1. javascript的typeof返回哪些数据类型.

```
答案：string,boolean,number,undefined,function,object
```

2. 例举3种强制类型转换和2种隐式类型转换?

```
答案：强制（parseInt,parseFloat,number）
隐式（== === ）
```

3. split() join() 的区别

```
答案：前者是将字符串切割成数组的形式，后者是将数组转换成字符串
```

4. 数组方法pop() push() unshift() shift()

```
答案：push()尾部添加 pop()尾部删除
unshift()头部添加 shift()头部删除
```

5. IE和标准下有哪些兼容性的写法

```
var ev = ev || window.event
document.documentElement.clientWidth || document.body.clientWidth
Var target = ev.srcElement||ev.target
```

6. ajax请求的时候get 和post方式的区别

```
答案：
一个在url后面 ，一个放在虚拟载体里面
get有大小限制(只能提交少量参数)
安全问题
应用不同 ，请求数据和提交数据
```

7. this问题

```
// 在函数中this指向谁:
  // 函数中的this指向谁,是由函数被调用的那一刻就确定下来的


  // 想要判断函数中的this指向谁,遵循两条原则:
  // 1. 我们要判断的this在哪个函数中

  // 2. 这个函数是那种调用模式调用的
```

```
8
9     // function fn(){
10    //     console.log(this);
11    // }
12
13    // 普通函数调用://this --> window
14    // fn();
15
16    // //对象调用 this -->obj
17    // var obj = {};
18    // obj.f = fn;
19    // obj.f();  //this -->obj
20
21    // // new 调用函数 this --> 新创建出来的实例对象
22    // var f = new fn();
23
24    // // 注册事件  this --> box
25    // box.onclick = fn;
26
27    // // 定时器  this --> window
28    // setInterval(fn,1000);
29
30
31
32  // 上下文调用模式: 其实就是js中提供给我们的三个方法.而这三个方法的作用就是随意控制函数中this的指向
33
34    // call
35    // 函数.call(第一个参数:想让函数中this指向谁,就传谁进来,
36    // 后面的参数:本身函数需要传递实参,需要几个实参,就一个一个的传递即可);
37    // call的作用: 1. 调用函数 2.指定函数中this指向
38    // apply
39    // 函数.apply(第一个参数:想让函数中this指向谁,就传谁进来,
40    // 第二个参数:要求传入一个数组,数组中包含了函数需要的实参)
41    // apply的作用: 1. 调用函数 2, 指定函数中this的指向
42
43    //   bind
44    // 函数.bind(第一个参数:想让函数中this指向谁,就传谁进来,
45    // 后面的参数:本身函数需要传递实参,需要几个实参,就一个一个的传递即可)
46    // bind的作用: 1. 克隆当前函数,返回克隆出来的新的函数
47    //             2. 新克隆出来的函数,这个函数的this被指定了
48    function fn(x, y){
49        console.log(this);
50        console.log(x + y);
51    }
52
53    // fn.call([1,3,4], 5, 8);
54    // fn.apply({o:1}, [12,13]);
55    // var f = fn.bind({a:1},3,5);
56    var f = fn.bind({a:1});
57    console.log(f);
58    f(4,5);
59
60
```

```
// 上下文调用模式的三个方法的总结:
// call, apply   这两个方法都会调用函数
// call, bind   这两个方法,后面的传参方式是一样的
// bind方法不会调用函数,只会克隆一个新的函数出来,这个新的函数中this已经被指定了
// apply方法第二个参数,要求传入一个数组,这个数组中包含函数需要的实参

