```
@echo off
chcp 65001 >nul
setlocal enabledelayedexpansion

:: 设置标题
title GreenWeb 启动工具

echo  ═══════════════════════════════════════════
echo           GreenWeb 启动工具
echo  ═══════════════════════════════════════════
echo.
";
:: 检查Node.js和npm是否安装
echo 检查环境依赖...
where node >nul 2>nul
if %errorlevel% neq 0 (
    echo Node.js未安装，准备自动安装...
    call :InstallNodeJS
) else (
    for /f "tokens=*" %%i in ('node -v') do set nodeVersion=%%i
    echo 已安装Node.js版本: !nodeVersion!
)

where npm >nul 2>nul
if %errorlevel% neq 0 (
    echo npm未安装，可能需要重新安装Node.js...
    call :InstallNodeJS
) else (
    for /f "tokens=*" %%i in ('npm -v') do set npmVersion=%%i
    echo 已安装npm版本: !npmVersion!
)

:: 检查项目依赖是否安装
if not exist "node_modules" (
    echo 项目依赖未安装，准备安装...

    :: 检查package.json文件是否存在
    if not exist "package.json" (
        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。
        goto End
    )

    echo 正在安装项目依赖，这可能需要几分钟...
    call npm install
    call cd server
    call npm install
    call cd ..
    if !errorlevel! neq 0 (
        echo 依赖安装失败，请检查网络连接或手动运行 npm install
        goto End
    )
    echo 依赖安装完成!
) else if not exist "server/node_modules" (
```

```
    echo 正在安装项目依赖，这可能需要几分钟 ...
    call cd server
    call npm install
    call cd ..
    echo 依赖安装完成!
)else (
    echo 项目依赖已安装
)

:Menu
echo.
echo  ═══════════════════════════════════════════════
echo                    请选择启动模式
echo  ───────────────────────────────────────────────
echo   1. 开发模式（前后端分离）
echo   2. 生产模式（整合前后端）
echo   3. 仅构建项目
echo   4. 更新项目依赖
echo   5. 退出
echo  ═══════════════════════════════════════════════
echo.

set /p choice=请选择操作 (1-5):

if "%choice%"=="1" (
    call :StartDevMode
) else if "%choice%"=="2" (
    call :StartProdMode
) else if "%choice%"=="3" (
    call :BuildProject
) else if "%choice%"=="4" (
    call :UpdateDependencies
) else if "%choice%"=="5" (
    echo 退出程序 ...
    exit /b
) else (
    echo 无效的选择，请重新输入!
    goto Menu
)

goto End

:InstallNodeJS
echo.
echo 准备安装Node.js ...
echo 正在下载Node.js安装程序 ...

:: 创建临时目录
set tempDir=%temp%\NodeJSInstall
mkdir %tempDir% 2>nul

:: 检查是否已安装curl
where curl >nul 2>nul
if %errorlevel% neq 0 (
    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 msiexec /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"
```

```
echo 2. 启动前端开发服务器 ...
start cmd /k "cd /d %~dp0 && npm run dev"
echo 服务已启动! 请在浏览器中打开 http://localhost:5173
goto End

:StartProdMode
echo 启动生产模式 ...

:: 检查并关闭占用端口的进程
call :CheckAndClosePort 3000

:: 确保package.json中有相应的脚本命令
findstr /c:"\"build\"" package.json >nul
if %errorlevel% neq 0 (
    echo 警告：package.json 中未找到 "build" 脚本。
    echo 请确保项目配置正确。
    goto End
)

findstr /c:"\"start\"" package.json >nul
if %errorlevel% neq 0 (
    echo 警告：package.json 中未找到 "start" 脚本。
    echo 请确保项目配置正确。
    goto End
)

echo 1. 构建前端应用 ...
call npm run build
if %errorlevel% neq 0 (
    echo 构建失败，请检查错误信息
    goto End
)

echo 2. 启动整合服务器 ...
start cmd /k "cd /d %~dp0 && npm run start"

echo 服务已启动! 请在浏览器中打开 http://localhost:3000
goto End

:BuildProject
echo 执行项目构建 ...

:: 确保package.json中有相应的脚本命令
findstr /c:"\"build\"" package.json >nul
if %errorlevel% neq 0 (
    echo 警告：package.json 中未找到 "build" 脚本。
    echo 请确保项目配置正确。
    goto End
)

call npm run build
if %errorlevel% neq 0 (
    echo 构建失败，请检查错误信息
) else (
    echo 构建完成! 可以通过 npm run start 启动服务
```

```
)
goto End

:UpdateDependencies
echo 更新项目依赖...
call npm update
if %errorlevel% neq 0 (
    echo 依赖更新失败，请检查网络连接或手动运行 npm update
) else (
    echo 依赖更新完成!
)
goto End

:CheckAndClosePort
echo 检查端口 %1 占用情况...
FOR /F "tokens=5" %%P IN ('netstat -ano ^| findstr :%1 ^| findstr LISTENING')
DO (
    echo 发现进程占用端口 %1: %%P，尝试关闭...
    taskkill /F /PID %%P 2>NUL
    if !errorlevel! equ 0 (
        echo 端口 %1 已释放
    ) else (
        echo 无法释放端口 %1，可能需要手动关闭进程
    )
)
goto :eof

:End
echo.
echo 按任意键继续...
pause >nul
```

```vue
<template>
  <div class="green-web-app">
    <!-- 标题和介绍 -->
    <header class="app-header">
      <div class="logo">
        <img src="./assets/logo.svg" alt="Green Web Checker Logo">
      </div>
      <h1>GreenWeb <span>碳排放检测</span></h1>
      <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>
    </header>

    <!-- 输入区域 -->
    <div class="input-section">
      <div class="url-input-group">
        <el-input
          v-model="domain"
          class="domain-input"
          placeholder="输入网站域名 (例如: example.com)"
          @keyup.enter="checkCarbon"
          :disabled="loading"
        />

        <!-- 浏览器选择下拉菜单 -->
        <el-select v-model="selectedBrowser" placeholder="分析方式" class="browser-select">
          <el-option label="自动选择" value="auto"></el-option>
          <el-option label="基础HTTP分析" value="basic"></el-option>
          <el-option label="仅HTTP头分析" value="headers"></el-option>
        </el-select>

        <el-button
          type="primary"
          @click="checkCarbon"
          :loading="loading"
          class="analyze-button"
        >
          分析碳排放
        </el-button>
      </div>
    </div>

    <main class="main-content">
      <div v-if="loading" class="loading-container">
        <div class="earth-container">
          <div class="earth"></div>
        </div>
        <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>
      </div>

      <div v-if="result && !loading" class="result-section">
        <!-- 添加错误提示区域 -->
        <div v-if="result.hasError" class="error-alerts">
```

```
          <div class="error-alert">
            <h3>注意：部分数据无法获取</h3>
            <ul>
              <li v-for="(error, index) in result.errorMessages"
:key="index">{{ error }}</li>
            </ul>
            <p>以下分析仅基于可获取的真实数据。</p>
          </div>
        </div>
        <div class="result-summary">
          <div class="summary-card" :class="result.isGreen ? 'green' :
'red'">
            <div class="summary-header">
              <div class="summary-title">碳排放评估结果</div>
              <div class="summary-subtitle">{{ domain }}</div>
            </div>
            <div class="summary-content">
              <div class="summary-section">
                <div class="total-emission">
                  <div class="carbon-icon">
                    <i :class="result.isGreen ? 'fa-solid fa-leaf fa-2x' :
'fa-solid fa-smog fa-2x'"></i>
                  </div>
                  <div class="carbon-details">
                    <div class="total-title">此网页单次访问的碳排放</div>
                    <div class="total-value">
                      {{ safeToFixed(result.totalCarbonEmission, 4) ||
'0.0000' }} gCO2e
                      <span v-if="result.estimatedData &&
result.estimatedData.includes('totalCarbonEmission')" class="estimated-data-
tag">估算</span>
                    </div>
                    <div class="total-breakdown">
                      <div class="total-item">
                        <span class="total-label">月度碳排放:</span>
                        <span class="total-value">
                          {{ safeToFixed(result.monthlyCarbonEmission, 2) ||
'0.00' }} kgCO2e
                          <span v-if="result.estimatedData &&
result.estimatedData.includes('monthlyCarbonEmission')" class="estimated-
data-tag">估算</span>
                        </span>
                      </div>
                      <div class="total-info">
                        基于{{
globalConstants.averageMonthlyVisits.toLocaleString() }}次月访问量
                        <span class="estimated-data-tag">估算</span>
                      </div>
                      <div class="total-item">
                        <span class="total-label">年度碳排放:</span>
                        <span class="total-value">
                          {{ safeToFixed(result.annualCarbonEmission, 2) ||
'0.00' }} kgCO2e
                          <span v-if="result.estimatedData &&
result.estimatedData.includes('annualCarbonEmission')" class="estimated-
```

```
data-tag">估算</span>
                        </span>
                      </div>
                      <div class="total-info">
                        相当于种植{{ Math.round((result.annualCarbonEmission || 0) / (globalConstants.treeCO2PerYear || 1)) }}棵树才能抵消
                        <span class="estimated-data-tag">估算</span>
                      </div>
                    </div>
                  </div>
                </div>

                <div class="summary-section">
                  <div class="energy-breakdown">
                    <div class="energy-title">能源消耗明细</div>
                    <div class="energy-items">
                      <div class="energy-item">
                        <div class="energy-label">数据中心</div>
                        <div class="energy-value">
                          {{ safeToFixed(result.dataCenterEnergy || 0, 4) }} Wh
                          <span v-if="result.estimatedData && result.estimatedData.includes('dataCenterEnergy')" class="estimated-data-tag">估算</span>
                        </div>
                      </div>
                      <div class="energy-item">
                        <div class="energy-label">网络传输</div>
                        <div class="energy-value">
                          {{ safeToFixed(result.transmissionEnergy || 0, 4) }} Wh
                          <span v-if="result.estimatedData && result.estimatedData.includes('transmissionEnergy')" class="estimated-data-tag">估算</span>
                        </div>
                      </div>
                      <div class="energy-item">
                        <div class="energy-label">用户设备</div>
                        <div class="energy-value">
                          {{ safeToFixed(result.deviceEnergy || 0, 4) }} Wh
                          <span v-if="result.estimatedData && result.estimatedData.includes('deviceEnergy')" class="estimated-data-tag">估算</span>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>

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

