

述职报告

软件研发部

聂骁骏

我心中的尚阳

- 待人友善
- 工作积极
- 气氛活跃

工作总结

- 金茂 H5: Vue + Vuex
- 金茂/东方锅炉 Web: AngularJS

金茂 H5

中国移动

13:49

57%

192.168.22.150

< 选择会议室

中国金茂

开始时间

 2019/01/23 13:50

时长

 01:00

○ 总部22层大会

07 08 09 10 11 12 13 14 15 16 17 18

○ 总部测试会议室

07 08 09 10 11 12 13 14 15 16 17 18

下一步

< >   

中国移动

13:50

57%

192.168.22.150

< 组织架构

中国金茂

返回上级

选择公司

○ 金茂北京

>

○ 金茂长沙

>

○ 金茂上海

>

○ 金茂广州

>

○ 金茂重庆

>

○ 金茂南京

>

○ 金茂酒店

>


☒ 金茂商业

>

○ 物业事业部

>

确定

< >   

金茂/东方锅炉 WEB

Umeet

三

我的会议

历史会议

会议报表

安排会议

监控管理

系统管理

V1.1.1 尚阳股份 400-618-3810

服务器时间：2019-03-03 10:20:41

2

超级管理员

首页 > 我的会议 > 预约会议

会议名称

本地会议室

所属部门

实施部

会议级别

C级

会议性质

本地会议

开会时间

2019-03-03

10:25

立即开会

结束时间

2019-03-03

11:20

是否有领导参加

会议联系人

选择会议联系人

参会人

选择参会人

会议要求

☐ 《今日东方锅炉》中文

☐ 《今日东方锅炉》英文

☐ 综合样本

☐ 纸、笔

☐ 矿泉水

☐ 咖啡

☐ 茶水

☐ 电脑

☐ 打印座位牌

☐ 投影仪

开会地点

☐ 德阳

☒ 成都

☐ 自贡

选择其他会议室

成都参会人数

10

自动匹配会议室

会议室	开会地点	类型	主会场
成都-本地-共有-会议室01	成都	本地会议室	

确定

返回

工作收获

1. 对angularJS中指令的理解
2. 合理的组件设计
3. 代码重构
4. 页面性能优化

I. 对指令的理解

与 Vue 和 React 不同，AngularJS 在 1.5 版本之前是没有组件方法的。

AngularJS 中组件是一种特殊的指令。

指令有四种类型：

- **元素名称(E)**: `<my-dir></my-dir>`
- **属性(A)**: ``
- **类名(C)**: ``
- **注释(M)**: `<!-- directive: my-dir exp -->`

下拉菜单 DEMO

原方案：引入整个 UI 库 **layui**，使用其中的下拉菜单组件

新方案：使用 **bootstrap** 提供的样式，创建两个 **angularJS** 组件：

infinite-dropdown: 包含按钮和第一级菜单，利用事件冒泡，监听点击事件

infinite-submenu: 二级及以下菜单，如果某一个菜单有子菜单，则递归调用

HTML 中使用组件 infinite-dropdown:

```
<infinite-dropdown selected-item="selectedCompany" list-array="compan
```

组件定义:

```
angular.module('myApp').directive('infiniteDropdown', [function() {  
  return {  
    templateUrl: 'infinite-dropdown.template.html',  
    restrict: 'E',  
    scope: {  
      selectedItem: '=',  
      listArray: '<'  
    },  
    link: function(scope, element, attrs) {  
      element.on('click', function(event) {  
        // 利用事件冒泡监听点击事件...  
      });  
    }  
  };  
}]);
```

子菜单组件 infinite-submenu:

```
angular.module('myApp').directive('infiniteSubmenu', [function() {  
  return {  
    templateUrl: 'infinite-submenu.template.html',  
    restrict: 'E',  
    scope: {  
      listArray: '<'  
    }  
  };  
}]);
```

模板文件 (模板内部递归地使用组件本身):

```
<ul class="dropdown-menu">  
  <li ng-repeat="item in listArray" ng-class="{ 'dropdown-submenu': it  
    <a href="#" data-company-id="{{item.value}}" data-company-name="{  
    <infinite-submenu ng-if="item.children" list-array="item.children  
  </li>  
</ul>
```

II.合理的组件设计

《JS设计模式与开发实践》中：

开放-封闭原则的思想：

当需要改变一个程序的功能或者给这个程序增加新功能的时候，可以使用增加代码的方式，但是**不允许改动程序的源代码**。

让程序尽量遵守**开放-封闭原则**的规律，是找出程序中将要发生变化的地方，然后把**变化**封装起来。

合理的组件设计: 找出组件中将要发生变化的地方，然后把**变化**封装起来。

如何封装变化:

- 向组件中传值(基本数据和对象，如开源库的配置文件)
- 放置挂钩(layDate 中的ready/change/done 挂钩)
- 向组件中传入回调函数(特殊的挂钩)

