WEB前端开发

—— 原生DOM&jQuery-DOM编程

美丽说商业前端负责人　熊伟烈

-wilee-　|　http://wilee.me　|　xiongwilee

课程大纲

第一章：DOM基础

1.1 认识DOM

1.2 DOM节点

1.3 DOM节点的查改增删

1.4 DOM相关的几个重要对象

1.5 实例分析

第二章：原生DOM事件模型

2.1 事件的分类

2.2 添加事件处理程序

2.3 移除事件处理程序

2.4 深入Event

2.5 实例分析

第三章：基于jQuery的DOM编程

3.1 万能的$

3.2 jQuery选择器

3.3 DOM节点的增加与移除

3.4 DOM属性与CSS

3.5 jQuery事件处理

3.6 jQuery动画

第四章：利用Chrome DevTool进行DOM调试

第五章：有趣的webAPP

随堂综合练习

Q/A时间

第一章：DOM基础

1.1 认识DOM

1.2 DOM节点

1.3 DOM节点的查改增删

1.4 DOM相关的几个重要对象

1.5 实例分析

1.1 认识DOM

1.1.1 DOM是什么?

Document Object Model(文档对象模型),可以以一种独立于平台和语言的方式访问和修改一个文档的内容和结构，是表示和处理一个HTML或XML文档的常用方法。

1.1.2 DOM能做什么?

能让网页实现更丰富的交互效果，下面随便列几个案例看欣赏一下DOM的奇幻魅力:

- 纸飞机： /demo/plane_move.html
- 标准时间比例的太阳系：/demo/solar.html (来自:codepen.io)
- 浏览器插件：Hi一下!

1.2 DOM节点

1.2.1 DOM中对节点的定义

HTML/XML**文档中的所有内容都是节点**， 例如：

- 整个文档是一个文档节点
- 每个 HTML 标签是一个元素节点
- 包含在 HTML 元素中的文本是文本节点
- 每一个 HTML 属性是一个属性节点
- 注释属于注释节点

1.2.2 节点类型

常见的DOM节点类型

元素名称	节点类型	节点值
元素	Node.ELEMENT_NODE	1
属性	Node.ATTRIBUTE_NODE	2
文本	Node.TEXT_NODE	3
注释	Node.COMMENT_NODE	8
文档	Node.DOCUMENT_NODE	9

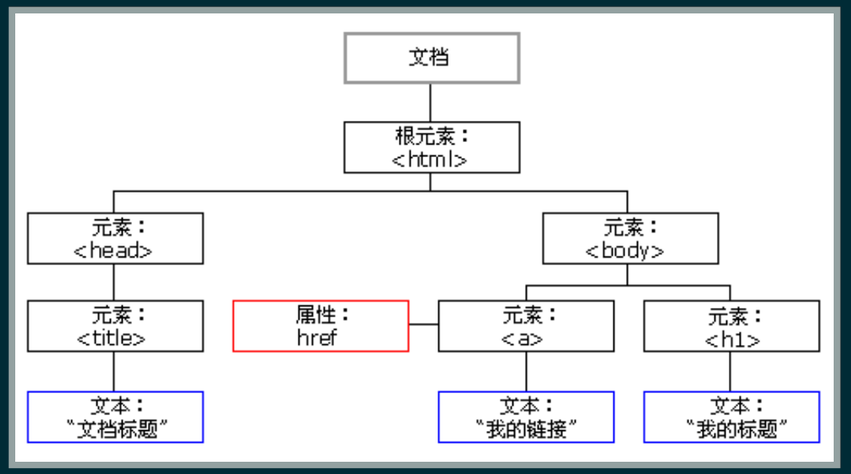
案例分析

1.2.3 节点层次

1.2.3.1 HTML文档

```
<!DOCTYPE HTML>
<html lang="en-US">
    <head>
        <meta charset="UTF-8">
        <title>文档标题</title>
    </head>
    <body>
        <h1>我的标题</h1>
        <a href="http://www.baidu.com">我的链接</a>
    </body>
</html>
```

1.2.3.2 DOM树



1.3 DOM节点的查改增删

本节将主要学习HTML DOM节点的查找、修改、创建与插入、删除等操作。

1.3.1 节点查询

1.3.1.1 查询DOM节点

简要介绍getElementById、getElementsByTagName、getElementsByClassName，示例代码：

```
// 查询id为 helloWorld 的节点
var helloWorld = document.getElementById('helloWorld');

// 查询页面上的所有 a 标签
var aTags = document.getElementsByTagName('a');

// 查询class为 .slide 的节点
var slides = document.getElementsByClassName('slide');
```

感兴趣的可以再试试：querySelector、querySelectorAll

案例分析

1.3.1.2 访问父节点与子节点

简要介绍parentNode、childNodes、firstChild、lastChild，示例代码：

```
// 查询id为 helloWorld 的节点
var helloWorld = document.getElementById('helloWorld');

// 获得helloWorld的父节点
// 这里使用的是parentNode，在非w3c标准下还有一个parentElement，必须理解二者的区别：
// ElementNode只是Node中的一种，nodeType为1
var parentNodeForHelloWorld = helloWorld.parentNode;

// 获得helloWorld的所有子节点
var childNodesForHelloWorld = helloWorld.childNodes;

// 获取hellWorld的第一个子节点
var firstChildForHelloWorld = helloWorld.firstChild;

// 获取hellWorld的最后一个子节点
var lastChildForHelloWorld = helloWorld.lastChild;
```

firstChild和lastChild都是获取所有类型的Node，而不仅仅是ElementNode。

案例分析

1.3.1.3 访问兄弟节点

简要介绍previousSibling、nextSibling，示例代码:

```
// 查询id为 helloWorld 的节点
var helloWorld = document.getElementById('helloWorld');

// 获得helloWorld的前一个兄弟节点
var previousNodeForHelloWorld = helloWorld.previousSibling;

// 获得helloWorld的后一个兄弟子节点
var nextNodeForHelloWorld = helloWorld.nextSibling;
```

previousSibling和nextSibling都是获取所有类型的Node，而不仅仅是ElementNode。

如果只想获取ElementNode，可以使用previousElementSibling和nextElementSilbling方法， 但这两个属性不是w3c标准，存在浏览器兼容性问题，慎用。

案例分析

1.3.2 节点修改

1.3.2.1 修改节点属性

节点属性的修改，可以通过element.setAttribute、element.attributeName的方式进行修改，比如针对如下节点：

```
<a id="helloWorld" class="cls-demo red" href="http://www.baidu.com">http://www.baidu.com</a>
```