```
            </div>
            <div class="detail-content">
              <div class="provider-info">
                <div class="provider-icon">
                  <i :class="result.isGreen ? 'fa-solid fa-solar-panel fa-2x' : 'fa-solid fa-industry fa-2x'"></i>
                </div>
                <div class="provider-details">
                  <div class="provider-name">{{ result.providerName || '未知服务商' }}</div>
                  <div class="provider-stats">
                    <div class="provider-item">
                      <span class="provider-label">可再生能源使用率:</span>
                      <span class="provider-value">
                        {{ result.renewablePercentage !== null ? result.renewablePercentage + '%' : '未知' }}
                        <span v-if="result.estimatedData && result.estimatedData.includes('renewablePercentage')" class="estimated-data-tag">估算</span>
                      </span>
                    </div>
                    <div class="provider-item">
                      <span class="provider-label">PUE:</span>
                      <span class="provider-value">
                        {{ result.pue || '未知' }}
                        <span v-if="result.estimatedData && result.estimatedData.includes('pue')" class="estimated-data-tag">估算</span>
                      </span>
                    </div>
                    <div class="provider-item">
                      <span class="provider-label">服务器位置:</span>
                      <span class="provider-value">{{ result.region || '未知' }}, {{ result.country || '未知' }}</span>
                    </div>
                    <div class="provider-item">
                      <span class="provider-label">绿色托管等级:</span>
                      <span class="provider-value" :class="{'green-value': result.isGreen}">
                        {{ result.isGreen ? '环保托管' : '标准托管' }}
                      </span>
                    </div>
                    <div class="provider-item">
                      <span class="provider-label">全球碳强度:</span>
                      <span class="provider-value">{{ globalConstants.averageCarbonIntensity }} gCO2e/kWh</span>
                    </div>
                  </div>
                </div>
              </div>
              <div v-if="result.isGreen" class="green-hosting-info">
                <p>该网站使用环保托管服务，有助于减少互联网碳足迹。</p>
              </div>
              <div v-else class="green-hosting-info">
                <p>建议考虑使用更环保的托管服务，以减少碳排放。</p>
              </div>
```

```
            </div>
          </div>
        </div>

        <div class="result-grid">

          <!-- 现代化的结果卡片网格布局 -->
          <div class="result-card-container">
            <!-- 第一行卡片 - 重要信息 -->
            <div class="result-row">
              <div class="result-card energy-source major-card">
                <div class="card-header">
                  <h3>可再生能源使用情况</h3>
                  <div class="card-icon">
                    <el-icon><DataBoard /></el-icon>
                  </div>
                </div>
                <div v-if="result.renewablePercentage !== null"
class="energy-chart">
                  <div class="donut-chart">
                    <div class="donut-hole">{{ result.renewablePercentage
}}%</div>
                    <div class="donut-ring">
                      <div class="renewable" :style="`--percent:
${result.renewablePercentage}%`"></div>
                    </div>
                  </div>
                  <div class="chart-legend">
                    <div class="legend-item">
                      <div class="legend-color renewable"></div>
                      <span>可再生能源 ({{ result.renewablePercentage }}%)
</span>
                    </div>
                    <div class="legend-item">
                      <div class="legend-color fossil"></div>
                      <span>化石能源 ({{ 100 - result.renewablePercentage
}}%)</span>
                    </div>
                  </div>
                </div>
                <div v-else class="data-unavailable">
                  无法获取可再生能源使用比例
                </div>
                <div class="card-summary">
                  <div class="summary-badge" :class="result.isGreen ?
'green-badge' : 'standard-badge'">
                    {{ result.isGreen ? '环保托管' : '标准托管' }}
                  </div>
                  <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>
                </div>
              </div>

              <div class="result-card carbon-map major-card">
                <div class="card-header">
```

```
                  <h3>碳排放评分</h3>
                  <div class="card-icon">
                    <el-icon><PieChart /></el-icon>
                  </div>
                </div>
                <div v-if="result.totalCarbonEmission !== null">
                  <div class="score-container" v-
if="result.carbonFootprintScore || result.energyEfficiencyScore">
                    <div class="carbon-score-card"
:class="getCarbonScoreClass(result.carbonFootprintScore)">
                      <div class="score-circle">{{
Math.round(result.carbonFootprintScore || 50) }}</div>
                      <div class="score-label">碳足迹评分</div>
                      <div class="score-desc">{{
getCarbonScoreDesc(result.carbonFootprintScore) }}</div>
                    </div>
                    <div class="carbon-score-card"
:class="getEnergyScoreClass(result.energyEfficiencyScore)">
                      <div class="score-circle">{{
Math.round(result.energyEfficiencyScore || 50) }}</div>
                      <div class="score-label">能源效率评分</div>
                      <div class="score-desc">{{
getEnergyScoreDesc(result.energyEfficiencyScore) }}</div>
                    </div>
                  </div>
                  <div class="total-emission-highlight">
                    <div class="highlight-value">{{
safeToFixed(result.totalCarbonEmission, 4) }} <span>gCO2e</span></div>
                    <div class="highlight-label">单次访问碳排放</div>
                  </div>
                </div>
                <div v-else class="data-unavailable">
                  无法获取碳排放分析数据
                </div>
              </div>
            </div>

            <!-- 第二行卡片 - 详细数据 -->
            <div class="result-row">
              <div class="result-card site-info">
                <div class="card-header">
                  <h3>网站基础信息</h3>
                  <div class="card-icon">
                    <el-icon><Connection /></el-icon>
                  </div>
                </div>
                <div class="details">
                  <div class="detail-item">
                    <span class="detail-label">服务商:</span>
                    <span class="detail-value highlight-text">{{
result.provider && typeof result.provider === 'string' ?
result.provider.toUpperCase() : '未知' }}</span>
                  </div>
                  <div class="detail-item">
                    <span class="detail-label">服务器位置:</span>
```

