

```
1 @echo off
2 chcp 65001 >nul
3 setlocal enabledelayedexpansion
4
5 :: 设置标题
6 title GreenWeb 启动工具
7
8 echo =====
9 echo          GreenWeb 启动工具
10 echo =====
11 echo.
12 ";
13 :: 检查Node.js和npm是否安装
14 echo 检查环境依赖 ...
15 where node >nul 2>nul
16 if %errorlevel% neq 0 (
17     echo Node.js未安装, 准备自动安装 ...
18     call :InstallNodeJS
19 ) else (
20     for /f "tokens=*" %i in ('node -v') do set nodeVersion=%i
21     echo 已安装Node.js版本: !nodeVersion!
22 )
23
24 where npm >nul 2>nul
25 if %errorlevel% neq 0 (
26     echo npm未安装, 可能需要重新安装Node.js ...
27     call :InstallNodeJS
28 ) else (
29     for /f "tokens=*" %i in ('npm -v') do set npmVersion=%i
30     echo 已安装npm版本: !npmVersion!
31 )
32
33 :: 检查项目依赖是否安装
34 if not exist "node_modules" (
35     echo 项目依赖未安装, 准备安装 ...
36
37     :: 检查package.json文件是否存在
38     if not exist "package.json" (
39         echo 错误: 未找到package.json文件, 请确保在正确的项目目录中执行此脚本。
40         goto End
41     )
42
43     echo 正在安装项目依赖, 这可能需要几分钟 ...
44     call npm install
45     call cd server
46     call npm install
47     call cd ..
48     if !errorlevel! neq 0 (
49         echo 依赖安装失败, 请检查网络连接或手动运行 npm install
50         goto End
51     )
52     echo 依赖安装完成!
53 ) else if not exist "server/node_modules" (
```

```
53     echo 正在安装项目依赖，这可能需要几分钟 ...
54     call cd server
55     call npm install
56     call cd ..
57     echo 依赖安装完成!
58 )else (
59     echo 项目依赖已安装
60 )
61
62 :Menu
63 echo.
64 echo =====
65 echo                    请选择启动模式
66 echo =====
67 echo 1. 开发模式（前后端分离）
68 echo 2. 生产模式（整合前后端）
69 echo 3. 仅构建项目
70 echo 4. 更新项目依赖
71 echo 5. 退出
72 echo =====
73 echo.
74
75 set /p choice=请选择操作（1-5）:
76
77 if "%choice%"=="1" (
78     call :StartDevMode
79 ) else if "%choice%"=="2" (
80     call :StartProdMode
81 ) else if "%choice%"=="3" (
82     call :BuildProject
83 ) else if "%choice%"=="4" (
84     call :UpdateDependencies
85 ) else if "%choice%"=="5" (
86     echo 退出程序 ...
87     exit /b
88 ) else (
89     echo 无效的选择，请重新输入!
90     goto Menu
91 )
92
93 goto End
94
95 :InstallNodeJS
96 echo.
97 echo 准备安装Node.js ...
98 echo 正在下载Node.js安装程序 ...
99
100 :: 创建临时目录
101 set tempDir=%temp%\NodeJSInstall
102 mkdir %tempDir% 2>nul
103
104 :: 检查是否已安装curl
105 where curl >nul 2>nul
106 if %errorlevel% neq 0 (
107     echo 未找到curl工具，使用PowerShell下载 ...
```

```
108
109     echo 使用PowerShell下载Node.js ...
110     powershell -Command "& {[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12; Invoke-WebRequest -Uri
'https://nodejs.org/dist/v18.17.0/node-v18.17.0-x64.msi' -OutFile
'%tempDir%\nodejs_setup.msi'}"
111 ) else (
112     :: 使用curl下载Node.js安装程序
113     curl -L -o "%tempDir%\nodejs_setup.msi"
'https://nodejs.org/dist/v18.17.0/node-v18.17.0-x64.msi'
114 )
115
116 if not exist "%tempDir%\nodejs_setup.msi" (
117     echo 下载Node.js失败, 请手动访问 https://nodejs.org/ 下载并安装
118     start https://nodejs.org/
119     goto End
120 )
121
122 echo 下载完成, 开始安装Node.js ...
123 :: 静默安装Node.js
124 msixexec /i "%tempDir%\nodejs_setup.msi" /qn
125 if %errorlevel% neq 0 (
126     echo Node.js安装失败, 尝试手动安装 ...
127     start "" "%tempDir%\nodejs_setup.msi"
128     goto End
129 )
130
131 echo Node.js安装完成!
132 echo 请重新启动此脚本以继续操作 ...
133 goto End
134
135 :StartDevMode
136 echo 启动开发模式 ...
137
138 :: 检查并关闭占用端口的进程
139 call :CheckAndClosePort 3000
140 call :CheckAndClosePort 5173
141
142 :: 确保package.json中有相应的脚本命令
143 findstr /c:"\"server:dev\""" package.json >nul
144 if %errorlevel% neq 0 (
145     echo 警告: package.json 中未找到 "server:dev" 脚本。
146     echo 请确保项目配置正确。
147     goto End
148 )
149
150 findstr /c:"\"dev\""" package.json >nul
151 if %errorlevel% neq 0 (
152     echo 警告: package.json 中未找到 "dev" 脚本。
153     echo 请确保项目配置正确。
154     goto End
155 )
156
157 echo 1. 启动后端服务器 ...
158 start cmd /k "cd /d %~dp0 && npm run server:dev"
```

```
159 echo 2. 启动前端开发服务器 ...
160 start cmd /k "cd /d %~dp0 && npm run dev"
161 echo 服务已启动! 请在浏览器中打开 http://localhost:5173
162 goto End
163
164 :StartProdMode
165 echo 启动生产模式 ...
166
167 :: 检查并关闭占用端口的进程
168 call :CheckAndClosePort 3000
169
170 :: 确保package.json中有相应的脚本命令
171 findstr /c:"\"build\"" package.json >nul
172 if %errorlevel% neq 0 (
173     echo 警告: package.json 中未找到 "build" 脚本。
174     echo 请确保项目配置正确。
175     goto End
176 )
177
178 findstr /c:"\"start\"" package.json >nul
179 if %errorlevel% neq 0 (
180     echo 警告: package.json 中未找到 "start" 脚本。
181     echo 请确保项目配置正确。
182     goto End
183 )
184
185 echo 1. 构建前端应用 ...
186 call npm run build
187 if %errorlevel% neq 0 (
188     echo 构建失败, 请检查错误信息
189     goto End
190 )
191
192 echo 2. 启动整合服务器 ...
193 start cmd /k "cd /d %~dp0 && npm run start"
194
195 echo 服务已启动! 请在浏览器中打开 http://localhost:3000
196 goto End
197
198 :BuildProject
199 echo 执行项目构建 ...
200
201 :: 确保package.json中有相应的脚本命令
202 findstr /c:"\"build\"" package.json >nul
203 if %errorlevel% neq 0 (
204     echo 警告: package.json 中未找到 "build" 脚本。
205     echo 请确保项目配置正确。
206     goto End
207 )
208
209 call npm run build
210 if %errorlevel% neq 0 (
211     echo 构建失败, 请检查错误信息
212 ) else (
213     echo 构建完成! 可以通过 npm run start 启动服务
```

```
214 )
215 goto End
216
217 :UpdateDependencies
218 echo 更新项目依赖 ...
219 call npm update
220 if %errorlevel% neq 0 (
221     echo 依赖更新失败，请检查网络连接或手动运行 npm update
222 ) else (
223     echo 依赖更新完成!
224 )
225 goto End
226
227 :CheckAndClosePort
228 echo 检查端口 %1 占用情况 ...
229 FOR /F "tokens=5" %%P IN ('netstat -ano ^| findstr :%1 ^| findstr LISTENING')
DO (
230     echo 发现进程占用端口 %1: %%P, 尝试关闭 ...
231     taskkill /F /PID %%P 2>NUL
232     if !errorlevel! equ 0 (
233         echo 端口 %1 已释放
234     ) else (
235         echo 无法释放端口 %1, 可能需要手动关闭进程
236     )
237 )
238 goto :eof
239
240 :End
241 echo.
242 echo 按任意键继续 ...
243 pause >nul
```

```

1 <template>
2   <div class="green-web-app">
3     <!-- 标题和介绍 -->
4     <header class="app-header">
5       <div class="logo">
6         
7       </div>
8       <h1>GreenWeb <span>碳排放检测</span></h1>
9       <p>分析网站服务器能源使用和碳足迹，助力可持续互联网发展</p>
10    </header>
11
12    <!-- 输入区域 -->
13    <div class="input-section">
14      <div class="url-input-group">
15        <el-input
16          v-model="domain"
17          class="domain-input"
18          placeholder="输入网站域名（例如：example.com）"
19          @keyup.enter="checkCarbon"
20          :disabled="loading"
21        />
22
23        <!-- 浏览器选择下拉菜单 -->
24        <el-select v-model="selectedBrowser" placeholder="分析方式"
25          class="browser-select">
26          <el-option label="自动选择" value="auto"></el-option>
27          <el-option label="基础HTTP分析" value="basic"></el-option>
28          <el-option label="仅HTTP头分析" value="headers"></el-option>
29        </el-select>
30
31        <el-button
32          type="primary"
33          @click="checkCarbon"
34          :loading="loading"
35          class="analyze-button"
36        >
37          分析碳排放
38        </el-button>
39      </div>
40
41      <main class="main-content">
42        <div v-if="loading" class="loading-container">
43          <div class="earth-container">
44            <div class="earth"></div>
45          </div>
46          <p>正在分析碳排放数据 ... <br><small>(使用Axios测量性能，可能需要5-60秒)</small></p>
47        </div>
48
49        <div v-if="result && !loading" class="result-section">
50          <!-- 添加错误提示区域 -->
51          <div v-if="result.hasError" class="error-alerts">

```

```

52     <div class="error-alert">
53         <h3>注意：部分数据无法获取</h3>
54         <ul>
55             <li v-for="(error, index) in result.errorMessages"
:key="index">{{ error }}</li>
56         </ul>
57         <p>以下分析仅基于可获取的真实数据。</p>
58     </div>
59 </div>
60 <div class="result-summary">
61     <div class="summary-card" :class="result.isGreen ? 'green' :
'red'">
62         <div class="summary-header">
63             <div class="summary-title">碳排放评估结果</div>
64             <div class="summary-subtitle">{{ domain }}</div>
65         </div>
66         <div class="summary-content">
67             <div class="summary-section">
68                 <div class="total-emission">
69                     <div class="carbon-icon">
70                         <i :class="result.isGreen ? 'fa-solid fa-leaf fa-2x' :
'fa-solid fa-smog fa-2x'"></i>
71                     </div>
72                     <div class="carbon-details">
73                         <div class="total-title">此网页单次访问的碳排放</div>
74                         <div class="total-value">
75                             {{ safeToFixed(result.totalCarbonEmission, 4) ||
'0.0000' }} gCO2e
76                             <span v-if="result.estimatedData &&
result.estimatedData.includes('totalCarbonEmission')" class="estimated-data-
tag">估算</span>
77                         </div>
78                         <div class="total-breakdown">
79                             <div class="total-item">
80                                 <span class="total-label">月度碳排放:</span>
81                                 <span class="total-value">
82                                     {{ safeToFixed(result.monthlyCarbonEmission, 2) ||
'0.00' }} kgCO2e
83                                     <span v-if="result.estimatedData &&
result.estimatedData.includes('monthlyCarbonEmission')" class="estimated-
data-tag">估算</span>
84                                 </span>
85                                 </div>
86                                 <div class="total-info">
87                                     基于{{
globalConstants.averageMonthlyVisits.toLocaleString() }}次月访问量
88                                     <span class="estimated-data-tag">估算</span>
89                                 </div>
90                                 <div class="total-item">
91                                     <span class="total-label">年度碳排放:</span>
92                                     <span class="total-value">
93                                         {{ safeToFixed(result.annualCarbonEmission, 2) ||
'0.00' }} kgCO2e
94                                         <span v-if="result.estimatedData &&
result.estimatedData.includes('annualCarbonEmission')" class="estimated-

```

```
data-tag">估算</span>
```

```
95         </span>
```

```
96     </div>
```

```
97     <div class="total-info">
```

```
98         相当于种植{{ Math.round((result.annualCarbonEmission  
|| 0) / (globalConstants.treeCO2PerYear || 1)) }}棵树才能抵消
```

```
99         <span class="estimated-data-tag">估算</span>
```

```
100     </div>
```

```
101 </div>
```

```
102 </div>
```

```
103 </div>
```

```
104 </div>
```

```
105
```

```
106     <div class="summary-section">
```

```
107         <div class="energy-breakdown">
```

```
108             <div class="energy-title">能源消耗明细</div>
```

```
109             <div class="energy-items">
```

```
110                 <div class="energy-item">
```

```
111                     <div class="energy-label">数据中心</div>
```

```
112                     <div class="energy-value">
```

```
113                         {{ safeToFixed(result.dataCenterEnergy || 0, 4) }}
```

```
Wh
```

```
114         <span v-if="result.estimatedData &&
```

```
result.estimatedData.includes('dataCenterEnergy')" class="estimated-data-  
tag">估算</span>
```

```
115     </div>
```

```
116 </div>
```

```
117 <div class="energy-item">
```

```
118     <div class="energy-label">网络传输</div>
```

```
119     <div class="energy-value">
```

```
120         {{ safeToFixed(result.transmissionEnergy || 0, 4) }}
```

```
Wh
```

```
121     <span v-if="result.estimatedData &&
```

```
result.estimatedData.includes('transmissionEnergy')" class="estimated-data-  
tag">估算</span>
```

```
122 </div>
```

```
123 </div>
```

```
124 <div class="energy-item">
```

```
125     <div class="energy-label">用户设备</div>
```

```
126     <div class="energy-value">
```

```
127         {{ safeToFixed(result.deviceEnergy || 0, 4) }} Wh
```

```
128     <span v-if="result.estimatedData &&
```

```
result.estimatedData.includes('deviceEnergy')" class="estimated-data-tag">估  
算</span>
```

```
129 </div>
```

```
130 </div>
```

```
131 </div>
```

```
132 </div>
```

```
133 </div>
```

```
134 </div>
```

```
135 </div>
```

```
136
```

```
137     <div class="detail-card">
```

```
138         <div class="detail-header">
```

```
139             <div class="detail-title">绿色托管信息</div>
```



```

140         </div>
141         <div class="detail-content">
142             <div class="provider-info">
143                 <div class="provider-icon">
144                     <i :class="result.isGreen ? 'fa-solid fa-solar-panel fa-
2x' : 'fa-solid fa-industry fa-2x'"></i>
145                 </div>
146                 <div class="provider-details">
147                     <div class="provider-name">{{ result.providerName || '未知
服务商' }}</div>
148                     <div class="provider-stats">
149                         <div class="provider-item">
150                             <span class="provider-label">可再生能源使用率:</span>
151                             <span class="provider-value">
152                                 {{ result.renewablePercentage !== null ?
result.renewablePercentage + '%' : '未知' }}
153                                 <span v-if="result.estimatedData &&
result.estimatedData.includes('renewablePercentage')" class="estimated-data-
tag">估算</span>
154                             </span>
155                         </div>
156                         <div class="provider-item">
157                             <span class="provider-label">PUE:</span>
158                             <span class="provider-value">
159                                 {{ result.pue || '未知' }}
160                                 <span v-if="result.estimatedData &&
result.estimatedData.includes('pue')" class="estimated-data-tag">估算</span>
161                             </span>
162                         </div>
163                         <div class="provider-item">
164                             <span class="provider-label">服务器位置:</span>
165                             <span class="provider-value">{{ result.region || '未知'
}}, {{ result.country || '未知' }}</span>
166                         </div>
167                         <div class="provider-item">
168                             <span class="provider-label">绿色托管等级:</span>
169                             <span class="provider-value" :class="{ 'green-value':
result.isGreen}">
170                                 {{ result.isGreen ? '环保托管' : '标准托管' }}
171                             </span>
172                         </div>
173                         <div class="provider-item">
174                             <span class="provider-label">全球碳强度:</span>
175                             <span class="provider-value">{{
globalConstants.averageCarbonIntensity }} gCO2e/kWh</span>
176                         </div>
177                     </div>
178                 </div>
179             </div>
180             <div v-if="result.isGreen" class="green-hosting-info">
181                 <p>该网站使用环保托管服务，有助于减少互联网碳足迹。</p>
182             </div>
183             <div v-else class="green-hosting-info">
184                 <p>建议考虑使用更环保的托管服务，以减少碳排放。</p>
185             </div>

```

```

186         </div>
187     </div>
188 </div>
189
190 <div class="result-grid">
191
192     <!-- 现代化的结果卡片网格布局 -->
193     <div class="result-card-container">
194         <!-- 第一行卡片 - 重要信息 -->
195         <div class="result-row">
196             <div class="result-card energy-source major-card">
197                 <div class="card-header">
198                     <h3>可再生能源使用情况</h3>
199                     <div class="card-icon">
200                         <el-icon><DataBoard /></el-icon>
201                     </div>
202                 </div>
203                 <div v-if="result.renewablePercentage !== null"
class="energy-chart">
204                     <div class="donut-chart">
205                         <div class="donut-hole">{{ result.renewablePercentage
}}%</div>
206                         <div class="donut-ring">
207                             <div class="renewable" :style="`--percent:
${result.renewablePercentage}%`"></div>
208                             </div>
209                         </div>
210                         <div class="chart-legend">
211                             <div class="legend-item">
212                                 <div class="legend-color renewable"></div>
213                                 <span>可再生能源 ({{ result.renewablePercentage }}%)
</span>
214                             </div>
215                             <div class="legend-item">
216                                 <div class="legend-color fossil"></div>
217                                 <span>化石能源 ({{ 100 - result.renewablePercentage
}}%)</span>
218                             </div>
219                         </div>
220                     </div>
221                     <div v-else class="data-unavailable">
222                         无法获取可再生能源使用比例
223                     </div>
224                     <div class="card-summary">
225                         <div class="summary-badge" :class="result.isGreen ?
'green-badge' : 'standard-badge'">
226                             {{ result.isGreen ? '环保托管' : '标准托管' }}
227                         </div>
228                         <p>{{ result.isGreen ? '此网站使用环保友好的托管服务' : '建议使
用更环保的托管服务' }}</p>
229                     </div>
230                 </div>
231
232                 <div class="result-card carbon-map major-card">
233                     <div class="card-header">

```

```

234         <h3>碳排放评分</h3>
235         <div class="card-icon">
236             <el-icon><PieChart /></el-icon>
237         </div>
238     </div>
239     <div v-if="result.totalCarbonEmission !== null">
240         <div class="score-container" v-
241 if="result.carbonFootprintScore || result.energyEfficiencyScore">
242             <div class="carbon-score-card"
243 :class="getCarbonScoreClass(result.carbonFootprintScore)">
244                 <div class="score-circle">{{
245 Math.round(result.carbonFootprintScore || 50) }}</div>
246                 <div class="score-label">碳足迹评分</div>
247                 <div class="score-desc">{{
248 getCarbonScoreDesc(result.carbonFootprintScore) }}</div>
249                 </div>
250                 <div class="carbon-score-card"
251 :class="getEnergyScoreClass(result.energyEfficiencyScore)">
252                     <div class="score-circle">{{
253 Math.round(result.energyEfficiencyScore || 50) }}</div>
254                     <div class="score-label">能源效率评分</div>
255                     <div class="score-desc">{{
256 getEnergyScoreDesc(result.energyEfficiencyScore) }}</div>
257                     </div>
258                 </div>
259                 <div class="total-emission-highlight">
260                     <div class="highlight-value">{{
261 safeToFixed(result.totalCarbonEmission, 4) }} <span>gCO2e</span></div>
262                     <div class="highlight-label">单次访问碳排放</div>
263                 </div>
264             </div>
265             <div v-else class="data-unavailable">
266                 无法获取碳排放分析数据
267             </div>
268         </div>
269     </div>
270
271     <!-- 第二行卡片 - 详细数据 -->
272     <div class="result-row">
273         <div class="result-card site-info">
274             <div class="card-header">
275                 <h3>网站基础信息</h3>
276                 <div class="card-icon">
277                     <el-icon><Connection /></el-icon>
278                 </div>
279             </div>
280             <div class="details">
281                 <div class="detail-item">
282                     <span class="detail-label">服务商:</span>
283                     <span class="detail-value highlight-text">{{
284 result.provider && typeof result.provider === 'string' ?
285 result.provider.toUpperCase() : '未知' }}</span>
286                 </div>
287                 <div class="detail-item">
288                     <span class="detail-label">服务器位置:</span>

```

```

279         <span class="detail-value">{{ result.region || '未知' }},
    {{ result.country || '未知' }}</span>
280     </div>
281     <div class="detail-item">
282         <span class="detail-label">页面大小:</span>
283         <span class="detail-value">{{ result.pageSize ?
    `${result.pageSize} KB` : '无法获取' }}</span>
284     </div>
285     <div class="detail-item">
286         <span class="detail-label">能源强度:</span>
287         <span class="detail-value">{{ result.energyIntensity ===
    null ? `${safeToFixed(result.energyIntensity, 2)} kWh/GB` : '无法获取' }}
    </span>
288     </div>
289     <div class="detail-item">
290         <span class="detail-label">PUE值:</span>
291         <span class="detail-value" :class="{ 'highlight-text':
    result.pue && result.pue < 1.5}">
292             {{ result.pue || '未知' }}
293             <span v-if="result.estimatedData &&
    result.estimatedData.includes('pue')" class="estimated-data-tag">估算</span>
294         </span>
295     </div>
296 </div>
297 </div>
298
299     <div class="result-card energy-consumption">
300         <div class="card-header">
301             <h3>能源消耗分析</h3>
302             <div class="card-icon">
303                 <el-icon><Connection /></el-icon>
304             </div>
305         </div>
306         <!-- 显示设置，确保始终显示能源消耗网格 -->
307         <div class="energy-consumption-grid">
308             <div class="energy-consumption-item">
309                 <div class="consumption-icon data-center-icon"></div>
310                 <div class="consumption-value">{{
    formatEnergyValue(result.dataCenterEnergy) }}</div>
311                 <div class="consumption-label">数据中心能耗</div>
312             </div>
313             <div class="energy-consumption-item">
314                 <div class="consumption-icon network-icon"></div>
315                 <div class="consumption-value">{{
    formatEnergyValue(result.transmissionEnergy) }}</div>
316                 <div class="consumption-label">网络传输能耗</div>
317             </div>
318             <div class="energy-consumption-item">
319                 <div class="consumption-icon device-icon"></div>
320                 <div class="consumption-value">{{
    formatEnergyValue(result.deviceEnergy) }}</div>
321                 <div class="consumption-label">用户设备能耗</div>
322             </div>
323         </div>

