# JavaScript 面试知识点合集

### Why put the script tag in the last of HTML file?

Putting the script tag in the last can reduce the delay time, users won't wait a long time to open the webpage. Because the render direction is from top to bottom.

### How many data types does JavaScript have?

Js have 7 data type in total, they are 6 primitives types and Object:

* 6 primitives are **Number, String, Boolean, Null, Undefined, Symbol**
* Object

### typeof运算符能返回几种结果？

能返回6种，typeof只能区分primitives types和function

undefined typeof undefined

string typeof 'abc'

number typeof 123

boolean typeof true

object typeof { } 或 typeof[ ] 或 typeof null

function typeof console.log

### Js的value type和reference type分别有哪些？

value type: Boolean, String, Number, null, undefined.

var a=10;

var b=a;

a=20;

console.log(b) //10复制代码

reference type: Array, Function. Object.

var a={age:20}

var b=a;

b.age=18;

console.log(a.age)//18;复制代码

### value type & reference type 分别做函数参数的情况

当value type做函数的参数时：函数内部的变量，也就是形参和实参只是简单的赋值操作，两个数据是独立存储于内存中的。在函数内部对形参进行修改，不会影响外面的变量。

var num=9;function changeNum(num){

num=10;

console.log(num);

}

changeNum(num);//10

console.log(num);//9复制代码

var val='xixi';function a(val){

val=30;

return val;

}

console.log(a(val)); //30

console.log(val); //xixi复制代码

当reference type做函数参数时：还是把实参存储的地址赋值给了形参，在函数内部，形参同样也指向该对象，所以在函数内部对该对象进行修改会影响到外面的变量。

var obj={name:"meryl"}function changeName(para){

para.name="jesscia";

}

changeName(obj);

console.log(obj.name); //jesscia;

/\* 注意：如果在函数内部重新创建对象,为该形参赋值，那么实参和形参这俩对象将不再有关系,修改其中

一个，另外一个不受影响,比如看下面的例子：\*/

var obj2={name:"mohan"}function changeName(para){

para.name="lx";

para={name:'zoe'};

para.name='phoebe';

}

changeName(obj2);

console.log(obj2.name); //lx 复制代码

### Js native objects / Js global objects/ Js中有哪些数据封装类对象？

Js共13类内置构造函数：Object Array Boolean String Number Date Function RegExp Error Global Math JSON

### null  VS undefined  VS undeclared

Undeclared is any variable that has not been declared yet. Console throws an error for this.

Undefined means a variable has been declared but has not yet been assigned a value.

Null is an assignment value, it can be assigned to a variable as a no value.

typeof(undefined); //undefined

typeof(null); //object

null == undefined //true

null===undefined //false复制代码

### i++ VS ++i

|  |  |
| --- | --- |
| i++ | returns the value of a variable before it has been incremented |
| ++i | returns the value of a variable after it has been incremented |

++i is more effective. because i++ can creative a buff variable to save the value before + operator. but ++i won't, like this example: i++ ===> m=1; i=i+1;    ++i===>i=i+1;

### Number.isNaN() VS isNaN()

|  |  |
| --- | --- |
| Number.isNaN ('a') ; // false; | 单纯的判断是不是NaN |
| isNaN('a'); // true | 判断是不是数字 |

### Ways to create an Object in JS? / 创建对象的几种方法（考察原型链/面向对象）

1. Use the bracket's syntax sugar to create an object

var obj={

name:'meryl',

gender:'girl',

sayHi：function(){console.log(`${this.name} is a ${this.gender}`)}

}

//缺点： 不能把json对象当作一个模板来进行new操作复制代码

2. Use Object() constructor to create object

var obj=new Object();

obj.age=19;

obj.show=function(){};

//缺点： 不能把json对象当作一个模板来进行new操作复制代码

3. Use the function constructor to create an object

function Car(){

this.made="斯巴鲁WRX";

this.show=function(){

console.log(this.made);

}

}

var myCar=new Car();//通过构造函数，构造出一个对象出来

myCar.show();//斯巴鲁WRX

/\*

new内部的原理：

第一步:创建一个空对象

第二步:把this指向到那个空对象上

第三步:使空对象的\_proto\_指向构造函数的原型对象

第四步:当构造函数执行完后，如果无return的话,那么就把当前的空对象返回

\*/复制代码

4. Use function constructor+prototype to create an object

function Car(model){

this.made=model;