```
                    <span class="detail-value">{{ result.region || '未知' }}, {{ result.country || '未知' }}</span>
                  </div>
                  <div class="detail-item">
                    <span class="detail-label">页面大小:</span>
                    <span class="detail-value">{{ result.pageSize ? `${result.pageSize} KB` : '无法获取' }}</span>
                  </div>
                  <div class="detail-item">
                    <span class="detail-label">能源强度:</span>
                    <span class="detail-value">{{ result.energyIntensity !== null ? `${safeToFixed(result.energyIntensity, 2)} kWh/GB` : '无法获取' }}</span>
                  </div>
                  <div class="detail-item">
                    <span class="detail-label">PUE值:</span>
                    <span class="detail-value" :class="{'highlight-text': result.pue && result.pue < 1.5}">
                      {{ result.pue || '未知' }}
                      <span v-if="result.estimatedData && result.estimatedData.includes('pue')" class="estimated-data-tag">估算</span>
                    </span>
                  </div>
                </div>
              </div>

              <div class="result-card energy-consumption">
                <div class="card-header">
                  <h3>能源消耗分析</h3>
                  <div class="card-icon">
                    <el-icon><Connection /></el-icon>
                  </div>
                </div>
                <!-- 显示设置，确保始终显示能源消耗网格 -->
                <div class="energy-consumption-grid">
                  <div class="energy-consumption-item">
                    <div class="consumption-icon data-center-icon"></div>
                    <div class="consumption-value">{{ formatEnergyValue(result.dataCenterEnergy) }}</div>
                    <div class="consumption-label">数据中心能耗</div>
                  </div>
                  <div class="energy-consumption-item">
                    <div class="consumption-icon network-icon"></div>
                    <div class="consumption-value">{{ formatEnergyValue(result.transmissionEnergy) }}</div>
                    <div class="consumption-label">网络传输能耗</div>
                  </div>
                  <div class="energy-consumption-item">
                    <div class="consumption-icon device-icon"></div>
                    <div class="consumption-value">{{ formatEnergyValue(result.deviceEnergy) }}</div>
                    <div class="consumption-label">用户设备能耗</div>
                  </div>
                </div>
```

```
                <div v-if="!result.dataCenterEnergy &&
!result.transmissionEnergy && !result.deviceEnergy" class="energy-note">
                  <p>注意: 能源消耗数据可能很低或无法获取。这可能是因为页面太小或性能分
析不完整。</p>
                </div>
              </div>

              <div class="result-card carbon-impact">
                <div class="card-header">
                  <h3>碳排放详情</h3>
                  <div class="card-icon">
                    <el-icon><PieChart /></el-icon>
                  </div>
                </div>
                <div v-if="result.totalCarbonEmission !== null"
class="carbon-stats">
                  <div class="carbon-stat-item">
                    <span class="stat-label">数据中心碳排放:</span>
                    <span class="stat-value">{{
formatNumber(result.dataTransferCarbon) }} gCO2e</span>
                  </div>
                  <div class="carbon-stat-item">
                    <span class="stat-label">网络传输碳排放:</span>
                    <span class="stat-value">{{
formatNumber(result.networkCarbon) }} gCO2e</span>
                  </div>
                  <div class="carbon-stat-item">
                    <span class="stat-label">客户端碳排放:</span>
                    <span class="stat-value">{{
formatNumber(result.clientCarbon) }} gCO2e</span>
                  </div>
                  <div class="carbon-annual-impact">
                    <div class="impact-title">年度碳排放估计</div>
                    <div class="impact-value">{{
safeToFixed(result.annualCarbonEmission, 2) }} kgCO2e</div>
                    <div class="impact-equivalent">
                      相当于种植 <span class="highlight-text">{{
Math.round((result.annualCarbonEmission || 0) /
globalConstants.treeCO2PerYear) }}</span> 棵树才能抵消
                    </div>
                  </div>
                </div>
                <div v-else class="data-unavailable">
                  无法获取碳排放分析数据
                </div>
              </div>
            </div>

            <!-- 第三行卡片 - 性能和建议 -->
            <div class="result-row">
              <div class="result-card performance wide-card">
                <div class="card-header">
                  <h3>性能指标</h3>
                  <div class="card-icon">
                    <el-icon><Timer /></el-icon>
```

```
                  </div>
                </div>
                <div v-if="result.performance &&
result.performance.measurable !== false" class="performance-metrics-v2">
                  <div class="metrics-grid">
                    <div
                      v-for="(value, metric, index) in result.performance"
                      :key="metric"
                      class="metric-card"
                      v-show="!['measuredBy', 'statusCode', 'measurable',
'requestCount', 'domainCount'].includes(metric)"
                    >
                      <div class="metric-visual">
                        <div class="metric-circle"
:class="getMetricGrade(metric, value)">
                          {{ formatMetricValue(metric, value) }}
                        </div>
                      </div>
                      <div class="metric-info">
                        <div class="metric-name">{{ formatMetricName(metric)
}}</div>
                        <div class="metric-description">{{
getMetricDescription(metric) }}</div>
                      </div>
                    </div>

                  <div v-if="result.performance.requestCount ||
result.performance.domainCount" class="network-stats">
                    <div class="network-title">网络请求数据</div>
                    <div class="network-grid">
                      <div class="network-item" v-
if="result.performance.requestCount">
                        <div class="network-value">{{
result.performance.requestCount }}</div>
                        <div class="network-label">请求数量</div>
                      </div>
                      <div class="network-item" v-
if="result.performance.domainCount">
                        <div class="network-value">{{
result.performance.domainCount }}</div>
                        <div class="network-label">域名数量</div>
                      </div>
                    </div>
                  </div>

                  <div v-if="result.performance.measuredBy" class="data-
source-info">
                    测量工具: {{ result.performance.measuredBy }}
                  </div>
                </div>
                <div v-else class="data-unavailable">
                  无法获取性能指标数据<br>
                  <small v-if="result.performance &&
result.performance.error">{{ result.performance.error }}</small>
```

```
                </div>
              </div>

              <div class="result-card suggestions wide-card">
                <div class="card-header">
                  <h3>优化建议</h3>
                  <div class="card-icon">
                    <el-icon><Opportunity /></el-icon>
                  </div>
                </div>
                <div v-if="result.suggestions && result.suggestions.length > 0" class="suggestions-container">
                  <div class="suggestions-list">
                    <div
                      v-for="(suggestion, index) in result.suggestions"
                      :key="index"
                      class="suggestion-card"
                    >
                      <div class="suggestion-header">
                        <div class="suggestion-icon">
                          <el-icon><Opportunity /></el-icon>
                        </div>
                        <div class="suggestion-title">优化建议 #{{ index + 1 }}</div>
                      </div>
                      <div class="suggestion-content">
                        {{ suggestion }}
                      </div>
                      <div class="suggestion-impact" :class="getImpactClass(index)">
                        {{ getImpactLevel(index) }}
                      </div>
                    </div>
                  </div>
                </div>
                <div v-else class="data-unavailable">
                  无法生成具体优化建议，请确保网站可访问
                </div>
              </div>
            </div>

            <!-- 绿色托管信息卡片 -->
            <div v-if="carbonResult" class="result-card green-hosting wide-card">
              <h3>绿色托管信息</h3>
              <div class="detail-content">
                <div class="provider-info">
                  <div><strong>提供商：</strong> {{ carbonResult.provider || '未知' }}</div>
                  <div><strong>服务器位置：</strong> {{ carbonResult.region || '未知' }}</div>

                  <div class="provider-stats">
                    <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
    === 'auto' ? '自动选择浏览器' : selectedBrowser.value})`)
544
545           // 获取使用的浏览器参数
546           const browserOption = selectedBrowser.value
547
548           let performanceResult = null
549           let locationResult = null
550           let providerResult = null
551           let carbonResult = null
552           let hasError = false
553           let errorMessages = []
554
555           // 1. 测量性能指标
556           try {
557             performanceResult = await measurePerformance(domain.value,
    browserOption)
558
```

```
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 - 只在有足够的真实数据时进行
601           if (performanceResult && !performanceResult.error && locationResult
    && !locationResult.error) {
602             try {
603               const carbonParams = {
604                 pageSize: performanceResult.pageSize ||
    (performanceResult.performance?.pageSize) || 0,
605                 country: locationResult?.country || 'unknown',
606                 requestCount: performanceResult.requests ||
    (performanceResult.performance?.requestCount) || 1,
607                 domainCount: performanceResult.domainCount ||
    (performanceResult.performance?.domainCount) || 1,
608                 // 添加必要的参数
609                 renewablePercentage: providerResult?.renewablePercentage || 0,
```

```
              pue: providerResult?.pue || 1.67, // 使用平均PUE
              responseTime: performanceResult.responseTime ||
(performanceResult.performance?.responseTime) || 0,
              hasCompression: performanceResult.headers?.supportsCompression
|| false,
              resourceStats:
JSON.stringify(performanceResult.performance?.resourceStats || {}),
              cacheControl: performanceResult.headers?.cacheControl || ''
            }

