Day01笔记

|  |
| --- |
| jQuery  **1. 初识jQuery**  **jQuery简介**  jQuery是一个轻量级的JavaScript库，主要功能包括元素查询、事件机制、DOM操作、属性操作、工具方法和Ajax。它对原生JavaScript（ECMAScript）、DOM和BOM进行了简化封装，使开发者能够更容易地操作网页元素。  **jQuery安装**  可以通过下载或使用CDN引入jQuery库，有两个版本：`jquery.js`（未压缩版，适合源码研究）和`jquery.min.js`（压缩版，适合生产环境）。  **使用方法**  <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  引入jQuery后，使用`$`或`jQuery`来调用核心函数，如`$(selector)`来选取元素，`$(html)`来创建元素，`$(document).ready(function(){...})`来绑定文档加载完成事件。  **2. 核心函数与对象**  **jQuery核心函数**  `$(selector)`：通过选择器获取DOM元素，并返回一个jQuery对象。  `$(function(){...})`：文档加载完成后执行的函数，类似于原生JS中的`window.onload`，但执行时间更早。  **jQuery对象**  jQuery对象是一个类数组对象，可以调用jQuery的方法对批量元素进行操作。如`$('div').css('color', 'red')`可以一次性修改多个`div`元素的样式。  jQuery通过封装DOM操作，使得常见任务如选择元素、绑定事件更加简洁，同时jQuery对象提供了丰富的链式操作。  **3. jQuery与原生JavaScript**  DOM操作对比  原生JavaScript使用`document.getElementById`、`document.getElementsByClassName`等来选择元素，并通过`style`属性直接操作样式。  jQuery使用更简洁的`$('id')`、`$('.class')`等选择器，并通过`.css()`来操作样式。  // 原生JavaScript  var div1 = document.getElementById('div1');  div1.style.backgroundColor = 'red';  // jQuery  $('div1').css('background-color', 'red');  jQuery极大地简化了常见的DOM操作，让代码更简洁易读。  **4. 事件绑定**  **jQuery事件绑定**  使用`$(element).on('event', handler)`来绑定事件，取代原生的`addEventListener`。  例如：`$('button').on('click', function(){...})`可以为按钮绑定点击事件。  jQuery的事件绑定方法更加直观且支持链式操作，还能方便地绑定多个事件。  **5. jQuery**  选择器种类  基本选择器：如`id`、`.class`、`element`。  层次选择器：如`parent > child`、`ancestor descendant`。  属性选择器：如`[type='text']`。  // 选择所有包含class为box的div元素  $('div.box').css('border', '1px solid black');  jQuery的选择器几乎完全基于CSS选择器，极大增强了元素选取的灵活性。  **6. 事件代理**  **事件代理机制**  使用`$(parent).on('event', selector, handler)`来代理事件。这种方法适用于动态添加的元素，可以避免为每个新元素重复绑定事件。  $('parent').on('click', '.child', function(){  alert('Child element clicked!');  });  事件代理通过委托父元素处理事件，提升了性能并减少了代码复杂度。  **7. 事件类型**  **常见事件类型**  `click`：点击事件  `focus`：聚焦事件  `submit`：表单提交事件  $('form').on('submit', function(){  alert('Form submitted!');  });  通过jQuery简化常见事件的处理过程，代码更直观易于维护。  **总结**  jQuery通过封装复杂的JavaScript操作，提供了简洁、强大的API，使得网页开发更加高效。掌握jQuery的核心函数、事件绑定、选择器等知识，是提高开发效率的重要一步。  日期2024.8.26 |

Day02笔记

|  |
| --- |
| **1. jQuery DOM操作**  $(document).ready(function () {  var $div1 = $('div:first');  var $div2 = $('.box1');  var $div3 = $('box2');  $div1.css({ backgroundColor: 'red' });  $div2.css({ backgroundColor: 'yellow', width: '200px', height: '200px' });  $div3.css({ backgroundColor: 'blue' });  });  `$(document).ready(function () {...})`：确保文档加载完成后执行内部代码。  `$('div:first')`：选择第一个`<div>`元素。  `$('.box1')`：选择所有`class`为`box1`的元素。  `$('box2')`：选择`id`为`box2`的元素。  `css({...})`：通过CSS修改元素的样式。  jQuery选择器简化了对DOM元素的操作，通过类名、ID等选择器，开发者可以迅速定位和操作页面元素。`css()`方法是jQuery中非常常用的一个方法，能够动态修改页面元素的样式。  **2. 事件处理**  $('button').click(function () {  alert($(this).text());  });  `$('button').click(function () {...})`：为所有`button`元素绑定点击事件。  `$(this).text()`：获取当前点击按钮的文本内容。  jQuery为事件处理提供了一个简洁的接口，比如`click()`。在这个例子中，使用`$(this)`获取触发事件的元素，结合其他jQuery方法，可以快速获取并处理元素内容。  **3. DOM 克隆**  var $clone = $('element').clone(true);  $('body').append($clone);  `$('element').clone(true)`：克隆`id`为`element`的元素，`true`表示深度克隆，包括事件处理函数。  `$('body').append($clone)`：将克隆的元素追加到`body`中。  克隆方法`clone()`在复杂的DOM结构复制中非常有用。深度克隆不仅复制元素本身，还复制其事件绑定，这在复制动态内容时尤为重要。  **4. jQuery 静态方法**  $.each([1, 2, 3], function(index, value) {  console.log(index, value);  });  `$.each(array, function(index, value) {...})`：遍历数组`array`中的每一项，并执行回调函数。  `index`：当前项的索引。  `value`：当前项的值。  `$.each()`是一个通用遍历方法，不仅能遍历数组，还可以遍历对象。相比于原生`for`循环，`$.each()`更具可读性，并且支持更多类型的数据结构。  **5. 测试操作**  console.log($.isNumeric("123")); // true  console.log($.isPlainObject({})); // true  `$.isNumeric("123")`：检查参数是否为数字或可转换为数字的字符串。  `$.isPlainObject({})`：判断对象是否通过`{}`或`new Object()`创建。  jQuery提供了一系列静态方法用于测试数据类型和结构，这在需要处理动态数据类型或判断数据结构时非常有帮助，提升了代码的健壮性。  **6. 字符串操作**  var trimmed = $.trim(" hello world ");  console.log(trimmed); // "hello world"  `$.trim(string)`：去除字符串两端的空白字符。  `$.trim()`简化了字符串处理，在处理用户输入或数据清洗时非常实用，避免了手动使用正则表达式或`String`方法来去除空白字符。  **7. AJAX 操作**  $.ajax({  url: 'https://api.example.com/data',  method: 'GET',  success: function(response) {  console.log(response);  },  error: function(error) {  console.log(error);  }  });  `$.ajax({...})`：发送AJAX请求。  `url`：请求的URL地址。  `method`：请求方法，如`GET`或`POST`。  `success`：请求成功时执行的回调函数。  `error`：请求失败时执行的回调函数。  jQuery的`$.ajax()`方法极大简化了发送异步请求的流程，集成了各种回调函数，可以轻松处理不同的请求结果。通过这种方式，可以实现与后台的数据交互而无需刷新页面。  jQuery为开发者提供了丰富的工具来操作DOM、处理事件、管理数据、以及进行异步通信，大大提升了开发效率和代码的可维护性。  日期2024.8.27 |

Day03笔记

|  |
| --- |
| jQuery提供的一些自定义全局函数，能够为各种与Ajax相关的事件注册回调函数。这些都是全局函数，因此无论创建他们的代码位于何处，只要有Ajax请求时，就会触发他们。使用这些方法的每一步是获取一个页面元素的引用。如果想使某个ajax不受全局方法的影响，那么可以在$.ajax(options)方法中，将参数中的global设置为false,如：  3.1 ajaxStart()  AJAX 请求开始时执行函数。Ajax 事件。Jquery中当一个Ajax请求启动时，并且没有其他未完成的Ajax请求时，将调用ajaxStart()方法。  3.2 ajaxStop()  AJAX 请求结束时执行函数。Ajax 事件。ajaxStop()方法则是在所有Ajax请求都完成时调用。  在开发的过程当中，我一般会用它们来做页面等待进度图片的显示，即所有用ajax加载又不能很快加载完成的（有时页面很慢半天没显示），那么将自动调用ajaxStart显示一个等待的图片出来，等页面所有内容ajax加载完成，自动调用ajaxStop再关闭该层。  //设置 AJAX 请求默认地址为 "/xmlhttp/"，禁止触发全局 AJAX 事件，用 POST 代替默认 GET 方法，设置默认的success函数。  //其后的 AJAX 请求不再设置  $.ajaxSetup({  url: "/xmlhttp/",  global: false,  type: "POST",  success:function(result){$("div").html(result)}  });  $.ajax({ data: myData });  //携带的token  $.ajaxSetup({headers:{Authorization:"xx"}})  $.ajax({  url:"test.html",  global:false  });  $("loading").ajaxStart(function(){  $(this).show();  });  $("loading").ajaxStop(function(){  $(this).hide();  });  $(function(){  $(document).ajaxStart(function(){  $("loading").show();  }).ajaxStop(function(){  $("loading").hide();  })  $("ul li").click(function(){  $.ajax({  url:"http://127.0.0.1:7788/baseUser/findAll",  data:{name:$(this).html()},  //global:false,  complete:function(data){  3.3 ajaxSend()  AJAX 请求发送前执行函数。Ajax 事件。  XMLHttpRequest 对象和设置作为参数传递给回调函数。  AJAX 请求发送前显示信息  3.4 ajaxSuccess()  AJAX 请求成功时执行函数。Ajax 事件。  XMLHttpRequest 对象和设置作为参数传递给回调函数。  当 AJAX 请求成功后显示消息  3.5 ajaxError()  AJAX 请求发生错误时执行函数。Ajax 事件。  XMLHttpRequest 对象和设置作为参数传递给回调函数。捕捉到的错误可作为最后一个参数传递。  3.6 ajaxComplete()  AJAX 请求完成时执行函数。Ajax 事件。  XMLHttpRequest 对象和设置作为参数传递给回调函数。  for(var i=0;i<999999999;i++){//模拟超长时间  var sum=Math.pow(i,i)Math.pow(i,i)  }  **速写接口**  速写接口使用起来相对简单一些，可以理解为速写接口是对低级别接口的封装  在jQuery中，$.ajax()方法属于最底层的方法，第2层是load()，$.get()，和$.post()，第3层是$.getJSON()方法。  get()  通过远程 HTTP GET 请求载入信息url,[data],[successCallback],[dataType]  4.2 post()  通过远程 HTTP POST 请求载入信息url,[data],[successCallback],[dataType]  url:发送请求地址。  data:待发送 Key/value 参数。  callback:发送成功时回调函数。  type:返回内容格式，xml, html, script, json, text, \_default。  </script>  </body>  </html>  $.get("test.php");  $.get("test.php", { name: "John", time: "2pm" } );  $.get("test.php", function(data){  alert("Data Loaded: " + data);  });  $.get("test.cgi", { name: "John", time: "2pm" },  function(data){  alert("Data Loaded: " + data);  });  $.post("test.php");  $.post("test.php", { name: "John", time: "2pm" } );  $.post("test.php", { 'choices[]': ["Jon", "Susan"] });  $.post("test.php", $("testform").serialize());  向页面 test.php 发送数据，并输出结果（HTML 或 XML，取决于所返回的内容）：  获得 test.php 页面的内容，并存储为 XMLHttpResponse 对象，并通过 process() 这个 JavaScript 函数进行处理：  获得 test.php 页面返回的 json 格式的内容：  携带token请求后台服务器数据:  4.3 getJson()  通过 HTTP GET 请求载入 JSON 数据  url,[data],[callback]  url:发送请求地址。  data:待发送 Key/value 参数。  callback:载入成功时回调函数。  扩展：  Jquery版本在1.5之前，返回的是XHR对象；当版本高于1.5之后，返回的是deferred对象，可以使用 done 和 fail等链式操作。  $.post("test.php", function(data){  alert("Data Loaded: " + data);  });  $.post("test.php", { name: "John", time: "2pm" },  function(data){  alert("Data Loaded: " + data);  });  $.post("test.php", { name: "John", time: "2pm" },  function(data){  process(data);  }, "xml");  $.post("test.php", { "func": "getNameAndTime" },  function(data){  alert(data.name); // John  console.log(data.time); // 2pm  }, "json");  var baseURL="http://127.0.0.1:7788";  $.ajaxSetup({  headers:  {Authorization:""}  })  $.getJSON(baseURL+"/baseUser/findAll",function(data){  console.log(data)  });  $.post(baseURL+"/role/saveOrUpdate",{name:"haha"},function(data){  console.log(data)  })  // ajax快捷函数  // $.get 第一个参数 url 第二个参数(可选)传递的参数 第三个参数 成功回调  $.get(baseURL+'/category/findAll',{},function (res) {  console.log('get',res);  })  // $.post 第一个参数 url 第二个参数(可选)传递的参数 第三个参数 成功回调  $.post(baseURL+'/user/logout',{},function (res) {  console.log('post',res);  })  <div id="images"></div>  $.getJSON("https://api.flickr.com/services/feeds/photos\_public.gne?  tags=cat&tagmode=any&format=json&jsoncallback=?",function(data){  $.each(data.items, function(i,item){  $("<img/>").attr("src", item.media.m).appendTo("images");  if ( i == 3 ) return false;  });  });  4.4 load()  jQuery load() 方法是简单但强大的 AJAX 方法  load() 方法从服务器加载数据，并把返回的数据放入被选元素中  语法：  必需的URL 参数规定您希望加载的 URL。  可选的data 参数规定与请求一同发送的查询字符串键/值对集合。  可选的callback 参数是 load() 方法完成后所执行的函数名称。  示例：  这个例子将会把文件demo\_test.txt的内容加载到指定的div元素中：  也可以把 jQuery 选择器添加到 URL 参数  示例：  这个例子把 "demo\_test.txt" 文件中 id="p1" 的元素的内容，加载到指定的 <div> 元素中：  可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：  responseTxt - 包含调用成功时的结果内容  statusTXT - 包含调用的状态  xhr - 包含 XMLHttpRequest 对象  // $.getJSON 后台返回的数据类型是json的时候使用该方法  $.getJSON('https://api.apiopen.top/musicRankings', {}, function (res) {  console.log(res);  }).done(function (res) { //表示ajax调用成功  console.log('done',res);  }).fail(function () {// 表示ajax调用错误  console.log('fail');  }).always(function () {// 无论是成功还是失败 都会执行 表示ajax完成  console.log('always');  })  $(selector).load(URL,data,callback);  $("div1").load("demo\_test.txt");  $("div1").load("demo\_test.txt p1");  $("button").click(function(){  $("div1").load("demo\_test.txt",function(responseTxt,statusTxt,xhr){  if(statusTxt=="success")  alert("外部内容加载成功！");  if(statusTxt=="error")  alert("Error: "+xhr.status+": "+xhr.statusText);  });  });  // 处理点击切换页面  $(function () {  // 获取到li 增加点击事件  $('.sc-body li').click(function (event) {  // 通过event.target获取到每个li的文本  var text = $(event.target).text()  // console.log(text,event);  // 通过我们点击的菜单 加载对应的界面  switch (text) {  // load 将对应的内容 加载到content里  // 注意 我们的js文件是在index.html中引入的 所以这里面的路径要根据index.html来获取  case '学生管理':  1. H5新增结构化标签  1.1 块元素  Html5新增了很多语义化标签，这些标签多为块元素，其表现形式与div完全一致，即没有特殊样式。但是div是无意义的块元素，而Html5新  增的这些元素多为有意义元素，例 如header表达了头部的意思，在网页中任意暗含头部的结构都可以使用header标签。  其他还有article、section、aside、header、nav、footer、address、hgroup、figure、figcaption等。  此外，Html5中还新增了一些功能性标签，但是由于其在各个浏览器中表现不一致，我们很少使用这些元素，取而代之的是组件，因此这里  不做过多讨论。  1.2 input属性拓展  input是通过type来区分不同功能的表单元素，h5中也做了拓展。  新增的type类型：  email、url、number、range、date、month、week、time、datetime、datetime-local  与功能性标签一样，这些表单元素在各个浏览器中表现效果不一致，我们一般也很少使用。  日期2024.8.28 |

