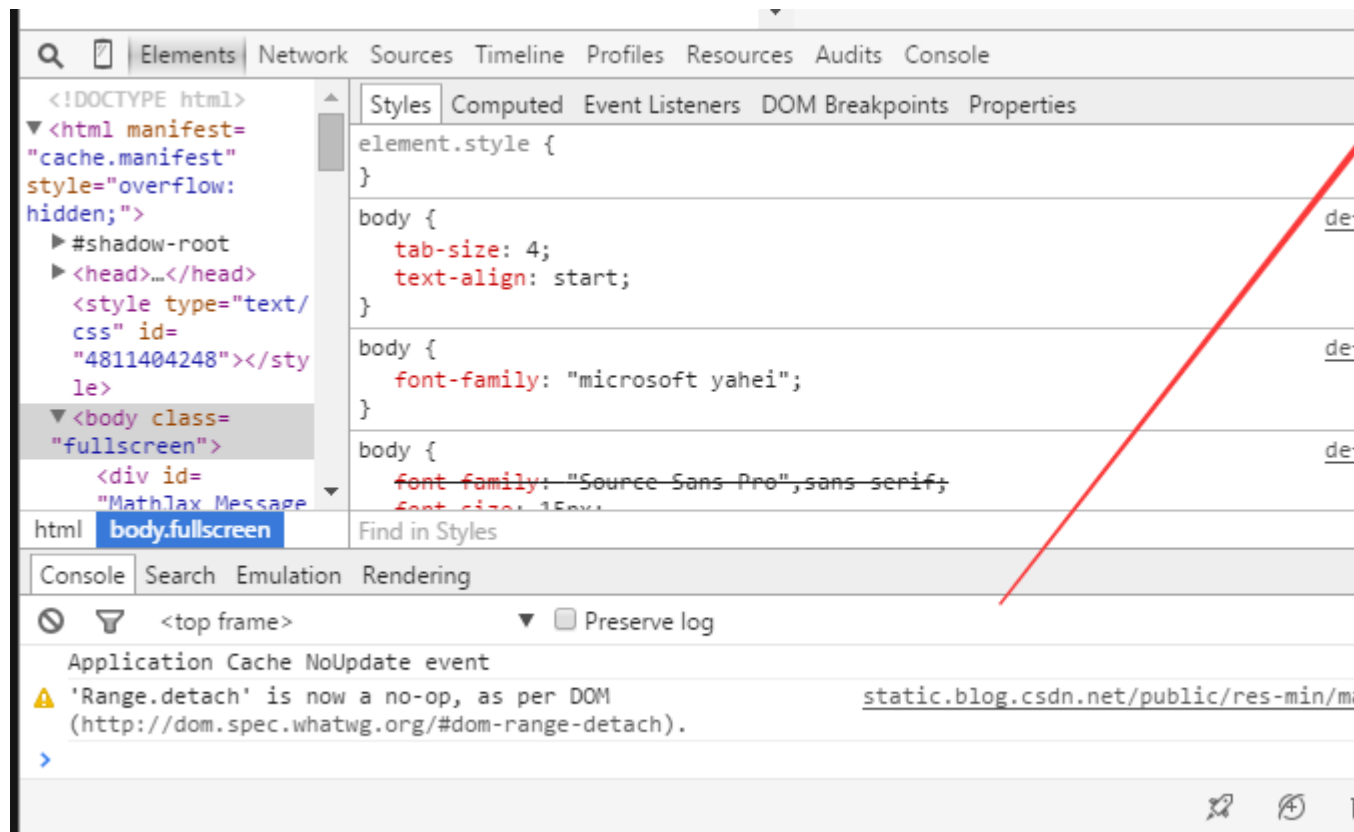


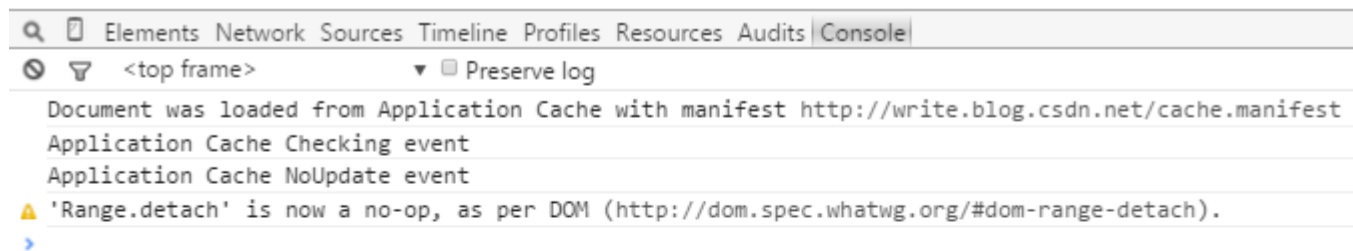
Chrome 调试

大多浏览器的调试功能的启用快捷键都一致...按下 F12;还是熟悉的味道;



或者直接

Ctrl> + Shift + J: 直接进入 console 面板



点击第一张截图圈圈那个进入,看到 Shortcuts 这个就是了....快捷键大全

快捷键大全

快捷键有这么以下几大类..且看我一一道来~~~注:All pane 是全局快捷键

Console()

- Ctrl + L : 清除控制台消息
- Tab : 自动完成通用常缀
- →: 接受建议
- Ctrl + U: 清除 console 的提示!!(实操查看源代码!!)
- ↓ / ↑ :选中下一行/上一行
- Enter: 执行代码或者命令

Debugger(调试面板)

- F8 or Ctrl + \: 暂停/继续
- F10 or Ctrl + ': 单步执行
- F11 or Ctrl + ;: 单步进入

- `Shift + F11` or `Ctrl + Shift + ;`: 单步退出
- `Ctrl + . / Ctrl + ,`: 上一帧/下一帧(译为框架怪怪的)
- `Ctrl + Shift + E`: 被选中代码在控制台中打印出 `console` 信息(非常实用)
- `Ctrl + Shift + A`: 添加到 `debugger` 的 `watch` 里面,可以关注你选中内容的变化
- `Ctrl + B`: 打断点/取消断点(很实用)

Text Editor(文本编辑器)

