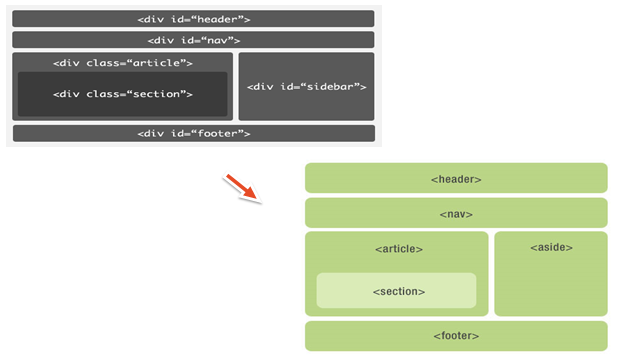
# HTML5 语义元素

## 为什么要语义



## HTML5新增加布局语义元素



新增加的布局语义元素都是块元素，和div类似，只是多了语义

### header元素

header 元素代表“网页”或“section”的页眉。通常包含h1-h6元素或hgroup，作为整个页面或者一个内容块的标题。也可以包裹一节的目录部分，一个搜索框，一个nav，或者任何相关logo。整个页面没有限制header元素的个数，可以拥有多个，可以为每个内容块增加一个header元素

<header>

<h1>网站标题</h1>

<h2>网站副标题</h2>

</header>

注意事项：

1. 可以是“网页”或任意“section”的头部部分；
2. 没有个数限制。
3. **如果hgroup或h1-h6自己就能工作的很好，那就不要用header。**

### footer元素

footer元素代表“网页”或“section”的页脚，通常含有该节的一些基本信息，譬如：作者，相关文档链接，版权资料。如果footer元素包含了整个节，那么它们就代表附录，索引，提拔，许可协议，标签，类别等一些其他类似信息。

<footer>

地址：昌平区平西王府 邮箱：info@atguigu.com 微博：weibo.com

</footer>

注意事项：

1. 可以是“网页”或任意“section”的底部部分；
2. 没有个数限制，除了包裹的内容不一样，其他跟header类似。

### hgroup元素

hgroup元素代表“网页”或“section”的标题，当元素有多个层级时，该元素可以将h1到h6元素放在其内，譬如文章的主标题和副标题的组合

<header>

<hgroup>

<h1>网站标题</h1>

<h2>网站副标题</h2>

</hgroup>

</header>

注意事项：

1. 如果只需要一个h1-h6标签就不用hgroup
2. 如果有连续多个h1-h6标签就用hgroup
3. 如果有连续多个标题和其他文章数据，h1-h6标签就用hgroup包住，和其他文章元数据一起放入header标签。

### nav元素

nav元素代表页面的导航链接区域。用于定义页面的**主要导航部分**。

<nav>

<ul>

<li>HTML 5</li>

<li>CSS3</li>

<li>JavaScript</li>

</ul>

</nav>

### aside元素

aside元素被包含在article元素中作为主要内容的附属信息部分，其中的内容可以是与当前文章有关的相关资料、标签、名词解释等。（特殊的section）

在article元素之外使用作为页面或站点全局的附属信息部分。最典型的是侧边栏，其中的内容可以是日志串连，其他组的导航，甚至广告，这些内容相关的页面。

<article>

<p>文章内容</p>

<aside>

<h1>作者简介</h1>

<p>张三，网络写手。</p>

</aside>

</article>

注意事项：

1. aside在article内表示主要内容的附属信息，
2. **在article之外则可做侧边栏，没有article与之对应，最好不用。**
3. 如果是广告，其他日志链接或者其他分类导航也可以用

### section元素

section元素代表文档中的“节”或“段”，“段”可以是指一篇文章里按照主题的分段；“节”可以是指一个页面里的分组。section通常还带标题，虽然html5中section会自动给标题h1-h6降级，但是最好手动给他们降级

<section>

<h1>section是啥？</h1>

<article>

<h2>关于section</h2>

<p>section的介绍</p>

<section>

<h3>关于其他</h3>

<p>关于其他section的介绍</p>

</section>

</article>

</section>

注意事项：

1. 一张页面可以用section划分为简介、文章条目和联系信息。不过在文章内页，最好用article。**section不是一般意义上的容器元素，如果想作为样式展示和脚本的便利，可以用div。**
2. 表示文档中的节或者段；
3. **article、nav、aside可以理解为特殊的section，所以如果可以用article、nav、aside就不要用section，没实际意义的就用div**