Day04笔记

|  |
| --- |
| **Canvas**  **1. 什么是 Canvas**  Canvas 是 HTML5 新增的一个标签，中文名为“画布”，允许开发者通过 JavaScript 在该标签上绘制各种图案。  应用场景：动画、游戏、数据可视化、图片编辑器、实时视频处理等。  **2. Canvas 与 SVG 的区别**  Canvas：适合大数据量渲染（如图形引擎 G6），受屏幕分辨率影响，不支持事件，数据变化需要重绘。  SVG：适合高交互性和高清晰度的需求（如图形引擎 X6），不受屏幕分辨率影响，支持事件，数据变化无需重绘。  **3. Canvas 的使用方法**  **3.1 绘制直线**  在 HTML 中创建 canvas 元素。  使用 JavaScript 获取 canvas 标签。  获取绘制工具（上下文对象 getContext('2d')）。  使用绘制工具在 canvas 中绘制图形。  关键：  设置宽高应通过 width 和 height 属性，而非 CSS，否则会拉伸画布。  线条宽度和颜色可以通过 lineWidth 和 strokeStyle 设置。  使用 beginPath() 开辟新路径，避免样式污染。  **3.2 绘制折线**  与绘制直线类似，通过反复使用 moveTo 和 lineTo 方法绘制多条直线。  **3.3 绘制矩形**  strokeRect(x, y, width, height)：描边矩形。  fillRect(x, y, width, height)：填充矩形。  rect(x, y, width, height)：绘制矩形，需要调用 stroke() 或 fill() 渲染。  **3.4 清空画布**  clearRect(x, y, width, height) 清空指定区域或整个画布。  **3.5 绘制虚线**  使用 setLineDash([5, 10]) 设置虚线。  **3.6 绘制多边形**  通过指定三个角的坐标，用 stroke() 或 fill() 绘制。  3.7 绘制圆形  arc(x, y, r, sAngle, eAngle, counterclockwise) 绘制圆弧或圆形。  3.8 文本设置  通过 font 设置字体样式。  描边与填充：  strokeText(text, x, y, maxWidth)：文本描边。  fillText(text, x, y, maxWidth)：文本填充。  3.9 图片处理  渲染方式：使用 drawImage(image, dx, dy) 在画布中渲染图片，可以通过 JavaScript 动态加载图片或从 DOM 中获取图片对象。  3.10 绘制渐变  线性渐变：createLinearGradient(x1, y1, x2, y2) 创建渐变对象，通过 addColorStop() 添加颜色，并通过 fillStyle 渲染。  径向渐变：createRadialGradient(x0, y0, r0, x1, y1, r1) 创建径向渐变对象，类似线性渐变的使用方法。  日期2024.8.29 |

Day05笔记

|  |
| --- |
| 1. Web存储与通信  1.1 Web存储  Web存储是HTML5新增的一项功能，它允许在用户浏览器中存储数据，主要分为两种类型：  localStorage: 用于存储持久化的数据，数据不会过期，除非手动删除。  sessionStorage: 用于存储会话级的数据，当页面会话结束（浏览器关闭）后，数据会被清除。  存储数据:  localStorage.setItem('key', 'value');  sessionStorage.setItem('key', 'value');  读取数据:    let data = localStorage.getItem('key');  let data = sessionStorage.getItem('key');    删除数据:    localStorage.removeItem('key');  sessionStorage.removeItem('key');    1.2 Web通信  Web通信技术允许在客户端与服务器之间建立持续的连接，以便实时发送和接收数据。  1.2.1 Ajax  Ajax是一种在不重新加载整个页面的情况下，通过与服务器进行少量数据交换来更新页面的技术。  用法：  let xhr = new XMLHttpRequest();  xhr.open('GET', 'url', true);  xhr.onreadystatechange = function() {  if (xhr.readyState === 4 && xhr.status === 200) {  console.log(xhr.responseText);  }  };  xhr.send();  1.2.2 Fetch API  Fetch API是现代替代Ajax的标准，它使用Promise处理网络请求。  