目录

[#css 1](#_Toc55459249)

[#24 盒子模型 1](#_Toc55459250)

[#25 浮动 2](#_Toc55459251)

[#26 link和button的样式 4](#_Toc55459252)

[#27 自制导航栏 4](#_Toc55459253)

[#28 更改标签本身的类型（行内标签/块标签/行内块标签的转换） 5](#_Toc55459254)

[#29 定位（把一个元素定到想要放置的位置） 5](#_Toc55459255)

[#31 制作注册页面 7](#_Toc55459256)

[#markdown入门 8](#_Toc55459257)

[#javascript 11](#_Toc55459258)

[#js面向对象概念（项目：搭建一个简单的添加课程的界面） 11](#_Toc55459259)

[#js项目：心情日历 15](#_Toc55459260)

[#jQuery 16](#_Toc55459261)

[#简介及调用 16](#_Toc55459262)

#========

#css

#========

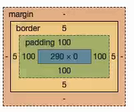
#========

#24 盒子模型

#========

1. 盒子模型的比喻

鼠标指向元素标签时会出现一个有四种颜色的盒子。这个盒子像是一个礼物。蓝色的content是礼物(**该元素的子标签集合**)，border是礼物包装，padding是二者之间的空隙。margin是礼物盒与礼物盒之间的间隙。



1. box-sizing: border-box；

如果设置width=500px，那么box-sizing可以让box从border算起，也就是这个礼物盒的总宽度是500px。如果不指定，则表示content的width为500px。

1. 去掉浏览器添加的属性

\*{

margin: 0;

padding: 0;

}

1. 全写/单写/简写

全写：

padding: 20px;

单写：

padding-top

padding-right

padding-bottom

padding-left

简写：

padding: 10px 20px;

#========

#25 浮动

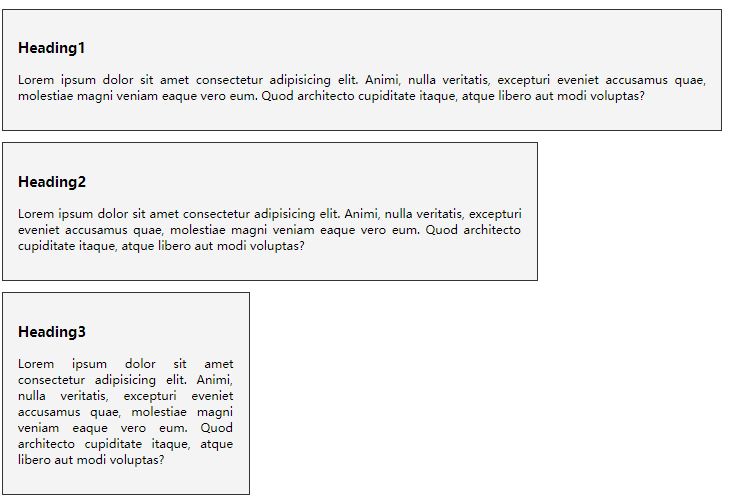
#========

1. max-width:960;

当浏览器缩小窗口时，内容的最大宽度也跟着改变。

1. 浮动

**功能：让它脱离文档流，去掉行标签左右的margin。**



如上图，两个**行标签(heading2和heading3)**的宽度总和为100%，让两个行标签放在同一行。

当style如下时，

#box-2 {

width: 60%;

float: left;

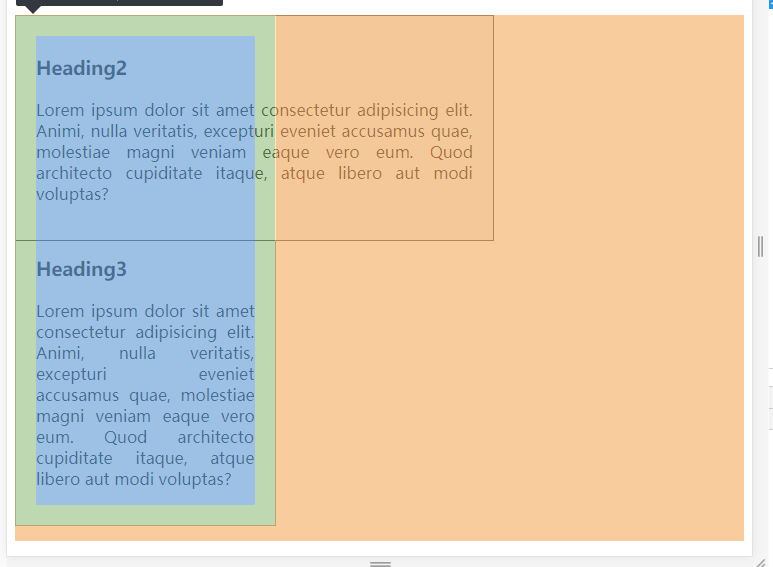
}

#box-3 {

width: 30%;

