<!DOCTYPE html>

# HTML5 第一天笔记

## 基础内容

### 1. HTML5 简介

HTML5 是 HTML 标准的下一个重要版本，用来替代 HTML 4.01，XHTML 1.0 以及 XHTML 1.1。HTML5 也是一种在万维网上构建和呈现内容的标准。

HTML5 是万维网联盟（W3C）和网页超文本技术工作小组（WHATWG）合作的产物。

HTML5 是近十年来 Web 开发标准最巨大的飞跃。HTML5 并非仅仅用来表示 Web 内容，它将 Web 带入一个成熟的应用平台，在 HTML5 平台上，视频、音频、图象、动画，以及同电脑的交互都被标准化。

#### 1）HTML5 引入了许多新元素和属性帮助我们构建现代化的网站。下面是 HTML5 引入的主要特性：

* 新的语义化元素： 比如 <header>，<footer> 和 <section>。
* 表单 2.0： 改进了 HTML Web 表单，为 <input> 标签引入了一些新的属性。
* 持久的本地存储： 为了不通过第三方插件实现。
* WebSocket： 用于 Web 应用程序的下一代双向通信技术。
* 服务器推送事件： HTML5 引入了从 Web 服务器到 Web 浏览器的事件，也被称作服务器推送事件（SSE）。
* Canvas： 支持用 JavaScript 以编程的方式进行二维绘图。
* 音频和视频： 在网页中嵌入音频或视频而无需借助第三方插件。
* 地理定位： 用户可以选择与我们的网页共享他们的地理位置。
* 拖放： 把同一网页上的条目从一个位置拖放到另一个位置。

#### 2）HTML5 浏览器支持：

最新版 Apple Safari，Mozilla FireFox 和 Opera 支持大部分 HTML5 新特性，IE9 也支持一些 HTML5 的新功能。

预装在 iPhones，iPads 和 Android 手机上的手机浏览器都对 HTML5 有良好的支持。

**值得注意的是：** IE 8及之前版本浏览器基本不支持 HTML5 新特性。

### 2. HTML5 参考网站

<https://developer.mozilla.org/zh-CN/>

## 表单 2.0

### 1. Input 新类型

| **类型** | **描述** | **说明** |
| --- | --- | --- |
| email | 邮箱类型 | 判断当前字符串中是否包含@符号 |
| search | 搜索类型 |  |
| url | 网址类型 | 判断当前字符串中是否包含http:// |
| number | 数字类型 |  |
| tel | 电话号码类型 | 只在移动端浏览器有效 |
| range | 范围类型 | 适用于应该包含某个范围内数值的输入字段 |
| color | 颜色类型 |  |
| date | 日期类型 | 按照 ISO 8601 编码的日期（包括年，月，日） |
| month | 月份类型 | 由 ISO 8601 编码的年和月组成的日期 |
| week | 星期类型 | 由 ISO 8601 编码的年和星期数组成的日期 |

下述示例代码使用了 HTML5 提供的表单 2.0 中的 input 类型：

<form>

Email:<input type="email"><br>

搜索:<input type="search"><br>

URL地址:<input type="url"><br>

电话号码:<input type="tel"><br>

数字类型:<input type="number" min="18" max="50" step="2"><br>

范围类型:<input type="range" min="0" max="10" step="2" value="10"><br>

颜色类型:<input type="color"><br>

日期类型:<input type="date"><br>

<input type="submit">

</form>

#### 1）email类型

email 类型是专门用于输入 email 地址的文本框。

如果输入内容不是 email 地址，则不允许提交，并提示错误信息。效果如下图所示：

根据上图效果，总结 email 类型的规则是：提交表单时，检查 email 类型的文本框中是否包含 @ 符号。

**值得注意的是：** email 类型的文本框并不能检查 email 地址是否真实存在。

<input type="email" />

#### 2）url类型

url 类型是专门用于输入 url 地址的文本框。

如果输入内容不是 url 地址，则不允许提交，并提示错误信息。效果如下图所示：

根据上图效果，总结 url 类型的规则是：提交表单时，检查 url 类型的文本框中是否包含 http:// 符号。

<input type="url" />

#### 3）search类型

search 类型是专门用于输入搜索关键词的文本框。

在移动设备中，当 search 类型的元素获取焦点时，键盘的“前往”会改为“搜索”字样。效果如下图所示：

* iPhone设备：
* Android设备：

**值得注意的是：**在PC端浏览器 search 类型与 text 类型外观相同。

<input type="search" />

#### 4）tel类型

tel 类型是专门用于输入电话号码的文本框。

在移动设备中，当 tel 类型的元素获取焦点时，键盘使用数字面板。效果如下图所示：

* iPhone设备：
* Android设备：

**值得注意的是：**该元素没有特殊的校验规则，不强制输入数字。

<input type="tel" />

#### 5）number类型

number 类型是专门用于输入数字的文本框。

如果输入内容不是数字，则不允许提交，并提示错误信息。效果如下图所示：

根据上图效果，总结 number 类型的规则是：提交表单时，检查 number 类型的文本框中是否为数字。

| **属性名称** | **类型** | **描述** |
| --- | --- | --- |
| min | Number | 允许输入的最小数值。 |
| max | Number | 允许输入的最大数值。 |
| step | Number | 设置增加或减少的间隔。（步长） |

<input type="number" min="0" max="100" step="5" />

#### 6）date类型

date 类型是专门用于输入日期，提供日期选择控件。

当用户单击向下箭头按钮时浏览器页面中显示日历。效果如下图所示：

<input type="date" />

#### 7）range类型

range 类型是专门用于输入一定范围内数字值。

range 类型在HTML页面显示为滑动条。效果如下图所示：

| **属性名称** | **类型** | **描述** |
| --- | --- | --- |
| min | Number | 允许输入的最小数值。 |
| max | Number | 允许输入的最大数值。 |
| step | Number | 设置增加或减少的间隔。（步长） |
| value | Number | 设置滑动条的当前值。 |

<input type="range" min="0" max="100" step="5" value="50" />

#### 8）color类型

color 类型是专门用于选取颜色。

color 类型提供了一个颜色选取器。效果如下图所示：

* 显示效果：
* 颜色选取器：

color 类型的显示效果，当用户单击 color 类型弹出显示颜色选取器。

<input type="color" />

#### 练习：自定义调色器

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>自定义调色器</title>

<style>

body {

padding-left: 100px;

padding-top: 100px;

}

.show {

width: 300px;

height: 300px;

border: 1px solid black;

}

</style>

</head>

<body>

<div class="show"></div>

<br><br>

红:<input onchange="mychange()" id="red" min="0" max="255" step="1" value="255" type="range"><br>

绿:<input onchange="mychange()" id="green" min="0" max="255" step="1" value="255" type="range"><br>

蓝:<input onchange="mychange()" id="blue" min="0" max="255" step="1" value="255" type="range">

<script>

function mychange(){

var red = document.getElementById("red").value;

var green = document.getElementById("green").value;

var blue = document.getElementById("blue").value;

var show = document.getElementsByTagName("div")[0];

show.style.backgroundColor = "rgb("+red+","+green+","+blue+")";

}

</script>

</body>

</html>

### 2. 表单新元素

| **元素名称** | **描述** |
| --- | --- |
| <datalist> | <input>标签定义选项列表。请与 input 元素配合使用该元素，来定义 input 可能的值 |
| <progress> | <progress> 标签定义运行中的任务进度（进程） |
| <meter> | <meter> 标签定义度量衡。仅用于已知最大和最小值的度量 |
| <output> | <output> 标签定义不同类型的输出，比如脚本的输出 |

#### 1）datalist元素

<datalist> 元素规定输入域的选项列表。

<datalist> 属性规定 form 或 input 域应该拥有自动完成功能。当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项：

使用 <input> 元素的列表属性与 <datalist> 元素绑定：

<input list="browsers">

<datalist id="browsers">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

#### 2）progress元素

<progress> 标签表示进度条，用于显示一个任务的完成进度。

**属性：**

| **属性** | **值** | **描述** |
| --- | --- | --- |
| max | number | 规定需要完成的值 |
| value | number | 规定进程的当前值 |

**提示：**请将 <progress> 标签与 JavaScript 一起使用来显示任务的进度。

**值得注意的是：**<progress> 标签不适合用来表示度量衡（例如，磁盘空间使用情况或相关的查询结果）。表示度量衡，请使用 <meter> 标签代替。

<progress id="progress" max="100" value="0"></progress>

<script>

var progress = document.getElementById("progress");

var max = progress.max;

var t = setInterval(function(){

var v = progress.value;

if(v == max){

clearInterval(t);

}else{

v++;

progress.value = v;

}

},100);

</script>

#### 3）meter元素

<meter> 标签定义度量衡。仅用于已知最大和最小值的度量。

比如：磁盘使用情况，查询结果的相关性等。

**属性：**

| **属性** | **值** | **描述** |
| --- | --- | --- |
| high | number | 规定被界定为高的值的范围 |
| low | number | 规定被界定为低的值的范围 |
| max | number | 规定范围的最大值 |
| min | number | 规定范围的最小值 |
| value | number | 必需。规定度量的当前值 |

**值得注意的是：**<meter> 不能作为一个进度条来使用， 进度条 <progress> 标签。

<meter min="0" max="100" value="95" low="10" high="90"></meter>

#### 4）output元素

<output> 元素用于不同类型的输出，比如计算或脚本输出：

<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0

<input type="range" id="a" value="50">100 +

<input type="number" id="b" value="50">=

<output name="x" for="a b"></output>

</form>

### 3. 表单新属性

#### 1）placeholder属性

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

简短的提示在用户输入值前会显示在输入域上。

**值得注意的是：**placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

<input type="text" name="fname" placeholder="First name">

#### 2）autofocus属性

autofocus 属性是一个 boolean 属性。

autofocus 属性规定在页面加载时，域自动地获得焦点。

First name:<input type="text" name="fname" autofocus>

#### 3）multiple属性

multiple 属性是一个 boolean 属性。

multiple 属性规定<input> 元素中可选择多个值。

**值得注意的是：**multiple 属性适用于以下类型的 <input> 标签：email 和 file。

Email: <input type="email" multiple>

#### 4）form属性

form 属性规定输入域所属的一个或多个表单。

**值得注意的是：**如需引用一个以上的表单，请使用空格分隔的列表。

位于form表单外的input 字段引用了 HTML form (该 input 表单仍然属于form表单的一部分)：

<form action="demo-form.php" id="form1">

First name: <input type="text" name="fname"><br>

<input type="submit" value="Submit">

</form>

Last name: <input type="text" name="lname" form="form1">

## 表单验证

### 1. 验证属性

| **属性名称** | **描述** |
| --- | --- |
| required | 验证不为空 |
| pattern | 匹配正则表达式 |
| min和max | 验证最小值和最大值 |
| minlength和maxlength | 验证最小长度和最大长度 |
| validity | 获取validityState对象 |

#### 1）required属性

required 属性规定必须在提交之前填写输入域（不能为空）。

**值得注意的是：**required 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, number, checkbox, radio 以及 file。

Username: <input type="text" name="usrname" required>

#### 2）pattern属性

pattern 属性的值与指定的正则表达式进行匹配（一般用于验证特定格式）。

**值得注意的是：**pattern 属性适用于以下类型的 <input> 标签: text, search, url, tel, email, 和 password。

Country code: <input type="text" name="country\_code" pattern="[A-Za-z]{3}" title="Three letter country code">

#### 3）min和max属性

min、max 和 step 属性用于为包含数字或日期的 input 类型规定限定（约束）。

**值得注意的是：**min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。

Enter a date before 1980-01-01:

<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:

<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):

<input type="number" name="quantity" min="1" max="5">

#### 4）maxlength属性

maxlength 属性用于设定允许输入的最大字符个数。

**值得注意的是：**maxlength 属性应该叫做限制属性。

### 2. 有效状态

ValidityState 代表了一个元素可以处于的与约束验证相关的有效性状态。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| valid | Boolean | 其他的约束验证条件都不为 true |
| valueMissing | Boolean | 该元素有 required 属性,但却没有值 |
| typeMismatch | Boolean | 该元素的值不符合元素类型所要求的格式(当 type 是 email 或者 url 时) |
| patternMismatch | Boolean | 该元素的值与指定的 pattern 属性不匹配 |
| stepMismatch | Boolean | 该元素的值不符合由 step 属性指定的规则 |
| rangeUnderflow | Boolean | 该元素的值小于指定的 min 属性 |
| tooLong | Boolean | 该元素的值的长度超过了 HTMLInputElement 或者 HTMLTextAreaElement 对象指定的 maxlength 属性中的值 |
| rangeOverflow | Boolean | 该元素的值大于指定的 max 属性 |
| customError | Boolean | 该元素的自定义有效性消息已经通过调用元素的setCustomValidity() 方法设置成为一个非空字符串 |

<form>

用户名:<input id="user" type="text" required><br>

密码:<input id="pwd" type="text" pattern="^[0-9]{6,8}$"><br>

Email:<input id="mail" type="email"><br>

年龄:<input id="age" step="5" type="number" min="20"><br>

<input type="submit">

</form>

<script>

var user = document.getElementById("user");

user.onblur = function(){

if(user.validity.valid){

alert("验证通过.");

}else if(user.validity.valueMissing){

alert("用户名为空.");

}

}

var pwd = document.getElementById("pwd");

pwd.onblur = function(){

if(pwd.validity.valid){

alert("验证通过.");

}else if(pwd.validity.patternMismatch){

alert("密码输入有误.");

}

}

var mail = document.getElementById("mail");

mail.onblur = function(){

if(mail.validity.valid){

alert("验证通过.");

}else if(mail.validity.typeMismatch){

alert("Email输入有误.");

}

}

var age = document.getElementById("age");

age.onblur = function(){

if(age.validity.valid){

alert("验证通过.");

}else if(age.validity.rangeUnderflow){

alert("年龄过小.");

}else if(age.validity.stepMismatch){

alert("年龄有误.");

}

}

</script>

### 3. setCustomValidity() 方法

setCustomValidity() 方法设置自定义验证信息，用于即将实施与验证的约束来覆盖预定义的信息。

<form>

<input id="data" type="text" required>

<input type="submit">

</form>

<script>

var data = document.getElementById("data");

data.onblur = function(){

if(data.validity.valueMissing){

data.setCustomValidity("不能为空.");

}else if(data.validity.customError){

data.setCustomValidity("");

}

}

</script>

# HTML5 第二天笔记

## 音视频处理

### 1. 视频处理

#### 1）基础内容

##### a. 目前实现网页视频播放的技术 - Flash

* Flash并不是浏览器原生支持(第三方组件)
* Flash的性能并不好
* 移动端不支持Flash技术

##### b. HTML 5 提供的视频处理技术 - <video>

* 提供了相对应的基本处理方式
* 提供了高级编程自定义方式

##### c. <video>元素所支持的视频格式

| **视频格式** | **类型** | **描述** |
| --- | --- | --- |
| MP4格式 | .mp4 | 目前最主流的视频格式。 |
| OGV格式 | .ogv |  |
| WEBM格式 | .webm | 是由Google公司推出的，目前唯一一个支持超高清的视频格式。 |

#### 2）如何使用<video>元素

##### a. 引入单个视频格式

<video> 标签的编写结构如下：

<video src="视频文件的路径" autoplay>

浏览器不支持的提示内容

</video>

例如以下示例代码：

<video src="../DATA/oceans-clip.mp4" autoplay width="640px" height="400px" style="background:black;">

非常抱歉,你的浏览器不支持该视频!

</video>

| **属性名称** | **类型** | **描述** |
| --- | --- | --- |
| width | Number | 设置视频的宽度。 |
| height | Number | 设置视频的高度。 |
| style |  | 设置CSS样式属性。 |
| class |  | 设置CSS样式属性。 |
| autoplay | Boolean | 设置是否自动播放视频。 |

##### b. 引入多个视频格式

<video> 标签和 <source> 标签的编写结构如下：

<video autoplay>

<source src="视频文件的路径" />

<source src="视频文件的路径" />

<source src="视频文件的路径" />

浏览器不支持的提示内容

</video>

例如以下示例代码：

<!-- 解决了浏览器对视频格式的兼容问题 -->

<video autoplay>

<!--

<source>元素

\* 引入视频文件(一个<video>元素允许包含多个<source>)

-->

<source src="../DATA/oceans-clip.mp4" />

<source src="../DATA/oceans-clip.ogv" />

<source src="../DATA/oceans-clip.webm" />

</video>

#### 3）<video>元素的属性

* autoplay属性：自动播放
* controls属性：提供控制面板

<video src="../DATA/oceans-clip.mp4" controls></video>

* loop属性：循环播放

<video src="../DATA/oceans-clip.mp4" autoplay loop></video>

* poster属性：播放之前实现一张图片

<video src="../DATA/oceans-clip.mp4" controls poster="../DATA/oceans-clip.png"></video>

* preload属性：预加载视频
  + none：不预加载
  + auto：默认值,尽快预加载
  + metadata：预加载除视频之外的内容(宽度、高度等)

#### 4）视频高级编程

##### a. 事件

| **事件名称** | **作用** |
| --- | --- |
| play | 视频播放时触发。 |
| pause | 视频暂停时触发。 |
| ended | 视频播放结束时触发。 |
| error | 视频播放错误时触发。 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>video元素的高级事件</title>

<style>

img {

position: absolute;

top : 66px;

left : 160px;

display : none;

}

</style>

</head>

<body>

<video id="mmedia" src="../DATA/oceans-clip.mp4" controls></video>

<img id="adv" src="../DATA/oceans-clip.png" width="320">

<script>

var mmedia = document.getElementById("mmedia");

var adv = document.getElementById("adv");

mmedia.addEventListener("pause",function(){

adv.style.display = "block";

});

mmedia.addEventListener("play",function(){

adv.style.display = "none";

});

</script>

</body>

</html>

##### b. 方法

| **方法名称** | **作用** |
| --- | --- |
| play() | 用于播放视频。 |
| pause() | 用于暂停视频。 |
| load() | 用于加载视频。 |
| canPlayType() | 判断当前浏览器是否支持当前视频格式。 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>video元素的高级方法</title>

</head>

<body>

<video id="mmedia" src="../DATA/oceans-clip.mp4"></video>

<br>

<input type="button" id="btn" value="播放">

<script>

var btn = document.getElementById("btn");

var mmedia = document.getElementById("mmedia");

btn.onclick = function(){

if(mmedia.paused){// 播放

mmedia.play();

this.value = "暂停";

}else{// 暂停

mmedia.pause();

this.value = "播放";

}

}

</script>

</body>

</html>

##### c. 属性

| **属性名称** | **作用** |
| --- | --- |
| paused | 如果视频为暂停或未播放时,返回true。 |
| ended | 如果视频播放完毕时,返回true。 |
| duration | 返回当前视频的时长。 |
| currentTime | 获取或设置视频的当前位置。 |

**参考内容：**[https://developer.mozilla.org/en-US/docs/Web/HTML/Supportedmediaformats](https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats)

### 2. 音频处理

#### 1）基础内容

##### a. 目前音频处理技术

* Flash技术也可以音频处理
* Media Player播放器允许嵌入在网页中

##### b. HTML5 提供的音频处理 - <audio>

* 浏览器原生支持
* 性能很好
* 移动端支持

##### c. <audio>元素支持的音频格式

| **音频格式** | **类型** | **描述** |
| --- | --- | --- |
| MP3格式 | .mp3 | 目前最主流的音频格式。 |
| OGG格式 | .ogg |  |
| WAV格式 | .wav |  |

