第 1 章　Web 开发概述

1.1　Web 开发前景与相关技术

随着移动互联网的快速发展，Web 开发技术被广泛应用，如 App 端、手机端、电脑端、微信端，市场需求非常旺盛。

目前 Web 开发主要包括前端和后台两部分，前端主要是指用户界面及其交互、动画的设计与实现，后台是指服务器端的功能逻辑、数据库的开发。

1.1.1　前端开发技术

前端使用的技术主要包括基础的 HTML、CSS、JavaScript，目前三者的主流版本分别是 HTML5、CSS3 和 ES6。

在实际开发中，会大量使用开发框架，其中轻量级框架开发中的佼佼者包括 jQuery、Bootstrap 等，使用这些开发框架可以大大地提升开发效率。

MVVM 的开发框架近年也得到了大量的应用，如 Vue. js、AngularJS、react 等。

前端工程化是前端实际项目的理想流程，常用的工程化的工具包括 webpack、gulp、grunt 等。

微信小程序、公众号的开发方法和前端开发方法非常相似，在实际工作中也是前端工程师的工作职责。

1.1.2　后台开发技术

后台开发技术为网站提供功能，是 Web 开发的核心。后台开发大多需要操作数据库，对数据库进行增删改查。

常用的后台开发技术包括 Java、PHP、微软. net、node. js、ruby、Python 等，后台常用的数据库包括 MySQL、Oracle、SQL Server、mongoDB 等。

后台代码和前端代码在代码层面的写法主要包括下列三种：

1. 后台代码和前端代码写在一起。

2. 后台代码和前端代码分离，使用接口交互，接口数据格式多为 JSON。

3. 后台代码渲染前端模板。

一般推荐后两种写法，尽量使前端和后台代码分离。避免前端代码和后台代码写在一

起,调试困难,逻辑复杂。

1.2 Web开发职业简介

Web开发在现实中有很多职位划分,常见的职位划分和说明如下:

1. 前端开发:移动端、响应式、PC端、微信端页面及页面交互、动画设计,接口解析。

2. 后台开发:后台逻辑设计、数据库操作。

3. UI:User Interface,用户界面设计。

4. UE:User Experience,用户体验设计。

5. 产品经理:需求分析、系统设计、数据库设计。

产品开发的一般流程是产品经理和相关人员根据用户需求完成系统原型设计,然后和用户确定需求并根据用户反馈意见完成需求分析的修改;UI设计师根据需求分析完成项目的UI设计;后台和产品经理根据需求设计数据库和接口;前端根据UI设计图和用户接口完成前端页面。

全栈工程师是指前端和后台工作等都能够胜任。

1.3 轻量级框架开发

在前端开发中,轻量级框架可以给开发者带来巨大的便利。

使用轻量级框架进行项目开发,可以大大降低和减少开发的难度和工作量。JavaScript的大量浏览器兼容的代码会被封装,开发者不必考虑,大量代码逻辑如DOM、事件等大大简化,很多基本的JavaScript效果被封装,常用的网页组件可以只通过写HTML的方式完成,其CSS和JavaScript大都是自动完成或只需要少量手工的固定代码的调用。

原生代码在学习阶段非常重要,但在实际的工程环境中,选择一个框架,尤其是不影响网页加载速度影响的轻量级框架是非常必要的。

jQuery和Bootstrap是优秀的JavaScript、CSS轻量级框架,它们可以使网站构建的过程更加简单、快速、高效,具备更好的浏览器兼容性和可扩展性。

1.4 响应式开发

用户用来访问网页的媒体(media)的类型越来越多,这些媒体的差异性非常大,最大的差异就是它们的视口(viewport)的宽度。

手机、平板电脑、笔记本电脑、大屏幕高分辨率的计算机,这些媒体的视口宽度差别很大,从几百像素(如320px)到几千像素(如2560px)都有。网页如何适应不同的视口宽度?

主流的解决方式有两种:

1. 使网页能够适应不同的视口宽度，响应视口宽度的变化。

2. 分别为移动端和 PC 端建立网页，分别完成网站的电脑版和手机版。

响应式开发是上面提到的第一种方案，它可以使网页在不同视口下有不同的显示效果，以使网页能更好地适应每个视口宽度。

响应式开发的本质是 media queries，即为每一个视口宽度写一套 CSS。

轻量级开发框架 Bootstrap 是开发响应式网站的有力工具，用户只需要编写 HTML 即可完成响应式网站的布局设计和组件设计，大大地降低了响应式开发的难度，具有广泛的适应性。

第 2 章　jQuery 基础

2.1　jQuery 概述

在编写 JavaScript 的时候，为了提高开发效率，降低开发难度，经常会用到 JavaScript 库，而不是原生的 JavaScript。常用的 JavaScript 库有 jQuery、Vue. js、React、anjularJS、mootools、Dojo、Yui、Prototype、jQuery 等，国内的一些公司如淘宝、百度等也公开了自己的 JavaScript 库。

jQuery 是一个快速、简洁的 JavaScript 库，支持 CSS3，能够简化阅读 HTML 文档、处理事件、实现动画以及向网页添加 AJAX 互动等过程，并且具有非常良好的浏览器兼容性。

jQuery 是被设计用于改进编写 JavaScript 的方式。jQuery 的设计目标是“write less，do more”，也就是说，通过 jQuery 可以写少量代码，完成多项功能。

总之，jQuery 有强大的功能和众多的优点：

1. jQuery 能够使网页的内容和行为相分离。

2. 简化 JavaScript 和 AJAX 编程。

3. 提供了强大的功能函数、解决浏览器兼容性问题、提供了丰富的 UI。

4. 方便地处理 HTML documents、events、实现动画效果。

5. 强大的链式操作。

6. 基于 jQuery 或语法类似 jQuery 的各种框架也得到了广泛的应用，如 Bootstrap、jQuery Mobile、zepto 等。

7. 具有丰富的插件，有很强的可扩展性。

jQuery 的缺点是不向后兼容，新的版本不能兼容早期的版本，有的插件只有老版本的 jQuery 的版本，使用多个插件时可能会因为版本出现冲突现象。

2.2　jQuery 的安装

jQuery 是一个独立的 JavaScript 文件，可以从 jQuery 官方网站下载它的各个版本，jQuery 的官方网站如图 2-1 所示，从网站上也可以获得 jQuery 的相关资源。

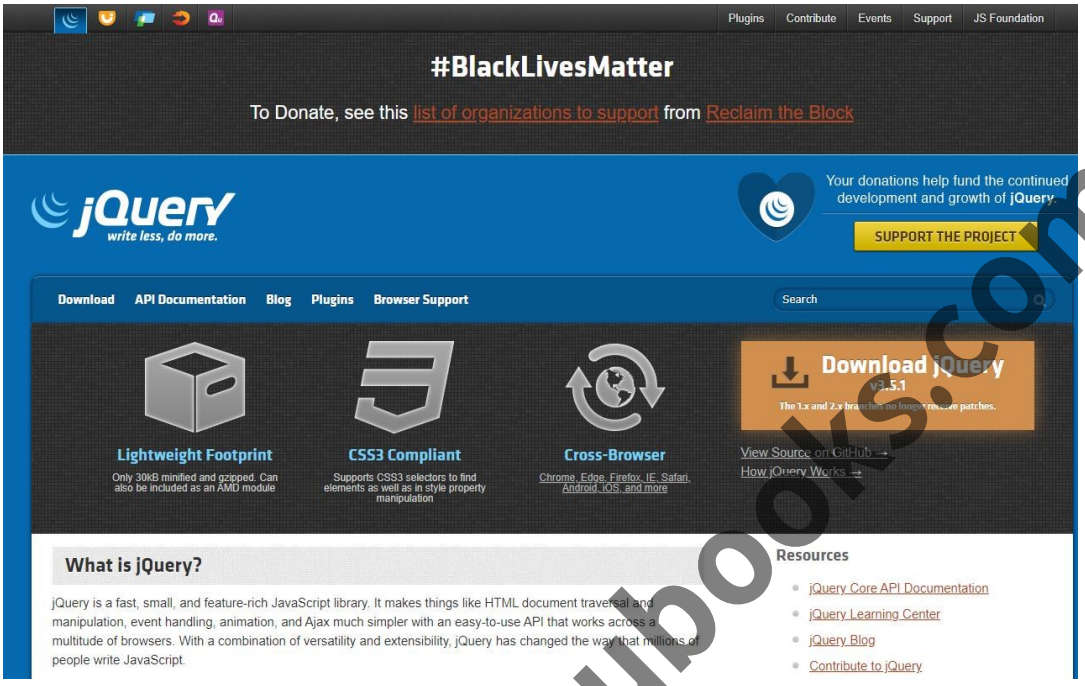

图 2-1　jQuery 官方网站页面

也可以使用 npm 等包管理工具安装 jQuery，或者使用 CDN 的方式。

jQuery 目前同时提供 3 个独立存在的版本，jQuery 1.0 的各个版本支持包括 IE6 在内的所有浏览器，jQuery 2.0 及 3.0 版本不再支持 IE6、IE7 和 IE8 浏览器。