}

此时heading3的布局如下，也就是heading2相当于脱离了整体的布局，漂浮在整个布局上面。



box-3添加float属性后：

#box-2 {

width: 60%;

float: left;

}

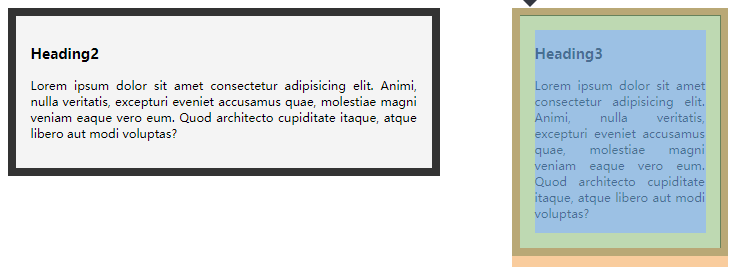
#box-3 {

width: 30%;

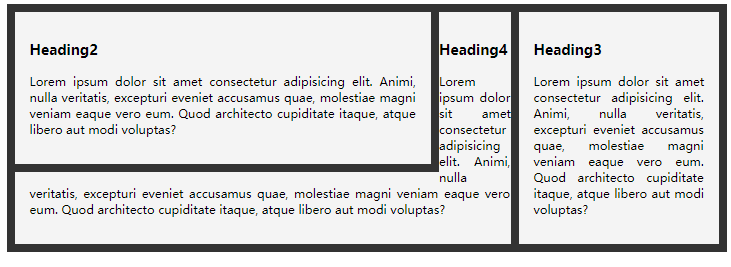
float: right;

}

可以看到float的作用是去掉行标签左右的margin



1. 清除浮动



<div class="clr"></div>

.clr {

clear: both;

}

#================

#26 link和button的样式

#================

a {

color:#333;

text-decoration:none;

}

a:hover {

color:coral;

font-size:120%;

}

a:visited {

color:red;

}

a:active {

color:green;

}

四种状态：

什么都不带的时候：静态

hover: 鼠标悬在上面

active: 鼠标点击的瞬间

visited: 鼠标点击后

a标签与button标签都常常包含链接，且能够被点击，所以二者在事件上比较相似。

#=========

#27 自制导航栏

#=========

**.navbar {**

list-style: none;

margin:0;

padding:0;

background:rgb(19, 9, 114);

border-radius:5px;

**overflow: auto;**

}

**.navbar li {**

**/\* 加上float后background消失，因此加上overflow \*/**

float:left;

}

**.navbar li a {**

display: block;

color:#fff;

text-decoration: none;

**/\* 添加padding后左右可以出现padding，上下没有padding。因为a是行内元素，上下padding是不起作用的，因此要改变他的display \*/**

padding:10px 20px;

}

**.navbar li a:hover {**

background:red;

color: yellow;

}

<ul class="navbar">

<li><a href="#">主页</a></li>

<li><a href="#">关于我们</a></li>

<li><a href="#">服务</a></li>

<li><a href="#">联系我们</a></li>

</ul>

1. **css书写逻辑**：
2. 可以从html文档最外层标签逐渐往里层标签写。.navbar 🡪 .navbar li 🡪 .navbar li a
3. 从静态到带事件。.navbar li a 🡪 .navbar li a:hover
4. **难点：**
5. 将块标签li转换成到一行。float:left;
6. padding对行内元素a的上下不起作用，需要将行元素转换成行内块元素。display:block;
7. overflow: **内容**溢出**元素框**时在对应的元素区间内添加滚动条

#===================================

#28 更改标签本身的类型（行内标签/块标签/行内块标签的转换）

#===================================

【参考】<https://www.cnblogs.com/cjeandailynotes/p/10811700.html>

1. **三种标签的特性**
2. 行内元素
3. 设置宽高无效
4. 对margin仅设置左右方向有效，上下无效；padding设置上下左右都有效，即会撑大空间
5. 不会自动进行换行

