Ajax

希望在页面进行数据提交的时候, 页面不会出现"刷新"(关闭与打开的逻辑).

ajax, 即 异步的 JavaScript 和 XML.

## 同步异步

### 同步：

代码从上往下依次运行，排队执行，上面的不执行完下面的不会执行

### 异步：

上面代码在执行的同时，下面的代码就可以开始执行了

## Ajax的操作模型

发送数据的同时 不去 "等待" 服务器响应返回数据

在 js 中可以创建一个 ajax 的对象, 这个对象提供一个发送请求的方法.

我们调用这个方法可以发送请求 但是不会等待请求的返回 其后的代码继续往下执行.

然后等待若干时间 服务器返回数据后 js会触发一个事件. 在这个事件(事件处理函数)中我们可以获得 服务器返回回来的数据.

## Ajax操作的语法

### Get请求

0）获取事件

1. 创建XMLHttpRequest 对象

Var xhr=new XMLHttpRequest( );

1. 设置请求的地址与方法，以及参数

xhr.open(“ GET ” , ” ./aaa/bbb/ccc/index.php?k=v ”);

1. 注册处理事件

Xhr . onreadystatechange = function( ) { //监听

If ( xhr.readyState == 4 && xhr.status==200) {

Xhr.responseText //拿到的数据，可以用来做各种操作

}

}

1. 发送请求

xhr.send( );

Ajax代码写在事件内

<script>

form.onsubmit = function () {

Ajax代码

return false;

};

</script>

### Post请求

POST 多一个 Content-Type: application/x-www-form-urlencoded

xhr 对象 setRequestHeader 方法来设置请求的请求头

1> open 中使用 POST 代替 GET, 同时将 url 中 的 参数放置 send 方法中( 参数形式不变 )

2> 增加一个请求头

xhr.setRequestHeader( "Content-Type", "application/x-www-form-urlencoded" );

<script>

document.getElementById( "btn" ).onclick = function () {

var xhr = new XMLHttpRequest();

xhr.open( "POST", "./02-index.php" );

xhr.onreadystatechange = function () {

if ( xhr.readyState == 4 && xhr.status == 200 ) {

alert( xhr.responseText );

}

}; xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded" );

xhr.send( "name=jim&age=19&gender=male" );

};

</script>

## 传统请求模型与ajax请求模型的异同

### 传统的网站开发模型：

特点就是所有的数据都在后台执行加载与渲染(集中式渲染)

由于集中式渲染, 一个动态的网站页面会混合 html 代码, css 代码, js 代码以及 后台的脚本代码( php )

在 移动设备 井喷的时代, 一种屏幕尺寸都需要进行适配, 混合代码结构使得开发维护的工作量过大, 效率低.

### Ajax请求模型：

请求结构

1) 有一个页面( 纯 html 的页面, 不包含任何后台代码 )

2) 可以发送 ajax 请求到服务器

3) 服务器可以在你传入的参数中进行验证, 处理, 获取数据. 最后将需要的数据响应给浏览器( 分离 )

4) 浏览器在拿到数据以后, 可以利用 dom 的方式来将数据渲染到页面中( 分布式渲染 )

图示

## json->对象

### eval

将字符串作为代码来执行

ES5+引入了严格模式，eval不允许被使用, 原始方法

例如:

var objStr = '{ name: "jim" }';

var obj = eval( "(" + objStr + ")" );

// var obj = ({ name: "jim" });

### IMG_256 JSON对象 (推荐使用)

提供两个方法：

Parse 将json形式的字符串 转换成对象

Stringify

将一个对象转换成json格式的字符串 与php的json\_encode功能一样

这个方法只能处理严格的json格式

### Function

(兼容早期和最新的浏览器)

var objStr = '{ name: "jim" }';

var fn = new Function( "return " + objStr ); // 一个参数 objStr

生成一个 无参数 的函数,

function xxxx () {

return { name: "jim" };

}

var obj = fn();

例如: 要生成一个函数

function fn ( num1, num2 ) {

return num1 + num2;

}

逻辑上等价

var fn = new Function ( 'num1', 'num2', 'return num1 + num2' );

得到 json 格式字符串对应的 对象

var obj = (new Function( "return " + objStr ))();

## 使用art-template

### 模板的写法

<script type="text/html" id="imageTemplate">

1) 在页面中引入一个 script 标签, 设置其属性 type="text/template".

2) 在 script 标签中 添加 id 属性, 用于后面引用.

3) 在 script 标签中写模板即可. 模板就是 html 标签

### 原生语法:

<%= 名字 %> 逻辑上等价于 php 中的 <?php echo 名字 ?>

简化语法:

{{ 名字 }}

### 使用 art-template

1) 引入 template-web.js

2) 调用函数 template

语法:

template( '模板的id', { 填坑用的对象 } ) 返回 html 格式的字符串(填好坑的字符串)

[重要]:

在模板引擎中采用了分离的思想

我们将模板与背后的这个对象, 在逻辑上视为一个整体

模板上面的每一个名字, 就是背后对象的一个属性.

## art-template( 原生的语法 )

1> 插值语法

<%= 表达式 %> 它与 php 的结构是一一对应的.

<% %> 等价于 <?php ?> 其中 = 等价于 echo

2> 原始输出

如果需要将 html 的标签数据以 html 的形式显示在页面中, 模板可以使用

<%- 表达式 %>

## art-template( 简化语法 )

1> 插值语法

{{ 表达式 }}

2> 原始输出使用

{{@ 表达式 }}

3> 分支

{{ if 表达式 }}

...

{{ else }}

...

{{ /if }}

有时也会使用三目运算符来表示分支

{{ list[ i ].stu\_gender == 'f' ? '女' : '男' }}

4> 循环

默认语法

{{ each 数组 }}

循环体

默认： 使用 $index 来表示索引

使用 $value 来表示索引对应的项

{{ /each }}

自定义语法

{{ each 数组 as 项 索引 }} as 可以省略

{{ /each }}

## jQuery中的ajax

### 快捷语法

$.get(url,参数,callback);

$.post(url,参数,callback);

回调函数想要拿到数据必须使用参数

$.get( "./02-index.php", "k=v&kk=vv&kkk=vvv", function ( data ) {

console.log( data );

});

data为拿到的数据 数组 [] 或 对象 {} （数组还是对象根据[] {} 区分）

### 原始语法

$.ajax({

url: '地址',

type: 'get|post',

data: '数据',

dataType: 'json|xml|html|josnp|text|script',

success: function ( responseData ) {

},

// 了解 一下

error: function ( responseData ) {

},

start: function ( responseData ) {

},

stop: function ( responseData ) {

},

complete: function ( responseData ) {

}

});

例：

$.ajax({

'url' : 'adduser\_deal.php',

'type': 'post', 传送方式 使用FormData时，必须使用post方式

'data': fd, 要传送的数据

'dataType': 'text', 返回数据的类型

//如果使用FormData对象，额外增加下面两个配置项

'contentType': false,

'processData': false,

'success': function(msg){ 得到的返回数据做什么操作

//判断msg

if(msg == 1){

alert('添加新用户成功');

} else {

alert('添加新用户失败');

}

});