}

Car.prototype.show=function(){

console.log(this.made);

}

var car1=new Car('beetle');

var car2=new Car('BMW E46');

car1.show();//beetle

car2.show();//BMW E46复制代码

5. Use ES6 class syntax:

class myObj{

constructor(name){

this.name=name;

}

}

var e = new myObj('meryl');复制代码

6. Use Object.create( ), this method creates a new object extending the prototype object passed as parameter.

const person={

isHuman:false,

printInfo:function(){

console.log(`My name is ${this.name}---Am I human? ${this.isHuman}`)

}

}

const me = Object.create(person);

me.name='meryl';

me.isHuman=true; // inherited propertied can be overwritten

me.printInfo(); //My name is meryl---Am I human? true复制代码

### 例举至少3种显式类型转换(explicit type conversion) & 2种隐式类型转换( implicit type conversion)

1. 显式类型转换有：Boolean Number String parseInt parseFloat

2. 隐式类型转换有：+   -    ==    !

### Object的拷贝

* 浅拷贝：以赋值的形式拷贝引用对象，仍指向同一个地址，修改时原对象也会受到影响
  + Object.assign
  + spread operator
* 深拷贝：完全拷贝一个新对象，修改时原对象不再受到任何影响
  + JSON.parse(JSON.stringfy(obj))
    - 当值为函数，undefined或symbol时，无法拷贝
    - 具有循环引用的对象时，报错

### 遍历Object有几种方法？

1. for  in

主要用于遍历对象的可枚举属性，包括自有属性，继承自原型的属性

var obj={name:"meryl", career:'FE'}

Object.defineProperty(obj,"age",{value:"forever21",enumerable:false});

Object.prototype.protoPer1=function(){console.log("proto")};

Object.prototype.protoPer2=2;

console.log("For in:");for(var key in obj){

console.log(key);

}复制代码

输出如下：

2.  Object.keys

返回一个数组，元素均为对象自有的可枚举属性

var obj={name:'meryl', career:'FE'}Object.defineProperty(obj,"age",{value:"forever21",enumerable:false});

Object.prototype.protoPer1=function(){console.log("proto")};

Object.prototype.protoPer2=2;

console.log("Object.keys:");

console.log(Object.keys(obj));

3. Object.getOwnProperty

用于返回对象的自由属性，包括可枚举和不可枚举的

var obj={name:'meryl', career:'FE'}

Object.defineProperty(obj,"age",{value:"forever21",enumerable:false});

Object.prototype.protoPer1=function(){console.log("proto")};

Object.prototype.protoPer2=2;

console.log("Object.getOwnPropertyNames:")

console.log(Object.getOwnPropertyNames(obj));复制代码

### JS中，如何防止object不被修改？

1. use Object.preventExtensions( ) to prevent new methods or properties being added to the object. (我们不能像对象中添加新的属性和方法了，但是还能修改原有的属性)：

var obj={name:'meryl'}

Object.preventExtensions(obj);//阻止篡改对象

obj.age=21;

console.log(obj.age);//undefined

//修改原有的属性：

obj.name='sean';

console.log(obj.name);//sean

2. use Object.seal( ) to seal an object. it not only makes an object as non-extensible but also gets objects' properties[[Configurable]] as false. it means we can't delete properties, but we still can modify the current properties.

var obj={name:'meryl'}

//密封对象

Object.seal(obj);

obj.age=21;

console.log(obj.age); // undefined 不能添加新的属性

delete obj.name;

console.log(obj.name); //meryl 不能删除对象的属性复制代码

3. use Object.freeze( ) to freeze an object. it makes an object as non-extensible and sealed also it set properties' [[Writable]] as false. it means we can't modify current properties.

var obj={name:'meryl'}

//冻结对象

Object.freeze(obj);

obj.age=21;

console.log(obj.age); // undefined 不可扩展

delete obj.name;

console.log(obj.name); //meryl 不可删除

obj.name='sean';

console.log(obj.name);//meryl 不可修改复制代码

三种方法的对比总结如下表所示：

### 你是如何理解JSON的？

* JSON就是JavaScript的其中一个内置对象，其他的还有Math,Date之类的
* JSON内置对象常用的api有：JSON.stringify({a:1,b:2})  JSON.parse('{"a":1,"b":2}')
* JSON也是一个数据结构

