Script标签

1、嵌入式

<script type="text/javascript">

alert('ok！');

</script>

2、外链式

<script type="text/javascript" src="js/index.js"></script>

3、行内式

<input type="button" name="" onclick="alert('ok！');">

弹窗

alert("hello world")

控制html

<script>

window.onload = function() --js入口函数

{

document.getElementById('box').innerHTML ="hehe"

}

</script>

<div id="box">haha</div>

控制css

css中带”-”的改为小驼峰形式

document.getElementById('box').style.color = ‘red’

document.getElementById('box'). style.background = ‘#CCC’

document.getElementById('box'). style.fontSize=’24px’

var + 名字 = 值

var oBox = document.getElementById('box')

oBox.style.color = ‘red’

oBox.innerHTML = ‘xxx’

数据类型

查看typeof()

1、number 数字类型

2、string 字符串类型

3、boolean 布尔类型

4、undefined类型，变量声明未初始化，它的值就是undefined

5、null类型，表示空对象，如果定义的变量将来准备保存对象，可以将变量初始化为null,在页面上获取不到对象，返回的值就是null

1种复合类型：object

函数

预解析函数和变量的声明(变量未初始化)

function fnAlert(){

…

}

fnAlert()

事件的匿名函数

事件 = function () { 执行内容 }

条件判断

if(){}

else if(){}

else{}

与或非

&& || !

==判断值

===判断值+数据类型

简单事件

<button id="btn1">单击</button>

<button id="btn2">滑过</button>

<button id="btn3">离开</button>

window.onload = function () {

var oBtn1 = document.getElementById("btn1");

var oBtn2 = document.getElementById("btn2");

var oBtn3 = document.getElementById("btn3");

oBtn1.onclick = function () { -->匿名函数

alert("单击成功")

}

oBtn2.onmouseover = function () {

alert("鼠标滑过")

}

oBtn3.onmouseout = function () {

alert("鼠标离开")

}

}

js修改标签中的属性(标签对象.属性),除了class=’’改为className=’’

<link rel="stylesheet" href="h6\_skin1.css" id="link1">

var oLink1 = document.getElementById("link1")

oBtn1.onclick = function () {

oLink1.href="h6\_skin1.css"

}

循环

for(var i=0;i<len;i++)

{

......

}

while(条件)

{

……

增量

}

数组

定义

var aList = new Array(1,2,3);

var aList2 = [1,2,3,'asd'];

获取数组的长度：aList.length;

用下标操作数组的某个数据：aList[0];

join() 将数组成员通过一个分隔符合并成字符串

str = aList.join(‘-’)

push(数据) 和 pop() 从数组最后增加成员或删除成员

arr.unshift(数据)开头添加

arr.shift()删除开头

reverse() 将数组反转

indexOf() 返回数组中元素第一次出现的索引值

aList.indexOf(数据)

splice(下标,删除个数,替换内容) 在数组中增加或删除成员

aList.splice(2,1,1,2,3) 从第2个元素开始，删除1个元素，然后在此位置增加1,2,3三个元素

数组去重

var aList = [10,20,10,30,50,40,20,40,10,60]

var newlist= []

for(var i=0 ; i<aList.length ; i++)

{

if( aList.indexOf ( aList[i] ) == i ) //判断每个元素是否是第一次出现

{

newlist.push( aList[i] )

}

}

字符串相关

parseInt() 将数字字符串转化为整数

parseFloat() 将数字字符串转化为小数

substring(start,end) 截取字符串,类似切片,不包含结束位

indexOf() 查找字符串是否含有某字符

split(‘分隔符’) 把一个字符串分隔成字符串组成的数组

str.split('').reverse().join('')字符串反转

定时器

setTimeout 只执行一次的定时器

clearTimeout 关闭只执行一次的定时器

setInterval 反复执行的定时器

clearInterval 关闭反复执行的定时器

setTimeout/ Interval(函数名,延时)

var time1= null

time1 = setTimeout(myalert,2000);

var time2 = null

time2 = setInterval(myalert,2000);

clearTimeout(time1);

clearInterval(time2);

time1 = null

time2 = null

function myalert(){

alert('ok!');

}

作用域

1. 函数内部可以直接调用修改全局变量
2. 声明变量不加var默认是全局变量,局部变量要加var
3. 变量查找规则(就近查找)

封闭函数

防止函数重名

1. ;(function(){

function fn1(){

alert(1)

}

fn1()

}) ()

2.!/~function(){

alert(2)

}()

JQury

入口函数

$(document).ready(function(){

......