用法：  fetch('url')  .then(response => response.json())  .then(data => console.log(data))  .catch(error => console.error('Error:', error));  2. WebSocket  WebSocket是一种在单个TCP连接上进行全双工通信的协议，它允许服务端主动向客户端推送数据。适用于聊天应用、实时数据更新等场景。  用法：  let socket = new WebSocket('ws://url');  socket.onopen = function() {  console.log('WebSocket 已经建立连接');  socket.send('Hello Server!');  };  socket.onmessage = function(event) {  console.log('Message from server:', event.data);  };  socket.onclose = function() {  console.log('WebSocket connection closed');  };  socket.onerror = function(error) {  console.error('WebSocket error:', error);  };  3. JS地理位置  JavaScript提供了`Geolocation API`，允许获取用户的地理位置信息。  用法：  if (navigator.geolocation) {  navigator.geolocation.getCurrentPosition(  position => {  console.log('Latitude:', position.coords.latitude);  console.log('Longitude:', position.coords.longitude);  },  error => console.error('Error Code:', error.code, 'Message:', error.message)  );  } else {  console.error('Geolocation is not supported by this browser.');  }  4. Bootstrap及可视化  Bootstrap是一个流行的前端框架，用于快速开发响应式网站和Web应用。  4.1 基本结构  Bootstrap基于12列栅格系统，提供了多种响应式布局和组件。  <div class="container">  <div class="row">  <div class="col-md-6">Column 1</div>  <div class="col-md-6">Column 2</div>  </div>  </div>  4.2 常用组件  按钮:  <button type="button" class="btn btn-primary">Primary Button</button>  导航栏:  <nav class="navbar navbar-expand-lg navbar-light bg-light">  <a class="navbar-brand" href="">Navbar</a>  </nav>  4.3 Bootstrap与可视化  Bootstrap与数据可视化库（如ECharts）可以结合使用，提供美观的UI和交互。  5. ECharts  ECharts是一个开源的、基于JavaScript的数据可视化库，支持丰富的图表类型，如折线图、柱状图、散点图等。  用法：  <div id="main" style="width: 600px;height:400px;"></div>  <script>  var myChart = echarts.init(document.getElementById('main'));  var option = {  title: { text: 'ECharts example' },  tooltip: {},  xAxis: { data: ['A', 'B', 'C', 'D'] },  yAxis: {},  series: [{  type: 'bar',  data: [5, 20, 36, 10]  }]  };  myChart.setOption(option);  </script>  6. Ant Design (Antd)  Ant Design是一个用于开发企业级UI的React组件库，提供了一套丰富的组件，支持快速构建用户界面。  用法：  import { Button, DatePicker } from 'antd';  import 'antd/dist/antd.css';  function App() {  return (  <div>  <Button type="primary">Primary Button</Button>  <DatePicker />  </div>  );  }  export default App;  6.1 常用组件  按钮:  <Button type="primary">Primary Button</Button>  表格:  import { Table } from 'antd';  const columns = [{ title: 'Name', dataIndex: 'name', key: 'name' }];  const data = [{ key: '1', name: 'John Brown' }];  <Table columns={columns} dataSource={data} />  日期2024.8.30 |