- `Ctrl + Shift + P`: 跳转到某个成员(不知鸟用)
- `Ctrl + Space`: 自动完成
- `Ctrl + G`: 跳转到某行
- `Ctrl + Shift + E`
- `Alt + -`: 跳转到之前的编辑位置
- `Alt + +`: 跳转到下一个编辑的位置
- `Ctrl + /`: 快捷键注释
- `Alt + ↑`: 调整 CSS 度量单位,每次增加一个单元
- `Alt + ↓`: 调整 CSS 度量单位,每次减少一个单元
- `Alt + PageUp`: 调整 CSS 度量单位,每次**增加**10个单元
- `Alt + PageDown`: 调整 CSS 度量单位,每次**减少**10个单元
- `Ctrl + D`: 选择选中内容的下一个匹配内容
- `Ctrl + U`: 软撤销
- `Ctrl + M`: 进入匹配的括号
- `Alt + W`: 关闭编辑便签
- `Alt + O`: 切换相同名字的或者不同后缀的文件

All Panels(所有面板)

- `Ctrl + [/ Ctrl +]`: 切换面板(向左向右)
- `Ctrl + Shift + [/ Ctrl + Shift +]`: 返回之前之后的面板状态
- `Ctrl + Tilde[~]`: 显示 Console
- `Esc`: 小菜单弹出隐藏
- `Ctrl + Shift + M`: 进入仿真设备模式(移动平板屏幕)
- `Ctrl + Shift + D`: 切换调试面板在底部还是侧边栏展示
- `Ctrl + F`: 搜索内容
- `Ctrl + Shift + F`: 在所有代码中搜索(跨域),很实用

- `Ctrl + Shift + C`: 选择页面节点并且查看代码,最常用!!
- `Ctrl + P`: 快速切换源码文件,很常用!

Style Pane(风格面板)

- `Tab / Shift + Tab`: 下一个/上一个属性
- `↑`: 增加 CSS 度量单位->1 单元
- `↓`:减少 CSS 度量单位->1 单元
- `PageUp or Shift + ↑`:增加 CSS 度量单位->10 单元
- `PageDown or Shift + ↓`:减少 CSS 度量单位->10 单元
- `Shift + PageUp`:增加 CSS 度量单位->100 单元
- `Shift + PageDown`:减少 CSS 度量单位->100 单元
- `Alt + ↑`:增加 CSS 度量单位->0.1 单元
- `Alt + ↓`:减少 CSS 度量单位->0.1 单元

Timeline Panel(时间轴面板)