```

```

324         <div v-if="!result.dataCenterEnergy &&
!result.transmissionEnergy && !result.deviceEnergy" class="energy-note">
325             <p>注意：能源消耗数据可能很低或无法获取。这可能是由于页面太小或性能分
析不完整。</p>
326         </div>
327     </div>
328
329     <div class="result-card carbon-impact">
330         <div class="card-header">
331             <h3>碳排放详情</h3>
332             <div class="card-icon">
333                 <el-icon><PieChart /></el-icon>
334             </div>
335         </div>
336         <div v-if="result.totalCarbonEmission !== null"
class="carbon-stats">
337             <div class="carbon-stat-item">
338                 <span class="stat-label">数据中心碳排放:</span>
339                 <span class="stat-value">{{
formatNumber(result.dataTransferCarbon) }} gCO2e</span>
340             </div>
341             <div class="carbon-stat-item">
342                 <span class="stat-label">网络传输碳排放:</span>
343                 <span class="stat-value">{{
formatNumber(result.networkCarbon) }} gCO2e</span>
344             </div>
345             <div class="carbon-stat-item">
346                 <span class="stat-label">客户端碳排放:</span>
347                 <span class="stat-value">{{
formatNumber(result.clientCarbon) }} gCO2e</span>
348             </div>
349             <div class="carbon-annual-impact">
350                 <div class="impact-title">年度碳排放估计</div>
351                 <div class="impact-value">{{
safeToFixed(result.annualCarbonEmission, 2) }} kgCO2e</div>
352                 <div class="impact-equivalent">
353                     相当于种植 <span class="highlight-text">{{
Math.round((result.annualCarbonEmission || 0) /
globalConstants.treeCO2PerYear) }}</span> 棵树才能抵消
354                 </div>
355             </div>
356         </div>
357         <div v-else class="data-unavailable">
358             无法获取碳排放分析数据
359         </div>
360     </div>
361 </div>
362
363     <!-- 第三行卡片 - 性能和建议 -->
364     <div class="result-row">
365         <div class="result-card performance wide-card">
366             <div class="card-header">
367                 <h3>性能指标</h3>
368                 <div class="card-icon">
369                     <el-icon><Timer /></el-icon>

```

```

370         </div>
371     </div>
372     <div v-if="result.performance &&
result.performance.measurable !== false" class="performance-metrics-v2">
373         <div class="metrics-grid">
374             <div
375                 v-for="(value, metric, index) in result.performance"
376                 :key="metric"
377                 class="metric-card"
378                 v-show="!['measuredBy', 'statusCode', 'measurable',
'requestCount', 'domainCount'].includes(metric)"
379                 >
380                 <div class="metric-visual">
381                     <div class="metric-circle"
: class="getMetricGrade(metric, value)">
382                         {{ formatMetricValue(metric, value) }}
383                     </div>
384                 </div>
385                 <div class="metric-info">
386                     <div class="metric-name">{{ formatMetricName(metric)
}}</div>
387                     <div class="metric-description">{{
getMetricDescription(metric) }}</div>
388                 </div>
389             </div>
390         </div>
391
392         <div v-if="result.performance.requestCount ||
result.performance.domainCount" class="network-stats">
393             <div class="network-title">网络请求数据</div>
394             <div class="network-grid">
395                 <div class="network-item" v-
if="result.performance.requestCount">
396                     <div class="network-value">{{
result.performance.requestCount }}</div>
397                     <div class="network-label">请求数量</div>
398                 </div>
399                 <div class="network-item" v-
if="result.performance.domainCount">
400                     <div class="network-value">{{
result.performance.domainCount }}</div>
401                     <div class="network-label">域名数量</div>
402                 </div>
403             </div>
404         </div>
405
406         <div v-if="result.performance.measuredBy" class="data-
source-info">
407             测量工具: {{ result.performance.measuredBy }}
408         </div>
409     </div>
410     <div v-else class="data-unavailable">
411         无法获取性能指标数据<br>
412         <small v-if="result.performance &&
result.performance.error">{{ result.performance.error }}</small>

```

```

413         </div>
414     </div>
415
416     <div class="result-card suggestions wide-card">
417         <div class="card-header">
418             <h3>优化建议</h3>
419             <div class="card-icon">
420                 <el-icon><Opportunity /></el-icon>
421             </div>
422         </div>
423         <div v-if="result.suggestions && result.suggestions.length >
0" class="suggestions-container">
424             <div class="suggestions-list">
425                 <div
426                     v-for="(suggestion, index) in result.suggestions"
427                     :key="index"
428                     class="suggestion-card"
429                 >
430                     <div class="suggestion-header">
431                         <div class="suggestion-icon">
432                             <el-icon><Opportunity /></el-icon>
433                         </div>
434                         <div class="suggestion-title">优化建议 #{{ index + 1
}}</div>
435                     </div>
436                     <div class="suggestion-content">
437                         {{ suggestion }}
438                     </div>
439                     <div class="suggestion-impact"
: class="getImpactClass(index)">
440                         {{ getImpactLevel(index) }}
441                     </div>
442                 </div>
443             </div>
444             <div v-else class="data-unavailable">
445                 无法生成具体优化建议，请确保网站可访问
446             </div>
447         </div>
448     </div>
449
450
451     <!-- 绿色托管信息卡片 -->
452     <div v-if="carbonResult" class="result-card green-hosting wide-
card">
453         <h3>绿色托管信息</h3>
454         <div class="detail-content">
455             <div class="provider-info">
456                 <div><strong>提供商: </strong> {{ carbonResult.provider ||
'未知' }}</div>
457                 <div><strong>服务器位置: </strong> {{ carbonResult.region ||
'未知' }}</div>
458             </div>
459             <div class="provider-stats">
460                 <div><strong>可再生能源使用率: </strong> <span :class="{
'green-value': carbonResult.renewablePercentage > 50 }">{{

```



```

carbonResult.renewablePercentage || 0 }%</span></div>
461     <div><strong>PUE值: </strong> <span :class="{ 'green-
value': carbonResult.pue < 1.5 }">{{ carbonResult.pue || '未知' }}</span>
</div>
462     <div><strong>绿色托管级别: </strong> <span :class="{
'green-value': isGreenHosting }">{{ isGreenHosting ? '环保托管' : '标准托管' }}
</span></div>
463     </div>
464
465     <div class="green-hosting-info">
466         <p v-if="isGreenHosting">此网站由环保友好的主机提供商托管，使
用高比例的可再生能源和高效的数据中心设施。</p>
467         <p v-else>此网站使用标准托管服务，可以通过迁移到使用更多可再生能
源的提供商来降低碳足迹。</p>
468         <p>全球服务器平均碳强度: {{ globalCarbonIntensity || '未知'
}} gCO2/kWh</p>
469     </div>
470 </div>
471 </div>
472 </div>
473 </div>
474 </div>
475 </div>
476 </main>
477
478 <footer class="footer">
479     <p>碳排放计算采用HTTP请求测量性能指标</p>
480     <p>结果仅供参考，不作为任何法律依据</p>
481     <p class="copyright">© {{ new Date().getFullYear() }} GreenWeb网站碳中和
检测平台 | GreenSide团队</p>
482 </footer>
483 </div>
484 </template>
485
486 <script>
487 import { ref, onMounted, reactive, nextTick, computed } from 'vue'
488 import { Search, Connection, DataBoard, PieChart, Timer, Opportunity, Check,
Close } from '@element-plus/icons-vue'
489 import {
490     analyzeWebsite,
491     measurePerformance,
492     analyzeCarbonEmission,
493     getServerLocation,
494     analyzeProvider
495 } from './services/websiteAnalyzer'
496
497 export default {
498     name: 'App',
499     components: {
500         Search,
501         Connection,
502         DataBoard,
503         PieChart,
504         Timer,
505         Opportunity,

```



```

506     Check,
507     Close
508 },
509
510 setup() {
511     // 引用和状态
512     const domain = ref('')
513     const loading = ref(false)
514     const result = ref(null)
515     const selectedBrowser = ref('auto')
516
517     // 全局常量
518     const globalConstants = reactive({
519         averageEnergyConsumption: 1.8, // kWh/GB
520         averageTransmissionPerGB: 0.06, // kWh/GB
521         averageDevicePerGB: 0.15, // kWh/GB
522         averageCarbonIntensity: 442, // gCO2e/kWh (全球平均)
523         greenEnergyCarbonIntensity: 50, // gCO2e/kWh (绿色能源)
524         averageMonthlyVisits: 10000, // 每月平均访问量
525         cachingEfficiency: 0.3, // 缓存效率 (30%)
526         treeCO2PerYear: 22, // 每棵树每年吸收的CO2量 (kg)
527         bestPUE: 1.1, // 最佳数据中心PUE
528         averagePUE: 1.67, // 平均数据中心PUE
529         greenEnergyThreshold: 80 // 绿色能源使用比例阈值
530     })
531
532     // 碳排放分析
533     async function checkCarbon() {
534         if (!domain.value) {
535             return
536         }
537
538         loading.value = true
539         result.value = null
540
541         try {
542             // 分析网站性能和碳排放
543             console.log(`开始分析网站: ${domain.value} (使用${selectedBrowser.value
544 ≡≡≡ 'auto' ? '自动选择浏览器' : selectedBrowser.value})`)
545
546             // 获取使用的浏览器参数
547             const browserOption = selectedBrowser.value
548
549             let performanceResult = null
550             let locationResult = null
551             let providerResult = null
552             let carbonResult = null
553             let hasError = false
554             let errorMessages = []
555
556             // 1. 测量性能指标
557             try {
558                 performanceResult = await measurePerformance(domain.value,

```

```
559         if (performanceResult.error) {
560             console.warn('性能测量警告:', performanceResult.error)
561             errorMessages.push(`性能指标: ${performanceResult.error}`)
562             hasError = true
563         }
564     } catch (perfError) {
565         console.error('性能测量失败:', perfError)
566         errorMessages.push(`性能指标: ${perfError.message} || '无法获取'`)
567         hasError = true
568     }
569
570     // 2. 获取服务器位置
571     try {
572         locationResult = await getServerLocation(domain.value)
573
574         if (locationResult.error) {
575             console.warn('位置获取警告:', locationResult.error)
576             errorMessages.push(`服务器位置: ${locationResult.error}`)
577             hasError = true
578         }
579     } catch (locError) {
580         console.error('位置获取失败:', locError)
581         errorMessages.push(`服务器位置: ${locError.message} || '无法获取'`)
582         hasError = true
583     }
584
585     // 3. 分析服务提供商
586     try {
587         providerResult = await analyzeProvider(domain.value)
588
589         if (providerResult.error) {
590             console.warn('提供商分析警告:', providerResult.error)
591             errorMessages.push(`服务提供商: ${providerResult.error}`)
592             hasError = true
593         }
594     } catch (provError) {
595         console.error('提供商分析失败:', provError)
596         errorMessages.push(`服务提供商: ${provError.message} || '无法获取'`)
597         hasError = true
598     }
599
600     // 4. 计算碳排放 - 只有在有足够的真实数据时进行
601     if (performanceResult && !performanceResult.error && locationResult
602         && !locationResult.error) {
603         try {
604             const carbonParams = {
605                 pageSize: performanceResult.pageSize ||
606                 (performanceResult.performance?.pageSize) || 0,
607                 country: locationResult?.country || 'unknown',
608                 requestCount: performanceResult.requests ||
609                 (performanceResult.performance?.requestCount) || 1,
610                 domainCount: performanceResult.domainCount ||
611                 (performanceResult.performance?.domainCount) || 1,
612                 // 添加必要的参数
613                 renewablePercentage: providerResult?.renewablePercentage || 0,
```

```

610         pue: providerResult?.pue || 1.67, // 使用平均PUE
611         responseTime: performanceResult.responseTime ||
(performanceResult.performance?.responseTime) || 0,
612         hasCompression: performanceResult.headers?.supportsCompression
|| false,
613         resourceStats:
JSON.stringify(performanceResult.performance?.resourceStats || {}),
614         cacheControl: performanceResult.headers?.cacheControl || ''
615     }
616
617     // 确保主要参数有值
618     if (carbonParams.pageSize ≤ 0) {
619         throw new Error('页面大小不能为零或负值');
620     }
621
622     carbonResult = await analyzeCarbonEmission(carbonParams)
623
624     if (carbonResult.error) {
625         console.warn('碳排放计算警告:', carbonResult.error)
626         errorMessages.push(`碳排放计算: ${carbonResult.error}`)
627         hasError = true
628     }
629 } catch (carbonError) {
630     console.error('碳排放计算失败:', carbonError)
631     errorMessages.push(`碳排放计算: ${carbonError.message} || '无法获
取'`)
632     hasError = true
633 }
634 } else {
635     console.warn('缺少性能或位置数据, 无法计算碳排放')
636     errorMessages.push('缺少性能或位置数据, 无法计算碳排放')
637     hasError = true
638 }
639
640 // 初始化结果对象 - 只包含真实数据
641 result.value = {
642     hasError,
643     errorMessages,
644     onlyRealData: true, // 标记只显示真实数据
645     provider: providerResult?.provider || 'unknown',
646     isGreen: false,
647     totalCarbonEmission: null,
648     monthlyCarbonEmission: null,
649     dataCenterEnergy: 0.0045, // 默认能源消耗值 (kWh)
650     transmissionEnergy: 0.0012, // 默认网络传输能耗 (kWh)
651     deviceEnergy: 0.0032, // 默认设备能耗 (kWh)
652     suggestions: [
653         '使用绿色托管提供商可以减少碳足迹, 选择使用可再生能源的服务商',
654         '优化JavaScript和CSS文件, 减少代码体积, 提高页面加载速度',
655         '将静态资源部署到CDN, 减少服务器负载和网络传输距离',
656         '优化和压缩图片, 减少页面总体积和加载时间',
657         '实施有效的浏览器缓存策略, 减少重复资源的传输'
658     ]
659 }
660

```

```

661 // 合并各种结果 - 只添加真实测量的数据
662 if (performanceResult && !performanceResult.error) {
663     result.value = {
664         ... result.value,
665         performance: performanceResult.performance,
666         pageSize: performanceResult.pageSize ||
performanceResult.performance?.pageSize
667     }
668 }
669
670 if (locationResult && !locationResult.error) {
671     result.value = {
672         ... result.value,
673         country: locationResult.country,
674         region: locationResult.region,
675         coordinates: locationResult.coordinates
676     }
677 }
678
679 if (providerResult && !providerResult.error) {
680     result.value = {
681         ... result.value,
682         provider: providerResult.provider || 'unknown',
683         providerName: providerResult.providerName || '未知提供商',
684         renewablePercentage: providerResult.renewablePercentage,
685         pue: providerResult.pue,
686         isGreen: providerResult.renewablePercentage ≥
globalConstants.greenEnergyThreshold
687     }
688 }
689
690 if (carbonResult && !carbonResult.error) {
691     result.value = {
692         ... result.value,
693         ... carbonResult,
694         // 明确标记估算数据
695         estimatedData: carbonResult.dataSourceInfo?.estimatedValues ||
[]
696     }
697 }
698
699 console.log('分析完成:', result.value)
700 } catch (error) {
701     console.error('分析过程中出错:', error)
702     result.value = {
703         error: error.message,
704         isGreen: false,
705         totalCarbonEmission: null,
706         monthlyCarbonEmission: null,
707         annualCarbonEmission: null,
708         performance: {
709             measurable: false,
710             error: error.message
711         },
712         provider: 'unknown',

```

```

713         renewablePercentage: null,
714         suggestions: [],
715         region: null,
716         country: null,
717         coordinates: null,
718         dataTransferCarbon: null,
719         networkCarbon: null,
720         clientCarbon: null,
721         dataCenterEnergy: null,
722         transmissionEnergy: null,
723         deviceEnergy: null,
724         hasError: true,
725         errorMessages: [error.message]
726     }
727     } finally {
728         loading.value = false
729     }
730 }
731
732 // 监听窗口大小变化
733 onMounted(() => {
734     console.log('应用已挂载')
735 })
736
737 const isGreenHosting = computed(() => {
738     if (!carbonResult.value) return false;
739     const renewable = carbonResult.value.renewablePercentage || 0;
740     const pue = carbonResult.value.pue || 2.0;
741     return renewable > 50 && pue < 1.5;
742 });
743
744 const globalCarbonIntensity = computed(() => {
745     // 全球平均碳强度, 单位: gCO2/kWh
746     return 475;
747 });
748
749 return {
750     domain,
751     loading,
752     result,
753     globalConstants,
754     selectedBrowser,
755     checkCarbon,
756     isGreenHosting,
757     globalCarbonIntensity
758 }
759 },
760
761 methods: {
762     // 格式化性能指标名称
763     formatMetricName(metric) {
764         const metricNames = {
765             fcp: 'First Contentful Paint',
766             lcp: 'Largest Contentful Paint',
767             cls: 'Cumulative Layout Shift',

```

```

768         fid: 'First Input Delay',
769         ttfb: 'Time to First Byte',
770         responseTime: 'Response Time',
771         pageSize: 'Page Size',
772         totalResourceSize: 'Total Resource Size',
773         requestCount: 'Request Count',
774         domainCount: 'Domain Count',
775         usesHttps: 'HTTPS Usage',
776         serverType: 'Server Type',
777         supportsCompression: 'Compression Support',
778         supportsCaching: 'Caching Support',
779         usesCdn: 'CDN Usage',
780         cdnProvider: 'CDN Provider',
781         resourceStats: 'Resource Statistics',
782         securityScore: 'Security Score',
783         energyConsumption: 'Energy Consumption',
784         dataCenterEnergy: 'Data Center Energy',
785         transmissionEnergy: 'Transmission Energy',
786         deviceEnergy: 'Device Energy',
787         carbonEmission: 'Carbon Emission',
788         annualCarbonEmission: 'Annual Carbon Emission',
789         renewablePercentage: 'Renewable Energy Percentage',
790         pue: 'Power Usage Effectiveness'
791     }
792
793     return metricNames[metric] || metric
794 },
795
796 // 格式化性能指标值
797 formatMetricValue(metric, value) {
798     if (value === null || value === undefined) {
799         return '无法获取';
800     }
801
802     // 时间相关指标
803     if (metric === 'fcp' || metric === 'lcp') {
804         return `${this.safeToFixed(value, 2)}s`;
805     } else if (metric === 'fid' || metric === 'ttfb' || metric ===
'responseTime') {
806         return `${Math.round(value)}ms`;
807     }
808     // 布局偏移指标
809     else if (metric === 'cls') {
810         return this.safeToFixed(value, 3);
811     }
812     // 大小相关指标
813     else if (metric === 'pageSize' || metric === 'totalResourceSize') {
814         if (value < 1024) {
815             return `${this.safeToFixed(value, 2)} B`;
816         } else if (value < 1024 * 1024) {
817             return `${this.safeToFixed(value / 1024, 2)} KB`;
818         } else {
819             return `${this.safeToFixed(value / (1024 * 1024), 2)} MB`;
820         }
821     }

```

```
822 // 计数指标
823 else if (metric === 'requestCount' || metric === 'domainCount') {
824     return Math.round(value);
825 }
826 // 布尔指标
827 else if (metric === 'usesHttps' || metric === 'supportsCompression' ||
metric === 'supportsCaching' || metric === 'usesCdn') {
828     return value ? '是' : '否';
829 }
830 // 能源指标
831 else if (metric === 'energyConsumption' || metric ===
'dataCenterEnergy' || metric === 'transmissionEnergy' || metric ===
'deviceEnergy') {
832     return `${this.safeToFixed(value, 4)} Wh`;
833 }
834 // 碳排放指标
835 else if (metric === 'carbonEmission') {
836     return `${this.safeToFixed(value, 2)} gCO2e`;
837 }
838 else if (metric === 'annualCarbonEmission') {
839     return `${this.safeToFixed(value, 2)} kgCO2e`;
840 }
841 // 百分比指标
842 else if (metric === 'renewablePercentage') {
843     return `${this.safeToFixed(value, 1)}%`;
844 }
845 // PUE指标
846 else if (metric === 'pue') {
847     return this.safeToFixed(value, 2);
848 }
849 // 评分指标
850 else if (metric === 'securityScore') {
851     return `${Math.round(value)}/100`;
852 }
853 // 默认处理
854 else {
855     return value.toString();
856 }
857 },
858
859 // 获取性能指标评级
860 getMetricGrade(metric, value) {
861     if (value === null || value === undefined) {
862         return 'unknown'
863     }
864
865     // 基于Web核心指标标准进行性能评级
866     if (metric === 'fcp') {
867         return value < 1.8 ? 'good' : value < 3 ? 'average' : 'poor'
868     } else if (metric === 'lcp') {
869         return value < 2.5 ? 'good' : value < 4 ? 'average' : 'poor'
870     } else if (metric === 'cls') {
871         return value < 0.1 ? 'good' : value < 0.25 ? 'average' : 'poor'
872     } else if (metric === 'fid') {
873         return value < 100 ? 'good' : value < 300 ? 'average' : 'poor'
```

```
874     } else if (metric === 'ttfb') {
875         return value < 200 ? 'good' : value < 500 ? 'average' : 'poor'
876     }
877     return 'average'
878 },
879
880 // 获取进度条样式
881 getProgressStyle(metric, value) {
882     if (value === null || value === undefined) {
883         return { width: `0%` }
884     }
885
886     let percentage
887
888     if (metric === 'cls') {
889         // CLS是越小越好, 0.1是理想值, 0.25是最大可接受值
890         percentage = Math.min(100, (value / 0.25) * 100)
891     } else if (metric === 'fcp') {
892         // FCP在1秒以内为理想, 3秒以上为较差
893         percentage = Math.min(100, (value / 3) * 100)
894     } else if (metric === 'lcp') {
895         // LCP在2.5秒以内为理想, 4秒以上为较差
896         percentage = Math.min(100, (value / 4) * 100)
897     } else if (metric === 'fid') {
898         // FID在100ms以内为理想, 300ms以上为较差
899         percentage = Math.min(100, (value / 300) * 100)
900     } else {
901         // 默认值
902         percentage = 50
903     }
904
905     return { width: `${percentage}%` }
906 },
907
908 // 获取进度条类
909 getProgressClass(metric, value) {
910     return this.getMetricGrade(metric, value)
911 },
912
913 // 格式化数字
914 formatNumber(value) {
915     if (value === null || value === undefined) return '无法获取'
916     return this.safeToFixed(parseFloat(value), 2)
917 },
918
919 // 获取碳排放分数样式类
920 getCarbonScoreClass(score) {
921     if (!score) return 'average'
922     score = parseInt(score)
923     if (score ≤ 30) return 'excellent'
924     if (score ≤ 50) return 'good'
925     if (score ≤ 70) return 'moderate'
926     return 'poor'
927 },
928
```



```

929 // 获取能源效率评分样式类
930 getEnergyScoreClass(score) {
931     if (!score) return 'average'
932     score = parseInt(score)
933     if (score ≥ 80) return 'excellent'
934     if (score ≥ 60) return 'good'
935     if (score ≥ 40) return 'moderate'
936     return 'poor'
937 },
938
939 // 获取指标样式类
940 getMetricClass(metric, value) {
941     return this.getMetricGrade(metric, value)
942 },
943
944 // 判断是否为时间指标
945 isTimingMetric(metric) {
946     return ['fcp', 'lcp', 'fid', 'ttfb'].includes(metric)
947 },
948
949 // 获取进度条样式
950 getProgressBarStyle(metric, value) {
951     if (value === null || value === undefined) {
952         return { width: '0%' }
953     }
954
955     let percentage = 0
956     let grade = this.getMetricGrade(metric, value)
957
958     if (metric === 'fcp' || metric === 'lcp') {
959         // 转换为毫秒
960         const ms = value * 1000
961         percentage = Math.min(100, Math.max(10, (ms / 5000) * 100))
962     } else if (metric === 'cls') {
963         percentage = Math.min(100, Math.max(10, (value / 0.25) * 100))
964     } else if (metric === 'fid' || metric === 'ttfb') {
965         percentage = Math.min(100, Math.max(10, (value / 1000) * 100))
966     }
967
968     return {
969         width: `${percentage}%`,
970         backgroundColor: grade === 'good' ? 'var(--color-good)' :
971             grade === 'average' ? 'var(--color-moderate)' :
972             'var(--color-poor)'
973     }
974 },
975
976 // 获取安全评分样式类
977 getSecurityScoreClass(score) {
978     if (!score && score !== 0) return 'poor'
979     if (score ≥ 80) return 'excellent'
980     if (score ≥ 60) return 'good'
981     if (score ≥ 40) return 'moderate'
982     return 'poor'
983 },

