解析数据时候使用:

JSON.parse(): 将字符串 --> json对象

JSON.stringify(): 将对象 --> 字符串

面向对象:

原型链:

先从自己的原型找-->上一级找-->上一级-->Object-->null

obj.\_\_proto\_\_ --> obj.\_\_proto\_\_.\_\_proto\_\_-->obj.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_

设计模式: 二十三种设计模式 (单例,工厂模式,委托)

创建:

工厂模式创建对象:

function creatObj (name,age) {

var obj = new Object(); // 原料

// 加工

obj.name = name;

obj.age = age;

obj.eat = function () {}

// 出厂

return obj;

}

构造函数的方式创建对象: 方法始终重复创建

new Person()

1.自动帮你创建了一个obj

2.将obj的值赋给了this

3.return this

function Person (name,age) {

this.name = name;

this.age = age;

this.eat = function () {}

}

构造函数+原型的方式创建:

原型: 构造函数名.prototype

对象获取原型: obj.\_\_proto\_\_

function Person (name,age) {

this.name = name;

this.age = age;

}

Person.prototype.eat = function () {}

原型对象可以扩展系统的方法:

String.prototype.trim = String.prototype.trim || function () {

// this

return this.replace(/^\s+|\s+$/g,'');

}

继承:

改变this指向

call():会自动调用函数

apply():会自动调用函数

bind(): 不会调用,方法的返回值就是一个函数

function Person (name,age) {

this.name = name;

this.age = age;

}

Person.prototype.eat = function () {

}

function Worker (name,age) {

//Person.call(this,name,age);

Person.apply(this,arguments);// 继承属性

}

// 继承方法

Worker.prototype = new Person();

Worker.prototype.constructor = Worker;

cookie session:

cookie存储在客户端,session存储在服务器端;

cookie大小只有4kb,最多存储20个

cookie存在过期时间

不安全

必须在服务器环境下执行

localStorage sessionStorage

都是存储在客户端

localStorage5M,sessionStorage5M

localStorage永久存储,sessionStorage,会话关闭,就消失

不安全

不用在服务器环境下

ajax: 原生ajax

get和post:

1.get只能发送4kb的数据,post1个G

2.get参数是拼接url后面,post放到请求头里面

3.get的send方法不需要参数,post需要将参数放到send()里面

4.get不需要设置请求头,post需要设置请求头

ajax兼容?

//1.创建Ajax对象

if(window.XMLHttpRequest)

{

// 兼容所有浏览器

var oAjax=new XMLHttpRequest();

}

else

{

var oAjax=new ActiveXObject("Microsoft.XMLHTTP");// IE6

}

// 2.打开连接 是否异步

oAjax.open('GET',url,true)

//post

oAjax.open('POST',url,true);

oAjax.setRequestHeader("CONTENT-TYPE","application/x-www-form-urlencoded");

// 3.发送

oAjax.send();

// post

oAjax.send('a=10&b=20')

// 4.接受

oAjax.onreadystatechange = function () {

if (oAjax.readyState == 4) {

if (oAjax.status >= 200 && oAjax <= 299 || oAjax.status == 304) {

// 执行成功的回调函数

} else {

// 执行失败的回调函数

}

}

}

如何用ajax实现跨域请求数据?

不能实现跨域,使用jsonp实现跨域

在script中,写src属性,将接口放入src中,得自己手动设置回调函数

1.先拿接口往浏览器里面测试

2.手动创建script,赋值src

3.设置全局的函数

4.用完即扔(从head里面移除)

$.ajax({

type: 'get post',

url: 接口,

data: 参数,

success: fn,

error: fn,

cache: false,缓存

async: true 异步,

timeout: 超时

complete: fn

})

jsonp:

$.ajax({

dataType: 'jsonp',

url:,

data:,

success: function(str) {

var data = JSON.parse(str);

// data = {list: [{},{},{},{}]};

// 渲染页面

}

});

angular的四大特性:

1.双向数据绑定

2.依赖注入: 形参位置不固定,但是形参的名字是固定的 (java)

3.指令机制

4.MVC 或者是 MVVM

angular使用了哪些机制?

1.脏检查

2.指令机制

3.路由机制

4.消息机制 (父子控制器的通信) $emit $on

MVC和MVVM的区别就是：

1.MVC是单向的数据绑定，MVVM是双向数据绑定。

2.MVVM是MVC衍生而来，而MVC是从后台而来的。

3.MVVM的js文件是外部引入而来的，修改时也比较方便。提高了用户体验，增快网页加载速度。

## React 的优点:

- 丰富的第三方资源,支持各种中大型项目的开发.

- 声明式渲染: {{msg}} 非声明式: document.getElementById.innerHTML = msg;

- 组件系统非常强大

- 平台化好,可以运行在各种终端,例如,PC, 手机, 原生APP, 大型终端等等

- 内部运行机制: 效率非常高.

- 虚拟DOM树, 当数据更新时, 触发差异化算法, 以最低的成本局部刷新页面.

- JSX --- 长得像HTML的JS

- 单向数据流: 数据是从顶层组件传递到子组件中

请写出几个常见的排序算法?

冒泡: 从第一个数开始,比较后面的每一个数据,如果第一个数大于后面的数,则调换位置

选择排序: 从当前下标开始,找从该下标+1以后的数的最小值,然后调换位置

归并排序(正宗二分法): 采用二分法,左边一个排序好的数组,右边一个排序好的数组,每次比较左右第一个数,小的,放到一个新的数组

快速排序(二分法): 采用二分法,取出中间数数组,也有左右两边的数组,和中间数组值比较,小的放在左边,大的放到右边,最后连接左中右三个数组