DEMO 1 - 分页组件

- {{item}}

页面中引入组件:

```
<tm-pagination config="paginationConfig"></tm-pagination>
```

页面中监听页面变化:

```
$scope.$watch(  
  'paginationConf.currentPage + "|" + paginationConf.itemsPerPage',  
  function(newVal, oldVal) {  
    if (newVal == oldVal) return;  
    var new_val_array = newVal.split("|");  
    var old_val_array = oldVal.split("|");  
    var pageSize = Math.ceil($scope.paginationConf.totalItems / $scope.  
    if (new_val_array[0] > pageSize) return;  
    if (!$scope.ids) $scope.initUser(0);  
    else $scope.initUser($scope.ids);  
  }, true);
```

不足之处：职责有点混乱，页面与组件之间存在一定的耦合性。

将不变的和变化的分隔开来：

- 不变的：监听当前页的变化，此职责应该属于组件
- 变化的：监听到变化后做的事情，此职责应该属于页面

更加合理的设计：组件内部监听变化后，执行页面委托给组件的函数。

页面中引入组件:

```
<tm-pagination config="paginationConfig" callback="filterListArray()">
```

组件内部监听页面变化:

```
directive("tmPagination", [
  function() {
    return {
      // 省略...
      scope: {
        config: "=", // 分页配置信息
        callback: "&" // 切换页面时的回调函数
      },
      link: function(scope, element, attrs) {
        scope.$watch(
          'config.currentPage + "|" + config.itemsPerPage',
          function(newVal, oldVal) {
            // 省略...
            callback(); // 执行回调函数
          }
        );
      }
    };
  }
]);
```


DEMO 2 - 时间选择组件

变化的:

- time: date对象, 时间
- minTime / maxTime: date对象, 最小/最大时间
- mode: 字符串, 为'date'时仅显示日期弹窗
- isSameDay: 布尔值, 为true时日期弹窗不可用
- minutesGradient: 数值, 分钟梯度, 用于向后取整
- increasedMinutes: 数值, 默认时间增加的分钟数

不变的:

- 初始化 layDate 插件
- 插件中选择日期和时间后, 更新 model 中的值
- 最大/最小值的限制等等

HTML 中使用组件 date-time-selector:

```
<date-time-selector time="startTime" minutes-gradient="5"></date-time-selector>
<date-time-selector time="endTime" min-time="startTime" minutes-gradient="5"
                    increased-minutes="60" is-same-day="true">
</date-time-selector>
<date-time-selector time="debugTime" max-time="startTime" mode="date">
```

组件定义:

```
angular.module('myApp').directive('dateTimeSelector', [function() {
  return {
    templateUrl: 'src/cases/case-date-time-picker/date-time-selector.html',
    scope: {
      time: '=',
      minTime: '<?',
      maxTime: '<?',
      mode: '@?',
      isSameDay: '<?',
      minutesGradient: '<?',
      increasedMinutes: '<?'
    },
    controller: ['$scope', '$timeout', '$filter', '$interval', 'dateTimeSelectorController'],
  };
  function controller($scope, $timeout, $filter, $interval) {
    var intervalTimer = null;
```

III. 代码重构

《JS设计模式与开发实践》中：
模式和重构之间有着一种与生俱来的关系。从某种角度来看，**设计模式**的目的就是为许多**重构**行为提供目标。

重构手段：

1. 优化 if/else 语句
2. 提炼过长的函数

1. 优化 IF/ELSE 语句

如果一个函数体内有一些条件分支语句，而这些条件分支语句内部散布了一些重复代码，那么就有必要进行合并去重工作。

向后端查询会议室前，设置查询条件（修改前）

```
if ($scope.searchConditions != null && $scope.searchConditions.length)
  if ($scope.meeting_group_id == null || $scope.meeting_group_id.length)
    $scope.QueryCondition = {
      pageSize: $scope.paginationConf.itemsPerPage,
      pageNo: $scope.paginationConf.currentPage,
      conditions: [$scope.searchConditions]
    };
  else {
    $scope.QueryCondition = {
      pageSize: $scope.paginationConf.itemsPerPage,
      pageNo: $scope.paginationConf.currentPage,
      conditions: [
        $scope.searchConditions,
        ['sql', 'meeting_group_id in (' + $scope.meeting_group_id + '
      ]
    ];
  }
```

向后端查询会议室前，设置查询条件（修改后）

```
var pageSize = $scope.paginationConf.itemsPerPage;
var pageNo = $scope.paginationConf.currentPage;
var conditions = [];
if ($scope.searchConditions != null && $scope.searchConditions.length)
    conditions.push($scope.searchConditions);
}
if ($scope.meeting_group_id != null && $scope.meeting_group_id.length)
    conditions.push(['sql', 'meeting_group_id in (' + $scope.meeting_group_id + ')']);
}
$scope.QueryCondition = {
    pageSize: pageSize,
    pageNo: pageNo,
    conditions: conditions
};
```