可通过下面的两种方式，直接修改该节点的链接地址：href属性

```
// 查询id为 helloWorld 的A节点
var helloWorld = document.getElementById('helloWorld');

// 修改helloWorld的href值：通过element.setAttribute的形式
helloWorld.setAttribute('href','http://www.baidu.com');

// 修改helloWorld的href值：通过element.attributeName的形式
helloWorld.href = 'http://www.baidu.com';
```

案例分析

关于节点CSS class的访问，在HTML5标准中新增了classList API，该API中新增了CSS class的add、remove、contains、item、toggle等方法

```
// 通过classList的contains方法，判断节点是否包含某个CSS class
var hasClass = helloWorld.classList.contains('red'); // true
// 其他API可线下单独尝试
```

1.3.2.2 修改节点内容

本小节主要介绍节点内容的修改方法，包括修改HTML片段，以及文本内容。主要API为：innerHTML、innerText

```
// 继续以上一小节的<a>节点作为示例
var helloWorld = document.getElementById('helloWorld');

// 修改链接文本为：清华大学
helloWorld.innerHTML = '清华大学';
// 或通过innerText属性修改
helloWorld.innerText = '清华大学';

// 将链接文本替换成一张图片，即插入一个<img/>标签，此时只能使用innerHTML属性
helloWorld.innerHTML = '<img src="static/img/picture.png" alt="图片"/>';
```

- 通过innerHTML属性，可以向节点中插入任意HTML片段，必须明白，不是所有标签都支持innerHTML属性
- 通过innerText属性，只能向节点内插入文本，并且不是w3c标准，所以存在浏览器兼容性问题， 比如在Firefox、Chrome等浏览器中，则是通过 textContent 来实现：

```
// 标准浏览器下，通过 textContent 来修改文本内容
helloWorld.textContent = '清华大学';
```

案例分析

1.3.2.3 增加新节点

本节将介绍如何通过DOM动态创建新节点，并插入到文档中，用到的API：createElement、appendChild，假定文档如下：

```
<div id="helloWorld">
    <div id="inner"></div>
</div>
```

此时则可以通过如下的DOM操作，动态创建一个img节点并插入到helloWorld中

```
// 获得helloWorld节点
var helloWorld = docuement.getElementById('helloWorld');

// 通过createElement创建img节点，并设置相关属性
var imgElement = document.createElement('img'); // 注意API的使用方法
imgElement.src = '/static/img/picture.png'; // 设置图片地址
imgElement.alt = '图片';

// 将img追加到helloWorld子节点之后
helloWorld.appendChild(imgElement); // 采用子节点追加的方式插入文档
```

当有大批量的DOM节点需要插入到文档流中，需要用DocumentFragment来实现

案例分析

通过上面的DOM操作，文档片段将发生如下的变化：

```
<div id="helloWorld">
    <div id="inner"></div>
    <img src="/static/img/picture.png" alt="图片"/>
</div>
```

除了通过createElement创建ElementNode之外，还可以通过document.createTextNode创建TextNode 在节点的动态插入方面，除了使用appendChild之外，还可以通过element.insertBefore方法将节点插入到指定节点之前， 通过element.insertAjdacentElement方法在指定节点的指定位置处进行插入。

列举一个简易的DocumentFragment示例：

```
// 创建一个文档碎片
var docFrag = document.createDocumentFragment();
// 在碎片上加入100个div节点
for(var i = 0;i < 100;i++){
    docFrag.appendChild(document.createElement('div'));
}
// 加docFrag加入到DOM-Tree
document.body.appendChild(docFrag);
```

在项目开发中，用得最多的还是createElement、appendChild，其他API一般都会进行封装后再使用，可留作课后作业自行练习。

1.3.2.4 删除节点

本节将简单介绍从文档中将DOM节点进行删除操作，主要通过父节点的removeChild方法来实现。已如下HTML文档为例：

```
<div id="container">
    <div id="helloWorld"></div>
</div>
```

现在通过下面的DOM操作，来实现hellWorld节点的删除操作

```
// 获得待删除的节点
var helloWorld = document.getElementById('helloWorld');

// 调用父节点的removeChild方法来实现自身节点的删除
helloWorld.parentNode.removeChild(helloWorld); // 注意参数
```

如上代码执行成功后，文档将会发生如下变化：

```
<div id="container"></div>
```

注：目前部分高大上一点的浏览器（如Chrome）已经支持element.remove()直接移除节点本身：

```
document.getElementById('helloWorld').remove();
```

实例分析

1.4 DOM相关的几个重要对象

关于DOM操作，浏览器提供了较多内置对象，具体可以从下面文档中得到参考：

http://www.w3school.com.cn/htmldom/htmldom_reference.asp

但是本节只简单介绍几个比较典型的对象。

1.4.1 window对象

window对象是BOM的核心，扮演着Global对象的角色。

包括了整个WEB页面中所有可执行的JavaScript API；另外，可详细了解alert()、confirm()、prompt()、open()、close()、onload()

1.4.2 document对象

DOM编程中，最常用的对象，可详细了解cookie、title、URL、write()、getElementById()

1.4.3 navigator对象

通过此对象可以获取操作系统信息、浏览器信息等，在HTML5中，还可以获取地理位置、联网状态等； 可详细了解platform、userAgent、geolocation、onLine

1.4.4 location对象

包含当前页面URL相关的信息，并能控制页面的跳转等操作，可详细了解href、hash、search、protocal、reload()、replace()； 另外，location.href值等同于document.URL

1.5 实例分析

1. 纸飞机： /demo/plane_move.html
2. 标准时间比例的太阳系：/demo/solar.html
3. 浏览器插件：Hi一下！

用截止目前讲到的DOM操作，分析一下上面实例的实现原理。

第二章：原生DOM事件模型

2.1 事件的分类

2.2 添加事件处理程序

2.3 移除事件处理程序

2.4 深入Event

2.5 实例分析

当DOM被绑定事件之后会产生什么样的神奇功能：

1. memory：http://wilee.me/demo/memory.html (来自:codepen.io)
2. Flappy Dragon：http://js1k.com/2014-dragons/demo/1659
3. 2048：http://js1k.com/2014-dragons/demo/1824

2.1 事件的分类

1. **鼠标事件** : click、dbclick、mousedown、mouseup、mouseover、mousemove、mouseout等
2. **键盘事件** : kyedown、keyup、keypress等
3. **表单事件** : select、change、submit、focus、blur等
4. **媒介事件** : play、playing、abort、cancel等
5. **HTML5事件** : online、offline、message、popstate等
6. **页面事件** : load、error、resize、scroll等
7. **WebApp事件** : orientationchange、touchstart、touchmove、touchend等