#### 2）如何使用<audio>元素

##### a. 引入单个音频格式

<audio src="音频文件的路径" autoplay>

浏览器不支持的提示内容

</audio>

##### b. 引入多个音频格式

<audio autoplay>

<source src="音频文件的路径" />

<source src="音频文件的路径" />

<source src="音频文件的路径" />

浏览器不支持的提示内容

</audio>

#### 3）<audio>元素的特有属性

| **属性名称** | **作用** |
| --- | --- |
| autoplay | 设置是否自动播放。 |
| controls | 提供控制面板。 |
| loop | 设置循环播放。 |
| preload | 预加载视频。 |

#### 4）<audio>元素的高级编程

##### a. 事件

| **事件名称** | **作用** |
| --- | --- |
| play | 视频播放时触发。 |
| pause | 视频暂停时触发。 |
| ended | 视频播放结束时触发。 |
| error | 视频播放错误时触发。 |

##### b. 方法

| **方法名称** | **作用** |
| --- | --- |
| play() | 用于播放视频。 |
| pause() | 用于暂停视频。 |
| load() | 用于加载视频。 |
| canPlayType() | 判断当前浏览器是否支持当前视频格式。 |

##### c. 属性

| **属性名称** | **作用** |
| --- | --- |
| paused | 如果视频为暂停或未播放时,返回true。 |
| ended | 如果视频播放完毕时,返回true。 |
| duration | 返回当前视频的时长。 |
| currentTime | 获取或设置视频的当前位置。 |

## Canvas 画布

### 1. 基础内容

#### 1）Canvas技术允许在HTML页面直接绘制图形

* 不再需要引入外部图片(图形),HTML页面性能有所提高。
* 可以实现一些比较复杂的图形绘制工作。

#### 2）Canvas技术主要应用方向

* Web应用方面主要实现图表类。
* 网页游戏方面 - 主要实现2D效果。

#### 3）HTML5 提供有关图形方面的技术

* Canvas - 主要以2D为主
* WebGL - 主要以3D为主
* SVG - 矢量图

#### 4）Canvas的特点

* 绘制的图形与HTML页面之间是无关系的
* 通过Canvas绘制的图形不能使用DOM API
* 通过Canvas绘制的图形不能绑定事件
* Canvas画布最终是以图片(png|jpg)形式出现
* 绘制图形默认的颜色为黑色

### 2. 如何使用Canvas画布

#### 1）在HTML页面中定义<canvas>元素

* 设置<canvas>元素宽度和高度使用属性方式。
* 使用CSS样式方式或属性方式设置<canvas>元素的宽度和高度。

<!--

在HTML页面中,定义<canvas>元素

\* 默认只定义<canvas>元素时

\* 效果非常类似于<div>元素,但不一样的地方:

\* <div>元素在默认情况下,不具有高度和宽度的

\* <canvas>元素在默认情况下,具有高度和宽度的

\* 宽度 - 300px

\* 高度 - 150px

\* 设置<canvas>元素的高度和宽度

\* (建议)使用属性width和height

\* 使用CSS中的属性width和height

\* 绘制的图形会被拉伸

-->

<canvas id="canvas" width="500px" height="500px" style="background:pink;"></canvas>

<!--

<canvas id="canvas" style="width:500px;height:500px;"></canvas>

-->

#### 2）在JS代码中实现

* 获取<canvas>元素
* 通过<canvas>元素,创建画布对象
  + getContext('2d')函数
  + 返回画布对象
* 利用画布对象进行图形的绘制

// 获取HTML页面中的<canvas>元素

var canvas = document.getElementById("canvas");

/\*

通过<canvas>元素,创建画布对象

使用getContext(type)函数,创建画布对象

\* 该函数返回画布对象

\* type参数

\* 设置当前创建的画布是2d还是3d

\* 注意

\* 参数选项是2d(3d效果使用WebGL)

\* 必须写成"2d"

\*/

var context = canvas.getContext("2d");

// 利用画布对象,进行绘制图形

context.fillRect(10,10,100,100);

### 3. 绘制图形

#### 1）绘制矩形

##### a. 绘制实心(填充)矩形

fillRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

##### b. 绘制空心(描边)矩形

strokeRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

##### c. 清除指定区域的矩形

clearRect(x,y,width,height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 绘制矩形的左上角的坐标值 |
| width | 设置绘制矩形的宽度(单位为px) |
| height | 设置绘制矩形的高度(单位为px) |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

// 1. 获取<canvas>元素

var canvas = document.getElementById("canvas");

// 2. 创建画布对象

var context = canvas.getContext('2d');

// 3. 绘制图形

// a. 绘制实心矩形

context.fillRect(10,10,100,100);

// b. 绘制空心矩形

context.strokeRect(120,10,100,100);

// c. 清除指定区域的矩形

context.fillRect(230,10,100,100);

context.clearRect(240,20,80,80);

</script>

#### 2）设置样式

| **属性名称** | **作用** |
| --- | --- |
| fillStyle | 设置填充样式 |
| strokeStyle | 设置描边样式 |
| globalAlpha | 设置透明度(0-1) |

**值得注意的是：**

* 一定要先设置样式(颜色),再绘制图形。
* 每次改变样式(颜色),重新设置。

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

// a. 设置填充样式

context.fillStyle = "pink";

// b. 绘制实心矩形

context.fillRect(10,10,100,100);

context.fillStyle = "blue";

context.fillRect(10,120,100,100);

// 设置描边样式

context.strokeStyle = "red";

context.strokeRect(120,10,100,100);

context.strokeStyle = "green";

context.strokeRect(120,120,100,100);

// 设置透明度

context.globalAlpha = 0.5;

context.fillRect(230,10,100,100);

context.fillStyle = "black";

context.globalAlpha = 0.1;

context.fillRect(230,120,100,100);

</script>

#### 3）设置渐变

##### a. 线性渐变

createLinearGradient(x1,y1,x2,y2)

线性渐变主要依靠基准线概念，所谓基准线就是设置线性渐变的标准。

| **参数名称** | **作用** |
| --- | --- |
| x1和y1 | 基准线的起点坐标值 |
| x2和y2 | 基准线的终点坐标值 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

/\*

设置线性渐变

createLinearGradient(x1,y1,x2,y2)方法

\* 该方法具有返回值,是渐变对象

\*/

var grd = context.createLinearGradient(0,0,100,100);

/\*

设置线性渐变的颜色和位置

addColorStop(position,color)

\* position - 设置颜色的位置

\* 值的范围为 0 - 1

\* color - 设置颜色

\*/

grd.addColorStop(0,"red");

grd.addColorStop(1,"blue");

grd.addColorStop(0.5,"yellow");

// 将设置的线性渐变,赋值给样式(fillStyle和strokeStyle)

context.fillStyle = grd;

// 绘制矩形

context.fillRect(0,0,100,100);

</script>

##### b. 射线(扇形)渐变

createRadialGradient(x1,y1,r1,x2,y2,r2)

射线渐变主要依靠基准圆概念，所谓基准圆就是设置射线渐变的标准。

| **参数名称** | **作用** |
| --- | --- |
| x1和y1 | 第一个基准圆的圆心 |
| r1 | 第一个基准圆的半径 |
| x2和y2 | 第二个基准圆的圆心 |
| r2 | 第二个基准圆的半径 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

/\*

设置射线渐变

createRadialGradient(x1,y1,r1,x2,y2,r2)

\* 该方法返回渐变对象

\*/

var grd = context.createRadialGradient(100,100,100,canvas.width,canvas.height,200);

/\*

设置渐变颜色

\*/

grd.addColorStop(0,"red");

grd.addColorStop(1,"blue");

// 将渐变对象赋值给样式

context.fillStyle = grd;

// 绘制矩形

context.fillRect(0,0,canvas.width,canvas.height);

</script>

##### c. 设置渐变颜色

addColorStop(position,color)

| **参数名称** | **作用** |
| --- | --- |
| position | 设置渐变颜色的位置，值的范围必须是0-1。 |
| color | 设置渐变的颜色 |

#### 4）绘制文字

| **属性名称** | **作用** |
| --- | --- |
| font | 设置文字的属性（用法同 CSS 属性font）。 |
| textAlign | 水平对齐方式。（left：基准线在左边；center：基准线在中间；right：基准线在右边） |
| textBaseline | 垂直对齐方式。（top：基准线在上边；middle：基准线在中间；bottom：基准线在下边；hanging：悬挂基线；alphabetic：字母基线） |

##### a. 绘制实心文字

fillText(text,x,y)

| **参数名称** | **作用** |
| --- | --- |
| text | 绘制的文字内容。 |
| x和y | 绘制文字的坐标值。 |

##### b. 绘制空心文字

strokeText(text,x,y)

| **参数名称** | **作用** |
| --- | --- |
| text | 绘制的文字内容。 |
| x和y | 绘制文字的坐标值。 |

<canvas id="canvas" width="500px" height="500px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext('2d');

// 设置文字样式

context.font = "bold 48px 宋体";

// 基准线

context.beginPath();

context.moveTo(100,0);

context.lineTo(100,400);

context.stroke();

// 设置水平对齐

context.textAlign = "right";

// 绘制文字

context.fillText("text",100,50);

// 设置水平对齐

context.textAlign = "center";

// 绘制文字

context.fillText("text",100,100);

// 设置水平对齐

context.textAlign = "left";

// 绘制文字

context.fillText("text",100,150);

// 基准线

context.beginPath();

context.moveTo(0,300);

context.lineTo(500,300);

context.stroke();

// 设置垂直对齐

context.textBaseline = "top";

context.strokeText("text",0,300);

context.textBaseline = "middle";

context.strokeText("text",100,300);

context.textBaseline = "bottom";

context.strokeText("text",200,300);

context.textBaseline = "hanging";

context.strokeText("text",300,300);

context.textBaseline = "alphabetic";

context.strokeText("text",400,300);

</script>

#### 5）设置阴影

| **属性名称** | **作用** |
| --- | --- |
| shadowColor | 设置阴影颜色。 |
| shadowOffsetX | 设置水平方向阴影。 |
| shadowOffsetY | 设置垂直方向阴影。 |
| shadowBlur | 设置阴影程度。 |

# HTML5 第三天笔记

## Canvas绘图

### 1. 创建路径

图形的基本元素是路径。路径是通过不同颜色和宽度的线段或曲线相连形成的不同形状的点的集合。一个路径，甚至一个子路径，都是闭合的。使用路径绘制图形需要一些额外的步骤。

1. 首先，需要创建路径起始点。
2. 然后，使用画图命令去画出路径。
3. 之后，把路径封闭。
4. 一旦路径生成，你就能通过描边或填充路径区域来渲染图形。

#### 1）绘制矩形

##### 绘制矩形的实现步骤

1. 调用beginPath()方法，创建新建一条路径。
2. 使用rect(x, y, width, height)方法，设置矩形的坐标值及宽度和高度。
3. 通过fill()或stroke()方法进行绘制。

##### 使用方法说明

rect(x, y, width, height)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 表示矩形的左上角坐标值。 |
| width和height | 表示矩形的宽度和高度。 |

#### 2）绘制圆形

##### 绘制圆形的实现步骤

1. 调用beginPath()方法，创建新建一条路径。
2. 使用arc(x, y, radius, startAngle, endAngle, anticlockwise)方法，设置矩形的坐标值及宽度和高度。
3. 通过fill()或stroke()方法进行绘制。

##### 使用方法说明

arc(x, y, radius, startAngle, endAngle, anticlockwise)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 表示圆形的圆心坐标值。 |
| radius | 表示圆形的半径。 |
| startAngle | 表示绘制圆形的开始点，值为 0。 |
| endAngle | 表示绘制圆形的结束点，值为 Math.PI\*2。 |
| anticlockwise | 表示是以顺时针还是以逆时针方式绘制圆形，Boolean值。（默认值为false，表示顺时针。） |

#### 3）绘制弧形

##### 绘制圆形的实现步骤

1. 调用beginPath()方法，创建新建一条路径。
2. 使用arc(x, y, radius, startAngle, endAngle, anticlockwise)方法，设置矩形的坐标值及宽度和高度。
3. 通过fill()或stroke()方法进行绘制。

##### 使用方法说明

arc(x, y, radius, startAngle, endAngle, anticlockwise)

| **参数名称** | **作用** |
| --- | --- |
| x和y | 表示圆形的圆心坐标值。 |
| radius | 表示圆形的半径。 |
| startAngle | 表示绘制圆形的开始点（取值范围：0 至 Math.PI\*2。）。 |
| endAngle | 表示绘制圆形的结束点（取值范围：0 至 Math.PI\*2。）。 |
| anticlockwise | 表示是以顺时针还是以逆时针方式绘制圆形，Boolean值。（默认值为false，表示顺时针。） |

**值得注意的是：**

如果绘制的是空心弧形的话，在arc()方法调用后：

* 如果使用closePath()方法的话，绘制的图形会自动将终点和起点连接成线。
* 如果不用closePath()方法的话，绘制的图形会呈现开口状。

#### 4）绘制直线

1. 调用beginPath()方法，创建新建一条路径。
2. 使用moveTo(x, y)方法，设置直线的起点坐标值。
3. 使用lineTo(x, y)方法，设置直线的终点坐标值。
4. 通过stroke()方法进行绘制。

#### 5）绘制折线

1. 调用beginPath()方法，创建新建一条路径。
2. 使用moveTo(x, y)方法，设置直线的起点坐标值。
3. 使用lineTo(x, y)方法，设置直线的终点坐标值。
4. 通过stroke()方法进行绘制。

**值得注意的是：**在绘制折线的时候，lineTo()方法既可以绘制折点，也可以绘制终点。

#### 6）绘制多边形

1. 调用beginPath()方法，创建新建一条路径。
2. 使用moveTo(x, y)方法，设置直线的起点坐标值。
3. 使用lineTo(x, y)方法，设置直线的终点坐标值。
4. 调用closePath()方法，闭合当前绘制的路径。
5. 通过fill()或stroke()方法进行绘制。

### 2. 设置线型

所有画布操作都使用相同的线型，即默认线型。实际上线条的宽度、端点都可以根据实际绘图需要进行调整。

#### 1）设置线宽

lineWidth：指定线条粗细，默认值是1.0。

<canvas id="myCanvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

context.beginPath();

context.moveTo(100, 150);

context.lineTo(450, 50);

context.lineWidth = 15;

context.stroke();

</script>

#### 2）设置端点形状

lineCap：指定线条端点形状。

* butt：默认，向线条的每个末端添加平直的边缘。
* round：向线条的每个末端添加圆形线帽。
* square：向线条的每个末端添加正方向线帽。

**值得注意的是：**round和square会使线条略变微长。

<canvas id="myCanvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

// butt line cap (top line)

context.beginPath();

context.moveTo(200, canvas.height / 2 - 50);

context.lineTo(canvas.width - 200, canvas.height / 2 - 50);

context.lineWidth = 20;

context.strokeStyle = '#0000ff';

context.lineCap = 'butt';

context.stroke();

// round line cap (middle line)

context.beginPath();

context.moveTo(200, canvas.height / 2);

context.lineTo(canvas.width - 200, canvas.height / 2);

context.lineWidth = 20;

context.strokeStyle = '#0000ff';

context.lineCap = 'round';

context.stroke();

// square line cap (bottom line)

context.beginPath();

context.moveTo(200, canvas.height / 2 + 50);

context.lineTo(canvas.width - 200, canvas.height / 2 + 50);

context.lineWidth = 20;

context.strokeStyle = '#0000ff';

context.lineCap = 'square';

context.stroke();

</script>

#### 3）设置交点形状

lineJoin：指定两条线之间的连接点形状。

* round：创建圆角。
* bevel：创建斜角。
* miter：默认，创建尖角。

miterLimit：与lineJoin一起使用，当lineJoin设置为miter时，可用于确定线条交接点的延伸范围。

<canvas id="myCanvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

// set line width for all lines

context.lineWidth = 25;

// miter line join (left)

context.beginPath();

context.moveTo(99, 150);

context.lineTo(149, 50);

context.lineTo(199, 150);

context.lineJoin = 'miter';

context.stroke();

// round line join (middle)

context.beginPath();

context.moveTo(239, 150);

context.lineTo(289, 50);

context.lineTo(339, 150);

context.lineJoin = 'round';

context.stroke();

// bevel line join (right)

context.beginPath();

context.moveTo(379, 150);

context.lineTo(429, 50);

context.lineTo(479, 150);

context.lineJoin = 'bevel';

context.stroke();

</script>

## 处理图像

在HTML5中，不仅可以使用Canvas API来绘制图形，还可以读取磁盘或网络中的图像文件，然后使用Canvas API将该图像绘制在画布中。

### 1. 绘制图像

#### 1）加载图像

* 使用相同页面中的图片。
* 使用相同页面中的其他Canvas元素。
* 可以脚本通过Image( )构造函数创建图像。

#### 2）绘制图像

使用 drawImage() 方法绘制图像，该方法具体用法如下：

drawImage(img, x, y)

| **参数名称** | **作用** |
| --- | --- |
| img | 需要绘制的图像。 |
| x和y | 绘制图像的坐标值。 |

drawImage(img, x, y, width, height)

| **参数名称** | **作用** |
| --- | --- |
| img | 需要绘制的图像。 |
| x和y | 绘制图像的坐标值。 |
| width和height | 设置绘制图像的宽度和高度。 |

<canvas id="myCanvas" width="578" height="400"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

var imageObj = new Image();

imageObj.src = 'darth-vader.jpg';

imageObj.onload = function() {

context.drawImage(imageObj, 69, 50);

};

</script>

**值得注意的是：**若调用 drawImage 时，图片没装载完，那什么都不会发生（在一些旧的浏览器中可能会抛出异常）。因此你应该用load时间来保证不会在加载完毕之前使用这个图片：

var img = new Image(); // 创建img元素

img.src = 'myImage.png'; // 设置图片源地址

img.onload = function(){

// 执行drawImage语句

}

### 2. 平铺图像

所谓图像平铺就是用按一定比例缩小后的图像将画布填满。

1. 加载图像。
2. 通过 createPattern(img, type) 方法设置平铺方式。
3. 将平铺对象赋值给filleStyle或strokeStyle属性。
4. 将平铺的图像进行绘制。

createPattern(img, type)

| **参数名称** | **作用** |
| --- | --- |
| img | 需要平铺的图像。 |
| type | 平铺方式。（no-repeat：不平铺；repeat-x：水平方向平铺；repeat-y：垂直方向平铺；repeat：全方向平铺） |

该方法返回平铺对象。

<canvas id="myCanvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

var imageObj = new Image();

imageObj.src = 'wood-pattern.png';

imageObj.onload = function() {

var pattern = context.createPattern(imageObj, 'repeat');

context.rect(0, 0, canvas.width, canvas.height);

context.fillStyle = pattern;

context.fill();

};

</script>

**值得注意的是：**若调用 createPattern 时，图片没装载完，那什么都不会发生（在一些旧的浏览器中可能会抛出异常）。因此你应该用load时间来保证不会在加载完毕之前使用这个图片：

var img = new Image(); // 创建img元素

img.src = 'myImage.png'; // 设置图片源地址

img.onload = function(){

// 执行createPattern语句

}

### 3. 切割图像

1. 调用beginPath()方法，创建新建一条路径。
2. 使用rect()或arc()方法
3. 通过clip()方法进行切割。

<canvas id="canvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('canvas');

var context = elem.getContext('2d');

var image=new Image();

image.src="img/flower.jpg";

image.onload=function(){

context.drawImage(image,0,0,280,190);