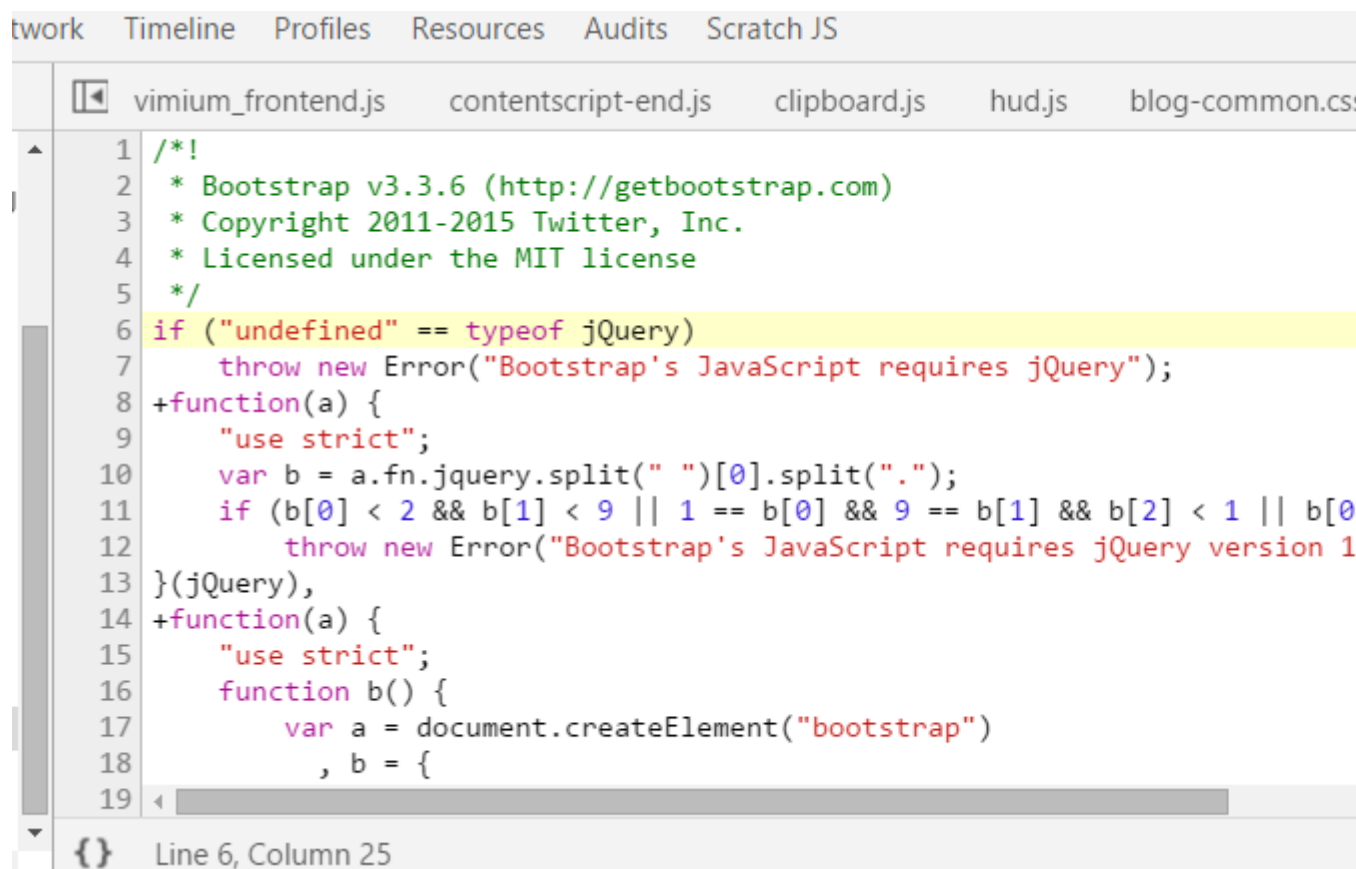
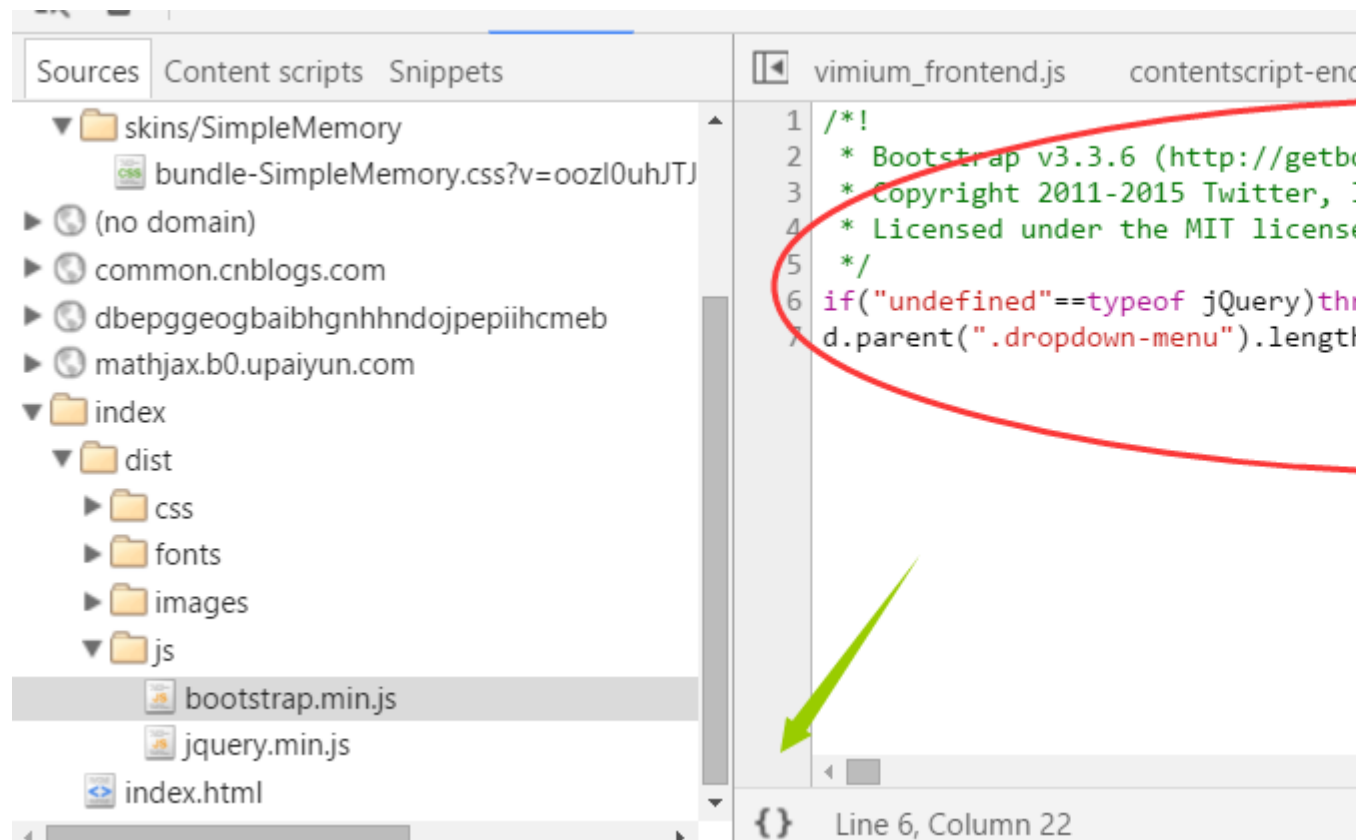
- `Ctrl + E`:开始
- `Ctrl + S`:保存时间轴数据
- `Ctrl + O`:加载时间轴数据

总结

Chrome 的 debugger 和 style pane 的快捷键和 Firefox 下的 firebug 大同小异; 就是 chrome 的丰富些; 没必要所有都去记忆,常用的就那么八九个;其他的鼠标点点就好了...当然,能记下来最好, 毕竟快捷键还是比鼠标高效的