```

8. ajax请求时，如何解析json数据

```
答案：使用JSON.parse
```

9. 事件委托是什么

```
答案: 利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！
<style>
   #grandfar{
     width: 300px;
     height: 300px;
     background-color: pink;
   }

   #far{
     width: 200px;
     height: 200px;
     background-color: green;
   }

   #son{
     width: 100px;
     height: 100px;
     background-color: red;
   }
 </style>
</head>
<body>
<div id="grandfar">
  <div id="far">
    <div id="son"></div>
  </div>
</div>
<script>
//  1.只要点击/移入...发生了,就一定触发了事件,跟我们有没有注册事件没有关系
//  2.一旦,触发了事件,就会产生一个事件流.
//  3.然后马上要确定下来一个事件路径

//  假如点击了grandfar,那么事件路径就是: window - > document - > body > grandfar
//  假如点击了son,那么事件路径就是: window - > document - > body > grandfar  --> far --> son

   var son = document.getElementById('son');
   var far = document.getElementById('far');
   var grandfar = document.getElementById('grandfar');
```

```
39
40  //    son.onclick  = function(){
41  //      console.log('son的点击事件');
42  //    }
43      far.onclick  = function(){
44        console.log('far的点击事件');
45      }
46      grandfar.onclick  = function(){
47        console.log('grandfar的点击事件');
48      }
49      document.body.onclick = function(){
50        console.log('body的点击事件');
51      }
52
53      事件委托
54        <style>
55      * {
56        margin: 0;
57        padding: 0;
58      }
59
60      div {
61        border: 1px solid blue;
62      }
63
64      ul {
65        padding: 20px;
66      }
67
68      li {
69        list-style: none;
70        margin-top: 10px;
71        background-color: skyblue;
72        height: 30px;
73        line-height: 30px;
74      }
75
76      button {
77        width: 100%;
78        height: 50px;
79        background-color: white;
80      }
81    </style>
82  </head>
83
84  <body>
85
86  <div>
87    <ul id="ul">
88      <li>你见，或者不见我    </li>
89      <li>我就在那里    </li>
90      <li>不悲不喜        </li>

91      <li>你念，或者不念我    </li>
```

```
    <li>情就在那里       </li>
    <li>不来不去       </li>
    <li>你爱，或者不爱我    </li>
    <li>爱就在那里    </li>
    <li>不增不减    </li>
  </ul>
  <button id="btn">点击加载更多..</button>
</div>

<script>
  var arr = [
    '你跟，或者不跟我',
    '我的手就在你手里',
    '不舍不弃',
    '来我的怀里',
    '或者',
    '让我住进你的心里',
    '默然    相爱',
    '寂静    喜欢',
  ]

  //       需求:
  // 1.无序列表中每一个li都有点击事件,点击之后,把对应的文本打印到控制台上
//  //       1. 获取元素 li
//  var lis = document.querySelectorAll('li');
  var ul= document.querySelector('#ul');
//
//  //       2. 给每一个li注册点击事件
//  for (var i = 0; i < lis.length; i++) {
//    lis[i].onclick = fn;
//  }
//  function fn() {
//    //       3. 在事件处理函数中,打印对应的文本
//    console.log(this.innerText);
//  }

  //事件委托: 本来是自己做的事件,委托给父级元素
//  事件委托的优点:
//  1. 代码简洁
//  2. 节省内存
//  事件委托的原理:
//    事件流(事件冒泡)
  ul.onclick = function(e){
    //找到点的是那个li
    console.log(e.target.innerText);
  }


//  2. 点击按钮,加载更多
//  2.1 获取元素
  var btn = document.querySelector('#btn');
//  2.2 给按钮注册点击事件

  btn.addEventListener('click', function(){
```

```
    //  2.3 在事件处理函数中,动态的创建li,添加到ul中
    for(var i = 0; i < arr.length; i++) {
      var li = document.createElement('li');
      li.innerText = arr[i];
//      li.onclick = fn; //由于后创建的没有注册事件,所以必须给他们在注册一遍
      ul.appendChild(li);
    }
  }, false);

```

10. 闭包是什么，有什么特性，对页面有什么影响

```
我们通俗理解的闭包: 一个内部函数引用了外部函数的变量,外部函数形成了一个闭包
闭包的作用:缓存数据,延长作用域链
闭包的优点和缺点:缓存数据


   //函数模式的闭包:在一个函数中有一个函数
//    function f1() {
//      var num=10;
//      //函数的声明
//      function f2() {
//        console.log(num);
//      }
//      //函数调用
//      f2();
//    }
//    f1();

   //对象模式的闭包:函数中有一个对象

//    function f3() {
//      var num=10;
//      var obj={
//        age:num
//      };
//      console.log(obj.age);//10
//    }
//    f3();




//    function f1() {
//      var num=10;
//      return function () {
//        console.log(num);
//        return num;
//      }
//    }
//
//   var ff= f1();
//   var result= ff();
```

```
//    console.log(result);


//    function f2() {
//      var num=100;
//      return {
//        age:num
//      }
//    }
//
//   var obj= f2();
//    console.log(obj.age);