如果要在网页设计的过程中使用 jQuery，只要在网页中包含 jQuery 文件即可。即需要将 jQuery 文件复制到网页所在的目录下，然后在网页中加入下列语句：

```
<script src="jquery.min.js"></script>
```

这样，就可以在写 JavaScript 的时候使用 jQuery 的方式书写代码。

2.3　ready 函数

JavaScript 的功能都需要在网页加载完成之后才被运行，在原生的 JavaScript 代码中，所有的功能都通过事件驱动或者在文件加载完成后被执行，文件加载的事件就是 onload。

但是当页面中多处都要使用 onload 时，onload 事件可能会产生冲突；尤其在使用一些外部库的时候，这种冲突非常难解决。

jQuery 引入了 ready()方法，它相当于 onload 函数，在网页加载完成之后运行，但是一个页面中如果有多个 ready()方法，也不会产生冲突。一般来说，jQuery 的代码都会写在 ready 方法中，这就是 jQuery 的程序框架。具体如下：

```
$(document).ready(function(){
  ...
})
```

上面的代码可以简写,省略(document).ready部分,具体如下:

```
$ (function(){
  ...
})
```

所有的jQuery代码都可以写在中间省略号的位置,这些代码会在网页加载完成后自动运行。

jQuery可以有多个ready函数。

【项目2-1】

【项目描述】

完成第一个jQuery页面,页面加载完成后执行alert和console.log,将页面背景设为蓝色,要求在ready函数中完成。

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>first jQuery</title>
</head>
<body>
  <script src="jquery.min.js"></script>
  <script>
    $(function () {
      alert("hello,jQuery!");
    });
    // ready的简写方式,可以有多个ready
    $(document).ready(function () {
      alert("hello,first jQuery!");
      console.log("Hello,world!")
      $("body").css("background", "blue");
    })
  </script>
</body>
</html>
```

【项目说明】

一个页面可以有多个jQuery ready函数,ready函数也有简化的写法,本项目分别使用两种ready函数。

出现错误或异常要查看浏览器的开发者工具的控制台的错误提示,如果控制台中出现错误提示:"Uncaught ReferenceError: $ is not defined",则表示jQuery文件没有正常引入成功,可以查看jquery.min.js文件和网页的相对位置,查看引入的script语句是否正确。

使用console.log可以在控制台中输出提示信息,便于调试错误。

也可以使用 CDN 的方式链入远程的 jQuery 文件，这样可以在正式发布网站后一定程度减轻服务器的负担。

$("body").css("background", "blue")是修改 body 的 CSS 属性，将 background 属性的属性值改为 blue。

2.4　jQuery 选择器

2.4.1　HTML DOM 结构

在正式学习 jQuery 和完成本章的项目之前，先对 HTML DOM 的基本结构强化一下，jQuery 选择器在很大程度上和 CSS 选择器很像，jQuery 选择器支持 CSS3 的各种选择器，理解了 CSS 选择器，就理解了 jQuery 选择器的大部分内容。

下列代码是一个简单的列表：

```
<ul>
  <li>北京</li>
  <li>上海</li>
  <li>沈阳</li>
  <li>天津</li>
  <li>大连</li>
  <li>青岛</li>
</ul>
```

阅读上述 HTML 代码，回答下列问题：

1. ul 有几个子元素(children)？

2. “
北京”有几个兄弟元素(siblings)？

3. “
大连”的下一个兄弟元素(next)是谁？上一个兄弟元素(pre)是谁？

4. li 的父元素(parent)是谁？

下面对上面的列表进行进一步的扩展，这也是一个非常经典的 HTML 结构。其中首页、项目、基本网页元素等对应的内容称为一级菜单，课程项目、四级项目、上图下文对应的内容称为二级菜单。

```
<div id="box">
  <ul id="menu">
    <li><a href="#">首页</a></li>
    <li><a href="#">项目</a>
      <ul>
        <li><a href="#">课程项目</a></li>
        <li><a href="#">四级项目</a></li>
      </ul>
    </li>
    <li><a href="#">基本网页元素</a>
      <ul>
```

