# JQuery

## ajax请求

##### 三级标题

$.ajax({

url:"<%=base%>/admin/dict/getByKey.jhtml?key=commonStatus&t=new Date.getTime()"

,dataType:'json'

,async:false

}).responseText

## 事件

$('.layui-btn.layuiadmin-btn-admin').on('click', function(){

var type = $(this).data('type');

active[type] ? active[type].call(this) : '';

});

$().onclick=function(){

}

## Dom操作

##### 添加元素

在标签之前添加元素

$("body").prepend("<div class='cjw'>=================</div>");

##### 添加删除class

e.parents("li").eq(0).find("i").removeClass('active-li');

e.find("i").eq(0).addClass('active-li');

##### 操作样式

$(".addPartWin").eq(0).css("display", "block");

$("p").css({"background-color":"yellow","font-size":"200%"});

## 其他

对象json互相转换

JSON.stringify

对象key值集获取

Object.keys(a)

# Layui

## 基本

##### 遍历数据

each的使用方法

layui.each(data.list, function(index, item){

##### Xxx

##### Xxx

##### Xx

##### X

## 事件

##### 表单事件的监听

语法：form.on('event(过滤器值)', callback);

event: 可选值【select，checkbox，switch(监听checkbox复选框开关事件)，radio，submit】

过滤器值: 如lay-filter="test" 此处过滤器值为test

callback：

中的参数data分析

data.elem //得到select原始DOM对象

data.value //得到被选中的值

data.othis //得到美化后的DOM对象

othis.innerHTML 返回原始html内容

othis.outerHTML 经过layui编译后的代码

othis.textContent 返回所有选项值拼接的字符串

data.elem.checked //是否被选中，true或者false checkbox时调用

以下两个是针对submit事件

data.form //被执行提交的form对象，一般在存在form标签时才会返回

data.field //当前容器的全部表单字段，名值对形式：{name: value}

callback中的返回值分析

只有在submit事件需要返回值

return false; //阻止表单跳转。如果需要表单跳转，去掉这段即可。

**案例：**

form.on("submit("+a+")", function(data){

cajax({

method: 'add'

,data: $.extend(data.field, {token : token})

});

return false;

})

##### 其他事件

layui.$("body").on('click', "\*[lay-href]",

function() {})

}

## 窗口的使用

layer.open({

type: 2

,title: "编辑角色"

,content: "save\_or\_update.chtml"

,area: ['420px', '480px']

,btn: ["确定", "取消"]

,yes: function(index, layero) {

layero.find(i).contents().find("#"+b).click();

}

,success: function(e, index) {

// 对编辑窗口的表单赋值

layui.util.formVal ({

el: e.find(i).contents().find("#"+f)

,list:[

{id: data[0].id}

,{name: data[0].name}

,{username: data[0].username}

]

});

}

})

## 表单

## 关于封装

##### Cutil

###### X

###### randomId

随机id

###### tableReq

layui表格请求， 可以写得更少，主要作用是统一配置token

内部执行的代码，和所需参数都在里面了：

data.table.render({

elem: data.elem,

url: data.url,

limit: data.limit,

page: data.page,

where: data.data,

method: data.method,

height: data.height,

cols: data.cols,

text: data.text

})

###### simpleAsyncPageBtn

action index pageSize 标签属性

method data success其他传入参数

一个简单的分页控件

###### simplePageBtn

###### X

###### redirectByForm

描述：

重定向 -- 重定向的方式有很多种可以用self.location.href="show.chtml"这种方式重定向

此处方法使用的是表单重定向

就是完成跳转，又采用post请求的方式 传递参数

运用的地方暂时没记录

参数介绍：

action

method = data.method || 'post';

enctype = data.enctype || 'application/x-www-form-urlencoded';

query 预设条件

例子：

redirectByForm({

action: 'show.chtml'

,query: queryStr

})

###### formVal

描述：表单赋值

参数介绍：

el

list

type表单类型可选值【input(默认),select,textarea】

另一个属性例如表单项‘ < input name = 'pass'value = '123456' > ’这个属性的写 法为：pass = ‘123456’

注意：1、type相同时可以添加前缀$符号表示如‘ {$type: data[0].type,type: 'select'}’

1. list属性中除了type和一个非固定属性外不能有其他属性否则异常

例子：layui.util.formVal({

el: layui.$.find(i).contents().find("#" + f),

list: [{

id: data[0].id

},

{

desc: data[0].desc,

type: 'textarea'

}]

});

###### copy

浅拷贝

和引用对象脱离引用关联 【对象的拷贝】

###### formAjax

ajax封装--适用于表单提交，提交过程有刷新提示、结果提示

默认 application/x-www-form-urlencoded; charset=UTF-8

描述：

ajax封装--适用于表单提交，提交过程有刷新提示、结果提示

固定规格

方式 :post

返回数据 :json

参数介绍：

url :请求地址

data :数据

loadStyle :加载样式 见layui官网文档

success :function 请求成功，返回状态成功

error :function 请求成功，返回状态失败

serverError :function 请求失败，服务器异常

###### parseImgPath

将不完整的路径替换成有效路径 ，资源位置默认在上传位置

注意，使用时要定义一个全局的变量base

###### parseFloat

转换以‘.00’结尾的字符串数据

###### iframeAjax

默认contentType: 'application/x-www-form-urlencoded'

###### CAjax

表单 底层调用上面的formAjax

默认contentType: 'application/json'