});

简写

$(function(){

……

});

操作内容

$(‘div’).html() --取值

$(‘div’).html(‘123’) –修改

$(‘div’).css(‘color) –取值

$(‘div’).css(‘color’,’red’) –修改

$(‘div’).css({‘color’:’red’,’font-size/fontSize’:’24px’}) –多值修改

选择器

$(‘div’).css(‘color)

$(‘.box).css(‘color)

$(‘#box’).css(‘color)

$(‘li[id=box]’).css(‘color) 选择li中id=’ box’

~~$(‘li’).not(‘.box’).css(‘color) 选择li中除class=’box’的所有~~

$(‘li:first’).css(‘color) 选择第一行

$(‘li:last’).css(‘color) 选择最后一行

$(‘li:eq(-1)’).css(‘color) 选择li中第n行, -1为最后一行

$(‘li’).eq(0).css(‘color)

选择器转移

$(‘.box).prev().css(‘color) 选择前一行

$(‘.box).prevAll().css(‘color) 选择前所有

$(‘.box).next().css(‘color) 选择后一行

$(‘.box).nextAll().css(‘color) 选择后所有

$(‘.box’).has(‘p’).css(‘background’,’green’) 找p标签,选中父级

$(‘.box’).find(‘p’).css(‘background’,’green’) 选中子级

兄弟选择器(排他) siblings

$(this).css(‘background’,’green’).siblings().css(‘background’,’ ’) 选中的变绿(链式调用)

父级选择器parent

$(‘span’).click(function(){

$(this).parent().hide(500)

})

子集选择器children

$(this).children(‘ul’).stop().show(500)

stop()清除动画的排队机制

获取元素的下标

index()

操作样式类名

$("#div1").addClass("divClass2") //为id为div1的对象追加样式divClass2

$("#div1").removeClass("divClass") //移除id为div1的对象的class名为divClass的样式

$("#div1").removeClass("")//全部删除

$("#div1").removeClass("divClass divClass2") //移除多个样式

$("#div1").toggleClass("anotherClass") //重复切换anotherClass样式

显示隐藏

$("div1").show()

$("div1").hide()

自定义动画函数

$(‘div’).animate(字典的形式写css键值对,时间,运动曲线,回调函数)

--时间默认600ms,运动曲线swing加速 linnear匀速

$(‘div’).animate({‘width’:’600px’,’height’:’600px’},200ms ,’ swing’,function(){

alert(‘OK’)

})

$(‘div’).slideUp() 向上卷起,隐藏

$(‘div’).slideDown() 向下展开,显示

$(‘div’).slideToggle()

$(‘div’).fadeIn() 透明显示,淡入

$(‘div’).fadeOut() 透明隐藏,淡出

$(‘div’).fadeToggle()

$(‘div’).fadeTo(时间,透明度小数) 设置透明度

JQ属性操作

prop() 取出或设置某个属性的值

var $src = $('#img1').prop('src');

$('#img1').prop({src: "test.jpg", alt: "Test Image" });

val() 类似html() 直接取值或修改

JQ的循环

$(' li').each(function(i,j){ //自带形参, i下标, j循环的每一项

j.innerHtml = j.innerHtml + ‘abc’

})