```
            <li><a href="#">上图下文</a></li>
            <li><a href="#">列表</a></li>
            <li><a href="#">左图右文</a></li>
          </ul>
        </li>
        <li><a href="#">JS网页元素</a>
          <ul >
            <li><a href="#">tab选项卡</a></li>
            <li><a href="#">下拉菜单</a></li>
          </ul>
        </li>
        <li><a href="#">网站说明</a></li>
      </ul>
    </div>
```

阅读上面的代码,回答下列问题。

1. #box有几个子元素?

2. 第1个li有几个子元素?分别是什么?

3. 第2个li有几个子元素?分别是什么?

写出能够修饰下列内容的CSS选择器的名称:

1. 所有菜单

2. 所有一级菜单

3. 所有二级菜单

4. 第2个一级菜单

5. 第2个二级菜单

CSS2不足够能够完成上面的要求,需要通过CSS3的语法对其进行扩展,下面给出CSS3选择器的语法示例及说明:

1. #menu>li:所有#menu的子元素li,只是子元素,不包括后代元素。

2. #menu>li:nth-child(1):#menu的第一个li子元素。

上述CSS3的语法有一定的浏览器兼容性问题,不支持低级浏览器。

2.4.2 基础选择器

jQuery选择器和CSS选择器类似,包括ID选择器、类选择器、标签选择器、伪类选择器等。

下面给出了几个基本的jQuery选择器的例子。

1. $("#box"):ID选择器。

2. $(".classname"):类选择器。

3. $("div"):标签选择器。

4. $("li:odd"):奇数个li。

5. $("li:eq(1)"):第1个li,从0开始计数。

6. $("li:lt(2)"):小于2的li。

7. $("li:first"):第一个li。

“$”符号是 jQuery 选择器的符号，里面的内容就是要查找的 HTML 元素的描述，内容必须在单引号或者双引号内。一般可以按照 CSS 选择器的意义来解释，通过 HTML 元素对应的 ID、class 或者元素名称来查找 HTML 元素。

在 ID、Class 或者元素名称的后面可以添加条件，条件在冒号的后面。常用的条件有 odd(奇数)、even(偶数)、lt(小于)、gt(大于)、eq(等于)、first(第一个)、last(最后一个)。

下面分别对上面的部分 jQuery 选择器进行解释。

1. $("#box")：在网页中找到 ID 为 box 的 HTML 元素，相当于 JavaScript 中的 getElementById。

2. $(".classname")：在网页中找到 class 为 classname 的 HTML 元素。

3. $("div")：在网页中找到标签名称为 div 的所有 HTML 元素，相当于 JavaScript 中的 getElementsByTagName，找到的结果是网页中所有的 div。

4. $("li:odd")：在网页中找到所有的奇数个 li。

5. $("li:eq(1)")：在网页中找到第 1 个 li。

6. $("li:lt(2)")：在网页中找到小于 2 的 li，即第 1 个和第 0 个 li。

【注意事项】

odd、gt、eq、even、lt 是对冒号前面找到的元素内容进行筛选的条件，它们都是从第 0 个开始的。

2.4.3　基础方法

通过 jQuery 的选择器找到相应的 HTML 元素后，可以对找到的 HTML 元素进行相关操作，最常用的操作就是修改找到 HTML 元素的内容或者 CSS 样式，可以通过 html()、css()和 addClass()方法来完成这两个操作。具体示例代码及说明如下：

1. $("span").html("详细")：将网页中所有 span 的内容都改为“详细”。

2. $('div').css("background","#bfa")：将网页中所有的 div 的背景颜色都改为#bfa。

3. $("span").addClass("cur")：将所有 span 添加类 cur。

jQuery 选择器后可以跟一个或多个方法，方法执行的对象就是 jQuery 选择器找到的 HTML 元素。

【项目 2-2】

【项目描述】

项目的显示效果如图 2-2 所示。预先定义类选择器 cur(文字颜色为白色，深色背景)，使用 jQuery 完成下列功能。

1. 隔行变色：将奇数行都添加类。

2. 将第 3 行添加类。

3. 将所有行的内容都改成你所在省的省会的名称。

4. 将第 3 行以后都添加类。

5. 将前 3 行添加背景颜色。

6. 将第 4 行的内容改成你最喜欢的城市的名称。

图 2-2　addClass

每个功能的答案是一行jQuery代码。为了避免各个功能之间互相影响,完成一个功能后即对该功能进行注释,运行时只保留一个功能的代码。

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
  <title>jquery选择器</title>
  <meta charset="UTF-8">
</head>
<body>
  <style>
    .cur {
      background-color: #039;
      color: #fff;
    }
    li {
      font-size: 14px;
      line-height: 30px;
      background-color: #eef;
      height: 30px;
      width: 130px;
      border: 2px solid #039;
      text-align: center;
      list-style-type: none;
      margin: 10px;
      user-select:none;
    }
  </style>
  </head>
  <body>
    <ul>
      <li>北京</li>
      <li>太原</li>
      <li>沈阳</li>
      <li>苏州</li>
      <li>大连</li>
      <li>青岛</li>
      <li>济南</li>
      <li>西安</li>
    </ul>
    <script src="jquery.min.js"></script>
    <script>
      $(function () {
            $("li:odd").addClass("cur");
```

```
            // $("li:eq(2)").addClass("cur");
            // $("li").eq(2).addClass("cur");
             //$("li").html("长春");
             //$("li:gt(2)").addClass("cur");
            // $("li:lt(3)").addClass("cur");
            // $("li:gt(3)").html("长春");
          });
    </script>
  </body>
</html>
```

【项目说明】

按照以下步骤完成 jQuery 项目：

1. 完成 HTML 和 CSS。

2. 将 jQuery 文件复制到网页所在目录(文件夹)中，建议将 jQuery 文件放在单独的目录(如 js)中。

3. 在网页中引入 jQuery 文件，加入下列代码 <script src="jquery.min.js"></script>。

4. 在引入 jQuery 后，在 jQuery 的 ready 函数中写代码，如下所示：

```
<script>
    $(function(){
      ...
    })
</script>
```

5. 把上一步 jQuery 代码框架中的省略号替换为具体 jQuery 代码。

首先通过 jQuery 选择器找到需要操作的 HTML 元素，然后对找到的 HTML 元素执行对应的方法，这就是 jQuery 编程的一般思路。

2.4.4　选择器进阶

jQuery 选择器展现了 jQuery 强大的功能，学习 jQuery 选择器适合循序渐进，分步骤分层次学习，避免求多求全，导致影响 jQuery 后续内容的学习。

下面给出 jQuery 选择器中常用的伪类，大部分和 CSS 选择器含义相同，该类型是指冒号前面的部分，如 div::

1. :first：第一个元素。
2. :last：最后一个元素。
3. :not：去除某些选择器。
4. :even：索引为偶数的元素，从 0 开始。
5. :odd：索引为奇数的元素。
6. :eq：索引等于某个数值的元素。
7. :gt：索引大于某个数值的元素。
8. :lt：索引小于某个数值的元素。

9. :nth-child(n):父元素的第n个子元素且为该类型的元素。

10. :nth-of-type():父元素里的倒数第n个该类型的元素。

11. :nth-last-child(n):父元素的倒数第n个子元素且为该类型的元素。

12. :nth-last-of-type(n):父元素里的倒数第n个该类型的元素。

13. :only-child:父元素只有一个子元素且为该类型元素的HTML元素。

14. :only-of-type:父元素只有一个该类型的子元素的HTML元素。

jQuery的上下文选择器和CSS选择器含义完全相同,假设A和B为HTML元素,上下文选择器列举如下:

1. A B:后代选择器,所有A元素里的B元素。

2. A>B:所有的A元素的子元素的B元素。

3. A+B:A后面的那个兄弟元素B。

4. A~B:A后面的所有兄弟元素B。

jQuery的属性选择器也和CSS相同,以href属性,字符串value为例,常见属性选择器列举如下:

1. [href]:有href属性的HTML元素。

2. [href=value]:href属性为value的HTML元素。

3. [href!=value]:href属性值不为value的HTML元素。

4. [href^=value]:href属性值以value开头的HTML元素。

5. [href$=value]:href属性值以value结尾的HTML元素。

6. [href =value]:href属性值包含value的HTML元素。

表单相关的选择器列举如下:

1. :button:按钮

2. :input:input,textarea,select和button。

3. :text:单行文本框。

4. :password:密码框。

5. :radio:单选按钮。

6. :checkbox:复选框。

7. :submit:提交按钮。

8. :reset:重置按钮。

9. :file:文件域。

10. :hidden:隐藏域。

11. :image:匹配所有图像域。

12. :enabled:可用表单元素。

13. :disabled:禁用表单元素。

14. :checked:被选中的raido和checkbox。

15. :selected:被选中的select的option。

下面给出一些常用的具体的选择器实例和它们的说明:

1. $("li:odd"):奇数个li

2. $("li:nth-child(odd)"):同一个父元素下的奇数个 li,每个元素下面的 li 子元素都单独计算。

3. $("ul li a"):ul 下的 li 下的 a。

4. $("#box li a"):#box 里的 li 里的 a。

5. $("ul　li"):ul 里的 li,li 是 ul 的后代。

6. $("ul>li"):ul 里的 li,li 必须是 ul 的子元素。

7. $("li:has(a)"):li 元素,其后代中包括 a。

8. $("ul.list"): ul,且该 ul 上应用了 list 这个类。

9. $("span, #one"):span 和#one。

10. $("a[title]"):有 title 属性的 a。

11. $("li.yellow"): li,且该 li 上应用了 yellow 这个类。

在上面的实例中,$("li:eq(2) ")是指第 2 个 li,是找到的所有 li 中的第 2 个,找到的结果只有一个或者没有,这个 li 的计数是从 0 开始的,不区分元素的关系,只考虑其在 HTML 中出现的顺序。$("li:nth-child(2)")是指所有的父元素第 2 个且为 li 的子元素,找到的结果可能有多个。这种区别也是 $("li:nth-child(odd)")和 $("li:odd")的区别。

【项目 2-3】

【项目描述】

项目的显示效果如图 2-3 所示。首页、项目、基本网页元素、JS 网页元素、网站说明为一级菜单,其他为二级菜单。不改变原有的 HTML 和 CSS,使用 jQuery 改变一级菜单或者二级菜单。其中,改变的具体内容建议是改变对应菜单的背景颜色。所有的菜单都是 a 元素。

- 首页
- 项目
  - 课内项目
  - 课外项目
- 基本网页元素
  - 上图下文
  - 列表
  - 左图右文
- JS网页元素
  - tab选项卡
  - 下拉菜单
- 网站说明