            // 确保主要参数有值
            if (carbonParams.pageSize <= 0) {
              throw new Error('页面大小不能为零或负值');
            }

            carbonResult = await analyzeCarbonEmission(carbonParams)

            if (carbonResult.error) {
              console.warn('碳排放计算警告:', carbonResult.error)
              errorMessages.push(`碳排放计算: ${carbonResult.error}`)
              hasError = true
            }
          } catch (carbonError) {
            console.error('碳排放计算失败:', carbonError)
            errorMessages.push(`碳排放计算: ${carbonError.message || '无法获
取'}`)
            hasError = true
          }
        } else {
          console.warn('缺少性能或位置数据，无法计算碳排放')
          errorMessages.push('缺少性能或位置数据，无法计算碳排放')
          hasError = true
        }

        // 初始化结果对象 - 只包含真实数据
        result.value = {
          hasError,
          errorMessages,
          onlyRealData: true, // 标记只显示真实数据
          provider: providerResult?.provider || 'unknown',
          isGreen: false,
          totalCarbonEmission: null,
          monthlyCarbonEmission: null,
          dataCenterEnergy: 0.0045, // 默认能源消耗值 (kWh)
          transmissionEnergy: 0.0012, // 默认网络传输能耗 (kWh)
          deviceEnergy: 0.0032,      // 默认设备能耗 (kWh)
          suggestions: [
            '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',
            '优化JavaScript和CSS文件，减少代码体积，提高页面加载速度',
            '将静态资源部署到CDN，减少服务器负载和网络传输距离',
            '优化和压缩图片，减少页面总体积和加载时间',
            '实施有效的浏览器缓存策略，减少重复资源的传输'
          ]
        }

```

```
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'
```

```
      } else if (metric === 'ttfb') {
        return value < 200 ? 'good' : value < 500 ? 'average' : 'poor'
      }
      return 'average'
    },

    // 获取进度条样式
    getProgressStyle(metric, value) {
      if (value === null || value === undefined) {
        return { width: `0%` }
      }

      let percentage

      if (metric === 'cls') {
        // CLS是越小越好，0.1是理想值，0.25是最大可接受值
        percentage = Math.min(100, (value / 0.25) * 100)
      } else if (metric === 'fcp') {
        // FCP在1秒以内为理想，3秒以上为较差
        percentage = Math.min(100, (value / 3) * 100)
      } else if (metric === 'lcp') {
        // LCP在2.5秒以内为理想，4秒以上为较差
        percentage = Math.min(100, (value / 4) * 100)
      } else if (metric === 'fid') {
        // FID在100ms以内为理想，300ms以上为较差
        percentage = Math.min(100, (value / 300) * 100)
      } else {
        // 默认值
        percentage = 50
      }

      return { width: `${percentage}%` }
    },

    // 获取进度条类
    getProgressClass(metric, value) {
      return this.getMetricGrade(metric, value)
    },

    // 格式化数字
    formatNumber(value) {
      if (value === null || value === undefined) return '无法获取'
      return this.safeToFixed(parseFloat(value), 2)
    },

    // 获取碳排放分数样式类
    getCarbonScoreClass(score) {
      if (!score) return 'average'
      score = parseInt(score)
      if (score <= 30) return 'excellent'
      if (score <= 50) return 'good'
      if (score <= 70) return 'moderate'
      return 'poor'
    },

```

```
    // 获取能源效率评分样式类
    getEnergyScoreClass(score) {
      if (!score) return 'average'
      score = parseInt(score)
      if (score ≥ 80) return 'excellent'
      if (score ≥ 60) return 'good'
      if (score ≥ 40) return 'moderate'
      return 'poor'
    },

    // 获取指标样式类
    getMetricClass(metric, value) {
      return this.getMetricGrade(metric, value)
    },

    // 判断是否为时间指标
    isTimingMetric(metric) {
      return ['fcp', 'lcp', 'fid', 'ttfb'].includes(metric)
    },

    // 获取进度条样式
    getProgressBarStyle(metric, value) {
      if (value === null || value === undefined) {
        return { width: '0%' }
      }

      let percentage = 0
      let grade = this.getMetricGrade(metric, value)

      if (metric === 'fcp' || metric === 'lcp') {
        // 转换为毫秒
        const ms = value * 1000
        percentage = Math.min(100, Math.max(10, (ms / 5000) * 100))
      } else if (metric === 'cls') {
        percentage = Math.min(100, Math.max(10, (value / 0.25) * 100))
      } else if (metric === 'fid' || metric === 'ttfb') {
        percentage = Math.min(100, Math.max(10, (value / 1000) * 100))
      }

      return {
        width: `${percentage}%`,
        backgroundColor: grade === 'good' ? 'var(--color-good)' :
                         grade === 'average' ? 'var(--color-moderate)' :
                         'var(--color-poor)'
      }
    },

    // 获取安全评分样式类
    getSecurityScoreClass(score) {
      if (!score && score !== 0) return 'poor'
      if (score ≥ 80) return 'excellent'
      if (score ≥ 60) return 'good'
      if (score ≥ 40) return 'moderate'
      return 'poor'
    },
```

```
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 !== 0) return 'N/A';
1054        if (length > 10000) return this.safeToFixed(length / 1000, 1) + 'K 字
      符';
1055        return length + ' 字符';
1056      },
1057
1058      // 获取内容质量评分的样式类
1059      getQualityScoreClass(score) {
1060        if (!score && score !== 0) return 'poor';
1061        if (score >= 80) return 'excellent';
1062        if (score >= 60) return 'good';
1063        if (score >= 40) return 'moderate';
1064        return 'poor';
1065      },
1066
1067      // 获取元数据状态
1068      getMetadataStatus(metadata) {
1069        if (!metadata) return '缺失';
1070
1071        const hasCount = (metadata.hasTitle ? 1 : 0) +
1072                         (metadata.hasDescription ? 1 : 0) +
1073                         (metadata.hasKeywords ? 1 : 0);
1074
1075        if (hasCount === 3) return '完整';
1076        if (hasCount >= 1) return '部分';
1077        return '缺失';
1078      },
1079
1080      // 获取元数据的样式类
1081      getMetadataClass(metadata) {
1082        if (!metadata) return 'poor';
1083
1084        const hasCount = (metadata.hasTitle ? 1 : 0) +
1085                         (metadata.hasDescription ? 1 : 0) +
1086                         (metadata.hasKeywords ? 1 : 0);
1087
1088        if (hasCount === 3) return 'good';
1089        if (hasCount >= 1) return 'moderate';
1090        return 'poor';
1091      },
1092
```

```
1093      // 获取资源类型的颜色
1094      getResourceColor(type) {
1095        const colors = {
1096          html: '#e34c26',
1097          css: '#563d7c',
1098          js: '#f1e05a',
1099          images: '#4caf50',
1100          fonts: '#ff9800',
1101          other: '#607d8b'
1102        };
1103        return colors[type] || '#607d8b';
1104      },
1105
1106      // 格式化资源类型名称
1107      formatResourceType(type) {
1108        const types = {
1109          html: 'HTML',
1110          css: 'CSS',
1111          js: 'JavaScript',
1112          images: '图像',
1113          fonts: '字体',
1114          other: '其他'
1115        };
1116        return types[type] || type;
1117      },
1118
1119      // 计算资源条形图的宽度
1120      getResourceBarWidth(count, allStats) {
1121        if (!allStats) return '0%';
1122        const total = Object.values(allStats).reduce((sum, c) => sum + c, 0);
1123        if (total === 0) return '0%';
1124        const percentage = (count / total) * 100;
1125        return Math.max(percentage, 3) + '%'; // 确保即使很小的值也有最小宽度
1126      },
1127
1128      // 获取性能指标（过滤掉非性能指标）
1129      getPerformanceMetrics(performance) {
1130        if (!performance) return {};
1131
1132        // 只选择性能指标来显示
1133        const metricsToShow = [
1134          'fcp', 'lcp', 'cls', 'fid', 'ttfb'
1135        ];
1136
1137        return Object.fromEntries(
1138          Object.entries(performance)
1139            .filter(([key]) => metricsToShow.includes(key))
1140        );
1141      },
1142
1143      // 格式化文件大小
1144      formatFileSize(bytes) {
1145        if (!bytes && bytes !== 0) return 'N/A';
1146        bytes = Number(bytes);
1147
```