2. 提炼过长的函数

在 JS 开发中，我们大部分时间都在与函数打交道。

如果一个函数过长，不得不加上若干注释才能让这个函数显得易读一些，那这些函数就很有必要进行重构。

提炼函数的好处

- 小函数比大函数更易于阅读
- 独立出来的函数有助于代码复用
- 独立出来的函数更容易被覆写
- 良好的函数命名本身就起到了注释的作用

用于新建、修改会议的函数(580行)

```
//会议添加或更新    0 不验证占用情况    否则验证
$scope.addOrUpdate = function(isValidate) {
    if ($scope.isMixed == 0) {
        $scope.meeting.capacity = 0;
    }
    delete $scope.meeting.isMixed;
    delete MeetingReserve.isMixed;
    //验证密码是否为数字
    $scope.checkPassword();
    //判断是否需要强制会议密码
    if (
        $scope.globalConfig.IS_NEED_PWD == 0 &&
        ($scope.meeting.password == '' || $scope.meeting.password == unde
    ) {
        if ($scope.meeting.type == 0) {
            logger.logWarning($rootScope.data.enterMeetingPassword):
```

函数内部结构

```
// 创建会议
if (!MeetingReserve.uuid) {
  if ($scope.meeting.is_forever == 1) {
    // 永久会议
  } else if ($scope.meeting.is_foreverOrNow == 1 && $scope.isMixed != 1) {
    // 立即视频会议
  } else if ($scope.meeting.is_foreverOrNow == 1 && $scope.isMixed == 1) {
    // 混合会议立即开始
  } else if ($scope.isMixed == 1) {
    // 正常预约混合会议
  } else {
    // 正常预约
  }
}
// 修改会议
if (MeetingReserve.uuid) {
```

```
fy($filter("meetingFilter"))($scope.meeting));  
addMixed(  
  validate },  
  lter("meetingFilter"))($scope.meeting)),  
  == 10026) {  
    _box = false;  
    .bandwidthObject = $scope.temple.bandwidthObject;  
    .categoriesObject = $scope.temple.categoriesObject;  
    .meetingtagObject = $scope.temple.meetingtagObject;  
    = layer.confirm(  
      ata.conferenceConflict,  
      tScope.data.confirm, $rootScope.data.cancel],  
      ootScope.data.hint, "font-size:18px;"] //按钮  
OrUpdate(0);  
e(indexModal);  
g("cancel");  
status == 200) {  
  Factory.reset(MeetingReserve);  
  out.conference_mine");  
  .bandwidthObject = $scope.temple.bandwidthObject;  
  .categoriesObject = $scope.temple.categoriesObject;  
  .meetingtagObject = $scope.temple.meetingtagObject;  
  _box = false;  
  s == 10027) {  
    _decs").modal("show");  
    ngRoomLists = data.data;
```

```
>> 1 1 << // 修改预约的混合会议  
2 2 // c 正常修改混合会议  
3 3 MixedMeetingService.updateMixed(  
4 4 { isValidate: isValidate },  
5 5 JSON.stringify($filter("meetingFilter"))($scope.meeting)),  
6 6 function(data) {  
7 7   if (data.status == 10026) {  
8 8     $scope.loading_box = false;  
9 9     $scope.meeting.bandwidthObject = $scope.temple.bandwidthObject;  
10 10    $scope.meeting.categoriesObject = $scope.temple.categoriesObject;  
11 11    $scope.meeting.meetingtagObject = $scope.temple.meetingtagObject;  
12 12    var indexModal = layer.confirm(  
13 13      $rootScope.data.conferenceConflict,  
14 14      {  
15 15        btn: [$rootScope.data.confirm, $rootScope.data.cancel],  
16 16        title: [$rootScope.data.hint, "font-size:18px;"] //按钮  
17 17      },  
18 18      function() {  
19 19        $scope.addOrUpdate(0);  
20 20        layer.close(indexModal);  
21 21      },  
22 22      function() {  
23 23        console.log("cancel");  
24 24      }  
25 25    );  
26 26   } else if (data.status == 200) {  
27 27     MeetingReserveFactory.reset(MeetingReserve);  
28 28     $state.go("layout.conference_mine");  
29 29   } else {  
30 30     console.log(data);  
31 31     $scope.loading_box = false;  
32 32     $scope.meeting.bandwidthObject = $scope.temple.bandwidthObject;  
33 33     $scope.meeting.categoriesObject = $scope.temple.categoriesObject;  
34 34     $scope.meeting.meetingtagObject = $scope.temple.meetingtagObject;  
35 35     if (data.status == 10027) {  
36 36       $("#template decs").modal("show");
```

新建 / 修改混合会议的区别

并且会议室是否超过最大方限制

```
attend_inuse != "0" &&  
roleId != "1" &&  
meetingRoomUuids.length > $scope.login.user.max_attend
```

```
$rootScope.data.chooseMeetingRoom);
```

```
= false;
```

```
bandwidthObject = $scope.template.bandwidthObject;
```

```
meetingtagObject = $scope.template.meetingtagObject;
```