【总结】可以多个标签存在一行，**不能直接设置行内标签的高度、宽度、行高以及顶和底边距**，**完全靠内容撑开宽高**！

1. 块标签
2. 能够识别宽高
3. margin和padding的上下左右均对其有效
4. 可以自动换行
5. 多个块状元素标签写在一起，默认排列方式为从上至下
6. 如：<div> ，h系列，<li>，<dt>，<dd>，<p> ，<form>，<ul>
7. 行内块标签

结合的行内和块级的有点，不仅可以对宽高属性值生效，还可以多个标签存在一行显示：<img>，<input>。

1. 不自动换行
2. 能够识别宽高
3. 默认排列方式为从左到右
4. **三者的转化**

【参考】<https://blog.csdn.net/yudiandemingzi/article/details/54613234>

块级标签转换为行内标签：display:inline;

行内标签转换为块级标签：display:block;

转换为行内块标签：display：inline-block;

#=========================

#29 定位（把一个元素定到想要放置的位置）

#=========================

1. position 属性的五个值：

1. static: HTML **元素的默认值**，即没有定位，遵循正常的文档流对象。静态定位的元素不会受到 top, bottom, left, right影响。
2. fixed: 元素的位置相对于**浏览器窗口**是固定位置。即使窗口是滚动的它也不会移动。就像有些网站的小广告，怎么滑动滚动条，他们都是不动的。
3. relative：相对定位元素的定位是**相对其正常位置**。
4. absolute：绝对定位的元素的位置**相对于**最近的**已定位父元素**，如果元素没有已定位的父元素，那么它的位置相对于<html>。

**已定位：具有position属性的**

1. sticky: 基于用户的滚动位置来定位。粘性定位的元素是依赖于用户的滚动，**在 position:relative 与 position:fixed 定位之间切换**。滚动到这里的时候就停住了。

**#box-5 {**

**position:sticky;**

**top:20px;**

**background: orange;**

**}**



元素可以使用的顶部，底部，左侧和右侧属性定位。然而，**这些属性无法工作，除非是先设定position属性***！！！*他们也有不同的工作方式，这取决于定位方法。

.container {

**/\* 这里没有更改container的top和left属性，也就是虽然container添加了position的属性，但它相对于自己应该在的位置并没有变 \*/**

position: relative;

background: orchid;

width: 500px;

height: 500px;

}

#box-2 {

position: absolute;

top:100px;

left:100px;

background: yellow;

}

#box-3 {

position: absolute;

bottom: 50px;

background: green;

}

<div class="container">

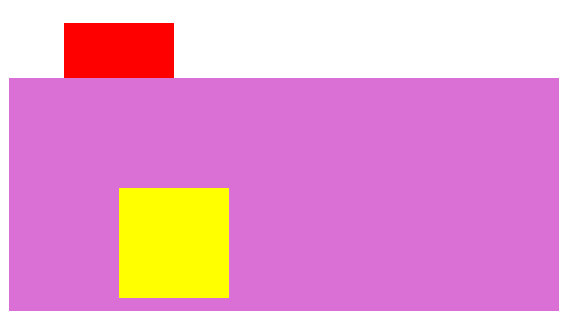
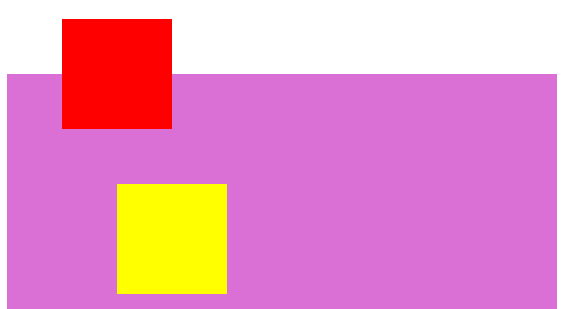
<div id="box-2" class="box"></div>