}

context.beginPath();

context.arc(140,95,60,0,Math.PI\*2,true);

context.closePath();

context.clip();

</script>

### 4. 画布方法

#### 1）状态方法

* save()，保存当前画布属性、状态。
* restore()，恢复画布属性、状态。

<canvas id="canvas" width="578" height="200"></canvas>

<script>

var ctx = document.getElementById('canvas').getContext('2d');

ctx.fillRect(0,0,150,150); // Draw a rectangle with default settings

ctx.save(); // Save the default state

ctx.fillStyle = '#09F' // Make changes to the settings

ctx.fillRect(15,15,120,120); // Draw a rectangle with new settings

ctx.save(); // Save the current state

ctx.fillStyle = '#FFF' // Make changes to the settings

ctx.globalAlpha = 0.5;

ctx.fillRect(30,30,90,90); // Draw a rectangle with new settings

ctx.restore(); // Restore previous state

ctx.fillRect(45,45,60,60); // Draw a rectangle with restored settings

ctx.restore(); // Restore original state

ctx.fillRect(60,60,30,30); // Draw a rectangle with restored settings

</script>

#### 2）转换方法

translate(x, y)：用来移动 canvas 和它的原点到一个不同的位置。

* x 是左右偏移量。
* y 是上下偏移量。

scale(x, y)：用它来增减图形在 canvas 中的像素数目，对形状，位图进行缩小或者放大。

* x,y 分别是横轴和纵轴的缩放因子，它们都必须是正值。
* 值分类：
  + 值比 1.0 小表示缩小。
  + 比 1.0 大则表示放大。
  + 值为 1.0 时什么效果都没有。

rotate(angle)：用于以原点为中心旋转 canvas。

* 旋转的角度(angle)，它是顺时针方向的，以弧度为单位的值。

## Chart.js库

### 1. 基础内容

Chart.js是一个简单、面向对象、为设计者和开发者准备的图表绘制工具库。

官方网址：<http://www.chartjs.org/>

Chart.js的特点：

* 基于HTML 5：Chart.js基于HTML5 canvas技术，支持所有现代浏览器，并且针对IE7/8提供了降级替代方案。
* 简单、灵活：Chart.js不依赖任何外部工具库，轻量级（压缩之后仅有4.5k），并且提供了加载外部参数的方法。

### 2. 使用Chart.js

#### 1）使用Chart.js的步骤

* 在HTML页面中引入Chart.js文件。

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

* 创建<canvas>元素：用于显示Chart图表的容器。

<canvas id="myChart" width="400" height="400"></canvas>

* 获取Canvas对象。

document.getElementById("myChart").getContext("2d");`

* 创建Chart图表对象。

new Chart(ctx);

* 通过Chart图表对象进行绘制。

chart.PolarArea(data);

#### 2）Chart.js全局配置

Chart.js 全局配置是在chart.js 第一个正式版本中引入。Chart.js 全局配置用于改变所有图表的类型，避免了需要在每一个图表中单独进行设置。当然，Chart.js 全局配置也可以专门为某一个特定的图表进行配置。

语法如下：

Chart.defaults.global.参数名 = 参数值;

例如如下方式设定：

Chart.defaults.global.responsive = true;

#### 3）绘制图表

##### a. 曲线图

曲线图就是将多个数据点绘制在一条线上，通常被用于展示趋势的数据或两组数据之间的对比。

new Chart(ctx).Line(data, options)

* data：用于设置曲线上的数据、样式及名称。
* options：选项，用于配置曲线图。

##### b. 柱状图

柱状图就是使用柱状方式显示数据的一种方式，通常被用于展示趋势的数据或多组数据之间的比较。

new Chart(ctx).Bar(data, options)

* data：用于设置柱状图上的数据、样式及名称。
* options：选项，用于配置柱状图。

##### c. 饼状图

饼状图可能是所有图表中最为常用的一种，就是将一个圆划分成若干个部分，每个弧形展示每个数据的比例值。通常被用于展示多组数据之间的比例。

new Chart(ctx).Pie(data,options)

* data：用于设置饼图的数据、样式及名称。
* options：选项，用于配置饼图。

##### d. 雷达图

雷达图就是一种展示多个数据点以及它们之间变化的方式，通常被用于比较点的两个或多个不同的数据集。

new Chart(ctx).Radar(data, options)

* data：用于设置雷达图的数据、样式及名称。
* options：选项，用于配置雷达图。

##### e. 环形图

环形图类似于饼状图，但环形图是一个空心的环形形状，通常被用于展示多组数据之间关系的比例。

new Chart(ctx).Doughnut(data,options)

* data：用于设置环形图的数据、样式及名称。
* options：选项，用于配置环形图。

##### f. 极地区域图

极地区域图类似于饼状图，但每一个扇形的角度和半径取决于不同的值，通常被用于需要展示类似于饼状图的比较数据的基础上，还需要展示范围值的比较。

new Chart(ctx).PolarArea(data, options)

* data：用于设置极地区域图的数据、样式及名称。
* options：选项，用于配置极地区域图。

<canvas id="canvas" width="300px" height="300px"></canvas>

<script>

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

// 初始化饼状图的数据 - 以数组格式

var data = [

{

// 当前数据占饼状图的比例

value: 30,

// 当前数据块的颜色

color:"blue",

// 当鼠标悬停在当前数据块时的颜色

highlight: "yellow",

// 当前数据块的提示信息

label: "Blue"

},

{

value: 50,

color: "#46BFBD",

highlight: "#5AD3D1",

label: "Green"

},

{

value: 100,

color: "#FDB45C",

highlight: "#FFC870",

label: "Yellow"

},

{

value: 40,

color: "#949FB1",

highlight: "#A8B3C5",

label: "Grey"

}

];

//Chart.defaults.global.responsive = true;

// 通过画布对象,创建Chart对象

var chart = new Chart(context);

// 利用Chart对象调用图表方法

chart.Pie(data);

</script>

# HTML5 第四天笔记

## 基础内容

### 1. 什么是 SVG

SVG 全称为 Scalable Vector Graphics，译为可缩放矢量图形,简称矢量图。是一种用来描述二维矢量图形的 XML 标记语言。

### 2. SVG与Flash的区别

|  | **SVG** | **Flash** |
| --- | --- | --- |
| 相同点 | 用于二维矢量图形 | 用于二维矢量图形 |
| 不同点 | 是一个开放的W3C标准，基于XML | 封闭的基于二进制格式的 |

### 3. SVG与Canvas的区别

| **SVG** | **Canvas** |
| --- | --- |
| 不依赖于分辨率 | 依赖于分辨率 |
| 使用DOM及事件处理器（DOM专门为SVG开放接口） | 不能使用DOM及事件处理器 |
| 不能实现游戏开发 | 可以实现游戏开发 |
| 实现大型渲染区域的应用(例如百度地图) | 是以图片(png |

### 4. SVG的优势

* 不需要专门的编辑器,文本编辑器都可以
* 可被搜索、索引、脚本化及压缩
* 图像不失真(和分辨率无关)

**值得注意的是：**

* SVG技术并不是专属于HTML5的
* SVG技术本身是一套独立的用于描述二维图形
* HTML5版本之前,以图片形式进行引入
* HTML5版本之后,允许在HTML页面直接使用SVG技术

**可参考的资源：**

* SVG 标准：<http://www.w3.org/Graphics/SVG/>
* SVG 教程：<https://developer.mozilla.org/zh-CN/docs/Web/SVG/Tutorial>

### 5. 如何使用 SVG

#### 1）SVG 文件（了解）

SVG 文件的扩展名为 “.svg”，使用的是 XML 技术的语法内容，并且可以在浏览器中直接运行。

<?xml version="1.0" encoding="utf-8" ?>

<!-- SVG的语法标准(必要的) -->

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<!-- xmlns="http://www.w3.org/2000/svg" SVG的命名空间 -->

<svg version="1.1" xmlns="http://www.w3.org/2000/svg">

<rect x="100" y="100" width="300" height="100" fill="blue" stroke="black" stroke-width="4"/>

</svg>

也可以在 HTML 页面引入 SVG 文件。

<img src="01\_SvgFile.svg" width="800" height="500">

#### 2）HTML 直接定义 SVG

HTML5 的新特性允许在 HTML 页面中直接定义 SVG 元素（需要学习有关SVG的一些HTML新元素）。

<svg>**元素：**

|  | **<svg>元素** |
| --- | --- |
| 作用 | 类似于canvas元素 |
| 特点 | 默认的宽度和高度分别为300px和150px，不具有任何的样式(默认情况下不显示) |
| 目的 | 表示当前使用SVG语法内容 |

<svg>**元素的属性：**

| **属性名称** | **描述** |
| --- | --- |
| width | 设置 <svg> 元素的宽度 |
| height | 设置 <svg> 元素的高度 |
| style | 设置 <svg> 元素的样式 |

<svg width="500" height="300" style="background:pink;">

<rect x="100" y="100" width="300" height="100" fill="blue" stroke="black" stroke-width="4"/>

</svg>

## 图像元素

### 1. 样式

| **属性名称** | **描述** |
| --- | --- |
| fill | 设置填充样式 |
| stroke | 设置描边样式 |
| stroke-width | 设置描边宽度 |

### 2. 矩形

<rect x="" y="" rx="" ry="" width="" height="" />

| **属性名称** | **描述** |
| --- | --- |
| x和y | 表示绘制矩形的左上角坐标值 |
| width | 表示绘制矩形的宽度 |
| height | 表示绘制矩形的高度 |
| rx和ry | 表示绘制矩形的四个角的水平圆角和垂直圆角（如果rx和ry的值分别为width/2和height/2,绘制圆形） |

**值得注意的是：**

* 默认绘制出来的效果：实心矩形效果。
* 可以实现"空心"的效果：不能实现空心效果。

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制矩形 -->

<rect x="10" y="10" width="100" height="100" />

<rect x="10" y="120" width="100" height="100" fill="white" stroke="black" />

<rect x="10" y="230" width="100" height="100" rx="10" ry="10" />

<rect x="10" y="340" width="100" height="100" rx="50" ry="50" style="fill:white;stroke:black;" />

</svg>

### 3. 圆形

<cirlcle cx="" cy="" r="" />

| **属性名称** | **描述** |
| --- | --- |
| cx和cy | 表示绘制圆形的圆心坐标值 |
| r | 表示绘制圆形的半径 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制圆形 -->

<circle cx="170" cy="60" r="50" />

<circle cx="170" cy="170" r="50" fill="white" stroke="black" />

</svg>

### 4. 椭圆

<ellipse cx="" cy="" rx="" ry="" />

| **属性名称** | **描述** |
| --- | --- |
| cx和cy | 表示绘制椭圆的圆心坐标值 |
| rx | 表示绘制椭圆的水平方向半径 |
| ry | 表示绘制椭圆的垂直方向半径 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制椭圆 -->

<ellipse cx="280" cy="70" rx="50" ry="60" />

<ellipse cx="280" cy="190" rx="50" ry="50" />

<ellipse cx="280" cy="300" rx="50" ry="30" />

</svg>

### 5. 线条

<line x1="" y1="" x2="" y2="" stroke="" />

| **属性名称** | **描述** |
| --- | --- |
| x1和y1 | 表示绘制直线的起点坐标值 |
| x2和y2 | 表示绘制直线的终点坐标值 |
| stroke | 设置绘制直线的样式(颜色) |
| stroke-width | 设置绘制直线的宽度 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制直线 -->

<line x1="350" y1="10" x2="500" y2="200" stroke="black" stroke-width="10" />

</svg>

### 6. 折线

<polyline points="" />

| **属性名称** | **描述** |
| --- | --- |
| points | 表示绘制折线的起点、折点及终点坐标值（格式：x1,y1 x2,y2 x3,y3 xn,yn） |
| stroke | 设置折线的颜色 |
| stroke-width | 设置折线的线宽 |
| fill | 设置与<svg>元素的背景色相同 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制折线 -->

<polyline points="520,20 520,200" stroke="black" stroke-width="10" />

<polyline points="550,20 550,200 700,200 700,20 540,20" stroke="black" stroke-width="20" fill="pink" />

</svg>

### 7. 多边形

<polygon points="" />

| **属性名称** | **描述** |
| --- | --- |
| points | 表示绘制多边形的所有点坐标值 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制多边形 -->

<polygon points="550,300 550,500 800,500" stroke="black" fill="pink" stroke-width="10" />

</svg>

**值得注意的是：**

* 一个HTML页面允许包含多个<svg>元素
* 一个<svg>元素允许包含多个图形元素
* SVG的图形元素基本都是起始元素
* 定义图形元素时,只定义开始元素,没有结束元素
  + 浏览器在运行页面时,并不报错
  + 浏览器解析这段元素代码时,自动补全结束元素
  + 自动补全的结束元素是不正确的

### 8. 渐变

#### 1）线性渐变

<linearGradient id="" x1="" y1="" x2="" y2=""></linearGradient>

| **属性名称** | **描述** |
| --- | --- |
| id | 标识，便于其他元素进行引用 |
| x1和y1 | 表示基准线的起点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| x2和y2 | 表示基准线的终点坐标值（值范围为 0-1, 是百分值 0%-100%） |

<stop offset="" stop-color=""/>

| **属性名称** | **描述** |
| --- | --- |
| offset | 表示设置渐变颜色的位置 |
| stop-color | 表示设置的渐变颜色 |
| stop-opacity | 表示设置渐变颜色的透明度 |

<svg width="800" height="500">

<defs>

<linearGradient id="grd" x1="0" y1="0" x2="1" y2="1">

<stop offset="0" stop-color="red" stop-opacity="0.5" />

<stop offset="0.5" stop-color="green" stop-opacity="0.5" />

<stop offset="1" stop-color="blue" stop-opacity="0.5" />

</linearGradient>

</defs>

<rect x="0" y="0" width="200" height="200" fill="url(#grd)" />

<circle cx="400" cy="400" r="100" fill="url(#grd)" />

</svg>

#### 2）射线渐变

<radialGradient id="" cx="" cy="" r="" fx="" fy=""></radialGradient>

| **属性名称** | **描述** |
| --- | --- |
| id | 标识，便于其他元素进行引用 |
| cx和cy | 表示基准线中的中心点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| fx和fy | 表示基准线中的焦点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| r | 设置其边缘位置（值范围为 0-1, 是百分值 0%-100%） |

<stop offset="" stop-color=""/>

| **属性名称** | **描述** |
| --- | --- |
| offset | 表示设置渐变颜色的位置 |
| stop-color | 表示设置的渐变颜色 |
| stop-opacity | 表示设置渐变颜色的透明度 |

<svg width="800" height="500">

<defs>

<radialGradient id="grd1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd2" cx="0" cy="0">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd3" r="1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd4" fx="1" fy="1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd5">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd6" cx="0" cy="0">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd7" r="1">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd8" fx="1" fy="1">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

</defs>

<rect x="0" y="0" width="200" height="200" fill="url(#grd1)" />

<rect x="210" y="0" width="200" height="200" fill="url(#grd2)" />

<rect x="420" y="0" width="200" height="200" fill="url(#grd3)" />

<rect x="630" y="0" width="200" height="200" fill="url(#grd4)" />

<rect x="0" y="210" width="200" height="200" fill="url(#grd5)" />

<rect x="210" y="210" width="200" height="200" fill="url(#grd6)" />

<rect x="420" y="210" width="200" height="200" fill="url(#grd7)" />

<rect x="630" y="210" width="200" height="200" fill="url(#grd8)" />

</svg>

**值得注意的是：**

* 渐变元素需要定义id属性,便于其他元素进行引用
* 渐变元素定义在元素内
  + 元素内定义：表示该元素允许重复使用

### 9. 滤镜

<filter id="myfilter"></filter>

**值得注意的是：**该元素只是滤镜的容器。

#### 高斯模糊

<feGaussianBlur in="" stdDeviation="" />

| **属性名称** | **描述** |
| --- | --- |
| in | 设置高斯模糊的样式 |
| stdDeviation | 设置模糊的程度 |

<svg width="800" height="500">

<defs>

<filter id="myfilter">

<feGaussianBlur in="SourceGraphic" stdDeviation="5" />

</filter>

</defs>

<rect x="100" y="100" width="100" height="100" filter="url(#myfilter)" />

</svg>

## Two.js库

### 1. 基础内容

Two.js支持不同的上下文环境：

* SVG（默认）
* Canvas
* WebGL

官网地址：<http://jonobr1.github.io/two.js/>

#### 如何使用 Two.js

##### 1）HTML 页面

* 引入Two.js库文件
* 定义容器元素 <div></div>

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

<!-- 1. 引入Two.js库文件 -->

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

</head>

<body>

<!--

2. 定义用于显示矢量图的容器元素

\* <div></div>元素 - 建议使用

-->

<div id="d1"></div>

</body>

</html>

##### 2）JavaScript逻辑

* 获取HTML页面的容器元素
* 通过 Two.js 库提供的 Two() 构造函数创建 Two 对象

var params = {// 创建svg时初始化的数据

width : 宽度,

height : 高度

}

var two = new Two(params);

* 将创建的 Two 对象添加到 HTML 页面容器元素内

two.appendTo(elem);

* 使用 Two.js 库提供的 API 方法绘制图形
* 调用 update() 方法进行绘制

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

<!-- 1. 引入Two.js库文件 -->

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

</head>

<body>

<!--

2. 定义用于显示矢量图的容器元素

\* <div></div>元素 - 建议使用

-->

<div id="d1"></div>

<script>

// 3. 获取HTML页面的容器元素

var elem = document.getElementById("d1");

/\*

4. 通过Two(params)构造函数来创建Two对象

\* params - 创建Two对象的初始化数据

\* Object类型,{ key : value }格式

\*/

var params = {

type : Two.Types.canvas,

width : 285, // 默认为640px

height : 200 // 默认为480px

};

var two = new Two(params);

/\*

5. 将创建的Two对象添加到页面容器元素中

\* Two.js库提供了appendTo()方法向容器添加Two对象

\* 注意 - appendTo()方法并不是jQuery的

\*/

two.appendTo(elem);

// 6. 通过调用Two对象提供的API方法进行绘制

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

// 7. 调用update()方法进行绘制

two.update();

</script>

</body>

</html>

### 2. 绘制图像

#### 1）绘制图像的方法

| **图像形状** | **方法** |
| --- | --- |
| 圆形 | makeCircle(x, y, radius) |
| 直线 | makeLine(x1, y1, x2, y2) |
| 矩形 | makeRectangle(x, y, width, height) |
| 圆角矩形 | makeRoundedRectangle(x, y, width, height, radius) |
| 椭圆 | makeEllipse(x, y, width, height) |
| 多边形 | makePolygon(ox, oy, r, sides) |
| 路径 | makePath(x1, y1, x2, y2, xN, yN, open) |
| 星形 | makeStar(ox, oy, or, ir, sides) |

#### 2）绘制图像的属性

| **属性名称** | **描述** |
| --- | --- |
| fill | 设置填充样式 |
| stroke | 设置描边样式 |
| linewidth | 设置线条宽度 |
| opacity | 设置透明度 |
| cap | 设置线条端点形状，默认为round |
| join | 设置线条交点形状，默认为round |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

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

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var params = {

type : Two.Types.canvas,

width : 285, // 默认为640px

height : 200 // 默认为480px

};

var two = new Two(params);

two.appendTo(elem);

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

// 设置绘制的圆形和矩形的样式

circle.fill = '#FF8000';

circle.stroke = 'orangered';

circle.linewidth = 5;

rect.fill = 'rgb(0, 200, 255)';

rect.opacity = 0.75;

rect.noStroke();// 设置矩形没有描边

two.update();

