正课:

1. 第三方插件:

1. 文件上传

2. wysiwyg

2. \*\*\*\*\*封装自定义插件

3. ajax

4. \*\*\*\*\*跨域请求

1. 第三方插件:

文件上传:

1. 普通表单上传文件:

<form action="xxx.php" method="post" encType="multipart/form-data"

2. ajax文件上传:

$.ajax({

type:"post",

url:"xxx.php",

data:new FormData($("#form1").get(0)),

//强调: FormData中必须用DOM的form元素

processData:false,

contentType:false

})

wysiwyg: richText 富文本编辑器

2. \*\*\*\*\*封装自定义插件:

1. 为jQuery扩展自定义函数:

将要扩展的函数，添加到jQuery的原型对象中

强调: 在引入并使用自定义函数时，必须先引入jQuery类型

2. \*\*\*\*\*封装自定义插件

前提: 已经用HTML，css，js实现了插件想要的效果

为什么: 重用

何时: 只要多个网页间需要相同的功能时

如何封装: 2种:

1. jQuery UI侵入方式

Step1: 提取插件专用的css文件

强调: 尽量仅用class和伪类定义样式。

尽量不要用元素,id选择器

Step2: 在jQuery的原型对象中添加插件函数

//Step2.1: 在函数内, 侵入class

//Step2.2: 为指定元素绑定事件处理函数

如何使用:

1. 按插件的约定，定义HTML结构

2. 引入插件css

3. 引入jquery和插件js

4. 编写自定义脚本: 找到要应用插件的父元素，调用插件函数

2. bootstrap 灵活方式：

vs jQuery UI : jQuery UI具有侵入性

bootstrap没有侵入性

Step1: 提取插件专用的css文件——同jQuery UI的方式

Step2: 在插件js中的页面加载后自动执行的函数内

//查找带data-toggle=XXX的元素，为期绑定事件

如何使用:

1. 按插件的约定，定义HTML结构

2. 引入插件css

3. 用插件css中的class，手动为HTML结构添加样式

因为bootstrap方式，没有侵入性

4. 引入jquery和插件js

插件js: 自动在页面加载时，就查找所有data-toggle=xxx的元素，自动绑定事件

无需再调用插件函数

3. jQuery中的ajax封装:

$.ajax({type: xxx, url:xxx, data:xxx, dataType:xxx,...})

三种简写:

1. 请求一段HTML代码片段，填充到父元素内容中:

$(父元素).load("url"[,data],callback) //不支持then

原理:$(父元素).html(xhr.responseText);

2. 专门简化$.ajax，专门发送get请求

$.get("url","data")

强调: $.get可自动识别php的响应头header

如果header是json，$.get会自动调用JSON.parse

如果header是text，$.get就什么也不做

如何header 是javascript，$get会自动调eval()

强烈建议: 所有php都要定义header中的内容类型

其余: $.getJSON() 总是调用JSON.parse(xhr.responseText)

$.getScript() 总是调用eval(xhr.responseText)

3. 专门简化$.ajax，专门发送post请求

$.post("url","data")

鄙视: 如何让一个函数，无论ajax请求是否成功，都能触发

$.ajax({

type: xxx,

url:xxx,

data:xxx,

dataType:xxx,

success:fn, //请求成功

error:fn, //请求发生错误时

complete:fn, //无论是否成功，都触发

beforeSend:fn //发送请求前触发

})//.then(...)

传统ajax

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

if(xhr.status==200){

success()

}else{

error()

}

complete()

}

}

xhr.open();

beforeSend();

xhr.send();

4. \*\*\*\*\*跨域请求:

什么是: 一个域名下的网页,想访问另一个域名下的资源

比如:



允许跨域请求:

img的src

link的href

script的src

iframe

禁止跨域请求:

xhr

如何: 4种:

1. document.domain

2. window.name

3. HTML5 的 window.postMessage

4. jsonp p:padding 填充

jsonp: 填充式json

什么是: 服务器端，不仅只发送json字符串，而是将json字符串填充在一个js语句中返回

如何:

服务器端: 将原来直接发送的json数据，填充到一个函数调用的js语句中返回

客户端:

Step1: 预先定义一个函数,函数名要和服务器端js语句的函数名一致

Step2: 创建script标签, 请求服务器端php返回的js语句

jQuery中:

2种:

1. $.getJSON("跨域.php?callback=?",函数对象)

2. $.ajax({

type:"get",

url:"跨域的php",//不要带callback=?

success:fn,

dataType:"jsonp"

})