

ng-submit 规定 onsubmit 事件发生时执行的表达式

操作DOM元素的指令

ng-hide 隐藏或显示 HTML 元素

ng-show 显示或隐藏 HTML 元素

AngularJS API

1. AngularJS 全局 API
2. 实例

<div ng-app='myApp' ng-controller='myCtrl'>

<h1>{{x1}}</h1>

<h1>{{x2}}</h1>

<h1>{{x3}}</h1>

<h1>{{x4}}</h1>

</div>

<script>

var app=angular.module('myApp',[]);

app.controller('myCtrl',function($scope){

$scope.x1='Json';

$scope.x2=angular.lowercase($scope.x1);

$scope.x3=angular.uppercase($scope.x1);

$scope.x3=angular.isNumber($scope.x1);

$scope.x4=angular.isString($scope.x1);

})

</script>

AngularJS 路由

1. AngularJS 路由允许我们通过不同的 URL 访问不同的内容。通过 AngularJS 可以实现多视图的单页Web应用（single page web application，SPA）。
2. 实例

<html>

<head>

<meta charset="utf-8">

<title>AngularJS 路由实例 - 菜鸟教程</title>

</head>

<body ng-app='routingDemoApp'>

<h2>AngularJS 路由应用</h2>

<ul>

<li><a href="#/">首页</a></li>

<li><a href="#/computers">电脑</a></li>

<li><a href="#/printers">打印机</a></li>

<li><a href="#/blabla">其他</a></li>

</ul>

<div ng-view></div>

<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>

<script src="http://apps.bdimg.com/libs/angular-route/1.3.13/angular-route.js"></script>

<script>

var app=angular.module('routingDemoApp',['ngRoute'])

app.config(['$routeProvider', function($routeProvider){

$routeProvider

.when('/',{template:'这是首页页面'})

.when('/computers', {

templateUrl: '3.html',})

.when('/printers',{template:'这是打印机页面'})

.otherwise({redirectTo:'/'});

}]);

</script>

</body>

</html>

## Bootstrap

Bootstrap 简介

1. Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是由 Twitter 的 Mark Otto 和 Jacob Thornton 开发的。Bootstrap 是 2011 年8月在 GitHub 上发布的开源产品。

注释: Bootstrap依赖于jQuery。

1. Bootstrap 包的内容: Bootstrap 基本结构、Bootstrap CSS、Bootstrap 布局组件和 Bootstrap 插件几个部分。
2. 基本结构: 带有网格系统、链接样式、背景的基本结构
3. CSS: 全局的CSS设置、基本的HTML元素样式、可扩展的class、先进的网格系统。
4. 组件: 可用于创建图像、下拉菜单、导航、警告框、弹出框等、
5. 插件:包含十几个自定义的jQuery插件

3. HTML5 文档类型

Bootstrap 使用到的某些 HTML 元素和 CSS 属性需要将页面设置为 HTML5 文档类型。

CSS排版包括:标题、页面主体、内联文本元素、对齐、改变大小写、缩略语、地址、引用、列表。

4. 实例

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

<link rel="stylesheet" href="bootstrap.min.css">

</head>

<body>

<div class="container">

<h1>育知同创</h1>

<!-- 可以使用small包裹起到副标题的效果 -->

<h3>育知同创<small>欢迎你的到来</small></h3>

<!-- mark标签添加背景效果 -->

<p>hello Word：<mark>育知同创</mark></p>

<!-- lead类使元素突出显示 -->

<p class="lead">hello</p>

<!-- 元素左/右/中对齐 -->

<p class="text-left">你好吗？</p>

<p class="text-right">你好吗？</p>

<p class="text-center">你好吗？</p>

<!-- 元素大小写 -->

<p class="text-lowercase">aAa</p>

<p class="text-uppercase">aAa</p>

<p class="text-capitalize">aAa</p>

<!-- 定义无样式的列表 -->

<ul class="list-unstyled">

<li>1</li>

<li>2</li>

</ul>

</div>

</body>

注释: ①引用bootstrap后，许多元素已经被默认修饰。②bootstrap将全局的字体大小设置为14px，行高为1.428。

Bootstrap CSS栅格、代码和表格

1. Bootstrap栅格系统必须依赖于容器(Containers)而存在。

Bootstrap提供一套响应式移动设备优先的流式栅格系统。随着屏幕视口尺寸而增加。

1. 实例1:

<div class="container">

//栅格系统等分12列，移动端class类名前缀为.col-xs-。

<div class="row">

<div class="col-xs-3">3</div>

<div class="col-xs-6">6</div>

<div class="col-xs-3">3</div>

</div>

//栅格系统偏移

<div class="row">

<div class="col-xs-4 col-xs-offset-1">4</div>

</div>

//栅格系统嵌套

<div class="row">

<div class="col-xs-5">

out

<div class="row">

<div class="col-xs-8">

fitst

</div>

<div class="col-xs-4">

two

</div>

</div>

</div>

</div>

//栅格系统排序

<div class="row">

<div class="col-xs-3 col-xs-push-9">3</div>

<div class="col-xs-9 col-xs-pull-3">9</div>

</div>

</div>

1. CSS代码

类语言代码、用户输入、代码块、变量、程序输出。

实例2: <div class="container">

<!-- 内联代码 -->

<code>&lt;section&gt;</code>

<kbd>cmd</kbd>

<!-- 代码段 -->

<pre>代码段</pre>

</div>

1. CSS表格

实例: <!-- table-responsive响应式表格:可能出现横向滚动条 -->

<div class="container table-responsive">

<!-- table:修饰后默认表格，table-striped:斑马线表格，table-bordered:带边框表格，table-hover:带鼠标悬停效果表格，table-condensed:紧凑型表格 -->

<table class="table table-striped table-bordered table-hover table-condensed">

<thead>

<tr class="active">

<th>表格标题</th>

<th>表格标题</th>

<th>表格标题</th>

</tr>

</thead>

<tbody>

<!--success/info/warning/danger定义不同颜色-->

<tr class="success">

<td>表格单元格</td>

<td>表格单元格</td>

<td>表格单元格</td>

</tr>

<tr class="info">

<td>表格单元格</td>

<td>表格单元格</td>

<td>表格单元格</td>

</tr>

<tr class="warning">

<td>表格单元格</td>

<td>表格单元格</td>

<td>表格单元格</td>

</tr>

<tr class="danger">

<td>表格单元格</td>

<td>表格单元格</td>

<td>表格单元格</td>

</tr>

</tbody>

</table>

</div>

Bootstrap CSS 表单、按钮和图片

1. 按钮

实例:

<div class="container">

<!-- 设置颜色 -->

<button class="btn-default">default</button>

<button class="btn-primary">primary</button>

<button class="btn-success">success</button>

<button class="btn-info">info</button>

<button class="btn-warning">warning</button>

<button class="btn-link">link</button>

<button class="btn-danger">dange</button>

</div>

<div class="div">

<!-- 设置大小 -->

<button class="btn-info btn-lg">info</button>

<button class="btn-info">info</button>

<button class="btn-info btn-sm">info</button>

<button class="btn-info btn-xs">info</button>

</div>

<!-- 设置块级 -->

<button class="btn-default btn-block">hello</button>

<a href="#" role='button' class="btn-default btn-lg active ">a</a>

<input class="btn-default" type="button" value="input">

## 移动端性能优化

1. 加载优化、执行脚本优化、渲染优化、CSS优化。
2. 加载优化:减少http请求数、压缩图片、异步加载、按需加载(滚动加载、点击加载)。
3. 执行脚本优化: 避免iframe、img等src为空、避免重设图片大小、点击事件优化、注意scroll resize事件绑定时机。
4. 渲染优化: 使用CSS3动画、使用requestAnimationFrame替代setTimeout、webp优于jpg、png8优于gif、对图片使用lazyload、GPU加速(使用transition触发、transform触发)。
5. CSS优化: 层次不超过3、合并CSS规则、去掉0的单位、不要申明过多的font-size。