图 2-3　jQuery 菜单

1. 改变所有的菜单。

2. 只改变一级菜单。

3. 只改变二级菜单。

4. 改变第 2 个一级菜单。

5. 改变第 1 个二级菜单。

6. 改变所有的二级菜单的第 1 项。

7. 改变所有有二级菜单的一级菜单。

每个功能的答案是一行 jQuery 代码。为了避免各个功能之间互相影响,完成一个功能后即对该功能进行注释,运行时只保留一个功能的代码。

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>jQuery 选择器</title>
</head>
```

```
<body>
    <ul id="menu">
      <li><a href="#">首页</a></li>
      <li><a href="#">项目</a>
        <ul>
          <li><a href="#">课内项目</a></li>
          <li><a href="#">课外项目</a></li>
        </ul>
      </li>
      <li><a href="#">基本网页元素</a>
        <ul>
          <li><a href="#">上图下文</a></li>
          <li><a href="#">列表</a></li>
          <li><a href="#">左图右文</a></li>
        </ul>
      </li>
      <li><a href="#">JS 网页元素</a>
        <ul>
          <li><a href="#">tab 选项卡</a></li>
          <li><a href="#">下拉菜单</a></li>
        </ul>
      </li>
      <li><a href="#">网站说明</a></li>
    </ul>
  <script src="jquery.min.js"></script>
  <script>
    $(function () {
      //改变所有的菜单:
      // $("#menu a").css("color", "#f00");
      //只改变一级菜单:$("#menu>ul>li>a").css("color","#f00")
      //只改变二级菜单:$("#menu li  li a").css("color","#f00")
      //改变第 2 个一级菜单:$("#menu>ul>li:eq(1)>a").css("color","#f00")
      //改变第 1 个二级菜单:$("#menu li li:eq(0)>a").css("color","#f00")
      //改变所有的二级菜单的第 1 项:$("#menu li li:nth-child(1)>a").css("color","#f00")
      //改变所有有二级菜单的一级菜单
      $("#menu li:has(ul)>a").css("color", "#f00");
      //如果控制台出现错误提示 $ is not defined,则表示 jquery 文件没有正确引入。
    });
  </script>
</body>
</html>
```

【项目说明】

jQuery 的要点在于 query 查找,就是通过 $ 符号快速地查找到要找的 dom 对象,jQuery 的功能非常强大,$ 符号里的内容的语法结构和 CSS 选择器的语法结构基本相同,

jQuery 支持 CSS3 的语法，但没有浏览器兼容问题。

本项目中的各个要求大多有多个答案，答案的关键在于对 HTML 结构的深入理解。

下面对 jQuery 选择器做一个相对全面的小结。

【项目 2-4】

【项目描述】

项目的显示效果如图 2-4 所示。上面有 7 个按钮，下面有若干个大盒子和小盒子。点击每个按钮，就会触发事件，事件的内容是选中给定的盒子增加类。

图 2-4　jQuery 选择器

7 个按钮的功能分别如下：

1. 改变所有的 box。
2. 改变所有的小盒子。
3. 改变第二个大盒子里的所有小盒子。
4. 改变所有小盒子里的文字的内容。
5. 改变所有的大盒子里的第二个小盒子。
6. 改变第三个大盒子里的第二个小盒子的内容。
7. 改变所有有小盒子的大盒子。

本项目中，点击每个按钮后需要刷新页面，不影响下一次按钮点击的结果。

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jquery 选择器</title>
    <style>
    .box,span{
        width:250px;
        height: 250px;
        border:1px solid #ccc;
        margin:10px;
        float:left;
    }
    .son{
        width:40 % ;
```

```
            margin:4%;
            height:100px;
            border:1px solid #ccc;
            float:left;
        }
        .cur{
            border-style:dotted;
            background:#dff;
        }
        </style>
    </head>
    <body>
        <button>改变所有的 box</button>
        <button>改变所有的小盒子</button>
        <button>改变第二个大盒子里的所有小盒子</button>
        <button>改变所有小盒子里的文字的内容</button>
        <button>改变所有的大盒子里的第二个小盒子</button>
        <button>改变第三个大盒子里的第二个小盒子的内容</button>
        <button>改变所有有小盒子的大盒子</button>
        <div id="jq">
          <div class="box">
          </div>
          <div class="box">
            <div class="son"></div>
            <div class="son"></div>
        </div>
        <div class="box">
            <div class="son"></div>
            <div class="son"></div>
            <div class="son"></div>
            <div class="son"></div>
        </div>
        <div class="box">
            <div class="son"></div>
            <div class="son"></div>
            <div class="son"></div>
            <div class="son"></div>
        </div>
        <span>span</span>
        </div>
        <script src="jquery.min.js"></script>
        <script>
          $(function(){
            $("button:first").click(function(){
```

```
            $(".box").addClass("cur");
        });
        $("button:eq(1)").click(function(){
            $(".son").addClass("cur");
        });
        $("button").eq(2).click(function(){
            $(".box:eq(1) .son").addClass("cur");
        })
        $("button:nth-child(4)").click(function(){
            $(".son").html("长春");
        })
        $("button:nth-child(5)").click(function(){
            $(".box .son:nth-child(2)").addClass("cur");
        })
        $("button:nth-child(6)").click(function(){
            $(".box:nth-child(3) .son:nth-child(2").addClass("cur");
        })
        $("button").eq(6).click(function(){
            $(".box:has(.son)").addClass("cur");
        })
    });
  </script>
</body>
</html>
```

【项目说明】

$("button").eq(2).click(function(){ })是 jQuery 的事件响应的写法，响应的代码写在匿名函数的内部。

eq(2)是一个方法，表示找到的结果里的第二个，也可以使用 $("button:eq(2)")实现。eq 是从 0 开始计数的。

click 是点击事件，对应着 JavaScript 里的 onclick。

【项目 2-5】

【项目描述】

项目的显示效果如图 2-5 所示。页面中有多个超链接，要求使用 jQuery 选择器选中满足给定要求的超链接，为选中的内容设置背景颜色。

要求都是独立的，项目的要求列举如下：

1. 选中 href 属性值中包含字母 a 的超链接。
2. 选中有 class 属性并且 href 的属性值中包含字母 a 的超链接。
3. 选中包含 href 属性值中包含 huan 的超链接和类 cur。
4. 选中不是 cur 类的超链接。

图 2-5 jQuery 属性选择器

每个功能的答案是一行 jQuery 代码。为了避免各个功能之间互相影响，完成一个功能后即对该功能进行注释，运行时只保留一个功能的代码。

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>jquery selector</title>
  <style>
    a {
      padding: 10px 50px;
      display: inline-block;
      color: #fff;
      background: #00f;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <a href="http://ltaaa.com" class="cur">龙腾</a>
  <a href="http://huanqiu.com">环球</a>
  <a href="http://baidu.com" class="cur">百度</a>
  <a href="http://sina.com.cn">新浪</a>
  <a href="http://jd.com">京东</a>
  <script src="jquery.min.js"></script>
  <script>
    $(function () {
      // $("a[href*=a]").css("background-color","#c00");
      // $("a[class][href*=a]").css("background-color","#c00");
      // $("a[href*=huan]").add(".cur").css("background-color","#c00");
      $("a:not(.cur)").css("background-color", "#c00");
    });
  </script>
</body>
</html>
```

【项目说明】

add 可以往已有的 jQuery 选择器选中的结果中增加新的元素。

href*=a 表示 href 的属性值中包含字母 a。

【项目 2-6】

【项目描述】

项目的显示效果如图 2-6 所示。点击按钮，文本输入框变为不可输入的 disabled 状态，按钮上的文字变为解禁；点击解禁按钮，文本输入框变为可以输入的状态，按钮上的文字变为禁用。

图 2-6　jQuery 表单选择器

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title> jquery form selecor</title>
</head>
<body>
    <div>姓 名:<input type="text"></div>
    <div>曾用名:<input type="text"></div>
    <input type="button" value="禁用">
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            $(":button").click(function () {
            if($(":button").val()=="禁用"){
            $(":text").attr("disabled", "disabled");
            $(":button").val("解禁");
            }
            else {
                $(":text").removeAttr("disabled");
                $(":button").val("禁用");
            }
            })
        })
    </script>
</body>
</html>
```

