- 对于协商缓存， 第一次请求缓存且保存缓存标识与时间， 重复请求向服务器发送缓存标识和最后缓存时间， 服务端进行校验， 如果失效则使用缓存

协商缓存相关设置

- `Exprires` ：服务端的响应头， 第一次请求的时候， 告诉客户端，该资源什么时候会过期。 `Exprires` 的缺陷是必须保证服务端时间和客户端时间严格同步。
- `Cache-control: max-age` ：表示该资源多少时间后过期，解决了客户端和服务端时间必须同步的问题，
- `If-None-Match/ETag` ：缓存标识，对比缓存时使用它来标识一个缓存， 第一次请求的时候， 服务端会返回该标识给客户端，客户端在第二次请求的时候会带上该标识与服务端进行对比并返回 `If-None-Match` 标识是否表示匹配。
- `Last-modified/If-Modified-Since` ：第一次请求的时候服务端返回 `Last-modified` 表明请求的资源上次的修改时间， 第二次请求的时候客户端带上请求头 `If-Modified-Since` ，表示资源上次的修改时间， 服务端拿到这两个字段进行对比

103 现在要你完成一个Dialog组件，说说你设计的思路？ 它应该有什么功能?

- 该组件需要提供 `hook` 指定渲染位置， 默认渲染在body下面 。
- 然后改组件可以指定外层样式， 如宽度等
- 组件外层还需要一层 `mask` 来遮住底层内容，点击 `mask` 可以执行传进来的 `onCancel` 函数关闭 `Dialog` 。
- 另外组件是可控的， 需要外层传入 `visible` 表示是否可见 。
- 然后 `Dialog` 可能需要自定义头head和底部 `footer` ， 默认有头部和底部，底部有一个确认按钮和取消按钮，确认按钮会执行外部传进来的 `onOk` 事件，然后取消按钮会执行外部传进来的 `onCancel` 事件。
- 当组件的 `visible` 为 `true` 时候，设置 `body` 的 `overflow` 为 `hidden` ， 隐藏 `body` 的滚动条， 反之显示滚动条。
- 组件高度可能大于页面高度， 组件内部需要滚动条。
- 只有组件的 `visible` 有变化且为 `ture` 时候，才重渲染组件内的所有内容

104 `caller` 和 `callee` 的区别

callee

> `caller` 返回一个函数的引用， 这个函数调用了当前的函数。

使用这个属性要注意

- 这个属性只有当函数在执行时才有用
- 如果在 javascript 程序中， 函数是由顶层调用的，则返回 null

> functionName.caller: functionName 是当前正在执行的函数。

```js
function a() {
  console.log(a.caller)
}
```

callee

> callee 放回正在执行的函数本身的引用， 它是 arguments 的一个属性

> 使用callee时要注意:

- 这个属性只有在函数执行时才有效
- 它有一个 length 属性， 可以用来获得形参的个数， 因此可以用来比较形参和实参个数是否一致， 即比较 arguments.length 是否等于 arguments.callee.length
- 它可以用来递归匿名函数。

```js
function a() {
  console.log(arguments.callee)
}
```

105 ajax 、axios 、fetch区别

jQuery ajax

```js
$.ajax({
   type: 'POST',
   url: url,
   data: data,
   dataType: dataType
   success: function () {},
   error: function () {}
});
```

优缺点：

- 本身是针对 MVC 的编程,不符合现在前端 MVVM 的浪潮
- 基于原生的 XHR 开发， XHR 本身的架构不清晰， 已经有了 fetch 的替代方案
- JQuery 整个项目太大， 单纯使用 ajax 却要引入整个 JQuery 非常的不合理（采取个性化打包的方案又不能享受CDN服务)

axios

```js
axios({
    method: 'post',
    url: '/user/12345',
    data: {
        firstName: 'Fred',
        lastName: 'Flintstone'
    }
})
.then(function (response) {
    console.log(response);
})
.catch(function (error) {
    console.log(error);
});
```

优缺点：

- 从浏览器中创建 XMLHttpRequest
- 从 node.js 发出 http 请求
- 支持 Promise API
- 拦截请求和响应
- 转换请求和响应数据
- 取消请求
- 自动转换 JSON 数据
- 客户端支持防止 CSRF/XSRF

fetch

```js
try {
  let response = await fetch(url);
  let data = response.json();
  console.log(data);
} catch(e) {
  console.log("Oops, error", e);
```

```
}
```