```

11. 如何阻止事件冒泡

答案：ie:阻止冒泡ev.cancelBubble = true;非IE ev.stopPropagation();

12. 如何阻止默认事件

答案：(1)return false；(2) ev.preventDefault();

13. 添加 删除 替换 插入到某个接点的方法

答案：

1）创建新节点 createElement() //创建一个具体的元素 createTextNode() //创建一个文本节点

2）添加、移除、替换、插入 appendChild() //添加 removeChild() //移除 replaceChild() //替换 insertBefore() //插入

3）查找 getElementsByTagName() //通过标签名称 getElementsByName() //通过元素的Name属性的值 getElementById() //通过元素Id，唯一性

14. 解释jsonp的原理，以及为什么不是真正的ajax

答案：动态创建script标签，回调函数 Ajax是页面无刷新请求数据操作

15. document load 和document ready的区别

答案：document.onload 是在结构和样式,外部js以及图片加载完才执行js document.ready是dom树创建完成就执行的方法，原生种没有这个方法，jquery中有 $().ready(function)

16. ”==”和“===”的不同

答案：前者会自动转换类型,再判断是否相等 后者不会自动类型转换，直接去比较

17. 函数声明与函数表达式的区别？

在Javscript中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

18. 对作用域上下文和this的理解，看下列代码：

```
var User = {
 count: 1,
 getCount: function() {
  return this.count;
 }
};
console.log(User.getCount()); // what?
var func = User.getCount;
console.log(func()); // what?
问两处console输出什么？为什么？
答案:是1和undefined。
      func是在window的上下文中被执行的，所以不会访问到count属性。
```

21. Javascript的事件流模型都有什么?

```
“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

```

25. javaScript的2种变量范围有什么不同？

全局变量：当前页面内有效

局部变量：函数方法内有效

26. null和undefined的区别？

null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。

当声明的变量还未被初始化时，变量的默认值为undefined。 null用来表示尚未存在的对象

undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。典型用法是：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2）调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

null表示"没有对象"，即该处不应该有值。典型用法是：

（1）作为函数的参数，表示该函数的参数不是对象。

（2）作为对象原型链的终点。

27. new操作符具体干了什么呢?

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

38. 列举javaScript的3种主要数据类型，2种复合数据类型和2种特殊数据类型。

主要数据类型：string, boolean, number

复合数据类型：function, object

特殊类型：undefined，null

41. 解释什么是Json:

(1)JSON 是一种轻量级的数据交换格式。

(2)JSON 独立于语言和平台，JSON 解析器和 JSON 库支持许多不同的编程语言。

(3)JSON的语法表示三种类型值，简单值(字符串，数值，布尔值，null),数组，对象

45. 可视区的大小：

(1)innerXXX（不兼容ie）

window.innerHeight 可视区高度，包含滚动条宽度

window.innerWidth 可视区宽度，包含滚动条宽度

(2)document.documentElement.clientXXX(兼容ie)

document.documentElement.clientWidth可视区宽度，不包含滚动条宽度

document.documentElement.clientHeight可视区高度，不包含滚动条宽度

46. 节点的种类有几种，分别是什么？

(1)元素节点：nodeType ===1;

(2)文本节点：nodeType ===3;

(3)属性节点：nodeType ===2;

47. innerHTML和outerHTML的区别

innerHTML(元素内包含的内容）

outerHTML(自己以及元素内的内容）

48. offsetWidth offsetHeight和clientWidth clientHeight的区别

(1)offsetWidth （content宽度+padding宽度+border宽度）

(2)offsetHeight（content高度+padding高度+border高度）

(3)clientWidth（content宽度+padding宽度）

(4)clientHeight（content高度+padding高度）

49. 闭包的好处

(1)希望一个变量长期驻扎在内存当中(不被垃圾回收机制回收)

(2)避免全局变量的污染

(3)私有成员的存在

(4)安全性提高

50. 冒泡排序算法

```
冒泡排序
var array = [5, 4, 3, 2, 1];
var temp = 0;
for (var i = 0; i <array.length; i++){
for (var j = 0; j <array.length - i; j++){
if (array[j] > array[j + 1]){
temp = array[j + 1];
array[j + 1] = array[j];
array[j] = temp;
 }
}
```