<div id="box-3" class="box"></div>

</div>

1. z-index

元素之间要是有堆叠，这个值可以指定谁在更上一层。如红色方块的位置。

【总结】

* position是决定元素在电脑平面的上的位置。z-index决定元素垂直于电脑平面的方向上的位置。
* 尽量不用，因为会脱离文档流，不利于响应式布局

#==========

#31 制作注册页面

#==========

**<style>**

**\* {**

**box-sizing:border-box;**

**margin:0;**

**padding:0;**

**}**

**body {**

**background: #344a72;**

**font-family:serif;**

**line-height: 1.8;**

**}**

**a {**

**text-decoration: none;**

**}**

**#container {**

**background: #fff;**

**max-width: 400px;**

**margin:30px auto;**

**padding:20px;**

**}**

**.form-wrap {**

**background: #fff;**

**padding:15px 25px;**

**}**

**.form-wrap h1,.form-wrap p {**

**text-align: center;**

**}**

**.form-wrap .form-group {**

**margin-top:15px;**

**}**

**.form-wrap .form-group label {**

**display: block;**

**color:#666;**

**}**

**.form-wrap .form-group input {**

**width: 100%;**

**padding:10px;**

**border: #ddd 1px solid;**

**border-radius: 5px;**

**}**

**.form-wrap button {**

**display: block;**

**width: 100%;**

**padding:10px;**

**margin-top:20px;**

**background:#49c1a2;**

**color:#fff;**

**cursor:pointer;**

**}**

**.form-wrap button:hover {**

**background: #37a08e;**

**}**

**.form-wrap bottom-text {**

**font-size:13px;**

**margin-top:20px;**

**}**

**footer {**

**text-align:center;**

**margin-top:10px;**

**}**

**footer a {**

**color:#49c1a2;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="container">**

**<div class="form-wrap">**

**<h1>注册</h1>**

**<p>注册成为网易云会员</p>**

**<form action="">**

**<div class="form-group"><label for="">姓名</label><input type="text" name="name" id="name"></div>**

**<div class="form-group"><label for="">性别</label><input type="text" name="sex" id="sex"></div>**

**<div class="form-group"><label for="">邮箱</label><input type="text" name="email" id="email"></div>**

**<div class="form-group"><label for="">密码</label><input type="text" name="password" id="password"></div>**

**<div class="form-group"><label for="">确认密码</label><input type="text" name="password\_again" id="password\_again"></div>**

**<button type="submit" class="btn">注册</button>**

**<p class="bottom-text">**

**<a href="#" class="">每一个 & 信息</a>完整**

**<a href="#" class="">我们保证不会透露你的信息</a>**

**</p>**

**</form>**

**</div>**

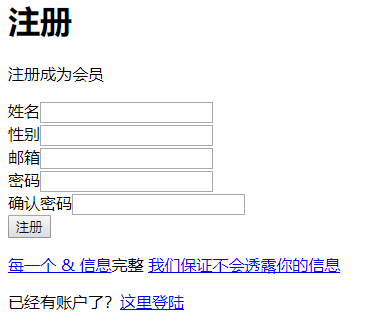
**<footer>**

**<p>已经有账户了？<a href="#" class="">这里登陆</a></p>**

**</footer>**

**</div>**

修改前：



修改后：



#==========

#markdown入门

#==========

1. 简介及vscode插件

轻量级标记语言。可以转换成html/xhtml和其他格式。常用的标记只有十几个。

vscode🡪Auto-Open Markdown Preview

1. 常用样式

<!-- 标题 -->

# 标题一

## 标题二

### 标题三

#### 标题四

<!-- 斜体 -->

我是\*斜体\*

我是\*\*加粗\*\*

我是~~删除线~~

---

<u>我是下划线</u>

我是脚注[^1]

[^1]:欢迎来到米修在线

> 我是引言

我是转义符号\

[我是链接 米修在线腾讯课堂](www.baidu.com)

我是无序列表：

- Item 1

- Item 2

- Nested Item 1

- Nested Item 2

我是有序列表：

1. Item 1

2. Item 2

3. Item 3

代码块：

`<p>我是段落文本</p>`

图片：

![markdown logo](https://markdown-here.com/img/icon256.png)

<!-- github中markdown特有功能 -->

<!-- 代码块 -->

```

import numpy

import pandas

```

```Javascript

function add(num1,num2){

return num1 + num2;

}

```

```python

def add(num1,num2):

return num1 + num2

```

<!-- 设置表格 -->

|姓名|邮箱|

|----|----|

|Summer|summer@163.com|

|Henry|henry@163.com|

<!-- 任务列表 -->

- [x] 任务1

- [x] 任务2

- [ ]任务3

#============

#javascript

#============

#==================================

#js面向对象概念（项目：搭建一个简单的添加课程的界面）

#==================================

【项目】搭建一个简单的添加课程界面：

1. 搭建html界面



可以直接引用bootstrap样式：

**<link href="https://cdn.bootcdn.net/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet">**

引入后：



###index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>课程列表</title>

    <link href="https://cdn.bootcdn.net/ajax/libs/twitter-bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