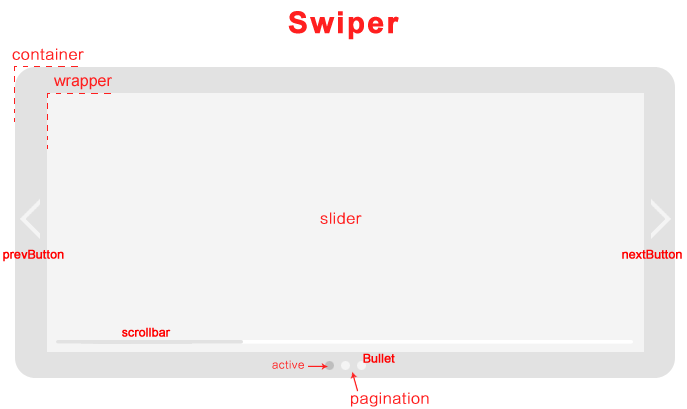
工作

1. 面试、入职、团队、定位、流程。
2. 面试: 缺人才会招人。
3. 入职: 刚入职多学多问，尽快熟悉。
4. 团队: 表现积极，充分沟通，尽快融入。
5. 流程:调研、调研报告、立项、设计师出效果图、研发、测试、上线、升级。
6. 项目组角色：负责人、RD后端、FE前端、PM产品、UE设计师、QA测试。
7. 研发: 制定一个计划、有可检查的成果、时间节点。

兼容性问题

1. CSS兼容性问题
2. PC页面布局兼容
3. JS兼容问题
4. 移动端页面兼容问题

## Swiper

1. Swiper基本结构  
   
2. Swiper初始化

new Swiper(swiperContainer, parameters)

用于初始化一个Swiper，返回初始化后的Swiper实例。

swiperContainer : 必选，HTML元素或者string类型，Swiper容器的css选择器，例如“.swiper-container”。

parameters : 可选，参见配置选项。

实例: var mySwiper = new Swiper('.swiper-container')

1. Basic(Swiper一般选项)

(1)initialSlide——初始索引

设定初始化时slide的索引。

类型：number 默认：0 举例： 2

(2)direction——滑动方向

Slides的滑动方向，可设置水平(horizontal)或垂直(vertical)