### article元素

article元素最容易跟section和div容易混淆，其实article代表一个在文档，页面或者网站中自成一体的内容，其**目的是为了让开发者独立开发或重用**。譬如论坛的帖子，博客上的文章，一篇用户的评论，一个互动的widget小工具。

<article>

<h1>一篇文章</h1>

<p>文章内容..</p>

<footer>

<p><small>版权：尚硅谷，作者：龙</small></p>

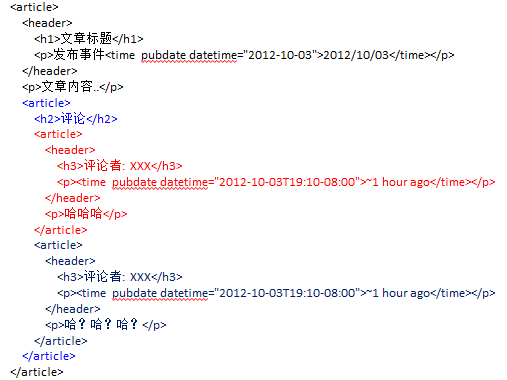
</footer>

</article>

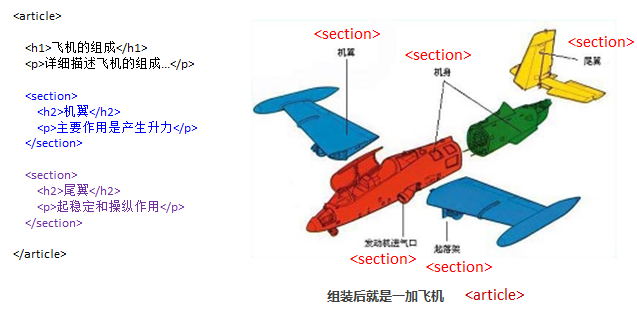
注意事项：

1. 一张页面可以用section划分为简介、文章条目和联系信息。不过在文章内页，最好用article。section不是一般意义上的容器元素，如果想作为样式展示和脚本的便利，可以用div。
2. 表示文档中的节或者段；
3. article、nav、aside可以理解为特殊的section，所以如果可以用article、nav、aside就不要用section，没实际意义的就用div

### article元素嵌套article



### article元素嵌套section



### section 元素嵌套 article



### address元素

* <address> 表示联系信息，无特殊属性
* 使用说明
  1. 要表示联系信息。例如电话，地址，邮箱，网站地址等。如果某一内容不是地址信息，使用<p>元素，而不是<address>元素。。
  2. 通常一个<address>元素可以放页面的<footer>元素中，如果有。

### pre元素

* pre 元素可定义预格式化的文本。被包围在 pre 元素中的文本通常会保留空格和换行符。而文本也会呈现为等宽字体。
* 最佳使用场景：展示程序源代码

### progress元素

* <progress>用来显示一项工作做完成的进度。
  + **元素/DOM属性：max** ~ 大于0的浮点型数字，表示完成一项工作总的“工作量”。
  + **元素/ DOM属性：value**
  + 通常情况 value值为0 – max 的任意浮点型数字，表示已经完成了max指定工作量的多少。
  + 无论是否设定了max的值，如果不指定value的值，进度条是不确定的，这表明一个活动正在进行中，没有迹象显示它需要多长时间。
  + 如果没有设定max的值，value的取值范围0~1浮点任意浮点数，表示完成工作的进度。
  + **DOM属性（只读）： position** , 返回一个double值，即value除以max的结果；如果进度条是一个不确定的进度条（比如没有设置max和value），它返回- 1。

## 什么时候用DIV呢？

使用新语义元素

* 没有语义的情况下，就需要是使用DIV了。
* 注意：千万不要为了语义而语义

## IE8兼容性

使用：html5shiv-master

直接应用源文件即可：

<script src="js/html5shiv.js" type="text/javascript"></script>

# HTML5语义表单

## 表单简介

<**form action="" method=""**></**form**>

利用form创建表单，action是表单提交地址，method有2个属性值：method="post" 用户输入的信息不显示在地址栏，method="get" 用户输入的信息显示在地址栏

label标签

<**label for="email"**>邮箱</**label**>  
<**input type="email" name="email" id="email"**>

label中有一个属性for，属性值是某标签的ID属性值，这样就实现了label和拥有指定ID的标签关联。如上例，label和input关联，此时在页面中点击“邮箱”两个字，input输入框直接获取焦点，不用点击输入框也可以