    <div class="container">

        <h1 class="text-center">添加课程</h1>

        <form action="" id="course-form">

            <div class="form-group">

                <label for="title">课程名称</label>

                <input type="text" id="title" class="form-control">

            </div>

            <div class="form-group">

                <label for="title">课程链接</label>

                <input type="text" value="https://imissu.ke.qq.com" id="link" class="form-control">

            </div>

            <div class="form-group">

                <label for="title">课程单号</label>

                <input type="text" id="isbn" class="form-control">

            </div>

            <div>

                <input type="submit" value="提交" class="btn btn-block btn-primary">

            </div>

        </form>

        <table class="table mt-4">

            <thead>

                <tr>

                    <th class="th">课程名称</th>

                    <th class="th">课程链接</th>

                    <th class="th">课程单号</th>

                    <th class="th"></th>

                </tr>

            </thead>

            <tbody id="course-list"></tbody>

        </table>

    </div>

    <script src="app.js"></script>

</body>

</html>

1. 实现三个功能
2. 实现在表单中添加的内容出现在下方的表格中

###app.js

// es5的时候没有class，可以使用prototype

// **课程构造函数，相当于把所有从form表单中提取的数据放到一个”类“中，也就是把数据变得更规矩**

function Course(title,link,isbn){

    this.title = title;

    this.link = link;

    this.isbn = isbn;

}

// UI构造函数

function UI(){} //**UI中什么都不用写，可以用prototype追加方法**

//当前的界面做出哪些反应

// 添加方法

UI.prototype.addCourseToList = function(course){

    // 想要将数据插入到tbody中，所以先获取"course-list"

    const list = document.getElementById("course-list");

    // 在tbody中追加一个tr元素

    const row = document.createElement("tr");

    // tr元素有了，还要在元素中添加数据

    row.innerHTML = `

        <td>${course.title}</td>

        <td><a href="${course.link}">查看</a></td>

        <td>${course.isbn}</td>

        <td><a href="#">X</a></td>

    `; // 使用模板语法

    // 将row中的内容追加到"list"变量中

    list.appendChild(row);

}

// 添加Submit时间

document.getElementById("course-form").addEventListener("submit",function(e){

        //e是事件对象

        //获取course-form这个元素，对它添加一个submit事件，点击”提交“时，触发function函数

    //点击事件发生后首先要获取表单中的数据

    const title = document.getElementById("title").value,

        link = document.getElementById("link").value,

        isbn = document.getElementById("isbn").value;

        console.log(title,link,isbn);

    //让当前的内容插入到DOM中，虽然可以直接写在函数中，但是最好写成面向对象的形式。

    //es5中没有class，但有prototype

    //实例化Course

    const course = new Course(title,link,isbn);

    //实例化UI

    const ui = new UI();

    ui.addCourseToList(course);

    e.preventDefault(); //阻止默认事件

})

【js基本逻辑】

1. 明确想要完成什么功能：实现在表单中添加的内容出现在下方的表格中
2. 点击什么位置，触发什么事件
3. 获取哪个元素的哪些数据，整理input数据有哪些，并构造出一个“类”，这样更清晰
4. 构建UI类，类中定义当前UI界面可以做出哪些反应，想要把input数据放在哪个元素的哪个位置(父元素?)，以什么元素的形式呈现(div/添加css样式)。
5. 表单整体形成，但还需要对填入的数据进行校验，如果形式不正确需要进行提醒
6. 删除某一行课程信息。
7. 实现迁移：将es迁移到es6
8. function构造函数 ==》class
9. class中有constructor这样的构造函数。

class Course {

**constructor(title,link,isbn){**

this.title = title;

this.link = link;

this.isbn = isbn;

}

}

// UI类

class UI {

**addCourseToList(course){**

// 想要将数据插入到tbody中，所以先获取"course-list"

const list = document.getElementById("course-list");

// 在tbody中追加一个tr元素

const row = document.createElement("tr");

// tr元素有了，还要在元素中添加数据

row.innerHTML = `

<td>${course.title}</td>

<td><a href="${course.link}">查看</a></td>

<td>${course.isbn}</td>

<td><a href="#" class="delete">X</a></td>

`; // 使用模板语法

// 将row中的内容追加到"list"变量中

list.appendChild(row);