开启, 并且会议时长超过设定时长

```
time != "-1" &&
```

```
roleId != "1" &&
```

```
$scope.approve.max_time || ($scope.login.user.roleId != "1" &&
```

```
> 0 && $scope.hours == $scope.approve.max_time))
```

```
$rootScope.data.meetingTimeLength);
```

```
= false;
```

```
bandwidthObject = $scope.template.bandwidthObject;
```

```
meetingtagObject = $scope.template.meetingtagObject;
```

```
$scope.meeting.is_foreverOrNow;
```

```
is_foreverOrNow = is_foreverOrNow;
```

```
startTime = new Date();
```

```
nowMixedMeeting(  
    validate },  
    filter("meetingFilter")($scope.meeting)),  
    == 10026) {  
        loading_box = false;
```

```
>> 1 1 << //b 立即视频会议
```

```
2 2 //判断最大方是否开启, 并且会议室是否超过最大方限制
```

```
3 3 if (  
4 4     $scope.approve.max_attend_inuse != "0" &&  
5 5     $scope.login.user.roleId != "1" &&  
6 6     $scope.meeting.meetingRoomUuids.length > $scope.login.user.max_attend  
7 7 ) {  
8 8     logger.logWarning($rootScope.data.chooseMeetingRoom);  
9 9     $scope.loading_box = false;  
10 10    $scope.meeting.bandwidthObject = $scope.template.bandwidthObject;  
11 11    $scope.meeting.meetingtagObject = $scope.template.meetingtagObject;  
12 12    return;  
13 13 }  
14 14 //判断会议时长限制是否开启, 并且会议时长超过设定时长  
15 15 if (  
16 16     $scope.approve.max_time != "-1" &&  
17 17     $scope.login.user.roleId != "1" &&  
18 18     ($scope.hours > 0 && $scope.hours == $scope.approve.max_time ||  
19 19     ($scope.minutes > 0 && $scope.hours == $scope.approve.max_time))  
20 20 ) {  
21 21     logger.logWarning($rootScope.data.meetingTimeLength);  
22 22     $scope.loading_box = false;  
23 23     $scope.meeting.bandwidthObject = $scope.template.bandwidthObject;  
24 24     $scope.meeting.meetingtagObject = $scope.template.meetingtagObject;  
25 25     return;  
26 26 }  
27 27 var is_foreverOrNow = $scope.meeting.is_foreverOrNow;  
28 28 //将数据设置为默认值  
29 29 $scope.meeting.is_foreverOrNow = is_foreverOrNow;  
30 30 $scope.meeting.startTime = new Date();  
31 31 << MeetingService.nowMeeting(  
32 32     { isValidate: isValidate },  
33 33     JSON.stringify($filter("meetingFilter")($scope.meeting)),  
34 34     function(data) {  
35 35         if (data.status == 10026) {  
36 36             $scope.loading_box = false;
```

新建立即 混合会议 / 视频会议的区别

把创建和修改会议封装到一个函数中

```
// 创建、修改会议
if ($scope.meeting.is_forever == 1) {
    // 永久会议
    processForeverMeeting();
} else if ($scope.meeting.is_foreverOrNow == 1) {
    // 立即视频、混合会议
    processForeverOrNowMeeting();
} else if ($scope.isMixed == 1) {
    // 正常预约混合会议
    processMixedMeeting();
} else {
    // 正常预约
    processNormalMeeting()
}
```

使用职责链模式

使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。

我们可以把如下四个对象连成一条链，并沿着该链条传递处理会议的请求，直到有一个对象能处理它为止：

- processForeverMeeting() 处理永久会议
- processForeverOrNowMeeting() 处理立即会议
- processMixedMeeting() 处理混合会议
- processNormalMeeting() 处理正常会议

首先给出 Function.prototype.after 方法：

- 接受一个函数 fn 作为参数，返回一个函数；
- 此函数先执行函数自身
 - 能处理，返回运行结果；
 - 不能处理(返回 'NEXT_NODE')，则返回 fn 的运行结果

```
Function.prototype.after = function(fn) {  
  var _self = this;  
  return function() {  
    var ret = _self.apply(this, arguments);  
    if (ret === 'NEXT_NODE') {  
      return fn.apply(this, arguments);  
    }  
    return ret;  
  }  
};
```

修改节点对象：判断是否能处理，不能处理时，返回'NEXT_NODE'。

```
// 永久会议
function processForeverMeeting() {
  // 不为永久会议，交给下个节点处理
  if ($scope.meeting.is_forever !== 1) {
    return 'NEXT_NODE';
  }
  // 处理永久会议...
}

// 立即视频、混合会议
function processForeverOrNowMeeting() {
  // 不为立即会议，交给下个节点处理
  if ($scope.meeting.is_foreverOrNow !== 1) {
    return 'NEXT_NODE';
  }
  // 处理立即视频、混合会议
```


把四个对象练成一条链，然后执行请求。

```
var processMeeting = processForeverMeeting
    .after(processForeverOrNowMeeting)
    .after(processMixedMeeting)
    .after(processNormalMeeting);

processMeeting(); // 处理会议
```