2.2 添加事件处理程序

给DOM节点增加事件处理程序也有多种方法，本节都将进行简单介绍

2.2.1 通过节点属性显式声明

如下示例代码，表示直接在HTML中，显式地为按钮绑定了click事件，当该按钮有用户点击行为时，便会触发myClickFunc方法

```
<button id="btnHello" onclick="myClickFunc()">ClickMe</button>
```

myClickFunc的定义则在js中完成，示例如下：

```
// 事件处理程序的定义
var myClickFunc = function(evt){
    // TODO ...
};
```

实例分析

2.2.2 通过节点属性动态绑定

这种事件处理程序的绑定，属于第一种的变种形式

```
<button id="btnHello">ClickMe</button>
```

通过DOM操作进行动态绑定：

```
// 事件处理程序的定义
var myClickFunc = function(evt){
    // TODO ...
};

// 直接给DOM节点的 onclick 方法赋值，注意这里接收的是一个function
document.getElementById('btnHello').onclick = myClickFunc;
```

实例分析

2.2.3 通过事件监听的方式

相比而言，是事件处理程序的升级模式，是最靠谱的绑定方式，并且能给DOM节点增加多个事件监听，但不能绑定多个onclick事件

```
<button id="btnHello">ClickMe</button>
```

通过DOM操作进行动态绑定：

```
// 获取btnHello节点
var btnHello = document.getElementById('btnHello');

// 增加第一个 click 事件监听处理程序
btnHello.addEventListener('click',function(evt){
    // TODO sth 1...
},false);

// 增加第二个 click 事件监听处理程序
btnHello.addEventListener('click',function(evt){
    // TODO sth 2...
},false);
```

通过此种形式，可以给btnHello按钮绑定任意多个click监听； 注意，执行顺序与添加顺序相关

实例分析

2.3 移除事件处理程序

移除的方法必须和添加的方法相对应

- 如果事件是通过onclick的方式来绑定的，要删除该处理程序，可重新给onclick赋值即可，示例如下：

```
document.getElementById('btnHello').onclick = function(){}; // 空function即可
```

- 如果事件是通过addEventListener监听方式绑定的，要删除处理程序，需要用removeEventListener进行解绑，示例如下：

```
// 获取btnHello节点
var btnHello = document.getElementById('btnHello');

// 定义监听处理程序
var myClickFunc1 = function(evt){
    // TODO ...
};
// 增加事件监听
btnHello.addEventListener('click',myClickFunc1,false); // 指定句柄

// 移除事件监听
btnHello.removeEventListener('click',myClickFunc1,false); // 指定句柄
```

实例分析

2.4 深入Event对象

Event 对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态。 事件通常与函数结合使用，函数不会在事件发生前被执行！

2.4.1 Event对象的一些重要属性和方法

- 键盘的相关状态

  altKey（Alt键是否被按下）、ctrlKey（Ctrl键是否被按下）、metaKey（Meta键是否被按下）、 shiftKey（Shift键是否被按下）、keyCode/which（可通过键盘码判断出具体是那个键被按下）

- 鼠标的相关状态

  button（判断是那个键被按下：左键、右键、中键）、clientX/clienY/screenX/screenY/offsetX/offsetY/x/y （获取事件触发时鼠标的具体位置）

- 其他标准属性与方法

  type、target、timeStamp、bubbles、cancelable、currentTarget、preventDefault()、stopPropagation()

2.4.2 事件冒泡（传播）

事件触发时，会从目标DOM元素向上传播，直到文档根节点，一般情况下，会是如下形式传播：

```
targetDOM → parentNode → ... → body → document → window
```

如果希望一次事件触发能在整个DOM树上都得到响应，那么就需要用到事件冒泡的机制，如下示例代码：

```
<button id="btnHello">ClickMe</button>
```

```
// 给按钮增加click监听
document.getElementById('btnHello').addEventListener('click',function(evt){
    alert('button clicked');
},false);

// 给body增加click监听
document.body.addEventListener('click',function(evt){
    alert('body clicked');
},false);
```

在这种情况下，点击按钮“ClickMe”后，其自身的click事件会被触发，同时，该事件将会继续向上传播， 所有的祖先节点都将得到事件的触发命令，并立即触发自己的click事件；所以如上代码，将会连续弹出两个alert

实例分析

有些情况下，我们的需求要求，每个事件都是独立触发，所以我们必须阻止事件冒泡，针对如上示例代码，可改造为：

```
// 给按钮增加click监听
document.getElementById('btnHello').addEventListener('click',function(evt){
    alert('button clicked');
    evt.stopPropagation(); // 这条命令就能将此次事件完全结束掉，阻止冒泡
},false);

// 给body增加click监听
document.body.addEventListener('click',function(evt){
    alert('body clicked');
},false);
```

此时，点击按钮后，只会触发按钮本身的click事件，得到一个alert效果；该按钮的点击事件，不会向上传播，body节点就接收不到此次事件命令。

有两点需要注意：

- 不是所有的事件都能冒泡，如：blur、focus、load、unload都不能
- 不同的浏览器，阻止冒泡的方式也不一样，在w3c标准中，通过event.stopPropagation()完成， 在IE中则是通过自身的event.cancelBubble=true来完成

实例分析

2.4.3 事件默认行为

在项目过程中，我们会遇到各种各样的需求，比如：

- 点击form表单中的submit按钮后，表单不允许提交
- 点击a标签后，不允许发送页面跳转
- 对键盘进行控制，禁止输入

其实，这些需求都可以通过阻止事件的默认行为来完成，具体可以看下面的这个示例：

```
<a id="linkToBaidu" href="http://www.baidu.com">Baidu</a>
```

要求：点击Baidu链接后，页面不能发生跳转，而是弹出一个alert

```
// 给a标签增加click事件监听
document.getElementById('linkToBaidu').addEventListener('click',function(evt){
    alert('对不起，页面不会发生跳转！');
    evt.preventDefault(); // 阻止默认行为
},false);
```

BTW：其他情况均可通过这样的方式进行事件默认行为的阻止，至于阻止的方法，在不同浏览器中会存在差异，后续章节将会介绍兼容性的处理办法

实例分析

2.5 实例分析

1. memory：http://wilee.me/demo/memory.html (来自:codepen.io)
2. Flappy Dragon：http://js1k.com/2014-dragons/demo/1659
3. 2048：http://js1k.com/2014-dragons/demo/1824