【项目说明】

attr()方法可以获得或者设置 HTML 属性,. val()方法可以设置表单控件(包括 button)的值,removeAttr()可以移除元素的某个 HTML 属性,:button 可以找到网页中的

所有按钮。

2.5 jQuery 方法

2.5.1 jQuery 基础方法

jQuery的选择器功能非常强大,通过选择器可以找到几乎所有逻辑上可以表达出来的东西;通过选择器找到要找的HTML元素,可以调用jQuery的各个方法来完成相关的功能。jQuery基础方法如表2-1所示。

表2-1　　常用jQuery基础方法

方法名称	方法含义	语法示例
size()	找到的HTML元素的数量	$("#box li").size()
html()	修改或获得找到的HTML元素的内部HTML,一个参数是获得,两个参数是修改	$(".num").html(4);
text()	修改或获得找到的HTML元素的内部的文字	$("#tip").text("展开")
css()	获得或设置HTML元素的CSS属性,一个参数是获得,两个参数是修改	$(".box").css("color","#ff0")
attr()	获得或修改HTML元素的属性	$("button").attr("disabled","true")
removeAttr()	删除HTML元素的属性	$('#button').removeAttr("disabled")
addClass()	为找到的HTML元素增加一个CSS类选择器,可以同时修改多个CSS属性	$("#box li").addClass("slected")
toggleClass()	切换类,如果找到的HTML元素有这个类,则移除;如果没有,则增加	$(this).toggleClass("current")
removeClass()	删除类	$(this).removeClass("selected");
hasClass()	检查当前的元素是否定义了某个类选择器,如果定义了就返回true	$("p:first").hasClass("cur")
each()	遍历选择器对象	$("li").each(function(index){})
add()	在之前选择器的基础上追加内容	$("div").add(".box)
not()	找到的HTML元素中进行筛选,不符合给定条件的	$("#box li").not(".selected")
filter()	对找到的HTML元素进行筛选	$("#box li").filter(".current")
find()	在找到HTML元素的后代元素中进一步查找	$(this).find("p")
children()	在找到HTML元素的子元素中进一步查找	$(this).children("p")
eq()	等于,在找到的HTML元素中找到第几个	$("#box li").eq(2)

（续表）

方法名称	方法含义	语法示例
first()	第一个元素	$("#box li").first()
last()	最后一个元素	$("#box li").last()
val()	获得或设置表单元素的值	$("input:first").val()
index()	正在操作的 HTML 元素的序号	$("#box li").index(this)
is	选择的成元素是否匹配选择器，如果匹配则返回 true	$(this).is("div")

下面通过项目进一步对这些方法的应用进行说明。

【项目 2-7】

【项目描述】

项目的显示效果如图 2-7 所示。点击增加按钮，数字会增加；点击减少按钮，数字会减少。

图 2-7　jQuery 表单控件

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <title>属性</title>
    <meta charset="UTF-8">
</head>
<body>
    <div id="box">
        <input id="num" name="num" type="text" size="3" value=0 disabled>
        <button>增加</button>
        <button>减少</button>
    </div>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            var num = $("#num").val();
            $("#box button:first").click(function () {
                $("#num").val(++num)
            })
            $("#box button:last").click(function () {
                if (num! = 0) {
                    num--;
                    $("#num").val(num);
                }
```

```
        })
    });
  </script>
</body>
</html>
```

【项目说明】

disabled属性可以使表单的控件无效，不能点击。

button是按钮的标签，它和input按钮都可以响应事件，button不是单标签，其内部可以包含其他HTML元素，显示效果更丰富。

button:first也可以写为button:eq(0)。

也可以允许文本输入框可以由用户输入数值，这时候需要判断用户的输入是否是数字，JavaScript提供了isNaN(val)方法来对判断一个变量是否是数字，如果不是数字，该方法将返回false。

【项目2-8】

【项目描述】

点击按钮，就可以在列表中添加新的菜名；如果输入为空，则不会有响应；每次添加成功菜名后，输入框都会清空，输入焦点也会在输入框，便于完成下一次输入。

具体显示效果如图2-8所示，具体代码框架如下：

图2-8 添加菜单

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>caidan</title>
</head>
<body>
   <input type="text" id="dishname">
   <button>添加菜单</button>
   <ul></ul>
    <script src="jquery.min.js"></script>
    <script>
```

```
    $(function(){
      $("button:first").click(function(){
          var sr= $("#dishname").val();
          if(sr! =""){
            $("ul").append("<li>"+sr+"</li>");
            $("#dishname").val("").focus();
          }
          });
    })
    </script>
</body>
</html>
```

【项目说明】

给#button 增加了 click 事件。

定义了变量 dn 存储#dishname 中输入的值。

如果 dn 不是空，那么就将其添加在列表的最后。

添加完毕后将#dishname 设为空，并使其获得焦点。

val()这个方法的意思就是获得或重置表单控件的取值，当其没有参数的时候，是获得表单控件的值，当其有参数的时候，是将表单的值设为它的参数对应的值。

2.5.2　jQuery DOM 方法

jQuery DOM 常用方法如表 2-2 所示。

表 2-2　　常用 jQuery DOM 方法

方法名称	方法含义	语法示例
append()	在找到的 HTML 元素的内部的最后增加内容	$(".list").append(" ")
appendTo()	将内容增加到指定 HTML 元素的内部的最后	$(" ").appendTo(".list")
prepend()	在找到的 HTML 元素的内部的最前面增加子元素	$(".list").prepend(" ")
after()	在每个匹配的 HTML 元素之后插入兄弟元素	$(".list"").after(" ")
before()	在每个匹配的 HTML 元素之前插入兄弟元素	$(".list"").before(" ")
insertAfter()	在所有指定 HTML 元素后为其增加兄弟元素	$(" ").insertAfter(".list")
insertBefore()	在所有指定 HTML 元素前为其增加兄弟元素	$(" ").insertBefore(".list")

（续表）

方法名称	方法含义	语法示例
replaceWith()	将所有匹配的元素替换成指定的HTML或DOM元素	$("p").replaceWith("hi");
replaceAll()	用匹配的元素替换掉所有selector匹配到的元素	$("hi").replaceAll("p");
empty()	删除匹配的元素集合中所有的子节点	$(this).empty();
remove()	从DOM中删除所有匹配的元素	$("div").remove();
clone()	克隆匹配的DOM元素并且选中克隆的元素	$("div").clone()
parent()	返回被选中的元素的直接父元素	$("p").parent()
parents()	获得包含所有匹配元素的祖先元素的集合(不包含根元素)	$("p").parents(".box")
next()	找到的HTML元素的下一个兄弟元素	$("#box li:odd").next()
prev()	找到的HTML元素的上一个兄弟元素	$("#box li:odd").prev()
siblings()	获得一个包含匹配的元素集合中每一个元素的兄弟元素集合	$("li").siblings(".cur")
prevAll()	匹配选中元素之前的所有的兄弟元素	$("#box h2").prevAll()
nextAll()	匹配选中元素之后的所有的兄弟元素	$("#box h2").nextAll()