```

```
984
985 // 格式化安全头信息
986 formatSecurityHeader(header) {
987     const headerNames = {
988         'strict-transport-security': 'HTTP严格传输安全',
989         'content-security-policy': '内容安全策略',
990         'x-content-type-options': '内容类型选项',
991         'x-frame-options': '框架选项',
992         'x-xss-protection': 'XSS保护'
993     }
994
995     return headerNames[header] || header
996 },
997
998 // 获取碳排放等级描述
999 getCarbonScoreDesc(score) {
1000     if (!score) return '评估中'
1001     score = parseInt(score)
1002     if (score ≤ 30) return '非常环保'
1003     if (score ≤ 50) return '较为环保'
1004     if (score ≤ 70) return '一般水平'
1005     return '需要改进'
1006 },
1007
1008 // 获取能源效率评分描述
1009 getEnergyScoreDesc(score) {
1010     if (!score) return '评估中'
1011     score = parseInt(score)
1012     if (score ≥ 80) return '高效利用'
1013     if (score ≥ 60) return '良好效率'
1014     if (score ≥ 40) return '一般效率'
1015     return '低效利用'
1016 },
1017
1018 // 获取碳排放等价物描述
1019 getCarbonEquivalent(emission) {
1020     if (!emission) return '无法计算'
1021
1022     if (emission < 0.5) {
1023         return '少于一封电子邮件的碳排放'
1024     } else if (emission < 1) {
1025         return '约等于一封电子邮件的碳排放'
1026     } else if (emission < 3) {
1027         return '约等于浏览一个简单网页'
1028     } else if (emission < 10) {
1029         return '约等于观看1分钟高清视频'
1030     } else {
1031         return `约等于观看${Math.round(emission/10)}分钟高清视频`
1032     }
1033 },
1034
1035 // 格式化能源消耗值
1036 formatEnergy(value) {
1037     if (!value) return '0 Wh'
1038     return this.safeToFixed(value * 1000, 3) + ' Wh'
```

```
1039     },
1040
1041     // 获取缓存效率描述
1042     getCachingEfficiency(value) {
1043         if (!value) return '标准优化'
1044         const percent = this.safeToFixed(value * 100, 0)
1045         if (value < 0.2) return `基础优化 (${percent}%)`
1046         if (value < 0.4) return `标准优化 (${percent}%)`
1047         if (value < 0.6) return `良好优化 (${percent}%)`
1048         return `优秀优化 (${percent}%)`
1049     },
1050
1051     // 格式化文本长度
1052     formatTextLength(length) {
1053         if (!length && length  $\equiv$  0) return 'N/A';
1054         if (length > 10000) return this.safeToFixed(length / 1000, 1) + 'K 字
1055         符';
1056         return length + ' 字符';
1057     },
1058
1059     // 获取内容质量评分的样式类
1060     getQualityScoreClass(score) {
1061         if (!score && score  $\equiv$  0) return 'poor';
1062         if (score  $\geq$  80) return 'excellent';
1063         if (score  $\geq$  60) return 'good';
1064         if (score  $\geq$  40) return 'moderate';
1065         return 'poor';
1066     },
1067
1068     // 获取元数据状态
1069     getMetadataStatus(metadata) {
1070         if (!metadata) return '缺失';
1071
1072         const hasCount = (metadata.hasTitle ? 1 : 0) +
1073             (metadata.hasDescription ? 1 : 0) +
1074             (metadata.hasKeywords ? 1 : 0);
1075
1076         if (hasCount  $\equiv$  3) return '完整';
1077         if (hasCount  $\geq$  1) return '部分';
1078         return '缺失';
1079     },
1080
1081     // 获取元数据的样式类
1082     getMetadataClass(metadata) {
1083         if (!metadata) return 'poor';
1084
1085         const hasCount = (metadata.hasTitle ? 1 : 0) +
1086             (metadata.hasDescription ? 1 : 0) +
1087             (metadata.hasKeywords ? 1 : 0);
1088
1089         if (hasCount  $\equiv$  3) return 'good';
1090         if (hasCount  $\geq$  1) return 'moderate';
1091         return 'poor';
1092     },
1093 }
```

```
11093 // 获取资源类型的颜色
11094 getResourceColor(type) {
11095     const colors = {
11096         html: '#e34c26',
11097         css: '#563d7c',
11098         js: '#f1e05a',
11099         images: '#4caf50',
11100         fonts: '#ff9800',
11101         other: '#607d8b'
11102     };
11103     return colors[type] || '#607d8b';
11104 },
11105
11106 // 格式化资源类型名称
11107 formatResourceType(type) {
11108     const types = {
11109         html: 'HTML',
11110         css: 'CSS',
11111         js: 'JavaScript',
11112         images: '图像',
11113         fonts: '字体',
11114         other: '其他'
11115     };
11116     return types[type] || type;
11117 },
11118
11119 // 计算资源条形图的宽度
11120 getResourceBarWidth(count, allStats) {
11121     if (!allStats) return '0%';
11122     const total = Object.values(allStats).reduce((sum, c) => sum + c, 0);
11123     if (total === 0) return '0%';
11124     const percentage = (count / total) * 100;
11125     return Math.max(percentage, 3) + '%'; // 确保即使很小的值也有最小宽度
11126 },
11127
11128 // 获取性能指标（过滤掉非性能指标）
11129 getPerformanceMetrics(performance) {
11130     if (!performance) return {};
11131
11132     // 只选择性能指标来显示
11133     const metricsToShow = [
11134         'fcp', 'lcp', 'cls', 'fid', 'ttfb'
11135     ];
11136
11137     return Object.fromEntries(
11138         Object.entries(performance)
11139             .filter(([key]) => metricsToShow.includes(key))
11140     );
11141 },
11142
11143 // 格式化文件大小
11144 formatFileSize(bytes) {
11145     if (!bytes && bytes !== 0) return 'N/A';
11146     bytes = Number(bytes);
11147 }
```

```
1148     if (bytes ≥ 1048576) {
1149         return this.safeToFixed(bytes / 1048576, 2) + ' MB';
1150     }
1151     if (bytes ≥ 1024) {
1152         return this.safeToFixed(bytes / 1024, 2) + ' KB';
1153     }
1154     return bytes + ' B';
1155 },
1156
1157 // 获取建议图标
1158 getSuggestionIcon(type) {
1159     const icons = {
1160         'image': 'fa-image',
1161         'script': 'fa-code',
1162         'font': 'fa-font',
1163         'cdn': 'fa-server',
1164         'cache': 'fa-database',
1165         'compress': 'fa-file-archive',
1166         'performance': 'fa-tachometer-alt',
1167         'default': 'fa-lightbulb'
1168     };
1169     return icons[type] || icons.default;
1170 },
1171
1172 // 获取影响等级
1173 getImpactLevel(index) {
1174     const levels = ['低', '中', '高'];
1175     return levels[index % levels.length];
1176 },
1177
1178 // 获取影响类
1179 getImpactClass(index) {
1180     const classes = ['low', 'medium', 'high'];
1181     return classes[index % classes.length];
1182 },
1183
1184 // 获取指标描述
1185 getMetricDescription(metric) {
1186     const descriptions = {
1187         fcp: '首次内容绘制时间',
1188         lcp: '最大内容绘制时间',
1189         cls: '累计布局偏移',
1190         fid: '首次输入延迟',
1191         ttfb: '首次字节时间',
1192         responseTime: '服务器响应时间',
1193         pageSize: '页面总大小',
1194         totalResourceSize: '资源总大小',
1195         requestCount: '资源请求数量',
1196         domainCount: '请求域名数量',
1197         usesHttps: '使用HTTPS',
1198         serverType: '服务器类型',
1199         supportsCompression: '支持压缩',
1200         supportsCaching: '支持缓存',
1201         usesCdn: '使用CDN',
1202         cdnProvider: 'CDN提供商',
```

```
1203     resourceStats: '资源统计',
1204     securityScore: '安全评分',
1205     energyConsumption: '能源消耗',
1206     dataCenterEnergy: '数据中心能源',
1207     transmissionEnergy: '传输能源',
1208     deviceEnergy: '设备能源',
1209     carbonEmission: '碳排放量',
1210     annualCarbonEmission: '年度碳排放',
1211     renewablePercentage: '可再生能源比例',
1212     pue: '能源使用效率'
1213 };
1214 return descriptions[metric] || '未知描述';
1215 },
1216
1217 // 格式化能源消耗值
1218 formatEnergyValue(value) {
1219     if (value === null || value === undefined) return '0.0000';
1220     const wattHours = value * 1000; // 转换为瓦时
1221
1222     if (wattHours < 0.001) {
1223         // 小于1微瓦时, 显示为微瓦时
1224         return `${this.safeToFixed(wattHours * 1000000, 2)} μWh`;
1225     } else if (wattHours < 1) {
1226         // 小于1毫瓦时, 显示为毫瓦时
1227         return `${this.safeToFixed(wattHours * 1000, 2)} mWh`;
1228     } else {
1229         // 正常显示瓦时
1230         return `${this.safeToFixed(wattHours, 4)} Wh`;
1231     }
1232 },
1233
1234 // 安全的格式化函数, 防止对null或undefined调用toFixed
1235 safeToFixed(value, digits = 2) {
1236     if (value === null || value === undefined) return '0.00';
1237     return Number(value).toFixed(digits);
1238 },
1239
1240 // 格式化数字
1241 formatNumber(value) {
1242     if (value === null || value === undefined) return '无法获取';
1243     return this.safeToFixed(parseFloat(value), 2);
1244 },
1245 }
1246 }
1247 </script>
1248
1249 <style>
1250 /* 全局样式 */
1251 .green-web-app {
1252     max-width: 1200px;
1253     margin: 0 auto;
1254     padding: 20px;
1255     font-family: 'Avenir', Helvetica, Arial, sans-serif;
1256     color: #333;
1257     background-color: #f9f9f9;
```

```
1258 min-height: 100vh;
1259 display: flex;
1260 flex-direction: column;
1261 }
1262
1263 /* 头部样式 */
1264 .app-header {
1265     text-align: center;
1266     margin-bottom: 40px;
1267     padding: 35px 0;
1268     background: linear-gradient(135deg, #34c759, #32ade6);
1269     color: white;
1270     border-radius: 16px;
1271     box-shadow: 0 10px 30px rgba(50, 173, 230, 0.25);
1272     position: relative;
1273     overflow: hidden;
1274     transform: translateZ(0);
1275 }
1276
1277 .app-header::before {
1278     content: '';
1279     position: absolute;
1280     top: 0;
1281     left: 0;
1282     right: 0;
1283     bottom: 0;
1284     background:
1285         radial-gradient(circle at 30% 40%, rgba(255,255,255,0.2) 0%, transparent
1286 60%),
1287         radial-gradient(circle at 70% 60%, rgba(255,255,255,0.15) 0%,
1288 transparent 50%);
1289     z-index: 1;
1290 }
1291
1292 .app-header::after {
1293     content: '';
1294     position: absolute;
1295     top: -50%;
1296     left: -50%;
1297     right: -50%;
1298     bottom: -50%;
1299     background: linear-gradient(0deg, rgba(255,255,255,0) 40%,
1300 rgba(255,255,255,0.1) 50%, rgba(255,255,255,0) 60%);
1301     z-index: 2;
1302     transform: rotateZ(-45deg);
1303     animation: shimmer 8s infinite linear;
1304     pointer-events: none;
1305 }
1306
1307 @keyframes shimmer {
1308     0% { transform: translateX(-50%) rotateZ(-45deg); }
1309     100% { transform: translateX(100%) rotateZ(-45deg); }
1310 }
1311
1312 .logo {
```

```
1310 width: 60px;
1311 height: 60px;
1312 background: rgba(255, 255, 255, 0.95);
1313 border-radius: 50%;
1314 display: flex;
1315 align-items: center;
1316 justify-content: center;
1317 margin-right: 15px;
1318 box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
1319 animation: float 5s ease-in-out infinite;
1320 }
1321
1322 @keyframes float {
1323   0%, 100% { transform: translateY(0); }
1324   50% { transform: translateY(-6px); }
1325 }
1326
1327 .logo-icon {
1328   font-size: 35px;
1329 }
1330
1331 .app-header h1 {
1332   margin: 0;
1333   font-size: 36px;
1334   font-weight: 700;
1335   position: relative;
1336   z-index: 3;
1337   text-shadow: 0 2px 4px rgba(0, 0, 0, 0.15);
1338   letter-spacing: 0.5px;
1339 }
1340
1341 .subtitle {
1342   font-size: 17px;
1343   opacity: 0.95;
1344   margin-top: 8px;
1345   position: relative;
1346   z-index: 3;
1347   font-weight: 500;
1348   letter-spacing: 0.3px;
1349   text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
1350 }
1351
1352 /* 输入区域样式 */
1353 .input-section {
1354   margin-bottom: 40px;
1355   transform: translateY(-20px);
1356 }
1357
1358 .url-input-group {
1359   display: flex;
1360   width: 100%;
1361   max-width: 800px;
1362   margin: 0 auto;
1363 }
1364
```



```
1365 .domain-input {
1366     flex: 1;
1367     margin-right: 10px;
1368 }
1369
1370 .domain-input :deep(.el-input__wrapper) {
1371     padding: 4px 15px;
1372     box-shadow: none !important;
1373 }
1374
1375 .domain-input :deep(.el-input__inner) {
1376     height: 50px;
1377     font-size: 16px;
1378 }
1379
1380 .domain-input :deep(.el-input-group__append) {
1381     padding: 0;
1382 }
1383
1384 .domain-input :deep(.el-input-group__append button) {
1385     height: 58px;
1386     padding: 0 25px;
1387     font-size: 16px;
1388     font-weight: 600;
1389     border-radius: 0;
1390     box-shadow: none;
1391 }
1392
1393 .input-hint {
1394     margin-top: 12px;
1395     color: #8c8c8c;
1396     font-size: 14px;
1397     text-align: center;
1398     transition: all 0.3s ease;
1399 }
1400
1401 /* 加载动画 */
1402 .loading-container {
1403     display: flex;
1404     flex-direction: column;
1405     align-items: center;
1406     justify-content: center;
1407     padding: 80px 0;
1408     animation: fadeIn 0.5s ease-out;
1409 }
1410
1411 @keyframes fadeIn {
1412     from { opacity: 0; transform: translateY(15px); }
1413     to { opacity: 1; transform: translateY(0); }
1414 }
1415
1416 .earth-container {
1417     position: relative;
1418     width: 120px;
1419     height: 120px;
```

```
1420     margin-bottom: 30px;
1421 }
1422
1423 .earth {
1424     width: 100%;
1425     height: 100%;
1426     border-radius: 50%;
1427     background: linear-gradient(135deg, #34c759, #32ade6);
1428     box-shadow:
1429         0 0 60px rgba(50, 173, 230, 0.4),
1430         0 0 30px rgba(52, 199, 89, 0.3),
1431         inset 0 0 15px rgba(255, 255, 255, 0.3);
1432     animation: spin 10s linear infinite, pulse 5s ease-in-out infinite;
1433     position: relative;
1434     overflow: hidden;
1435 }
1436
1437 .earth::before {
1438     content: '';
1439     position: absolute;
1440     top: -10%;
1441     left: -10%;
1442     right: -10%;
1443     bottom: -10%;
1444     background-image: url('data:image/svg+xml;utf8,<svg width="100"
1445 height="100" viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"><rect
1446 x="0" y="0" width="10" height="10" fill="rgba(255,255,255,0.15)" /></svg>');
1445     background-size: 20px 20px;
1446     border-radius: 50%;
1447     animation: spin 20s linear infinite reverse;
1448     opacity: 0.7;
1449 }
1450
1451 @keyframes spin {
1452     0% { transform: rotate(0deg); }
1453     100% { transform: rotate(360deg); }
1454 }
1455
1456 @keyframes pulse {
1457     0%, 100% { transform: scale(1); }
1458     50% { transform: scale(1.05); }
1459 }
1460
1461 .loading-container p {
1462     font-size: 18px;
1463     font-weight: 500;
1464     margin-top: 0;
1465     text-align: center;
1466     color: #555;
1467     line-height: 1.6;
1468 }
1469
1470 .loading-container small {
1471     display: block;
1472     margin-top: 8px;
```

```
1473     font-size: 14px;
1474     color: #888;
1475 }
1476
1477 /* 结果摘要 */
1478 .result-section {
1479     animation: fadeInUp 0.8s ease-out;
1480 }
1481
1482 @keyframes fadeInUp {
1483     from { opacity: 0; transform: translateY(30px); }
1484     to { opacity: 1; transform: translateY(0); }
1485 }
1486
1487 .result-summary {
1488     display: flex;
1489     flex-wrap: wrap;
1490     gap: 20px;
1491     width: 100%;
1492     margin-bottom: 40px;
1493 }
1494
1495 .summary-card {
1496     flex: 1;
1497     min-width: 300px;
1498     background-color: #fff;
1499     border-radius: 8px;
1500     box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
1501     overflow: hidden;
1502     transition: all 0.3s ease;
1503     display: block;
1504     padding: 0;
1505     align-items: initial;
1506     margin: 0;
1507 }
1508
1509 .summary-card.green {
1510     border-top: 5px solid #4caf50;
1511     background: #fff;
1512 }
1513
1514 .summary-card.red {
1515     border-top: 5px solid #f44336;
1516     background: #fff;
1517 }
1518
1519 /* 移除冲突的伪元素 */
1520 .summary-card::before,
1521 .summary-card::after {
1522     display: none;
1523     content: none;
1524 }
1525
1526 .summary-header {
1527     padding: 15px 20px;
```

```
1528     background-color: #f9f9f9;
1529     border-bottom: 1px solid #eee;
1530 }
1531
1532 .summary-title {
1533     font-size: 18px;
1534     font-weight: 600;
1535     color: #333;
1536 }
1537
1538 .summary-subtitle {
1539     font-size: 14px;
1540     color: #666;
1541     margin-top: 5px;
1542 }
1543
1544 .summary-content {
1545     padding: 20px;
1546 }
1547
1548 .summary-section {
1549     margin-bottom: 25px;
1550 }
1551
1552 .summary-section:last-child {
1553     margin-bottom: 0;
1554 }
1555
1556 /* 重置原来summary-content的样式 */
1557 .summary-content h2,
1558 .summary-content p {
1559     margin: 0;
1560     text-align: left;
1561     color: #333;
1562     text-shadow: none;
1563 }
1564
1565 /* 总碳排放显示 */
1566 .total-emission {
1567     display: flex;
1568     align-items: flex-start;
1569 }
1570
1571 .carbon-icon {
1572     margin-right: 15px;
1573     color: #555;
1574     padding-top: 5px;
1575 }
1576
1577 .carbon-details {
1578     flex: 1;
1579 }
1580
1581 .total-title {
1582     font-size: 16px;
```

```
1583     font-weight: 500;
1584     color: #333;
1585     margin-bottom: 5px;
1586 }
1587
1588 .total-value {
1589     font-size: 24px;
1590     font-weight: 700;
1591     color: #2c3e50;
1592     margin-bottom: 15px;
1593 }
1594
1595 .total-breakdown {
1596     background-color: #f5f5f5;
1597     border-radius: 6px;
1598     padding: 12px 15px;
1599 }
1600
1601 .total-item {
1602     display: flex;
1603     justify-content: space-between;
1604     margin-bottom: 8px;
1605     font-size: 14px;
1606 }
1607
1608 .total-label {
1609     color: #555;
1610     font-weight: 500;
1611 }
1612
1613 .total-item .total-value {
1614     font-size: 16px;
1615     margin-bottom: 0;
1616 }
1617
1618 .total-info {
1619     font-size: 12px;
1620     color: #777;
1621     margin-bottom: 12px;
1622     padding-left: 5px;
1623 }
1624
1625 /* 确保每个section内的最后一个元素没有margin-bottom */
1626 .summary-section > *:last-child {
1627     margin-bottom: 0;
1628 }
1629
1630 .summary-icon {
1631     width: 80px;
1632     height: 80px;
1633     background: rgba(255, 255, 255, 0.25);
1634     border-radius: 50%;
1635     display: flex;
1636     align-items: center;
1637     justify-content: center;
```

```
1638     margin-right: 30px;
1639     position: relative;
1640     z-index: 1;
1641     box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);
1642 }
1643
1644 .summary-content {
1645     position: relative;
1646     z-index: 1;
1647 }
1648
1649 .summary-content h2 {
1650     margin: 0;
1651     font-size: 32px;
1652     font-weight: 700;
1653     margin-bottom: 10px;
1654     text-shadow: 0 2px 4px rgba(0, 0, 0, 0.15);
1655 }
1656
1657 .summary-content p {
1658     margin: 8px 0 0;
1659     font-size: 17px;
1660     opacity: 0.95;
1661     text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
1662 }
1663
1664 /* 结果网格 */
1665 .result-grid {
1666     max-width: 1200px;
1667     margin: 0 auto;
1668     padding: 20px 0;
1669     position: relative;
1670 }
1671
1672 .result-card-container {
1673     display: flex;
1674     flex-direction: column;
1675     gap: 30px;
1676     margin-top: 20px;
1677 }
1678
1679 .result-row {
1680     display: grid;
1681     grid-template-columns: repeat(12, 1fr);
1682     gap: 25px;
1683     margin-bottom: 10px;
1684 }
1685
1686 .result-card {
1687     background: #fff;
1688     border-radius: 18px;
1689     box-shadow: 0 10px 30px rgba(0, 0, 0, 0.05);
1690     overflow: hidden;
1691     transition: all 0.35s cubic-bezier(0.25, 0.8, 0.25, 1);
1692     display: flex;
```

```
1693     flex-direction: column;
1694     position: relative;
1695     border: 1px solid rgba(230, 230, 230, 0.6);
1696     backdrop-filter: blur(10px);
1697     padding: 25px;
1698     grid-column: span 4;
1699 }
1700
1701 .result-card::before {
1702     content: '';
1703     position: absolute;
1704     top: 0;
1705     left: 0;
1706     width: 100%;
1707     height: 6px;
1708     background: linear-gradient(90deg, #32ade6, #34c759);
1709     opacity: 0.8;
1710 }
1711
1712 /* 不同类型卡片的顶部颜色条 */
1713 .result-card.energy-source::before {
1714     background: linear-gradient(90deg, #34c759, #4cd964);
1715 }
1716
1717 .result-card.carbon-map::before {
1718     background: linear-gradient(90deg, #32ade6, #007aff);
1719 }
1720
1721 .result-card.energy-consumption::before {
1722     background: linear-gradient(90deg, #5ac8fa, #007aff);
1723 }
1724
1725 .result-card.carbon-impact::before {
1726     background: linear-gradient(90deg, #ff9500, #ff3b30);
1727 }
1728
1729 .result-card.site-info::before {
1730     background: linear-gradient(90deg, #007aff, #5856d6);
1731 }
1732
1733 .result-card.performance::before {
1734     background: linear-gradient(90deg, #5856d6, #af52de);
1735 }
1736
1737 .result-card.suggestions::before {
1738     background: linear-gradient(90deg, #af52de, #ff2d55);
1739 }
1740
1741 .result-card.green-hosting::before {
1742     background: linear-gradient(90deg, #34c759, #00c7be);
1743 }
1744
1745 .result-card:hover {
1746     transform: translateY(-5px);
1747     box-shadow: 0 15px 35px rgba(0, 0, 0, 0.07);
```

```
1748 }
1749
1750 .result-card.wide-card {
1751     grid-column: span 6;
1752 }
1753
1754 .result-card.major-card {
1755     grid-column: span 6;
1756 }
1757
1758 .card-header {
1759     display: flex;
1760     justify-content: space-between;
1761     align-items: flex-start;
1762     margin-bottom: 30px;
1763     position: relative;
1764 }
1765
1766 .card-header h3 {
1767     margin: 0;
1768     font-size: 20px;
1769     font-weight: 600;
1770     color: #333;
1771     position: relative;
1772     padding-bottom: 15px;
1773 }
1774
1775 .card-header h3::after {
1776     content: '';
1777     position: absolute;
1778     width: 40px;
1779     height: 3px;
1780     background: linear-gradient(to right, #34c759, #32ade6);
1781     left: 0;
1782     bottom: 0;
1783     border-radius: 3px;
1784 }
1785
1786 .card-icon {
1787     width: 46px;
1788     height: 46px;
1789     border-radius: 14px;
1790     display: flex;
1791     align-items: center;
1792     justify-content: center;
1793     background: linear-gradient(135deg, rgba(50, 173, 230, 0.1), rgba(52, 199,
1794 89, 0.1));
1795     color: #32ade6;
1796     box-shadow: 0 6px 15px rgba(0, 0, 0, 0.05);
1797     margin-left: 15px;
1798 }
1799 /* 卡片内容区域共享样式 */
1800 .card-content {
1801     flex: 1;
```



```
1802     display: flex;
1803     flex-direction: column;
1804 }
1805
1806 /* 数据展示样式 */
1807 .details {
1808     display: flex;
1809     flex-direction: column;
1810     flex: 1;
1811     gap: 16px;
1812 }
1813
1814 .detail-item {
1815     display: flex;
1816     justify-content: space-between;
1817     font-size: 15px;
1818     align-items: center;
1819     padding: 10px 0;
1820     border-bottom: 1px solid rgba(230, 230, 230, 0.5);
1821 }
1822
1823 .detail-item:last-child {
1824     border-bottom: none;
1825 }
1826
1827 .detail-label {
1828     color: #666;
1829     font-weight: 500;
1830 }
1831
1832 .detail-value {
1833     font-weight: 600;
1834     color: #333;
1835     background: linear-gradient(135deg, #f9fafc, #f1f4f8);
1836     padding: 6px 14px;
1837     border-radius: 10px;
1838     box-shadow: 0 2px 6px rgba(0, 0, 0, 0.03);
1839 }
1840
1841 .detail-value.highlight-text {
1842     color: #32ade6;
1843     background: linear-gradient(135deg, rgba(50, 173, 230, 0.1), rgba(52, 199,
1844 89, 0.05));
1845     font-weight: 700;
1846 }
1847
1848 .green-value {
1849     color: #34c759 !important;
1850 }
1851
1852 /* 能源消耗卡片样式 */
1853 .energy-consumption-grid {
1854     display: grid;
1855     grid-template-columns: repeat(3, 1fr);
1856     gap: 20px;
```

```
1856     margin-top: 20px;
1857 }
1858
1859 .energy-consumption-item {
1860     display: flex;
1861     flex-direction: column;
1862     align-items: center;
1863     text-align: center;
1864     background: linear-gradient(135deg, #f9fafc, #f1f4f8);
1865     border-radius: 14px;
1866     padding: 20px 15px;
1867     transition: all 0.3s ease;
1868     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.03);
1869 }
1870
1871 .energy-consumption-item:hover {
1872     transform: translateY(-3px);
1873     box-shadow: 0 8px 20px rgba(0, 0, 0, 0.08);
1874 }
1875
1876 .consumption-icon {
1877     width: 50px;
1878     height: 50px;
1879     border-radius: 50%;
1880     margin-bottom: 12px;
1881     display: flex;
1882     align-items: center;
1883     justify-content: center;
1884     background-size: 60%;
1885     background-position: center;
1886     background-repeat: no-repeat;
1887     box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);
1888 }
1889
1890 .data-center-icon {
1891     background-color: rgba(50, 173, 230, 0.1);
1892     background-image: url('data:image/svg+xml;utf8,<svg
1893 xmlns="http://www.w3.org/2000/svg" fill="%2332ADE6" viewBox="0 0 24 24">
1894 <path d="M4 1h16c1.1 0 2 .9 2 2v4c0 1.1-.9 2-2 2H4c-1.1 0-2-.9-2-2V3c0-
1895 1.1-.9-2 2-2zm0 9h16c1.1 0 2 .9 2 2v4c0 1.1-.9 2-2 2H4c-1.1 0-2-.9-2-2v-4c0-
1896 1.1-.9-2 2-2zm0 10c0-1.1-.9-2-2-2h12c1.1 0 2 .9 2 2v0c0 1.1-.9 2-2 2H6c-1.1 0-
1897 2-.9-2-2v0z" /></svg>');
1898 }
1899
1900 .network-icon {
1901     background-color: rgba(255, 149, 0, 0.1);
1902     background-image: url('data:image/svg+xml;utf8,<svg
1903 xmlns="http://www.w3.org/2000/svg" fill="%23FF9500" viewBox="0 0 24 24">
1904 <path d="M12 21l-8-9h6V3h4v9h6l-8 9z" /></svg>');
1905 }
1906
1907 .device-icon {
1908     background-color: rgba(52, 199, 89, 0.1);
1909     background-image: url('data:image/svg+xml;utf8,<svg
1910 xmlns="http://www.w3.org/2000/svg" fill="%2334C759" viewBox="0 0 24 24">
```

```
<path d="M20 18c1.1 0 1.99-.9 1.99-2L22 6c0-1.1-.9-2-2-2H4c-1.1 0-2 .9-2
2v10c0 1.1.9 2 2 2H0v2h24v-2h-4zM4 16V6h16v10.01H4z"/></svg>');
1903 }
1904
1905 .consumption-value {
1906     font-size: 18px;
1907     font-weight: 700;
1908     color: #333;
1909     margin: 5px 0;
1910 }
1911
1912 .consumption-label {
1913     font-size: 14px;
1914     color: #666;
1915 }
1916
1917 /* 碳排放数据样式 */
1918 .carbon-stats {
1919     display: grid;
1920     grid-template-columns: repeat(2, 1fr);
1921     gap: 20px;
1922     margin-top: 10px;
1923 }
1924
1925 .carbon-stat-item {
1926     display: flex;
1927     flex-direction: column;
1928     font-size: 14px;
1929     background: linear-gradient(135deg, #f9fafc, #f1f4f8);
1930     border-radius: 14px;
1931     padding: 16px;
1932     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.03);
1933     transition: all 0.3s ease;
1934 }
1935
1936 .carbon-stat-item:hover {
1937     transform: translateY(-3px);
1938     box-shadow: 0 8px 20px rgba(0, 0, 0, 0.08);
1939 }
1940
1941 .stat-label {
1942     color: #666;
1943     margin-bottom: 10px;
1944     font-weight: 500;
1945 }
1946
1947 .stat-value {
1948     font-weight: 700;
1949     color: #333;
1950     font-size: 18px;
1951 }
1952
1953 .carbon-annual-impact {
1954     grid-column: span 2;
```

```
1955     background: linear-gradient(135deg, rgba(50, 173, 230, 0.05), rgba(52,
1956     border-radius: 14px;
1957     padding: 20px;
1958     margin-top: 10px;
1959     box-shadow: 0 6px 15px rgba(0, 0, 0, 0.05);
1960     border: 1px dashed rgba(50, 173, 230, 0.2);
1961     text-align: center;
1962 }
1963
1964 .impact-title {
1965     font-weight: 600;
1966     color: #333;
1967     margin-bottom: 10px;
1968     font-size: 16px;
1969 }
1970
1971 .impact-value {
1972     font-size: 24px;
1973     font-weight: 700;
1974     color: #32ade6;
1975     margin: 10px 0;
1976 }
1977
1978 .impact-equivalent {
1979     font-size: 14px;
1980     color: #666;
1981 }
1982
1983 .highlight-text {
1984     color: #34c759;
1985     font-weight: 700;
1986 }
1987
1988 /* 能源图表 */
1989 .energy-chart {
1990     display: flex;
1991     flex-direction: column;
1992     align-items: center;
1993     margin: 25px 0;
1994 }
1995
1996 .donut-chart {
1997     position: relative;
1998     width: 170px;
1999     height: 170px;
2000     filter: drop-shadow(0 10px 20px rgba(0, 0, 0, 0.15));
2001 }
2002
2003 .donut-hole {
2004     position: absolute;
2005     top: 50%;
2006     left: 50%;
2007     transform: translate(-50%, -50%);
2008     width: 110px;
```

```
2009     height: 110px;
2010     background: #fff;
2011     border-radius: 50%;
2012     display: flex;
2013     align-items: center;
2014     justify-content: center;
2015     font-size: 32px;
2016     font-weight: bold;
2017     color: #34c759;
2018     box-shadow:
2019         inset 0 0 20px rgba(0, 0, 0, 0.05),
2020         0 6px 20px rgba(52, 199, 89, 0.15);
2021 }
2022
2023 .donut-ring {
2024     width: 100%;
2025     height: 100%;
2026     border-radius: 50%;
2027     background: #ff3b30;
2028     filter: drop-shadow(0 6px 12px rgba(255, 59, 48, 0.25));
2029 }
2030
2031 .renewable {
2032     position: absolute;
2033     top: 0;
2034     left: 0;
2035     width: 100%;
2036     height: 100%;
2037     border-radius: 50%;
2038     background: conic-gradient(#34c759 var(--percent), transparent 0);
2039     --percent: 0%;
2040     filter: drop-shadow(0 6px 12px rgba(52, 199, 89, 0.25));
2041     animation: fillDonut 1.5s ease-out forwards;
2042 }
2043
2044 @keyframes fillDonut {
2045     from { opacity: 0; transform: scale(0.8); }
2046     to { opacity: 1; transform: scale(1); }
2047 }
2048
2049 .chart-legend {
2050     display: flex;
2051     justify-content: center;
2052     gap: 30px;
2053     margin-top: 25px;
2054 }
2055
2056 .legend-item {
2057     display: flex;
2058     align-items: center;
2059     gap: 10px;
2060     font-size: 15px;
2061     font-weight: 500;
2062 }
2063
```

```
2064 .legend-color {
2065     width: 18px;
2066     height: 18px;
2067     border-radius: 5px;
2068 }
2069
2070 .legend-color.renewable {
2071     background-color: #34c759;
2072     box-shadow: 0 3px 8px rgba(52, 199, 89, 0.3);
2073 }
2074
2075 .legend-color.fossil {
2076     background-color: #ff3b30;
2077     box-shadow: 0 3px 8px rgba(255, 59, 48, 0.3);
2078 }
2079
2080 /* 可用性提示 */
2081 .data-unavailable {
2082     color: #888;
2083     font-style: italic;
2084     padding: 20px;
2085     background-color: #f9fafc;
2086     border-radius: 14px;
2087     font-size: 15px;
2088     text-align: center;
2089     margin: 15px 0;
2090     box-shadow: inset 0 0 15px rgba(0, 0, 0, 0.03);
2091     border: 1px dashed rgba(0, 0, 0, 0.1);
2092 }
2093
2094 /* 卡片摘要 */
2095 .card-summary {
2096     margin-top: auto;
2097     padding-top: 20px;
2098     text-align: center;
2099 }
2100
2101 .summary-badge {
2102     display: inline-block;
2103     padding: 8px 16px;
2104     border-radius: 30px;
2105     font-size: 14px;
2106     font-weight: 600;
2107     margin-bottom: 10px;
2108 }
2109
2110 .green-badge {
2111     background: linear-gradient(135deg, rgba(52, 199, 89, 0.15), rgba(50, 173,
2112 230, 0.15));
2113     color: #34c759;
2114     box-shadow: 0 4px 10px rgba(52, 199, 89, 0.2);
2115 }
2116
2117 .standard-badge {
```

```
2117     background: linear-gradient(135deg, rgba(255, 149, 0, 0.15), rgba(255, 59,
2118     color: #ff9500;
2119     box-shadow: 0 4px 10px rgba(255, 149, 0, 0.2);
2120 }
2121
2122 /* 响应式调整 */
2123 @media (max-width: 1200px) {
2124     .result-row {
2125         grid-template-columns: repeat(6, 1fr);
2126     }
2127
2128     .result-card {
2129         grid-column: span 3;
2130     }
2131
2132     .result-card.wide-card,
2133     .result-card.major-card {
2134         grid-column: span 6;
2135     }
2136 }
2137
2138 @media (max-width: 768px) {
2139     .result-row {
2140         grid-template-columns: repeat(1, 1fr);
2141         gap: 20px;
2142     }
2143
2144     .result-card,
2145     .result-card.wide-card,
2146     .result-card.major-card {
2147         grid-column: span 1;
2148     }
2149
2150     .carbon-stats,
2151     .energy-consumption-grid {
2152         grid-template-columns: 1fr;
2153     }
2154
2155     .carbon-annual-impact {
2156         grid-column: span 1;
2157     }
2158 }
2159
2160 /* 能源消耗数据提示 */
2161 .energy-note {
2162     padding: 15px;
2163     background-color: #f9fafc;
2164     border-radius: 12px;
2165     margin-top: 20px;
2166     font-size: 14px;
2167     color: #666;
2168     border: 1px dashed rgba(0, 0, 0, 0.1);
2169 }
2170
```

```
2171 /* 性能指标和优化建议布局 */
2172 .performance-metrics-v2 {
2173     display: flex;
2174     flex-direction: column;
2175     gap: 16px;
2176     margin-top: 15px;
2177     animation: fadeInUp 0.6s ease-out forwards;
2178 }
2179
2180 .metrics-grid {
2181     display: flex;
2182     flex-wrap: wrap;
2183     gap: 15px;
2184 }
2185
2186 .metric-card {
2187     flex: 1;
2188     min-width: 200px;
2189     background-color: #fff;
2190     border-radius: 16px;
2191     box-shadow: 0 8px 20px rgba(0, 0, 0, 0.06);
2192     overflow: hidden;
2193     transition: all 0.3s ease;
2194     display: flex;
2195     flex-direction: column;
2196     align-items: center;
2197     text-align: center;
2198     margin-bottom: 10px;
2199     border: 1px solid rgba(0, 0, 0, 0.03);
2200 }
2201
2202 .metric-card:hover {
2203     transform: translateY(-5px);
2204     box-shadow: 0 12px 28px rgba(0, 0, 0, 0.1);
2205 }
2206
2207 .metric-visual {
2208     width: 100%;
2209     padding: 25px 15px;
2210     background: linear-gradient(135deg, #f9fafc, #f1f4f8);
2211     border-bottom: 1px solid rgba(0, 0, 0, 0.05);
2212     display: flex;
2213     justify-content: center;
2214     align-items: center;
2215 }
2216
2217 .metric-circle {
2218     width: 110px;
2219     height: 110px;
2220     border-radius: 50%;
2221     display: flex;
2222     align-items: center;
2223     justify-content: center;
2224     font-size: 28px;
2225     font-weight: 700;
```



```
2226     color: #fff;
2227     box-shadow: 0 8px 20px rgba(0, 0, 0, 0.12);
2228     position: relative;
2229     overflow: hidden;
2230 }
2231
2232 .metric-circle::before {
2233     content: '';
2234     position: absolute;
2235     width: 100%;
2236     height: 100%;
2237     background: radial-gradient(circle at 30% 30%, rgba(255, 255, 255, 0.25),
transparent 70%);
2238 }
2239
2240 .metric-circle.good {
2241     background: linear-gradient(135deg, #34c759, #32ade6);
2242 }
2243
2244 .metric-circle.average {
2245     background: linear-gradient(135deg, #ff9500, #ff6a00);
2246 }
2247
2248 .metric-circle.poor {
2249     background: linear-gradient(135deg, #ff3b30, #ff375f);
2250 }
2251
2252 .metric-circle.unknown {
2253     background: linear-gradient(135deg, #8e8e93, #aeaeb2);
2254 }
2255
2256 .metric-info {
2257     padding: 20px 15px;
2258     background-color: #fff;
2259     width: 100%;
2260 }
2261
2262 .metric-name {
2263     font-size: 16px;
2264     font-weight: 600;
2265     color: #333;
2266     margin-bottom: 8px;
2267 }
2268
2269 .metric-description {
2270     font-size: 14px;
2271     color: #666;
2272     line-height: 1.5;
2273 }
2274
2275 .network-stats {
2276     background: linear-gradient(135deg, #f9fafc, #f5f7fa);
2277     padding: 20px;
2278     border-radius: 16px;
2279     box-shadow: 0 8px 20px rgba(0, 0, 0, 0.04);
```

```
2280     margin-top: 15px;
2281     border: 1px solid rgba(0, 0, 0, 0.03);
2282 }
2283
2284 .network-title {
2285     font-size: 17px;
2286     color: #333;
2287     font-weight: 600;
2288     margin-bottom: 15px;
2289     padding-bottom: 10px;
2290     border-bottom: 1px solid rgba(0, 0, 0, 0.08);
2291     position: relative;
2292 }
2293
2294 .network-title::after {
2295     content: '';
2296     position: absolute;
2297     width: 40px;
2298     height: 3px;
2299     background: linear-gradient(to right, #34c759, #32ade6);
2300     left: 0;
2301     bottom: -1px;
2302     border-radius: 3px;
2303 }
2304
2305 .network-grid {
2306     display: grid;
2307     grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
2308     gap: 15px;
2309 }
2310
2311 .network-item {
2312     background: #fff;
2313     border-radius: 12px;
2314     padding: 15px;
2315     box-shadow: 0 6px 16px rgba(0, 0, 0, 0.04);
2316     transition: all 0.3s ease;
2317     display: flex;
2318     flex-direction: column;
2319     align-items: center;
2320     text-align: center;
2321     border: 1px solid rgba(0, 0, 0, 0.03);
2322 }
2323
2324 .network-item:hover {
2325     transform: translateY(-3px);
2326     box-shadow: 0 10px 24px rgba(0, 0, 0, 0.06);
2327 }
2328
2329 .network-value {
2330     font-size: 22px;
2331     font-weight: 700;
2332     color: #32ade6;
2333     margin-bottom: 8px;
2334 }
```

```
2335
2336 .network-label {
2337     font-size: 14px;
2338     color: #666;
2339 }
2340
2341 .data-source-info {
2342     margin-top: 20px;
2343     padding: 12px 15px;
2344     background-color: #f9fafc;
2345     color: #888;
2346     font-size: 13px;
2347     font-style: italic;
2348     border-radius: 8px;
2349     border: 1px dashed rgba(0, 0, 0, 0.1);
2350     text-align: center;
2351 }
2352
2353 /* 优化建议布局 */
2354 .suggestions-container {
2355     padding: 25px;
2356 }
2357
2358 .suggestions-list {
2359     display: grid;
2360     grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
2361     gap: 20px;
2362 }
2363
2364 .suggestion-card {
2365     background-color: #fff;
2366     border-radius: 12px;
2367     box-shadow: 0 6px 16px rgba(0, 0, 0, 0.08);
2368     overflow: hidden;
2369     transition: all 0.3s ease;
2370     display: flex;
2371     flex-direction: column;
2372     height: 100%;
2373     border: 1px solid rgba(0, 0, 0, 0.05);
2374 }
2375
2376 .suggestion-card:hover {
2377     transform: translateY(-5px);
2378     box-shadow: 0 12px 28px rgba(0, 0, 0, 0.12);
2379 }
2380
2381 .suggestion-header {
2382     padding: 18px;
2383     background: linear-gradient(135deg, #f8faff, #f0f4f9);
2384     border-bottom: 1px solid #eee;
2385     display: flex;
2386     align-items: center;
2387 }
2388
2389 .suggestion-icon {
```

```
2390     margin-right: 12px;
2391     width: 38px;
2392     height: 38px;
2393     border-radius: 50%;
2394     display: flex;
2395     align-items: center;
2396     justify-content: center;
2397     background: linear-gradient(135deg, rgba(50, 173, 230, 0.15), rgba(52,
199, 89, 0.15));
2398     color: #32ade6;
2399     box-shadow: 0 4px 8px rgba(50, 173, 230, 0.15);
2400 }
2401
2402 .suggestion-title {
2403     font-size: 16px;
2404     font-weight: 600;
2405     color: #333;
2406     letter-spacing: 0.3px;
2407 }
2408
2409 .suggestion-content {
2410     padding: 20px;
2411     color: #555;
2412     font-size: 15px;
2413     line-height: 1.7;
2414     flex: 1;
2415     background-color: #fff;
2416     position: relative;
2417 }
2418
2419 .suggestion-content::before {
2420     content: '';
2421     position: absolute;
2422     left: 20px;
2423     right: 20px;
2424     height: 2px;
2425     top: 0;
2426     background: linear-gradient(to right, rgba(50, 173, 230, 0.1),
transparent);
2427     border-radius: 2px;
2428 }
2429
2430 .suggestion-impact {
2431     padding: 12px;
2432     text-align: center;
2433     font-weight: 600;
2434     font-size: 14px;
2435     color: #fff;
2436     letter-spacing: 0.5px;
2437     text-transform: uppercase;
2438     display: flex;
2439     align-items: center;
2440     justify-content: center;
2441 }
2442
```

```
2443 .suggestion-impact.low {
2444     background: linear-gradient(to right, #4caf50, #66bb6a);
2445 }
2446
2447 .suggestion-impact.medium {
2448     background: linear-gradient(to right, #ff9800, #ffa726);
2449 }
2450
2451 .suggestion-impact.high {
2452     background: linear-gradient(to right, #f44336, #ef5350);
2453 }
2454
2455 @media (max-width: 768px) {
2456     .suggestions-list {
2457         grid-template-columns: 1fr;
2458     }
2459
2460     .metrics-grid {
2461         flex-direction: column;
2462     }
2463
2464     .metric-card {
2465         width: 100%;
2466         min-width: 100%;
2467     }
2468 }
2469
2470 .energy-note {
2471     padding: 15px;
2472     background-color: #f5f7fa;
2473     border-radius: 6px;
2474     margin-top: 20px;
2475     font-size: 14px;
2476     color: #666;
2477 }
2478
2479 /* 页脚样式 */
2480 .footer {
2481     margin-top: 80px;
2482     padding: 30px 0;
2483     color: #666;
2484     font-size: 14px;
2485     border-top: 1px solid #eee;
2486     text-align: center;
2487     background: linear-gradient(180deg, transparent, rgba(240, 240, 240,
2488 0.5));
2489     border-radius: 0 0 16px 16px;
2490 }
2491
2492 .copyright {
2493     margin-top: 10px;
2494     font-size: 13px;
2495     opacity: 0.8;
2496 }
```

```
2497  /* 响应式调整 */
2498  @media (max-width: 768px) {
2499      .app-header {
2500          padding: 25px 0;
2501      }
2502
2503      .app-header h1 {
2504          font-size: 28px;
2505      }
2506
2507      .logo {
2508          width: 50px;
2509          height: 50px;
2510      }
2511
2512      .logo-icon {
2513          font-size: 28px;
2514      }
2515
2516      .url-input-group {
2517          flex-direction: column;
2518      }
2519
2520      .domain-input {
2521          margin-right: 0;
2522          margin-bottom: 10px;
2523      }
2524
2525      .browser-select {
2526          width: 100%;
2527          margin-bottom: 10px;
2528          margin-right: 0;
2529      }
2530
2531      .result-grid {
2532          grid-template-columns: 1fr;
2533          gap: 20px;
2534      }
2535
2536      .result-card {
2537          padding: 25px;
2538      }
2539
2540      .carbon-stats {
2541          grid-template-columns: 1fr;
2542      }
2543
2544      .summary-icon {
2545          width: 60px;
2546          height: 60px;
2547          margin-right: 20px;
2548      }
2549
2550      .summary-content h2 {
2551          font-size: 26px;
```

```
2552     }
2553
2554     .summary-card {
2555         padding: 25px;
2556     }
2557
2558     .score-container {
2559         flex-direction: column;
2560         align-items: center;
2561     }
2562
2563     .carbon-score-card {
2564         width: 90%;
2565         margin-bottom: 10px;
2566     }
2567
2568     .donut-chart {
2569         width: 120px;
2570         height: 120px;
2571     }
2572
2573     .donut-hole {
2574         width: 70px;
2575         height: 70px;
2576         font-size: 22px;
2577     }
2578 }
2579
2580 @media (max-width: 480px) {
2581     .app-header h1 {
2582         font-size: 24px;
2583     }
2584
2585     .logo {
2586         width: 40px;
2587         height: 40px;
2588         margin-right: 10px;
2589     }
2590
2591     .logo-icon {
2592         font-size: 22px;
2593     }
2594
2595     .result-card {
2596         padding: 20px;
2597     }
2598
2599     .summary-icon {
2600         width: 50px;
2601         height: 50px;
2602         margin-right: 15px;
2603     }
2604
2605     .summary-content h2 {
2606         font-size: 22px;
```

```
2607     }
2608
2609     .summary-content p {
2610         font-size: 14px;
2611     }
2612
2613     .carbon-score-card {
2614         width: 100%;
2615         padding: 12px;
2616     }
2617
2618     .score-circle {
2619         width: 50px;
2620         height: 50px;
2621         font-size: 20px;
2622     }
2623 }
2624
2625 /* 添加浏览器选择下拉菜单的样式 */
2626 .browser-select {
2627     margin-right: 10px;
2628     width: 140px;
2629 }
2630
2631 /* 调整URL输入框组的样式以容纳新的下拉菜单 */
2632 .url-input-group {
2633     display: flex;
2634     width: 100%;
2635     max-width: 800px;
2636     margin: 0 auto;
2637 }
2638
2639 .domain-input {
2640     flex: 1;
2641     margin-right: 10px;
2642 }
2643
2644 /* 碳排放分数显示 */
2645 .score-container {
2646     display: flex;
2647     justify-content: space-around;
2648     margin-bottom: 20px;
2649     flex-wrap: wrap;
2650     gap: 15px;
2651 }
2652
2653 .carbon-score-card {
2654     display: flex;
2655     flex-direction: column;
2656     align-items: center;
2657     padding: 15px;
2658     background: white;
2659     border-radius: 12px;
2660     box-shadow: 0 4px 15px rgba(0, 0, 0, 0.08);
2661     min-width: 110px;
```



```
2662     text-align: center;
2663     position: relative;
2664     overflow: hidden;
2665     transition: all 0.3s ease;
2666 }
2667
2668 .carbon-score-card:hover {
2669     transform: translateY(-5px);
2670     box-shadow: 0 8px 25px rgba(0, 0, 0, 0.12);
2671 }
2672
2673 .carbon-score-card::before {
2674     content: '';
2675     position: absolute;
2676     top: 0;
2677     left: 0;
2678     right: 0;
2679     height: 6px;
2680     background: linear-gradient(90deg, #ff3b30, #ff9500);
2681 }
2682
2683 .carbon-score-card.excellent::before {
2684     background: linear-gradient(90deg, #4cd964, #34c759);
2685 }
2686
2687 .carbon-score-card.good::before {
2688     background: linear-gradient(90deg, #34c759, #5ac8fa);
2689 }
2690
2691 .carbon-score-card.moderate::before {
2692     background: linear-gradient(90deg, #ffcc00, #ff9500);
2693 }
2694
2695 .carbon-score-card.poor::before {
2696     background: linear-gradient(90deg, #ff3b30, #ff9500);
2697 }
2698
2699 .score-circle {
2700     font-size: 24px;
2701     font-weight: 700;
2702     background: #f5f7fa;
2703     width: 60px;
2704     height: 60px;
2705     border-radius: 50%;
2706     display: flex;
2707     align-items: center;
2708     justify-content: center;
2709     margin-bottom: 10px;
2710     color: #333;
2711     border: 3px solid #eee;
2712 }
2713
2714 .carbon-score-card.excellent .score-circle {
2715     background: rgba(76, 217, 100, 0.15);
2716     color: #34c759;
```

```
2717     border-color: rgba(76, 217, 100, 0.3);
2718 }
2719
2720 .carbon-score-card.good .score-circle {
2721     background: rgba(90, 200, 250, 0.15);
2722     color: #007aff;
2723     border-color: rgba(90, 200, 250, 0.3);
2724 }
2725
2726 .carbon-score-card.moderate .score-circle {
2727     background: rgba(255, 204, 0, 0.15);
2728     color: #ff9500;
2729     border-color: rgba(255, 204, 0, 0.3);
2730 }
2731
2732 .carbon-score-card.poor .score-circle {
2733     background: rgba(255, 59, 48, 0.15);
2734     color: #ff3b30;
2735     border-color: rgba(255, 59, 48, 0.3);
2736 }
2737
2738 .score-label {
2739     font-size: 14px;
2740     color: #555;
2741     margin-bottom: 5px;
2742     font-weight: 500;
2743 }
2744
2745 .score-desc {
2746     font-size: 13px;
2747     color: #888;
2748 }
2749
2750 /* 错误提示样式 */
2751 .error-alerts {
2752     margin-bottom: 20px;
2753     width: 100%;
2754 }
2755
2756 .error-alert {
2757     background-color: #fff8f8;
2758     border-left: 4px solid #ff5252;
2759     padding: 15px;
2760     border-radius: 4px;
2761     box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
2762 }
2763
2764 .error-alert h3 {
2765     color: #d32f2f;
2766     margin-top: 0;
2767     margin-bottom: 10px;
2768     font-size: 16px;
2769 }
2770
2771 .error-alert ul {
```

```
2772     margin: 10px 0;
2773     padding-left: 20px;
2774 }
2775
2776 .error-alert li {
2777     margin-bottom: 5px;
2778     color: #616161;
2779 }
2780
2781 /* 数据真实性提示样式 */
2782 .data-note {
2783     background-color: #f0f4ff;
2784     border-left: 4px solid #5c6bc0;
2785     padding: 10px 15px;
2786     margin-bottom: 20px;
2787     border-radius: 4px;
2788     font-size: 14px;
2789 }
2790
2791 .estimated-data-tag {
2792     display: inline-block;
2793     background-color: #ffecb3;
2794     color: #795548;
2795     font-size: 11px;
2796     padding: 2px 6px;
2797     border-radius: 10px;
2798     margin-left: 6px;
2799     vertical-align: middle;
2800     font-weight: bold;
2801 }
2802
2803 /* 能源消耗详情 */
2804 .energy-breakdown {
2805     background-color: #f5f7fa;
2806     border-radius: 6px;
2807     padding: 15px;
2808 }
2809
2810 .energy-title {
2811     font-size: 16px;
2812     font-weight: 500;
2813     color: #333;
2814     margin-bottom: 12px;
2815 }
2816
2817 .energy-items {
2818     display: flex;
2819     flex-wrap: wrap;
2820     gap: 15px;
2821 }
2822
2823 .energy-item {
2824     flex: 1;
2825     min-width: 120px;
2826     background-color: #fff;
```

```
2827 padding: 10px;
2828 border-radius: 6px;
2829 box-shadow: 0 1px 3px rgba(0, 0, 0, 0.05);
2830 }
2831
2832 .energy-label {
2833   font-size: 13px;
2834   color: #666;
2835   margin-bottom: 5px;
2836 }
2837
2838 .energy-value {
2839   font-size: 15px;
2840   font-weight: 600;
2841   color: #2c3e50;
2842 }
2843
2844 /* 详情卡片样式 */
2845 .detail-card {
2846   flex: 1;
2847   min-width: 300px;
2848   background-color: #fff;
2849   border-radius: 8px;
2850   box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
2851   overflow: hidden;
2852 }
2853
2854 .detail-header {
2855   padding: 15px 20px;
2856   background-color: #f9f9f9;
2857   border-bottom: 1px solid #eee;
2858 }
2859
2860 .detail-title {
2861   font-size: 18px;
2862   font-weight: 600;
2863   color: #333;
2864 }
2865
2866 .detail-content {
2867   padding: 20px;
2868 }
2869
2870 /* 托管提供商信息 */
2871 .provider-info {
2872   display: flex;
2873   align-items: flex-start;
2874 }
2875
2876 .provider-icon {
2877   margin-right: 15px;
2878   color: #555;
2879   padding-top: 5px;
2880 }
2881
```

```
2882 .provider-details {
2883     flex: 1;
2884 }
2885
2886 .provider-name {
2887     font-size: 18px;
2888     font-weight: 600;
2889     color: #2c3e50;
2890     margin-bottom: 10px;
2891 }
2892
2893 .provider-stats {
2894     background-color: #f5f5f5;
2895     border-radius: 6px;
2896     padding: 12px 15px;
2897 }
2898
2899 .provider-item {
2900     display: flex;
2901     justify-content: space-between;
2902     margin-bottom: 10px;
2903     font-size: 14px;
2904 }
2905
2906 .provider-label {
2907     color: #555;
2908     font-weight: 500;
2909 }
2910
2911 .provider-value {
2912     font-weight: 600;
2913     color: #2c3e50;
2914 }
2915
2916 /* 移动端响应式 */
2917 @media (max-width: 768px) {
2918     .result-summary {
2919         flex-direction: column;
2920     }
2921
2922     .total-emission,
2923     .provider-info {
2924         flex-direction: column;
2925     }
2926
2927     .carbon-icon,
2928     .provider-icon {
2929         margin-right: 0;
2930         margin-bottom: 15px;
2931     }
2932
2933     .energy-items {
2934         flex-direction: column;
2935     }
2936 }
```

```
2937     .energy-item {
2938         width: 100%;
2939     }
2940 }
2941
2942 /* 绿色托管信息卡片样式 */
2943 .green-hosting .detail-content {
2944     padding: 15px;
2945 }
2946
2947 .green-hosting .provider-info {
2948     margin-bottom: 20px;
2949 }
2950
2951 .green-hosting .provider-stats {
2952     background-color: #f5f7fa;
2953     border-radius: 8px;
2954     padding: 15px;
2955     margin-top: 15px;
2956 }
2957
2958 .green-value {
2959     color: #34c759 !important;
2960     font-weight: bold;
2961 }
2962
2963 .green-hosting-info {
2964     background-color: #f0f9f4;
2965     border-radius: 8px;
2966     padding: 12px 15px;
2967     border-left: 4px solid #34c759;
2968     margin-top: 15px;
2969 }
2970
2971 .green-hosting-info p {
2972     margin: 0;
2973     color: #2c3e50;
2974     font-size: 14px;
2975     line-height: 1.5;
2976 }
2977
2978 /* 新的卡片布局样式 */
2979 .result-card-container {
2980     display: flex;
2981     flex-direction: column;
2982     gap: 30px;
2983     width: 100%;
2984     margin-bottom: 40px;
2985 }
2986
2987 .result-row {
2988     display: flex;
2989     flex-wrap: wrap;
2990     gap: 30px;
2991     width: 100%;
```

```
2992 }
2993
2994 .result-card {
2995     flex: 1;
2996     min-width: 320px;
2997     background: white;
2998     border-radius: 16px;
2999     box-shadow: 0 10px 25px rgba(0, 0, 0, 0.08);
3000     transition: all 0.4s cubic-bezier(0.175, 0.885, 0.32, 1.275);
3001     position: relative;
3002     overflow: hidden;
3003     border: 1px solid rgba(230, 230, 230, 0.7);
3004     display: flex;
3005     flex-direction: column;
3006     padding: 0;
3007 }
3008
3009 .result-card::before {
3010     content: '';
3011     position: absolute;
3012     top: 0;
3013     left: 0;
3014     right: 0;
3015     height: 4px;
3016     background: linear-gradient(to right, #34c759, #32ade6);
3017     opacity: 0.8;
3018 }
3019
3020 .result-card:hover {
3021     transform: translateY(-8px);
3022     box-shadow: 0 20px 40px rgba(0, 0, 0, 0.12);
3023 }
3024
3025 .major-card {
3026     min-height: 380px;
3027 }
3028
3029 .wide-card {
3030     width: 100%;
3031     flex-basis: 100%;
3032 }
3033
3034 .card-header {
3035     display: flex;
3036     justify-content: space-between;
3037     align-items: center;
3038     padding: 20px 25px;
3039     border-bottom: 1px solid #f0f0f0;
3040     background-color: #fcfcfc;
3041 }
3042
3043 .card-header h3 {
3044     margin: 0;
3045     font-size: 18px;
3046     font-weight: 600;
```

```
3047     color: #333;
3048     position: relative;
3049 }
3050
3051 .card-icon {
3052     width: 38px;
3053     height: 38px;
3054     border-radius: 10px;
3055     display: flex;
3056     align-items: center;
3057     justify-content: center;
3058     background: linear-gradient(135deg, #f5f7fa, #e4e7eb);
3059     color: #32ade6;
3060     box-shadow: 0 4px 12px rgba(0, 0, 0, 0.06);
3061 }
3062
3063 .result-card .details,
3064 .result-card .carbon-stats,
3065 .result-card .performance-metrics,
3066 .result-card .suggestion-list,
3067 .result-card .energy-chart,
3068 .result-card .detail-content {
3069     padding: 25px;
3070     flex: 1;
3071     display: flex;
3072     flex-direction: column;
3073 }
3074
3075 /* 详情项目样式 */
3076 .detail-item {
3077     display: flex;
3078     justify-content: space-between;
3079     align-items: center;
3080     padding: 12px 0;
3081     border-bottom: 1px solid #f0f0f0;
3082 }
3083
3084 .detail-item:last-child {
3085     border-bottom: none;
3086 }
3087
3088 .detail-label {
3089     color: #666;
3090     font-weight: 500;
3091     font-size: 14px;
3092 }
3093
3094 .detail-value {
3095     font-weight: 600;
3096     color: #333;
3097     background: linear-gradient(135deg, #f5f7fa, #e4e7eb);
3098     padding: 6px 12px;
3099     border-radius: 6px;
3100     box-shadow: 0 2px 6px rgba(0, 0, 0, 0.05);
3101     font-size: 14px;
```



```
3102 }
3103
3104 .highlight-text {
3105     color: #34c759;
3106     font-weight: bold;
3107 }
3108
3109 /* 卡片摘要样式 */
3110 .card-summary {
3111     padding: 20px 25px;
3112     background-color: #f9f9f9;
3113     margin-top: auto;
3114     border-top: 1px solid #f0f0f0;
3115 }
3116
3117 .card-summary p {
3118     margin: 10px 0 0;
3119     color: #666;
3120     font-size: 14px;
3121 }
3122
3123 .summary-badge {
3124     display: inline-block;
3125     padding: 6px 12px;
3126     border-radius: 20px;
3127     font-size: 14px;
3128     font-weight: 600;
3129     margin-bottom: 5px;
3130 }
3131
3132 .green-badge {
3133     background-color: rgba(52, 199, 89, 0.15);
3134     color: #34c759;
3135 }
3136
3137 .standard-badge {
3138     background-color: rgba(255, 149, 0, 0.15);
3139     color: #ff9500;
3140 }
3141
3142 /* 能源消耗网格 */
3143 .energy-consumption-grid {
3144     display: grid;
3145     grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));
3146     gap: 20px;
3147     padding: 25px;
3148 }
3149
3150 .energy-consumption-item {
3151     background: linear-gradient(135deg, #f9f9f9, #f5f5f5);
3152     border-radius: 12px;
3153     padding: 20px;
3154     display: flex;
3155     flex-direction: column;
3156     align-items: center;
```

```
3157     text-align: center;
3158     box-shadow: 0 4px 15px rgba(0, 0, 0, 0.05);
3159     transition: transform 0.3s ease;
3160 }
3161
3162 .energy-consumption-item:hover {
3163     transform: translateY(-5px);
3164 }
3165
3166 .consumption-icon {
3167     width: 50px;
3168     height: 50px;
3169     border-radius: 25px;
3170     margin-bottom: 15px;
3171     background-position: center;
3172     background-repeat: no-repeat;
3173     background-size: 25px;
3174     box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
3175 }
3176
3177 .data-center-icon {
3178     background-color: rgba(52, 199, 89, 0.15);
3179     background-image: url('data:image/svg+xml;utf8,<svg
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="%2334c759">
<path d="M4 1h16c1.1 0 2 .9 2 2v1c0 .55-.45 1-1 1H3c-.55 0-1-.45-1-1V3c0-
1.1.9-2 2-2zm0 5h16c1.1 0 2 .9 2 2v1c0 .55-.45 1-1 1H3c-.55 0-1-.45-1-1V8c0-
1.1.9-2 2-2zm0 5h16c1.1 0 2 .9 2 2v1c0 .55-.45 1-1 1H3c-.55 0-1-.45-1-1v-
1c0-1.1.9-2 2-2z"/></svg>');
3180 }
3181
3182 .network-icon {
3183     background-color: rgba(90, 200, 250, 0.15);
3184     background-image: url('data:image/svg+xml;utf8,<svg
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="%235ac8fa">
<path d="M1 9l2 2c4.97-4.97 13.03-4.97 18 0l2-2C16.93 2.93 7.08 2.93 1 9zm8
8l3 3 3-3c-1.65-1.66-4.34-1.66-6 0zm-4-4l2 2c2.76-2.76 7.24-2.76 10 0l2-
2C15.14 9.14 8.87 9.14 5 13z"/></svg>');
3185 }
3186
3187 .device-icon {
3188     background-color: rgba(255, 149, 0, 0.15);
3189     background-image: url('data:image/svg+xml;utf8,<svg
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="%23ff9500">
<path d="M4 18h16c1.1 0 1.99-.9 1.99-2L22 5c0-1.1-.9-2-2-2H4c-1.1 0-2 .9-2
2v11c0 1.1.9 2 2 2zM4 5h16v11H4V5zm8 14h2v-2h-2v2zm-4 0h2v-2H8v2zm8 0h2v-2h-
2v2z"/></svg>');
3190 }
3191
3192 .consumption-value {
3193     font-size: 18px;
3194     font-weight: 700;
3195     color: #333;
3196     margin-bottom: 5px;
3197 }
3198
```

```
3199 .consumption-label {
3200     font-size: 14px;
3201     color: #666;
3202 }
3203
3204 /* 碳排放详情 */
3205 .carbon-impact .carbon-stats {
3206     padding: 20px 25px;
3207 }
3208
3209 .carbon-annual-impact {
3210     margin-top: 20px;
3211     padding: 20px;
3212     background: linear-gradient(135deg, #f0f9f4, #e6f7eb);
3213     border-radius: 12px;
3214     text-align: center;
3215     box-shadow: 0 4px 15px rgba(0, 0, 0, 0.05);
3216 }
3217
3218 .impact-title {
3219     font-size: 15px;
3220     font-weight: 600;
3221     color: #333;
3222     margin-bottom: 10px;
3223 }
3224
3225 .impact-value {
3226     font-size: 24px;
3227     font-weight: 700;
3228     color: #34c759;
3229     margin-bottom: 5px;
3230 }
3231
3232 .impact-equivalent {
3233     font-size: 14px;
3234     color: #666;
3235     line-height: 1.6;
3236 }
3237
3238 /* 碳排放评分 */
3239 .total-emission-highlight {
3240     background: linear-gradient(135deg, #f0f9f4, #e6f7eb);
3241     padding: 25px;
3242     margin: 20px 25px;
3243     border-radius: 12px;
3244     text-align: center;
3245     animation: pulse 3s infinite alternate;
3246 }
3247
3248 @keyframes pulse {
3249     0% { box-shadow: 0 4px 15px rgba(52, 199, 89, 0.1); }
3250     100% { box-shadow: 0 4px 25px rgba(52, 199, 89, 0.25); }
3251 }
3252
3253 .highlight-value {
```

```
3254     font-size: 28px;
3255     font-weight: 700;
3256     color: #34c759;
3257     margin-bottom: 5px;
3258 }
3259
3260 .highlight-value span {
3261     font-size: 16px;
3262     font-weight: 500;
3263     color: #666;
3264 }
3265
3266 .highlight-label {
3267     font-size: 14px;
3268     color: #666;
3269 }
3270
3271 /* 数据源信息 */
3272 .data-source-info {
3273     padding: 10px 25px;
3274     background-color: #f9f9f9;
3275     color: #888;
3276     font-size: 13px;
3277     font-style: italic;
3278     margin-top: auto;
3279     border-top: 1px solid #f0f0f0;
3280 }
3281
3282 /* 响应式调整 */
3283 @media (max-width: 1024px) {
3284     .result-row {
3285         flex-direction: column;
3286     }
3287
3288     .result-card {
3289         width: 100%;
3290         min-width: 100%;
3291     }
3292 }
3293
3294 @media (max-width: 768px) {
3295     .energy-consumption-grid {
3296         grid-template-columns: 1fr;
3297     }
3298
3299     .score-container {
3300         flex-direction: column;
3301     }
3302
3303     .carbon-score-card {
3304         width: 100%;
3305         margin-bottom: 15px;
3306     }
3307 }
```

```
3308 </style>
```

```
1  /**
2   * GreenWeb后端代理服务器
3   * 用于处理跨域请求和获取网站性能数据
4   */
5
6  const express = require('express');
7  const axios = require('axios');
8  const cors = require('cors');
9  const dns = require('dns').promises;
10 const { performance } = require('perf_hooks');
11 const { URL } = require('url');
12 const geoip = require('geoip-lite');
13 const whois = require('whois-json');
14 const useragent = require('express-useragent');
15 const path = require('path');
16 const https = require('https');
17 const os = require('os');
18 const { execSync } = require('child_process');
19 const puppeteer = require('puppeteer');
20 const fetch = require('node-fetch'); // 用于API请求
21
22 // 声明chromeLauncher变量，稍后通过动态导入获取
23 let chromeLauncher;
24
25 // 异步初始化函数用于导入ESM模块
26 async function initializeESModules() {
27   try {
28     // 动态导入chrome-launcher
29     chromeLauncher = await import('chrome-launcher');
30     console.log('Chrome Launcher模块已成功导入');
31   } catch (err) {
32     console.error('导入ESM模块失败:', err);
33   }
34 }
35
36 // 执行初始化
37 initializeESModules().catch(err => {
38   console.error('初始化ESM模块时出错:', err);
39 });
40
41 const app = express();
42 const PORT = process.env.PORT || 3000;
43
44 // 中间件
45 app.use(cors());
46 app.use(express.json());
47 app.use(useragent.express());
48 app.use(express.static(path.join(__dirname, '../dist')));
49
50 // 创建不验证SSL证书的axios实例
51 const axiosInstance = axios.create({
52   httpsAgent: new https.Agent({
53     rejectUnauthorized: false
```

```
54     }),
55     timeout: 10000
56   });
57
58   // 全局常量 - 用于碳排放计算
59   const GLOBAL_CONSTANTS = {
60     // 能源消耗常量
61     averageEnergyConsumption: 1.805, // kWh/GB
62     greenEnergyCarbonIntensity: 50, // gCO2e/kWh
63     averageCarbonIntensity: 475, // gCO2e/kWh
64
65     // 数据中心效率
66     averagePUE: 1.67, // 电能使用效率
67     bestPUE: 1.1,
68
69     // 传输和设备能耗
70     averageTransmissionPerGB: 0.06, // kWh/GB
71     averageDevicePerGB: 0.08, // kWh/GB
72
73     // 缓存效率
74     cachingEfficiency: 0.2, // 20%的流量被缓存
75
76     // 访问量估算
77     averageMonthlyVisits: 10000,
78
79     // 碳中和阈值
80     greenEnergyThreshold: 80, // 80%以上可再生能源视为碳中和
81
82     // 每年一棵树可吸收二氧化碳量 (kg)
83     treeCO2PerYear: 25
84   };
85
86   // 国家碳强度数据 (gCO2e/kWh)
87   const COUNTRY CARBON_INTENSITY = {
88     'US': 383,
89     'CN': 554,
90     'IN': 739,
91     'JP': 478,
92     'DE': 344,
93     'GB': 231,
94     'FR': 56,
95     'IT': 331,
96     'BR': 87,
97     'CA': 135,
98     'KR': 415,
99     'RU': 351,
100    'AU': 656,
101    'ES': 200,
102    'MX': 428,
103    'ID': 736,
104    'NL': 358,
105    'SA': 523,
106    'CH': 24,
107    'TR': 461,
108    'SE': 13,
```

```
109     'PL': 751,
110     'BE': 161,
111     'TH': 471,
112     'AT': 109,
113     'IE': 291,
114     'SG': 418,
115     'IL': 529,
116     'DK': 135,
117     'FI': 89,
118     'NO': 8,
119     // 默认值将使用全球平均值
120 };
121
122 // 主要云服务提供商信息
123 const PROVIDER_INFO = {
124     'aws': {
125         name: 'Amazon Web Services',
126         renewableRange: [60, 85],
127         renewableChance: 0.8,
128         pueRange: [1.1, 1.4],
129         serverEfficiency: 1.2,
130         regions: ['us-east', 'us-west', 'eu-west', 'eu-central', 'ap-northeast',
131 'ap-southeast', 'sa-east']
132     },
133     'azure': {
134         name: 'Microsoft Azure',
135         renewableRange: [65, 90],
136         renewableChance: 0.85,
137         pueRange: [1.1, 1.35],
138         serverEfficiency: 1.25,
139         regions: ['us-east', 'us-west', 'eu-west', 'eu-north', 'asia-east',
140 'asia-southeast', 'australia-east']
141     },
142     'google': {
143         name: 'Google Cloud',
144         renewableRange: [80, 95],
145         renewableChance: 0.9,
146         pueRange: [1.1, 1.2],
147         serverEfficiency: 1.3,
148         regions: ['us-central', 'us-east', 'us-west', 'europe-west', 'europe-
149 north', 'asia-east', 'asia-south']
150     },
151     'cloudflare': {
152         name: 'Cloudflare',
153         renewableRange: [70, 90],
154         renewableChance: 0.8,
155         pueRange: [1.1, 1.3],
156         serverEfficiency: 1.2,
157         regions: ['global-edge']
158     },
159     'alibaba': {
160         name: 'Alibaba Cloud',
161         renewableRange: [45, 70],
162         renewableChance: 0.6,
163         pueRange: [1.3, 1.6],
```



```
161     serverEfficiency: 1.1,
162     regions: ['cn-hangzhou', 'cn-shanghai', 'cn-beijing', 'us-west', 'eu-
central', 'ap-southeast']
163   },
164   'tencent': {
165     name: 'Tencent Cloud',
166     renewableRange: [40, 65],
167     renewableChance: 0.5,
168     pueRange: [1.35, 1.65],
169     serverEfficiency: 1.0,
170     regions: ['ap-guangzhou', 'ap-shanghai', 'ap-beijing', 'na-
siliconvalley', 'eu-frankfurt', 'ap-singapore']
171   },
172   'other': {
173     name: 'Unknown Provider',
174     renewableRange: [30, 60],
175     renewableChance: 0.4,
176     pueRange: [1.5, 1.9],
177     serverEfficiency: 0.9,
178     regions: ['unknown']
179   }
180 };
181
182 /**
183  * 健康检查端点
184  */
185 app.get('/api/health', (req, res) => {
186   res.json({ status: 'ok', timestamp: new Date().toISOString() });
187 });
188
189 /**
190  * 获取服务器位置信息
191  */
192 app.get('/api/location', async (req, res) => {
193   try {
194     const { domain } = req.query;
195
196     if (!domain) {
197       return res.status(400).json({ error: '缺少域名参数' });
198     }
199
200     // 使用DNS查询获取IP地址
201     const addresses = await dns.lookup(domain, { all: true });
202     const ip = addresses[0]?.address;
203
204     if (!ip) {
205       return res.status(404).json({ error: '无法解析域名' });
206     }
207
208     // 使用GeoIP获取位置信息
209     const geo = geoip.lookup(ip);
210
211     // 尝试使用WHOIS获取额外信息
212     let whoisData = {};
213     try {
```

```
214     whoisData = await whois(domain);
215   } catch (whoisError) {
216     console.error('WHOIS查询失败:', whoisError);
217   }
218
219   res.json({
220     ip,
221     country: geo?.country || null,
222     region: geo?.region || null,
223     city: geo?.city || null,
224     timezone: geo?.timezone || null,
225     coordinates: geo?.ll || null,
226     org: geo?.org || null,
227     registrar: whoisData.registrar || null,
228     registrantCountry: whoisData.registrantCountry || null
229   });
230 } catch (error) {
231   console.error('位置查询错误:', error);
232   res.status(500).json({ error: '位置查询失败', message: error.message });
233 }
234 });
235
236 /**
237  * 获取HTTP响应头信息
238  */
239 app.get('/api/headers', async (req, res) => {
240   try {
241     const { url } = req.query;
242
243     if (!url) {
244       return res.status(400).json({ error: '缺少URL参数' });
245     }
246
247     const startTime = performance.now();
248     const response = await axiosInstance.head(url, {
249       maxRedirects: 5,
250       validateStatus: null
251     });
252     const endTime = performance.now();
253
254     res.json({
255       status: response.status,
256       statusText: response.statusText,
257       headers: response.headers,
258       responseTime: endTime - startTime
259     });
260   } catch (error) {
261     console.error('头信息获取错误:', error);
262     res.status(500).json({ error: '头信息获取失败', message: error.message });
263   }
264 });
265
266 /**
267  * 测量页面大小
268  */
```

```
269 app.get('/api/size', async (req, res) => {
270   try {
271     const { url } = req.query;
272
273     if (!url) {
274       return res.status(400).json({ error: '缺少URL参数' });
275     }
276
277     const startTime = performance.now();
278     const response = await axiosInstance.get(url, {
279       maxRedirects: 5,
280       validateStatus: null,
281       responseType: 'arraybuffer' // 获取二进制响应以精确计算大小
282     });
283     const endTime = performance.now();
284
285     // 计算页面大小 (KB)
286     const contentLength = response.headers['content-length'] ||
response.data.length;
287     const sizeInKB = Math.round(contentLength / 1024);
288
289     res.json({
290       size: sizeInKB,
291       contentType: response.headers['content-type'],
292       responseTime: endTime - startTime
293     });
294   } catch (error) {
295     console.error('页面大小测量错误:', error);
296     res.status(500).json({ error: '页面大小测量失败', message: error.message
});
297   }
298 });
299
300 /**
301  * 分析服务提供商
302  */
303 app.get('/api/provider', async (req, res) => {
304   try {
305     const { domain } = req.query;
306
307     if (!domain) {
308       return res.status(400).json({ error: '缺少域名参数' });
309     }
310
311     // 使用DNS和WHOIS信息尝试确定服务提供商
312     const addresses = await dns.lookup(domain, { all: true });
313     const ip = addresses[0]?.address;
314
315     if (!ip) {
316       return res.status(404).json({ error: '无法解析域名' });
317     }
318
319     // 使用GeoIP获取组织信息
320     const geo = geoip.lookup(ip);
321     const org = geo?.org || '';
```

```

322
323 // 尝试确定服务提供商
324 let provider = 'other';
325
326 if (org.includes('AMAZON') || org.includes('AWS')) {
327     provider = 'aws';
328 } else if (org.includes('MICROSOFT') || org.includes('MSFT')) {
329     provider = 'azure';
330 } else if (org.includes('GOOGLE')) {
331     provider = 'google';
332 } else if (org.includes('ALIBABA') || org.includes('ALIYUN')) {
333     provider = 'alibaba';
334 } else if (org.includes('TENCENT')) {
335     provider = 'tencent';
336 } else if (org.includes('CLOUDFLARE')) {
337     provider = 'cloudflare';
338 }
339
340 // 获取提供商详细信息
341 const providerInfo = PROVIDER_INFO[provider] || PROVIDER_INFO.other;
342
343 // 估计可再生能源使用比例
344 const [minRenewable, maxRenewable] = providerInfo.renewableRange;
345 const renewablePercentage = Math.floor(Math.random() * (maxRenewable -
minRenewable + 1)) + minRenewable;
346
347 // 计算数据中心PUE（电能使用效率）
348 const [minPUE, maxPUE] = providerInfo.pueRange;
349 const dataCenterPUE = parseFloat(safeToFixed(Math.random() * (maxPUE -
minPUE) + minPUE, 2));
350
351 res.json({
352     provider,
353     providerName: providerInfo.name,
354     ip,
355     org,
356     renewablePercentage,
357     pue: dataCenterPUE,
358     serverEfficiency: providerInfo.serverEfficiency
359 });
360 } catch (error) {
361     console.error('提供商分析错误:', error);
362     res.status(500).json({ error: '提供商分析失败', message: error.message });
363 }
364 });
365 /**
366  * 修改性能测量API端点，添加WebPageTest选项
367  */
368 app.get('/api/performance', async (req, res) => {
369     try {
370         const url = req.query.url;
371         const requestedMethod = req.query.browser || 'auto';
372
373         if (!url) {
374             return res.status(400).json({ error: '请提供URL' });