优缺点：

- fetcht 只对网络请求报错，对 400 ， 500 都当做成功的请求， 需要封装去处理
- fetch 默认不会带 cookie ， 需要添加配置项
- fetch 不支持 abort ，不支持超时控制，使用 setTimeout 及 Promise.reject 的实现的超时控制并不能阻止请求过程继续在后台运行， 造成了量的浪费
- fetch 没有办法原生监测请求的进度， 而XHR可以

四、jQuery

1 你觉得jQuery或zepto源码有哪些写的好的地方

- jquery源码封装在一个匿名函数的自执行环境中，有助于防止变量的全局污染，然后通过传入window对象参数， 可以使window对象作为局部变量使用， 好处是当jquery中访问window对象的时候，就不用将作用域链退回到顶层作用域了，从而可以更快的访问window对象 。同样，传入undefined参数， 可以缩短查找undefined时的作用域链

```js
(function( window, undefined ) {

        //用一个函数域包起来，就是所谓的沙箱

        //在这里边var定义的变量，属于这个函数域内的局部变量，避免污染全局

        //把当前沙箱需要的外部变量通过函数参数引入进来

        //只要保证参数对内提供的接口的一致性，你还可以随意替换传进来的这个参数

    window.jQuery = window.$ = jQuery;

})( window );
```

- jquery将一些原型属性和方法封装在了jquery.prototype中， 为了缩短名称， 又赋值给了jquery.fn， 这是很形象的写法
- 有一些数组或对象的方法经常能使用到，jQuery将其保存为局部变量以提高访问速度
- jquery实现的链式调用可以节约代码，所返回的都是同一个对象， 可以提高代码效率

2 jQuery 的实现原理

```
(function(window, undefined) {})(window);
```

- jQuery 利用 JS 函数作用域的特性， 采用立即调用表达式包裹了自身，解决命名空间和变量污染问题
- window.jQuery = window.$ = jQuery;
- 在闭包当中将 jQuery 和 $ 绑定到 window 上， 从而将 jQuery 和 $ 暴露为全局变量

3 jQuery.fn 的 init 方法返回的 this 指的是什么对象

- jQuery.fn 的 init 方法 返回的 this 就是 jQuery 对象
- 用户使用 jQuery() 或 $() 即可初始化 jQuery 对象，不需要动态的去调用 init 方法

4 jQuery.extend 与 jQuery.fn.extend 的区别

- $.fn.extend() 和 $.extend() 是 jQuery 为扩展插件提拱了两个方法
- $.extend(object) ; // 为jQuery添加“静态方法”（工具方法)

```js
$.extend({
   min: function(a, b) { return a < b ? a : b; },
   max: function(a, b) { return a > b ? a : b; }
});
$.min(2,3); //  2
$.max(4,5); //  5
```

- $.extend([true,] targetObject, object1[, object2]); // 对targt对象进行扩展

```js
var settings = {validate:false, limit:5};
var options = {validate:true, name:"bar"};
$.extend(settings, options);  // 注意：不支持第一个参数传 false
// settings == {validate:true, limit:5, name:"bar"}
```

- $.fn.extend(json) ; // 为jQuery添加“成员函数”（实例方法)

```js
$.fn.extend({
   alertValue: function() {
      $(this).click(function(){
        alert($(this).val());
      });
```

```
    }
});

$("#email").alertValue();
```

5 jQuery 的属性拷贝(extend)的实现原理是什么， 如何实现深拷贝

- 浅拷贝（只复制一份原始对象的引用） `var newObject = $.extend({}, oldObject);`
- 深拷贝（对原始对象属性所引用的对象进行进行递归拷贝） `var newObject = $.extend(true, {}, oldObject);`

6 jQuery 的队列是如何实现的

- jQuery 核心中有一组队列控制方法， 由 `queue()/dequeue()/clearQueue()` 三个方法组成。
- 主要应用于 `animate()` ， `ajax` ， 其他要按时间顺序执行的事件中

```
var func1 = function(){alert( '事件1');}
var func2 = function(){alert( '事件2');}
var func3 = function(){alert( '事件3');}
var func4 = function(){alert( '事件4');}

// 入栈队列事件
$( '#box').queue("queue1", func1);  // push func1 to queue1
$( '#box').queue("queue1", func2);  // push func2 to queue1

// 替换队列事件
$( '#box').queue("queue1", []);  // delete queue1 with empty array
$( '#box').queue("queue1", [func3, func4]);  // replace queue1

// 获取队列事件（返回一个函数数组)
$( '#box').queue("queue1");  // [func3(), func4()]

// 出栈队列事件并执行
$( '#box').dequeue("queue1"); // return func3 and do func3
$( '#box').dequeue("queue1"); // return func4 and do func4

// 清空整个队列
$( '#box').clearQueue("queue1"); // delete queue1 with clearQueue
```