###### cajax

适用于表格 数据的获取

默认表单 application/x-www-form-urlencoded 编码

默认后缀为.do

最后会有一个成功或者失败的提示

###### addSubCtrlbtn

加减控件

###### clickEvent

点击事件

标注属性click-event和值

###### layHref

打开一个标签页

标注属性lay-href和所跳转的地址值

###### getParameter

获取在链接中的参数

###### X

###### X

###### X

###### X

###### x

##### Xxx

##### xxx

# Vue

## Export

区别：

export 可以导出多个命名模块

export default 只能导出一个默认模块，这个模块可以匿名

案例：

export default 只能导出一个默认模块，这个模块可以匿名

//demo1.js

export default {

a: 'hello',

b: 'world'

}

对应的引入方式：

//demo2.js

import obj from 'demo1'

# ES6

## ES6新特性（2015）

<https://www.jianshu.com/p/e2ba90e89754>

### 类

class Animal {

constructor(name, color) {

this.name = name;

this.color = color;

}

toString() {

console.log('name:'+ this.name + ',color:' + this.color);

}

}

var animal = new Animal('dog', 'white');

class Cat extends Animal {

constructor(action) {

// 如果没有置顶consructor,默认带super函数的constructor将会被添加、

super('cat', 'white');

this.action = action;

}

toString() {

console.log(super.toString());

}

}

### 模块化

一个文件为一个模块

模块之间的相互调用关系是通过 export 来规定模块对外暴露的接口

##### 导出(export)

Export可以导出变量

export var name = 'Rainbow'

Export可以导出多个变量

var name = 'Rainbow';

var age = '24';

export {

name,

age

};

Export可以导出常量

导出函数

export function myModule (someArg) {

return someArg;

}

##### 导入(import)

import myModule from 'myModule';

import { name ,age } from 'test';

### 箭头函数

箭头函数与包围它的代码共享同一个 this,能帮你很好的解决this的指向问题。

和之前是不同的

箭头函数的箭头=>之前是一个空括号、单个的参数名、或用括号括起的多个参数名，而箭头之后可以是一个表达式（作为函数的返回值），或者是用花括号括起的函数体（需要自行通过return来返回值，否则返回的是undefined）。

() => 1;

(a, b) => a + b;

() => ([1, 2]);

() => ({

a: 1,

b: 2

});

() => {

alert()

}

setTimeout(() => {

// to do

}, 500)

bind还是箭头函数，每次被执行都返回的是一个新的函数引用,必须自己保存这个引用。

### 函数参数默认值

es6

function foo(width= 20, height = 50) {

console.log(width, height)

}

以往

function foo(width, height) {

var w = width || 20;

var h = height || 50;

console.log(w, h)

}

解决了以往布尔值默认参数设置的问题

### 模板字符串

更加的简洁、直观

let str = `hello, ${name}, my name is ${myName}`;

### 解构赋值

从数组中获取值并赋值到变量中，变量的顺序与数组中对象顺序对应。

var foo = ["one", "two", "three", "four"];

var [one, two, three] = foo;

获取首尾

var [first, , , last] = foo;

var a, b; [a, b] = [1, 2];

设置初始值，数组没有值则使用初始值

var a, b; [a = 5, b = 7] = [1];

两个变量进行值交换

var a = 1;

var b = 3; [a, b] = [b, a];

很方便的获取student对应的值 对象属性

const student = {

name: 'Ming',

age: '18',

city: 'Shanghai'

};

const {

name,

age,

city

} = student;

### 延展操作符

var arr1 = [0, 1, 2];

var arr2 = [3, 4, 5];

var arr3 = [...arr1, ...arr2];

var arr4 = arr1.concat(arr2);

var obj1 = {

foo: 'bar',

x: 42

};

var obj2 = {

foo: 'baz',

y: 13

};

var clonedObj = { ...obj1 };

var mergedObj = { ...obj1, ...obj2 };

var params = {

name: '123',

title: '456',

type: 'aaa'

}

var {

type, ...other// 排除type元素，自成数组

} = params;

<CustomComponent type = 'normal'number = {2} {...other} />

<CustomComponent type='normal'number={2}name='123'title='456' />

### 对象属性简写

const name = 'Ming',

age = '18',

city = 'Shanghai';

const student = {

name,

age,

city

};

console.log(student);

### Promise异步编程的一种解决方案，比传统的解决方案callback更加的优雅。

两个setTimeout 可以这么写

var waitSecond = new Promise(function(resolve, reject) {

setTimeout(resolve, 1000)

});

waitSecond.then(function() {

console.log("Hello");

return waitSecond

}).then(function() {

console.log("Hi")

});

### Let与Const

let a = 10

# Cutil脚本解说

# Css

对滚动条的设置

.code::-webkit-scrollbar {width: 4px;border-radius:2px;}

.code\_box .code::-webkit-scrollbar-button {background-color: #fff;}

.code\_box .code::-webkit-scrollbar-track {height:100px;background: #fff;}

.code\_box .code::-webkit-scrollbar-thumb {background: #ccc;border-radius: 5px;}

### 伪类：

:link

:visited

:hover

:active

:focus

:lang(fr)

:not(s)

:root

:first-child

:last-child

:only-child

:nth-child(n)

:nth-last-child(n)

:first-of-type

:last-of-type

:only-of-type

:nth-of-type(n)

:nth-last-of-type(n)

:empty

:checked

:enabled

:disabled

:target

# JS

谈nodeList

其中包含 text、一般元素等

<https://blog.csdn.net/qq_42127861/article/details/82145842?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.channel_param>

# xxx

## 二级标题

##### 三级标题

###### 四级标题

###### xxx