JQ的事件

blur() 元素失去焦点

focus() 元素获得焦点

$(‘input’).focus/ blur (function(){ … })

click() 鼠标单击

mouseover() 鼠标进入（进入子元素也触发）

mouseout() 鼠标离开（离开子元素也触发）

mouseenter() 鼠标进入（进入子元素不触发）

mouseleave() 鼠标离开（离开子元素不触发）

hover() 同时为mouseenter和mouseleave事件指定处理函数

$(‘div’).hover(function(){

鼠标移入

},function(){

鼠标移出

})

ready() DOM加载完成

submit() 用户递交表单

表单form中使用

$(‘#myform’).submit(function(){

//不直接提交表单数据,先判断是否合法

if(){

//提交

}

else{

//报错

return false

}

})

js正则

//不常用 var regUser = new RegExp(规则,正则参数)

var re01 = /规则/参数i(忽略大小写)

var re02 = /a/i

var str1=’abc’

//使用方法:正则变量.test(数据)

re02.test(str1)

[表单验证的正则](正则表达式/表单验证正则.doc)

表单验证

失去焦点是验证

$(function(){

//失去焦点时验证

var $user\_name = $(‘#user\_name’)

$user\_name.blur(function(){ 🡪$user\_name.blur(fnCheckUser)

//封装函数

fnCheckUser()

})

function fnCheckUser(){

var vals = $user\_name.val()

var reUser = /正则表达式/

if(vals == ’’){

$user\_name.next().show().html(‘用户名为空’)

return

}

if(reUser.test(vals))

{

//隐藏提示

$user\_name.next().hide()

}

else

{

//显示提示

$user\_name.next().show().html(‘用户名是大于……’)

return

}

}

})

事件冒泡

相同事件,由内向外,逐层触发

阻止事件冒泡

return false

事件捕获

由外向内

事件委托

委托给父标签

父标签.delegate(事件发生的目标子标签,事件属性,执行命令)

可以给未来元素绑定命令,提高效率

$(‘ul’).delegate(‘li’,’click’,function(){

alert(‘ok’)

})

节点操作

追加节点

1.append()和appendTo():在现存元素的内部，从后面放入元素(父子级)

var $li = $(‘<li>123</li>’)

$(‘ul’).append($li) 🡨🡪 $li.appendTo($(‘ul’))

2.prepend()和prependTo():在现存元素的内部，从前面放入元素

3.after()和insertAfter()：在现存元素的外部，从后面放入元素(兄弟级)

var $div =$(‘<div>123</div>’)

$(‘ul’). after ($div) 🡨🡪 $div. insertAfter($(‘ul’))

4.before()和insertBefore()：在现存元素的外部，从前面放入元素

删除节点

$(‘ul’).remove()

清空节点

$(‘ul’).empty()

js对象

定义一

var person = new Object()

person.name = ‘Jack’

person.age = 20

person.getAge = function(){

alert(this.age)

}

调用

alert(person.age)

person. getAge ()

定义二

var person2 = {

name:'Rose',

age: 18,

sayName:function(){

alert('My name is' + this.name);

}

}

调用

alert(person2.age);

person2.sayName();

Json

字典{} 双引号

var json = {

"name":"jack",

"age":29,

"hobby":["reading","travel","photography"]

"school":{

"name":"Merrimack College",

"location":'North Andover, MA'

}

}

json.hobby[0] 🡪 reading

ajax

默认异步 局部刷新

不操作本地文件和数据库

常用参数：

1、url 请求地址

2、type 请求方式，默认是'GET'，常用的还有'POST'

3、dataType 设置返回的数据格式，常用的是'json'格式，也可以设置为'html'

4、data 设置发送给服务器的数据

5、success 设置请求成功后的回调函数

6、error 设置请求失败后的回调函数

7、async 设置是否异步，默认值是'true'，表示异步

$.ajax({

url: '/change\_data',

type: 'GET',

dataType: 'json',

data:{'code':300268}

success:function(dat){

alert(dat.name);

},

error:function(){

alert('服务器超时，请重试！');

}

});

写法二:

$.ajax({

url: '/change\_data',

type: 'GET',

dataType: 'json',

data:{'code':300268}

})