7 jQuery 中的 bind(), live(), delegate(), on()的区别

- `bind()` 直接绑定在目标元素上
- `live()` 通过冒泡传播事件， 默认 `document` 上， 支持动态数据
- `delegate()` 更精确的小范围使用事件代理，性能优于 live
- `on()` 是最新的 `1.9` 版本整合了之前的三种方式的新事件绑定机制

8 是否知道自定义事件

- 事件即“发布/订阅”模式， 自定义事件即“消息发布”，事件的监听即“订阅订阅”
- JS 原生支持自定义事件， 示例：

```js
document.createEvent(type); // 创建事件
event.initEvent(eventType, canBubble, prevent); // 初始化事件
target.addEventListener( 'dataavailable', handler, false); // 监听事件
target.dispatchEvent(e);  // 触发事件
```

- jQuery 里的 `fire` 函数用于调用 `jQuery` 自定义事件列表中的事件

9 jQuery 通过哪个方法和 Sizzle 选择器结合的

- `Sizzle` 选择器采取 `Right To Left` 的匹配模式， 先搜寻所有匹配标签， 再判断它的父节点
- `jQuery` 通过 `$(selecter).find(selecter);` 和 `Sizzle` 选择器结合

10jQuery 中如何将数组转化为 JSON 字符串，然后再转化回来

```js
// 通过原生 JSON.stringify/JSON.parse 扩展 jQuery 实现
$.array2json = function(array) {
    return JSON.stringify(array);
}

$.json2array = function(array) {
    // $.parseJSON(array); // 3.0 开始， 已过时
    return JSON.parse(array);
}
```

```
// 调用
var json = $.array2json( [ 'a', 'b', 'c']);
var array = $.json2array(json);
```

11jQuery 一个对象可以同时绑定多个事件， 这是如何实现的

```
$("#btn").on("mouseover mouseout", func);

$("#btn").on({
    mouseover: func1,
    mouseout: func2,
    click: func3
});
```

12 针对 jQuery 的优化方法

- 缓存频繁操作 DOM 对象
- 尽量使用 id 选择器代替 class 选择器
- 总是从 #id 选择器来继承
- 尽量使用链式操作
- 使用时间委托 on 绑定事件
- 采用 jQuery 的内部函数 data() 来存储数据
- 使用最新版本的 jQuery

13jQuery 的 slideUp 动画， 当鼠标快速连续触发, 动画会滞后反复执行，该如何处理呢

- 在触发元素上的事件设置为延迟处理：使用 JS 原生 setTimeout 方法
- 在触发元素的事件时预先停止所有的动画， 再执行相应的动画事件：

```
$('.tab').stop().slideUp();
```

14jQuery UI 如何自定义组件

- 通过向 $.widget() 传递组件名称和一个原型对象来完成
-

```
$.widget("ns.widgetName", [baseWidget], widgetPrototype);
```

15jQuery 与 jQuery UI 、jQuery Mobile 区别

- jQuery 是 JS 库，兼容各种PC浏览器， 主要用作更方便地处理 DOM 、事件 、动画、AJAX
- jQuery UI 是建立在 jQuery 库上的一组用户界面交互 、特效 、小部件及主题
- jQuery Mobile 以 jQuery 为基础，用于创建 “移动Web应用” 的框架

16jQuery 和 Zepto 的区别？ 各自的使用场景

- jQuery 主要目标是 PC 的网页中，兼容全部主流浏览器 。在移动设备方面， 单独推出 、jQuery Mobile
- Zepto 从一开始就定 位移动设备，相对更轻量级 。它的 API 基本兼容 jQuery、，但对PC浏览器兼容不理想

17jQuery对象的特点

- 只有 JQuery 对象才能使用 JQuery 方法
- JQuery 对象是一个数组对象

五、Bootstrap

1 什么是Bootstrap？ 以及为什么要使用Bootstrap?

> Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架。
> Bootstrap 是基于 HTML 、 CSS 、 JAVASCRIPT 的

- Bootstrap 具有移动设备优先 、浏览器支持良好 、容易上手 、响应式设计等优点，所以 Bootstrap 被广泛应用

2 使用Bootstrap时，要声明的文档类型是什么？ 以及为什么要这样声明?

- 使用 Bootstrap 时， 需要使用 HTML5 文档类型 （ Doctype ）。 <!DOCTYPE html>
- 因为 Bootstrap 使用了一些 HTML5 元素和 CSS 属性， 如果在 Bootstrap 创建的网页开头不使用 HTML5 的文档类型 （ Doctype ）， 可能会面临一些浏览器显示不一致的问题， 甚至可能面临一些特定情境下的不一致， 以致于代码不能通过 W3C 标准的验证

3 什么是Bootstrap网格系统

> Bootstrap 包含了一个响应式的、移动设备优先的、不固定的网格系统，可以随着设备或视口大小的增加而适当地扩展到 12 列。它包含了用于简单的布局选项的预定义类，也包含了用于生成更多语义布局的功能强大的混合类

- 响应式网格系统随着屏幕或视口（viewport）尺寸的增加，系统会自动分为最多 12 列。

4 Bootstrap 网格系统（Grid System）的工作原理

- （1）行必须放置在 .container class 内，以便获得适当的对齐（alignment）和内边距（padding）。
- （2）使用行来创建列的水平组。
- （3）内容应该放置在列内，且唯有列可以是行的直接子元素。
- （4）预定义的网格类，比如 .row 和 .col-xs-4，可用于快速创建网格布局。LESS 混合类可用于更多语义布局。
- （5）列通过内边距（padding）来创建列内容之间的间隙。该内边距是通过 .rows 上的外边距（margin）取负，表示第一列和最后一列的行偏移。
- （6）网格系统是通过指定您想要横跨的十二个可用的列来创建的。例如，要创建三个相等的列，则使用三个 .col-xs-4

5 对于各类尺寸的设备，Bootstrap设置的class前缀分别是什么

- 超小设备手机（<768px）：.col-xs-*
- 小型设备平板电脑（>=768px）：.col-sm-*
- 中型设备台式电脑（>=992px）：.col-md-*
- 大型设备台式电脑（>=1200px）：.col-lg-*

6 Bootstrap 网格系统列与列之间的间隙宽度是多少

> 间隙宽度为 30px（一个列的每边分别是 15px）

7 如果需要在一个标题的旁边创建副标题， 可以怎样操作

> 在元素两旁添加 `<small>` ， 或者添加 `.small` 的 `class`

8 用Bootstrap， 如何设置文字的对齐方式?

- `class="text-center"` 设置居中文本
- `class="text-right"` 设置向右对齐文本
- `class="text-left"` 设置向左对齐文本

9 Bootstrap如何设置响应式表格?

> 增加 `class="table-responsive"`

10 使用Bootstrap创建垂直表单的基本步骤?

- （1）向父 `<form>` 元素添加 `role="form"` ；
- （2）把标签和控件放在一个带有 `class="form-group"` 的 `<div>` 中， 这是获取最佳间距所必需的;
- （3）向所有的文本元素 `<input>` 、 `<textarea>` 、 `<select>` 添加 `class="form-control"`

11 使用Bootstrap创建水平表单的基本步骤?

- （1）向父 `<form>` 元素添加 `class="form-horizontal"` ；
- （2）把标签和控件放在一个带有 `class="form-group"` 的 `<div>` 中;
- （3）向标签添加 `class="control-label"` 。

12 使用Bootstrap如何创建表单控件的帮助文本?

> 增加 class="help-block" 的 span 标签或 p 标签。

13 使用Bootstrap激活或禁用按钮要如何操作?

- 激活按钮：给按钮增加 .active 的 class
- 禁用按钮：给按钮增加 disabled="disabled" 的属性

14 Bootstrap有哪些关于的class?

- （1） .img-rounded 为图片添加圆角
- （2） .img-circle 将图片变为圆形
- （3） .img-thumbnail 缩略图功能
- （4） .img-responsive 图片响应式 (将很好地扩展到父元素)

15 Bootstrap中有关元素浮动及清除浮动的class?

- （1） class="pull-left" 元素浮动到左边
- （2） class="pull-right" 元素浮动到右边
- （3） class="clearfix" 清除浮动

16 除了屏幕阅读器外， 其他设备上隐藏元素的class?