### how to restrict the scope of a var variable from 'function' to 'block'?

1. using 'IIFE'(immediately-invoked function expression)：

(function(){

var tmp='test';

})()

console.log(tmp);//Uncaught ReferenceError: tmp is not defined复制代码

2. use "let" keywords

### this是什么，说明this几种不同的使用场景？

this就是函数运行时所在的环境对象，（“this" means the environment object where the function is running.）

**this作为构造函数执行**：就是通过这个构造函数能生成一个新对象，这时this就指向这个新对象

function Fn(name,age){

this.name=name;

}

var f=new Fn('meryl');

f.name// meryl复制代码

**this作为对象属性执行:**

var obj={

a:1,

b:function(){

console.log(this.a);

}

}

obj.b();// this指向obj这个对象

var test=obj.b;test();//undefined 因为this指向了全局,全局中未定义变量a复制代码

**this作为普通函数执行：**这是函数的最常用用法，属于全局性调用，因此this就代表全局对象

var a = 1;function fn(){

console.log(this.a);

}

fn(); //1复制代码

**call, apply, bind:** 他们都是函数的方法，可以改变函数的调用对象

function fn(name,age){

console.log(name);

console.log(this);

}

fn.call({x:100},'meryl',20); //meryl {x:100}复制代码

### 说下对变量提升(hoisting)的理解？

Hoisting is a javascript mechanism where variables and function declarations are moved to the top of their scope before code execution. here is the example:

var a=3;

fn();function fn(){

var b=9;

console.log(a); //undefined

console.log(b); //9

var a=9;

}

console.log(a); //3复制代码

### what's closure? 闭包是什么？

A closure is an inner function that has access to the outer function's variables . The closure has three scope chains: it has access to its own scope, it has access to the outer function's variables, and it has access to global variables. For example:

function counter(){

var count=0;

return function(){

count++;

return count;

}

}

var c1=counter();

c1();//1

c1();//2

var c2=counter();

c2();//1

c2();//2

c1();//3 this is not affected by c2;复制代码

### 实际开发中，闭包有啥应用？

主要用于封装变量，收敛权限。下面的例子中，在isFirstLoad function 外面，是不可能修改掉变量list的值的..

function isFirstLoad(id){

var list=[];

return function(id){

if(list.indexOf(id)>=0){

//说明不是第一次load了

return false;

}

else{

list.push(id);

return true;

}

}

}复制代码

### 为什么JS在浏览器中是单线程的？ 怎么解决单线程阻塞的情形？

因为避免DOM渲染的冲突

用" 异步" 解决， 异步是个**解决方案**

### 单线程和异步什么关系？

单线程就是同时只做一件事，两段JS不能同时执行， 原因就是为了避免DOM渲染冲突，异步算是一种“无奈”的解决方案，虽然有挺多问题（缺点）。比如没按照书写方式执行，可读性较差， callback中不容易模块化。

异步是一个解决方案。具体的实现方式就是event-loop.

### 前端啥时需要异步？/什么时候不能阻塞程序运行？

1. 定时任务：setTimeout, setInterval
2. 网络请求：ajax请求， 动态<img>加载
3. 事件绑定

### 当前异步的解决方案有哪些？

1. Jquery Deferred
2. Promise
3. Async/await
4. Generator

### 什么是event-loop? (事件循环)

 event-loop是JS实现异步的具体实现方式，原则是同步代码直接按顺序执行，异步代码先放在异步队列（queue）中，等所有的同步代码/同步函数执行完毕后，轮询执行异步队列（queue）中的函数。

//例子1:setTimeout(function(){

console.log(1);

})

console.log(2);

/\*

结果： 先输出2,后1

主进程：console.log(2);

异步队列：function(){console.log(1)}

\*/

//例子2：setTimeout(function(){

console.log(1);

}，100)setTimeout(function(){

console.log(2);

})

console.log(3);

/\*

结果： 先输出3,后2,最后1

主进程：console.log(3);

异步队列：function(){console.log(2)}立刻被放入queue中,

100ms之后function(){console.log(1)}被放入queue中

\*/

//例子3：

$.ajax({

url:'http://google.com',

success:function(result){console.log('a');}

});setTimeout(function(){console.log('b')},100);setTimeout(function(){console.log('c')});console.log('d');

/\*

结果1： 先输出d,后c, 再a, 最后b

结果2： 先输出d,后c, 再b, 最后a

主进程：console.log('d');

异步队列：function(){console.log(c)}最先被放入queue中,

.ajax或function(){console.log('b')}是不确定哪一个最被先放进去queue中的.