</script>

</body>

</html>

### 3. 图像分组

调用 Two 对象的 makeGroup() 方法对图像进行分组操作。

* makeGroup() 方法任意图形对象作为参数
* makeGroup() 方法返回分组对象

**值得注意的是：**在统一对一组图像设置后,针对不同图像进行个性化设置。

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Two.js库进行图形分组</title>

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

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var two = new Two({

width : 285,

height : 200

}).appendTo(elem);

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

var group = two.makeGroup(circle,rect);

group.fill = 'rgb(0, 200, 255)';

group.opacity = 0.75;

group.noStroke();// 设置矩形没有描边

// 针对不同的图形进行个性化样式设置

circle.stroke = 'orangered';

circle.linewidth = 5;

two.update();

</script>

</body>

</html>

### 4. 动画效果（了解）

| **方法名称** | **描述** |
| --- | --- |
| play() | 提供一组循环动画 |
| pause() | 提供中止动画效果 |
| update() | 提供更新当前绘制或设置 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Two.js库实现动画效果</title>

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

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var two = new Two({

width : 285,

height : 200

}).appendTo(elem);

// 设置圆形和矩形

var circle = two.makeCircle(-70, 0, 50);

var rect = two.makeRectangle(70, 0, 100, 100);

// 设置圆形和矩形的样式

circle.fill = '#FF8000';

rect.fill = 'rgba(0, 200, 255, 0.75)';

// 将圆形和矩形分为一组

var group = two.makeGroup(circle, rect);

// 针对这一组进行样式设置

group.translation.set(two.width / 2, two.height / 2);// 将"画布"平移水平和垂直一半距离

group.scale = 0;// 将圆形和矩形缩放为 0(不显示)

group.noStroke();// 设置圆形和矩形没有边框

// Two对象绑定update事件(方法)

two.bind('update', function(frameCount) {

// 判断如果缩放为原大小,将缩放和旋转设置为 0

if (group.scale > 0.9999) {

group.scale = group.rotation = 0;

}

// 每次执行的增量

var t = (1 - group.scale) \* 0.125;

// 每次执行后缩放值进行累加

group.scale += t;

// 每次执行后旋转至进行累加

group.rotation += t \* 4 \* Math.PI;

}).play();// 开始无限循环动画

</script>

</body>

</html>

# HTML5 第五天笔记

## 百度地图

### 1. 如何使用

#### 1）HTML 页面

* 引入百度地图提供的JS文件

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

* 定义容器元素<div></div>

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图如何使用</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<!--

引入百度地图提供的JS文件

\* 是在线提供的JS文件 - 必须可以联网

http://api.map.baidu.com/api?v=2.0&ak=您的密钥

\* v=2.0 - 使用的百度地图的版本

\* ak=您的密钥 - 传递当前应用的秘钥

\* 只有秘钥正确,正常地使用百度地图的功能

-->

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

<!-- 定义一个容器元素 - 用于显示百度地图 -->

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

</body>

</html>

#### 2）JavaScript 逻辑

* 创建百度地图对象

var map = new BMap.Map(容器元素id);

* 初始化百度地图

map.centerAndZoom(中心点坐标,显示级别)

如下述示例代码：

/\*

1. 通过Map()构造函数,创建百度地图对象

var map = new BMap.Map(容器元素id属性值);

\*/

var map = new BMap.Map("allmap");

/\*

2. (必要)初始化百度地图 - 设置中心点和显示级别

map.centerAndZoom(center,zoom)

\* center - 设置百度地图当前的中心点坐标

\* Point类型,zoom参数必须设置

\* String类型,zoom参数可有可无

\* 如果不设置zoom,自动匹配最佳显示级别

\* zoom - 设置百度地图当前的显示级别

\* 一般情况下,值范围为 3-19

\* 高清地图(移动端),值范围为 3-18

\*/

map.centerAndZoom("北京",12);

### 2. 核心类

#### Map类

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Map(container) | container：页面容器元素id | 创建地图实例对象 |

**配置方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| enableScrollWheelZoom() |  | 开启鼠标滚轮放大或缩小地图的显示级别 |
| disableScrollWheelZoom() |  | 禁用鼠标滚轮功能 |
| enableDragging() |  | 启用鼠标拖拽功能 |
| disableDragging() |  | 禁用鼠标拖拽功能 |

**地图状态方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| getCenter() |  | 返回地图当前中心点(Point类型) |
| getZoom() |  | 返回地图当前缩放级别(Number类型) |

**修改地图状态方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| centerAndZoom(center,zoom) | center：设置地图的中心点坐标；zoom：设置地图的显示级别 | 初始化地图中心点和级别 |
| setCenter(center) | center：设置地图的中心点坐标 | 设置地图中心点(Point |
| setCurrentCity(city) | city：设置地图城市 | 设置地图城市 |
| setZoom(zoom) | zoom：设置地图的显示级别 | 设置显示级别 |

**控件方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| addControl(control) |  | 将控件添加到地图 |
| removeControl(control) |  | 从地图中移除控件 |

**覆盖物方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| addOverlay(overlay) |  | 将覆盖物添加到地图中 |
| removeOverlay(overlay) |  | 从地图中移除覆盖物 |
| openInfoWindow(InfoWindow, Point) | InfoWindow：设置信息窗口；Point：在指定点打开 | 设置信息窗口 |

### 3. 控件类

#### ScaleControl类

表示比例尺控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| ScaleControl(ScaleControlOptions) | - | 创建地图比例尺对象 |

#### ScaleControlOptions类

表示ScaleControl构造函数的可选参数。

**选项**

* anchor：设置当前控件显示的位置。
  + BMAP\_ANCHOR\_TOP\_LEFT：左上角
  + BMAP\_ANCHOR\_BOTTOM\_LEFT：左下角
  + BMAP\_ANCHOR\_TOP\_RIGHT：右上角
  + BMAP\_ANCHOR\_BOTTOM\_RIGHT：右下角

#### NavigationControl类

表示地图的平移缩放控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| NavigationControl(NavigationControlOptions) | - | 创建地图平移缩放对象 |

#### MapTypeControl类

负责切换地图类型的控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| MapTypeControl(MapTypeControlOptions) | - | 创建地图类型对象 |

#### OverviewMapControl类

表示缩略地图控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| OverviewMapControl(OverviewMapControlOptions) | - | 创建缩略地图对象 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图的控件类</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

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

<script>

// 1. 创建地图对象

var map = new BMap.Map("allmap");

// 2. 初始化地图

map.centerAndZoom("北京",13);

map.enableScrollWheelZoom();

/\*

3. 添加比例尺控件

\* 创建比例尺控件对象

\* 将比例尺控件添加到地图中

\*/

var scale = new BMap.ScaleControl({

anchor: BMAP\_ANCHOR\_TOP\_RIGHT

});

map.addControl(scale);

/\*

4. 添加平移缩放控件

\* 创建平移缩放控件对象

\* 将平移缩放控件添加到地图中

\*/

var nav = new BMap.NavigationControl({

anchor: BMAP\_ANCHOR\_TOP\_RIGHT

});

map.addControl(nav);

// 5. 添加地图类型切换控件

var type = new BMap.MapTypeControl({

anchor: BMAP\_ANCHOR\_TOP\_LEFT

});

map.addControl(type);

// 6. 添加缩略地图控件

var overview = new BMap.OverviewMapControl({

isOpen : true//表示是否自动打开

});

map.addControl(overview);

</script>

</body>

</html>

### 4. 覆盖物类

#### Marker类

表示地图上一个图像标注。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Marker(point) | - | 创建标注对象 |

#### InfoWindow类

表示地图上包含信息的窗口。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| InfoWindow(content) | content：设置当前点的地址信息内容 | 创建信息的窗口对象 |

**方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| setWidth(width) |  | 设置信息窗口的宽度 |
| setHeight(height) |  | 设置信息窗口的高度 |
| setTitle(title) |  | 设置信息窗口的标题 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>为百度地图进行标注</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

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

<script>

var map = new BMap.Map("allmap");

var point = new BMap.Point(116.404, 39.915);

map.centerAndZoom(point, 15);

/\*

如何进行标注

\* 创建标注对象

\* 将标注添加到地图中

\*/

var marker = new BMap.Marker(point);

map.addOverlay(marker);

marker.setAnimation(BMAP\_ANIMATION\_BOUNCE);

</script>

</body>

</html>

### 5. 服务类

#### Geocoder类

用于获取用户的地址解析。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Geocoder() | content：设置当前点的地址信息内容 | 创建地址解析对象 |

**方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| getPoint(address,callback,city) | address：要解析的地址内容；callback：回调函数（解析address地址成功的话,回调函数具有一个形参）；city：当前地址所在的城市名称 | 对指定的地址进行解析 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图案例</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

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

<script>

// 1. 创建地图对象

var map = new BMap.Map("allmap");

// 2. 初始化地图

//map.centerAndZoom("北京",13);

// 3. 启用鼠标的滚轮功能

map.enableScrollWheelZoom();

/\*

4. 根据给定的地址获取Point的经度和纬度

var geocoder = new BMap.Geocoder();

geocoder.getPoint(address,function(point){},city);

\*/

var geocoder = new BMap.Geocoder();

geocoder.getPoint("北京市海淀区万寿路西街2号",function(point){

// 5. 根据Point进行标注

var marker = new BMap.Marker(point);

map.addOverlay(marker);

// 6. 重新设置中心点位置

map.centerAndZoom(point,16);

// 7. 设置信息窗口

var opts = {

width : 200,

height: 100,

title : "文博大厦"

}

var info = new BMap.InfoWindow("地址:北京市海淀区万寿路西街2号",opts);

marker.addEventListener("click",function(){

map.openInfoWindow(info,point);

});

},"北京");

</script>

</body>

</html>

## 拖放 API

允许用户在一个元素上点击并按住鼠标按钮，拖动它到别的位置，然后松开鼠标按钮将元素放到那儿。

在拖动操作过程中，被拖动元素会以半透明形式展现，并跟随鼠标指针移动。放置元素的位置可能会在不同的应用内。

### 1. 源元素事件

**所谓源元素**就是被拖动的元素。

| **事件名称** | **作用** | **触发次数** |
| --- | --- | --- |
| dragstart事件 | 开始拖动源元素时被触发 | 只被触发一次 |
| drag事件 | 拖动的过程中，实时被触发 | 被触发多次 |
| dragend事件 | 结束拖动源元素时被触发 | 只被触发一次 |

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<script>

// 获取HTML页面中的源元素<img>

var img = document.getElementById("img");

// 为源元素绑定拖动事件

img.addEventListener("dragstart",myDragstart);

img.addEventListener("drag",myDrag);

img.addEventListener("dragend",myDragend);

// 定义事件的处理函数

function myDragstart(event){

console.log("开始拖动啦...");

}

function myDrag(event){

console.log("拖动过程中...");

}

function myDragend(event){

console.log("结束拖动啦...");

}

</script>

### 2. 目标元素事件

**所谓目标元素**就是投放到的元素。

| **事件名称** | **作用** | **触发次数** |
| --- | --- | --- |
| dragenter事件 | 当源元素到达目标元素时被触发 | 只被触发一次 |
| dragover事件 | 当源元素到达目标元素时被触发 | 被触发多次。阻止默认行为，触发drop事件 |
| drop事件 | 当源元素投放在目标元素时被触发 | 默认该事件不会被触发 |
| dragleave事件 | 当源元素离开目标元素时被触发 | 只被触发一次 |

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<div id="d2"></div>

<script>

// 获取目标元素

var d2 = document.getElementById("d2");

// 为目标元素绑定事件

d2.addEventListener("dragenter",myDragenter);

d2.addEventListener("dragover",myDragover);

d2.addEventListener("drop",myDrop);

d2.addEventListener("dragleave",myDragleave);

// 定义事件的处理函数

function myDragenter(event){

//event.preventDefault();

console.log("大爷,你来啦...");

}

function myDragover(event){

event.preventDefault();

console.log("大爷,又来啦...");

}

function myDrop(event){

//event.preventDefault();

console.log("大爷,别走啦...");

}

function myDragleave(event){

//event.preventDefault();

console.log("大爷,要走啦...");

}

</script>

### 3. dataTransfer对象

该对象已被集成在 event 对象中，起到剪切板的功能。

| **方法名称** | **作用** | **场景** |
| --- | --- | --- |
| setData(type,data) | 存储数据 | 在源元素事件中 |
| getData(type) | 取出数据 | 在目标元素事件中 |
| clearData() | 清除剪切板中所有数据 |  |

参数：

* type：指定当前存储数据的类型(标识)。
* data：需要中转的数据内容。

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<div id="d2"></div>

<script>

// 1. 获取源元素和目标元素

var img = document.getElementById("img");

var d2 = document.getElementById("d2");

/\*

2. 为源元素和目标元素绑定事件

\* 绑定源元素事件的目的:

\* 通过dataTransfer对象获取源元素的关键数据

\* 只需绑定dragstart事件即可

\* 绑定目标元素事件的目的:

\* 通过dataTransfer对象使用源元素的关键数据

\* 需要绑定dragover和drop事件

\*/

img.addEventListener("dragstart",function(event){

// 通过dataTransfer对象,存储源元素的关键数据

event.dataTransfer.setData("text",img.src);

});

d2.addEventListener("dragover",function(event){

event.preventDefault();

});

d2.addEventListener("drop",function(event){

// 通过dataTransfer对象,使用源元素的关键数据

var src = event.dataTransfer.getData("text");

//d2.innerHTML = "<img src='"+src+"' width='256'>";

var newImg = document.createElement("img");

newImg.src = src;

newImg.width = "256";

newImg.id = "newImg";

d2.appendChild(newImg);

img.parentNode.removeChild(img);

});

</script>

# HTML5 第六天笔记

## Web Workers

### 1. 基础内容

#### 1）什么是 Web Workers

能够把 JavaScript 计算委托给后台线程，通过允许这些活动以防止使交互型事件变得缓慢。

上句话是 MDN 对象 Web Workers 的简单描述。

Web Workers 是可以在后台（页面端）运行的任务，它能够被轻松的创建，还能向它的创建者发送消息。

Web Workers 的三大主要特征：能够长时间运行（响应），理想的启动性能以及理想的内存消耗。

#### 2）两种 Web Workers

Web Workers 可以分为两种类型：Dedicated Web Worker（专用线程）和 Shared Web Worker（共享线程）。

##### a. Dedicated Web Worker

目前绝大多数的应用场景所使用的是 Dedicated Web Worker（专用线程）。Dedicated Web Worker随着 HTML 页面关闭而被终止，这就意味着 Dedicated Web Worker 只能被创建它的 HTML 页面所访问。

在 JavaScript 代码中，Work 类型代表 Dedicated Web Worker。

##### b. Shared Web Worker

与 Dedicated Web Worker（专用线程）相比，Shared Web Worker 一般用于一些特定情况，可以为多个 HTML 页面服务，即可以被多个 HTML 页面所访问。与之对应的，只有将与之相关的所有 HTML 页面都关闭才能被终止。

在 JavaScript 代码中，SharedWorker 类型代表 Shared Web Worker。

#### 3）Web Workers 的限制

Web Workers 在具体使用中，具有以下几种限制：

* Web Workers 无法访问 DOM 节点；
* Web Workers 无法访问全局变量或是全局函数；
* Web Workers 无法调用 alert() 或者 confirm() 之类的函数；
* Web Workers 无法访问 window、document 之类的浏览器全局变量；

Web Workers 可以在它的作用域内访问 navigator 对象。它含有如下能够识别浏览器的字符串，就像在普通脚本中做的那样：

* appName
* appVersion
* platform
* userAgent

#### 4）Web Workers 的使用场景

* Web Workers 线程能够在不干扰 UI 的情况下执行一些计算的任务。
* Web Workers 线程能够使用 XMLHttpRequest 对象与服务器端进行异步交互。

#### 5）参考资料

* [Web Workers - W3C](http://www.w3.org/TR/workers/)
* [使用 Web Workers](https://developer.mozilla.org/zh-CN/docs/Web/API/Web_Workers_API/Using_web_workers)
* [Microsoft Web Workers](https://msdn.microsoft.com/library/hh673568(v=vs.85).aspx)

### 2. Worker API

#### 1）Worker 构造函数

Worker(DOMString scriptURL)

通过该构造函数可以直接创建 Worker 对象，并传递 Worker 文件的路径（scriptURL）。具体用法如下：

var worker = new Worker(my\_task.js);

通过 Worker 的构造函数得到 Worker 对象，在 HTML 页面与 Worker 文件之间创建通信。

**值得注意的是：**指定的 Worker 文件路径必须遵循同源策略。

#### 2）监听通信消息机制

worker.onmessage = function(event){

逻辑编写在这里...

}

当 Worker 传递消息给 HTML 页面（或 HTML 页面传递消息给 Worker）时，可以通过 onmessage 事件进行监听并且捕获。

传递的消息存储在 onmessage 事件处理函数的参数 data 中。

worker.onmessage = function(event) {

console.log("Worker's message: " + event.data);

};

#### 3）发送通信消息方法

postMessage(JSObject message)

HTML 页面通过该方法向 Worker 的内部作用域内传递消息。该方法接收一个单独的参数，即要传递给 Worker 的数据。

message 参数用于传递给 Worker 的数据，该数据将包含于传递给 onmessage 处理函数的事件对象中的 data 字段内。

// 普通数据

worker.postMessage("ali");

// JSON格式的数据

worker.postMessage({"cmd": "init", "timestamp": Date.now()});

#### 4）终止 Worker 通信

terminate()

该方法用于立即终止 Worker。该方法不会给 Worker 留下任何完成操作的机会；就是简单的立即停止。

#### 5）监听通信错误机制

worker.onerror = function(event){

错误处理编写在这里...

