Ajax 编程

目录 Contents

模板引擎概述

作用：使用模板引擎提供的模板语法，可以将数据和 HTML 拼接起来。

官方地址： https://aui.github.io/art-template/zh-cn/index.html

模板引擎

使用步骤

1. 下载 art-template 模板引擎库文件并在 HTML 页面中引入库文件

```
<script src="./js/template-web.js"></script>
```

2. 准备 art-template 模板

```
<script id="tpl" type="text/html">
    <div class="box"></div>
</script>
```

模板引擎

使用步骤

3. 告诉模板引擎将哪一个模板和哪个数据进行拼接

```
var html = template('tpl', {username: 'zhangsan', age: '20'});
```

4. 将拼接好的html字符串添加到页面中

```
document.getElementById('container').innerHTML = html;
```

5. 通过模板语法告诉模板引擎，数据和html字符串要如何拼接

```
<script id="tpl" type="text/html">
    <div class="box"> {{ username }} </div>
</script>
```

目录 Contents

案例

验证邮箱地址唯一性

1. 获取文本框并为其添加离开焦点事件
2. 离开焦点时，检测用户输入的邮箱地址是否符合规则
3. 如果不符合规则，阻止程序向下执行并给出提示信息
4. 向服务器端发送请求，检测邮箱地址是否被别人注册
5. 根据服务器端返回值决定客户端显示何种提示信息

```
/^[A-Za-z\d]+([-_.][A-Za-z\d]+)*@([A-Za-z\d]+[-.])+[A-Za-z\d]{2,4}$/
```

案例

搜索框内容自动提示

1. 获取搜索框并为其添加用户输入事件
2. 获取用户输入的关键字
3. 向服务器端发送请求并携带关键字作为请求参数
4. 将响应数据显示在搜索框底部

案例

省市区三级联动

1. 通过接口获取省份信息
2. 使用JavaScript获取到省市区下拉框元素
3. 将服务器端返回的省份信息显示在下拉框中
4. 为下拉框元素添加表单值改变事件（onchange）
5. 当用户选择省份时，根据省份id获取城市信息
6. 当用户选择城市时，根据城市id获取县城信息

目录 Contents

FormData 对象的作用

1. 模拟HTML表单，相当于将HTML表单映射成表单对象，自动将表单对象中的数据拼接成请求参数的格式。
2. 异步上传二进制文件

FormData 对象的使用

1. 准备 HTML 表单

```
 <form id="form">
     <input type="text" name="username" />
     <input type="password" name="password" />
     <input type="button"/>
</form>
```

2. 将 HTML 表单转化为 formData 对象

```
var form = document.getElementById('form');
var formData = new FormData(form);
```

FormData 对象的使用

3. 提交表单对象

```
xhr.send(formData);
```

注意：

1. Formdata 对象不能用于 get 请求，因为对象需要被传递到 send 方法中，而 get 请求方式的请求参数只能放在请求地址的后面。
2. 服务器端 bodyParser 模块不能解析 formData 对象表单数据，我们需要使用 formidable 模块进行解析。

FormData 对象的实例方法

1. 获取表单对象中属性的值

```
formData.get('key');
```

2. 设置表单对象中属性的值

```
formData.set('key', 'value');
```

FormData 对象的实例方法

3. 删除表单对象中属性的值

```
formData.delete('key');
```

4. 向表单对象中追加属性值

```
formData.append('key', 'value');
```

注意：set 方法与 append 方法的区别是，在属性名已存在的情况下，set 会覆盖已有键名的值，append会保留两个值。

FormData 二进制文件上传

```
<input type="file" id="file"/>
```

```
 var file = document.getElementById('file')
// 当用户选择文件的时候
 file.onchange = function () {
     // 创建空表单对象
     var formData = new FormData();
     // 将用户选择的二进制文件追加到表单对象中
     formData.append('attrName', this.files[0]);
     // 配置ajax对象，请求方式必须为post
     xhr.open('post', 'www.example.com');
     xhr.send(formData);
 }
```

FormData 文件上传进度展示

```
// 当用户选择文件的时候
file.onchange = function () {
    // 文件上传过程中持续触发onprogress事件
    xhr.upload.onprogress = function (ev) {
        // 当前上传文件大小/文件总大小 再将结果转换为百分数
        // 将结果赋值给进度条的宽度属性
        bar.style.width = (ev.loaded / ev.total) * 100 + '%';
    }
}
```

FormData 文件上传图片即时预览

在我们将图片上传到服务器端以后，服务器端通常都会将图片地址做为响应数据传递到客户端，客户端可以从响应数据中获取图片地址，然后将图片再显示在页面中。

```
xhr.onload = function () {
    var result = JSON.parse(xhr.responseText);
    var img = document.createElement('img');
    img.src = result.src;
    img.onload = function () {
        document.body.appendChild(this);
    }
}
```

目录 Contents

Ajax请求限制

Ajax 只能向自己的服务器发送请求。比如现在有一个A网站、有一个B网站，A网站中的 HTML 文件只能向A网站服务器中发送 Ajax 请求，B网站中的 HTML 文件只能向 B 网站中发送 Ajax 请求，但是 A 网站是不能向 B 网站发送 Ajax 请求的，同理，B 网站也不能向 A 网站发送 Ajax请求。