优点：各节点之间互不影响，可灵活地拆分重组。

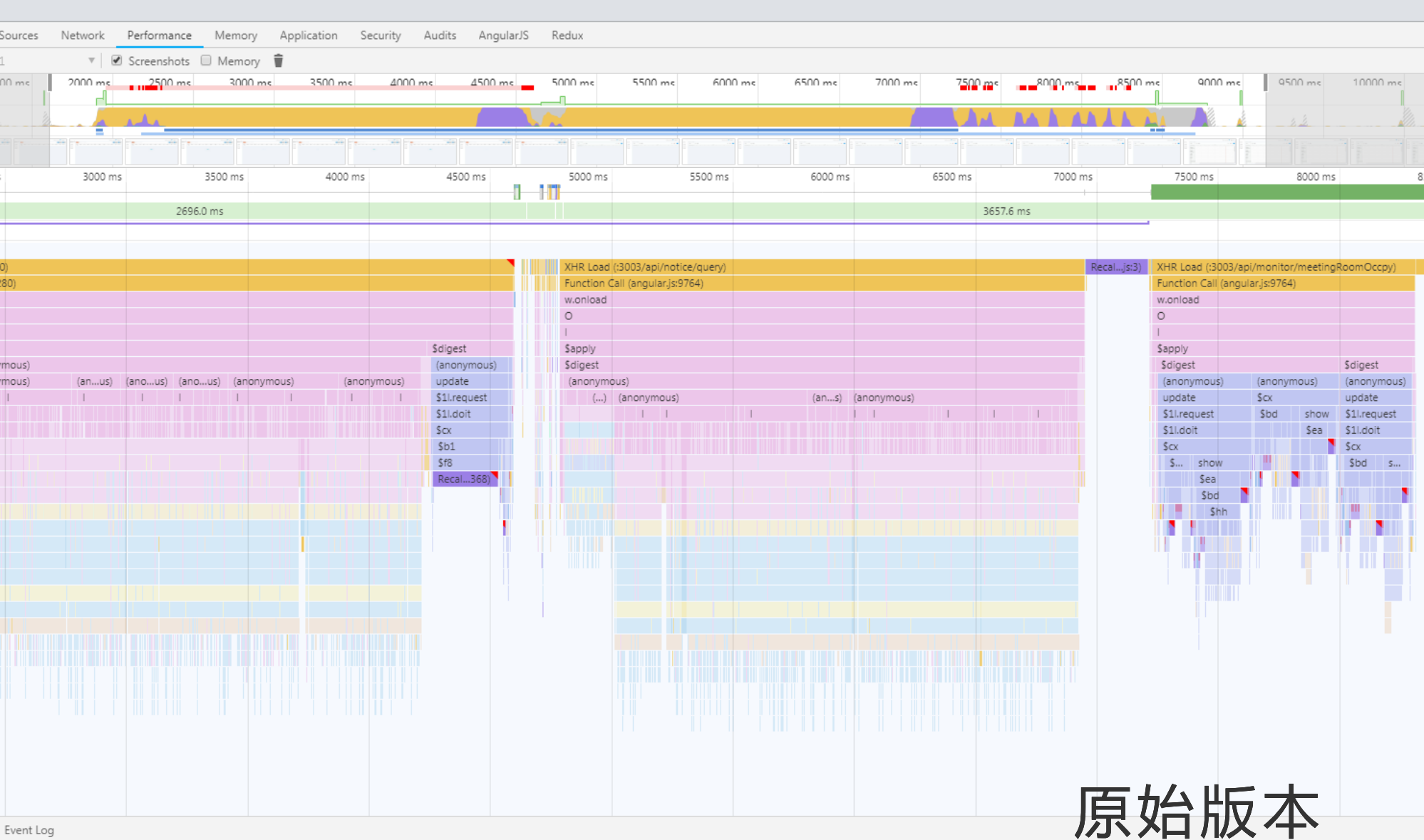
缺点：叠加了函数的作用域，链条太长会对性能有影响。

注：不支持异步。

IV. 页面性能优化

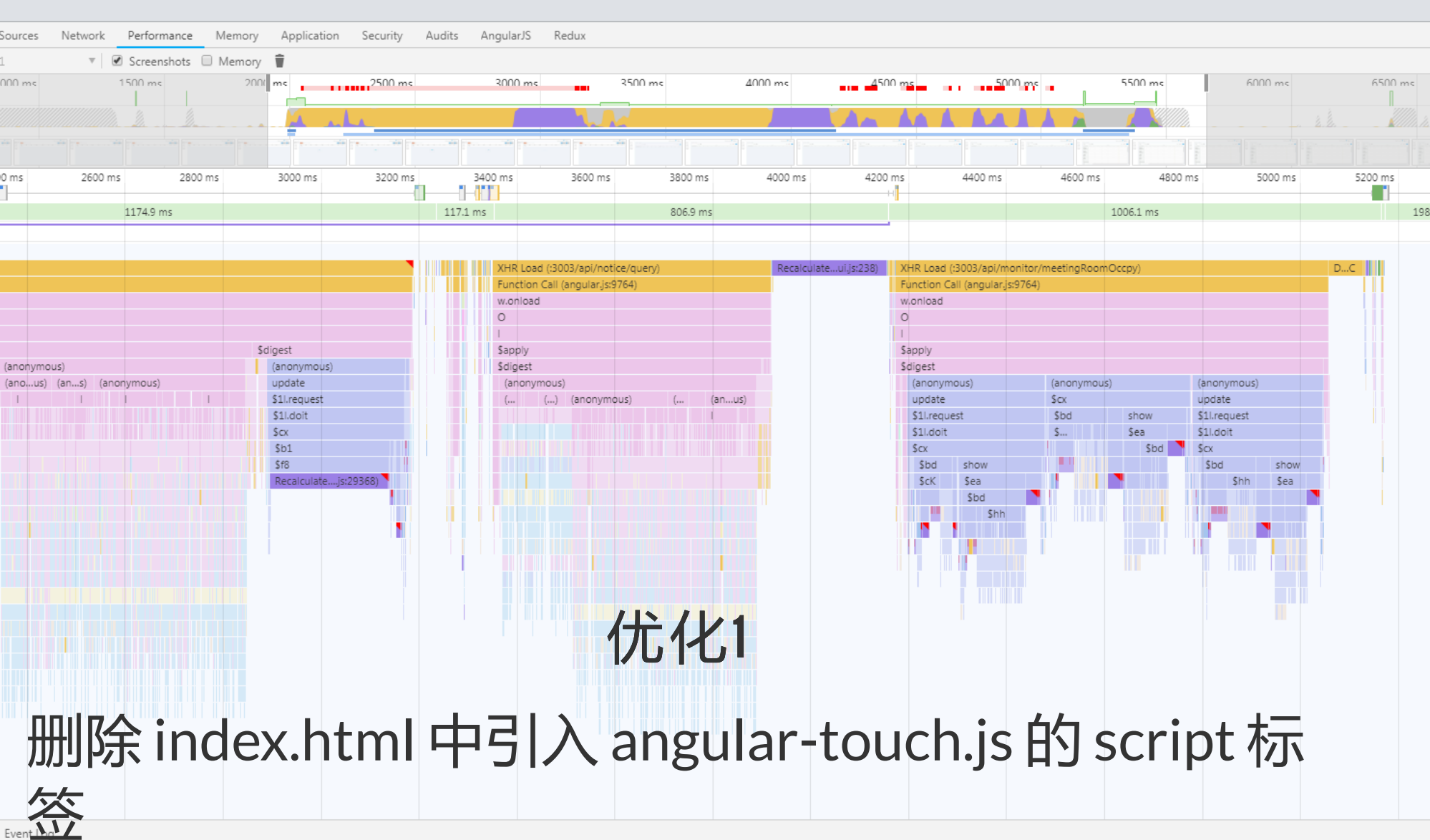
进入会议室预约页面时，等待时间过长

使用谷歌浏览器开发工具中的 **Performance** 对页面进行优化



原始版本

loading
scripting
rendering
painting
other
idle



优化1

删除 index.html 中引入 angular-touch.js 的 script 标签

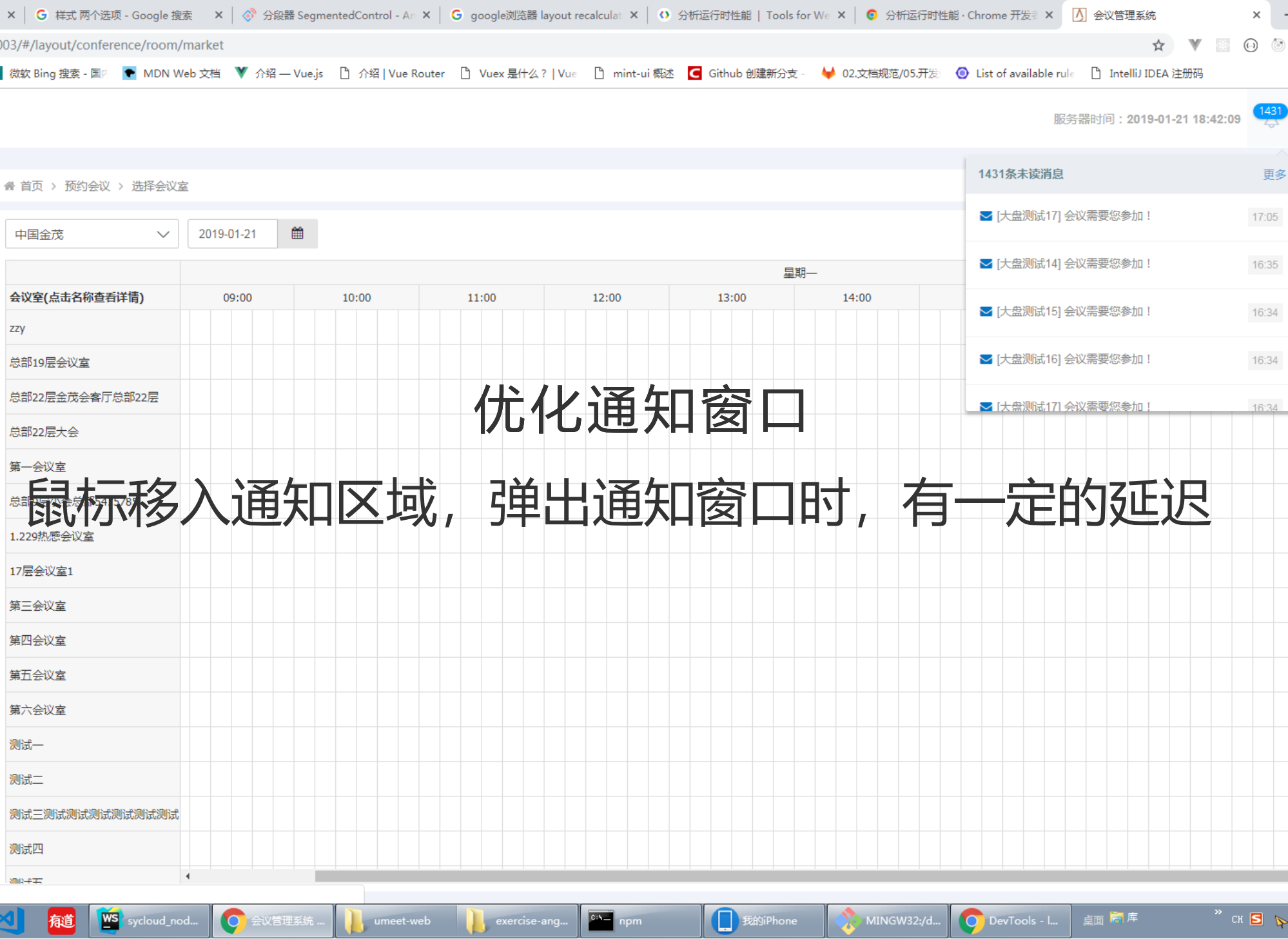
loading
scripting
rendering
painting
other
idle

页面加载时间对比

Item	Timer Fired	XHR Load(1)	Recalculate Style	XHR Load(2)