类型：string 默认horizontal 举例： vertical

(3)speed——滑动速度

滑动速度，即slider自动滑动开始到结束的时间（单位ms）。

类型：number默认：300 举例： 1000

(4)autoplay——自动切换

自动切换的时间间隔（单位ms），不设定该参数slide不会自动切换。

类型：number默认：0(不轮播) 举例： 5000

1. autoplayDisableOnInteraction

用户操作swiper之后，是否禁止autoplay。默认为true：停止。

如果设置为false，用户操作swiper之后自动切换不会停止，每次都会重新启动autoplay。

操作包括触碰，拖动，点击pagination等。

类型：boolean默认：true举例： false

注释: autoplayDisableOnInteraction与autoplay共同使用时生效。

1. autoplayStopOnLast

如果设置为true，当切换到最后一个slide时停止自动切换。（loop模式下无效）。

类型：boolean 默认：false 举例： true 启用版本：3.3.0

1. grabCursor

设置为true时，鼠标覆盖Swiper时指针会变成手掌形状，拖动时指针会变成抓手形状。（根据浏览器形状有所不同）

1. autoHeight

自动高度。设置为true时，wrapper和container会随着当前slide的高度而发生变化。

类型：boolean默认：false 举例： true 启用版本：3.2.5

1. nested

用于嵌套相同方向的swiper时，当切换到子swiper时停止父swiper的切换。

请将子swiper的nested设置为true。

需要在slideChangeEnd时判断作用块，因此快速滑动时这个选项无效。

类型：boolean 默认：false 举例：true 启用版本：3.3.0

注意: ①通过后代选择器覆盖swiper.min.css中的样式，实现个性化样式定义。②通过添加类名实现个性化样式定义。

## React

1. MVC，MVP 和 MVVM 的图示

http://www.ruanyifeng.com/blog/2015/02/mvcmvp\_mvvm.html

复杂的软件必须有清晰合理的架构，否则无法开发和维护。MVC(Model-View-Controller)是最常见的软件架构之一，业界有着广泛应用。

1. MVC

MVC模式的意思是，软件可以分成三个部分。

视图(View): 用户界面。

控制器(Controller): 业务逻辑

模型(Model): 数据保存

View 传送指令到 Controller。Controller 完成业务逻辑后，要求 Model 改变状态。Model 将新的数据发送到 View，用户得到反馈。所有通信都是单向的。

1. 互动模式

接受用户指令时，MVC 可以分成两种方式。一种是通过 View 接受指令，传递给 Controller。另一种是直接通过controller接受指令。

1. 实例：Backbone

实际项目往往采用更灵活的方式，以 Backbone.js 为例。

①用户可以向 View 发送指令（DOM 事件），再由 View 直接要求 Model 改变状态。

②用户也可以直接向 Controller 发送指令（改变 URL 触发 hashChange 事件），再由 Controller 发送给 View。

③Controller 非常薄，只起到路由的作用，而 View 非常厚，业务逻辑都部署在 View。所以，Backbone 索性取消了 Controller，只保留一个 Router（路由器） 。

1. MVP
2. React 起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设 Instagram 的网站。 React 的设计思想极其独特，属于革命性创新，性能出众，代码逻辑却非常简单。React 可以在浏览器运行，也可以在服务器运行。
3. HTML 模板

## 简历

1. 简历针对专业技能、项目经验详细介绍。
2. 准备简历→投递简历→初试→复试→试用期→转正→发展
3. 初试一般情况下会有笔试。
4. 复试一般聊简历上的内容。
5. 试用期一般为三个月，薪资80%。
6. 简历:个人信息、求职意向、专业技能、工作经历、项目经验、教育背景。
7. 个人信息: 姓名、性别、年龄、毕业院校、学历专业、联系方式(电话、邮箱)。(专业正常写)
8. 求职意向: HTML5软件工程师 Web前端工程师
9. 专业技能: (1)排版清晰明了，具有层次性(精通、掌握、熟练、了解)。
10. 技术氛围浓厚：基本技术、前言技术、熟悉的其他开发技术等。
11. 专业度：专业术语，做到能英文的不中文。
12. 不要进传统软件公司、外包公司。最好选择PC端与移动端都有业务的互联网公司。
13. 公司50人:Web前端2人。
14. 项目经验要素:

项目名称(不要千篇一律)、开发工具(突出精通语言)、项目周期(延长项目时间)、项目描述(专业性、具体性)、项目职责(提高自己项目中的作用)、项目功能(价值、作用)、技术要点(实现具体功能的具体技术)、项目收获(自主产品)。

1. 项目经验注意事项：
2. 项目经验要跟公司相匹配
3. 尽可能全面的体现你的技术点
4. 上线、有剩余价值
5. 项目技术点最好不要重复
6. 项目周期(开发时间长、上线时长)
7. 项目顺序(主打项目，自己最熟悉了解的)
8. 团队项目+个人独立完成项目
9. 教育背景
10. 按照时间倒序排序，最近的在最先。
11. 教育经历可以放到工作经验之前。
12. 证书作为特别技能的证明在面试中起到加分。
13. 简历投递

