正课:

1. 第三方插件:

Fileupload

Wysiwyg

2. \*\*\*封装自定义插件

3. Ajax

4. \*\*\*\*\*跨域请求

1. 第三方插件:

fileupload:

1. 普通文件上传:

html: form method=“post” enctype=”multipart/form-data” action=”xxx.php”

单击提交按钮时，已经将文件上传到服务器

php: apache隐式自动将上传的文件，保存在tmp目录下

获得文件的基本信息($\_FILES[“name”])，验证或保存

将tmp目录中临时存储的文件，转储到目标目录下

2. ajax上传:

$.ajax({

type:”post”,  
 [url:”xxx.php](url:)”,

data: new FormData($(“#form”)[0]),

contentType:false,  
 processData:false

})

3. 上传插件:

Wysiwyg: 富文本编辑器: rich text editor

扩展: masonry

2. 封装自定义插件:

前提: 必须使用HTML，css和js的方式已经实现了插件的效果

如何: 2种:

1. jQuery侵入方式:

如何定义插件:

1. css文件封装插件样式:

2. 提前约定页面的HTML结构

3. 在jQuery的原型对象中添加插件函数

插件函数中2件事:

1. 侵入class和data-\*扩展属性

2. 为拥有data-\*扩展属性的元素添加事件

使用插件四步:

引入插件的css:

定义页面结构:

引入jquery和插件js

调用方式: $(父元素).插件函数()

2. bootstrap方式:

如何定义插件:

1. css文件封装插件样式:

2. 提前约定页面的HTML结构

3. 手动添加class和data-\*属性

4. 在插件的js文件中，查找data-\*属性的元素，绑定事件

使用插件四步:

引入插件的css:

定义页面结构, 选择要用的css样式类

为元素添加自定义扩展属性，从而添加行为

引入插件的js:

3. jquery中的ajax:

$.ajax({

type:“get/post”,

<url:”xxx.php>”,

data: 查询字符串/对象/new FormData(),

dataType: “json”

}).then(data=>{

… …

})

其中:

data: 即支持查询字符串，又支持对象结构

dataType: 可省略

省略dataType, 则jquery ajax函数可根据服务端header的响应 头设置，自动决定是否调用JSON.parse

更简化:

$.get(“url”[,data]).then()

$.getJSON()

$.getScript()

$.post(“url”[,data]).then()

更简化: 获得一段HTML片段，并填充到指定父元素下

$(父元素).load(“url”)

$.get(“url”).then(html=>{ $(html).appendTo($父元素)})

4. \*\*\*\*\*跨域请求:

跨域:请求的发起网页地址origin和请求的目标地址不一致:

3种: 协议不同, 一级/二级域名不同, 端口号不同

问题: xhr对象禁止发起跨域请求的

解决: 不是所有元素/对象都禁止跨域

link script img iframe … 都支持跨域

方法一: script (可加载并执行js脚本)

请服务端返回一条可执行的js语句，填充要返回的数据

用script去请求服务端，获得可执行的js语句

专业名词: JSONP JSON with padding (填充式json)

实现一: 将处理数据的语句，直接定义到服务器端

缺点: 众口难调

实现二: 2步:

1. 在客户端提前定义处理函数，接收一个参数
2. 将调用函数的语句放在服务器端返回

缺点: 函数名写死！

实现三: 3步:

1. 在客户端提前定义处理函数，接收一个参数
2. 将函数名以查询字符串参数的形式传递给服务器端
3. 服务器端获得函数名，将其和要返回的数据拼接为一条js语句

特殊: 动态发起跨域请求:

1. 动态创建script并追加到body结尾, 并将script元素保存在公共的变量中
2. 在处理函数结尾，自动删除保存的script元素

jQuery中跨域:

$.ajax({type:”get”,url:”xxx.php”,datatype:”jsonp”}).then(处理函数！)

方法二: 服务器端设置允许跨域

其实: 服务器端一句话可解决跨域:

header(“Access-Control-Allow-Origin: \* ”)