```

```

    }

    console.log(`接收到性能分析请求: ${url}, 方法: ${requestedMethod}`);

    let result;
    let allErrors = [];

    // 尝试基础HTTP分析
    try {
        if (requestedMethod === 'basic' || requestedMethod === 'basic-http' ||
requestedMethod === 'auto') {
            console.log('尝试使用基础HTTP方法分析 ... ');
            result = await measurePerformanceBasic(url);

            if (result.success) {
                console.log('基础HTTP分析成功');
                // 发起碳排放计算请求
                try {
                    const carbonParams = new URLSearchParams({
                        pageSize: result.performance.pageSize,
                        requestCount: result.performance.requestCount,
                        domainCount: result.performance.domainCount,
                        responseTime: result.performance.responseTime,
                        hasCompression: result.headers.supportsCompression,
                        resourceStats:
JSON.stringify(result.performance.resourceStats)
                    });

                    const carbonResult = await
axios.get(`http://localhost:${PORT}/api/carbon?${carbonParams}`);

                    if (carbonResult.data && carbonResult.data.measurable) {
                        result.carbonEmission = carbonResult.data;
                    }
                } catch (carbonError) {
                    console.error('碳排放计算错误:', carbonError);
                }

                // 清除任何非真实测量的数据
                for (const metric in result.performance) {
                    if (result.performance[metric] === null ||
result.performance[metric] === undefined) {
                        delete result.performance[metric];
                    }
                }

                // 只返回真实测量的数据
                return res.json({
                    success: true,
                    performance: result.performance,
                    headers: result.headers,
                    carbonEmission: result.carbonEmission || { measurable: false },
                    measurementMethod: result.measurementMethod,
                    measuredBy: 'basic-http',
                    allDataIsReal: true // 标记所有数据都是真实的
                });
            }
        }
    } catch (error) {
        console.error('基础HTTP分析失败:', error);
    }
}

// 尝试使用基础HTTP方法分析
function measurePerformanceBasic(url) {
    return new Promise((resolve, reject) => {
        // 发起性能分析请求
        axios.get(url)
            .then(response => {
                const result = {
                    success: true,
                    performance: {},
                    headers: response.headers,
                    measurementMethod: 'basic-http',
                    measuredBy: 'basic-http'
                };

                // 处理性能数据
                const performanceData = response.data;
                if (performanceData) {
                    result.performance = performanceData;
                }

                // 处理碳排放数据
                const carbonData = response.data?.carbon;
                if (carbonData) {
                    result.carbonEmission = carbonData;
                }

                resolve(result);
            })
            .catch(error => {
                reject(error);
            });
    });
}