.done(function(dat) {

alert(dat.name);

})

.fail(function() {

alert('服务器超时，请重试！');

});

简写

$.get("/change\_data", {'code':300268},

function(dat){

alert(dat.name);

});

$.post("/change\_data", {'code':300268},

function(dat){

alert(dat.name);

});

html自定义属性

自定义属性名=”自定义属性”

prop() 访问已有的html属性

attr() 访问自定义的html属性

Jsonp

$(function () {

$('input').keyup(function () { //键盘抬起事件

var vals = $(this).val();

$.ajax({

url:'https://sug.so.360.cn/suggest',

type:'get',

dataType:'jsonp',

data:{'word':vals,'encodeout':'utf-8'}

})

.done(function (data) {

var str =''

for (var i =0;i<data.s.length;i++)

{

str+='<li>'+data.s[i]+'</li>'

}

$('ul').html(str)

})

.fail(function (err) {

console.log(err);

})

})

})

Vue

window.onload = function(){

//vue从一个对象开始

var vm = new Vue({

el : '#app' , //控制标签

data:{ //存储的数据

str:'hello world!',

num:0

},

methods:{

fnAdd:function () {

this.num+=1

}

}

})

}

工作中写下面

vue控制html

{{ num + 1 }}

{{ bool ? 'YES' : 'NO' }} 三元表达式: 条件?成立:不成立

{{ str.split('').reverse().join('') }}

<a v-bind:href="url">链接文字</a> 简写🡪 <a :href="url">链接文字</a>

vue绑定事件

<button v-on:click ='num+=1'>按</button> 简写🡪<button @click='num+=1'>按</button>

<button @click ='fnAdd()'>按</button>

vue控制css

class属性:字典,对象,列表

字典{‘类名’:布尔数据} true添加对象 false不加

<div class="static" v-bind:class="{active:isActive,'text-danger':hasError }"></div>

data: {

isActive: true,

hasError: false

}

对象:true添加对象 false不加

<div v-bind:class="classObject"></div>

data: {

classObject: {

active: true,

'text-danger': false

}

}

列表

<div v-bind:class="[activeClass, errorClass]"></div>

data: {

activeClass: 'active',

errorClass: 'text-danger'

}

<div v-bind:class="[isActive ? activeClass : '', errorClass]"></div>\

条件渲染

v-if,v-else-if,v-else指令

<div v-if="type === 'A'">

A

</div>

<div v-else-if="type === 'B'">

B

</div>

<div v-else-if="type === 'C'">

C

</div>

<div v-else>

Not A/B/C

</div>

v-show指令

<h1 v-show="ok">Hello!</h1>

列表渲染

v-for 指令

<li v-for=”(i,j) in list1” >{{ j+1 }}、{{ i }}</li> // i 数据, j 下标,app内全局使用

…data:{ list1:[‘1’,’2’,’3’] }

<li v-for=”(i,j) in dict1” >{{ j }}、{{ i }}</li> // i value, j key值,局部使用

…data:{ dict1:{‘name’:’小明’,’age’:20} }

表单数据绑定

v-model命令,修改默认值

文本框

<input type=”text” v-model=”txt1”>

<div>{{ txt1 }}</div>

…data:{ txt1:’ ’ }

下拉框

<select v-model=”sel1”>

<option value=”0”>上海</option>

<option value=”1”>北京</option>

</select>

…data:{ sel1:’0’ } //默认

单选框

<div id=”app”>

<input type=”radio” v-model=”myradio” value=”0”>男

<input type=”radio” v-model=”myradio” value=”1”>女

</div>

…data:{ myradio:’0’ } //默认

复选框

<div id=”app”>

<input type=”checkbox” value=” read” v-model=”list1”>read

<input type=”checkbox” value=” run” v-model=”list1”>run

<input type=”checkbox” value=” lol” v-model=”list1”>lol

</div>

…data:{ list1:[“run”,”lol”] } //默认

axios

axios({

method: 'post',

url: '/user/12345',

data: {

firstName: 'Fred',

lastName: 'Flintstone'