}

错误信息对象包含三个属性：

* message：一个可读性良好的错误信息。
* filename：产生错误的脚本文件名。
* lineno：发生错误时所在的脚本文件行号。

### 3. 双向通信

所谓“单向通信”就是指 HTML 页面与 Worker 建立通信后，只是 Worker 向 HTML 页面传递消息。

所谓“双向通信”就是指 HTML 页面与 Worker 之间可以相互传递消息的一种方式。

要建立双向通信，HTML 页面和 Worker 线程都要侦听 onmessage 事件。

* 首先，该脚本创建 worker 线程。

var echo = new Worker('echo.js');

echo.onmessage = function(e) {

alert(e.data);

}

* 当用户单击提交按钮时，脚本会将两条信息以 JavaScript 对象文本的形式传递给 Worker。

<script>

window.onload = function() {

var echoForm = document.getElementById('echoForm');

echoForm.addEventListener('submit', function(e) {

echo.postMessage({

message : e.target.message.value,

timeout : e.target.timeout.value

});

e.preventDefault();

}, false);

}

</script>

<form id="echoForm">

<p>Echo the following message after a delay.</p>

<input type="text" name="message" value="Input message here."/><br/>

<input type="number" name="timeout" max="10" value="2"/> seconds.<br/>

<button type="submit">Send Message</button>

</form>

* 最后，Worker 侦听消息，并在指定的超时间隔之后将其返回。

onmessage = function(e){

setTimeout(function(){

postMessage(e.data.message);

},

e.data.timeout \* 1000);

}

## Web 存储

在开发 Web 应用时，开发人员有时需要在本地存储数据。当前浏览器支持 cookie 存储，但其大小有 4KB 的限制。

HTML5 中新引入了 Web Storage 机制，通过使用键值对在客户端保存数据，并且提供了更大容量的存储空间。

HTML5 中的 Web 存储对象有两种类型：

* sessionStorage 对象负责存储一个会话的数据。如果用户关闭了页面或浏览器，则会销毁数据。
* localStorage 对象负责存储没有到期的数据。当 Web 页面或浏览器关闭时，仍会保持数据的存储，当然这还取决于为此用户的浏览器设置的存储量。

这两种存储对象具有相同的方法和属性。

**值得注意的是：**Web 存储并不比 cookies 安全。所以不要在客户端存储敏感信息，比如密码或信用卡信息。

官方规范资料地址：<https://html.spec.whatwg.org/multipage/webstorage.html>

### 1. sessionStorge

#### 1）浏览器支持性检查

//sessionStorage

if(window.sessionStorage){

alert(“support sessionStorage”);

}else{

alert(“not support sessionStorage”);

// 不支持 sessionStorage

}

#### 2）sessionStorage的方法

| **方法名称** | **描述** |
| --- | --- |
| setItem(key, value) | 为 Web 存储对象添加一个键/值对，供以后使用。该值可以是任何的数据类型：字符串、数值、数组等。 |
| getItem(key) | 基于起初用来存储它的这个键检索值。 |
| clear() | 从此 Web 存储对象清除所有的键/值对。 |
| removeItem(key) | 基于某个键从此 Web 存储对象清除特定的键/值对。 |
| key(n) | 检索 key[n] 的值。 |

##### a. 向 Web 存储对象添加键/值对

* 使用 setItem() 方法

sessionStorage.setItem('myKey', 'myValue');

* 使用 Web 存储对象来直接设置此键的值

sessionStorage.myKey = 'myValue';

##### b. 从 Web 存储对象中检索键/值对

* 使用 getItem() 方法

sessionStorage.getItem('myKey');

* 使用 Web 存储对象来直接获取此键的值

sessionStorage.myKey;

##### c. 删除所有键/值对

sessionStorage.clear();

##### d. 删除单个键/值对

sessionStorage.removeItem('myKey');

#### 示例代码

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>sessionStorage</title>

</head>

<body>

<input type="text" id="data">

<input type="button" id="save" value="保存">

<input type="button" id="read" value="读取">

<input type="button" id="dele" value="删除">

<script>

var save = document.getElementById("save");

save.onclick = function(){

// 将用户输入的数据,保存到sessionStorage中

// 1. 获取用户输入的数据

var value = document.getElementById("data").value;

// 2. 如何生成存储数据所使用的KEY

var key = new Date().getTime();// 时间戳

// 3. 存储数据

window.sessionStorage.setItem(key,value);

}

var read = document.getElementById("read");

read.onclick = function(){

// 读取sessionSotorage中所有的数据

// 1. 获取当前sessionStorage中所有数据的个数

var count = window.sessionStorage.length;

// 2. 进行遍历

for(var i=0;i<count;i++){

// 3. 根据索引值得到key

var key = window.sessionStorage.key(i);

// 4. 根据key值得到value

var value = window.sessionStorage.getItem(key);

// 5. 进行测试

console.log(key + " : " + value);

}

}

var dele = document.getElementById("dele");

dele.onclick = function(){

// removeItem()

//window.sessionStorage.removeItem(1452666425707);

window.sessionStorage.clear();

}

</script>

</body>

</html>

### 2. localStorage

#### 1）浏览器支持性检查

//localStorage

if(window.localStorage){

alert(“support localStorage”);

}else{

alert(“not support localStorage”);

// 不支持 localStorage

}

#### 2）localStorage的方法

| **方法名称** | **描述** |
| --- | --- |
| setItem(key, value) | 为 Web 存储对象添加一个键/值对，供以后使用。该值可以是任何的数据类型：字符串、数值、数组等。 |
| getItem(key) | 基于起初用来存储它的这个键检索值。 |
| clear() | 从此 Web 存储对象清除所有的键/值对。 |
| removeItem(key) | 基于某个键从此 Web 存储对象清除特定的键/值对。 |
| key(n) | 检索 key[n] 的值。 |

##### a. 向 Web 存储对象添加键/值对

* 使用 setItem() 方法

localStorage.setItem('myKey', 'myValue');

* 使用 Web 存储对象来直接设置此键的值

localStorage.myKey = 'myValue';

##### b. 从 Web 存储对象中检索键/值对

* 使用 getItem() 方法

localStorage.getItem('myKey');

* 使用 Web 存储对象来直接获取此键的值

localStorage.myKey;

##### c. 删除所有键/值对

localStorage.clear();

##### d. 删除单个键/值对

localStorage.removeItem('myKey');

#### 3）localStorage的事件

如果想在存储成功或修改存储的值时执行一些操作，可以用 Web Storage 接口提供的事件。

// 显示存储事件的相关内容

function handleStorageEvent(e) {

document.write(“key” + e.key + “oldValue” + e.oldValue + “newValue” + e.newValue);

}

// 添加存储事件监听

window.addEventListener(“storage”, handleStorageEvent, false);

| **事件参数** | **描述** |
| --- | --- |
| key | 发生改变的键 |
| oldValue | 键改变之前的值 |
| newValue | 键改变之后的值 |
| url | 触发存储事件的页面 url |

## Web Sockets

### 1. 什么是 Web Sockets

#### 1）什么是 Web Sockets

Web Sockets 技术使得浏览器直接与服务器端的程序通过 socket 可以实时的推送或者获取信息的通讯方式成为可能。

在 HTML5 之前实现浏览器与服务器端实时通信的技术如下：

* 轮询：原理简单易懂，就是客户端通过一定的时间间隔以频繁请求的方式向服务器发送请求，来保持客户端和服务器端的数据同步。
* Flash：Flash 通过自己的 Socket 实现完成数据交换，再利用 Flash 暴露出相应的接口为 JavaScript 调用，从而达到实时传输目的。

#### 2）Web Sockets 的优势与劣势

##### a. Web Sockets 的优势

* Web Sockets 提供强大的、双向、低延迟和易于处理的错误。
* 它没有很多连接，比如：Comet 长轮询。

##### b. Web Sockets 的劣势

* Web Sockets 是 HTML5 的新规范，并不是所有浏览器都支持。
* 无请求作用域。由于 Web Sockets 是一个 TCP 套接字，而不是一个 HTTP 请求，因此无法轻松使用请求作用域服务。

#### 3）Web Sockets 的实现原理

下图展示了如何使用 WebSockets 进行通信：

1. HTTP 握手被发送到带有特定标头的服务器。
2. 在 JavaScript 的服务器或客户端上提供某种类型的套接字。
3. 使用该套接字来通过事件处理器异步接收数据。

### 2. Web Sockets API

#### 1）Web Sockets 的构造函数

var websocket = new WebSocket(url, protocols);

| **参数名称** | **描述** |
| --- | --- |
| url | 表示要连接的URL,这个URL应该是响应WebSocket的地址。 |
| protocols | 单个协议名称或字符串数组,默认为空字符串。 |

**值得注意的是：**Web Sockets 的请求地址（url）必须是 ws:// 或 wss://。

var exampleSocket = new WebSocket("ws://www.example.com/socketserver");

#### 2）Web Sockets 的方法

| **方法名称** | **参数** | **描述** |
| --- | --- | --- |
| send(data) | data表示发送的请求数据。 | 通过 WebSocket 连接向服务器发送数据。 |
| close() |  | 关闭WebSocket连接或停止正在进行的连接请求。 |

exampleSocket.send("Here's some text that the server is urgently awaiting!");

#### 3）Web Sockets 的属性

Web Sockets 的属性主要使用的就是 readyState，该属性表示连接的当前状态。

| **常量名** | **值** | **描述** |
| --- | --- | --- |
| CONNECTING | 0 | 连接还没开启。 |
| OPEN | 1 | 连接已开启并准备好进行通信。 |
| CLOSING | 2 | 连接正在关闭的过程中。 |
| CLOSED | 3 | 连接已经关闭，或者连接无法建立。 |

#### 4）Web Sockets 的事件

| **事件名称** | **描述** |
| --- | --- |
| onopen | 用于监听 Web Sockets 打开的事件。 |
| onmessage | 用于监听 Web Sockets 服务器端传递消息的事件。 |
| onerror | 用于监听 Web Sockets 发生错误的事件。 |
| onclose | 用于监听 Web Sockets 通信关闭的事件。 |

#### 5）Web Sockets 使用步骤

var ws = new WebSocket('ws://127.0.0.1:8080/async');

ws.onopen = function() {

// called when connection is opened

};

ws.onerror = function(e) {

// called in case of error, when connection is broken in example

};

ws.onclose = function() {

// called when connexion is closed

};

ws.onmessage = function(msg) {

// called when the server sends a message to the client.

// msg.data contains the message.

};

// Here is how to send some data to the server

ws.send('some data');

// To close the socket:

ws.close();

### 3. Web Sockets 案例

* 客户端代码示例：

// 假设服务端ip为127.0.0.1

ws = new WebSocket("ws://127.0.0.1:2346");

ws.onopen = function() {

alert("连接成功");

ws.send('tom');

alert("给服务端发送一个字符串：tom");

};

ws.onmessage = function(e) {

alert("收到服务端的消息：" + e.data);

};

* 服务器端代码示例：

这里利用了 PHP 的第三方框架 Workerman 实现 Web Sockets 的服务器端逻辑。

<?php

use Workerman\Worker;

require\_once '/Workerman/Autoloader.php';

// 创建一个Worker监听2346端口，使用websocket协议通讯

$ws\_worker = new Worker("websocket://0.0.0.0:2346");

// 启动4个进程对外提供服务

$ws\_worker->count = 4;

// 当收到客户端发来的数据后返回hello $data给客户端

$ws\_worker->onMessage = function($connection, $data)

{

// 向客户端发送hello $data

$connection->send('hello ' . $data);

};

// 运行worker

Worker::runAll();

?>

# BOOTSTRAP第一天笔记

## 响应式页面

### 响应式页面概述

#### 1. 什么是响应式页面

响应式网页（RWD，Responsive Web Design），2010年5月由Ethan Marcotte提出。简而言之，就是一个网站能够针对多个终端提供不同的显示样式，而不是针对每个终端制作一个特定的版本。

由于智能手机和平板电脑的兴起，网站页面的制作不仅要满足PC端浏览器的显示，还要满足智能手机浏览器的显示或者平板电脑浏览器的显示。（**PC端、智能手机或平板电脑的浏览器之间的区别，主要集中在分辨率的不同。**）

#### 2. 为什么需要响应式页面

想要同时满足PC端、智能手机和平板电脑浏览器的正常显示，实际上有以下两种实现方式：

a. 分别针对PC端、智能手机和平板电脑的浏览器制作特定的页面。例如 <https://www.tmall.com/>

* 优点：用户体验好，适合复杂页面。
* 缺点：工作量大，不够灵活。

b. 制作一个页面同时满足PC端、智能手机和平板电脑的浏览器显示。例如 <http://www.intel.cn/>

* 优点：比较灵活，一个页面适应不同终端。
* 缺点：页面加载时间长，效率降低。

如果PC端显示的页面与智能手机或平板电脑显示的页面不同时，建议选择 a 方案。比如智能手机端的可以利用 GPS 定位当前位置等不同于PC端的功能。但大多数网站的页面，无论是PC端还是智能手机或平板电脑显示的内容基本相同，建议选择 b 方案。

#### 3. 响应式页面的构成

目前实现响应式页面可以通过以下两种方式：

a. 使用框架实现，例如 [Bootstrap](https://getbootstrap.com) 等。这种方式开发响应式页面更快速、简单，但由于CSS样式是集成式的，导致页面样式相似，没有特点。

b. 手写方式实现。这种方式开发响应式页面速度比较慢、代码多，但可以根据网站特点定制CSS样式。

手写方式需要使用到以下几种技术：

* 流式网格布局
* CSS3 的媒体查询
* 相对单位替换绝对单位

### 测试响应式页面

#### 1. 如何测试响应式页面

测试响应式页面可以通过以下三种方式：

* 通过真实设备进行测试。这种方式进行测试效果最好，但也是最复杂，并且需要不同分辨率的设备，成本较高。
* 通过第三方模拟器进行测试。这种方式一般都是针对智能手机终端研发工具自带，需要在电脑安装不同开发工具，比较麻烦，并且性能较差。或者通过在线测试网站进行测试，测试周期较长。
* 通过浏览器自带的设备模拟器进行测试。这种方式是最方便的，主流浏览器都自带了设备模拟器，并且测试效果较好。

#### 2. 使用浏览器测试

以 Chrome 浏览器为例。

打开 Chrome 浏览器，点击右上角的菜单，选择“更多工具”中的“开发者工具”。（**Windows操作系统版本的 Chrome 浏览器可以通过 F12 快捷键直接打开。**）

**值得注意的是：**

* 每次更换测试设备时，需要重新刷新页面显示。
* 有些功能无法正常测试，例如电话薄、摄像头等。

#### 3. 使用第三方工具测试

#### 4. 使用真实设备测试

### 编写响应式页面

#### 1. viewport元元素

##### 1）什么是 viewport ？

viewport 被翻译为“视区”，表示页面的可视区域。通过 HTML 页面的 meta 元元素进行设置，是响应式页面设计的必备内容。

移动端浏览器在一个通常比屏幕更宽的虚拟“窗口”（视口）中渲染页面，从而无序将所有页面都压缩进小屏幕里（那样会把很多没有针对移动端进行优化的站点打乱）。用户可以通过平移和缩放来浏览页面的不同区域。

通过设置 viewport 元元素可以允许开发者控制显示的尺寸及比例。如今大部分移动端浏览器都支持 viewport 元元素，尽管它**不是 web 标准的一部分**。

##### 2）设置 viewport

一个典型的针对移动端优化的站点包含类似下面的内容：

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">

viewport 元元素的属性如下表：

| **属性名称** | **说明** | **可选值** |
| --- | --- | --- |
| width | 设置窗口的宽度 | 像素值或device-width（表示100%时屏幕宽度的像素值） |
| height | 设置窗口的高度 | 一般不指定 |
| initial-scale | 设置初始时的缩放倍率 |  |
| minimum-scale | 设置允许的最小缩放倍率 |  |
| maximum-scale | 设置允许的最大缩放倍率 |  |
| user-scalable | 是否允许用户手动缩放 | 1或yes表示允许；0或no表示不允许 |

meta 元元素建议设定在 head 元素中的最前面。

#### 2. 使用相对单位

##### 1）使用相对宽度值

通过学习 CSS 知道设置宽度可以使用 px 像素值和百分值。使用像素值设置宽度是固定值，不能随着屏幕大小变化而变化。实现响应式页面的宽度只能使用百分值或者auto。

###### 如何从固定布局修改为百分比布局

在《无懈可击的 Web 设计》一书中，提供了一个简易可行的公式：

目标元素宽度 / 上下文元素宽度 = 百分比宽度

以下代码是使用像素值设置 HTML 页面元素：

div {

width : 500px;

height : auto;

border : 1px solid black;

}

p {

width : 480px;

padding-left: 10px;

}

将上述像素值修改为百分比：

div {

width : 60%;

height : auto;

border : 1px solid black;

}

p {

width : 96%; /\* 480/500=96% \*/

padding-left: 2%; /\* 10/500=2% \*/

}

##### 2）用 em 替换 px

之前使用 em 替代 px 设置字体，主要是为了解决老版本的 IE 浏览器无法缩放以像素值为单位的文字。

目前使用 em 替代 px 设置字体，主要的目的具有以下两个：

* 实现老版本 IE 浏览器可以缩放文字。
* 更好地实现响应式页面，使页面效果更好。

em 的实际大小事相对于上下文的字体大小而言的，例如 1em 表示与上下文字体相同大小。em 单位设置字体同样可以通过以下公式进行计算：

目标元素尺寸 / 上下文元素尺寸 = 百分比尺寸

**值得注意的是：**浏览器显示默认文字大小都是 16 像素。

#### 3. 流式布局

##### 1）什么是流式布局

所谓流式布局，就是页面元素的宽度按照屏幕进行适配调整。简单来说，就是 HTML 页面中的元素会根据分辨率的不同而变化大小，但位置并不会有任何变化。

这种布局的主要问题就是，如果屏幕尺度跨度太大，那么在相对其原始设计而言过小或过大的屏幕上不能正常显示。

##### 2）创建流式布局

流式布局通常被分为 12 列。将 HTML 页面的宽度设置为 100%，这样 HTML 页面会根据浏览器窗口大小的变化而自动伸缩。

计算每列所占的百分比：100% / 12 列 = 8.33%

根据以上的计算结果，设置 CSS 的 class：

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

* 将所有列设置为浮动（一般为左浮动）

[class\*="col-"] {

float: left;

}

将 HTML 页面中所有的元素 box-sizing 设置为 border-box。

当 box-sizing 的值为 content-box（默认值），标准盒模型。

尺寸计算公式：width = 内容的宽度，height = 内容的高度。

宽度和高度都不包含内容的边框（border）和内边距（padding）。

div {

width : 300px;

height : 200px;

border : 10px solid black;

background : deepskyblue;

}

当 box-sizing 的值为 border-box，IE 怪异模式（Quirks mode）使用的 盒模型 。

尺寸计算公式：width = border + padding + 内容的宽度，height = border + padding + 内容的高度。

**注意：**外边距和边框将会包括在盒子中。

div {

box-sizing: border-box;

width : 300px;

height : 200px;

border : 10px solid black;

background : deepskyblue;

}

根据上述流式布局的内容，实现以下案例：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

html {

font-family: "Lucida Sans", sans-serif;

}

header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

aside ul {

list-style-type: none;

margin: 0;

padding: 0;

}

aside li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0, 0.24);

}

aside li:hover {

background-color: #0099cc;

}

</style>

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

#### 4. 响应式图片

实现图片的显示根据浏览器窗口大小的变化而变化，相对比较简单。只需要将图片的 width 设置为 100% 即可。

img {

width: 100%;

height: auto;

}

**但这样设置后，图片的显示可能会比实际大小更大。解决这个问题可以利用 max-width 替代 width 属性即可。**

img {

max-width: 100%;

height: auto;

}

这样设置后，图片的显示永远不会比实际大小更大了。

#### 5. CSS媒体查询

想要满足更复杂的情况，需要使用 CSS3 中提供的媒体查询进行解决。

媒体查询 包含了一个媒体类型和至少一个使用如宽度、高度和颜色等媒体属性来限制样式表范围的表达式。CSS3加入的媒体查询使得无需修改内容便可以使样式应用于某些特定的设备范围。

##### 媒体查询语法

媒体查询 包含 一个媒体类型、逻辑操作符和一个或多个媒体属性。如果媒体查询中的媒体类型与文档要展示的设备相符则查询结果为真，并且媒体查询中的所有表达式为真。

媒体查询具体的写法有以下两种方式：

* link 元素中的 CSS 媒体查询。

<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">

这种方式需要针对不同设备编写不同的 CSS 样式表。

* 样式表中的 CSS 媒体查询。

@media mediatype and|not|only (media feature) {

CSS-Code;

}

这种方式只需要编写一个 CSS 样式表，在样式表中根据不同的设备设置不同的样式。（**g更常用**）

##### 媒体类型

| **值** | **描述** |
| --- | --- |
| all | 用于所有设备 |
| print | 用于打印机和打印预览 |
| screen | 用于电脑屏幕，平板电脑，智能手机等 |
| speech | 用于屏幕阅读器等发声设备 |