```

```

426         });
427     } else {
428         console.log('基础HTTP分析失败:', result.error);
429         allErrors.push(result.error || '基础HTTP分析失败, 无具体错误信息');
430     }
431 }
432 } catch (basicError) {
433     console.error('基础HTTP分析抛出异常:', basicError);
434     allErrors.push(`基础HTTP分析异常: ${basicError.message}`);
435 }
436
437 // 如果所有方法都失败, 返回基础HTTP分析结果或错误信息
438 return res.status(500).json({
439     success: false,
440     error: '所有分析方法都失败',
441     details: allErrors,
442     measurable: false
443 });
444 } catch (error) {
445     console.error('性能测量API错误:', error);
446     res.status(500).json({
447         success: false,
448         error: '性能测量API发生错误',
449         message: error.message,
450         measurable: false
451     });
452 }
453 });
454
455 /**
456  * 碳排放计算 - 优化使用实际测量数据
457  */
458 app.get('/api/carbon', async (req, res) => {
459     try {
460         const {
461             pageSize,
462             country,
463             renewablePercentage,
464             pue,
465             requestCount,
466             domainCount,
467             resourceStats,
468             responseTime,
469             hasCompression
470         } = req.query;
471
472         // 检查必须的真实测量数据
473         if (!pageSize || parseInt(pageSize) <= 0) {
474             return res.status(400).json({
475                 error: '缺少页面大小参数或值为零',
476                 measurable: false,
477                 message: '无法获取页面大小数据, 无法计算能源消耗'
478             });
479         }
480