同源政策

什么是同源

如果两个页面拥有相同的协议、域名和端口，那么这两个页面就属于同一个源，其中只要有一个不相同，就是不同源。

http://www.example.com/dir/page.html

http://www.example.com/dir2/other.html：同源

http://example.com/dir/other.html：不同源（域名不同）

http://v2.www.example.com/dir/other.html：不同源（域名不同）

http://www.example.com:81/dir/other.html：不同源（端口不同）

https://www.example.com/dir/page.html：不同源（协议不同）

同源政策的目的

同源政策是为了保证用户信息的安全，防止恶意的网站窃取数据。最初的同源政策是指 A 网站在客户端设置的 Cookie，B网站是不能访问的。

随着互联网的发展，同源政策也越来越严格，在不同源的情况下，其中有一项规定就是无法向非同源地址发送 Ajax 请求，如果请求，浏览器就会报错。

使用 JSONP 解决同源限制问题

jsonp 是 json with padding 的缩写，它不属于 Ajax 请求，但它可以模拟 Ajax 请求。

1. 将不同源的服务器端请求地址写在 script 标签的 src 属性中

```
<script src="www.example.com"></script>
```

```
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
```

2. 服务器端响应数据必须是一个函数的调用，真正要发送给客户端的数据需要作为函数调用的参数。

```
const data = 'fn({name: "张三", age: "20"})';
res.send(data);
```

使用 JSONP 解决同源限制问题

3. 在客户端全局作用域下定义函数 fn

```
function fn (data) { }
```

4. 在 fn 函数内部对服务器端返回的数据进行处理

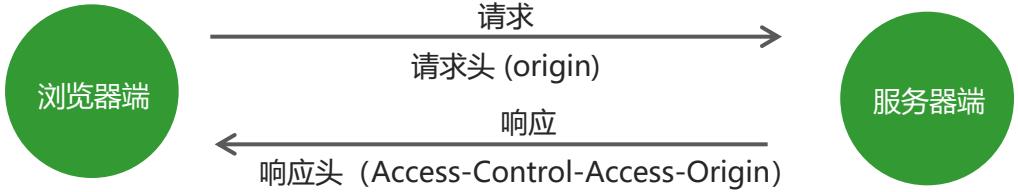
```
function fn (data) { console.log(data); }
```

JSONP 代码优化

1. 客户端需要将函数名称传递到服务器端。
2. 将 script 请求的发送变成动态请求。
3. 封装 jsonp 函数，方便请求发送。
4. 服务器端代码优化之 res.jsonp 方法。

CORS 跨域资源共享

CORS：全称为 Cross-origin resource sharing，即跨域资源共享，它允许浏览器向跨域服务器发送 Ajax 请求，克服了 Ajax 只能同源使用的限制。




```
origin: http://localhost:3000
```

```
Access-Control-Allow-Origin: 'http://localhost:3000'
Access-Control-Allow-Origin: '*'
```

同源政策

CORS 跨域资源共享

Node 服务器端设置响应头示例代码：

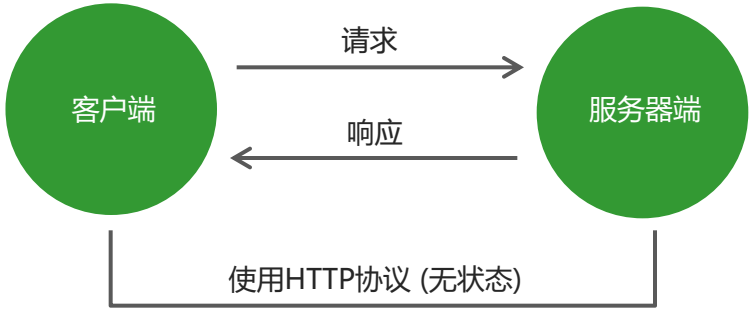
```
app.use((req, res, next) => {
    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Methods', 'GET, POST');
    next();
})
```

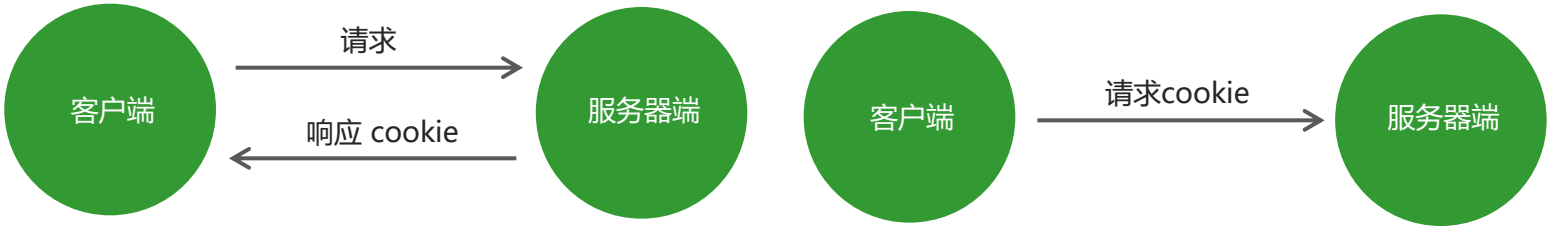
访问非同源数据 服务器端解决方案

同源政策是浏览器给予Ajax技术的限制，服务器端是不存在同源政策限制。

同源政策

cookie复习





withCredentials属性

在使用Ajax技术发送跨域请求时，默认情况下不会在请求中携带cookie信息。

withCredentials：指定在涉及到跨域请求时，是否携带cookie信息，默认值为false

Access-Control-Allow-Credentials：true 允许客户端发送请求时携带cookie