}

}

1. es6中可以实现缓存方法

// es5的时候没有class，可以使用prototype

// 课程 类 ，相当于把所有从form表单中提取的数据放到一个”类“中，也就是把数据变得更规矩

class Course {

constructor(title,link,isbn){

this.title = title;

this.link = link;

this.isbn = isbn;

}

}

// UI类

class UI {

addCourseToList(course){

// 想要将数据插入到tbody中，所以先获取"course-list"

const list = document.getElementById("course-list");

// 在tbody中追加一个tr元素

const row = document.createElement("tr");

// tr元素有了，还要在元素中添加数据

row.innerHTML = `

<td>${course.title}</td>

<td><a href="${course.link}">查看</a></td>

<td>${course.isbn}</td>

<td><a href="#" class="delete">X</a></td>

`; // 使用模板语法

// 将row中的内容追加到"list"变量中

list.appendChild(row);

}

showAlert(message,className){

// 要展示的数据放在什么元素里

const div = document.createElement("div");

// 给上面的元素添加一个子元素，子元素是文本

div.appendChild(document.createTextNode(message));

// 为要展示的元素添加样式，样式在indexl.html的style标签中

div.className = `alert ${className}`; //alert也是一个类

// 定位到想要插入的位置

// 获取父级元素

const container = document.querySelector(".container");

// 获取表单

const form = document.getElementById("course-form");

// 想要把div插入container和form中，插入DOM

container.insertBefore(div,form);

// 清除弹窗内容

setTimeout(function(){

document.querySelector(".alert").remove();

},2000);

}

clearFields(){

document.getElementById("title").value = "";

document.getElementById("isbn").value = "";

}

deleteCourse(target){

if (target.className == "delete"){

**// 找到父级元素(td)的父级元素(tr)，并删除**

**// 假设添加了三个课程虽然她们的e.target内容打印出来都是一样的，且className也都为"delete"，但是其实e中记录着其他的信息，这就是为什么只有被点击的行被删掉了，而其他两行没有被删掉的原因。**

target.parentElement.parentElement.remove();

// 弹窗提示

}

}

}

// 缓存类

class Store {

// 静态方法，取出缓存

static getCourses(){

let courses;

if (localStorage.getItem('courses') == null){

courses = [];

}else{

courses = JSON.parse(localStorage.getItem('courses'));

// 难点：courses中的内容是个字符串，JSON.parse：双引号包裹的列表->列表

// localStorage.getItem("courses")的内容为：

// "[{"title":"的发射点","link":"https://imissu.ke.qq.com","isbn":"十大"},{"title":"富士达","link":"https://imissu.ke.qq.com","isbn":"的说法啊"}]"

// 也就是courses的值是一个被转换成字符串形式的列表，列表的每个元素是一个字典

}

return courses;

}

**// 展示缓存**

static displayCourses(){

const courses = Store.getCourses();

courses.forEach((course) => {

// 新知识点：forEach的用法

const ui = new UI();

ui.addCourseToList(course);

})

}

// 存入缓存

static addCourse(course){

const courses = Store.getCourses(); //获取localStorage中的courses，如果有则返回courses，没有则返回空列表

courses.push(course);

localStorage.setItem('courses',JSON.stringify(courses));

// 难点：需要将列表转换成字符串后才能作为键值对中的值存入字典中

}

// 删除缓存

**// 如果只不删除缓存中的数据，则在刷新的时候，缓存中的数据会被displayCourses数据重新加载出来**

static removeCourse(isbn){

const courses = Store.getCourses();

courses.forEach((course,index) => {

// 新知识点：course,index相当于python中的enumerate出的内容

if (course.isbn == isbn){

courses.splice(index,1);

}

});

// 删掉缓存列表中的一个数据后，还要把courses重新存入

localStorage.setItem("courses",JSON.stringify(courses))

}

}

// 为了展示缓存中的内容，添加DOM加载事件

document.addEventListener("DOMContentLoaded",Store.displayCourses);

// 添加Submit事件

document.getElementById("course-form").addEventListener("submit",function(e){

//e是事件对象

**//获取course-form这个元素，对它添加一个submit事件，点击”提交“时，触发function函数**

//点击事件发生后首先要获取表单中的数据

const title = document.getElementById("title").value,

link = document.getElementById("link").value,

isbn = document.getElementById("isbn").value;

console.log(title,link,isbn);