JS 调试技巧

技巧一： 格式化压缩代码



技巧二：快速跳转到某个断点的位置

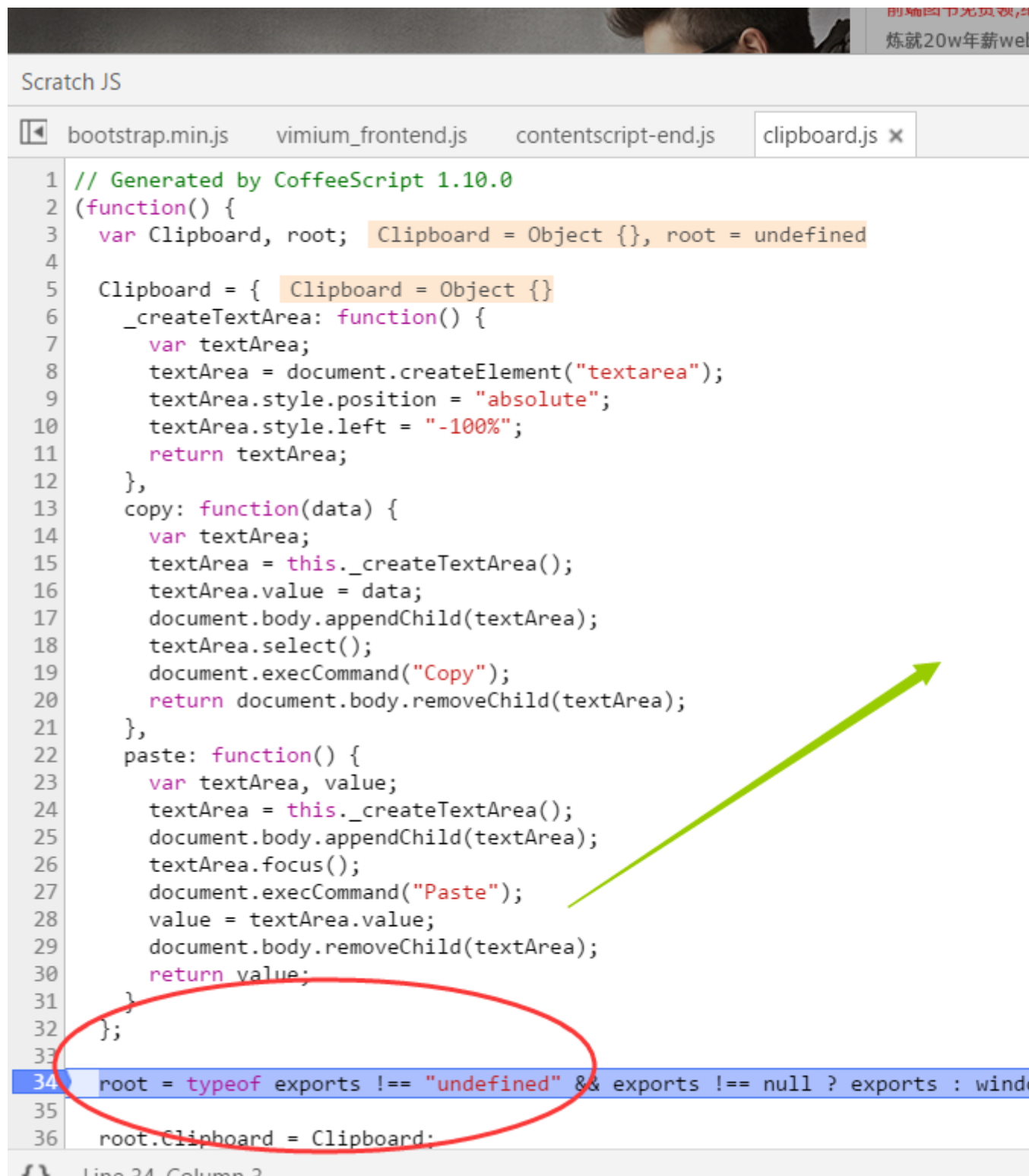
右侧的 **Breakpoints** 会汇总你在 JS 文件所有打过的断点，点击跟 checkbox 同一行的会暂时取消这个断点，若是点击 checkbox 下一行的会直接跳转到该断点的位置

```
12         if (e === "none" || e === "") {
13             cl || (cl = c.createElement("iframe"),
14             cl.frameBorder = cl.width = cl.height = 0),
15             b.appendChild(cl);
16         if (!cm || !cl.createElement)
17             cm = (cl.contentWindow || cl.contentDocument).document,
18             cm.write((c.compatMode === "CSS1Compat" ? "<!doctype html"
19             cm.close();
20         d = cm.createElement(a),
21         cm.body.appendChild(d),
22         e = f.css(d, "display"),
23         b.removeChild(cl)
24     }
25     ck[a] = e
26 }
27 return ck[a]
28 }
29 function cu(a, b) {
30     var c = {};
31     f.each(cq.concat.apply([], cq.slice(0, b)), function() {
32         c[this] = a
33     });
34     return c
35 }
36 function ct() {
37     cr = b
38 }
39 function cs() {
40     setTimeout(ct, 0);
41     return cr = f.now()
42 }
43 function cj() {
44     try {
45         return new a.ActiveXObject("Microsoft.XMLHTTP")
46     } catch (b) {}
47 }
```

Line 25, Column 1

技巧三：查看断点内部的作用范围
【很实用】

右侧的 **scope** 可以看到相当多实用的信息，比如 **this** 的指向，是否有值，断点
是对象还是其他等。。



```
Scratch JS
bootstrap.min.js  vimium_frontend.js  contentscript-end.js  clipboard.js x

1 // Generated by CoffeeScript 1.10.0
2 (function() {
3   var Clipboard, root; Clipboard = Object {}, root = undefined
4
5   Clipboard = { Clipboard = Object {}
6     _createTextArea: function() {
7       var textArea;
8       textArea = document.createElement("textare");
9       textArea.style.position = "absolute";
10      textArea.style.left = "-100%";
11      return textArea;
12    },
13    copy: function(data) {
14      var textArea;
15      textArea = this._createTextArea();
16      textArea.value = data;
17      document.body.appendChild(textArea);
18      textArea.select();
19      document.execCommand("Copy");
20      return document.body.removeChild(textArea);
21    },
22    paste: function() {
23      var textArea, value;
24      textArea = this._createTextArea();
25      document.body.appendChild(textArea);
26      textArea.focus();
27      document.execCommand("Paste");
28      value = textArea.value;
29      document.body.removeChild(textArea);
30      return value;
31    }
32  };
33
34  root = typeof exports !== "undefined" && exports !== null ? exports : window
35
36  root.Clipboard = Clipboard;
```


技巧 4：监听事件断点

右侧的 **Event Listener Breakpoints** 可以选择性的监听某类行为事件，比如键盘输入，拖拉等。。勾选前面的 checkbox 就可以生效，当你触发改行为的时候就会跳转到触发的 JS