```
        if (bytes ≥ 1048576) {
          return this.safeToFixed(bytes / 1048576, 2) + ' MB';
        }
        if (bytes ≥ 1024) {
          return this.safeToFixed(bytes / 1024, 2) + ' KB';
        }
        return bytes + ' B';
      },

      // 获取建议图标
      getSuggestionIcon(type) {
        const icons = {
          'image': 'fa-image',
          'script': 'fa-code',
          'font': 'fa-font',
          'cdn': 'fa-server',
          'cache': 'fa-database',
          'compress': 'fa-file-archive',
          'performance': 'fa-tachometer-alt',
          'default': 'fa-lightbulb'
        };
        return icons[type] || icons.default;
      },

      // 获取影响等级
      getImpactLevel(index) {
        const levels = ['低', '中', '高'];
        return levels[index % levels.length];
      },

      // 获取影响类
      getImpactClass(index) {
        const classes = ['low', 'medium', 'high'];
        return classes[index % classes.length];
      },

      // 获取指标描述
      getMetricDescription(metric) {
        const descriptions = {
          fcp: '首次内容绘制时间',
          lcp: '最大内容绘制时间',
          cls: '累计布局偏移',
          fid: '首次输入延迟',
          ttfb: '首次字节时间',
          responseTime: '服务器响应时间',
          pageSize: '页面总大小',
          totalResourceSize: '资源总大小',
          requestCount: '资源请求数量',
          domainCount: '请求域名数量',
          usesHttps: '使用HTTPS',
          serverType: '服务器类型',
          supportsCompression: '支持压缩',
          supportsCaching: '支持缓存',
          usesCdn: '使用CDN',
          cdnProvider: 'CDN提供商',
```

```
        resourceStats: '资源统计',
        securityScore: '安全评分',
        energyConsumption: '能源消耗',
        dataCenterEnergy: '数据中心能源',
        transmissionEnergy: '传输能源',
        deviceEnergy: '设备能源',
        carbonEmission: '碳排放量',
        annualCarbonEmission: '年度碳排放',
        renewablePercentage: '可再生能源比例',
        pue: '能源使用效率'
      };
      return descriptions[metric] || '未知描述';
    },

    // 格式化能源消耗值
    formatEnergyValue(value) {
      if (value === null || value === undefined) return '0.0000';
      const wattHours = value * 1000; // 转换为瓦时

      if (wattHours < 0.001) {
        // 小于1微瓦时，显示为微瓦时
        return `${this.safeToFixed(wattHours * 1000000, 2)} μWh`;
      } else if (wattHours < 1) {
        // 小于1毫瓦时，显示为毫瓦时
        return `${this.safeToFixed(wattHours * 1000, 2)} mWh`;
      } else {
        // 正常显示瓦时
        return `${this.safeToFixed(wattHours, 4)} Wh`;
      }
    },

    // 安全的格式化函数，防止对null或undefined调用toFixed
    safeToFixed(value, digits = 2) {
      if (value === null || value === undefined) return '0.00';
      return Number(value).toFixed(digits);
    },

    // 格式化数字
    formatNumber(value) {
      if (value === null || value === undefined) return '无法获取';
      return this.safeToFixed(parseFloat(value), 2);
    },
  }
}
</script>

<style>
/* 全局样式 */
.green-web-app {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  color: #333;
  background-color: #f9f9f9;
```

```
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

/* 头部样式 */
.app-header {
  text-align: center;
  margin-bottom: 40px;
  padding: 35px 0;
  background: linear-gradient(135deg, #34c759, #32ade6);
  color: white;
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(50, 173, 230, 0.25);
  position: relative;
  overflow: hidden;
  transform: translateZ(0);
}

.app-header::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background:
    radial-gradient(circle at 30% 40%, rgba(255,255,255,0.2) 0%, transparent 60%),
    radial-gradient(circle at 70% 60%, rgba(255,255,255,0.15) 0%, transparent 50%);
  z-index: 1;
}

.app-header::after {
  content: '';
  position: absolute;
  top: -50%;
  left: -50%;
  right: -50%;
  bottom: -50%;
  background: linear-gradient(0deg, rgba(255,255,255,0) 40%, rgba(255,255,255,0.1) 50%, rgba(255,255,255,0) 60%);
  z-index: 2;
  transform: rotateZ(-45deg);
  animation: shimmer 8s infinite linear;
  pointer-events: none;
}

@keyframes shimmer {
  0% { transform: translateX(-50%) rotateZ(-45deg); }
  100% { transform: translateX(100%) rotateZ(-45deg); }
}

.logo {
```

```
  width: 60px;
  height: 60px;
  background: rgba(255, 255, 255, 0.95);
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-right: 15px;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
  animation: float 5s ease-in-out infinite;
}

@keyframes float {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(-6px); }
}

.logo-icon {
  font-size: 35px;
}

.app-header h1 {
  margin: 0;
  font-size: 36px;
  font-weight: 700;
  position: relative;
  z-index: 3;
  text-shadow: 0 2px 4px rgba(0, 0, 0, 0.15);
  letter-spacing: 0.5px;
}

.subtitle {
  font-size: 17px;
  opacity: 0.95;
  margin-top: 8px;
  position: relative;
  z-index: 3;
  font-weight: 500;
  letter-spacing: 0.3px;
  text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
}

/* 输入区域样式 */
.input-section {
  margin-bottom: 40px;
  transform: translateY(-20px);
}

.url-input-group {
  display: flex;
  width: 100%;
  max-width: 800px;
  margin: 0 auto;
}

```

```
.domain-input {
  flex: 1;
  margin-right: 10px;
}

.domain-input :deep(.el-input__wrapper) {
  padding: 4px 15px;
  box-shadow: none !important;
}

.domain-input :deep(.el-input__inner) {
  height: 50px;
  font-size: 16px;
}

.domain-input :deep(.el-input-group__append) {
  padding: 0;
}

.domain-input :deep(.el-input-group__append button) {
  height: 58px;
  padding: 0 25px;
  font-size: 16px;
  font-weight: 600;
  border-radius: 0;
  box-shadow: none;
}

.input-hint {
  margin-top: 12px;
  color: #8c8c8c;
  font-size: 14px;
  text-align: center;
  transition: all 0.3s ease;
}

/* 加载动画 */
.loading-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  padding: 80px 0;
  animation: fadeIn 0.5s ease-out;
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(15px); }
  to { opacity: 1; transform: translateY(0); }
}

.earth-container {
  position: relative;
  width: 120px;
  height: 120px;
```

```
  margin-bottom: 30px;
}

.earth {
  width: 100%;
  height: 100%;
  border-radius: 50%;
  background: linear-gradient(135deg, #34c759, #32ade6);
  box-shadow:
    0 0 60px rgba(50, 173, 230, 0.4),
    0 0 30px rgba(52, 199, 89, 0.3),
    inset 0 0 15px rgba(255, 255, 255, 0.3);
  animation: spin 10s linear infinite, pulse 5s ease-in-out infinite;
  position: relative;
  overflow: hidden;
}

.earth::before {
  content: '';
  position: absolute;
  top: -10%;
  left: -10%;
  right: -10%;
  bottom: -10%;
  background-image: url('data:image/svg+xml;utf8,<svg width="100" height="100" viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"><rect x="0" y="0" width="10" height="10" fill="rgba(255,255,255,0.15)"/></svg>');
  background-size: 20px 20px;
  border-radius: 50%;
  animation: spin 20s linear infinite reverse;
  opacity: 0.7;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}

@keyframes pulse {
  0%, 100% { transform: scale(1); }
  50% { transform: scale(1.05); }
}

.loading-container p {
  font-size: 18px;
  font-weight: 500;
  margin-top: 0;
  text-align: center;
  color: #555;
  line-height: 1.6;
}

.loading-container small {
  display: block;
  margin-top: 8px;
```

```
  font-size: 14px;
  color: #888;
}

/* 结果摘要 */
.result-section {
  animation: fadeInUp 0.8s ease-out;
}

@keyframes fadeInUp {
  from { opacity: 0; transform: translateY(30px); }
  to { opacity: 1; transform: translateY(0); }
}

.result-summary {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  width: 100%;
  margin-bottom: 40px;
}