**//让当前的内容插入到DOM中，虽然可以直接写在函数中，但是最好写成面向对象的形式。**

**//es5中没有class，但有prototype**

//实例化Course

const course = new Course(title,link,isbn);

//实例化UI

const ui = new UI();

// 检验填入的数据是否符合要求

if (course.title == '' || course.isbn == ''){

// 弹出提醒

ui.showAlert("请填写内容！","error");

}else{

// 添加内容

ui.addCourseToList(course);

// 添加缓存

Store.addCourse(course);

// 弹出提醒

ui.showAlert("添加成功！","success");

// 清除表单

ui.clearFields();

}

e.preventDefault(); //阻止默认事件

})

// 点击X号删除课程

// 对整个course-list添加click事件，这样它的所有子元素都可以触发click事件

document.getElementById("course-list").addEventListener('click',(e) => {

**// es6中可以使用箭头符号代替function关键字**

console.log(e); // e是事件,是一个字典，包含非常多的信息

console.log(e.target); **// target是e的一个属性，但是在e的字典中本身也是一个复杂的内容，**

// 只是e.targetdayinchulai

//e.preventDefault();

// 初始化UI

const ui = new UI();

// 删除

ui.deleteCourse(e.target);

ui.showAlert('删除成功！','success');

console.log(e.target.parentElement.previousElementSibling);

Store.removeCourse(e.target.parentElement.previousElementSibling.textContent);

})

#============

#js项目：心情日历

#============

#======

#jQuery

#======

#==============================

#Let’s have fun 场景（在原有DOM上添加样式或效果）

#==============================

可以快速实现js代码。jQuery里面封装了js代码，是js的一个库(比插件体积大)。

**js为什么能改变网页，最核心的就是js能拿到DOM，然后对DOM操作。**

[www.bootcss.com](http://www.bootcss.com) 可以获取jquery库。

1. 搭建场景并利用bootstrap等库添加css样式
2. jquery实现的内容

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Document</title>

**<!-- 最新版本的 Bootstrap 核心 CSS 文件 -->**

**<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">**

**<!-- 可选的 Bootstrap 主题文件（一般不用引入） -->**

**<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">**

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

**<script src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>**

**<!-- 本地引用 -->**

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

**<!-- cdn引用 -->**

**<!-- <script src="https://...."></script> -->**

</head>

<body>

<div class="container">

<div class="jumbotron">

<h1>Let's Have Fun!</h1>

<button id="btn1" class="btn btn-primary">按钮1</button>

<button id="btn2" class="btn btn-success">按钮2</button>

<button id="btn3" class="btn btn-info">按钮3</button>

<button id="btn5" class="btn btn-warning">按钮4</button>

</div>

<div class="row">

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

<div id="panel1" style="display: none;" class="panel panel-primary">

<div class="panel-heading">

#panel1

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel2" style="display: none;" class="panel panel-primary">

<div class="panel-heading">

#panel2

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel3" style="display: none;" class="panel panel-primary">

<div class="panel-heading">

#panel3

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel4" style="display: none;" class="panel panel-primary">

<div class="panel-heading">

#panel4

</div>

<div class="panel-body">

content

</div>

</div>

</div>

</div>

</div>

<script>

// 整个DOM加载出来之后执行。美元符号可以获得某一个标签。ready是一个事件？？。jquery中封装的是：

// function $(id){

// return document.getElementById(id);

// }

// $('panel1');

$(document).ready(function(){

**// 一. 添加动画**

// hide是一个方法；

$('#panel1').hide(300).show(300);

$('#panel1').slideUp(1000).delay(1000).slideDown(1000);

// Toggle:如果当前是In，则让它out，如果当前是out，则让它in；toggle简化了代码；

$('#panel1').fadeToggle(1000).fadeToggle(1000);

// toggle跟hide/show相似

$('#panel1').toggle(1000).toggle(1000);

**// 二. 添加样式css**

$('#panel1').css({

color:'red',

fontWeight:'bold',

// display:none,

});

$('#panel2').css({

opacity:'0.5',

})

**// 三. 改变元素内容**

// $('#panel1').html("what's going on <strong>Baby!</strong>");

$('#panel1 .panel-body').html("new content");

});

**// $(document).ready(function(){})简化的写法：**

$(function(){

**// add Event**

$('#btn1').on('click',function(){

$('#panel1').fadeToggle();

});

$('#btn1').on('mouseenter',function(){

$('#panel1').toggle();

})

});

</script>

</body>

</html>

【使用jquery的逻辑和目的】

1. 对what元素执行呢？

利用$()方法获取要被修改的元素标签。

1. 在when情况下执行呢？

用on方法添加事件。$('#btn1').**on('click',function**(){$('#panel1').fadeToggle();});

1. how样的动作?