\*/复制代码

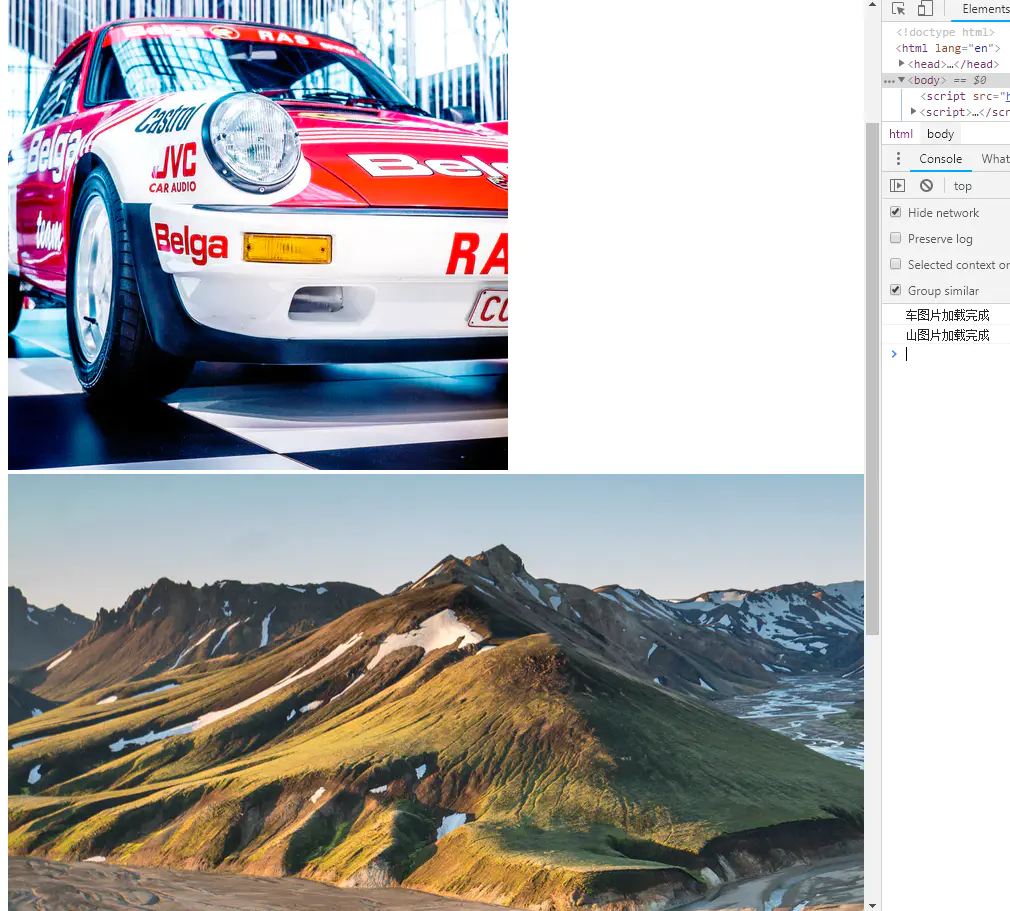
### What's the promise? How to use promise?

* Promise is like a container. In this container have the result of future things.
* Promise is a cleaner way for running asynchronous tasks.
* Promise use methods like ".then" and "catch" to deal with all kinds of asynchronous operation

Here is a promise example:



结果如下：



### new关键字做了什么？ (构造函数考点)

//构造函数function Person(name,age,job){

this.name=name;

this.age=age;

this.job=job;

}

Person.prototype.Intro=function(){

console.log(this.name+this.age)

}

Person.prototype.showJob=function(){

console.log(this.job)

}

//实例对象

var p1 = new Person('meryl',25,'web developer');复制代码

new关键字和构造函数搭配使用，拿上面例子而言，其中背后的操作有分下面几个步骤：

1. first create an empty object: p1{}  
2. set the 'this' to be that empty object: this=== p1;  
3. add the line 'return this' to the end of the constructor function:

function Person(name,age,job){

this.name=name;

this.age=age;

this.job=job;

return this;