Origin	2.39 s	1.89 s	0.24 s	0.90 s
opt1	1.37 s	0.60 s	0.25 s	0.95 s

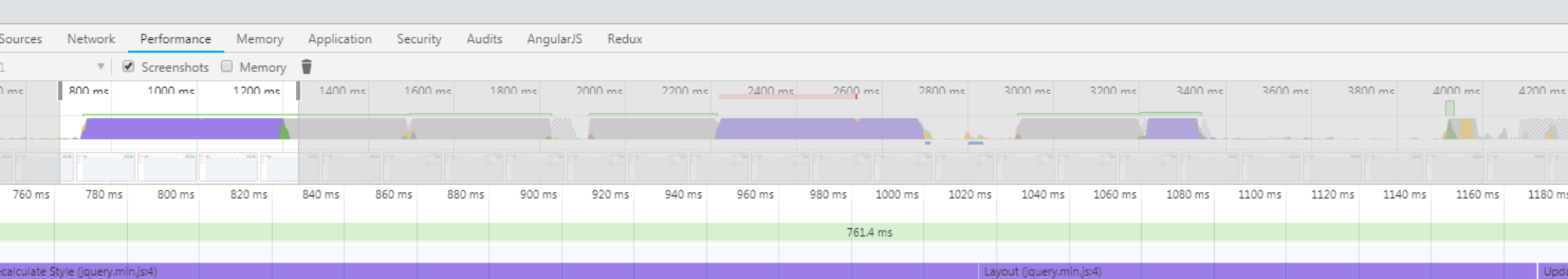
XHR load 1: /api/notice/query

XHR load 2: /api/monitor/meetingRoomOccupy



优化通知窗口

鼠标移入通知区域，弹出通知窗口时，有一定的延迟



原始通知窗口显示时的性能:

8602 个单元发生了样式重计算

16022 个节点中有 15763 个发生了布局计算(大小及位置)

Event Log

Scripting
Rendering
Painting
Other
Idle

app.js 中，获取了全部1431条未读消息：

```
$scope.query = function() {  
  NoticeService.query({}, function(data) {  
    $rootScope.allNotices = data.data.data;  
    $rootScope.headNotices = $filter("filter")($rootScope.allNotices,  
  });  
};
```

修改如下：

```
$scope.query = function() {  
  NoticeService.query({}, function(data) {  
    $rootScope.allNotices = data.data.data;  
    var headNotices = $filter("filter")($rootScope.allNotices, { stat  
    $rootScope.unreadMessageNum = headNotices.length;  
    $rootScope.headNotices = headNotices.slice(0, 20);  
  });  
};
```