媒体类型更常用的值为 screen。

##### 逻辑操作符

| **值** | **描述** |
| --- | --- |
| and | 用于把多个 媒体属性 组合起来，合并到同一条媒体查询中。只有当每个属性都为真时，这条查询的结果才为真. |
| not | 用于对一条媒体查询的结果进行取反。 |
| only | 表示仅在媒体查询匹配成功的情况下应用指定样式。可以通过它让选中的样式在老式浏览器中不被应用。 |

也可以将多个媒体查询以逗号分隔放在一起；只要其中任何一个为真，整个媒体语句就返回真。相当于 or 操作符。

| **值** | **描述** |
| --- | --- |
| width | 定义输出设备中的页面可见区域宽度 |
| max-width | 定义输出设备中的页面最大可见区域宽度 |
| max-device-width | 定义输出设备的屏幕最大可见宽度 |
| min-width | 定义输出设备中的页面最小可见区域宽度 |
| min-device-width | 定义输出设备的屏幕最小可见宽度 |

##### 媒体查询示例

###### link 元素中的 CSS 媒体查询

* HTML 代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>LINK元素方式的媒体查询</title>

<meta name="viewport" content="width=device-width,initial-scale=1"/>

<link rel="stylesheet" href="css/main.css">

<link rel="stylesheet" media="screen and (min-width:768px)" href="css/pc.css">

<link rel="stylesheet" media="screen and (max-width:768px)" href="css/phone.css">

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

* 针对分辨率的宽度在 768px 之上的样式表

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

* 针对分辨率的宽度在 768px 之下的样式表

[class\*="col-"] {

width: 100%;

padding: 15px;

}

###### 样式表中的 CSS 媒体查询

* HTML 代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>CSS样式表中的媒体查询</title>

<meta name="viewport" content="width=device-width,initial-scale=1"/>

<link rel="stylesheet" href="css/main.css">

<link rel="stylesheet" href="css/media.css">

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

* CSS 代码

/\* For desktop: \*/

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

@media only screen and (max-width: 768px) {

/\* For mobile phones: \*/

[class\*="col-"] {

width: 100%;

}

}

## BOOTSTRAP

### 起步

#### 1. 什么是BOOTSTRAP

##### 1）BOOTSTRAP概述

Bootstrap 是 Twitter 公司开发的一个基于 HTML、CSS、JavaScript 的技术框架，符合 HTML、CSS 规范，且代码简介、视觉优美。

Bootstrap 集合 HTML、CSS 和 JavaScript，使用了最新的浏览器技术，为实现快速开发提供了一套前端工具包。使用 Bootstrap 不仅可以构建出非常优雅的 HTML 页面，而且占用资源非常少，使用 gzip 压缩后大小仅用 10KB。并且 Bootstrap 在跨浏览器兼容方面表示也很好。

##### 2）选择BOOTSTRAP的理由

* Bootstrap 的 HTML 是基于 HTML5 的最新技术。
* Bootstrap 可以快速实现响应式页面。
* Bootstrap 集成了非常友好的 CSS 样式表，对于非设计人员也可以制作出很漂亮的网页。

##### 3）BOOTSTRAP提供的功能

根据 Bootstrap 官网提供的文档，可以知道 Bootstrap 分为以下几个模块：

* 全局 CSS 样式：提供了格栅系统、表格、表单、按钮等集成样式。
* 组件：提供了下拉菜单、输入框、导航、列表组等组件。
* 插件：提供了模态框、滚动监听、警告框、弹出框等插件。

##### 4）BOOTSTRAP的版本变化

* 2011年8月，Twitter 推出了 Bootstrap 1 版本，该工具由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作完成。
* 2012年1月，Twitter 推出了 Bootstrap 2 版本。相比 Bootstrap 1 版本，Bootstrap 2 添加了响应式设计、采用 12 栏栅格布局，并且清晰地划分出 CSS 布局、组件和插件等功能。
* 2013年8月，Twitter 推出了 Bootstrap 3 版本。在这个版本中，采用移动优先和更好的盒子模型等。并且官方不再支持 IE 7及以下版本的浏览器。
* 目前，Bootstrap 4 只是开发者预览版。

#### 2. 下载BOOTSTRAP

##### 1）Bootstrap 官网及下载地址

* Bootstrap 官网：<http://getbootstrap.com>
* Bootstrap 下载地址：[http://getbootstrap.com/getting-started/#download](http://getbootstrap.com/getting-started/" \l "download)

##### 2）Bootstrap 使用的两种方式

* 将 Bootstrap 提供的压缩包下载，导入到工程目录中使用。
* 使用 CDN 加速服务，例如以下内容：

<!-- 新 Bootstrap 核心 CSS 文件 -->

<link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.6/css/bootstrap.min.css">

<!-- 可选的Bootstrap主题文件（一般不用引入） -->

<link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->

<script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script src="//cdn.bootcss.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

##### 3）Bootstrap 目录结构及说明

下载压缩包之后，将其解压缩到任意目录即可看到以下（压缩版的）目录结构：

bootstrap/

├── css/ 样式库

│ ├── bootstrap.css 样式文件

│ ├── bootstrap.css.map

│ ├── bootstrap.min.css 压缩后的样式文件

│ ├── bootstrap-theme.css 主题文件

│ ├── bootstrap-theme.css.map

│ └── bootstrap-theme.min.css 压缩后的主题文件

├── js/ 核心 js 文件

│ ├── bootstrap.js 核心 js 文件

│ └── bootstrap.min.js 压缩后的核心 js 文件

└── fonts/ 字体库

├── glyphicons-halflings-regular.eot

├── glyphicons-halflings-regular.svg

├── glyphicons-halflings-regular.ttf

├── glyphicons-halflings-regular.woff

└── glyphicons-halflings-regular.woff2

#### 3. 对浏览器的支持情况

##### 1）目前支持的浏览器

|  | **Chrome** | **Firefox** | **Internet Explorer** | **Opera** | **Safari** |
| --- | --- | --- | --- | --- | --- |
| Android | 支持 | 支持 | N/A | 不支持 | N/A |
| IOS | 支持 | N/A | N/A | 不支持 | 支持 |
| Mac OS X | 支持 | 支持 | N/A | 支持 | 支持 |
| Windows | 支持 | 支持 | 支持 | 支持 | 不支持 |

##### 2）IE 8及之前版本的兼容

* 对 HTML 5 提供的新元素，需要通过 html5shiv.js 库进行兼容。
* 对 CSS 3 提供的媒体查询，需要通过 respond.js 库进行支持。

需要在使用 Bootstrap 框架的页面 head 元素中，插入以下内容：

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

##### 3）IE 兼容模式

Bootstrap 不支持 IE 古老的兼容模式。为了让 IE 浏览器运行最新的渲染模式下，建议将此 <meta> 元素加入到页面中：

<meta http-equiv="X-UA-Compatible" content="IE=edge">

##### 4）国产浏览器高速模式

国内浏览器厂商一般都支持兼容模式（即 IE 内核）和高速模式（即 webkit 内核），不幸的是，所有国产浏览器都是默认使用兼容模式，这就造成由于低版本 IE （IE8 及以下）内核让基于 Bootstrap 构建的网站展现效果很糟糕的情况。

将下面的 <meta> 元素加入到页面中，可以让部分国产浏览器默认采用高速模式渲染页面：

<meta name="renderer" content="webkit">

#### 4. box-sizing

Bootstrap 默认使用的盒子模型是 border-box，可能会和一些第三方组件冲突。

为了避免 Bootstrap 设置的全局盒模型所带来的影响，可以重置单个页面元素或覆盖整个区域的盒模型。

* 覆盖单个页面元素

/\* 通过 CSS 代码覆盖单个页面元素的盒模型 \*/

.element {

-webkit-box-sizing: content-box;

-moz-box-sizing: content-box;

box-sizing: content-box;

}

* 重置整个区域

/\* 通过 CSS 代码重置整个区域 \*/

.reset-box-sizing,

.reset-box-sizing \*,

.reset-box-sizing \*:before,

.reset-box-sizing \*:after {

-webkit-box-sizing: content-box;

-moz-box-sizing: content-box;

box-sizing: content-box;

}

#### 5. 基本模板

Bootstrap 提供了一个最基本的 HTML 模板，基于这个模板开始实现 Bootstrap 响应式页面。

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<!-- 设置当前 HTML 页面的编码格式为 UTF-8 -->

<meta charset="utf-8">

<!-- 设置 IE 的兼容模式 -->

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- 设置移动优先 -->

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- 上述3个meta标签必须放在最前面，任何其他内容都必须跟随其后！ -->

<title>Bootstrap最基本的HTML模板</title>

<!-- 导入 Bootstrap 框架的 CSS 文件 -->

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

<!--

html5shiv.js 文件解决 IE8及之前版本的浏览器支持 HTML 5 元素的问题。

respond.js 文件解决 IE8及之前版本的浏览器支持 CSS 3 的媒体查询问题。

-->

<!--[if lt IE 9]>

<script src="bootstrap/js/html5shiv.min.js"></script>

<script src="bootstrap/js/respond.min.js"></script>

<![endif]-->

</head>

<body>

<h1>你好，世界！</h1>

<!--

由于 Bootstrap 是基于 jQuery 实现的核心 js 功能,

所以想要使用 Bootstrap 提供的插件功能,需要先导入 jQuery 文件。

-->

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<!-- 导入 Bootstrap 框架的 js 文件 -->

<script src="bootstrap/js/bootstrap.min.js"></script>

</body>

</html>

### 全局CSS样式

#### 1. 概述

##### 1）HTML 5 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。

<!DOCTYPE html>

<html lang="zh-CN">

...

</html>

##### 2）移动设备优先

Bootstrap 是移动设备优先的。为了确保适当的绘制和触屏缩放，需要在 <head> 之中添加 viewport 元元素。

<meta name="viewport" content="width=device-width, initial-scale=1">

在移动设备浏览器上，通过为 viewport设置 meta 属性为 user-scalable=no 可以禁用其缩放功能。这样禁用缩放功能后，用户只能滚动屏幕，就能让你的网站看上去更像原生应用的感觉。

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

##### 3）排版与链接

Bootstrap 排版、链接样式设置了基本的全局样式。

* 为 body 元素设置 background-color: #fff;
* 使用 @font-family-base、@font-size-base 和 @line-height-base 变量作为排版的基本参数
* 为所有链接设置了基本颜色 @link-color ，并且当链接处于 :hover 状态时才添加下划线

#### 2. Normalize.css

##### 1）CSS 重置样式库

为了增强跨浏览器表现的一致性，Bootstrap 使用 Normalize.css（CSS 重置样式库）。

##### 2）什么是 Normalize.css

Normalize.css 只是一个很小的 CSS 文件，但它在默认的 HTML 元素样式上提供了跨浏览器的高度一致性。相比于传统的 CSS reset，Normalize.css 是一种现代的、为 HTML5 准备的优质替代方案。

Normalize.css 的主要特点如下：

* 保护有用的浏览器默认样式而不是完全去掉它
* 一般化的样式：为大部分HTML元素提供
* 修复浏览器自身的bug并保证各浏览器的一致性
* 优化CSS可用性：用一些小技巧
* 解释代码：用注释和详细的文档来

##### 3）Normalize.css 与传统 Reset 的区别

* Normalize.css 保护了有价值的默认值
* Normalize.css 修复了浏览器的bug
* Normalize.css 不会让你的调试工具变的杂乱
* Normalize.css 是模块化的
* Normalize.css 拥有详细的文档

#### 3. 布局容器

Bootstrap 提供了以下两种布局容器：

.container 类用于固定宽度并支持响应式布局的容器。

| **媒体查询** | **宽度值** |
| --- | --- |
| >1200px | 1170px |
| >992px | 970px |
| >768px | 750px |
| 小分辨率 | 100% |

.container-fluid 类用于 100% 宽度，占据 viewport 的容器。

**值得注意的是：**这两种 容器类不能互相嵌套。

#### 4. 按钮

##### 1）作为按钮的元素

Bootstrap 利用 <a>、<button>和<input>元素作为按钮样式。

<div class="container">

<div class="row">

<a class="btn btn-default" href="#" role="button">Link</a>

<button class="btn btn-default" type="submit">Button</button>

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

<input class="btn btn-default" type="submit" value="Submit">

</div>

</div>

**值得注意的是：**

1. 导航和导航条组件只支持 <button> 元素。
2. 如果 <a> 元素被作为按钮使用，务必为其设置 role="button" 属性。
3. 建议尽可能使用 <button> 元素。

##### 2）预定义样式

Bootstrap 为按钮预定义了很多样式，具体样式说明如下：

| **样式名称** | **描述** |
| --- | --- |
| btn-default | 默认样式 |
| btn-primary | 首选项 |
| btn-success | 成功样式 |
| btn-info | 一般信息 |
| btn-warning | 警告样式 |
| btn-danger | 危险样式 |
| btn-link | 链接样式 |

<div class="container">

<div class="row">

<button type="button" class="btn btn-default">（默认样式）Default</button>

<button type="button" class="btn btn-primary">（首选项）Primary</button>

<button type="button" class="btn btn-success">（成功）Success</button>

<button type="button" class="btn btn-info">（一般信息）Info</button>

<button type="button" class="btn btn-warning">（警告）Warning</button>

<button type="button" class="btn btn-danger">（危险）Danger</button>

<button type="button" class="btn btn-link">（链接）Link</button>

</div>

</div>

##### 3）不同尺寸的按钮

* Bootstrap 为按钮提供了大、默认、小和很小几个尺寸。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg">（大按钮）Large button</button>

<button type="button" class="btn btn-default">（默认尺寸）Default button</button>

<button type="button" class="btn btn-default">（默认尺寸）Default button</button>

<button type="button" class="btn btn-default btn-xs">（超小尺寸）Extra small button</button>

</div>

</div>

* 通过给按钮添加 .btn-block 类可以将其拉伸至父元素100%的宽度，而且按钮也变为了块级（block）元素。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg btn-block">（块级元素）Block level button</button>

</div>

</div>

##### 4）按钮激活状态

为按钮元素添加 .active 类，设置其为激活状态。当按钮处于激活状态时，其表现为被按压下去（底色更深、边框夜色更深、向内投射阴影）。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg active">Button</button>

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

</div>

</div>

##### 5）按钮禁用状态

通过为按钮的背景设置 opacity 属性就可以呈现出无法点击的效果。

* 为 <button> 元素添加 disabled 属性，使其表现出禁用状态。

<button type="button" class="btn btn-default btn-lg" disabled="disabled">Button</button>

**跨浏览器兼容性**

Internet Explorer 9 及更低版本的浏览器将会把按钮中的文本绘制为灰色，并带有恶心的阴影，目前还没有解决办法。

* 为基于 <a> 元素创建的按钮添加 .disabled 类。

<a href="#" class="btn btn-default btn-lg disabled" role="button">Link</a>

#### 5. 图片

##### 1）响应式图片

通过为图片添加 .img-responsive 类可以让图片支持响应式布局。

如果需要让使用了 .img-responsive 类的图片水平居中，请使用 .center-block 类，不要用 .text-center。

<div class="container">

<div class="row">

<img src="imgs/1.jpg" class="img-responsive" alt="Responsive image">

</div>

<div class="row">

<img src="imgs/1.jpg" class="img-responsive center-block" alt="Responsive image">

</div>

</div>

##### 2）图片形状

通过为 元素添加以下相应的类，可以让图片呈现不同的形状。

| **类名** | **描述** |
| --- | --- |
| img-rounded | 圆角矩形 |
| img-circle | 圆形 |
| img-thumbnail | 矩形 |

IE 8 不支持 CSS3 中的圆角属性。

<div class="container">

<div class="row">

<img src="imgs/1.jpg" alt="..." class="img-rounded">

<img src="imgs/1.jpg" alt="..." class="img-circle">

<img src="imgs/1.jpg" alt="..." class="img-thumbnail">

</div>

</div>

#### 6. 文本

##### 1）文本对齐

通过以下文本对齐类，可以简单方便的将文字重新对齐。

| **类名** | **描述** |
| --- | --- |
| text-left | 左对齐 |
| text-center | 居中对齐 |
| text-right | 右对齐 |
| text-justify | 两端对齐，段落中超出屏幕部分文字自动换行 |
| text-nowrap | 段落中超出屏幕部分不换行 |

<div class="container">

<div class="row">

<p class="text-left">Left aligned text.</p>

<p class="text-center">Center aligned text.</p>

<p class="text-right">Right aligned text.</p>

<p class="text-justify">Justified text.</p>

<p class="text-nowrap">No wrap text.</p>

</div>

</div>

##### 2）改变大小写

通过以下这几个类可以改变文本的大小写。

| **类名** | **描述** |
| --- | --- |
| text-lowercase | 小写 |
| text-uppercase | 大写 |
| text-capitalize | 首字母大写 |

<div class="container">

<div class="row">

<p class="text-lowercase">Lowercased text.</p>

<p class="text-uppercase">Uppercased text.</p>

<p class="text-capitalize">Capitalized text.</p>

</div>

</div>

##### 3）文本颜色

Bootstrap 允许为文本设置不同颜色表示不同含义，具体如下：

| **类名** | **描述** |
| --- | --- |
| text-muted | 提示，使用浅灰色 |
| text-primary | 首选项，使用蓝色 |
| text-success | 成功，使用浅绿色 |
| text-info | 信息，使用浅蓝色 |
| text-warning | 警告，使用黄色 |
| text-danger | 危险，使用褐色 |

<div class="container">

<div class="row">

<p class="text-muted">提示信息</p>

<p class="text-primary">首选项</p>

<p class="text-success">成功</p>

<p class="text-info">信息</p>

<p class="text-warning">警告</p>

<p class="text-danger">危险</p>

</div>

</div>

# BOOTSTRAP第二天笔记

## 全局 CSS 样式

### 1. 列表

#### 1）无序列表

<div class="container">

<div class="row">

<ul>

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

#### 2）有序列表

<div class="container">

<div class="row">

<ol>

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ol>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ol>

</li>

<li>芒果</li>

<li>樱桃</li>

</ol>

</div>

</div>

#### 3）无样式列表

移除了默认的 list-style 样式和左侧外边距的一组元素（只针对直接子元素）。

<div class="container">

<div class="row">

<ul class="list-unstyled">

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

#### 4）内联列表

通过设置 display: inline-block; 并添加少量的内补（padding），将所有元素放置于同一行。

<div class="container">

<div class="row">

<ul class="list-inline">

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

#### 5）描述

##### a. 垂直排列的描述

带有描述的短语列表。

<div class="container">

<div class="row">

<dl>

<dt>Description lists</dt>

<dd>A description list is perfect for defining terms.</dd>

<dt>Euismod</dt>

<dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.<br>

Donec id elit non mi porta gravida at eget metus.</dd>

<dt>Malesuada porta</dt>

<dd>Etiam porta sem malesuada magna mollis euismod.</dd>

</dl>

</div>

</div>

##### b. 水平排列的描述

.dl-horizontal 可以让

内的短语及其描述排在一行。

<div class="container">

<div class="row">

<dl class="dl-horizontal">

<dt>Description lists</dt>

<dd>A description list is perfect for defining terms.</dd>

<dt>Euismod</dt>

<dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.<br>

Donec id elit non mi porta gravida at eget metus.</dd>

<dt>Malesuada porta</dt>

<dd>Etiam porta sem malesuada magna mollis euismod.</dd>

</dl>

</div>

</div>

### 2. 表格

#### 1）基本表格

为任意 <table> 标签添加 .table 类可以为其赋予基本的样式。

<div class="container">

<div class="row">

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

#### 2）条纹状表格

通过 .table-striped 类可以给 <tbody> 之内的每一行增加斑马条纹样式。

<div class="container">

<div class="row">

<table class="table table-striped">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