.summary-card {
  flex: 1;
  min-width: 300px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  overflow: hidden;
  transition: all 0.3s ease;
  display: block;
  padding: 0;
  align-items: initial;
  margin: 0;
}

.summary-card.green {
  border-top: 5px solid #4caf50;
  background: #fff;
}

.summary-card.red {
  border-top: 5px solid #f44336;
  background: #fff;
}

/* 移除冲突的伪元素 */
.summary-card::before,
.summary-card::after {
  display: none;
  content: none;
}

.summary-header {
  padding: 15px 20px;
```

```
  background-color: #f9f9f9;
  border-bottom: 1px solid #eee;
}

.summary-title {
  font-size: 18px;
  font-weight: 600;
  color: #333;
}

.summary-subtitle {
  font-size: 14px;
  color: #666;
  margin-top: 5px;
}

.summary-content {
  padding: 20px;
}

.summary-section {
  margin-bottom: 25px;
}

.summary-section:last-child {
  margin-bottom: 0;
}

/* 重置原来summary-content的样式 */
.summary-content h2,
.summary-content p {
  margin: 0;
  text-align: left;
  color: #333;
  text-shadow: none;
}

/* 总碳排放显示 */
.total-emission {
  display: flex;
  align-items: flex-start;
}

.carbon-icon {
  margin-right: 15px;
  color: #555;
  padding-top: 5px;
}

.carbon-details {
  flex: 1;
}

.total-title {
  font-size: 16px;
```

```
  font-weight: 500;
  color: #333;
  margin-bottom: 5px;
}

.total-value {
  font-size: 24px;
  font-weight: 700;
  color: #2c3e50;
  margin-bottom: 15px;
}

.total-breakdown {
  background-color: #f5f5f5;
  border-radius: 6px;
  padding: 12px 15px;
}

.total-item {
  display: flex;
  justify-content: space-between;
  margin-bottom: 8px;
  font-size: 14px;
}

.total-label {
  color: #555;
  font-weight: 500;
}

.total-item .total-value {
  font-size: 16px;
  margin-bottom: 0;
}

.total-info {
  font-size: 12px;
  color: #777;
  margin-bottom: 12px;
  padding-left: 5px;
}

/* 确保每个section内的最后一个元素没有margin-bottom */
.summary-section > *:last-child {
  margin-bottom: 0;
}

.summary-icon {
  width: 80px;
  height: 80px;
  background: rgba(255, 255, 255, 0.25);
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
```

```
  margin-right: 30px;
  position: relative;
  z-index: 1;
  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);
}

.summary-content {
  position: relative;
  z-index: 1;
}

.summary-content h2 {
  margin: 0;
  font-size: 32px;
  font-weight: 700;
  margin-bottom: 10px;
  text-shadow: 0 2px 4px rgba(0, 0, 0, 0.15);
}

.summary-content p {
  margin: 8px 0 0;
  font-size: 17px;
  opacity: 0.95;
  text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
}

/* 结果网格 */
.result-grid {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px 0;
  position: relative;
}

.result-card-container {
  display: flex;
  flex-direction: column;
  gap: 30px;
  margin-top: 20px;
}

.result-row {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
  gap: 25px;
  margin-bottom: 10px;
}

.result-card {
  background: #fff;
  border-radius: 18px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.05);
  overflow: hidden;
  transition: all 0.35s cubic-bezier(0.25, 0.8, 0.25, 1);
  display: flex;
```

```
    flex-direction: column;
    position: relative;
    border: 1px solid rgba(230, 230, 230, 0.6);
    backdrop-filter: blur(10px);
    padding: 25px;
    grid-column: span 4;
  }

  .result-card::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 6px;
    background: linear-gradient(90deg, #32ade6, #34c759);
    opacity: 0.8;
  }

  /* 不同类型卡片的顶部颜色条 */
  .result-card.energy-source::before {
    background: linear-gradient(90deg, #34c759, #4cd964);
  }

  .result-card.carbon-map::before {
    background: linear-gradient(90deg, #32ade6, #007aff);
  }

  .result-card.energy-consumption::before {
    background: linear-gradient(90deg, #5ac8fa, #007aff);
  }

  .result-card.carbon-impact::before {
    background: linear-gradient(90deg, #ff9500, #ff3b30);
  }

  .result-card.site-info::before {
    background: linear-gradient(90deg, #007aff, #5856d6);
  }

  .result-card.performance::before {
    background: linear-gradient(90deg, #5856d6, #af52de);
  }

  .result-card.suggestions::before {
    background: linear-gradient(90deg, #af52de, #ff2d55);
  }

  .result-card.green-hosting::before {
    background: linear-gradient(90deg, #34c759, #00c7be);
  }

  .result-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 15px 35px rgba(0, 0, 0, 0.07);
```

```
}

.result-card.wide-card {
  grid-column: span 6;
}

.result-card.major-card {
  grid-column: span 6;
}

.card-header {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  margin-bottom: 30px;
  position: relative;
}

.card-header h3 {
  margin: 0;
  font-size: 20px;
  font-weight: 600;
  color: #333;
  position: relative;
  padding-bottom: 15px;
}

.card-header h3::after {
  content: '';
  position: absolute;
  width: 40px;
  height: 3px;
  background: linear-gradient(to right, #34c759, #32ade6);
  left: 0;
  bottom: 0;
  border-radius: 3px;
}

.card-icon {
  width: 46px;
  height: 46px;
  border-radius: 14px;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(135deg, rgba(50, 173, 230, 0.1), rgba(52, 199, 89, 0.1));
  color: #32ade6;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.05);
  margin-left: 15px;
}

/* 卡片内容区域共享样式 */
.card-content {
  flex: 1;
```

```
  display: flex;
  flex-direction: column;
}

/* 数据展示样式 */
.details {
  display: flex;
  flex-direction: column;
  flex: 1;
  gap: 16px;
}

.detail-item {
  display: flex;
  justify-content: space-between;
  font-size: 15px;
  align-items: center;
  padding: 10px 0;
  border-bottom: 1px solid rgba(230, 230, 230, 0.5);
}

.detail-item:last-child {
  border-bottom: none;
}

.detail-label {
  color: #666;
  font-weight: 500;
}

.detail-value {
  font-weight: 600;
  color: #333;
  background: linear-gradient(135deg, #f9fafc, #f1f4f8);
  padding: 6px 14px;
  border-radius: 10px;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.03);
}

.detail-value.highlight-text {
  color: #32ade6;
  background: linear-gradient(135deg, rgba(50, 173, 230, 0.1), rgba(52, 199, 89, 0.05));
  font-weight: 700;
}

.green-value {
  color: #34c759 !important;
}

/* 能源消耗卡片样式 */
.energy-consumption-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
```

```
  margin-top: 20px;
}

.energy-consumption-item {
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  background: linear-gradient(135deg, #f9fafc, #f1f4f8);
  border-radius: 14px;
  padding: 20px 15px;
  transition: all 0.3s ease;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.03);
}

.energy-consumption-item:hover {
  transform: translateY(-3px);
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.08);
}

.consumption-icon {
  width: 50px;
  height: 50px;
  border-radius: 50%;
  margin-bottom: 12px;
  display: flex;
  align-items: center;
  justify-content: center;
  background-size: 60%;
  background-position: center;
  background-repeat: no-repeat;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);
}

.data-center-icon {
  background-color: rgba(50, 173, 230, 0.1);
  background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg" fill="%2332ADE6" viewBox="0 0 24 24"><path d="M4 1h16c1.1 0 2 .9 2 2v4c0 1.1-.9 2-2 2H4c-1.1 0-2-.9-2-2V3c0-1.1.9-2 2-2zm0 9h16c1.1 0 2 .9 2 2v4c0 1.1-.9 2-2 2H4c-1.1 0-2-.9-2-2v-4c0-1.1.9-2 2-2zm0 10c0-1.1.9-2 2-2h12c1.1 0 2 .9 2 2v0c0 1.1-.9 2-2 2H6c-1.1 0-2-.9-2-2v0z"/></svg>');
}

.network-icon {
  background-color: rgba(255, 149, 0, 0.1);
  background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg" fill="%23FF9500" viewBox="0 0 24 24"><path d="M12 21l-8-9h6V3h4v9h6l-8 9z"/></svg>');
}

.device-icon {
  background-color: rgba(52, 199, 89, 0.1);
  background-image: url('data:image/svg+xml;utf8,<svg 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>');
}