【项目2-9】

【项目描述】

项目的显示效果如图2-9所示。点击网页上的指定部分，每次点击可以增加一个盒子，同时也可以改变盒子中的文字和特定位置的文字。

这是一个基本的jQuery事件响应的项目，主要技能点包括jQuery选择器、使用jQuery修改CSS或HTML等。

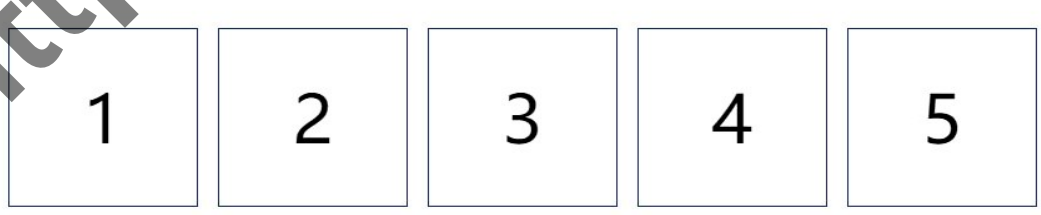

图2-9 点击增加盒子

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>jquery append</title>
    <style>
      .box{
          width:150px;
          height:150px;
          border:1px solid #039;
          float:left;
          margin:8px;
          text-align: center;
          line-height: 150px;
          font-size:56px;
      }
      body,html{
          height:100%;
          user-select:none;
              }
    </style>
</head>
<body>
    <h2>网页中一共有<span>0</span>个盒子</h2>
    <script src="jquery.min.js"></script>
    <script>
     $(function(){
        $("body").click(function(){
          var n= $(".box").length+1;
          var str="<div class='box'>"+n+"</div>";
           $("body").append(str);
          $("span").text(n);
        });
     })
    </script>
</body>
</html>
```

【项目说明】

该项目以下面思路来实现：

1. 每次点击时网页中增加一个盒子。

2. 每次点击时盒子内部的数字是不一样的，对应着该盒子的序号。

3. 标题中的数字每次点击的时候改变。

具体实现过程如下：

首先，增加 click 事件，点击的区域是 body。具体代码为 $("body").click()。

其次，点击 body 之后会增加一个新的盒子。在 click 方法的内部增加下列代码：$("

body").append(" 1 ")。append是一个方法,它的意思为在找到的HTML元素的内部的最后面增加一个子元素,append的内容最好是比较完整的HTML,而不单纯是字符。在本项目中,就会增加一个盒子,盒子的CSS已经预先定义好。

再次,将盒子里的内容改为序号。盒子里的内容都是1,可以将其改为盒子的序号,即$("div").length+1,length是一个属性,意为找到的div的个数,也可以用size()方法完成同样的功能。这样修改后的代码就变成了:

```
$("body").append("<div>"+ ($("div").length+1) + "</div>");
```

最后,修改上面的数字,使其和下面的div的个数相对应,数字是通过span标签修饰的,可以通过选择器找到span标签,然后对其执行相关的方法。代码为:$("span").html($("div").length)。

【项目2-10】

【项目描述】

项目的显示效果如图2-10所示。点击列表的每一项都会改变它的显示方式,再点击一下就会变回之前的显示样式,每次点击都会有反应。

通过CSS来表示列表项的显示效果,点击表单项的时候,就是在该li标签上添加或删除CSS样式。

点击每个列表项时,上方的文字都会提示点击的是第几个li。

双击每个列表项,可以删除它。

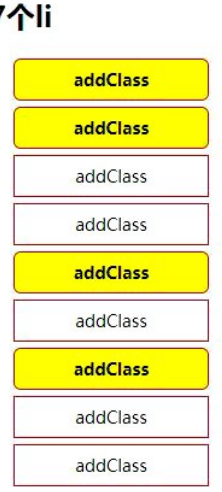

图2-10 toggleClass和remove

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>remove</title>
    <style>
    #list>li{
        width:160px;
        height:33px;
        border:1px solid #c00;
        line-height: 33px;
        text-align: center;
        font-size:14px;
        margin:5px;
        list-style:none;
        user-select:none;
    }
    #list>.hg{
        background-color: #ff0;
```

```
            font-weight:bold;
            border-radius: 6px;
        }
    </style>
</head>
<body>
    <h2>第<span>0</span>个 li</h2>
    <ul id="list">
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
        <li>addClass</li>
    </ul>
    <script src="jquery.min.js"></script>
    <script>
      $("#list>li").click(function(){
          $(this).toggleClass("hg");
      })
      $("#list>li").mouseover(function(){
          var index= $("#list>li").index(this)+1;
          $("h2 span").text(index);
      })
      //双击一个 li,该 li 被删除
      $("#list>li").dblclick(function(){
          $(this).remove();
      })
    </script>
</body>
</html>
```

【项目说明】

给所有的#list 里的 li 都增加了 click 事件。

点击 li 的时候只有被点击的 li 改变，而不是所有的 li 都改变；$(this)表示正在被点击的 li，它和事件定义函数前面的 $ 是有对应关系的。

index()方法可以获知鼠标正在点击的 div 是第几个，序号从 0 开始，而不是 1。index 前面的选择器原则上要求和事件定义前面的选择器相同。

方法 eq 的含义是等于，$(".list")eq(index)的含义是第 index 个.list。

给很多个 HTML 元素同时定义了事件，但是只想对鼠标点击的那个 HTML 元素进行事件响应可以使用 $(this)或者是 index 方法。

改变li的效果通过CSS样式中的类选择器hg完成,具体就是添加类和删除类。

点击某li的时候,如果该li已经有了CSS类hg的显示效果,则移除掉CSS类选择器hg;如果其没有类hg的显示效果,则为其增加该类选择器;通过toggleClass这个方法可以完成切换类的效果。toggleClass同时包含了removeClass(移除类)和addClass(增加类)以及条件判断。

dblclick是双击,双击使用remove方法删除该li。

2.6 jQuery事件

2.6.1 基础事件

jQuery的事件本质上和JavaScript的事件都是相同的,增加了一些能让代码更简练的事件,常用的jQuery事件如下:

1. click:鼠标单击,对应JavaScript的onclick。
2. mouseover:鼠标在对应元素上面,对应JavaScript的onmouseover。
3. mouseout:鼠标离开对应元素,对应JavaScript的onmouseout。
4. dblclick:双击,对应JavaScript的ondblclick。
5. mouseenter:和mouseover相似,子元素不触发该事件。
6. mouseleave:和mouseout相似,子元素不触发该事件。
7. blur::表单控件失去焦点。
8. change:表单控件内容改变,失去焦点时触发。
9. focus:表单控件获得焦点。
10. submit:表单提交。

初学阶段,重点掌握上述基本事件即可。事件相应的基本代码结构如下:

```
$('').click(function(){
    ...
}
```

上面的事件响应函数的定义是一个匿名函数,不必像原生JavaScript那样为响应的函数给定函数名称,jQuery的函数定义也不需要在HTML中添加任何的额外内容,做到了真正的内容(HTML)、样式(CSS)和行为(JavaScript)的分离。

jQuery的事件响应可以同时为多个HTML元素定义事件响应,不必像原生JavaScript那样使用任何循环结构。

jQuery的事件响应有多种定义方式,除了上面的基本结构还有其他定义方法。

同一个HTML元素可以绑定多个事件。

【项目2-11】

【项目描述】

项目的显示效果如图2-11所示。鼠标在右边的文字上移动的时候,左边的图片就会相

应变化，每一行文字对应一张图片。

图 2-11　jQuery 事件

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>mouseover</title>
    <style>
        #cont {
            width: 320px;
            border: 1px solid #999;
            height: 180px;
        }
        #left {
            float: left;
            height: 155px;
            width: 215px;
            padding: 10px 5px;
        }
        #right {
            height: 160px;
            width: 75px;
            float: right;
            margin-top: 10px;
            margin-right: 5px;
        }
        #left img {
            padding: 6px;
            border: 1px solid #999;
        }
        #right  a {
            text-decoration: none;
```

```
            display: block;
            font-size: 14px;
            line-height: 180%;
            color: #00F;
            text-align: center;
            border-bottom: 1px dashed #999;
        }
        #right a:hover {
            color: #F00;
            background-color: #9FF;
            font-weight: bold;
        }
        #right ul {
            margin: 0;
            list-style-type: none;
            padding: 0;
        }
    </style>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            $("#right li").mouseover(function () {
                var i = $("#right li").index(this);
                $("#left img").attr("src", "img/" + (i + 1) + ".jpg");
            });
        });
    </script>
</head>
<body>
    <div id="cont">
        <div id="left"><img src="img/1.jpg"></div>
        <div id="right">
            <ul>
                <li><a href="#">繁花似锦</a></li>
                <li><a href="#">锦绣山河</a></li>
                <li><a href="#">北国风光</a></li>
                <li><a href="#">原驰蜡象</a></li>
                <li><a href="#">橘子洲头</a></li>
                <li><a href="#">万山红遍</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