用截止目前学到的DOM操作 & 事件模型 知识，分析如上实例的实现原理

第三章：基于jQuery的DOM编程

3.1 万能的$

3.2 jQuery选择器

3.3 DOM节点的增加与移除

3.4 DOM属性与CSS

3.5 jQuery事件处理

3.6 jQuery动画

3.1 万能的$

$符是jQuery(http://jquery.com/)的一个别名，通过它，能完成非常丰富的DOM操作，比如拿百度首页来做个小测试：

给百度首页所有的A链接都增加一个点击事件，点击后弹框显示链接内容； 并且将搜索框上方的前两个A链接颜色改为红色，第三个之后的A链接背景色改为随机。

```
// 给所有的A标签都增加点击事件
$('a').click(function(e){
    alert($(this).attr('href'));
});
```

```
// 生成随机颜色值：#RRGGBB
var getRandomColor = function(){
    return '#' + (new Number(Math.floor(Math.random() * 256))).toString(16)
        + (new Number(Math.floor(Math.random() * 256))).toString(16)
        + (new Number(Math.floor(Math.random() * 256))).toString(16);
};
// 将搜索框上方的A标签改成不同颜色
$('#nv a')                                          // 查询到搜索框上方的所有A链接
    .filter(':lt(2)').css('color','#f00')           // 将前两个A标签颜色改成红色
    .end()
    .filter(':gt(2)').each(function(){             // 将第三个标签之后的全部标签背景色改为随机
        $(this).css('background',getRandomColor());
    });
```

3.2 jQuery选择器

3.2.1 简单介绍选择器引擎：Sizzle

Sizzle是jQuery的御用选择器引擎，是jQuery作者John Resig写的DOM选择器引擎，速度号称业界第一。

- 项目主页

  http://sizzlejs.com/

- 项目文档

  https://github.com/jquery/sizzle/wiki

- 示例

选择器	实例	选取
*	$("*")	所有元素
#*id*	$("#lastname")	id="lastname" 的元素
.*class*	$(".intro")	所有 class="intro" 的元素
element	$("p")	所有 <p> 元素
.*class*.*class*	$(".intro.demo")	所有 class="intro" 且 class="demo" 的元素
[*attribute*]	$("[href]")	所有带有 href 属性的元素
[*attribute*=*value*]	$("[href='#']")	所有 href 属性的值等于 "#" 的元素
[*attribute*!=*value*]	$("[href!='#']")	所有 href 属性的值不等于 "#" 的元素
:enabled	$(":enabled")	所有激活的 input 元素
:disabled	$(":disabled")	所有禁用的 input 元素
:selected	$(":selected")	所有被选取的 input 元素
...		

案例分析

更多选择器可以参考：jQuery官网 ， 或者 W3School

3.2.3 jQuery对象和DOM对象

首先简单分析下jQuery源代码，搞清楚$查找出来的结果是什么样的： $(selector, context)

```
jQuery.fn.init = function(selector,context){
        // HANDLE: $(#id)
        } else {
            elem = document.getElementById( match[2] );
            if ( elem && elem.parentNode ) {
                // Inject the element directly into the jQuery object
                this.length = 1;
                this[0] = elem;
            }
            this.context = document;
            this.selector = selector;
            return this;
        }
        // HANDLE: $(expr, $(...))
        } else if ( !context || context.jquery ) {
            return ( context || rootjQuery ).find( selector );
        // HANDLE: $(expr, context)
        } else {
            return this.constructor( context ).find( selector );
        }
    // HANDLE: $(DOMElement)
    } else if ( selector.nodeType ) {
        this.context = this[0] = selector;
        this.length = 1;
        return this;
}
```

从前面的代码不难总结出，jQuery DOM元素和原生DOM元素之间存在这样的关系：

从jQuery对象转换为DOM对象：

```
// 获取 id=helloWorld 节点的jQuery对象
var jqDom = $('#helloWorld');
// 从jQuery对象中提取原生DOM
var el = jqDom[0];
// 或者通过jQuery对象提供的get(index)方法
var el = jqDom.get(0);
```

从DOM对象转换为jQuery对象：

```
// 获取 id=helloWorld 节点的原生对象
var el = document.getElementById('helloWorld');
// 转 jQuery 对象，直接用 $ 包装一下即可（参考前面jQuery.fn.init方法中的实现）
var jqDom = $(el);
```

另外，必须知道jQuery提供的each等遍历方法，获取到的item都是原生DOM：

```
// 获取所有 class=x-hello 的标签，并遍历进行单独处理
$('.x-hello').each(function(index,item){
    // 这里的 item 是DOM原生对象，也可以用 this 获取
    // 将每个 item 作为jQuery对象来使用
    $(item);    // or $(this);
});
```

3.3 DOM节点的增加与移除

3.3.1 jQuery操作DOM的一些常用方法：

方法	描述
addClass()	向匹配的元素添加指定的类名。
after()	在匹配的元素之后插入内容。
append()	向匹配元素集合中的每个元素结尾插入由参数指定的内容。
appendTo()	向目标结尾插入匹配元素集合中的每个元素。
attr()	设置或返回匹配元素的属性和值。
before()	在每个匹配的元素之前插入内容。
clone()	创建匹配元素集合的副本。
empty()	删除匹配的元素集合中所有的子节点。
hasClass()	检查匹配的元素是否拥有指定的类。
...	

更多DOM操作的方法可以参考：jQuery官网 ， 或者 W3School

3.3.2 如何创建jQuery DOM节点

通过 $(htmlContent) 的方式创建jQuery节点

```
// 创建一个 div 节点
$('<div/>');

// 创建一个 id=helloWorld，class=x-hello 的div节点
$('<div id="helloWorld" class="x-hello"/>');

// 或者用原生DOM的方式先创建，再包装成jQuery DOM（如果有这个需要的话）
var div = document.createElement('div');
div.id = 'helloWorld';
div.className = 'x-hello';
var jqDiv = $(div);
```

3.3.3 用append、appendTo或prepend、prependTo将DOM添加到页面

```
// 创建一个 id=helloWorld，class=x-hello 的div节点
var jqDiv = $('<div id="helloWorld" class="x-hello"/>');

// 通过 append 添加到页面
$('body').append(jqDiv);

// 通过appendTo添加到页面
jqDiv.appendTo('body');
```

案例分析

其他添加DOM节点的API这里就不列举了，课后可单独练习。

3.3.4 DOM节点移除

```
// 将 id=helloWorld、class=x-hello、以及所有strong标签都删掉
$('#helloWorld, .x-hello, strong').remove();

// 将 div#container 下的所有子节点都删掉
$('#container').empty();
```

可结合前面章节讲到的原生DOM操作，思考jQuery中这些API的实现原理。

其他添加DOM节点的API这里就不列举了，课后可单独练习。

3.4 DOM属性与CSS操作

3.4.1 DOM属性

方法	描述
addClass()	向匹配的元素添加指定的类名。
attr()	设置或返回匹配元素的属性和值。
hasClass()	检查匹配的元素是否拥有指定的类。
html()	设置或返回匹配的元素集合中的 HTML 内容。
removeAttr()	从所有匹配的元素中移除指定的属性。
removeClass()	从所有匹配的元素中删除全部或者指定的类。
toggleClass()	从匹配的元素中添加或删除一个类。
val()	设置或返回匹配元素的值。