> 、class="sr-only"、、

17 Bootstrap如何制作下拉菜单?

- （1）将下拉菜单包裹在 class="dropdown" 的 中;
- （2）在触发下拉菜单的按钮中添加： class="btn dropdown-toggle" id="dropdownMenu1" data-toggle="dropdown"
- （3）在包裹下拉菜单的ul中添加： class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1"
- （4）在下拉菜单的列表项中添加： role="presentation" 。其中，下拉菜单的标题要添加 class="dropdown-header" ， 选项部分要添加 tabindex="-1" 。

18 Bootstrap如何制作按钮组？ 以及水平按钮组和垂直按钮组的优先级？

- （1）用 class="btn-group" 的 去包裹按钮组； class="btn-group-vertical" 可设置垂直按钮组。
- （2） btn-group 的优先级高于 btn-group-vertical 的优先级。

19 Bootstrap如何设置按钮的下拉菜单？

> 在一个 .btn-group 中放置按钮和下拉菜单即可。

20 Bootstrap中的输入框组如何制作？

- （1）把前缀或者后缀元素放在一个带有 class="input-group" 中的 中
- （2）在该 内，在 class="input-group-addon" 的 里面放置额外的内容；
- （3）把 放在 <input> 元素的前面或后面。

21 Bootstrap中的导航都有哪些？

- （1）导航元素：有 class="nav nav-tabs" 的标签页导航， 还有 class="nav nav-pills" 的胶囊式标签页导航；
- （2）导航栏： class="navbar navbar-default" role="navigation" ；
- （3）面包屑导航： class="breadcrumb"

22 Bootstrap中设置分页的class？

- 默认的分页： class="pagination"
- 默认的翻页： class="pager"

23 Bootstrap中显示标签的class？

> class="label"

24 Bootstrap中如何制作徽章?

```
<span class="badge">26</span>
```

25 Bootstrap中超大屏幕的作用是什么?

> 设置 class="jumbotron" 可以制作超大屏幕，该组件可以增加标题的大小并增加更多的外边距

六、微信小程序

1 微信小程序有几个文件

- WXSS (WeiXin Style Sheets) 是一套样式语言，用于描述 WXML 的组件样式， js 逻辑处理， 网络请求 json 小程序设置， 如页面注册， 页面标题及 tabBar 。
- app.json 必须要有这个文件， 如果没有这个文件，项目无法运行， 因为微信框架把这个作为配置文件入口，整个小程序的全局配置 。包括页面注册， 网络设置， 以及小程序的 window 背景色， 配置导航条样式， 配置默认标题。
- app.js 必须要有这个文件，没有也是会报错！但是这个文件创建一下就行 什么都不需要写以后我们可以在这个文件中监听并处理小程序的生命周期函数 、声明全局变量。
- app.wxss 配置全局 css

2 微信小程序怎样跟事件传值

> 给 HTML 元素添加 data-* 属性来传递我们需要的值，然后通过 e.currentTarget.dataset 或 onload 的 param 参数获取 。但 data - 名称不能有大写字母和不可以存放对象

3 小程序的 wxss 和 css 有哪些不一样的地方?

- wxss 的图片引入需使用外链地址
- 没有 Body ； 样式可直接使用 import 导入

4 小程序关联微信公众号如何确定用户的唯一性

> 使用 wx.getUserInfo 方法 withCredentials 为 true 时 可获取 encryptedData ， 里面有 union_id 。后端需要进行对称解密

5 微信小程序与vue区别

- 生命周期不一样，微信小程序生命周期比较简单
- 数据绑定也不同，微信小程序数据绑定需要使用 {{}} ， vue 直接 : 就可以
- 显示与隐藏元素， vue 中，使用 v-if 和 v-show 控制元素的显示和隐藏，小程序中，使用 wx-if 和 hidden 控制元素的显示和隐藏
- 事件处理不同，小程序中，全用 bindtap(bind+event) ， 或者 catchtap(catch+event) 绑定事件, vue: 使用 v-on:event 绑定事件， 或者使用 @event 绑定事件
- 数据双向绑定也不也不一样在 vue 中,只需要再表单元素上加上 v-model ,然后再绑定 data 中对应的一个值， 当表单元素内容发生变化时， data 中对应的值也会相应改变，这是 vue 非常 nice 的一点 。微信小程序必须获取到表单元素， 改变的值，然后再把值赋给一个 data 中声明的变量。