79、dom事件委托什么原理，有什么优缺点

事件委托原理:事件冒泡机制

优点

1.可以大量节省内存占用，减少事件注册。比如ul上代理所有li的click事件就很不错。 2.可以实现当新增子对象时，无需再对其进行事件绑定，对于动态内容部分尤为合适

缺点

事件代理的常用应用应该仅限于上述需求，如果把所有事件都用事件代理，可能会出现事件误判。即本不该被触发的事件被绑定上了事件。

85、vue双向数据绑定的原理是什么

首先传输对象的双向数据绑定 Object.defineProperty(target, key, decription),在decription中设置get和set属性（此时应注意description中get和set不能与描述属性共存）数组的实现与对象不同。 同时运用观察者模式实现wather，用户数据和view视图的更新

86、 react和vue比较来说有什么区别

1 component层面，web component和virtual dom 2 数据绑定（vue双向，react的单向）等好多 3 计算属性 vue 有，提供方便；而 react 不行 4 vue 可以 watch 一个数据项；而 react 不行 5 vue 由于提供的 direct 特别是预置的 directive 因为场景场景开发更容易；react 没有 6 生命周期函数名太长 directive

88、 网页布局有哪几种，有什么区别

静态、自适应、流式、响应式四种网页布局 静态布局：意思就是不管浏览器尺寸具体是多少，网页布局就按照当时写代码的布局来布置； 自适应布局：就是说你看到的页面，里面元素的位置会变化而大小不会变化； 流式布局：你看到的页面，元素的大小会变化而位置不会变化——这就导致如果屏幕太大或者太小都会导致元素无法正常显示。 自适应布局：每个屏幕分辨率下面会有一个布局样式，同时位置会变而且大小也会变。

95、 http协议属于七层协议中的哪一层，下一层是什么

七层结构：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层 tcp属于传输层；http属于应用层。表现层

98、 websocket长连接原理是什么

含义

Websocket是一个持久化的协议，相对于HTTP这种非持久的协议来说。

原理

类似长轮循长连接；发送一次请求；源源不断的得到信息

100、 理解web安全吗？都有哪几种，介绍以及如何预防

1.XSS，也就是跨站脚本注入

```
攻击方法：
1\. 手动攻击:
编写注入脚本，比如”/><script>alert(document.cookie());</script><!--等，
手动测试目标网站上有的input, textarea等所有可能输入文本信息的区域
2\. 自动攻击
利用工具扫描目标网站所有的网页并自动测试写好的注入脚本，比如：Burpsuite等
防御方法：
1\. 将cookie等敏感信息设置为httponly，禁止Javascript通过document.cookie获得
2\. 对所有的输入做严格的校验尤其是在服务器端，过滤掉任何不合法的输入，比如手机号必须是数字，通常可以采用正则表达式
3\. 净化和过滤掉不必要的html标签，比如：<iframe>, alt,<script> 等
4\. 净化和过滤掉不必要的Javascript的事件标签，比如：onclick, onfocus等
5\. 转义单引号，双引号，尖括号等特殊字符，可以采用htmlencode编码 或者过滤掉这些特殊字符
6\. 设置浏览器的安全设置来防范典型的XSS注入

作者：O蚂蚁O
链接：https://www.jianshu.com/p/f1f39d5b2a2e
来源：简书
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。
```

2.SQL注入\

```
攻击方法：
编写恶意字符串，比如' or  1=1--等，
手动测试目标网站上所有涉及数据库操作的地方
防御方法：
1\. 禁止目标网站利用动态拼接字符串的方式访问数据库
2\. 减少不必要的数据库抛出的错误信息
3\. 对数据库的操作赋予严格的权限控制
4\. 净化和过滤掉不必要的SQL保留字，比如：where, or, exec 等
5\. 转义单引号，上引号，尖括号等特殊字符，可以采用htmlencode编码 或者过滤掉这些特殊字符
```

3.CSRF，也就是跨站请求伪造

```
就是攻击者冒用用户的名义，向目标站点发送请求
防范方法：
1\. 在客户端进行cookie的hashing，并在服务端进行hash认证
2\. 提交请求是需要填写验证码
3\. 使用One-Time Tokens为不同的表单创建不同的伪随机值
```