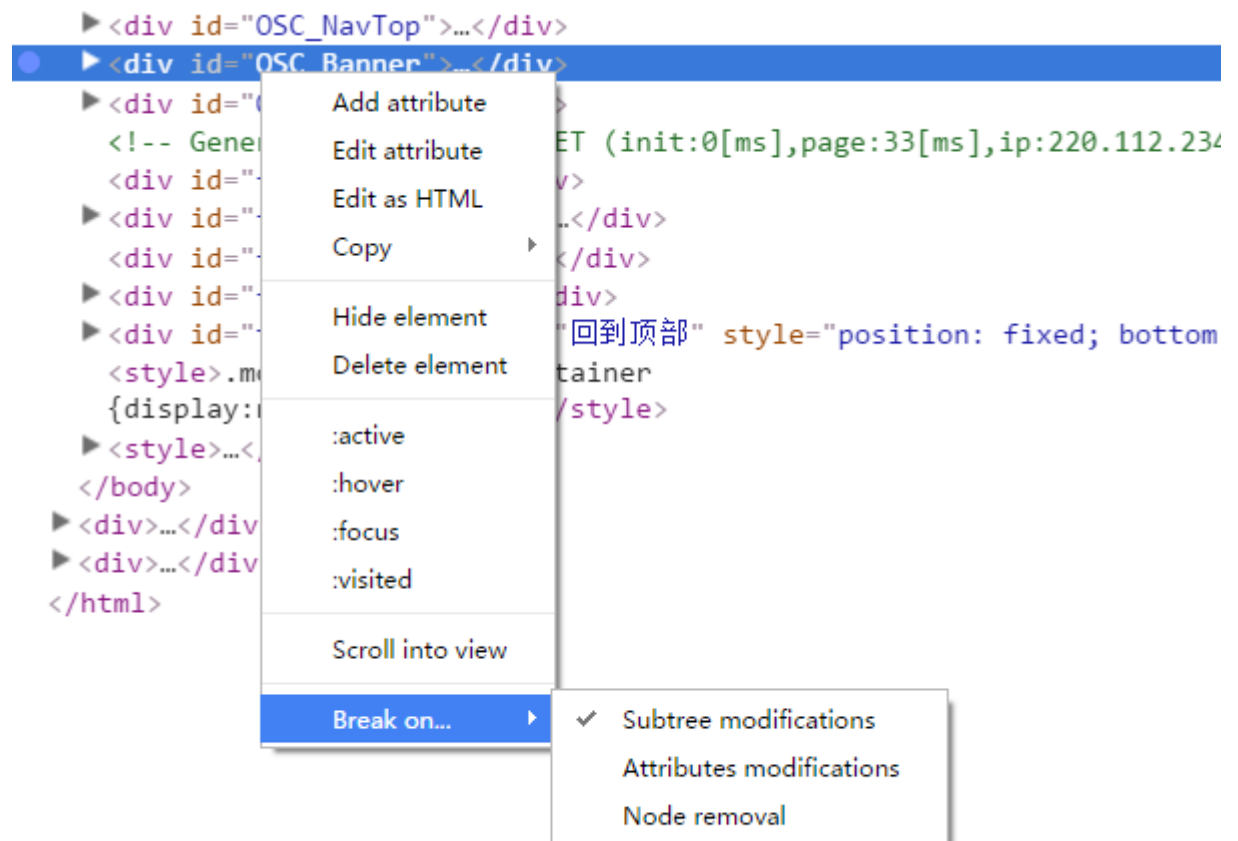
```
h JS
mium_frontend.js*  global-shortcut.js* x  fsContent.js  VM78 vimium_frontend.js  >>  [
//
// This file is part of ColorZilla
//
// Written by Alex Sirota (alex @ iosart.com)
//
// Copyright (c) iosart labs llc 2011, All Rights Reserved
//
function() {
  if (!document.body.hasAttribute('cz-shortcut-listen')) {
    document.body.setAttribute('cz-shortcut-listen', 'true');
    document.body.addEventListener('keydown', function(e) {
      var isMac = navigator.userAgent.toLowerCase().indexOf('mac') > -1;
      var keyCode = e.keyCode;
      if ((e.ctrlKey && e.altKey && !isMac ||
        e.metaKey && e.altKey && isMac) &&
        keyCode > 64 && keyCode < 91) {
        chrome.extension.sendRequest({op: 'hotkey-pressed', keyCode: ke
      }
    }, false);
  }
})();

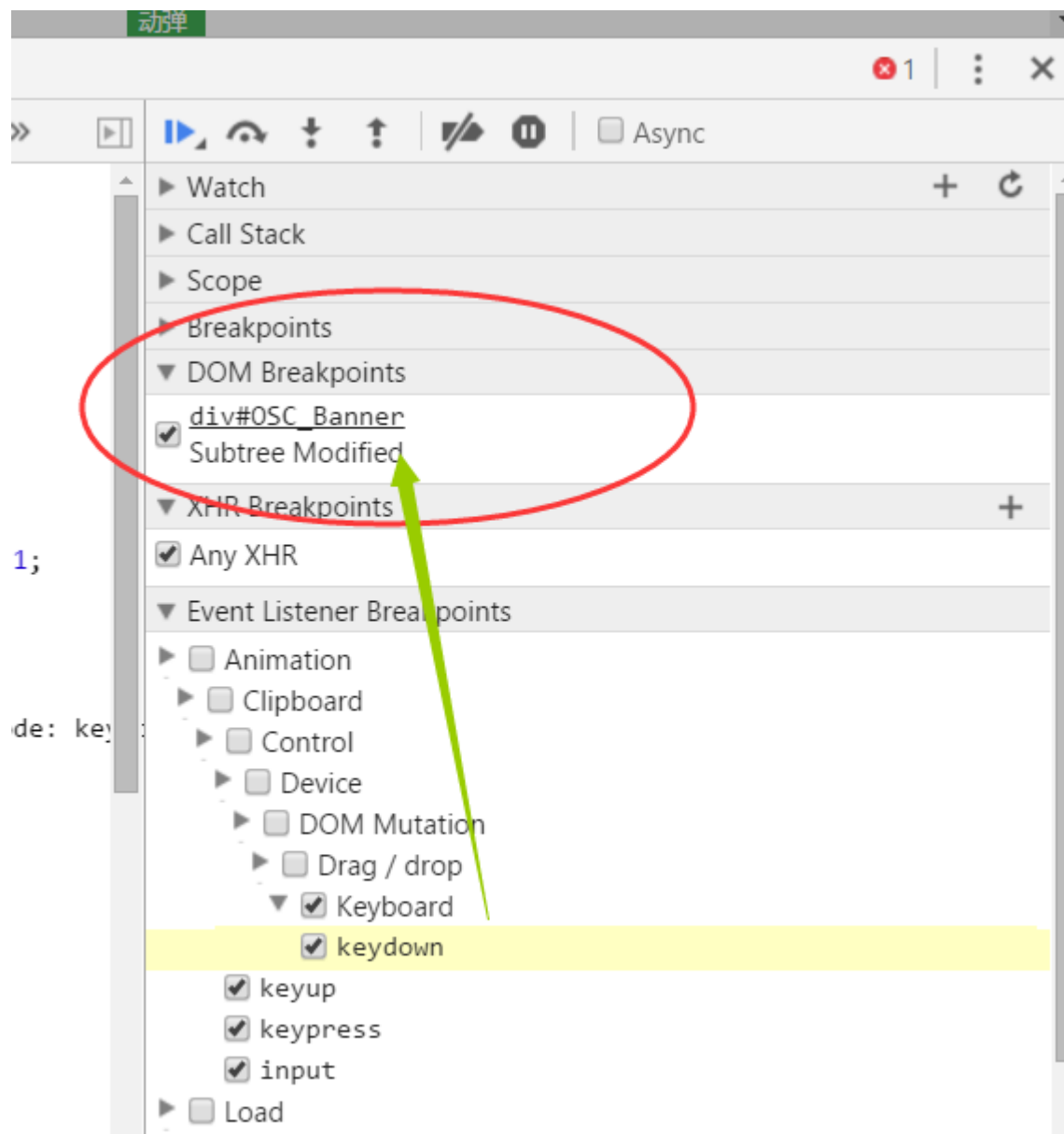
```