.consumption-value {
  font-size: 18px;
  font-weight: 700;
  color: #333;
  margin: 5px 0;
}

.consumption-label {
  font-size: 14px;
  color: #666;
}

/* 碳排放数据样式 */
.carbon-stats {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
  margin-top: 10px;
}

.carbon-stat-item {
  display: flex;
  flex-direction: column;
  font-size: 14px;
  background: linear-gradient(135deg, #f9fafc, #f1f4f8);
  border-radius: 14px;
  padding: 16px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.03);
  transition: all 0.3s ease;
}

.carbon-stat-item:hover {
  transform: translateY(-3px);
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.08);
}

.stat-label {
  color: #666;
  margin-bottom: 10px;
  font-weight: 500;
}

.stat-value {
  font-weight: 700;
  color: #333;
  font-size: 18px;
}

.carbon-annual-impact {
  grid-column: span 2;
```

```
  background: linear-gradient(135deg, rgba(50, 173, 230, 0.05), rgba(52, 199, 89, 0.05));
  border-radius: 14px;
  padding: 20px;
  margin-top: 10px;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.05);
  border: 1px dashed rgba(50, 173, 230, 0.2);
  text-align: center;
}

.impact-title {
  font-weight: 600;
  color: #333;
  margin-bottom: 10px;
  font-size: 16px;
}

.impact-value {
  font-size: 24px;
  font-weight: 700;
  color: #32ade6;
  margin: 10px 0;
}

.impact-equivalent {
  font-size: 14px;
  color: #666;
}

.highlight-text {
  color: #34c759;
  font-weight: 700;
}

/* 能源图表 */
.energy-chart {
  display: flex;
  flex-direction: column;
  align-items: center;
  margin: 25px 0;
}

.donut-chart {
  position: relative;
  width: 170px;
  height: 170px;
  filter: drop-shadow(0 10px 20px rgba(0, 0, 0, 0.15));
}

.donut-hole {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 110px;
```

```
  height: 110px;
  background: #fff;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 32px;
  font-weight: bold;
  color: #34c759;
  box-shadow:
    inset 0 0 20px rgba(0, 0, 0, 0.05),
    0 6px 20px rgba(52, 199, 89, 0.15);
}

.donut-ring {
  width: 100%;
  height: 100%;
  border-radius: 50%;
  background: #ff3b30;
  filter: drop-shadow(0 6px 12px rgba(255, 59, 48, 0.25));
}

.renewable {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  border-radius: 50%;
  background: conic-gradient(#34c759 var(--percent), transparent 0);
  --percent: 0%;
  filter: drop-shadow(0 6px 12px rgba(52, 199, 89, 0.25));
  animation: fillDonut 1.5s ease-out forwards;
}

@keyframes fillDonut {
  from { opacity: 0; transform: scale(0.8); }
  to { opacity: 1; transform: scale(1); }
}

.chart-legend {
  display: flex;
  justify-content: center;
  gap: 30px;
  margin-top: 25px;
}

.legend-item {
  display: flex;
  align-items: center;
  gap: 10px;
  font-size: 15px;
  font-weight: 500;
}

```

```
.legend-color {
  width: 18px;
  height: 18px;
  border-radius: 5px;
}

.legend-color.renewable {
  background-color: #34c759;
  box-shadow: 0 3px 8px rgba(52, 199, 89, 0.3);
}

.legend-color.fossil {
  background-color: #ff3b30;
  box-shadow: 0 3px 8px rgba(255, 59, 48, 0.3);
}

/* 可用性提示 */
.data-unavailable {
  color: #888;
  font-style: italic;
  padding: 20px;
  background-color: #f9fafc;
  border-radius: 14px;
  font-size: 15px;
  text-align: center;
  margin: 15px 0;
  box-shadow: inset 0 0 15px rgba(0, 0, 0, 0.03);
  border: 1px dashed rgba(0, 0, 0, 0.1);
}

/* 卡片摘要 */
.card-summary {
  margin-top: auto;
  padding-top: 20px;
  text-align: center;
}

.summary-badge {
  display: inline-block;
  padding: 8px 16px;
  border-radius: 30px;
  font-size: 14px;
  font-weight: 600;
  margin-bottom: 10px;
}

.green-badge {
  background: linear-gradient(135deg, rgba(52, 199, 89, 0.15), rgba(50, 173, 230, 0.15));
  color: #34c759;
  box-shadow: 0 4px 10px rgba(52, 199, 89, 0.2);
}

.standard-badge {
```

```
  background: linear-gradient(135deg, rgba(255, 149, 0, 0.15), rgba(255, 59, 48, 0.15));
  color: #ff9500;
  box-shadow: 0 4px 10px rgba(255, 149, 0, 0.2);
}

/* 响应式调整 */
@media (max-width: 1200px) {
  .result-row {
    grid-template-columns: repeat(6, 1fr);
  }

  .result-card {
    grid-column: span 3;
  }

  .result-card.wide-card,
  .result-card.major-card {
    grid-column: span 6;
  }
}

@media (max-width: 768px) {
  .result-row {
    grid-template-columns: repeat(1, 1fr);
    gap: 20px;
  }

  .result-card,
  .result-card.wide-card,
  .result-card.major-card {
    grid-column: span 1;
  }

  .carbon-stats,
  .energy-consumption-grid {
    grid-template-columns: 1fr;
  }

  .carbon-annual-impact {
    grid-column: span 1;
  }
}

/* 能源消耗数据提示 */
.energy-note {
  padding: 15px;
  background-color: #f9fafc;
  border-radius: 12px;
  margin-top: 20px;
  font-size: 14px;
  color: #666;
  border: 1px dashed rgba(0, 0, 0, 0.1);
}

```

```
/* 性能指标和优化建议布局 */
.performance-metrics-v2 {
  display: flex;
  flex-direction: column;
  gap: 16px;
  margin-top: 15px;
  animation: fadeInUp 0.6s ease-out forwards;
}

.metrics-grid {
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
}

.metric-card {
  flex: 1;
  min-width: 200px;
  background-color: #fff;
  border-radius: 16px;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.06);
  overflow: hidden;
  transition: all 0.3s ease;
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  margin-bottom: 10px;
  border: 1px solid rgba(0, 0, 0, 0.03);
}

.metric-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 12px 28px rgba(0, 0, 0, 0.1);
}

.metric-visual {
  width: 100%;
  padding: 25px 15px;
  background: linear-gradient(135deg, #f9fafc, #f1f4f8);
  border-bottom: 1px solid rgba(0, 0, 0, 0.05);
  display: flex;
  justify-content: center;
  align-items: center;
}

.metric-circle {
  width: 110px;
  height: 110px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 28px;
  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);
```

```
  margin-top: 15px;
  border: 1px solid rgba(0, 0, 0, 0.03);
}

.network-title {
  font-size: 17px;
  color: #333;
  font-weight: 600;
  margin-bottom: 15px;
  padding-bottom: 10px;
  border-bottom: 1px solid rgba(0, 0, 0, 0.08);
  position: relative;
}

.network-title::after {
  content: '';
  position: absolute;
  width: 40px;
  height: 3px;
  background: linear-gradient(to right, #34c759, #32ade6);
  left: 0;
  bottom: -1px;
  border-radius: 3px;
}

.network-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  gap: 15px;
}

.network-item {
  background: #fff;
  border-radius: 12px;
  padding: 15px;
  box-shadow: 0 6px 16px rgba(0, 0, 0, 0.04);
  transition: all 0.3s ease;
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  border: 1px solid rgba(0, 0, 0, 0.03);
}

.network-item:hover {
  transform: translateY(-3px);
  box-shadow: 0 10px 24px rgba(0, 0, 0, 0.06);
}

.network-value {
  font-size: 22px;
  font-weight: 700;
  color: #32ade6;
  margin-bottom: 8px;
}
```

```
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 {
```

```
  margin-right: 12px;
  width: 38px;
  height: 38px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(135deg, rgba(50, 173, 230, 0.15), rgba(52, 199, 89, 0.15));
  color: #32ade6;
  box-shadow: 0 4px 8px rgba(50, 173, 230, 0.15);
}

.suggestion-title {
  font-size: 16px;
  font-weight: 600;
  color: #333;
  letter-spacing: 0.3px;
}

.suggestion-content {
  padding: 20px;
  color: #555;
  font-size: 15px;
  line-height: 1.7;
  flex: 1;
  background-color: #fff;
  position: relative;
}

