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

```
(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）

```
$.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

```
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）

```
$.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：

```
  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

10 jQuery 中如何将数组转化为 JSON 字符串，然后再转化回来

```
// 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);
```

11 jQuery 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`

13 jQuery 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();`

14 jQuery 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);`

15 jQuery 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

16 jQuery 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

17 jQuery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