## 新增表单

|  |  |
| --- | --- |
| **<input tpye="">** | |
| **文本输入** | |
| email | 电子邮箱输入框（@前后有值就可以，不会验证） |
| tel | 电话号码输入框（即使输入字母也不会校验） |
| url | 网页URL输入框（以<http://www>开头） |
| search | 搜索输入框（自动填充默认打开） |
| number | 数字选择 |
| number – min、max、step | 最小，最大取值，间隔取值 |
| **数据选择** | |
| range | 特定范围内的数值选择器 |
| range – min、max、step | 最小，最大取值，间隔取值 |
| color | 颜色选择 |
| datetime-local | 日期+时间选择 |
| time | 时间选择 |
| date | 日期选择 |
| week | 周选择 |
| month | 月选择 |

## input新属性

### placeholder

提示用户输入

**placeholder="请输入手机号"**

### autocomplete

是否保存用户输入值。默认为on，关闭提示选择off

### autofocus

自动获取输入焦点，可以不写属性值，推荐写属性值

**autofocus="autofocus"**

### required

验证类：不能为空值。当表单提交时，输入项必须有内容

### pattern

验证类：基于正则表达式。但不用写正则表达式中//和开始结尾符号^$

如验证手机号正则表达式：/^1[3-9][0-9]{9}$/

**pattern="1[3-9][0-9]{9}"**

pattern 有安全隐患，重要的验证必须放在服务器端！！！

### formaction

在submit里定义提交地址，可以分开提交表单内容至不同的服务器地址。常见应用场景：保存草稿

<**input type="submit" value="保存" formaction="email\_save.html"**>

此时点保存，页面提交到email\_save.html，而不是提交到form的action地址

### formnovalidate

不做表单验证，设置特定的请求不需要验证

<**input type="text" name="userId" placeholder="请输入手机号" required="required"**>

<**input type="submit" value="保存" formaction="email\_save.html" formnovalidate="formnovalidate"**>

第一个input使用required，当表单提交时，输入项必须有内容，否则无法提交；而第二个input设置了formnovalidate="formnovalidate"，当点击保存时，如果手机号没有填写，依然可以提交

### formmethod

可以在input中定义是否在地址栏显示用户输入的信息

<**input type="submit" value="其他" formaction="other.html" formmethod="post"**>

### maxlength

控制表单输入项的文本长度

### list

配合<datalist>元素使用，实现数据过滤选单效果。

## 新元素datalist

数据过滤选单数据源。

<**input type="text" name="text" list="dl"**>  
<**datalist id="dl"**>  
 <**option value="1"**>Html5</**option**>  
 <**option value="2"**>Java</**option**>  
 <**option value="3"**>Php</**option**>  
 <**option value="4"**>Php-2</**option**>  
</**datalist**>

## input事件

onchange：当input输入值发生改变，触发该事件

oninput：用户输入数据时，实时触发

区别：onchange是当用户输入完成，由获取焦点--输入--失去焦点后触发，oninput是用户每输入一次，触发一次

### 自定义验证信息





当验证不通过时，系统默认会有一个提示，但这个提示有时候用户会看不懂，比如上面这个，所请求的格式到底是什么格式？这时候就需要自定义提示信息比如“请输入正确的手机号”

自定义验证失败信息

1. inputDom 监听 oninput

2. inputDom 监听 setCustomValidity()

**var *testDom*** = **document**.querySelector(**"#test"**);

*//inputDom 监听 oninput*

***testDom***.oninput = **function** () {

*//定义正则表达式***var** exp = /^1[3-9][0-9]{9}$/;  
*//验证用户输入是否符合要求***var** res = exp.test(***testDom***.**value**);

**if**(res){  
 ***testDom***.setCustomValidity(**""**);

*/\*这里如果不写空串，当用户第一次验证错误后，会弹出提示；  
 而当填写正确的手机号后也提示错误，因为把上一次的提示信息记录了  
 正确的时候需要把提示信息清除掉，用空串覆盖\*/*  
 *//null不能作为 setCustomValidity 的参数* }**else**{  
 *//验证错误* ***testDom***.setCustomValidity(**"请输入一个合法的手机号"**)  
 }  
}