```

```
481 // 检查其他必要的真实数据
482 if (!requestCount || !domainCount) {
483     return res.status(400).json({
484         error: '缺少网络请求数据',
485         measurable: false,
486         message: '无法获取网络请求和域名数量, 无法准确计算碳排放'
487     });
488 }
489
490 // 使用实际测量值, 不进行估算
491 const pageSizeKB = parseInt(pageSize);
492 const countryCode = country || null; // 不提供默认值, 如果没有则返回错误
493 const renewable = parseFloat(renewablePercentage);
494 const dataCenterPUE = parseFloat(safeToFixed(pue, 2));
495 const actualRequestCount = parseInt(requestCount);
496 const actualDomainCount = parseInt(domainCount);
497 const actualResponseTime = parseInt(responseTime) || 0;
498 const isCompressed = hasCompression === 'true' || hasCompression ===
true;
499
500 // 检查国家信息
501 if (!countryCode) {
502     return res.status(400).json({
503         error: '缺少国家/地区信息',
504         measurable: false,
505         message: '无法获取服务器地理位置, 无法准确计算碳排放'
506     });
507 }
508
509 // 检查可再生能源信息
510 if (isNaN(renewable)) {
511     return res.status(400).json({
512         error: '缺少可再生能源使用比例',
513         measurable: false,
514         message: '无法获取服务器使用的可再生能源比例, 无法准确计算碳排放'
515     });
516 }
517
518 // 检查PUE信息
519 if (isNaN(dataCenterPUE)) {
520     return res.status(400).json({
521         error: '缺少数据中心PUE',
522         measurable: false,
523         message: '无法获取数据中心PUE, 无法准确计算碳排放'
524     });
525 }
526
527 // 解析资源统计数据 - 只使用真实值
528 let resourceStatsData = {};
529 if (resourceStats) {
530     try {
531         resourceStatsData = typeof resourceStats === 'string' ?
JSON.parse(resourceStats) : resourceStats;
532     } catch (e) {
533         console.error('解析资源统计数据失败:', e);
```



```
534         return res.status(400).json({
535             error: '资源统计数据无效',
536             measurable: false,
537             message: '无法解析资源统计数据, 无法准确计算碳排放'
538         });
539     }
540 } else {
541     return res.status(400).json({
542         error: '缺少资源统计数据',
543         measurable: false,
544         message: '无法获取资源统计数据, 无法准确计算碳排放'
545     });
546 }
547
548 // 使用真实数据计算而不是估算
549 const pageSizeInGB = pageSizeKB / 1024 / 1024;
550
551 // 基于实际测量的缓存效率, 不使用估算值
552 const cacheControlHeader = req.query.cacheControl;
553 let measuredCacheEfficiency = 0;
554
555 // 只有当存在真实的Cache-Control头时才使用缓存效率
556 if (cacheControlHeader) {
557     const maxAge = /max-age=(\d+)/.exec(cacheControlHeader);
558     if (maxAge && maxAge[1]) {
559         const maxAgeValue = parseInt(maxAge[1]);
560         // 基于max-age值计算缓存效率
561         if (maxAgeValue > 86400) { // 1天以上
562             measuredCacheEfficiency = 0.6;
563         } else if (maxAgeValue > 3600) { // 1小时以上
564             measuredCacheEfficiency = 0.4;
565         } else if (maxAgeValue > 0) {
566             measuredCacheEfficiency = 0.2;
567         }
568     }
569 }
570
571 // 压缩效率 - 根据是否启用压缩 (这是真实测量的)
572 const compressionEfficiency = isCompressed ? 0.7 : 1.0;
573
574 // 能源强度计算 - 使用国际能源署的真实数据
575 const countryCarbonValue = COUNTRY_CARBON_INTENSITY[countryCode] ||
576 null;
577 if (!countryCarbonValue) {
578     return res.status(400).json({
579         error: '无法获取国家电网碳强度',
580         measurable: false,
581         message: `无法获取 ${countryCode} 的电网碳强度数据, 无法准确计算碳排放`
582     });
583 }
584
585 // 基于真实测量值计算能源消耗
586 const baseEnergyIntensity = GLOBAL_CONSTANTS.averageEnergyConsumption;
587 const energyIntensity = baseEnergyIntensity * dataCenterPUE;
```



```

588 // 计算能源消耗 - 确保即使页面很小也有最小值
589 // 使用Math.max确保能源消耗有一个最小基准值
590 const pageSizeInGBSafe = Math.max(pageSizeInGB, 0.0001); // 确保至少有
0.1MB
591 const dataCenterEnergy = pageSizeInGBSafe * (energyIntensity /
dataCenterPUE);
592 const transmissionEnergy = pageSizeInGBSafe *
GLOBAL_CONSTANTS.averageTransmissionPerGB;
593 const deviceEnergy = pageSizeInGBSafe *
GLOBAL_CONSTANTS.averageDevicePerGB;
594
595 // 数据中心碳排放 - 考虑可再生能源比例
596 const dataTransferCarbon = dataCenterEnergy * (
597     (renewable / 100) * GLOBAL_CONSTANTS.greenEnergyCarbonIntensity +
598     ((100 - renewable) / 100) * countryCarbonValue
599 );
600
601 // 网络传输和客户设备碳排放
602 const networkCarbon = transmissionEnergy * countryCarbonValue;
603 const clientCarbon = deviceEnergy * countryCarbonValue;
604
605 // 计算总碳排放量 (单位: g CO2e)
606 const totalCarbonEmission = dataTransferCarbon + networkCarbon +
clientCarbon;
607
608 // 估计月访问量 - 基于固定值, 但明确标记这不是测量值
609 const monthlyCarbonEmission = (totalCarbonEmission *
GLOBAL_CONSTANTS.averageMonthlyVisits) / 1000; // 单位: kg CO2e
610 const annualCarbonEmission = monthlyCarbonEmission * 12;
611
612 // 计算碳中和所需的树木数量
613 const treesNeeded = Math.ceil(annualCarbonEmission /
GLOBAL_CONSTANTS.treeCO2PerYear);
614
615 // 碳足迹和能源效率评分 - 基于真实测量值计算
616 const carbonFootprintScore = Math.min(100, Math.max(1,
617     (totalCarbonEmission / 0.5) * 20 + // 基础碳排放
618     (dataCenterPUE / GLOBAL_CONSTANTS.bestPUE - 1) * 20 + // 数据中心效率
619     ((100 - renewable) / 100) * 40 // 可再生能源使用
620 ));
621
622 const energyEfficiencyScore = Math.min(100, Math.max(1,
623     100 - (totalCarbonEmission / 3) * 20 - // 基础能源效率
624     (actualDomainCount - 1) * 3 // 域名数量惩罚 (实际测量)
625 ));
626
627 res.json({
628     measurable: true,
629     pageSize: pageSizeKB,
630     energyIntensity,
631     cachingEfficiency: measuredCacheEfficiency,
632     compressionEfficiency: isCompressed ? 0.3 : 0, // 压缩节省百分比 (实际测
量)
633     dataCenterEnergy,
634     transmissionEnergy,