}

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

});

事件冒泡

阻止冒泡 @click.stop=’…’

阻止表单提交 @click.stop.prevent=’…’

计算属性

vue中对象,类似methods,必须是返回值

{{ fnRs }}

computed:{

fnRs:function(){

return this.str+1

}

}

侦听属性

watch:{

//侦听属性名等于侦听的属性名

num:function(newVal,oldVal){ //变化后的值,变化前的值

console.log(oldVal + ‘|’ + newVal)

}

}

过滤器

{{price | yuan}}

filters:{

//局部过滤器

yuan:function(temp){

if(temp==0){return temp}

return temp + ’元’

}

}

//全局过滤器 在所有对象定义之前

Vue.filter(dollor,function(value){

if(value==0'')

{

return value;

}

return '$'+ value;

});

自定义全局指令

<input … v-focus>

Vue.directive('focus',{

inserted:function(el,binding){ // inserted固定的,当指令生效后执行匿名函数

// binding 指令的用法

el.focus(); //el使用该指令的标签

el.style.background = 'gold';

}

})

ES6

定义变量关键字

var 可以预解析

let const不能修改

箭头函数

var fn = (a,b) =>{ //一个参数省略’=’

return a+b

}

fn(1,2)

var fn1 = () =>{ //无参数加’()’

…

}

模块导出与导入

export import

组件

<aa></aa>

< mycom >< /mycom >

…

Vue.component(‘aa’,{

template:’<h2>mycomponent</h2>’

})

Vue.component(‘mycom’,{

template:’<div><input type=”text” v-model=”num”><button @click= ”num+=1”> sum </button> </div>’,

data:function(){ //注意写法,要有返回值

return {

num:1,

str:’abc’

}

}

})

单文件组件

//html部分

<template>

<div class="breadcrumb" @click="fnLight">

当前位置：<span :class="{hot:isHot}">{{pos}}</span>

</div>

</template>

// js要写成模块导出的形式：

<script>

export default{

props:['pos'],

name:'breadcrumb',

data:function(){

return {

isHot:false

}

},

methods:{

fnLight:function(){

this.isHot = !this.isHot;

}

}

}

</script>

// css中如果有scope关键字，表示这些样式是组件局部的，不会影响其他元素

<style scoped>

.breadcrumb{

width:90%;

line-height:50px;

border-bottom:1px solid #ddd;

margin:0px auto;

}

.breadcrumb .hot{

font-weight:bold;

color:red;

letter-spacing:2px;

}

</style>

单页面vue开发

一、单页面网站开发需要使用webpack -- webpack基于node服务器环境开发 -- 安装node -- 安装vue自动化工具 -- 命令提示行下载webpack目录 -- 切到目录里面运行cmd -- 启动服务npm run dev -- localhost：8080 -- index.html首页

二、认识webpack目录的各个文件的作用

build -- 打包要用的配置文件

config -- webpack服务环境的配置文件

dist -- 将来打包生成的文件（上传的文件）

node\_modules -- 模块

src -- 前端开发的目录

components -- 组件 \*\*\*\* 找对应组件放对应的数据

route -- 路由跳转页面

App.vue -- webpack最大的组件

static -- 静态文件：css js

index.html -- 首页

三、跨域请求数据

1、src文件夹main.js粘贴：

import axios from 'axios'

Vue.prototype.axios = axios

2、config/index.js 的proxyTable粘贴

// 配置跨域代理

// apis 自定义的代理名

'/apis': {

// 代理的服务器

target: 'http://localhost:7890',

// 允许跨域

changeOrigin: true,

pathRewrite: {

'^/apis': ''

}

3、重启服务

4、请求数据

this.axios({

url:'/apis/index\_data',

})

四、打包上线

1、去掉接口的代理名/apis

2、停止服务

3、num run build

4、复制dist文件夹的内容到自己服务器