案例分析

更多DOM属性操作可以参考：jQuery官网 ， 或者 W3School

3.4.2 CSS操作

CSS 属性	描述
css()	设置或返回匹配元素的样式属性。
height()	设置或返回匹配元素的高度。
offset()	返回第一个匹配元素相对于文档的位置。
offsetParent()	返回最近的定位祖先元素。
position()	返回第一个匹配元素相对于父元素的位置。
scrollLeft()	设置或返回匹配元素相对滚动条左侧的偏移。
scrollTop()	设置或返回匹配元素相对滚动条顶部的偏移。
width()	设置或返回匹配元素的宽度。

案例分析

更多CSS操作可以参考：jQuery官网，或者 W3School

3.4.3 数据操作

这些方法允许我们将指定的 DOM 元素与任意数据相关联。

方法名	功能
.data()	存储与匹配元素相关的任意数据。
jQuery.data()	存储与指定元素相关的任意数据。
.removeData()	移除之前存放的数据。
jQuery.removeData()	移除之前存放的数据。

更多数据操作可以参考：jQuery官网 ， 或者 W3School

3.5 jQuery事件处理

事件方法会触发匹配元素的事件，或将函数绑定到所有匹配元素的某个事件。

- 绑定实例：

```
$(document.body).click(function(event){
    $(this).hide();
})
```

- 触发实例：

```
$(document.body).click()
```

上面的例子将触发 body 元素的 click 事件。

3.5.1 jQuery事件的相关方法

- **浏览器事件** .resize()、.scroll()
- **表单事件** .blur()、.change()、.focus()、.select()、.submit()
- **键盘事件** .keydown()、.keypress()、.keyup()
- **鼠标事件** .click()、.dblclick()、.focusin()、.focusout()、.hover()、.mousedown()、.mouseenter()、.mouseleave()、.mousemove()、.mouseout()、.mouseover()、.mouseup()
- **文档加载** .holdReady()、.ready()
- **事件对象** event.currentTarget、event.data、event.isDefaultPrevented() …
- **事件绑定** .bind()、.delegate()、.off()、.on()、.one()、.trigger()、.triggerHandler()、.unbind()、.undelegate()

3.5.1 添加事件监听

3.5.1.1 通过jQuery对象的事件方法

```
// 给 id=helloWorld 的节点增加mousedown、mouseup事件
$('#helloWorld').mousedown(function(e){
    // TODO ...
}).mouseup(function(e){
    // TODo ...
});
```

3.5.1.2 通过bind或者on方式绑定事件，包括自定义事件

```
//给 id=helloWord 的节点绑定mousedown事件
$('#helloWorld').on('click',function(){
    console.log(this);
})

// 给 id=helloWorld 的节点增加mousedown、mouseup事件，以及自定义事件：sayHello
$('#helloWorld').bind({
    mousedown : function(e){
        // TODO ...
    },
    mouseup : function(e){
        // TODO ...
    },
    sayHello : function(e){ // 自定义事件
        // TODO ...
    }
});
```

3.5.1.3 事件委托/代理

如果要对非常多的节点进行事件绑定，或者如果新增节点需要绑定事件，例如：

```
<table>
    <tr><td>数据0001</td><td><button class="delete">删除</button></td></tr>
    <tr><td>数据0002</td><td><button class="delete">删除</button></td></tr>
    ……
    <tr><td>数据1000</td><td><button class="delete">删除</button></td></tr>
</table>
```

这时候就可以采用事件委托的办法：

```
$('table').on('click','.delete',function(event){
    console.log(this);
})
```

事件委托可以用在这两种场景：

1. 需要对非常多节点进行事件绑定，例如：上面案例中的table中的删除按钮
2. 需要对新增的节点进行事件绑定，例如：瀑布流的交互中，对新增的元素绑定事件；就不需要对新增的节点再去绑定事件了

3.5.2 触发事件

3.5.2.1 通过trigger触发

```
// 触发 id=hellWorld 节点的 mouseup, sayHello 事件
$('#helloWorld').trigger('mouseup').trigger('sayHello');
```

3.5.2.2 浏览器操作触发

3.5.3 在事件绑定时传递参数

有时候有种特殊需求，在事件处理的handler中需要用到外界的数据，于是可以这样做：

```
var message = 'Hi,I am Alien!';
$('#helloWorld').click({msg : message},function(e){
    // 通过 e.data 的方式来获取handler绑定时传入的参数
    alert(e.data.msg);
});
```

可能还有更变态的需求，希望在事件触发时动态传入参数，比如自定义事件：

```
// 当 sayHello 事件触发的时候，显示 message
$('#helloWorld').bind('sayHello',function(e, name, message) {
    // 动态的输出 message
    alert(name + ',' + message);
});

// 强制触发 sayHello 的时候，动态指定 message
// 注意，trigger的第二个参数是一个 Array，代表 eventHandler第一个参数之后的所有参数
$('#hellWorld').trigger('sayHello', [ 'John', 'Hi,I am Alien!' ] );
```

感兴趣的同学，课后可练习一下

3.5.4 在jQuery事件中处理冒泡和默认行为

jQuery已经处理完了所有浏览器的兼容性，所以直接用标准的方法来操作就行

```
// id=helloWorld 的节点点击后，阻止事件冒泡
$('#helloWorld').click(function(e){
    // TODO ...

    // 阻止冒泡
    e.stopPropagation();
});

// 表单 form#info 提交时，如果数据不合法则取消提交
$('form#info').submit(function(e){
    // 验证表单
    var isValid = validateTheForm(this);

    // 如果不合法，则取消提交
    if(!isValid){
        e.preventDefault();
    }
});
```

当然，如果你的需求是阻止事件冒泡，并且取消事件的默认行为，则可以直接 return false

```
// 用 a 标签做了个按钮，点击时需要阻止事件冒泡，同时取消默认的跳转行为
$('a.btn').click(function(e){
    // TODO ...

    return false;

})
```

3.6 jQuery动画

- 隐藏显示

  .hide()、.show()、.toggle()

- 淡入淡出

  .fadeIn()、.fadeOut()、.fadeTo()、.fadeToggle()