条纹状表格是依赖 :nth-child CSS 选择器实现的，而这一功能不被 IE 8 支持。

#### 3）带边框表格

添加 .table-bordered 类为表格和其中的每个单元格增加边框。

<div class="container">

<div class="row">

<table class="table table-bordered">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

#### 4）鼠标悬停

通过添加 .table-hover 类可以让 <tbody> 中的每一行对鼠标悬停状态作出响应。

<div class="container">

<div class="row">

<table class="table table-hover">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

#### 5）紧缩表格

通过添加 .table-condensed 类可以让表格更加紧凑，单元格中的 padding 均会减半。

<div class="container">

<div class="row">

<table class="table table-condensed">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

#### 6）状态类

通过这些状态类可以为行或单元格设置颜色。

| 类名 | 描述 | | active | 鼠标悬停在行或单元格上时所设置的颜色 | | success | 标识成功或积极的动作 | | info | 标识普通的提示信息或动作 | | warning | 标识警告或需要用户注意 | | danger | 标识危险或潜在的带来负面影响的动作 |

<div class="container">

<div class="row">

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr class="active">

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr class="success">

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td class="info">3</td>

<td>笔记本</td>

<td class="warning">2000</td>

</tr>

<tr>

<td>3</td>

<td class="danger">笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**值得注意的是：**

这些状态类不仅可以作用于<tr>元素，还可以作用于<td>元素。

#### 7）响应式表格

将任何 .table 元素包裹在 .table-responsive 元素内，即可创建响应式表格。其会在小屏幕设备上（小于768px）水平滚动。当屏幕大于 768px 宽度时，水平滚动条消失。

### 3. 辅助样式

#### 1）背景颜色

和文本颜色一样，Bootstrap 允许添加不同的背景颜色表示不同含义，具体如下：

| **类名** | **描述** |
| --- | --- |
| bg-primary | 首选项，使用蓝色 |
| bg-success | 成功，使用浅绿色 |
| bg-info | 信息，使用浅蓝色 |
| bg-warning | 警告，使用黄色 |
| bg-danger | 危险，使用褐色 |

<div class="container">

<div class="row">

<p class="bg-muted">提示信息</p>

<p class="bg-primary">首选项</p>

<p class="bg-success">成功</p>

<p class="bg-info">信息</p>

<p class="bg-warning">警告</p>

<p class="bg-danger">危险</p>

</div>

</div>

#### 2）浮动

##### a. 浮动

通过添加 .pull-left 或 .pull-right 类，可以将任意元素向左或向右浮动。

* CSS 代码

.block1 {

width : 300px;

height : 200px;

border : 1px solid black;

background : dodgerblue;

}

.block2 {

width : 500px;

height : 300px;

border : 1px solid black;

background : yellowgreen;

}

* HTML 代码

<div class="container">

<div>

<div class="block1 pull-left"></div>

<div class="block1 pull-right"></div>

</div>

<div class="block2"></div>

</div>

##### b. 清除浮动

通过为父元素添加 .clearfix 类可以很容易地清除浮动。

* CSS 代码

.block1 {

width : 300px;

height : 200px;

border : 1px solid black;

background : dodgerblue;

}

.block2 {

width : 500px;

height : 300px;

border : 1px solid black;

background : yellowgreen;

}

* HTML 代码

<div class="container">

<div class="clearfix">

<div class="block1 pull-left"></div>

<div class="block1 pull-right"></div>

</div>

<div class="block2"></div>

</div>

#### 3）居中

为任意元素设置 .center-block 类让其中的内容居中。

<div class="center-block"></div>

#### 4）显示或隐藏

.show 和 .hidden 类可以强制任意元素显示或隐藏(对于屏幕阅读器也能起效)，这些类只对块级元素起作用。

另外，.invisible 类可以被用来仅仅影响元素的可见性，也就是说，元素的 display 属性不被改变，并且这个元素仍然能够影响布局。

<div class="show"></div>

<div class="hidden"></div>

<div class="invisible"></div>

<div></div>

### 4. 栅格系统

#### 1）什么是栅格系统

栅格系统用于通过一系列的行（row）与列（column）的组合来创建页面布局，内容就可以放入这些创建好的布局中。

Bootstrap 提供的栅格系统，主要特点如下：

* 行的宽度，可以是固定宽度，也可以是相对宽度（100%）。
* 一行中最多可以包含 12 个列。如果一行中包含的列大于 12，多余的列则被作为一个整体另起一行排列。
* Bootstrap 根据 4 种分辨率提供不同的栅格系统预定义样式。

#### 2）栅格系统工作原理

Bootstrap 的栅格系统的工作原理如下：

* 栅格系统必须指定一个容器元素，该元素的 class 必须为 .container（固定宽度）或 .container-fluid（100%宽度）。
* 作为“行”的元素必须作为容器元素的直接子元素，并且 class 设置为 .row。
* 作为“列”的元素必须是“行”元素的直接子元素（**“行”元素的子元素不能直接包含内容，内容应该被包含在“列”元素中。**），并且一行最多允许创建 12 列。
* “列”元素的预定义样式，具体内容请参考 “3）栅格系统的参数” 内容。

#### 3）栅格系统的参数

Bootstrap 根据 4 种不同的分辨率提供了不同的预定义样式（栅格类）。栅格类适用于与屏幕宽度大于或等于分界点大小的设备 ， 并且针对小屏幕设备覆盖栅格类。

|  | **超小屏幕 手机（<768px）** | **小屏幕 平板（>=768px）** | **中等屏幕 桌面显示器（>=992px）** | **大屏幕 大桌面显示器（>=1200px）** |
| --- | --- | --- | --- | --- |
| 栅格系统行为 | 总是水平排列 | 以折叠开始，大于断点成水平方式 | 以折叠开始，大于断点成水平方式 | 以折叠开始，大于断点成水平方式 |
| 容器宽度 | None（自动） | 750px | 970px | 1170px |
| class 前缀 | .col-xs- | .col-sm- | .col-md- | .col-lg- |
| 列数 | 12 | 12 | 12 | 12 |
| 列宽 | 自动 | 60px | 78px | 95px |
| 槽宽 | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） |
| 可嵌套 | 是 | 是 | 是 | 是 |
| 偏移 | 是 | 是 | 是 | 是 |
| 排序 | 是 | 是 | 是 | 是 |

#### 4）栅格系统示例

* CSS 代码

body {

margin-top : 100px;

}

[class\*=col-] {

border : 1px solid black;

padding-top: 15px;

padding-bottom: 15px;

}

* HTML 代码

<div class="container">

<h3>三等分列</h3>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

</div>

<h3>三不等分列</h3>

<div class="row">

<div class="col-md-3">.col-md-3</div>

<div class="col-md-6">.col-md-6</div>

<div class="col-md-3">.col-md-3</div>

</div>

<h3>手机、平板电脑和PC桌面不同布局</h3>

<div class="row">

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

</div>

</div>

#### 5）调整列的顺序

##### a. 列偏移

使用 .col-md-offset-\* 类可以将列向右侧偏移。（\* 表示向右偏移的列数）

<div class="container">

<h3>列偏移</h3>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4 col-md-offset-4">.col-md-4 .col-md-offset-4</div>

</div>

<div class="row">

<div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>

<div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>

</div>

<div class="row">

<div class="col-md-6 col-md-offset-3">.col-md-6 .col-md-offset-3</div>

</div>

</div>

##### b. 嵌套列

在 Bootstrap 的栅格系统允许进行嵌套。所谓**栅格系统嵌套**，就是指在 Bootstrap 栅格系统中的每列中允许包含一个栅格系统。

被嵌套的“行”中所包含的列的个数依旧不能超过 12 列。

<div class="container">

<h3>嵌套列</h3>

<div class="row">

<div class="col-sm-9">

Level 1: .col-sm-9

<div class="row">

<div class="col-xs-8 col-sm-6">Level 2: .col-xs-8 .col-sm-6</div>

<div class="col-xs-4 col-sm-6">Level 2: .col-xs-4 .col-sm-6</div>

</div>

</div>

</div>

</div>

##### c. 列排序

通过将栅格系统中的列向右移动（push）或向左移动（pull）来改变列的顺序。

* .col-md-push-\*表示向右移动
* .col-md-pull-\*表示向左移动

<div class="container">

<h3>列排序</h3>

<div class="row">

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

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

</div>

</div>

#### 6）响应式工具

通过单独或联合使用以下列出的 class，可以针对不同屏幕尺寸隐藏或显示页面内容。

|  | **超小屏幕 手机（<768px）** | **小屏幕 平板（>=768px）** | **中等屏幕 桌面（>=992px）** | **大屏幕 大桌面（>=1200px）** |
| --- | --- | --- | --- | --- |
| .hidden-xs | 隐藏 | 可见 | 可见 | 可见 |
| .hidden-sm | 可见 | 隐藏 | 可见 | 可见 |
| .hidden-md | 可见 | 可见 | 隐藏 | 可见 |
| .hidden-lg | 可见 | 可见 | 可见 | 隐藏 |

<div class="container">

<h3>响应式工具</h3>

<div class="row">

<div class="col-md-3 col-sm-4 hidden-xs">.col-md-3</div>

<div class="col-md-6 col-sm-8 col-xs-12">.col-md-6</div>

<div class="col-md-3 hidden-sm hidden-xs">.col-md-3</div>

</div>

</div>

#### 7）响应式栅格系统示例

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Bootstrap栅格系统案例</title>

<!-- Bootstrap -->

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

<!--[if lt IE 9]>

<script src="bootstrap/js/html5shiv.min.js"></script>

<script src="bootstrap/js/respond.min.js"></script>

<![endif]-->

</head>

<body>

<div class="jumbotron">

<div class="container">

<h1>Hello, world!</h1>

<p>This is a template for a simple marketing or informational website. It includes a large callout called a

jumbotron and three supporting pieces of content. Use it as a starting point to create something more

unique.</p>

</div>

</div>

<div class="container">

<!-- Example row of columns -->

<div class="row">

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

<h2>Heading</h2>

<p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris

condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis

euismod. Donec sed odio dui. </p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

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

<h2>Heading</h2>

<p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris

condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis

euismod. Donec sed odio dui. </p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

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

<h2>Heading</h2>

<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula

porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut

fermentum massa justo sit amet risus.</p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

</div>

<hr>

<footer>

<p>&copy; 2015 Company, Inc.</p>

</footer>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

</body>

</html>

### 5. 表单

#### 1）基本表单

Bootstrap 为单独的表单元素预定义了全局样式，表单内元素的具体使用如下：

* 设置了 .form-control 类的 <input>、<textarea> 和 <select> 元素都将被默认设置宽度属性为 width: 100%;。
* 将 label 元素和前面提到的控件包裹在 .form-group 中可以获得最好的排列。

<div class="container">

<div class="row">

<form>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</div>

</div>

#### 2）内联表单

为 <form> 元素添加 .form-inline 类可使其内容左对齐并且表现为 inline-block 级别的控件。

只适用于 viewport 至少在 768px 宽度时（视口宽度再小的话就会使表单折叠）。

<div class="container">

<div class="row">

<form class="form-inline">

<div class="form-group">

<label for="exampleInputName2">Name</label>

<input type="text" class="form-control" id="exampleInputName2" placeholder="Jane Doe">

</div>

<div class="form-group">

<label for="exampleInputEmail2">Email</label>

<input type="email" class="form-control" id="exampleInputEmail2" placeholder="jane.doe@example.com">

</div>

<button type="submit" class="btn btn-default">Send invitation</button>

</form>

</div>

</div>

**需要注意的是：**

在内联表单，我们将这些元素的宽度设置为 width: auto;，因此，多个控件可以排列在同一行。

如果你没有为每个输入控件设置 label 标签，屏幕阅读器将无法正确识别。对于这些内联表单，你可以通过为 label 设置 .sr-only 类将其隐藏。

<div class="container">

<div class="row">

<form class="form-inline">

<div class="form-group">

<label class="sr-only" for="exampleInputEmail3">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail3" placeholder="Email">

</div>

<div class="form-group">

<label class="sr-only" for="exampleInputPassword3">Password</label>

<input type="password" class="form-control" id="exampleInputPassword3" placeholder="Password">

</div>

<button type="submit" class="btn btn-default">Sign in</button>

</form>

</div>

</div>

#### 3）水平排列的表单

通过为表单添加 .form-horizontal 类，并联合使用 Bootstrap 预置的栅格类，可以将 label 标签和控件组水平并排布局。

这样做将改变 .form-group 的行为，使其表现为栅格系统中的行（row），因此就无需再额外添加 .row 了。

<div class="container">

<div class="row">

<form class="form-horizontal">

<div class="form-group">

<label for="inputEmail3" class="col-sm-2 control-label">Email</label>

<div class="col-sm-10">

<input type="email" class="form-control" id="inputEmail3" placeholder="Email">

</div>

</div>

<div class="form-group">

<label for="inputPassword3" class="col-sm-2 control-label">Password</label>

<div class="col-sm-10">

<input type="password" class="form-control" id="inputPassword3" placeholder="Password">

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-2 col-sm-10">

<button type="submit" class="btn btn-default">Sign in</button>

</div>

</div>

</form>

</div>

</div>

#### 4）单选框和多选框

##### a. 默认外观

* 单选框使用 class 为 .radio
* 多选框使用 class 为 .checkbox

单选框或多选框的样式要定义在 div 元素中（该元素作为容器元素，包含文本和单选或多选框元素）。

<div class="container">

<div class="row">

<form>

<div class="checkbox">

<label>

<input type="checkbox" value="">

Option one is this and that&mdash;be sure to include why it's great

</label>

</div>

<div class="checkbox disabled">

<label>

<input type="checkbox" value="" disabled>

Option two is disabled

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios1" value="option1" checked>

Option one is this and that&mdash;be sure to include why it's great

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios2" value="option2">

Option two can be something else and selecting it will deselect option one

</label>

</div>

<div class="radio disabled">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios3" value="option3" disabled>

Option three is disabled

</label>

</div>

</form>

</div>

</div>

##### b. 内联单选和多选框

通过将 .checkbox-inline 或 .radio-inline 类应用到一系列的多选框（checkbox）或单选框（radio）控件上，可以使这些控件排列在一行。

<div class="container">

<div class="row">

<form>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox1" value="option1"> 1

</label>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox2" value="option2"> 2

</label>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox3" value="option3"> 3

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio1" value="option1"> 1

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio2" value="option2"> 2

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio3" value="option3"> 3

</label>

</form>

</div>

</div>

## 组件

### 1. 字体图标

Bootstrap 使用的字体图标是来自 Glyphicions Halflings 提供的。**在使用的时候，需要遵守以下几点规则：**

* 图标使用的 class 不能与其他组件的 class 混合使用。换言之，图标的 class 必须被定义在 span 元素上。
* 图标使用的 class 具有一个基类和一个修饰类。基类（class）统一为 .glyphicon，修饰类（class）为 .glyphicon-\*-\*或.glyphicon-\*。
* 只对内容为空的元素起作用（图标使用的 class 只能应用在不包含任何文本内容或子元素的元素上）。

<div class="container">

<p class="text-danger">

<span class="glyphicon glyphicon-warning-sign"></span>

<span class="glyphicon glyphicon-alert"></span>

警告：您的浏览器版本太低了!

</p>

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

<span class="glyphicon glyphicon-apple"></span>

删除

</button>

<button class="btn btn-danger" type="button">

<span class="glyphicon glyphicon-arrow-left"></span>

</button>

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

<span class="glyphicon glyphicon-star"></span>

</button>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-play"></span>

</a>

</div>

### 2. 下拉菜单

#### 1）基本样式

Bootstrap 提供的下拉菜单并不是 HTML 默认的 <select> 元素，而且利用一组 HTML 元素组合而成。

* <div>元素作为容器元素

<button>或<a>元素作为下拉菜单的提示项。

作为下拉菜单的提示项，Bootstrap 默认显示的是按钮样式（也就是需要设置 class 为 btn btn-\*）。

按钮在被点击后，会提供不同的样式效果。如果想改变这样的样式，需要添加样式dropdown-toggle。

通过为下拉菜单的提示项设置属性 data-toggle=dropdown ，实现下拉菜单的动态显示。（此内容会在 **组件** 详细讲解）

无序列表作为下拉菜单的列表项。

##### a. 向下弹出

为作为下拉菜单的容器元素设置样式dropdown即可。

<div class="container">

<div class="dropdown">

<button class="btn btn-default" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

##### b. 向上弹出

为作为下拉菜单的容器元素设置样式dropup即可。

<div class="container">

<div class="dropup">

<button class="btn btn-default" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

#### 2）标题

可以为作为下拉菜单的列表项中，添加 class 为 dropdown-header 的 <li> 元素作为标题，表示一组动作。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li class="dropdown-header">Dropdown header</li>

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li class="dropdown-header">Dropdown header</li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

#### 3）分割线

可以为作为下来菜单的列表项中，添加 class 为 divider 的 <li> 元素作为分割线，表示将多个选项进行分组。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li class="divider"></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

#### 4）禁用项

为下拉菜单中的 <li> 元素添加 .disabled 类，从而禁用相应的菜单项。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Regular link</a></li>

<li class="disabled"><a href="#">Disabled link</a></li>

<li><a href="#">Another link</a></li>

</ul>

</div>

</div>

### 3. 按钮组

#### 1）基本按钮组

通过按钮组容器把一组按钮放在同一组里，为按钮组容器设置 class 为 btn-group。

<div class="container">

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

#### 2）不同布局

##### a. 两端对齐

让一组按钮拉长为相同的尺寸，填满父元素的宽度。为按钮组容器添加样式 btn-group-justified。

<div class="container">

<div class="btn-group btn-group-justified">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

##### b. 垂直排列

让一组按钮垂直堆叠排列显示而不是水平排列，为按钮组容器设置 class 为 btn-group-vertical。

<div class="container">

<div class="btn-group-vertical">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

#### 3）尺寸

只要给 .btn-group 加上 .btn-group-\* 类，就省去为按钮组中的每个按钮都赋予尺寸类。

Bootstrap 提供了以下几种尺寸：

| **class 名称** | **描述** |
| --- | --- |
| btn-group-lg | 大 |
| btn-group-sm | 小 |
| btn-group-xs | 很小 |

除上述尺寸外，Bootstrap 还提供一种默认尺寸（无需设置）。

<div class="container">

<div class="btn-group btn-group-lg">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group btn-group-sm">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group btn-group-xs">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

</div>

#### 4）按钮工具栏

把一组 <div class="btn-group"> 组合包裹在一个 <div class="btn-toolbar"> 中就可以做成更复杂的组件。

<div class="container">

<div class="btn-toolbar">

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

</div>

#### 5）嵌套下拉菜单

想要把下拉菜单混合到一系列按钮中，只须把 .btn-group 放入另一个 .btn-group 中。

<div class="container">

<div class="btn-group">

<button type="button" class="btn btn-default">1</button>

<button type="button" class="btn btn-default">2</button>

<div class="btn-group" role="group">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Dropdown link</a></li>

<li><a href="#">Dropdown link</a></li>

</ul>

</div>

</div>

</div>

### 4. 警告框

警告框组件通过提供一些灵活的预定义消息，为常见的用户动作提供反馈消息。

#### 1）基本警告框

Bootstrap 的警告框样式由 .alert 和 4 个修饰class组成。4 个修饰class表示不同的警告信息。

| **class 名称** | **描述** |
| --- | --- |
| alert-success | 成功 |
| alert-info | 信息 |
| alert-warning | 警告 |
| alert-danger | 危险 |

<div class="container">

<div class="alert alert-danger">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器禁用了Cookie！本站的部分功能无法启用！

</div>

<div class="alert alert-info">

<span class="glyphicon glyphicon-alert"></span>

没有更多数据可供加载了！

</div>

<div class="alert alert-warning">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器OUT啦！请使用更新版本的浏览器!