}复制代码

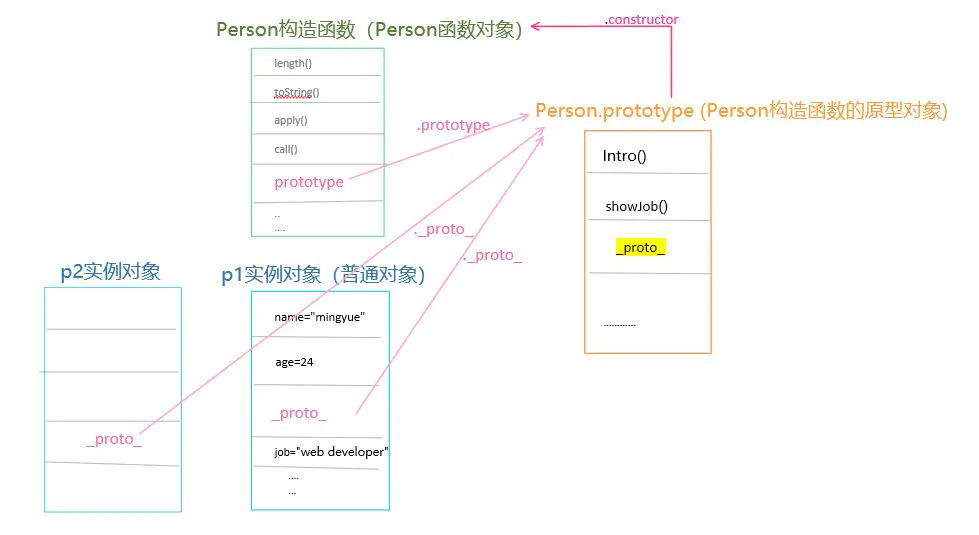
4. set the instance object (p1) 's \_proto\_ links to the constructor function's prototype:

p1.\_proto\_= Person.prototype (Person构造函数的原型对象)

*注意：*

1. Person构造函数的实例对象都是相互独立的， 即数据各保留自己的，但是方法（Intro.showJob）都指向同一个原型对象（Person.prototype）
2. 原型对象（Person.prototype）的作用是用来”装“共享方法(Intro,showJob)和共享属性的
3. 只有函数对象有prototype属性，普通对象没有prototype属性，但是有\_proto\_属性

下图是构造函数，实例对象和原型对象的关系图：



### 原型规则有哪些呢？

1. 所有的引用类型（array, function, object）都具有对象特性， 即 可自由扩展属性 （除了null以外）

自由扩展属性就比如：

var obj ={ };

obj.a=100;

var arr=[ ];

arr.a=100;

function fn(){ }

fn.a=100;复制代码

2.  所有的引用类型（array, function, object）都有一个\_proto\_属性（隐式原型），属性值是一个普通的对象

console.log(obj.\_proro\_); //\_proto\_是一个对象

console.log(arr.\_proro\_); //\_proto\_是一个对象 console.log(fn.\_proro\_); //\_proto\_是一个对象复制代码

3. 所有的函数，都有一个prototype属性(显示原型),属性值也是一个普通的函数

console.log(fn.prototype); //prototype也是一个对象复制代码

4. 所有的引用类型（array, function, object）的\_proto\_属性值指向它的构造函数的prototype属性值  **(构造函数的显式原型===实例对象的隐式原型)**

console.log( obj.\_proto\_ === Object.prototype) //true 因为obj的构造函数就是Object

console.log( arr.\_proto\_ === Object.prototype) //true 因为arr的构造函数就是Object

console.log( fn.\_proto\_ === Object.prototype) //true 因为fn的构造函数就是Object复制代码

5. 当试图得到一个对象的某个属性时，如果这个对象本身没有这个属性，那么会去它的\_proto\_（即它的构造函数的prototype）中寻找.

function Foo(name,age){

this.name=name;

}

Foo.prototype.alertName=function(){alert(this.name);}

var f = new Foo('meryl');

f.printName=function(){console.log(this.name);} //printName是对象自身新添的一个属性

f.printName()// this指向f,不用去f的构造函数的原型中找printName这个属性

f.alertName()// this也指向f,其中alertName函数是f对象的原型中的函数复制代码

### 什么是JS prototype 和 JS prototype chain?

**what's the prototype：**

The prototype is the property of constructor function, and It's an object. This prototype object can share methods and values among the instance objects. (每创建一个函数，函数上都会有一个属性为prototype. 它的值是一个对象，这个对象的作用就是当使用函数创建实例的时候，那么那些实例都会共享原型上的属性和方法)