与js相似，利用jquery对文档进行的操作主要有三个：

1. **添加动画。**如$(‘#panel’).fadeToggle()/slideToggle()/hide()/show()
2. **添加css样式。**$(‘#panel’).**css({})**
3. **添加/改变元素内容。**$(‘#panel .panel-body’)**.html()**
4. **动态获取元素**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Document</title>

<!-- 最新版本的 Bootstrap 核心 CSS 文件 -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- 可选的 Bootstrap 主题文件（一般不用引入） -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

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

<script src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

<!-- 本地引用 -->

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

</head>

<body>

<div class="container">

<div class="jumbotron">

<h1>Let's Have Fun!</h1>

**<!-- 【技巧】**

**1. 去掉了id属性，然后添加了data-panelId这个属性，这个属性的名称是随意起的。**

**2. 添加了panel-button类，这样可以直接对该类添加相同的点击事件 -->**

<button data-panelId='panel1' class="panel-button btn btn-primary">按钮1</button>

<button data-panelId='panel2' class="panel-button btn btn-success">按钮2</button>

<button data-panelId='panel3' class="panel-button btn btn-info">按钮3</button>

<button data-panelId='panel4' class="panel-button btn btn-warning">按钮4</button>

</div>

<div class="row">

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

<div id="panel1" class="panel panel-primary">

<div class="panel-heading">

#panel1

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel2" class="panel panel-primary">

<div class="panel-heading">

#panel2

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel3" class="panel panel-primary">

<div class="panel-heading">

#panel3

</div>

<div class="panel-body">

content

</div>

</div>

</div>

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

<div id="panel4" class="panel panel-primary">

<div class="panel-heading">

#panel4

</div>

<div class="panel-body">

content

</div>

</div>

</div>

</div>

</div>

<script>

$(function(){

**// 为每个button添加点击事件**

$('.panel-button').on('click',function(){

**// 如何确定点击的是哪一个按钮?现在无法判断，所以需要一个动态的东西拿到点击的按钮是谁。**

**// js中是this，juery中用$(this).点击的是谁，this就能拿到谁。**

**// attr方法可以获得自定义属性**

**// 【重点】动态获取被点击的button的自定义属性值。**

**var panelId = $(this).attr('data-panelId');**

// 进行操作

$('#'+panelId).toggle();

});

});

</script>

</body>

</html>

【总结】

1. 不同🡪同。当相同按钮很多且实现的功能相似的时候，可以为按钮添加相同的类，这样可以一次性对所有类添加相同事件和动作。
2. 求同存异。每个按钮对应了不同的panel的动作。为button添加相同的属性，然后动态获取button的自定义属性值。$(this)
3. 最简单地表示两个标签的唯一性的方法就是为其赋予id值

#======================

#操作DOM，实现元素的增删查改

#======================

1. **# html中的样式**

<ul class="list">

<li>one</li>

<li class="special">two</li>

<li>three</li>

<li>

<ul class="sublist">

<li>1</li>

<li>2</li>

<li>3</li>

</ul>

</li>

</ul>

1. **# 控制台中的结果**

> $('li')

S.fn.init(7) [li, li.special, li, li, li, li, li, prevObject: S.fn.init(1)]

> $('li:first')

S.fn.init [li, prevObject: S.fn.init(1)]

> $('li:first').parent() // 父节点

S.fn.init [ul.list, prevObject: S.fn.init(1)]

> $('li:first').parent().parent() // 链式

S.fn.init [body, prevObject: S.fn.init(1)]

> $('li').parent()

S.fn.init(2) [**ul.list, ul.sublist,** prevObject: S.fn.init(7)]

// 7哥’li’有两种父节点

> $('li')**.eq(3) //可以直接找到第i个’li’**

S.fn.init [li, prevObject: S.fn.init(7)]0: lilength: 1prevObject: S.fn.init(7) [li, li.special, li, li, li, li, li, prevObject: S.fn.init(1)]\_\_proto\_\_: Object(0)

> $('li').eq(3).siblings() // 兄弟节点

S.fn.init(3) [li, li.special, li, prevObject: S.fn.init(1)]