- 自定义

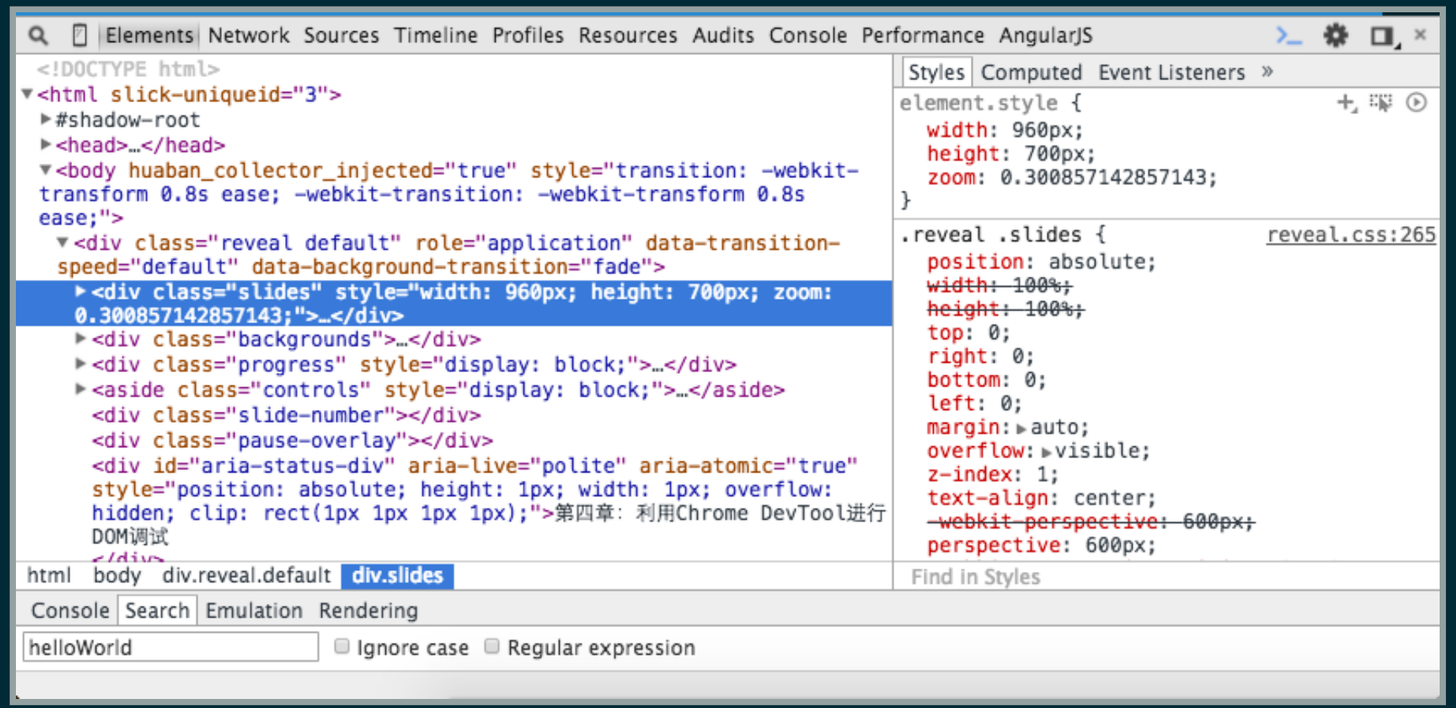
  .animate()、.clearQueue()、.delay()、.dequeue() jQuery.dequeue()、.finish() …

- 滑动

  .slideDown()、.slideToggle()、.slideUp()