</div>

<div class="alert alert-success">

<span class="glyphicon glyphicon-alert"></span>

您已经登录了！无需再次登录！

</div>

</div>

#### 2）可关闭的警告框

为警告框添加一个可选的 .alert-dismissible 类和一个关闭按钮。

<div class="container">

<div class="alert alert-success alert-dismissible">

<!--.close元素必须是alert的第一个子元素-->

<span data-dismiss="alert" class="close">&times;</span>

<span class="glyphicon glyphicon-alert"></span>

您已经登录了！无需再次登录！

</div>

</div>

#### 3）警告框中的链接

用 .alert-link 工具类，可以为链接设置与当前警告框相符的颜色。

<div class="container">

<div class="alert alert-warning">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器OUT啦！请使用更新版本的<a href="#" class="alert-link">浏览器</a>!

</div>

</div>

# BOOTSTRAP第三天笔记

## 组件

### 1. 导航

Bootstrap 中的导航组件都依赖同一个 .nav 类。

#### 1）两种导航样式

Bootstrap 提供标签页主要利用的是 **无序列表** 元素。

##### a. 标签页式导航

标签页式导航，就是为作为标签页的<ul>元素添加 class 为 .nav-tab 的样式。

**需要注意的是：**

* .nav-tabs 类依赖 .nav 基类。
* 默认被点击的样式为 active。
* 实现动态切换效果，需要为作为导航项的元素设置 data-toggle 为 tab。（具体内容会在 **组件** 详细讲解）

<div class="container">

<h3>标签页式导航</h3>

<ul class="nav nav-tabs">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

##### b. 胶囊式标导航

胶囊式导航，就是为作为标签页的<ul>元素添加 class 为 .nav-pills 的样式。

###### 水平方向

胶囊式导航的默认方向，就是水平方向。

<div class="container">

<h3>胶囊式导航</h3>

<ul class="nav nav-pills">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

###### 垂直方向

想要将胶囊式导航设置为垂直方向，只需为作为标签页的<ul>元素添加 class 为 .nav-stacked 的样式。

<div class="container">

<h3>垂直方向的胶囊式导航</h3>

<ul class="nav nav-pills nav-stacked">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

#### 2）两端对齐

在大于 768px 的屏幕上，通过 .nav-justified 类可以很容易的让标签页或胶囊式导航呈现出同等宽度。在小屏幕上，导航链接以垂直方向呈现。

<div class="container">

<ul class="nav nav-tabs nav-justified">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

<ul class="nav nav-pills nav-justified">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

#### 3）嵌套下拉菜单

导航中也可以嵌入下拉菜单。

<div class="container">

<ul class="nav nav-pills">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Help</a></li>

<li data-toggle="tab" class="dropdown">

<a class="dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</a>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li><a href="#">Separated link</a></li>

</ul>

</li>

</ul>

</div>

### 2. 导航条

导航条是在应用或网站中作为导航页头的响应式基础组件。

#### 1）导航条样式

Bootstrap 提供的导航条使用 HTML5 的语义化元素 <nav> 作为容器元素，并为该元素设置 class 为 .navbar。

##### a. 默认样式的导航条

默认样式的导航条，需要为作为容器元素的 <nav> 元素，添加 .navbar-default 样式。

<div class="container">

<nav class="navbar navbar-default">

<!-- 导航条组件 -->

</nav>

</div>

##### b. 反色的导航条

通过添加 .navbar-inverse 类可以改变导航条的外观。

<div class="container">

<nav class="navbar navbar-inverse">

<!-- 导航条组件 -->

</nav>

</div>

#### 2）导航条组件

##### a. 品牌图标

一般在导航条添加网站的名称或图标，使用 <a> 元素，并且为其设置 class 为 .navbar-brand。

<div class="container">

<nav class="navbar navbar-default">

<a class="navbar-brand" href="#">

<img src="imgs/logo.png" alt="Brand" width="20">

</a>

</nav>

</div>

**需要注意的是：**

* .navbar-brand样式设置的高度为 50px，并且内边距为 15px。所以，如果使用图标的话，一定设置图标的高度为 20px。

##### b. 表单

在导航条添加表单的话，需要为 <form> 元素设置 class 为 navbar-form。表单内组件可以呈现很好的垂直对齐，并且较窄的宽度中呈现折叠状态。

<div class="container">

<nav class="navbar navbar-default">

<form class="navbar-form">

<div class="form-group">

<input type="text" class="form-control" placeholder="Search">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</nav>

</div>

##### c. 按钮

如果要在导航条中添加非表单内的按钮时，需要为其添加 navbar-btn 样式，使之在导航条中垂直居中。

<div class="container">

<nav class="navbar navbar-default">

<button class="btn btn-default navbar-btn">Sign in</button>

</nav>

</div>

##### d. 文本

在导航条中包含普通的文本内容，需要使用 <p> 元素，并且为其设置 class 为 navbar-text，使其具有正确的行距和颜色。

<div class="container">

<nav class="navbar navbar-default">

<p class="navbar-text">Signed in as Mark Otto</p>

</nav>

</div>

##### e. 导航

在导航条中最主要的功能就是集成导航，需要为作为导航的 <ul> 元素设置 class 为 navbar-nav。

<div class="container">

<nav class="navbar navbar-default">

<ul class="nav navbar-nav">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</nav>

</div>

##### f. 组件排列

通过添加 .navbar-left 和 .navbar-right 工具类让导航链接、表单、按钮或文本对齐。

<div class="container">

<nav class="navbar navbar-default">

<ul class="nav navbar-nav navbar-left">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

<form class="navbar-form navbar-right">

<div class="form-group">

<input type="text" class="form-control" placeholder="Search">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</nav>

</div>

#### 3）固定的导航条

固定的导航条不会随着页面滚动而消失。

##### a. 固定在顶部

为 <nav> 元素添加 .navbar-fixed-top 类可以让导航条固定在顶部，并且导航条的宽度与页面宽度一致。

<div class="container">

<nav class="navbar navbar-default navbar-fixed-top">

<!-- 导航条组件 -->

</nav>

</div>

##### b. 固定在底部

为 <nav> 元素添加 .navbar-fixed-bottom 类可以让导航条固定在底部，并且导航条的宽度与页面宽度一致。

<div class="container">

<nav class="navbar navbar-default navbar-fixed-bottom">

<!-- 导航条组件 -->

</nav>

</div>

#### 4）响应式导航条

Bootstrap 以 768px 宽度为分界点，分别进行设置导航条的内容及样式。

| **class 名称** | **描述** |
| --- | --- |
| navbar-header | 针对宽度小于 768px 的屏幕 |
| navbar-collapse | 针对宽度大于等于 768px 的屏幕 |

###### 针对小于 768px 屏幕的导航条

* 使用 <div> 作为容器元素，并为其设置 class 为 navbar-header。
* 一般使用 <button> 或 <a> 元素，并设置 class 为 navbar-toggle。
* 通过为 <button> 或 <a> 元素设置属性 data-toggle 为 collapse，实现点击交互效果。
* 通过为 <button> 或 <a> 元素设置属性 data-target 为 对应菜单容器元素的 id 属性值。
* 在 <button> 或 <a> 元素内添加三个 <span class="icon-bar"></span> 元素（汉堡包按钮样式）。

<div class="container">

<nav class="nav navbar-default">

<div class="navbar-header">

<button class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<!-- 导航条组件 -->

</nav>

</div>

###### 针对大于等于 768px 屏幕的导航条

使用 <div> 作为容器元素，并为其设置 class 为 collapse navbar-collapse 和 id 属性（**用于与针对小于 768px 屏幕的**button**元素的**data-target**属性值对应。**）。

<div class="container">

<nav class="nav navbar-default">

<!-- 针对宽度小于 768px 的汉堡包按钮 -->

<div class="collapse navbar-collapse" id="navbar-collapse">

<ul class="nav navbar-nav">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

</nav>

</div>

### 3. 媒体对象

Bootstrap 媒体对象常用于用户的评论、帖子或商品列表。

* 使用 <div> 作为媒体对象的容器元素，并设置 class 为 .media 。
* 媒体对象提供左（media-left）、主体（media-body）和右（media-right）三种结构。
* 设置垂直方向的上（media-top）、中（media-middle）和下（media-bottom）三种。
* 如果使用图片的话，需要设置为media-object。

<div class="container">

<h3>左中结构</h3>

<div class="media">

<div class="media-left">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

<div class="media-body">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

</div>

<hr>

<h3>中右结构</h3>

<div class="media">

<div class="media-body media-middle">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

<div class="media-right">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

</div>

<hr>

<h3>左中右结构</h3>

<div class="media">

<div class="media-left">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

<div class="media-body media-bottom">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

<div class="media-right">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

</div>

</div>

### 4. 列表组

#### 1）基本列表组

默认的列表组使用无序列表实现。

<ul> 元素作为列表组的容器元素，并为其设置 class 为 list-group。<li> 元素作为列表组的列表项，并为其设置 class 为 list-group-item。

也可以将 <ul> 元素替换为 <div> 元素，将 <li> 元素替换为 <button> 或 <a> 元素。

<div class="container">

<ul class="list-group">

<li class="list-group-item">Cras justo odio</li>

<li class="list-group-item">Dapibus ac facilisis in</li>

<li class="list-group-item">Morbi leo risus</li>

<li class="list-group-item">Porta ac consectetur ac</li>

<li class="list-group-item">Vestibulum at eros</li>

</ul>

</div>

#### 2）徽章

给链接、导航等元素嵌套 <span class="badge"> 元素，可以很醒目的展示新的或未读的信息条目。

<div class="container">

<ul class="list-group">

<li class="list-group-item">

<span class="badge">14</span>

Cras justo odio

</li>

<li class="list-group-item">

<span class="badge">2</span>

Dapibus ac facilisis in

</li>

<li class="list-group-item">Morbi leo risus</li>

</ul>

</div>

### 5. 路径导航

在一个带有层次的导航结构中标明当前页面的位置。

<div class="container">

<ol class="breadcrumb">

<li><a href="#">Home</a></li>

<li><a href="#">Library</a></li>

<li class="active">Data</li>

</ol>

</div>

### 6. 标签

为具有特定含义的文本设置为标签样式。标签具有一个基类（label）和一个修饰类（label-\*），修饰类预定义以下几种 class 即可改变标签的外观。

| **class 名称** | **描述** |
| --- | --- |
| label-default | 默认样式 |
| label-primary | 首选项样式 |
| label-success | 成功样式 |
| label-info | 信息样式 |
| label-warning | 警告样式 |
| label-danger | 危险样式 |

<div class="container">

<span class="label label-default">Default</span>

<span class="label label-primary">Primary</span>

<span class="label label-success">Success</span>

<span class="label label-info">Info</span>

<span class="label label-warning">Warning</span>

<span class="label label-danger">Danger</span>

</div>

### 7. Well

把 Well 用在元素上，能有嵌入（inset）的简单效果。一般用于文章的导读等功能。

Well 由一个基类（well）和一个修饰类（well-\*）组成。

| **class 名称** | **描述** |
| --- | --- |
| well-lg | 大边距样式 |
| well-sm | 小边距样式 |

<div class="container">

<div class="well">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Animi commodi, corporis debitis doloremque harum illo ipsam ipsum itaque labore minima non obcaecati qui quis quos recusandae reiciendis sunt unde ut.</div>

<div class="well well-lg">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus ad assumenda atque beatae debitis ducimus esse explicabo fuga, id ipsum maiores nobis non odio pariatur quae quos rerum sequi voluptas.</div>

</div>

### 8. 巨幕

这是一个轻量、灵活的组件，它能延伸至整个浏览器视口来展示网站上的关键内容。

<div class="container">

<div class="jumbotron">

<h1>Hello, world!</h1>

<p>This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>

<p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>

</div>

</div>

### 9. 页头

页头组件能够为 h1 标签增加适当的空间，并且与页面的其他部分形成一定的分隔。

<div class="page-header">

<h1>Example page header <small>Subtext for header</small></h1>

</div>

### 10. 分页

#### 1）分页

Bootstrap 提供的分页由以下两组元素组成：

* <nav>元素作为分页的容器元素。
* 使用无序列表作为分页的成员元素，并为 <ul> 元素设置 class 为 pagination。

<div class="container">

<nav>

<ul class="pagination">

<li>

<a href="#" aria-label="Previous">

<span aria-hidden="true">&laquo;</span>

</a>

</li>

<li><a href="#">1</a></li>

<li><a href="#">2</a></li>

<li><a href="#">3</a></li>

<li><a href="#">4</a></li>

<li><a href="#">5</a></li>

<li>

<a href="#" aria-label="Next">

<span aria-hidden="true">&raquo;</span>

</a>

</li>

</ul>

</nav>

</div>

另外，通过为 <ul> 元素添加 .pagination-lg 或 .pagination-sm 类提供了额外可供选择的尺寸。

#### 2）翻页

用简单的标记和样式，就能做个上一页和下一页的简单翻页。用在像博客和杂志这样的简单站点上效果会很好。

<div class="container">

<nav>

<ul class="pager">

<li><a href="#">Previous</a></li>

<li><a href="#">Next</a></li>

</ul>

</nav>

</div>

## 插件

### 1. 概览

Bootstrap 提供相应插件在 HTML 页面中实现动态交互效果。

#### 1）前提条件

由于 Bootstrap 提供的插件都是依赖于 jQuery 的，所以必须要先引入 jQuery 文件。

Bootstrap 的每个插件都对应具有一个 JavaScript 文件，允许单独引入到 HTML 页面。也提供了一个完整版本（Bootstrap.js 或 Bootstrap.min.js 文件），允许一次性将所有插件全部引入到 HTML 页面中。

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

#### 2）两种使用方式

##### a. data 属性

通过 data 属性 API 就能使用所有的 Bootstrap 插件，无需写一行 JavaScript 代码。

<div class="container">

<div class="dropdown">

<a data-toggle="dropdown" class="btn btn-default" href="#">产品大全</a>

<ul class="dropdown-menu">

<li><a href="#">冰箱</a></li>

<li><a href="#">洗衣机</a></li>

<li><a href="#">电视</a></li>

</ul>

</div>

</div>

如果在某些情况下可能需要将此功能关闭，提供了关闭 data 属性 API 的方法（**即解除以 data-api 为命名空间并绑定在文档上的事件**）。

$(document).off('.data-api')

##### b. JavaScript API

所有 Bootstrap 插件提供了纯 JavaScript 方式的 API。所有公开的 API 都是支持单独或链式调用方式，并且返回其所操作的元素集合(**和jQuery的调用形式一致**)。

<div class="container">

<div class="dropdown">

<button id="btn2" class="btn btn-default">产品大全</button>

<ul class="dropdown-menu">

<li><a href="">冰箱</a></li>

<li><a href="">洗衣机</a></li>

<li><a href="">电视</a></li>

</ul>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

//var num = 1;

$('#btn2').dropdown();

</script>

### 2. 下拉菜单

#### 1）通过 data-\* 属性方式实现

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

#### 2）通过 JavaScript 编程方式实现

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('button[class\*=btn]').dropdown();

</script>

### 3. 警告框

#### 1）通过 data-\* 属性方式实现

<div class="container">

<div class="alert alert-danger alert-dismissible">

<span data-dismiss="alert" class="close">&times;</span>

<h4><span class="glyphicon glyphicon-alert"></span>警告！</h4>

<p>您的浏览器版本太老了！请更新到最新版本的浏览器！</p>

</div>

</div>

#### 2）通过 JavaScript 编程方式实现

<div class="container">

<div class="alert alert-danger alert-dismissible">

<span id="btn\_close" class="close">&times;</span>

<h4><span class="glyphicon glyphicon-alert"></span>警告！</h4>

<p>您的浏览器版本太老了！请更新到最新版本的浏览器！</p>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

//$('#btn\_close').alert(); //如此书写，将直接消失

$('#btn\_close').click(function(){

$(this).alert('close');

});

</script>

### 4. 标签页

#### 1）通过 data-\* 属性方式实现

<div class="container">

<ul class="nav nav-tabs">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

#### 2）通过 JavaScript 编程方式实现

<div class="container">

<ul class="nav nav-tabs" id="mytab">

<li class="active"><a href="#">Home</a></li>

<li><a href="#">Profile</a></li>

<li><a href="#">Messages</a></li>

</ul>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('#mytab a').click(function(event){

event.preventDefault();

$(this).tab('show');

})

</script>

### 5. 折叠框

#### 1）通过 data-\* 属性方式实现

<div class="container">

<a class="btn btn-primary" role="button" data-toggle="collapse" href="#collapseExample">

Link with href

</a>

<button class="btn btn-primary" data-toggle="collapse" data-target="#collapseExample">

Button with data-target

</button>

<div class="collapse" id="collapseExample">

<div class="well">

...

</div>

</div>

</div>

#### 2）通过 JavaScript 编程方式实现

<div class="container">

<a id="href" class="btn btn-primary" href="#collapse2">

Link with href

</a>

<button id="btn" class="btn btn-primary" data-target="#collapse2">

Button with data-target

</button>

<div class="collapse" id="collapse2">

<div class="well">

...

</div>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('#href,#btn').click(function(){

$("#collapse2").collapse('toggle');

});

</script>

#### 3）实现手风琴组件效果

##### a. panel 组件

* 通过 <div> 元素作为 panel 组件的容器元素，并设置 class 为 panel，以及一个修饰类。
* panel 组件是由一个标题（panel-heading）和一个主体（panel-body）组成。

<div class="container">

<div class="panel panel-default">

<div class="panel-heading">

<h4 class="panel-title">Collapsible Group</h4>

</div>

<div class="panel-body">

...

</div>

</div>

</div>

##### b. 动态的 panel 组件

想要实现动态的 panel 组件，需要 .collapse 和 .panel 两种组件组合而成。

<div class="container">

<div class="panel panel-default">

<div class="panel-heading" id="headingOne">

<h4 class="panel-title">

<a role="button" data-toggle="collapse" href="#collapseOne">

Collapsible Group Item #1

</a>

</h4>

</div>

<div id="collapseOne" class="panel-collapse collapse in">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

</div>

##### c. panel-group 组成手风琴

将 .panel-group 元素作为手风琴的容器元素，与 .panel 组件相关联。

<div class="container">

<div class="panel-group" id="accordion">

<div class="panel panel-default">

<div class="panel-heading" id="headingOne">

<h4 class="panel-title">

<a role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseOne">

Collapsible Group Item #1

</a>

</h4>

</div>

<div id="collapseOne" class="panel-collapse collapse in">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" id="headingTwo">

<h4 class="panel-title">

<a class="collapsed" data-toggle="collapse" data-parent="#accordion" href="#collapseTwo">

Collapsible Group Item #2

</a>

</h4>

</div>

<div id="collapseTwo" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwo">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" id="headingThree">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseThree">

Collapsible Group Item #3

</a>

</h4>

</div>

<div id="collapseThree" class="panel-collapse collapse">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

</div>

</div>