Line 11, Column 59

技巧 5：DOM 及 XHR 监听跳转

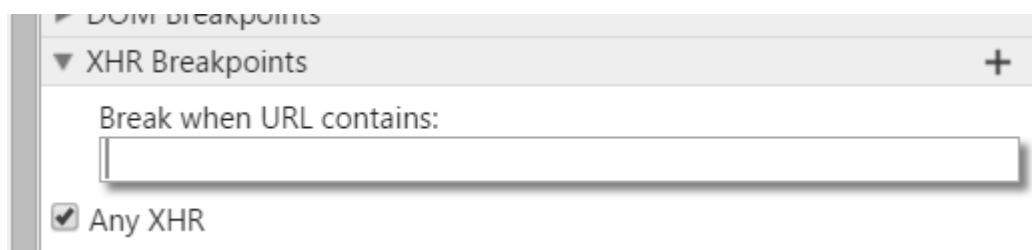
DOM Breakpoints：是你 Elements 页，感觉要监听某段 dom 的可能有一些行为，但是又不具体知道确切位置就可以用了





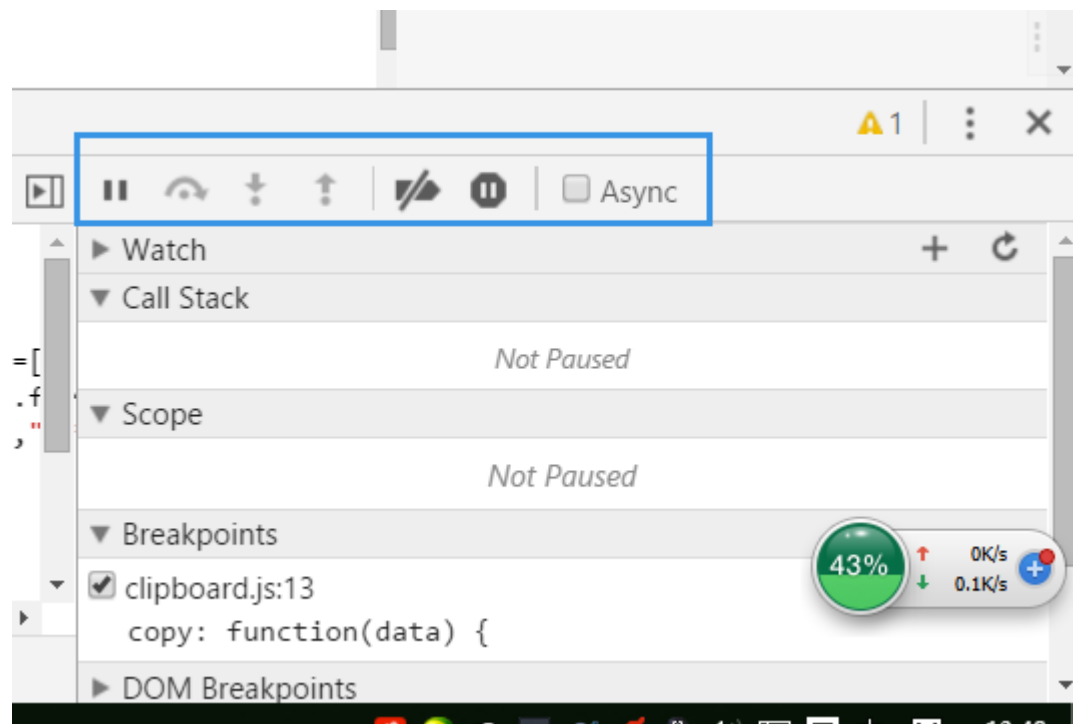
XHR Breakpoints: 向服务器请求的，ajax 的核心要点

默认勾选了，所有 XHR 行为，可选项是判断 URL。。。我用的不多



技巧 6：单步执行、单步进入、强制进入、单步退出

这个东东是调试中必不可少的，其实用过 firebug 的小伙伴，对这个就有一个清晰的认识你。基本一样；先上图；



功能名词依次，不懂的可以看看我 firebug 那个系列的

- **Pause script execution** 【单步执行，在断点处暂停，等待调试—不是直译】： 暂停后这个按钮会变成 **Resume script execution** 【继续执行】，快捷键 【F8 或者 Ctrl + \】
- **Step over next function call** 【单步跳过】： 会跳到下一个断点，快捷键 【F10 或者 Ctrl + `】
- **Step into next function call** 【单步进入】： 会进入函数内部调试，快捷键 【F11 或者 Ctrl + ;】
- **Step out of current function** 【单步跳出】： 会跳出当前这个断点的函数，快捷键 【Shift + F11 或者 Ctrl + Shift + ;】

后面的就是 chrome 的特色功能

- **Deactivate breakpoints** : 使所有断点临时失效，快捷键【Ctrl + F8】
- **Don't Pause on exceptions**: 不要在表达式处暂停，还有一个可选项【Pause On Caught Exceptions– 若抛出异常则需要暂停在那里】