jQuery官网

第四章：利用Chrome DevTool进行DOM调试

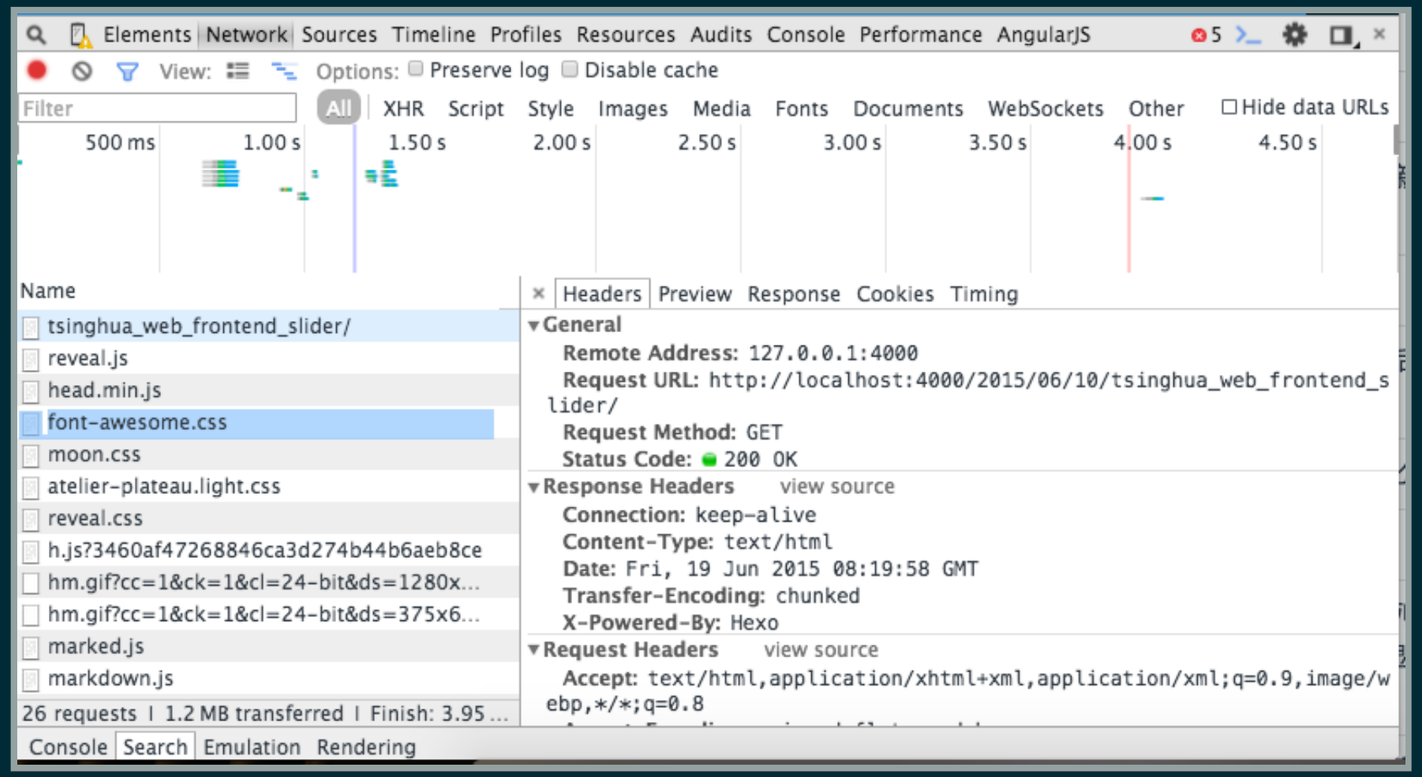


4.1 查看DOM结构

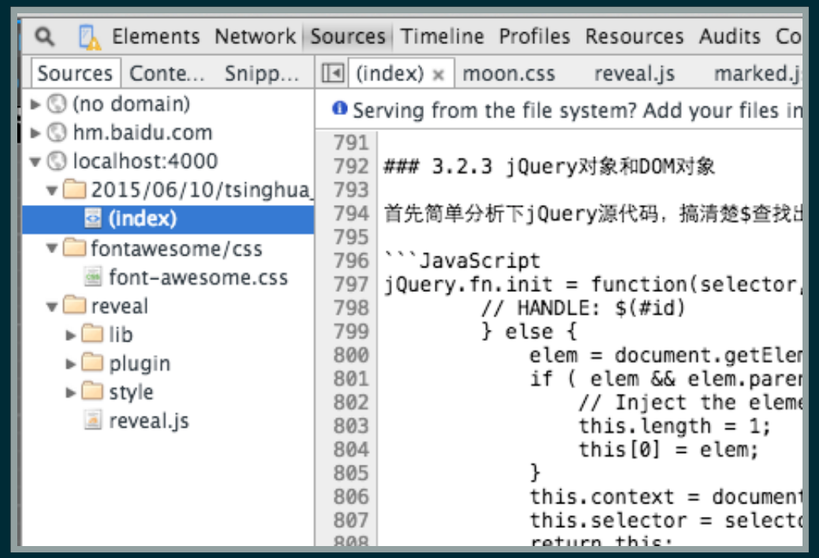

WEB前端开发
h1 959.984px × 107.185px
编程
美丽说商业前端负责人 熊伟烈
-wilee- | http://wilee.me | xiongwilee
Elements Network Sources Timeline Profiles Resources Audits Console Performance AngularJS
<section data-markdown data-markdown-parsed="true" class="present" style="display: block;">
<h1>WEB前端开发</h1>
<h3>— 原生DOM&jQuery-DOM编程</h3>

<hr>
…
美丽说商业前端负责人 熊伟烈
…
</section>
<section data-markdown data-markdown-parsed="true" class="future" aria-hidden="true" hidden style="display: block;">…</section>
</section>
Styles
element.style {
}
moon.css:61
.reveal h1 {
text-shadow: none;
}
moon.css:49
.reveal h1 {
font-size: 3.77em;
}
Find in Styles
html body div.reveal.default div.slides section.stack.present section.present h1

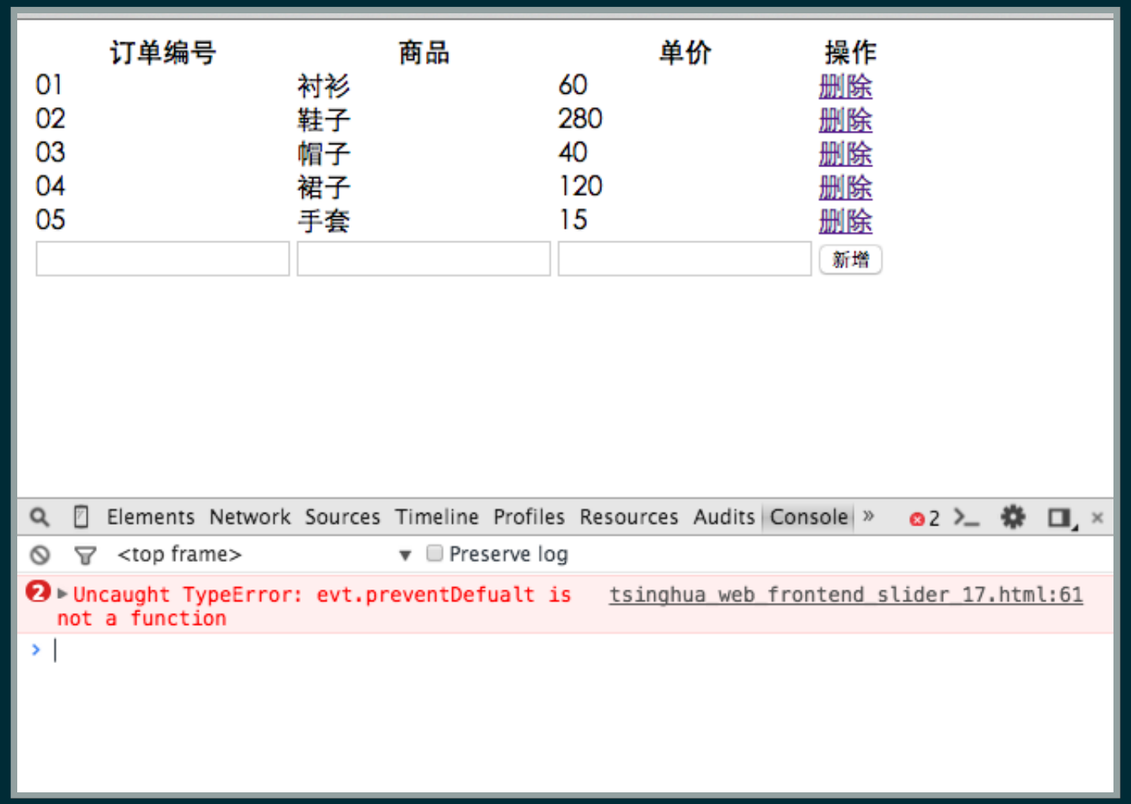
4.2 查看网络请求


Elements Network Sources Timeline Profiles Resources Audits Console Performance AngularJS
5
View: Options: Preserve log Disable cache
Filter All XHR Script Style Images Media Fonts Documents WebSockets Other Hide data URLs
500 ms 1.00 s 1.50 s 2.00 s 2.50 s 3.00 s 3.50 s 4.00 s 4.50 s
Name
tsinghua_web_frontend_slider/
reveal.js
head.min.js
font-awesome.css
moon.css
atelier-plateau.light.css
reveal.css
h.js?3460af47268846ca3d274b44b6aeb8ce
hm.gif?cc=1&ck=1&cl=24-bit&ds=1280x...
hm.gif?cc=1&ck=1&cl=24-bit&ds=375x6...
marked.js
markdown.js
26 requests | 1.2 MB transferred | Finish: 3.95 ...
Headers Preview Response Cookies Timing
General
Remote Address: 127.0.0.1:4000
Request URL: http://localhost:4000/2015/06/10/tsinghua_web_frontend_slider/
Request Method: GET
Status Code: 200 OK
Response Headers view source
Connection: keep-alive
Content-Type: text/html
Date: Fri, 19 Jun 2015 08:19:58 GMT
Transfer-Encoding: chunked
X-Powered-By: Hexo
Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Console Search Emulation Rendering