101、 sessionStorage和localstorage能跨域拿到吗？比如我在www.baidu.com设置的值能在m.baidu.com能拿到吗？为什么

localStorage会跟cookie一样受到跨域的限制，会被document.domain影响

102、 localstorage不能手动删除的时候，什么时候过期

除非被清除，否则永久保存 clear()可清楚 sessionStorage 仅在当前会话下有效，关闭页面或浏览器后被清除

109、.四种定位的区别

`static` 是默认值 `relative` 相对定位 相对于自身原有位置进行偏移，仍处于标准文档流中 `absolute` 绝对定位 相对于最近的已定位的祖先元素, 有已定位(指 `position` 不是 `static` 的元素)祖先元素, 以最近的祖先元素为参考标准。如果无已定位祖先元素, 以 `body` 元素为偏移参照基准, 完全脱离了标准文档流。 `fixed` 固定定位的元素会相对于视窗来定位,这意味着即便页面滚动，它还是会停留在相同的位置。一个固定定位元素不会保留它原本在页面应有的空隙。

128、WebSocket

HTML5带来的新协议，通过类似HTTP的请求建立连接。主要目的是可以获取服务端的推送。 原来的方式可能是使用long poll（即不中断连接一直等待数据），或者是ajax轮询的方式（每隔一段时间发送请求，建立连接，询问是否有新的数据）。这两种方式的缺点在于long poll的阻塞，以及ajax轮询的冗余连接。 WebSocket的设计思想有点类似于回调，在发送请求升级服务端的协议并收到确认信息后，服务端一有新的信息/数据就会主动推送给客户端，至于要一次HTTP握手便可以建立持久连接

133、forEach和map的区别

相同点

- 都是循环遍历数组中的每一项

- forEach和map方法里每次执行匿名函数都支持3个参数，参数分别是item（当前每一项）、index（索引值）、arr（原数组）
- 匿名函数中的this都是指向window
- 只能遍历数组
- 都有兼容问题

不同点

- map速度比foreach快
- map会返回一个新数组，不对原数组产生影响,foreach不会产生新数组，
- map因为返回数组所以可以链式操作，foreach不能

134、浅拷贝和深拷贝

jQuery.extend第一个参数可以是布尔值，用来设置是否深度拷贝的

```
jQuery.extend(true, { a : { a : "a" } }, { a : { b : "b" } } );
jQuery.extend( { a : { a : "a" } }, { a : { b : "b" } } );
```

最简单的深拷贝

```
aa = JSON.parse( JSON.stringify(a) )
```

浅复制--->就是将一个对象的内存地址的""编号""复制给另一个对象。深复制--->实现原理，先新建一个空对象，内存中新开辟一块地址，把被复制对象的所有可枚举的(注意可枚举的对象)属性方法一一复制过来，注意要用递归来复制子对象里面的所有属性和方法，直到子子.....属性为基本数据类型。总结，深复制理解两点，1,新开辟内存地址，2,递归来刨根复制。

52、javascript继承的 6 种方法？

1. 原型链继承
2. 借用构造函数继承
3. 组合继承(原型+借用构造)
4. 原型式继承
5. 寄生式继承
6. 寄生组合式继承

54、JavaScript 原型，原型链？有什么特点？

1. 原型对象也是普通的对象，是对象一个自带隐式的 **proto** 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 null 的话，我们就称之为原型链

59、说说你对this的理解？

在JavaScript中，this通常指向的是我们正在执行的函数本身，或者是，指向该函数所属的对象。

全局的this → 指向的是Window

函数中的this → 指向的是函数所在的对象 错误答案

对象中的this → 指向其本身

事件中this → 指向事件对象原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

56、简述一下Sass、Less，且说明区别？

他们是动态的样式语言，是CSS预处理器,CSS上的一种抽象层。他们是一种特殊的语法/语言而编译成CSS。

变量符不一样，less是@，而Sass是$;

Sass支持条件语句，可以使用if{}else{},for{}循环等等。而Less不支持;

Sass是基于Ruby的，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出Css到浏览器

JavaScript中常用的设计模式

模式分类	名称
创建型	工厂模式
	单例模式
	原型模式
结构型	适配器模式
	代理模式
行为型	策略模式
	迭代器模式
	观察者模式(发布-订阅模式)
	命令模式
	状态模式

创建型模式之单例模式

单例模式（singleton）:又称为单体模式，只允许实例化一次的对象类。

用一个对象来规划一个命名空间，减少网页中全局变量的数量。从全局命名空间里提供一个唯一的访问点来访问该对象。

滑动效果