前端调试技巧

1.debugger

代码中直接写debugger，代码在执行过程中会直接停在debugger处

```
PrintAction.prototype.doAction = function(param) { // 实现打印特定div的方法
  var rows = param.viewModel.dataTable.getSelectedRows();
  debugger
  if(rows.length > 0){
    var datas = [];
    rows.map(function(item,index){
      var data = param.viewModel.dataTable.getSelectedRows()[index].getSimpleData();
      datas.push(data);
    });
  }
};
```

2.以表格的形式查看复杂的对象，console.table()

The screenshot shows the Chrome DevTools Console with the following commands and outputs:

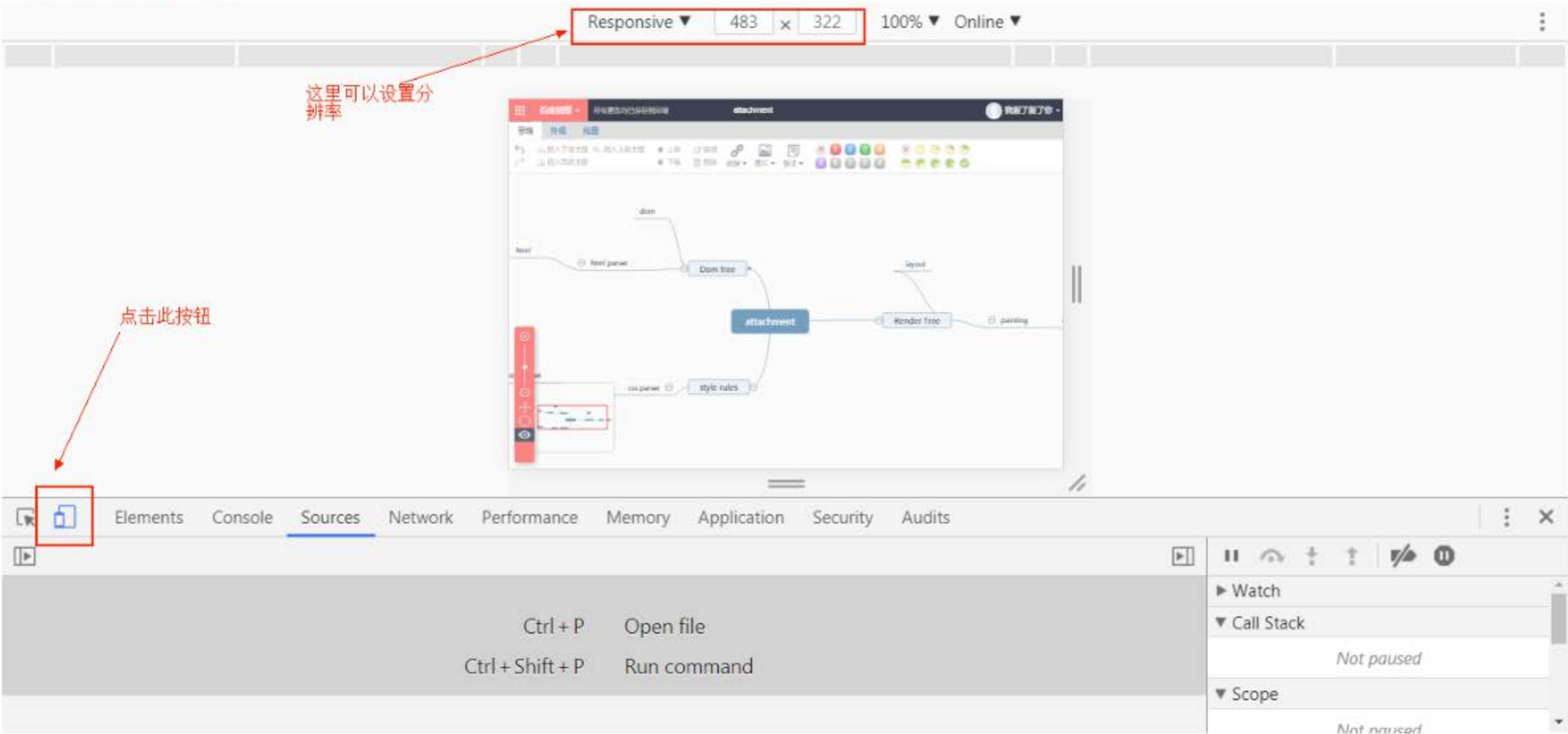
```
> var a = {item1:'a',item2:'b',item3:'c'}
< undefined
> console.log(a)
  {item1: "a", item2: "b", item3: "c"}
< undefined
> console.table(a)
```

The output of `console.table(a)` is displayed as a table with 2 columns: (index) and Value. The table contains 3 rows of data. A red box highlights the table output, and a red arrow points to it from the right.

(index)	Value
item1	"a"
item2	"b"
item3	"c"

Below the table, it says `Object`.