智联招聘、前程无忧、大街网、若邻网、国人才热线、聘网、中华英才网

周伯通、拉勾网、BOSS直播、脉脉

1. 简历投递注意事项

广撒网、多渠道

求职职位、地域搜索要明确

网上投递简历名称精确、凸显优势

投递时间:工作日8:30-9:30；13:30-14:30为最佳

## 面试

1. 企业对IT人才素质要求

工作经验、专业技能、沟通能力、稳定性、解决问题的能力、自信心、团队协作、创新能力。

1. 我们具备的素质

综合素质: 稳定性、自学能力、抗压能力、自信等

专业技能: 技术、独立开发等

1. 面试流程

电话邀约面试、初始、复试、offer

注意:拿出简历、保持一致，准备纸笔、方面记录，注意语速语调、模仿对方语速、听不清大胆问，礼貌结束、表示感谢、对方先挂，礼貌问候、良好印象。

1. 物质方面的准备

最多上午面试一家，下午面试两家。

5份简历。

手机里保存自己的项目和该公司的项目。

1. 信息方面的准备

企业性质、背景：全国企业信用信息公示系统

企业上线项目：查询该企业的资料，深入了解该企业的企业文化、特点。

企业招聘信息、岗位要求。

1. 初始的形式

笔试、机试、技术面试

笔试: 细心认真、字迹清楚、收集或总结笔试题目。

机试: 认真细心、灵活处理。

技术面试: 真诚、不卑不亢、询问公司团队及其他情况，询问对方贵姓或索要对方名片或者联系方式，询问什么时候可以出结果。

1. 常见面试题

自我介绍、最擅长的技术方向、谈谈之前做过的项目、你对我们公司了解多少、自己的优点缺点、为什么离开上一家公司、工作中面临的最大问题、短期与长期职业目标、对加班的看法、对薪资的看法、有什么要问我的、经验不够如何胜任这份工作。

1. 面试后的注意事项

面试总结: 技术问题总结、HR总结

跟踪回访: 2天内电话回访询问结果(结果)，注意沟通方式和措辞。

准备下家: 面试总结(技术和HR)，物质材料与信息材料。

1. 接offer注意事项

保底: 找房子推辞入职时间

1. 劳动合同注意事项

薪资谈判: 试用期薪资80%-100%。

试用期: 试用期一般是0-6个月。

1. 建议分享

不要把所有的希望寄托在一家公司。

不要幻想一次面试就能找到工作。

收到挫折、切勿沮丧。

用平等的心态面对面试官。

变被动为主动，回答问题尽量转到自己擅长的技术方面深入回答。

始终保持微笑。

1. 入职材料：身份证、户口本、离职证明、学历证、银行流水、体检报告。
2. 试用期常见的问题：

公司要求独立完成

负责开发人离职，接手二手项目

根据项目需求未接触过的技术

进入公司有技术大牛合作，有对比

工作环境太压抑

项目工期紧，每天加班到深夜

1. 努力通过试用期

学会邮件沟通

勤快积极做事

**面试中HR常见的问题**

1. 请你做一下自我介绍？

150-200字

教育背景、工作经历、岗位内容……

1. 个人优缺点

性格、工作能力、逻辑思维、习惯

注意:不能说性格上的缺点，不能说能力方面的缺点，不能与工作有关。

1. 对于加班的看法
2. 离职原因
3. 工作经历

基本信息:地址在哪，住在哪，有多少人，哪些部门，什么业务，上班时间

工作流程:有哪些部门，怎么合作的，需求文档，上线流程，测试流程

工作细节:遇到的困难如何解决，最有成就感的事情

离职原因:根据公司所在的地点和面试公司情况针对回答

职业规划:为什么选择开发，如何规划职业

薪资谈判:与实际情况相匹配，薪资谈判技巧

1. 工作流程

产品部:原型图、需求文档

美工UI:图片，整体架构

后台数据:接口、数据来源

测试:产品测试bug处理

1. 项目经验

项目大小：开发周期、下载量、上线时间

技术价值：项目难点、技术要点、商业价值

其他：项目如何分工、团队如何合作

1. 你对我们的公司了解多少

北京市企业信息网

公司官网

企业文化、规模、项目情况等

熟悉公司上线APP

1. 薪资谈判

从自身能力和经验以及人脉出发

争取薪资是税后的价位

试用期价位，转正后达到自己的期望价位

有其他福利待遇

1. 还有什么想问的

团队人数，技术培训，学习团队

1. 银行流水: 银行卡已注销