七、webpack相关

1 打包体积 优化思路

- 提取第三方库或通过引用外部文件的方式引入第三方库
- 代码压缩插件 UglifyJsPlugin
- 服务器启用gzip压缩
- 按需加载资源文件 require.ensure
- 优化 devtool 中的 source-map
- 剥离 css 文件， 单独打包
- 去除不必要插件， 通常就是开发环境与生产环境用同一套配置文件导致

2 打包效率

- 开发环境采用增量构建，启用热更新

- 开发环境不做无意义的工作如提取 css 计算文件hash等
- 配置 devtool
- 选择合适的 loader
  - 个别 loader 开启 cache 如 babel-loader
  - 第三方库采用引入方式
  - 提取公共代码
- 优化构建时的搜索路径 指明需要构建目录及不需要构建目录
- 模块化引入需要的部分

3 Loader

编写一个loader

> loader 就是一个 node 模块， 它输出了一个函数 。当某种资源需要用这个 loader 转换时， 这个函数会被调用 。并且， 这个函数可以通过提供给它的 this 上下文访问 Loader API 。 reverse-txt-loader

```js
// 定义
module.exports = function(src) {
  //src是原文件内容（abcde)，下面对内容进行处理， 这里是反转
  var result = src.split('').reverse().join('');
  //返回JavaScript源码，必须是String或者Buffer
  return `module.exports = '${result}'`;
}
//使用
{
    test: /\.txt$/,
    use: [
        {
            './path/reverse-txt-loader'
        }
    ]
},
```

4 说一下webpack的一些plugin， 怎么使用webpack对项目进行优化

构建优化

- 减少编译体积 ContextReplacementPugin 、 IgnorePlugin 、 babel-plugin-import 、 babel-plugin-transform-runtime

- 并行编译 happypack 、 thread-loader 、 uglifyjsWebpackPlugin 开启并行
- 缓存 cache-loader 、 hard-source-webpack-plugin 、 uglifyjsWebpackPlugin 开启缓存 、 babel-loader 开启缓存
- 预编译 dllWebpackPlugin && DllReferencePlugin 、 auto-dll-webapck-plugin

性能优化

- 减少编译体积 Tree-shaking 、 Scope Hositing
- hash 缓存 webpack-md5-plugin
- 拆包 splitChunksPlugin 、 import() 、 require.ensure

八、编程题

1 写一个通用的事件侦听器函数

```js
// event(事件)工具集，来源：github.com/markyun
markyun.Event = {

    // 视能力分别使用dom0| |dom2 | |IE方式 来绑定事件
    // 参数:  操作的元素 ,事件名称 ,事件处理程序
    addEvent : function(element, type, handler) {
        if (element.addEventListener) {
            //事件类型、需要执行的函数、是否捕捉
            element.addEventListener(type, handler, false);
        } else if (element.attachEvent) {
            element.attachEvent( 'on' + type, function() {
                handler.call(element);
            });
        } else {
            element [ 'on' + type] = handler;
        }
    },
    // 移除事件
    removeEvent : function(element, type, handler) {
        if (element.removeEventListener) {
            element.removeEventListener(type, handler, false);
        } else if (element.datachEvent) {
            element.detachEvent( 'on' + type, handler);
        } else {
            element [ 'on' + type] = null;
        }
    },
    // 阻止事件 (主要是事件冒泡， 因为IE不支持事件捕获)
    stopPropagation : function(ev) {
```

```
            if (ev.stopPropagation) {
                ev.stopPropagation();
            } else {
                ev.cancelBubble = true;
            }
        },
        // 取消事件的默认行为
        preventDefault : function(event) {
            if (event.preventDefault) {
                event.preventDefault();
            } else {
                event.returnValue = false;
            }
        },
        // 获取事件目标
        getTarget : function(event) {
            return event.target || event.srcElement;
        }
```

2 如何判断一个对象是否为数组

```js
function isArray(arg) {
    if (typeof arg === 'object') {
        return Object.prototype.toString.call(arg) === '[object Array]';
    }
    return false;
}
```

3 冒泡排序

- 每次比较相邻的两个数， 如果后一个比前一个小，换位置

```js
var arr = [3, 1, 4, 6, 5, 7, 2];

function bubbleSort(arr) {
for (var i = 0; i < arr.length - 1; i++) {
    for(var j = 0; j < arr.length - i - 1; j++) {
        if(arr[j + 1] < arr[j]) {
            var temp;
            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
```