```

```

635     deviceEnergy,
636     dataTransferCarbon,
637     networkCarbon,
638     clientCarbon,
639     totalCarbonEmission,
640     monthlyCarbonEmission,
641     annualCarbonEmission,
642     treesNeeded,
643     isGreen: renewable ≥ GLOBAL_CONSTANTS.greenEnergyThreshold,
644     carbonFootprintScore,
645     energyEfficiencyScore,
646     resourceBreakdown: resourceStatsData,
647     requestCount: actualRequestCount,
648     domainCount: actualDomainCount,
649     dataSourceInfo: {
650         allRealMeasurements: true,
651         estimatedValues: ['monthlyCarbonEmission', 'annualCarbonEmission',
        'treesNeeded'],
652         explanation: '月度和年度碳排放基于固定的月访问量估算，树木数量基于年均CO2吸收
        量估算。所有其他数据基于实际测量。'
653     }
654 });
655 } catch (error) {
656     console.error('碳排放计算错误:', error);
657     res.status(500).json({
658         error: '碳排放计算失败',
659         message: error.message,
660         measurable: false
661     });
662 }
663 });
664
665 /**
666  * 生成优化建议
667  */
668 app.get('/api/suggestions', async (req, res) => {
669     try {
670         const {
671             pageSize,
672             performance,
673             provider,
674             renewablePercentage,
675             pue,
676             totalCarbonEmission,
677             carbonFootprintScore,
678             energyEfficiencyScore,
679             treesNeeded,
680             requestCount,
681             domainCount
682         } = req.query;
683
684         // 提取性能数据
685         const performanceData = typeof performance === 'string' ?
        JSON.parse(performance) : (performance || {});
686

```

```
687 // 准备建议生成的数据
688 const data = {
689   pageSize: parseInt(pageSize) || 0,
690   performance: {
691     fcp: parseFloat(performanceData.fcp) || 1.5,
692     lcp: parseFloat(performanceData.lcp) || 2.5,
693     cls: parseFloat(performanceData.cls) || 0.05,
694     fid: parseFloat(performanceData.fid) || 100,
695     ttfb: parseFloat(performanceData.ttfb) || 200
696   },
697   provider: provider || 'other',
698   renewablePercentage: parseFloat(renewablePercentage) || 50,
699   pue: parseFloat(pue) || 1.67,
700   totalCarbonEmission: parseFloat(totalCarbonEmission) || 0,
701   carbonFootprintScore: parseFloat(carbonFootprintScore) || 0,
702   energyEfficiencyScore: parseFloat(energyEfficiencyScore) || 0,
703   treesNeeded: parseInt(treesNeeded) || 0,
704   requestCount: parseInt(requestCount) || 1,
705   domainCount: parseInt(domainCount) || 1
706 };
707
708 // 生成建议
709 const suggestions = generateOptimizationSuggestions(data);
710
711 res.json({ suggestions });
712 } catch (error) {
713   console.error('建议生成错误:', error);
714   res.status(500).json({ error: '建议生成失败', message: error.message });
715 }
716 });
717
718 /**
719  * 综合网站分析
720  */
721 app.get('/api/analyze', async (req, res) => {
722   try {
723     const { domain, browser = 'auto' } = req.query;
724
725     if (!domain) {
726       return res.status(400).json({ error: '缺少域名参数' });
727     }
728
729     // 确保域名格式正确
730     let url;
731     try {
732       if (domain.startsWith('http')) {
733         url = new URL(domain).href;
734       } else {
735         url = `https://${domain}`;
736       }
737     } catch (error) {
738       return res.status(400).json({ error: '无效的域名格式' });
739     }
740
741     // 并行执行所有分析任务
```

```
742     let [locationData, headerData, sizeData, providerData] = [null, null,
null, null];
743     let performanceData = null;
744
745     try {
746         [locationData, headerData, providerData] = await Promise.all([
747             axios.get(`http://localhost:${PORT}/api/location?
domain=${domain}`).then(resp => resp.data),
748             axios.get(`http://localhost:${PORT}/api/headers?
url=${encodeURIComponent(url)}`).then(resp => resp.data),
749             axios.get(`http://localhost:${PORT}/api/provider?
domain=${domain}`).then(resp => resp.data)
750         ]);
751     } catch (error) {
752         console.error('基础数据获取错误:', error);
753     }
754
755     // 单独执行性能测量，因为它可能需要更长时间
756     try {
757         performanceData = await
axios.get(`http://localhost:${PORT}/api/performance?
url=${encodeURIComponent(url)}&browser=${browser}`)
758         .then(resp => resp.data);
759
760         // 如果有性能数据，使用它的页面大小
761         if (performanceData && performanceData.pageSize) {
762             sizeData = { size: performanceData.pageSize };
763         }
764     } catch (error) {
765         console.error('性能测量错误:', error);
766         performanceData = { measurable: false, error: error.message };
767     }
768
769     // 如果没有页面大小数据，尝试单独获取
770     if (!sizeData) {
771         try {
772             sizeData = await axios.get(`http://localhost:${PORT}/api/size?
url=${encodeURIComponent(url)}`).
773             .then(resp => resp.data);
774         } catch (error) {
775             console.error('页面大小测量错误:', error);
776             sizeData = { measurable: false, error: error.message };
777         }
778     }
779
780     // 尝试计算碳排放（即使部分数据缺失）
781     let carbonData = null;
782     try {
783         const pageSize = sizeData?.size || 0;
784         const country = locationData?.country || null;
785         const renewablePercentage = providerData?.renewablePercentage || 50;
786         const pue = providerData?.pue || GLOBAL_CONSTANTS.averagePUE;
787
788         // 获取性能数据中的请求数和域名数
789         const requestCount = performanceData?.requestCount || 1;
```

```
790     const domainCount = performanceData?.domainCount || 1;
791
792     if (pageSize > 0) {
793         carbonData = await axios.get(`http://localhost:${PORT}/api/carbon`,
794 {
795             params: {
796                 pageSize,
797                 country,
798                 renewablePercentage,
799                 pue,
800                 requestCount,
801                 domainCount
802             }
803         }).then(resp => resp.data);
804     } else {
805         carbonData = { measurable: false, error: '缺少页面大小数据' };
806     }
807 } catch (error) {
808     console.error('碳排放计算错误:', error);
809     carbonData = { measurable: false, error: error.message };
810 }
811
812 // 只有当性能数据可用时才生成优化建议
813 let suggestionsData = { suggestions: [] };
814 if (performanceData && !performanceData.error) {
815     try {
816         suggestionsData = await
817 axios.get(`http://localhost:${PORT}/api/suggestions`, {
818     params: {
819         pageSize: sizeData?.size || 0,
820         performance: JSON.stringify(performanceData),
821         provider: providerData?.provider || 'other',
822         renewablePercentage: providerData?.renewablePercentage || 50,
823         pue: providerData?.pue || GLOBAL_CONSTANTS.averagePUE,
824         totalCarbonEmission: carbonData?.totalCarbonEmission || 0,
825         carbonFootprintScore: carbonData?.carbonFootprintScore || 0,
826         energyEfficiencyScore: carbonData?.energyEfficiencyScore || 0,
827         treesNeeded: carbonData?.treesNeeded || 0,
828         requestCount: performanceData?.requestCount || 1,
829         domainCount: performanceData?.domainCount || 1
830     }
831 }).then(resp => resp.data);
832 } catch (error) {
833     console.error('建议生成错误:', error);
834     suggestionsData = {
835         suggestions: ['由于性能数据获取失败，无法生成针对性优化建议。'],
836         measurable: false,
837         error: error.message
838     };
839 } else {
840     suggestionsData = {
841         suggestions: ['由于性能数据获取失败，无法生成针对性优化建议。'],
842         measurable: false
843     };
844 }
```

```

843     }
844
845     // 整合所有数据
846     const result = {
847         domain,
848         url,
849         browser,
850         timestamp: new Date().toISOString(),
851
852         // 位置信息
853         location: locationData || { measurable: false },
854
855         // 页面信息
856         pageSize: sizeData?.size || 0,
857         contentType: sizeData?.contentType || null,
858         headers: headerData?.headers || null,
859
860         // 服务提供商信息
861         provider: providerData?.provider || 'unknown',
862         providerName: providerData?.providerName || 'Unknown Provider',
863         renewablePercentage: providerData?.renewablePercentage || null,
864         pue: providerData?.pue || null,
865
866         // 性能指标
867         performance: performanceData || { measurable: false },
868
869         // 碳排放信息
870         energyIntensity: carbonData?.energyIntensity || null,
871         dataCenterEnergy: carbonData?.dataCenterEnergy || null,
872         transmissionEnergy: carbonData?.transmissionEnergy || null,
873         deviceEnergy: carbonData?.deviceEnergy || null,
874         dataTransferCarbon: carbonData?.dataTransferCarbon || null,
875         networkCarbon: carbonData?.networkCarbon || null,
876         clientCarbon: carbonData?.clientCarbon || null,
877         totalCarbonEmission: carbonData?.totalCarbonEmission || null,
878         monthlyCarbonEmission: carbonData?.monthlyCarbonEmission || null,
879         annualCarbonEmission: carbonData?.annualCarbonEmission || null,
880         treesNeeded: carbonData?.treesNeeded || null,
881         isGreen: carbonData?.isGreen || false,
882
883         // 优化建议
884         suggestions: suggestionsData.suggestions || []
885     };
886
887     res.json(result);
888 } catch (error) {
889     console.error('综合分析错误:', error);
890     res.status(500).json({
891         error: '综合分析失败',
892         message: error.message,
893         stack: process.env.NODE_ENV === 'development' ? error.stack :
undefined
894     });
895 }
896 });