3.在pc端模拟调试移动设备

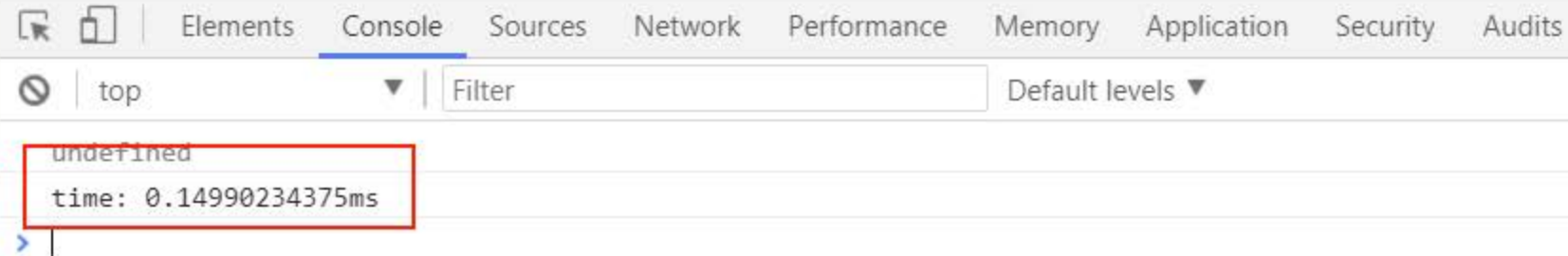


4.查看程序执行时间console.time()方法和console.timeEnd()

```
// uper(str)
// console.log(str)
console.time('time')
function a(){
  console.log(this.x)
}

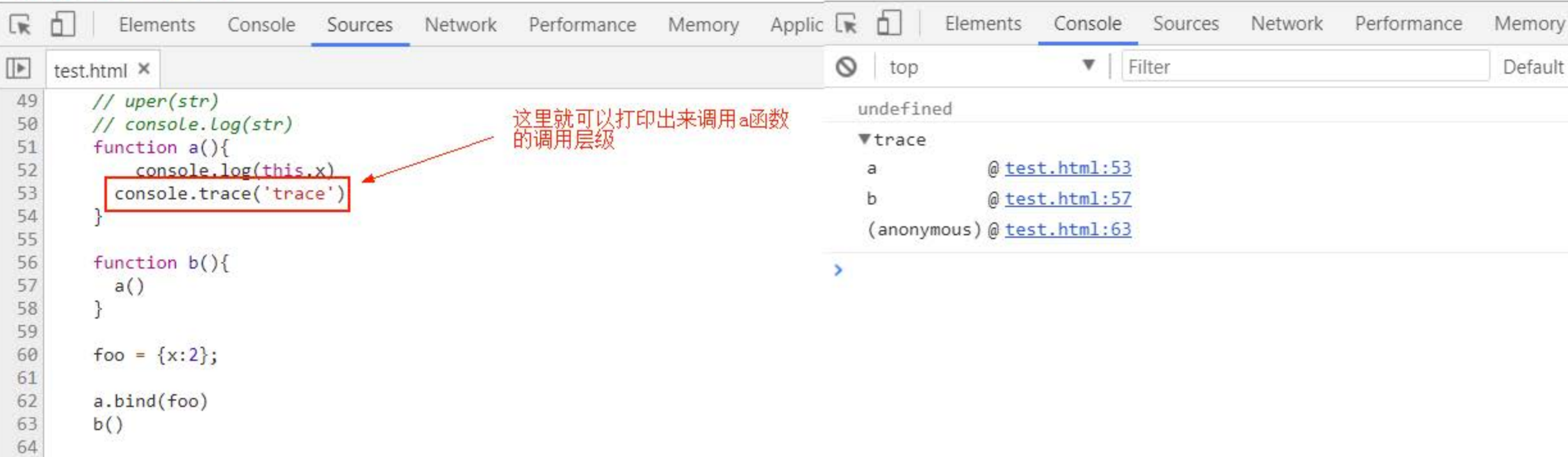
foo = {x:2};

a.bind(foo)
a()
console.timeEnd('time')
```



5.查看函数调用的轨迹

一种是使用console.trace命令:



另一种是在浏览器打断点：

Elements Console Sources Network Performance Memory Application Security Audits

jquery.min.js printAction.js x pubListPage.js

Serving from the file system? Add your files into the workspace. [more](#) [never show](#) x

```
22
23 PrintAction.prototype.doAction = function(param) { // 实现打印特定div的方法
24   console.time('time')
25   var rows = param.viewModel.dataTable.getSelectedRows();
26   if(rows.length > 0){
27     var datas = [];
28     rows.map(function(item,index){
29       var data = param.viewModel.dataTable.getSelectedRows()[index].getSimpleData();
30       datas.push(data);
31     });
32     szydinforeportcountform.setData(datas);
33     $('#nc-container').hide();
34     $('#alertStyle')?$('#alertStyle').css('display', 'none'):'';
35     $('#info-panel').css('display')=='block'?$('#info-panel').css('display','none'):null;
36     $('#szydCountDiv').css('display', 'block');
37     console.timeEnd('time')
38   }
```

CardUtils, RowEvents:
RowEvents, ...

当前程序运行位置

选择下一行就可以看到当前运行程序是在哪里调用的

Paused on breakpoint

Watch

Call Stack

- PrintAction.doAction printAction.js:25
- pubDoAction pubListPage.js:150
- (anonymous) VM1977:3
- (anonymous) knockout.debug.js:3714
- dispatch jquery.min.js:3
- q.handle jquery.min.js:3

Scope

Local

6.解压缩代码以便更好地调试 JavaScript , 比如jQuery.min.js文件

ElementsConsoleSourcesNetworkPerformanceMemoryApplicationSecurityAudits

jquery.min.js x

Serving from the file system? Add your files into the workspace.

more never show x

Pretty-print this minified file?

more never show x

1 /*! jQuery v3.2.1 | (c) JS Foundation and other contributors | jquery.org/license */

2 !function(a,b){"use strict";"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):f

3 a.removeEventListener("load",S),r.ready()}"complete"===d.readyState||"loading"!==d.readyState&&!d.documentElement.doScro

4 null==d?void 0:d)),attrHooks:{type:{set:function(a,b){if(!o.radioValue&&"radio"===b&&B(a,"input")){var c=a.value;return

5

{}

Line 1, Column 1

S 中 月 % ☺

ElementsConsoleSourcesNetworkPerformanceMemoryApplicationSecurityAudits

jquery.min.js jquery.min.js.formatted x

1 /*! jQuery v3.2.1 | (c) JS Foundation and other contributors | jquery.org/license */

2 !function(a, b) {

3 "use strict";

4 "object" == typeof module && "object" == typeof module.exports ? module.exports = a.document ? b(a, !0) : function(

5 if (!a.document)

6 throw new Error("jQuery requires a window with a document");

7 return b(a)

8 } : b(a)

9 }("undefined" != typeof window ? window : this, function(a, b) {

10 "use strict";

11 var c = []

12 , d = a.document

13 , e = Object.getPrototypeOf

14 , f = c.slice

15

Line 1, Column 1

7.快速定位引起元素变化的js代码

有时候需要知道元素状态的变化是哪里的js控制的，有以下两种方法可以快速定位



8.查看dom元素绑定的事件