**what's the prototype chain?**

The prototype chain is made of prototype objects. Each instance object has a property called '\_proto\_', this proto point to the object's prototype, but prototype also has his own prototype. In the final. The prototype will be null; This chain the structure is a prototype chain. (在JS中.每个实例对象都有一个指向它原型对象(prototype)的内部链接叫\_proto\_, 这个原型对象又有自己的原型，直到某个对象的原型为null为止 ，这种一级级的链式结构就是原型链，当查找一个对象的属性时，js会向上遍历原型链，直到找到给定名称的属性为止，到查找到达原型链的顶部,若仍然没找到指定的属性，就会返回undefined)

### 写一个原型链继承的例子？

这是一个模仿jQuery中html(), on函数的例子：

//一个封装DOM的例子：function Elem (id){

this.elem=document.getElementById(id);

}

Elem.prototype.html=function(val){

var elem=this.elem;

if(val){

elem.innerHTML=val;

return this;//链式操作

}else{

return elem.innerHTML;

}

}

Elem.prototype.on=function(type,fn){

var elem=this.elem;

elem.addEventListener(type,fn);

return this;

}

var div1=new Elem('div1');

div1.html('<p>hello world</p>').on('click',function(){alert('clicked!')}).html('<p>JS</p>')复制代码

### instanceof的原理是什么？

 This operator is used to check the type of an object at run time. The instanceof operator returns a boolean value that indicates if an object is an instance of a particular class.(instanceof被用来判断引用类型属于哪个构造函数).  instanceof工作原理就是按着\_proto\_(隐式原型)往上找到prototype(显示原型)，后来进行隐显式原型比对，一样的话就返回true,不一样则继续往上找(因为prototype object中也有\_proto\_)

### Js event flow model(事件流模型)是什么？

事件流模型包含三个阶段：

* 事件捕捉阶段：事件开始由顶层对象触发，然后逐级向下传播，直到目标的元素
* 处于目标阶段：处于绑定事件的元素上
* 事件冒泡阶段：事件由具体的元素先接收，然后逐级向上传播，直到不具体的元素

### what's the event propagation？

Event propagation including **event bubbling** and **event capturing**, when an event occurs in an element inside another element and both elements have registered a handler for that event. The event propagation mode determines in [which order the elements receive the event](http://www.quirksmode.org/js/events_order.html).

With bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.

With capturing, the event is first captured by the outermost element and propagated to the inner elements.

### What is event bubbling?/Define event bubbling

Because js allows DOM elements to be nested inside each other. If the handler of the child is clicked, the handler of the parent will also work as if it were clicked too. That sums up what event bubbling really is.

### what's Event Delegation?

Event Delegation is the method. this method will let you create one event listener to trigger all child elements. Each child element will be targeted by a unique ID.

### e.target VS e.currentTarget ?

|  |  |
| --- | --- |
| e.target | The thing under the mouse (the thing that triggers the event) |
| e.currentTarget | is the element with the added event listener |

### What is cookie？

* It was designed for communication between server and client side
* cookie also can save data, but it has many disadvantages.
* We can use "document.cookie = …." to get or edit cookie.

### cookie用于存储的缺点

* 存储量太小，只有4KB
* 所有http请求都带着cookie,会影响获取资源的效率

### 描述一下cookie 和 sessionStorage，localStorage的区别？

* 容量
  + cookie最大容量4kb, sessionStore & localStorage最大5M
* 是否会携带到ajax中
  + 所有的http请求都带着cookie
  + http请求中无sessionStotage和localStorage. (sessionStotage和localStorage被设计来专门存放数据)；
* Api易用性
  + cookie需要程序员自己做封装
  + sessionStorage和localStorage的api简单易用，常见的api有：.setItem(key,value); .getItem(key,value);
* 存在理由：
  + cookie是服务器进行交互，作为http规范的一部分而存在
  + localStorage和sessionStorage是专门为了在本地“存储”数据而生

### 判断是否是数组有几种方法？

1. 使用Array.isArray( )

function is\_array(arr){

if(Array.isArray(arr)){

return true;

}else{

return false;

}

}复制代码

2. 使用instanceof Array

function is\_array(arr){

return arr instanceof Array;

3. 使用Object.prototype.toString.call( )=='[object Array]'

function is\_array(arr){

return Object.prototype.toString.call(arr)=='[object Array]';