【项目说明】

给所有 #right 里的 li 都增加了 mouseover 事件。

使用 index 方法获得鼠标正在操作的 li 的序号，从 0 开始。

修改 #left 里的 img 的 src 属性，属性值是一个动态字符串，它的值就是图片的路径。

在这里使用了一个小技巧，即图片的名字恰好是 1.jpg、2.jpg 等，如果需要没有规律的图片名称，可以使用数组、xml 或者 json 存储图片名称。

可以在触发事件时进一步修改图片对应的超链接的值。

2.6.2　hover 事件

hover 是 jQuery 特有的事件，它的本质是同时定义 mouseenter 和 mouseleave 事件，mouseenter 和 mouseleave 事件也是 jQuery 特有的事件，在简单的场景下，可以把它们视为 mouseover 和 mouseout。

hover 事件的语法结构如下：

```
$("").hover(function(){
  ...
},function(){
  ...
});
```

【项目 2-12】

【项目描述】

项目的显示效果如图 2-12 所示。鼠标在图片上，会显示图片的提示文字，提示是透明的，可以显示出下面的图片被遮住的部分；鼠标离开图片，该提示就会消失。

图 2-12　jQuery hover

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hover</title>
    <style>
        p{
            margin: 0;
            padding: 0;
        }
        a {
            color: #06346F;
            font-size: 12px;
            text-decoration: none;
        }
```

```
        .box p {
            background-color:#999;
            bottom: 0px;
            font-size: 18px;
            font-weight: bold;
            height: 37px;
            line-height: 37px;
            text-align: center;
            width: 100%;
            position: absolute;
            left: 0;
            opacity: .8;
            display: none;
        }
        .box a{
            color: #fff;
        }
        .box {
            height: 200px;
            width: 200px;
            position: relative;
            z-index: 100;
            padding: 3px;
            border: 1px solid #CCC;
        }
    </style>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            $(".box").hover(function () {
                $(".box p").show();
            },
            function () {
                $(".box p").hide();
            });
        });
    </script>
</head>
<body>
    <div class="box">
        <img src="img/1.jpg" >
        <p><a href="#">工信部 web 前端职业认证</a></p>
    </div>
</body>
```

</html>

【项目说明】

给.box 增加了 hover 事件，hover 有两个参数，两个参数都是匿名函数，第 1 个函数是 mouseover 对应的事件，第 2 个函数是 mouseout 对应的事件，hover 是同时定义这两个事件的一种简单的写法。

一定要给.box 添加 hover 事件，而不是给 img 添加，img 上有绝对定位的 p 元素，会严重影响显示效果；p 元素默认是隐藏的。

透明的层需要考虑不同浏览器下的显示效果。

hover 的语法结构非常清晰，$(".box").hover(function(){},function(){})，就是给某 HTML 元素定义了一个方法，这个方法有两个参数，每个参数都是函数。

show()和 css("display",block)的显示效果相同，可以加参数，表示显示的时间，如果参数是数值，数值的默认单位是毫秒，如 show(500)表示在 500 毫秒(0.5 秒)时间内显示出来；hide()和 show()作用相反。

【项目 2-13】

【项目描述】

项目显示效果如图 2-13 所示。鼠标在图片上的时候，会显该图片的文字提示；鼠标离开图片，图片的文字提示就会隐藏。

这个项目的显示效果和上一个项目虽然很像，在实现上却必须要考虑到同时给多个图片定义事件的处理方法。

图 2-13　jQuery 多图 hover

【项目实现】

项目的 HTML 结构和 CSS 样式可参考上一个项目。jQuery 代码如下：

```
$(function () {
    $(".box").hover(function () {
        $(this).find("p").show();
    }, function () {
        $(this).find("p").hide();
```

```
    });
});
```

【项目说明】

给所有的.box 都增加了 hover 事件,要处理正在鼠标下面的.box,需要使用$(this)。

$(this)会找到.box,然后通过 find()这个方法找到.box 的后代元素 p,对其进行显示或隐藏;这种链式操作是 jQuery 常用的一种基本操作。

几个基本的链式操作和常用方法的例子如下:

1. $("div").children(".selected").css("color", "blue"):改变所有 div 的名字为.selected 的子元素的字体颜色为蓝色。

2. $("p").siblings(".selected"):所有 p 元素的叫作.selected 的兄弟元素。

3. $("p").prev():p 元素的前一个兄弟元素。

这里,兄弟元素是指有相同父元素的 HTML 元素。

2.6.3 事件的参数

jQuery 的事件的参数的含义和语法需要明确 jQuery 的版本。在最新版本中,和 JavaScript 一样,都是通过事件定义的参数 e 获得相关信息,可以获得基于网页、基于触发事件的元素的坐标信息等。

【项目 2-14】

【项目描述】

项目的显示效果如图 2-14 所示。鼠标在盒子上移动时,盒子内部可以显示鼠标位置相对于整个网页的 x 坐标和 y 坐标,相对于盒子的 x 坐标和 y 坐标。

图 2-14 jQuery 事件的参数

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>innerWidth</title>
    <style>
        body {
            padding: 0;
```

```
            margin: 0;
        }
        #box {
            width:60 % ;
            margin:30px auto;
            background: #c00;
            color:#fff;
            font-size:50px;
            padding:20px;
            height:500px;
            user-select:none;
        }
    </style>
</head>
<body>
    <div id="box"></div>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
           $("#box").mousemove(function(e){
               $("#box").html(e.offsetX+","+e.offsetY+"<br>"+e.pageX+" "+e.pageY);
               if(e.offsetX< $("#box").position().left+ $("#box").innerWidth()/2)
                 {
                      $("#box").css("background","#00f");
                 }
               else
               {
                  $("#box").css("background","#0f0");
               }
        });//end of mousemove
         });
    </script>
</body>
</html>
```

【项目说明】

定义 mousemove 方法以随时响应鼠标的移动，position()方法可以获得盒子在页面中的位置，offsetX 是相对于事件响应的元素的鼠标位置，pageX 是相对于页面的鼠标位置。

2.6.4　on 和 off

之前定义的 jQuery 事件的方法，都不能给动态添加的 HTML 元素定义事件，只能给网页加载完成时具有的 HTML 元素定义事件。

旧版本的 jQuery 曾经有过很多事件定义的方法，如 bind()、live() 和 delegate()，在 jQuery 1.7 版本之后，on()方法代替了之前的事件定义方法。使用 on()方法可以给任何的

HTML 元素定义事件，包括有 JavaScript 动态添加的 HTML 元素。使用 on 方法可以同时给 HTML 元素定义多个事件。

使用 off()方法移除事件，使用 one()方法定义只运行一次之后即移除的事件。

值得注意的是，on()方法可以给脚本添加的 HTML 元素定义事件，需要在 HTML 元素动态添加后再使用 on()方法为新添加的元素定义事件，而不是在 HTML 元素动态添加前就可以为其定义事件。

【项目 2-15】

【项目描述】

项目的显示效果如图 2-15 所示。使 on()方法为所有的盒子定义 mouseover 事件和 click 事件，点击按钮，解除所有盒子的 mouseover 事件绑定。