4.3 查看加载资源


Elements Network Sources Timeline Profiles Resources Audits Co
Sources Conte... Snipp...
(index) × moon.css reveal.js marked.j
(no domain)
hm.baidu.com
localhost:4000
2015/06/10/tsinghua_
(index)
fontawesome/css
font-awesome.css
reveal
lib
plugin
style
reveal.js
Serving from the file system? Add your files in
791
792 ### 3.2.3 jQuery对象和DOM对象
793
794 首先简单分析下jQuery源代码，搞清楚$查找出
795
796 ```JavaScript
797 jQuery.fn.init = function(selector,
798 // HANDLE: $(#id)
799 } else {
800 elem = document.getEler
801 if (elem && elem.parer
802 // Inject the eleme
803 this.length = 1;
804 this[0] = elem;
805 }
806 this.context = document
807 this.selector = selecto
808 return this;

4.4 JavaScript错误调试



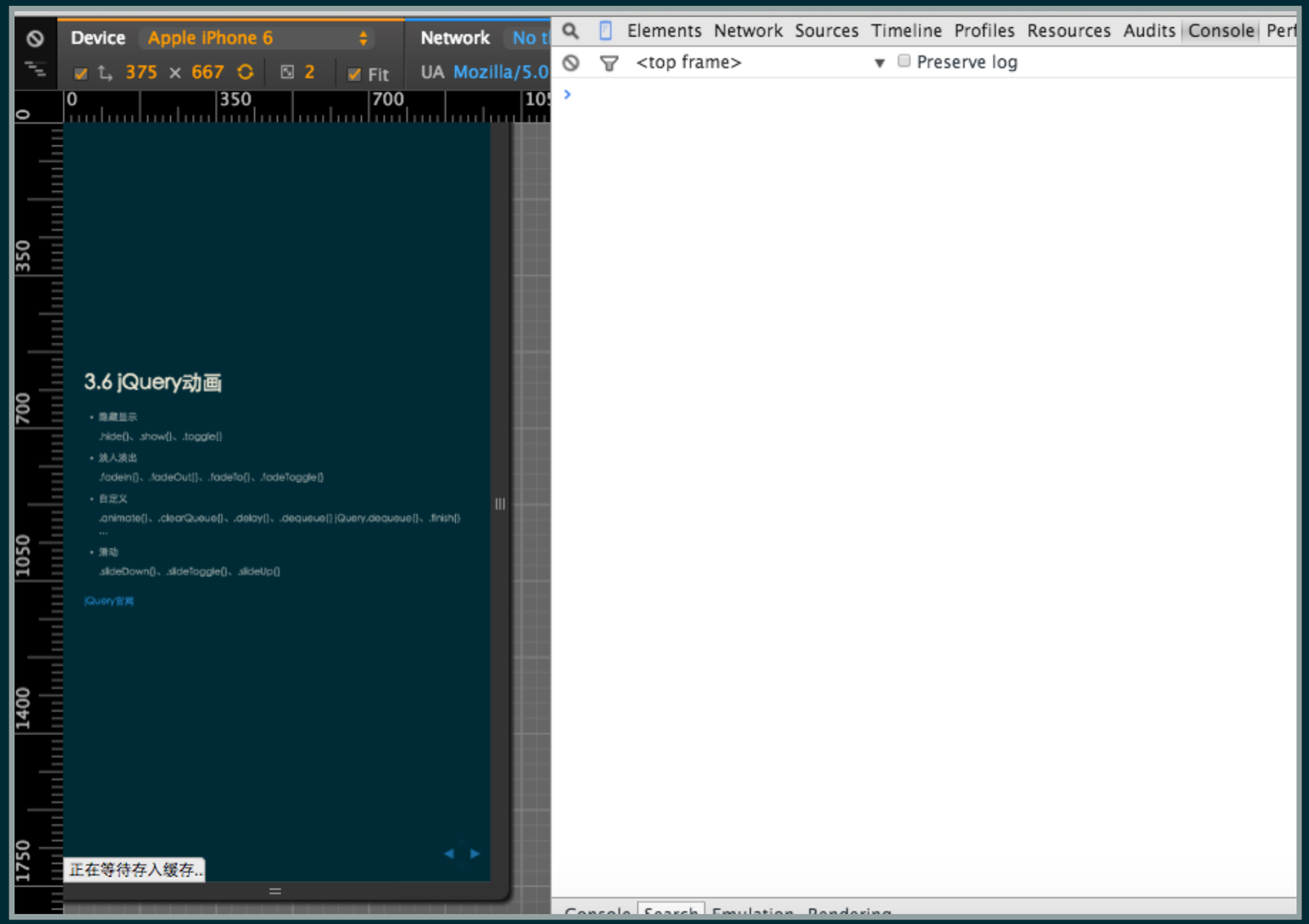
- 案例分析：

  http://codepen.io/xiongwilee/pen/bdvYwV?editors=101

- 实战操作：

  http://life.tsinghua.edu.cn/

4.5 WebAPP模拟调试



第五章：有趣的webAPP

webApp 并不是H5(HTML5) 而是采用HTML5、CSS3等技术手段实现的跨终端Web页面。（本章只简单介绍在PC Web开发的基础上进行移动端Web开发的一些差异，如果有兴趣可以参看HTML5，WebApp开发的相关的书籍。）

5.1 页面布局

同样的DOM结构需要适配在不同尺寸的终端上（响应式设计）。

5.2 事件

新增orientationchange、touchstart、touchmove、touchend等事件。

5.3 基础库

在webApp上更倾向于Zepto、jQuery mobil等轻量级的基础库。

- 案例分析

  http://wilee.me/2015/06/10/tsinghua_web_frontend_slider/

  http://passport.baidu.com/export/multi/index.html

  http://m.meilishuo.com/wx/mew_lucky/main?frm=catred_from_goqunred

- 综合讨论

  http://e.paipai.com/2015/618torture/index.html?from=timeline&isappinstalled=0

随堂综合练习

如下是2013年Google I/O大会上关于实时WEB应用的一个简短视频，在多个端（pad、phone、pc）上同时打开网页， 并拼接一个环形赛车跑道，玩家可以在这个跑道上进行赛车游戏。请看完视频后， 结合本次DOM编程的相关知识，分析该功能的实现原理

查看视频

Q/A