}复制代码

### 实现一个函数 clone()，可以对 JavaScript 中的5种主要的数据类型（包括Number、String、Object、Array、Boolean）进行值复制

这个问题考察了的知识点有：使用typeof判断值类型，使用toString区分数组和对象；递归函数的使用..

function clone(obj){

//判断是对象,就进行循环赋值

if(typeof obj ==='object' && typeof obj !=='null'){

//区分是数组还是对象，创建空的数组或对象

var o = Object.prototype.toString.call(obj)==='[object Array]'?[]:{};

for(var k in obj){

//如果属性对应的值为对象，则递归赋值

if(typeof obj[k] ==='object' && typeof obj[k] !=='null'){

o[k]=clone(obj[k])

}else{

o[k]=obj[k];

}

}

}

else{

//不是对象，直接把值返回

return obj;

}

retrun o;

}复制代码

### what is the use of async and defer attributes used inside <script> tag?

* The defer attribute means delaying script execution until the HTML parse has finished. A nice effect of this attribute is that the DOM will be available for your script.
* The async attribute means the script is executed asynchronously while the page continues parsing. Since we do not care exactly at which point the file is executed, asynchronous loading is the most suitable option.

### what's jQuery method chaining? Why the method could be a chain? plz write implementation by using JS.

**what's Jquery chaining?**

jquery chaining allows us to operate multiple actions on the same set of elements, all within a single line of code. like: $('div').html().css().removeClass()…

**why the method could be a chain?**

Because it returning a reference to this when the method finishes.

write example:

var testObj={

html:function(){console.log('method1');return this}

css: function(){console.log('method2');return this}

}testObj.html().css();复制代码

### what is $(function(){ }) mean in Jquery?

1. $(function(){ }) is short for $(document).ready(function(){ });
2. it means code will run until DOM was loaded.

### what's a Document Object Model(DOM)?

* DOM is short for "Document object model";
* DOM is a conceptual way of visualizing a document and its contents.
* we can use javascript to access and edit the DOM, like adding, changing, deleting elements, etc.

### DOM事件的级别是什么意思？

|  |  |
| --- | --- |
| DOM0 | element.onclick=function(){ … } |
| DOM2 | element.addEventListener( 'click' , function(){…} , false ) 第三个参数默认是false,表示event bubbling |
| DOM3 | element.addEventListener( 'keyup' ,function(){…},false ) DOM3事件类型增多了，比如鼠标 键盘事件 |

### DOM事件流（DOM event flow）是什么？

The typical DOM event flow is conceptually divided into three phases: **capture** phase，     **target** phase and **bubbling** phase.

### What is reflow? What causes reflow? How could you reduce reflow?

**what is reflow?**

Reflow is the web browser process for re-calculating the positions of elements in the document, for the purpose of re-rendering the document.

**what caused reflow?**

* change layout (geometry of the page)
* resize the window
* change height/width of an element
* changing font
* change font size
* move DOM element (animation)
* adding or removing stylesheet
* calculating offset height or offset width
* display: none;

**How to reduce reflow?**

1. [Change classes on the element you wish to style (as low in the dom tree as possible)](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#low)
2. [Avoid setting multiple inline styles](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#inline)
3. [Apply animations to elements that are position fixed or absolute](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#animations)
4. [Trade smoothness for speed](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#smooth)
5. [Avoid tables for layout](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#tables)
6. [Avoid JavaScript expressions in the CSS](http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/#expressions) (IE only)

### ES6 新特性有什么？

* Default Parameters in ES6.
* Template Literals in ES6.
* Multi-line Strings in ES6.
* Destructuring Assignment in ES6.
* Enhanced Object Literals in ES6.
* Arrow Functions in ES6.
* Promises in ES6.
* Block-Scoped Constructs Let and Const.

### arrow function VS traditional function

* arrow function are less verbose than traditional function.
* arrow function has a lexical this, its value is determined by the surrounding scope. the traditional function has a dynamic this, its value is determined by how the function is called.
* arrow function can not be a Constructor
* arrow function has no arguments:

let fn=(...args)=>{console.log(args)};

fn(1,2,'rest'); //[1,2,'rest']

function fn2(){

console.log(arguments);

}

fn2(1,2,'rest');复制代码

### class和普通的构造函数有啥区别？

* class在语法上更贴合"面向对象"的写法
* class实现继承更加容易理解
* class本质就是语法糖，使用的还是prototype