```



```
897
898 /**
899  * 生成智能优化建议
900  * @param {Object} data - 性能和碳排放数据
901  * @returns {Array} 优化建议列表
902  */
903 function generateOptimizationSuggestions(data) {
904     // 如果没有性能数据，返回通用建议
905     if (!data.performance || !data.performance.lcp) {
906         return [
907             '无法获取性能指标，请确保网站可访问并且服务器配置正确',
908             '考虑使用CDN分发静态资源，减少数据传输距离和能耗',
909             '实施高效的HTTP缓存策略，延长缓存有效期减少重复请求',
910             '优化图片资源，考虑使用WebP或AVIF等新一代图片格式'
911         ];
912     }
913
914     const suggestions = [];
915
916     // 基于页面大小的建议
917     if (data.pageSize > 4000) {
918         suggestions.push('大幅压缩图片资源，当前页面大小过大('+data.pageSize+'KB)，严重影响加载速度和能源消耗');
919         suggestions.push('使用WebP或AVIF等新一代图片格式，可减少50-90%的图片大小');
920         suggestions.push('实施延迟加载(Lazy Loading)技术，仅加载可视区域内容');
921     } else if (data.pageSize > 2500) {
922         suggestions.push('压缩图片和媒体资源，减少页面大小('+data.pageSize+'KB)和传输量');
923         suggestions.push('优化JavaScript和CSS文件，减少不必要的代码');
924     } else if (data.pageSize > 1500) {
925         suggestions.push('考虑进一步优化资源大小('+data.pageSize+'KB)，提高页面加载速度');
926     }
927
928     // 基于域名数量的建议
929     if (data.domainCount > 10) {
930         suggestions.push('网站加载资源来自过多域名(${data.domainCount}个)，建议合并资源来源减少DNS查询和连接建立开销');
931     } else if (data.domainCount > 5) {
932         suggestions.push('考虑减少加载资源的域名数量(${data.domainCount}个)，以降低DNS查询时间');
933     }
934
935     // 基于请求数量的建议
936     if (data.requestCount > 100) {
937         suggestions.push('网页请求数过多(${data.requestCount}个)，严重影响加载速度，建议合并资源并减少不必要的API调用');
938     } else if (data.requestCount > 50) {
939         suggestions.push('网页请求数较多(${data.requestCount}个)，建议通过合并小文件减少HTTP请求');
940     }
941
942     // 基于性能指标的建议
943     if (data.performance.lcp > 2.5) {
```

```
944     suggestions.push(`优化最大内容绘制(LCP=${safeToFixed(data.performance.lcp,
945     2)}s), 重点优化主要内容元素的加载时间`);
946     if (data.performance.lcp > 5) {
947         suggestions.push('LCP值严重超标, 建议使用预加载(preload)关键资源并优化服务器响
948         应速度');
949     }
950     if (data.performance.cls > 0.1) {
951         suggestions.push(`减少累积布局偏移(CLS=${safeToFixed(data.performance.cls,
952         3)}), 预先设置图片和元素尺寸`);
953         if (data.performance.cls > 0.25) {
954             suggestions.push('CLS值严重超标, 检查是否有动态注入内容导致布局偏移, 为所有图片
955             和嵌入元素设置明确尺寸');
956         }
957     }
958     if (data.performance.ttfb > 300) {
959         suggestions.push(`优化服务器响应时间
960         (TTFB=${Math.round(data.performance.ttfb)}ms), 考虑使用边缘CDN或优化后端处理`);
961         if (data.performance.ttfb > 1000) {
962             suggestions.push('服务器响应时间过长, 建议使用服务端缓存, 优化数据库查询, 或升级
963             服务器配置');
964         }
965     }
966     if (data.performance.fid > 130) {
967         suggestions.push(`提高首次输入延迟
968         (FID=${Math.round(data.performance.fid)}ms), 减少主线程阻塞的JavaScript执行`);
969         if (data.performance.fid > 300) {
970             suggestions.push('输入延迟严重, 拆分长任务为较小任务, 并延迟加载非关键
971             JavaScript');
972         }
973     }
974     // 基于碳排放的建议
975     if (data.carbonFootprintScore > 70) {
976         suggestions.push(`当前网站碳足迹评分较高
977         (${Math.round(data.carbonFootprintScore)}分), 建议全面优化能源使用效率`);
978     }
979     if (data.energyEfficiencyScore < 50) {
980         suggestions.push(`能源效率评分较低
981         (${Math.round(data.energyEfficiencyScore)}分), 建议采用更现代的Web技术和优化策略
982         `);
983     }
984     if (data.renewablePercentage < GLOBAL_CONSTANTS.greenEnergyThreshold) {
985         suggestions.push(`当前服务器使用的可再生能源比例
986         (${data.renewablePercentage}%)偏低, 建议迁移至更环保的数据中心`);
987     }
988     if (data.pue > 1.5) {
989         suggestions.push(`当前数据中心PUE值(${safeToFixed(data.pue, 2)})较高, 选择更
990         高能效的服务提供商可降低碳排放`);
991     }
```



```
986     }
987
988     if (data.totalCarbonEmission > 1.5) {
989         suggestions.push(`当前页面单次访问碳排放
990 ($${safeToFixed(data.totalCarbonEmission, 2)}gCO2e)偏高，建议全面优化页面资源`);
991         if (data.totalCarbonEmission > 3) {
992             suggestions.push('碳排放量远高于平均水平，建议对页面进行全面性能审计并减少不必要的
993 的资源加载');
994         }
995     }
996
997     // 加入碳中和相关建议
998     if (data.treesNeeded > 5) {
999         suggestions.push(`抵消网站年度碳排放需要种植约${data.treesNeeded}棵树，建议考虑
1000 参与碳抵消项目或减少能源消耗`);
1001     }
1002
1003     // 通用绿色建议
1004     if (suggestions.length < 3) {
1005         suggestions.push('实施高效的HTTP缓存策略，延长缓存有效期减少重复请求');
1006         suggestions.push('使用CDN分发静态资源，减少数据传输距离和能耗');
1007         suggestions.push('考虑使用绿色主机服务，选择使用可再生能源的数据中心');
1008     }
1009
1010     // 确保建议数量不会太多
1011     return suggestions.slice(0, 10);
1012 }
1013
1014 /**
1015  * 检查系统环境
1016  */
1017 function checkEnvironment() {
1018     console.log('== 系统环境信息 ==');
1019     console.log(`操作系统: ${os.platform()} ${os.release()}`);
1020     console.log(`Node.js 版本: ${process.version}`);
1021     console.log(`CPU 架构: ${os.arch()}`);
1022     console.log(`可用内存: ${Math.round(os.freemem() / 1024 / 1024)} MB /
1023 ${Math.round(os.totalmem() / 1024 / 1024)} MB`);
1024     console.log('== 环境信息结束 ==');
1025 }
1026
1027 /**
1028  * 基础版网页分析 - 不依赖浏览器，使用纯HTTP请求
1029  * @param {string} url - 目标URL
1030  * @returns {Promise<Object>} 基础性能指标
1031  */
1032 async function measurePerformanceBasic(url) {
1033     try {
1034         const startTime = Date.now();
1035
1036         // 使用HEAD请求测量TTFB和响应时间
1037         const headStartTime = Date.now();
1038         const headResponse = await axiosInstance.head(url);
1039         const headEndTime = Date.now();
1040         const ttfb = headEndTime - headStartTime;
```

```
1037
1038 // 获取HTTP头信息
1039 const headers = headResponse.headers;
1040 const contentLength = headers['content-length'];
1041 const contentType = headers['content-type'] || '';
1042 const serverType = headers['server'] || 'unknown';
1043
1044 // 使用GET请求获取完整内容
1045 const getStartTime = Date.now();
1046 const response = await axiosInstance.get(url);
1047 const getEndTime = Date.now();
1048 const totalTime = getEndTime - getStartTime;
1049
1050 // 计算实际页面大小
1051 let pageSize = 0;
1052 if (contentLength) {
1053   pageSize = parseInt(contentLength);
1054 } else {
1055   // 如果没有content-length头, 使用响应数据长度
1056   pageSize = Buffer.byteLength(response.data);
1057 }
1058
1059 // 更精确地估算FCP和LCP
1060 // FCP通常是TTFB + DOM解析时间 + 关键资源加载时间
1061 const fcpEstimate = ttfb + Math.min(500, totalTime * 0.3);
1062
1063 // LCP基于总加载时间, 但通常早于完全加载完成
1064 const lcpEstimate = Math.min(totalTime, ttfb + totalTime * 0.7);
1065
1066 // CLS估计 - 由于无法精确测量布局稳定性, 使用基于HTML大小的启发式方法
1067 let clsEstimate = 0.02; // 默认值
1068 if (pageSize > 500000) clsEstimate = 0.25; // 大型页面可能有更多的布局偏移
1069 else if (pageSize > 200000) clsEstimate = 0.15;
1070 else if (pageSize > 100000) clsEstimate = 0.08;
1071
1072 // FID估计 - 由于无法精确测量交互延迟, 使用基于响应时间的启发式方法
1073 let fidEstimate = 50; // 默认值 (毫秒)
1074 if (totalTime > 3000) fidEstimate = 200;
1075 else if (totalTime > 1500) fidEstimate = 100;
1076
1077 // 分析HTML提取资源URL
1078 const resourceAnalysis = extractResourceUrls(response.data, url);
1079 const { $, resourceUrls, uniqueDomains } = resourceAnalysis;
1080
1081 // 统计不同类型的资源
1082 const resourceStats = countResourceTypes(resourceUrls);
1083
1084 // 估计总资源大小
1085 const totalResourceSize = estimateResourceSize(resourceStats, pageSize);
1086
1087 // 检查页面是否使用CDN
1088 const cdnInfo = checkCdnUsage(uniqueDomains, headers);
1089
1090 // 计算安全分数
1091 const securityInfo = calculateSecurityScore(headers);
```

```
1092
1093 // 检查页面是否支持HTTPS
1094 const supportsHTTPS = url.startsWith('https://');
1095
1096 // 检查页面是否使用压缩
1097 const supportsCompression = headers['content-encoding'] &&
1098     (headers['content-encoding'].includes('gzip')
1099     ||
1100     headers['content-encoding'].includes('br')
1101     ||
1102     headers['content-encoding'].includes('deflate'));
1103
1104 // 检查是否使用缓存控制
1105 const cacheControl = headers['cache-control'] || '';
1106 const supportsCaching = cacheControl !== '' &&
1107     (cacheControl.includes('max-age') ||
1108     cacheControl.includes('s-maxage') ||
1109     cacheControl.includes('public'));
1110
1111 // 提取HTML内容质量信息
1112 const contentQuality = analyzeContentQuality($);
1113
1114 // 构建测量结果对象
1115 const performance = {
1116     fcp: fcpEstimate,
1117     lcp: lcpEstimate,
1118     cls: clsEstimate,
1119     fid: fidEstimate,
1120     ttfb: ttfb,
1121     pageSize: pageSize,
1122     totalResourceSize: totalResourceSize,
1123     requestCount: resourceUrls.length + 1, // 加1是因为主HTML请求
1124     domainCount: uniqueDomains.size,
1125     responseTime: totalTime,
1126     resourceStats: resourceStats,
1127     usesHttps: supportsHTTPS,
1128     serverType: serverType,
1129     supportsCompression: supportsCompression,
1130     supportsCaching: supportsCaching,
1131     usesCdn: cdnInfo.usesCdn,
1132     cdnProvider: cdnInfo.cdnProvider,
1133     contentQuality: contentQuality
1134 };
1135
1136 const securityHeaders = {
1137     score: securityInfo.score,
1138     details: securityInfo.details
1139 };
1140
1141 return {
1142     success: true,
1143     performance,
1144     headers: {
1145         supportsCompression,
```

```
1144         supportsCaching,
1145         securityHeaders,
1146         serverType,
1147         supportsHTTPS
1148     },
1149     measurementMethod: 'basic-http'
1150 };
1151 } catch (error) {
1152     console.error('基础性能测量错误:', error);
1153     return {
1154         success: false,
1155         error: `基础性能测量失败: ${error.message}`
1156     };
1157 }
1158 }
1159
1160 /**
1161  * 分析HTML内容质量
1162  * @param {CheerioStatic} $ - Cheerio对象
1163  * @returns {Object} 内容质量分析结果
1164  */
1165 function analyzeContentQuality($) {
1166     if (!$) return { score: 0 };
1167
1168     try {
1169         // 计算文本内容量
1170         const bodyText = $('body').text().trim();
1171         const textLength = bodyText.length;
1172
1173         // 计算图像数量和是否有alt属性
1174         const images = $('img');
1175         const imageCount = images.length;
1176         let imagesWithAlt = 0;
1177
1178         images.each((i, img) => {
1179             if ($(img).attr('alt')) imagesWithAlt++;
1180         });
1181
1182         // 检查标题层次结构
1183         const h1Count = $('h1').length;
1184         const h2Count = $('h2').length;
1185         const h3Count = $('h3').length;
1186
1187         // 检查链接质量
1188         const links = $('a');
1189         const linkCount = links.length;
1190         let linksWithText = 0;
1191
1192         links.each((i, link) => {
1193             if ($(link).text().trim().length > 0) linksWithText++;
1194         });
1195
1196         // 检查元数据
1197         const hasTitle = $('title').length > 0;
1198         const hasDescription = $('meta[name="description"]').length > 0;
```

```
1199     const hasKeywords = $('meta[name="keywords"]').length > 0;
1200
1201     // 计算内容质量分数 (0-100)
1202     let score = 50; // 基础分数
1203
1204     // 文本内容加分
1205     if (textLength > 2000) score += 15;
1206     else if (textLength > 1000) score += 10;
1207     else if (textLength > 500) score += 5;
1208
1209     // 图像质量加分
1210     if (imageCount > 0 && imagesWithAlt / imageCount > 0.8) score += 10;
1211     else if (imageCount > 0 && imagesWithAlt / imageCount > 0.5) score += 5;
1212
1213     // 标题结构加分
1214     if (h1Count === 1) score += 5; // 最佳实践是只有一个h1
1215     if (h2Count > 0) score += 5;
1216     if (h3Count > 0) score += 3;
1217
1218     // 链接质量加分
1219     if (linkCount > 0 && linksWithText / linkCount > 0.9) score += 7;
1220
1221     // 元数据加分
1222     if (hasTitle) score += 5;
1223     if (hasDescription) score += 5;
1224     if (hasKeywords) score += 3;
1225
1226     // 确保分数范围在0-100之间
1227     score = Math.min(100, Math.max(0, score));
1228
1229     return {
1230         score,
1231         textLength,
1232         imageCount,
1233         imagesWithAlt,
1234         headingStructure: {
1235             h1Count,
1236             h2Count,
1237             h3Count
1238         },
1239         linkCount,
1240         linksWithText,
1241         metadata: {
1242             hasTitle,
1243             hasDescription,
1244             hasKeywords
1245         }
1246     };
1247 } catch (e) {
1248     console.error('内容质量分析错误:', e);
1249     return { score: 0 };
1250 }
1251 }
1252
1253 /**
```

```
1254 * 估计资源大小
1255 * @param {Object} resourceStats - 资源统计
1256 * @param {number} pageSize - HTML页面大小
1257 * @returns {number} 估计的总资源大小（字节）
1258 */
1259 function estimateResourceSize(resourceStats, pageSize) {
1260     // 不同资源类型的平均大小估计（字节）
1261     const averageSizes = {
1262         css: 20000, // 平均CSS文件约20KB
1263         js: 80000, // 平均JS文件约80KB
1264         images: 200000, // 平均图片约200KB
1265         fonts: 30000, // 平均字体约30KB
1266         other: 10000 // 其他资源约10KB
1267     };
1268
1269     let totalSize = pageSize || 0; // HTML大小
1270
1271     // 计算各类资源的估计总大小
1272     for (const [type, count] of Object.entries(resourceStats)) {
1273         if (averageSizes[type]) {
1274             totalSize += count * averageSizes[type];
1275         }
1276     }
1277
1278     return totalSize;
1279 }
1280
1281 /**
1282 * HTTP头部分析 - 检查与性能相关的HTTP头
1283 * @param {string} url - 目标URL
1284 * @returns {Promise<Object>} 基于头部的指标
1285 */
1286 async function analyzeHttpHeaders(url) {
1287     console.log('分析HTTP头部信息 ... ');
1288     try {
1289         const response = await axiosInstance.head(url, {
1290             maxRedirects: 5,
1291             validateStatus: null
1292         });
1293
1294         const headers = response.headers;
1295
1296         // 检查性能相关的HTTP头
1297         const hasCompression = headers['content-encoding'] &&
1298             (headers['content-encoding'].includes('gzip') ||
1299             headers['content-encoding'].includes('br') ||
1300             headers['content-encoding'].includes('deflate'));
1301
1302         const hasCaching = headers['cache-control'] || headers['expires'];
1303
1304         const securityHeaders = [
1305             'strict-transport-security',
1306             'content-security-policy',
1307             'x-content-type-options',
1308             'x-frame-options',
```

```
1309     'x-xss-protection'
1310 ];
1311
1312 // 检测存在的安全头
1313 const presentSecurityHeaders = [];
1314 securityHeaders.forEach(header => {
1315     if (headers[header]) {
1316         presentSecurityHeaders.push(header);
1317     }
1318 });
1319
1320 const securityScore = (presentSecurityHeaders.length /
securityHeaders.length) * 100;
1321
1322 // 分析CDN使用情况
1323 let usingCDN = false;
1324 let cdnProvider = null;
1325
1326 // 常见CDN标识头
1327 if (headers['server'] &&
1328     (headers['server'].includes('cloudflare') ||
1329     headers['server'].includes('akamai') ||
1330     headers['server'].includes('fastly'))) {
1331     usingCDN = true;
1332     cdnProvider = headers['server'].split(' ')[0];
1333 }
1334
1335 // 检查CDN特有头
1336 if (headers['cf-ray'] || headers['cf-cache-status']) {
1337     usingCDN = true;
1338     cdnProvider = 'Cloudflare';
1339 } else if (headers['x-cache'] && headers['x-served-by']) {
1340     usingCDN = true;
1341     cdnProvider = 'Fastly/Varnish';
1342 } else if (headers['x-amz-cf-id']) {
1343     usingCDN = true;
1344     cdnProvider = 'Amazon CloudFront';
1345 } else if (headers['x-azure-ref']) {
1346     usingCDN = true;
1347     cdnProvider = 'Azure CDN';
1348 }
1349
1350 return {
1351     hasCompression,
1352     hasCaching,
1353     usingCDN,
1354     cdnProvider,
1355     securityScore,
1356     securityHeaders: presentSecurityHeaders,
1357     server: headers['server'] || 'unknown',
1358     headers: headers
1359 };
1360 } catch (error) {
1361     console.error('HTTP头分析失败:', error);
1362     throw error;
```



```
1363     }
1364 }
1365
1366 // 所有其他路由返回前端应用
1367 const serveIndex = (req, res) => {
1368   res.sendFile(path.join(__dirname, '../dist/index.html'));
1369 };
1370
1371 app.get('*', serveIndex);
1372
1373 // 启动服务器
1374 app.listen(PORT, async () => {
1375   console.log(`GreenWeb服务器运行在端口 ${PORT}`);
1376   console.log(`访问 http://localhost:${PORT} 以使用应用`);
1377 });
1378
1379 /**
1380  * 从HTML中提取资源URL
1381  * @param {string} html - HTML内容
1382  * @param {string} baseUrl - 基础URL
1383  * @returns {Object} 提取的资源URL和分析结果
1384  */
1385 function extractResourceUrls(html, baseUrl) {
1386   try {
1387     const cheerio = require('cheerio');
1388     const $ = cheerio.load(html);
1389     const resourceUrls = [];
1390     const uniqueDomains = new Set();
1391
1392     // 获取URL的域名部分
1393     const getUrlDomain = (url) => {
1394       try {
1395         if (!url || typeof url !== 'string') return '';
1396         const urlObj = new URL(url, baseUrl);
1397         return urlObj.hostname;
1398       } catch (e) {
1399         return '';
1400       }
1401     };
1402
1403     // 提取域名
1404     const baseDomain = getUrlDomain(baseUrl);
1405
1406     // 处理URL
1407     const processUrl = (url) => {
1408       if (!url) return null;
1409       try {
1410         // 处理相对URL
1411         const absoluteUrl = new URL(url, baseUrl).href;
1412         const domain = getUrlDomain(absoluteUrl);
1413         if (domain) {
1414           uniqueDomains.add(domain);
1415         }
1416         return absoluteUrl;
1417       } catch (e) {
```



```
1418         return null;
1419     }
1420 };
1421
1422 // 提取CSS链接
1423 $('link[rel="stylesheet"]').each((_, el) => {
1424     const url = processUrl($(el).attr('href'));
1425     if (url) resourceUrls.push({ url, type: 'css' });
1426 });
1427
1428 // 提取JavaScript
1429 $('script[src]').each((_, el) => {
1430     const url = processUrl($(el).attr('src'));
1431     if (url) resourceUrls.push({ url, type: 'js' });
1432 });
1433
1434 // 提取图片
1435 $('img[src]').each((_, el) => {
1436     const url = processUrl($(el).attr('src'));
1437     if (url) resourceUrls.push({ url, type: 'images' });
1438 });
1439
1440 // 提取背景图像从内联样式
1441 $('[style*="background"]').each((_, el) => {
1442     const style = $(el).attr('style');
1443     if (style) {
1444         const match = style.match(/url\(['"]?([^"']+)['"]?\)/);
1445         if (match && match[1]) {
1446             const url = processUrl(match[1]);
1447             if (url) resourceUrls.push({ url, type: 'images' });
1448         }
1449     }
1450 });
1451
1452 // 提取字体
1453 $('link[rel="preload"][as="font"]').each((_, el) => {
1454     const url = processUrl($(el).attr('href'));
1455     if (url) resourceUrls.push({ url, type: 'fonts' });
1456 });
1457
1458 return {
1459     $,
1460     resourceUrls,
1461     uniqueDomains
1462 };
1463 } catch (error) {
1464     console.error('提取资源URL错误:', error);
1465     return {
1466         $: null,
1467         resourceUrls: [],
1468         uniqueDomains: new Set()
1469     };
1470 }
1471 }
1472
```

```
1473 /**
1474  * 统计不同类型的资源
1475  * @param {Array} resourceUrls - 资源URL数组
1476  * @returns {Object} 资源类型统计
1477  */
1478 function countResourceTypes(resourceUrls) {
1479     const stats = {
1480         css: 0,
1481         js: 0,
1482         images: 0,
1483         fonts: 0,
1484         other: 0
1485     };
1486
1487     resourceUrls.forEach(resource => {
1488         if (stats[resource.type] === undefined) {
1489             stats[resource.type]++;
1490         } else {
1491             stats.other++;
1492         }
1493     });
1494
1495     return stats;
1496 }
1497
1498 /**
1499  * 检查CDN使用情况
1500  * @param {Set} uniqueDomains - 唯一域名集合
1501  * @param {Object} headers - HTTP响应头
1502  * @returns {Object} CDN使用信息
1503  */
1504 function checkCdnUsage(uniqueDomains, headers) {
1505     const cdnProviders = {
1506         'cloudflare': ['cloudflare', 'cloudflare-nginx', 'cloudfront.net'],
1507         'akamai': ['akamai', 'akamaiedge.net', 'akamaized.net'],
1508         'fastly': ['fastly'],
1509         'cloudfront': ['cloudfront.net'],
1510         'vercel': ['vercel-edge', 'vercel.app'],
1511         'netlify': ['netlify', 'netlify.app']
1512     };
1513
1514     // 检查响应头中是否包含CDN信息
1515     let cdnProvider = 'unknown';
1516     let usesCdn = false;
1517
1518     // 检查常见CDN响应头
1519     if (headers['server']) {
1520         for (const [provider, keywords] of Object.entries(cdnProviders)) {
1521             if (keywords.some(keyword =>
1522                 headers['server'].toLowerCase().includes(keyword))) {
1523                 cdnProvider = provider;
1524                 usesCdn = true;
1525                 break;
1526             }
1527         }
1528     }
```

```
1527     }
1528
1529     // 检查CDN特定头部
1530     if (!usesCdn && headers['cf-ray']) {
1531         cdnProvider = 'cloudflare';
1532         usesCdn = true;
1533     } else if (!usesCdn && headers['x-fastly-request-id']) {
1534         cdnProvider = 'fastly';
1535         usesCdn = true;
1536     } else if (!usesCdn && headers['x-amz-cf-id']) {
1537         cdnProvider = 'cloudfront';
1538         usesCdn = true;
1539     } else if (!usesCdn && headers['x-vercel-cache']) {
1540         cdnProvider = 'vercel';
1541         usesCdn = true;
1542     } else if (!usesCdn && headers['x-nf-request-id']) {
1543         cdnProvider = 'netlify';
1544         usesCdn = true;
1545     }
1546
1547     // 检查域名中是否包含CDN信息
1548     if (!usesCdn) {
1549         const domains = Array.from(uniqueDomains);
1550         for (const domain of domains) {
1551             for (const [provider, keywords] of Object.entries(cdnProviders)) {
1552                 if (keywords.some(keyword => domain.includes(keyword))) {
1553                     cdnProvider = provider;
1554                     usesCdn = true;
1555                     break;
1556                 }
1557             }
1558             if (usesCdn) break;
1559         }
1560     }
1561
1562     return {
1563         usesCdn,
1564         cdnProvider
1565     };
1566 }
1567
1568 /**
1569  * 计算安全分数
1570  * @param {Object} headers - HTTP响应头
1571  * @returns {Object} 安全分数和详情
1572  */
1573 function calculateSecurityScore(headers) {
1574     let score = 0;
1575     const details = {};
1576
1577     // 检查常见安全响应头
1578     const securityHeaders = {
1579         'strict-transport-security': { score: 20, name: '严格传输安全' },
1580         'content-security-policy': { score: 20, name: '内容安全策略' },
1581         'x-content-type-options': { score: 10, name: '内容类型选项' },
```

```
1582     'x-frame-options': { score: 10, name: '框架选项' },
1583     'x-xss-protection': { score: 10, name: 'XSS保护' },
1584     'referrer-policy': { score: 10, name: '引用策略' },
1585     'permissions-policy': { score: 10, name: '权限策略' },
1586     'feature-policy': { score: 10, name: '功能策略' }
1587 };
1588
1589 for (const [header, info] of Object.entries(securityHeaders)) {
1590     if (headers[header]) {
1591         score += info.score;
1592         details[header] = {
1593             present: true,
1594             value: headers[header],
1595             name: info.name
1596         };
1597     } else {
1598         details[header] = {
1599             present: false,
1600             name: info.name
1601         };
1602     }
1603 }
1604
1605 return {
1606     score: Math.min(93, score),
1607     details
1608 };
1609 }
1610
1611 // 添加安全的toFixed函数，避免对null/undefined调用toFixed
1612 function safeToFixed(value, digits = 2) {
1613     if (value === null || value === undefined) return '0.00';
1614     return Number(value).toFixed(digits);
1615 }
```

```
1  /**
2   * 网站分析服务
3   * 用于向目标网站发送请求并获取性能指标和位置信息
4   */
5
6  import axios from 'axios';
7
8  // 获取当前环境的基础URL
9  function getBaseUrl() {
10   // 在生产环境中，API和前端在同一域下运行
11   // 在开发环境中，API在localhost:3000上运行
12   const isProd = import.meta.env.PROD;
13   return isProd ? '/api' : 'http://localhost:3000/api';
14 }
15
16 // 代理服务器URL
17 const API_URL = getBaseUrl();
18
19 /**
20 * 解析URL获取完整域名
21 * @param {string} url - 输入的URL或域名
22 * @returns {string} 处理后的完整URL
23 */
24 function parseUrl(url) {
25   if (!url) return '';
26
27   // 如果没有协议，添加https://
28   if (!url.startsWith('http://') && !url.startsWith('https://')) {
29     url = 'https://' + url;
30   }
31
32   try {
33     const parsedUrl = new URL(url);
34     return parsedUrl.href;
35   } catch (error) {
36     console.error('URL解析错误:', error);
37     return '';
38   }
39 }
40
41 /**
42 * 获取网站服务器位置
43 * @param {string} domain - 目标域名
44 * @returns {Promise<Object>} 服务器位置信息
45 */
46 export async function getServerLocation(domain) {
47   try {
48     const response = await axios.get(`${API_URL}/location`, {
49       params: { domain }
50     });
51     return response.data;
52   } catch (error) {
53     console.error('获取服务器位置失败:', error);
```

```

54     // 失败时返回错误信息
55     return {
56         error: '无法获取服务器位置信息',
57         details: error.message,
58         measurable: false
59     };
60 }
61 }
62
63 /**
64  * 获取网站HTTP响应头信息
65  * @param {string} url - 目标URL
66  * @returns {Promise<Object>} HTTP响应头信息
67  */
68 export async function getHttpHeaders(url) {
69     try {
70         const fullUrl = parseUrl(url);
71         const response = await axios.get(`${API_URL}/headers`, {
72             params: { url: fullUrl }
73         });
74         return response.data;
75     } catch (error) {
76         console.error('获取HTTP头信息失败:', error);
77         return {
78             error: '无法获取HTTP头信息',
79             details: error.message,
80             measurable: false
81         };
82     }
83 }
84
85 /**
86  * 测量网站页面大小
87  * @param {string} url - 目标URL
88  * @returns {Promise<number>} 页面大小 (KB)
89  */
90 export async function measurePageSize(url) {
91     try {
92         const fullUrl = parseUrl(url);
93         const response = await axios.get(`${API_URL}/size`, {
94             params: { url: fullUrl }
95         });
96         return response.data;
97     } catch (error) {
98         console.error('测量页面大小失败:', error);
99         // 失败时返回错误信息
100         return {
101             error: '无法测量页面大小',
102             details: error.message,
103             measurable: false
104         };
105     }
106 }
107
108 /**

```

```

109  * 测量网站性能指标
110  * @param {string} url - 目标URL
111  * @param {string} [browser='auto'] - 使用的浏览器, 可选值: 'auto'(自
    动)、'chrome'、'edge'
112  * @returns {Promise<Object>} 性能指标
113  */
114  export async function measurePerformance(url, browser = 'auto') {
115      let retries = 0;
116      const maxRetries = 2;
117      const retryDelay = 3000; // 重试间隔3秒
118
119      const attemptMeasure = async () => {
120          try {
121              const fullUrl = parseUrl(url);
122              console.log(`尝试测量性能指标 (使用${browser} === 'auto' ? '自动选择浏览器' :
    browser)) (尝试 ${retries + 1}/${maxRetries + 1}): ${fullUrl}`);
123
124              const response = await axios.get(`${API_URL}/performance`, {
125                  params: {
126                      url: fullUrl,
127                      browser: browser // 将浏览器参数传递给后端
128                  },
129                  timeout: 120000 // 增加超时时间到120秒, 因为Lighthouse分析可能需要较长时间
130              });
131
132              console.log(`性能指标测量成功 (使用${response.data.measuredBy} || '未知方
    法'):`), response.data);
133              return response.data;
134          } catch (error) {
135              if (axios.isAxiosError(error)) {
136                  console.error(`性能测量失败 (尝试 ${retries + 1}/${maxRetries + 1}):`,
    error.message);
137
138                  if (error.response) {
139                      // 服务器返回了错误状态码
140                      console.error('服务器错误:', error.response.status,
    error.response.data);
141                      return {
142                          error: `性能测量失败: ${error.response.data.message} || '服务器错
    误'`,
143                          details: error.response.data,
144                          measurable: false
145                      };
146                  } else if (error.request) {
147                      // 请求已经发出, 但没有收到响应
148                      console.error('未收到响应:', error.request);
149                      if (retries < maxRetries) {
150                          retries++;
151                          console.log(`等待 ${retryDelay}ms 后重试 ...`);
152                          await new Promise(resolve => setTimeout(resolve, retryDelay));
153                          return attemptMeasure();
154                      }
155                      return {
156                          error: '性能测量超时, 请检查网络连接或网站可访问性',
157                          details: '请求已发出但未收到响应',

```

```

158         measurable: false
159     };
160 } else {
161     // 设置请求时发生了错误
162     if (retries < maxRetries) {
163         retries++;
164         console.log(`等待 ${retryDelay}ms 后重试 ...`);
165         await new Promise(resolve => setTimeout(resolve, retryDelay));
166         return attemptMeasure();
167     }
168     return {
169         error: '无法测量性能指标',
170         details: error.message,
171         measurable: false
172     };
173 }
174 } else {
175     // 非Axios错误
176     console.error('性能测量过程中发生非网络错误:', error);
177     return {
178         error: '无法测量性能指标',
179         details: error.message,
180         measurable: false
181     };
182 }
183 }
184 };
185
186 return attemptMeasure();
187 }
188
189 /**
190  * 分析网站服务器提供商
191  * @param {string} domain - 目标域名
192  * @returns {Promise<Object>} 服务提供商信息
193  */
194 export async function analyzeProvider(domain) {
195     try {
196         const response = await axios.get(`${API_URL}/provider`, {
197             params: { domain }
198         });
199         return response.data;
200     } catch (error) {
201         console.error('分析服务提供商失败:', error);
202         return {
203             error: '无法分析服务提供商',
204             details: error.message,
205             measurable: false
206         };
207     }
208 }
209
210 /**
211  * 综合分析网站（使用后端的单一接口）
212  * @param {string} domain - 目标域名

```



```
213 * @param {string} [browser='auto'] - 使用的浏览器，可选值：'auto'(自
动)、'chrome'、'edge'
214 * @returns {Promise<Object>} 分析结果
215 */
216 export async function analyzeWebsite(domain, browser = 'auto') {
217   try {
218     // 使用后端的综合分析端点
219     const response = await axios.get(`${API_URL}/analyze`, {
220       params: {
221         domain,
222         browser // 添加浏览器参数
223       },
224       timeout: 120000 // 增加超时时间到120秒
225     });
226
227     return response.data;
228   } catch (error) {
229     console.error('网站分析失败:', error);
230     return {
231       error: '网站分析失败',
232       details: error.message,
233       measurable: false
234     };
235   }
236 }
237
238 /**
239 * 单独调用，不使用综合接口（用于测试和调试）
240 * @param {string} domain - 目标域名
241 * @param {string} [browser='auto'] - 使用的浏览器，可选值：'auto'(自
动)、'chrome'、'edge'
242 * @returns {Promise<Object>} 分析结果
243 */
244 export async function analyzeWebsiteSeparately(domain, browser = 'auto') {
245   try {
246     // 准备完整URL
247     const url = parseUrl(domain);
248     if (!url) throw new Error('无效的域名');
249
250     // 并行执行所有请求以提高性能
251     const [locationData, headersData, sizeData, performanceData,
providerData] = await Promise.all([
252       getServerLocation(domain),
253       getHttpHeaders(url),
254       measurePageSize(url),
255       measurePerformance(url, browser), // 传递浏览器参数
256       analyzeProvider(domain)
257     ]);
258
259     // 检查是否有任何请求失败
260     if (locationData.error || headersData.error || sizeData.error ||
performanceData.error || providerData.error) {
261       const errors = [];
262       if (locationData.error) errors.push(locationData.error);
263       if (headersData.error) errors.push(headersData.error);
264
```

```

265     if (sizeData.error) errors.push(sizeData.error);
266     if (performanceData.error) errors.push(performanceData.error);
267     if (providerData.error) errors.push(providerData.error);
268
269     return {
270         url,
271         domain,
272         browser, // 添加使用的浏览器信息
273         error: '部分数据获取失败',
274         details: errors.join('; '),
275         measurable: false,
276         location: locationData,
277         provider: providerData.provider || 'unknown',
278         pageSize: sizeData.size,
279         performance: performanceData,
280         headers: headersData.headers
281     };
282 }
283
284 return {
285     url,
286     domain,
287     browser, // 添加使用的浏览器信息
288     provider: providerData.provider,
289     location: locationData,
290     pageSize: sizeData.size,
291     performance: performanceData,
292     headers: headersData.headers
293 };
294 } catch (error) {
295     console.error('网站分析失败:', error);
296     return {
297         error: '网站分析失败',
298         details: error.message,
299         measurable: false
300     };
301 }
302 }
303
304 /**
305  * 分析网站碳排放
306  * @param {Object} params - 碳排放分析参数
307  * @param {number} params.pageSize - 页面大小(KB)
308  * @param {string} params.country - 服务器所在国家代码
309  * @param {number} params.requestCount - 请求数量
310  * @param {number} params.domainCount - 域名数量
311  * @param {number} [params.renewablePercentage] - 可再生能源使用百分比
312  * @param {number} [params.pue] - 数据中心PUE
313  * @returns {Promise<Object>} 碳排放分析结果
314  */
315 export async function analyzeCarbonEmission(params) {
316     try {
317         const response = await axios.get(`${API_URL}/carbon`, { params });
318         return response.data;
319     } catch (error) {

```

```
320     console.error('碳排放分析失败:', error);
321     return {
322       error: '无法分析碳排放',
323       details: error.message,
324       measurable: false,
325       totalCarbonEmission: 0,
326       monthlyCarbonEmission: 0,
327       annualCarbonEmission: 0,
328       isGreen: false
329     };
330   }
331 }
332
333 export default {
334   analyzeWebsite,
335   analyzeWebsiteSeparately,
336   getServerLocation,
337   measurePageSize,
338   measurePerformance,
339   analyzeProvider,
340   getHttpHeaders,
341   analyzeCarbonEmission
342 };
```