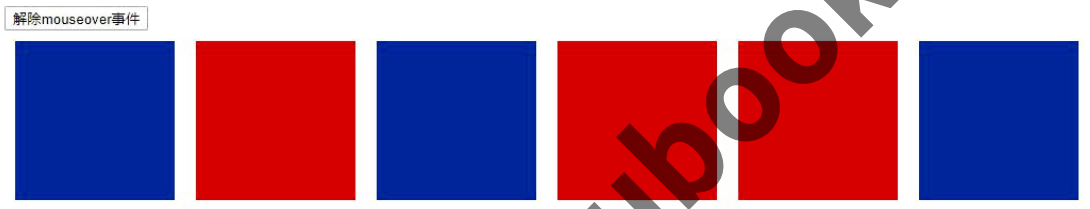

图 2-15 jQuery 的 on 和 off

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
  <title>on and off</title>
  <meta charset="UTF-8">
  <style>
    .box {
      width: 150px;
      height: 150px;
      text-align: center;
      line-height: 40px;
      font-size: 20px;
      background: #c00;
      margin: 10px;
      float:left;
    }
    .cur {
      background-color: #039;
    }
  </style>
</head>
<body>
  <div><button id="off">解除 mouseover 事件</button></div>
```

```
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<script src="jquery.min.js"></script>
<script>
  $(function () {
    $(".box").on("mouseover click", function () {
      $(this).toggleClass("cur");
    });
    $("#off").click(function () {
      $(".box").off("mouseover");
    });
  });
</script>
</body>
</html>
```

【项目说明】

toggleClass("cur")方法是切换类 cur，$(this)是指鼠标正在操作的定义事件的元素。使用 on()可以同时定义多个事件，使用 off()可以解除事件绑定。

2.6.5　事件冒泡和阻止默认行为

1. 事件冒泡

一个 HTML 元素可能和它的父元素上都定义了某个事件，在事件发生时，该 HTML 元素和它所有父元素上的相同事件都会按照由内及外的顺序被触发，这个触发的顺序就像“冒泡”一样，也可以通过具体的代码阻止这种“冒泡”，只执行子元素的事件，父元素的事件不被执行。

jQuery 中阻止事件冒泡的常用方法如下：

(1) JavaScript 方法：通过事件的参数 event 在事件响应函数里执行：event.stopPropagation()；

(2)jQuery 方法：在事件响应函数的最后执行：return false；

2. 阻止默认行为

有的 HTML 元素有默认事件，比如超链接 a 点击即跳转到 href 属性的属性值对应的页面。可以组织 HTML 的默认行为，jQuery 中阻止默认行为的方法如下：

JavaScript 方法：通过事件的参数 event 在事件响应函数里执行：event.preventDefault()

ljQuery 方法：在事件响应函数的最后执行：return false；

return false 和 preventDefault 有区别，后者可以触发父元素事件，不阻止冒泡。

【项目 2-16】

【项目描述】

项目的显示效果如图 2-16 所示。父元素是大的盒子，子元素是在父元素的右侧的小的盒子。点击父元素，父元素内的计数增加；点击子元素，子元素内的计数增加，但父元素的计数不增加。点击百度超链接，不跳转到百度，只是出现警告框。

图 2-16 jQuery 阻止冒泡和阻止默认行为

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <title>阻止冒泡和阻止默认行为</title>
    <meta charset="UTF-8">
    <style>
        #box {
            width: 600px;
            height: 600px;
            position: relative;
            margin: 20px auto;
            background-color: #c00;
            user-select:none;
            padding:10px;
        }
        #small {
            padding: 30px;
            height:30px;
            position: absolute;
            bottom: 45 % ;
            right: 0;
            background-color: #ff0;
        }
        #box a{
            color:#fff;
            text-decoration: none;
            font-size:60px;
            }
    </style>
</head>
<body>
    <div id="box">
        <h2>冒泡和阻止默认行为,点击次数<span>0</span></h2>
```

```
        <a href="http://baidu.com">百度</a>
        <div id="small">0</div>
    </div>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            var n = 0,
                m = 0;
            $("#box").click(function (event) {
                $("#box h2 span").text(++n);
                //event.stopPropagation();
                return false;
            });
            $("#small").click(function (event) {
                $("#small").text(++m);
               // event.stopPropagation();
              return false;
            });
            $("#box a").on("click",function(e){
                //阻止默认行为
                alert(this.href+m+n);
               e.preventDefault();
                //return false;
                            });
        })
    </script>
</body>
</html>
```

【项目说明】

点击作为大盒子 #box 的子元素的小盒子 #small 时，如果不阻止冒泡，由于 #box 和 #small 都定义了 click 事件，则两个 click 事件都会触发，先触发子元素的 click 事件，再触发父元素的 click 事件。如果阻止冒泡，则可以只触发子元素的 click 事件，使用 return false 即可阻止冒泡。

2.6.6　键盘事件

jQuery 键盘事件可以响应键盘的操作，获取用户的按键的值，根据用户的按键的值进行事件响应。常用的键盘事件如下：

1. keydown：按键被按下。

2. keypress：按键被按下并抬起。

keyup：键盘被按下后抬起。

在 JavaScript 中，通过事件参数 event 的属性 keycode 获取用户按键的值，在 jQuery 中，仍然可以使用这种方法，也可以使用 event 的 which 属性来获得用户按键的值，以获得更

好的浏览器兼容性。

keycode 或 which 的值即键盘按键的值，下面列出常见的按键的 keycode：

1. A：65，大写字母 A。
2. 0：96，数字 0。
3. Esc：27，ESC 键。
4. Right Arrow：39，右箭头。
5. Down Arrow：40，下箭头。
6. Delete：46，删除键。
7. Tab：9，Tab 键。
8. Spacebar：32，空格键。
9. Control：17，ctrl 键。
10. Enter：13，回车键。

【项目 2-17】

【项目描述】

项目的显示效果如图 2-17 所示。网页中有一个按钮，按钮有其响应事件，点击回车，也可以触发按钮的响应事件。

图 2-17 jQuery 键盘事件

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <title>key</title>
    <meta charset="UTF-8">
</head>
<body>
    <button id="login">登录</button>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            $(document).keypress(function (event) {
                if (event.which == 13) {
                    $("#login").click();
                }
            })
```

```
            $("#login").click(function(){
                alert("登录系统,按回车键也可以登录。");
            });
        });
    </script>
</body>
</html>
```

【项目说明】

$("#login").click()是指 $("#login")定义过的 click 事件,触发元素已经定义的事件还可以使用 trigger 方法,如 $("#login").trigger("click")。

【项目 2-18】

【项目描述】

网页中有一个小球(盒子),按键盘上的上、下、左、右键时,小球会分别往上、下、左、右移动,多次按键多次移动。

【项目实现】

```
<! DOCTYPE html>
<html>
<head>
    <title>key event</title>
    <meta charset="UTF-8">
    <style>
        #box{
            width:80px;
            height:80px;
            border-radius:50%;
            background-color:#c00;
            position:absolute;
            left:100px;
            top:100px;
        }
    </style>
</head>
<body>
    <div id="box"></div>
    <script src="jquery.min.js"></script>
    <script>
        $(function () {
            $(document).keydown(function (event) {
                var e = event || window.event;
                var k = e.keyCode || e.which;
                switch (k) {
                    case 37:
```

```
                $("#box").css({
                    'left': '-=8'
                });
                break;
            case 38:
                $("#box").css({
                    'top': '-=8'
                });
                break;
            case 39:
                $("#box").css({
                    'left': '+=8'
                });
                break;
            case 40:
                $("#box").css({
                    'top': '+=8'
                });
                break;
        }
        return false;
    });
})
</script>
</body>
</html>
```

【项目说明】

event || window.event可以获得事件参数，e.keyCode || e.which可以获得键盘按键的键值，使用||的写法可以获得更好的浏览器兼容性。

keyCode的值为37、38、39、40分别对应着键盘上的方向键的左、上、右、下。

盒子设置为绝对定位，改变其left和top改变盒子在网页中的位置。

2.6.7 window事件

【项目2-19】

【项目描述】

使用jQuery的方法，不使用CSS的方法定义个可以随着视口宽度变化而变化的盒子，盒子的宽度在变化的过程中，始终和视口的宽度和高度相同。

【项目实现】

```
<! DOCTYPE html>
<html lang="en">
<head>
```