136 个单元发生了样式重计算

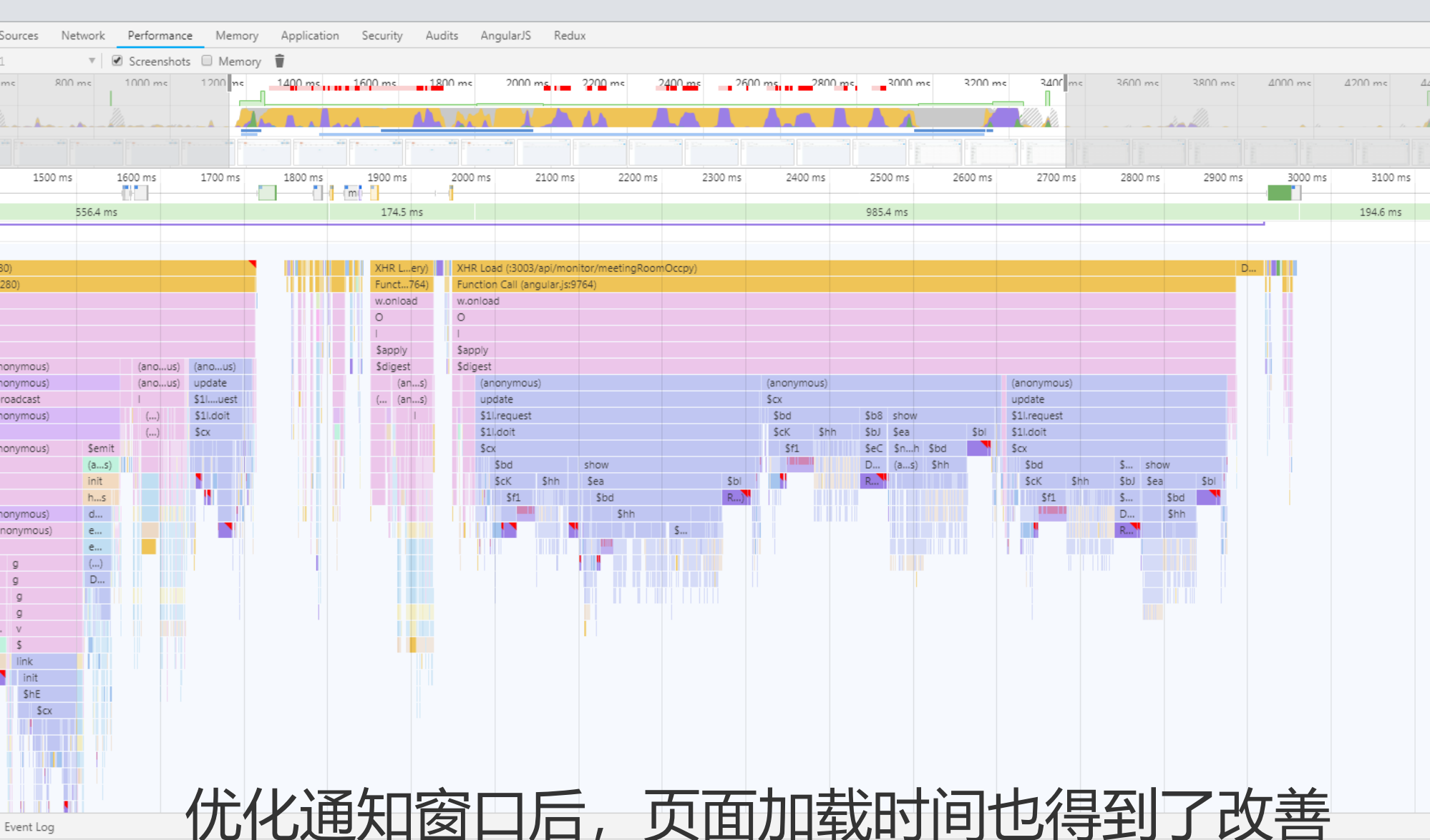
2213 个节点中有 241 个发生了布局计算(大小及位置)

通知窗口加载时间对比

Item	Event mouseover	Recalculate Style	layout	Update LayerTree	paint
Origin	6 ms	275 ms	154 ms	26 ms	13 ms
Opt	2 ms	5 ms	3 ms	13 ms	3 ms

优化前: 8602 个单元发生了样式重计算, 15763/16022 个节点发生了布局

优化后: 136 个单元发生了样式重计算, 241/2213 个节点发生了布局



优化通知窗口后，页面加载时间也得到了改善

loading
scripting
rendering
painting
other
idle

页面加载时间对比

Item	Timer Fired	XHR Load(1)	Recalculate Style	XHR Load(2)
Origin	2.39 s	1.89 s	0.24 s	0.90 s
Opt1	1.37 s	0.60 s	0.25 s	0.95 s
Opt2	455 ms	76 ms	8 ms	937 ms

目标

1. 成为一名合格的前端工程师：编写高性能、可复用和可维护性高的程序。
2. 了解其他相关知识：数据结构、算法和 HTTP 等。
3. 关注前沿技术：如 JavaScript 的超集 TypeScript，三大主流前端框架已支持。

谢谢!