## form新属性

### novalidate：

统一设置：不对输入进行验证（默认开启验证）

### autocomplete

统一设置：输入是否自动填充

# 多媒体选择器

@media screen 视口选择器 and与括号之间必须有空格

**@media screen and** (**min-width**: 768**px**) **and** (**max-width**: 1024**px**){  
 .**box**{  
 **background-color**: **#00f**;  
 }  
}

当视口的宽度大于等于768px，小于等于1024px时，.box显示蓝色

**@media screen and** (**min-width**: 768**px**) **and** (**max-width**: 1024**px**){  
 */\*多媒体选择器内部样式可能被正常样式覆盖，可以使用 !important 强制生效\*/* .**box**{  
 **background-color**: **#00f !important**;  
 }  
 }  
 .**box**{  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **#f00**;  
 }

当多媒体选择器样式书写时在正常样式上面，会被正常样式覆盖，这时加一个!important来强制生效，由于这个样式只在特定条件下（宽度大于等于768px，小于等于1024px时）会生效，可以放心用

监控移动设备横屏或者竖屏

orientation: landscape 横屏

orientation: portrait 竖屏（默认值）

**@media screen and** (**orientation**: **landscape**) {  
 .**box**{  
 **background-color**: **blue**;  
 }  
}  
**@media screen and** (**orientation**: **portrait**) {  
 .**box**{  
 **background-color**: **green**;  
 }  
}

# 自适应页面与响应式页面

自适应页面主体结构随着浏览器宽度的调整无变换

响应式页面主体结构随着浏览器宽度的调整位置变换，或者隐藏。

# 视口

## 布局视口

固定的，每个手机浏览器厂商都有自己的设置，一般980px (1024px)

var layoutViewPort = document.documentElement.clientWidth;

## 可视视口

用户正在看到的网站的区域

var visViewPort = window.innerWidth;

PC端默认布局视口和可视视口是一样的，和屏幕分辨率一致，并且它的CSS像素的数量会随着用户缩放而改变。但如果有滚动条时，布局视口是不包含滚动条的，而可视视口包含滚动条

**多媒体选择器检测的是布局视口的宽度**

## 理想视口

布局视口的默认宽度并不是一个理想的宽度。显然用户希望在进入页面时可以不需要缩放就可以有一个理想的浏览和阅读尺寸。理想视口就是为移动端设备准备的。只有手动添加meta视口标签方才生效。如果没有meta视口标签，那么布局将会维持它的默认宽度。

* <meta name=”viewport” content=””>
  + width ：可视区域宽度，可以是一个具体的数字也可以是device-width:**可以让网页的宽度自动适应手机屏幕的宽度**
  + height ：可视区域高度，可以是一个具体的数字也可以是device-height。
  + inital-scale：页面首次显示时，可视区域的缩放比例，取值1.0页面按实际比列显示，无任何缩放。
  + **minimun-scale**：可视区域最小缩放级别，取值1.0禁止用户缩放至实际尺寸以下。
  + **maximun-sacle**：可视区域最大缩放级别，取值1.0禁止用户放大至实际尺寸以上。
  + **user-scalable**：用户是否可以对页面进行缩放。no禁止缩放，yes可以缩放，如果可以缩放，minimun-scale和maximun-sacle不能设置为1.0

**移动端的网页需要设置如下meta**

* **<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">**

# rem

* rem (root em) ，与px ， em 一样也是一个单位。不同的是它是基于html元素（html = root element）计算。
* 该单位常与媒体查询一起使用。
* 优点是：只要改变html的font-size大小，所有基于该元素计算的元素相关尺寸随之等比改变。
* rem 缺点
  + 1rem = 16px 需要进行换算，非常麻烦。
  + 解决办法： html{ font-size: 62.5%}, 此时可以使用1.2rem表示12px,计算比较方便。但是致命的问题随之而来，**Chrome浏览器最小字体是12px**，如果设置了11px及以下，仍显示12px。可以**使用缩放函数scale()**来缩小到12px以下，但缩放函数只能缩放块元素，需要把文字放到块元素中
  + 目前普遍被接收的解决办法：html{ font-size: 625%}。但是这种方法使用不是很方便。

## 如何选择使用哪种测量单位

* px：在PC端使用，移动端谨慎使用。
* em：避免嵌套使用，**基于自身字体的大小**。
* %：避免嵌套使用, **基于父元素。**
* rem：**基于html元素字体大小**。注意Chrome不支持12px以下像素。

# 渐进增强和优雅降级

* **渐进增强**：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。
* **优雅降级**：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。