.suggestion-content::before {
  content: '';
  position: absolute;
  left: 20px;
  right: 20px;
  height: 2px;
  top: 0;
  background: linear-gradient(to right, rgba(50, 173, 230, 0.1), transparent);
  border-radius: 2px;
}

.suggestion-impact {
  padding: 12px;
  text-align: center;
  font-weight: 600;
  font-size: 14px;
  color: #fff;
  letter-spacing: 0.5px;
  text-transform: uppercase;
  display: flex;
  align-items: center;
  justify-content: center;
}

```

```
.suggestion-impact.low {
  background: linear-gradient(to right, #4caf50, #66bb6a);
}

.suggestion-impact.medium {
  background: linear-gradient(to right, #ff9800, #ffa726);
}

.suggestion-impact.high {
  background: linear-gradient(to right, #f44336, #ef5350);
}

@media (max-width: 768px) {
  .suggestions-list {
    grid-template-columns: 1fr;
  }

  .metrics-grid {
    flex-direction: column;
  }

  .metric-card {
    width: 100%;
    min-width: 100%;
  }
}

.energy-note {
  padding: 15px;
  background-color: #f5f7fa;
  border-radius: 6px;
  margin-top: 20px;
  font-size: 14px;
  color: #666;
}

/* 页脚样式 */
.footer {
  margin-top: 80px;
  padding: 30px 0;
  color: #666;
  font-size: 14px;
  border-top: 1px solid #eee;
  text-align: center;
  background: linear-gradient(180deg, transparent, rgba(240, 240, 240,
0.5));
  border-radius: 0 0 16px 16px;
}

.copyright {
  margin-top: 10px;
  font-size: 13px;
  opacity: 0.8;
}

```

```
/* 响应式调整 */
@media (max-width: 768px) {
  .app-header {
    padding: 25px 0;
  }

  .app-header h1 {
    font-size: 28px;
  }

  .logo {
    width: 50px;
    height: 50px;
  }

  .logo-icon {
    font-size: 28px;
  }

  .url-input-group {
    flex-direction: column;
  }

  .domain-input {
    margin-right: 0;
    margin-bottom: 10px;
  }

  .browser-select {
    width: 100%;
    margin-bottom: 10px;
    margin-right: 0;
  }

  .result-grid {
    grid-template-columns: 1fr;
    gap: 20px;
  }

  .result-card {
    padding: 25px;
  }

  .carbon-stats {
    grid-template-columns: 1fr;
  }

  .summary-icon {
    width: 60px;
    height: 60px;
    margin-right: 20px;
  }

  .summary-content h2 {
    font-size: 26px;
```

```
  }

  .summary-card {
    padding: 25px;
  }

  .score-container {
    flex-direction: column;
    align-items: center;
  }

  .carbon-score-card {
    width: 90%;
    margin-bottom: 10px;
  }

  .donut-chart {
    width: 120px;
    height: 120px;
  }

  .donut-hole {
    width: 70px;
    height: 70px;
    font-size: 22px;
  }
}

@media (max-width: 480px) {
  .app-header h1 {
    font-size: 24px;
  }

  .logo {
    width: 40px;
    height: 40px;
    margin-right: 10px;
  }

  .logo-icon {
    font-size: 22px;
  }

  .result-card {
    padding: 20px;
  }

  .summary-icon {
    width: 50px;
    height: 50px;
    margin-right: 15px;
  }

  .summary-content h2 {
    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;
```

```
  text-align: center;
  position: relative;
  overflow: hidden;
  transition: all 0.3s ease;
}

.carbon-score-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.12);
}

.carbon-score-card::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  height: 6px;
  background: linear-gradient(90deg, #ff3b30, #ff9500);
}

.carbon-score-card.excellent::before {
  background: linear-gradient(90deg, #4cd964, #34c759);
}

.carbon-score-card.good::before {
  background: linear-gradient(90deg, #34c759, #5ac8fa);
}

.carbon-score-card.moderate::before {
  background: linear-gradient(90deg, #ffcc00, #ff9500);
}

.carbon-score-card.poor::before {
  background: linear-gradient(90deg, #ff3b30, #ff9500);
}

.score-circle {
  font-size: 24px;
  font-weight: 700;
  background: #f5f7fa;
  width: 60px;
  height: 60px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-bottom: 10px;
  color: #333;
  border: 3px solid #eee;
}

.carbon-score-card.excellent .score-circle {
  background: rgba(76, 217, 100, 0.15);
  color: #34c759;
```

```
  border-color: rgba(76, 217, 100, 0.3);
}

.carbon-score-card.good .score-circle {
  background: rgba(90, 200, 250, 0.15);
  color: #007aff;
  border-color: rgba(90, 200, 250, 0.3);
}

.carbon-score-card.moderate .score-circle {
  background: rgba(255, 204, 0, 0.15);
  color: #ff9500;
  border-color: rgba(255, 204, 0, 0.3);
}

.carbon-score-card.poor .score-circle {
  background: rgba(255, 59, 48, 0.15);
  color: #ff3b30;
  border-color: rgba(255, 59, 48, 0.3);
}

.score-label {
  font-size: 14px;
  color: #555;
  margin-bottom: 5px;
  font-weight: 500;
}

.score-desc {
  font-size: 13px;
  color: #888;
}

/* 错误提示样式 */
.error-alerts {
  margin-bottom: 20px;
  width: 100%;
}

.error-alert {
  background-color: #fff8f8;
  border-left: 4px solid #ff5252;
  padding: 15px;
  border-radius: 4px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.error-alert h3 {
  color: #d32f2f;
  margin-top: 0;
  margin-bottom: 10px;
  font-size: 16px;
}

.error-alert ul {
```

```
  margin: 10px 0;
  padding-left: 20px;
}

.error-alert li {
  margin-bottom: 5px;
  color: #616161;
}

/* 数据真实性提示样式 */
.data-note {
  background-color: #f0f4ff;
  border-left: 4px solid #5c6bc0;
  padding: 10px 15px;
  margin-bottom: 20px;
  border-radius: 4px;
  font-size: 14px;
}

.estimated-data-tag {
  display: inline-block;
  background-color: #ffecb3;
  color: #795548;
  font-size: 11px;
  padding: 2px 6px;
  border-radius: 10px;
  margin-left: 6px;
  vertical-align: middle;
  font-weight: bold;
}

/* 能源消耗详情 */
.energy-breakdown {
  background-color: #f5f7fa;
  border-radius: 6px;
  padding: 15px;
}

.energy-title {
  font-size: 16px;
  font-weight: 500;
  color: #333;
  margin-bottom: 12px;
}

.energy-items {
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
}

.energy-item {
  flex: 1;
  min-width: 120px;
  background-color: #fff;
```

```
  padding: 10px;
  border-radius: 6px;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.05);
}

.energy-label {
  font-size: 13px;
  color: #666;
  margin-bottom: 5px;
}

.energy-value {
  font-size: 15px;
  font-weight: 600;
  color: #2c3e50;
}

/* 详情卡片样式 */
.detail-card {
  flex: 1;
  min-width: 300px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  overflow: hidden;
}

.detail-header {
  padding: 15px 20px;
  background-color: #f9f9f9;
  border-bottom: 1px solid #eee;
}

.detail-title {
  font-size: 18px;
  font-weight: 600;
  color: #333;
}

.detail-content {
  padding: 20px;
}

/* 托管提供商信息 */
.provider-info {
  display: flex;
  align-items: flex-start;
}

.provider-icon {
  margin-right: 15px;
  color: #555;
  padding-top: 5px;
}

```

```
.provider-details {
  flex: 1;
}

.provider-name {
  font-size: 18px;
  font-weight: 600;
  color: #2c3e50;
  margin-bottom: 10px;
}

.provider-stats {
  background-color: #f5f5f5;
  border-radius: 6px;
  padding: 12px 15px;
}

.provider-item {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
  font-size: 14px;
}

.provider-label {
  color: #555;
  font-weight: 500;
}

.provider-value {
  font-weight: 600;
  color: #2c3e50;
}

/* 移动端响应式 */
@media (max-width: 768px) {
  .result-summary {
    flex-direction: column;
  }

  .total-emission,
  .provider-info {
    flex-direction: column;
  }

  .carbon-icon,
  .provider-icon {
    margin-right: 0;
    margin-bottom: 15px;
  }

  .energy-items {
    flex-direction: column;
  }

```

```
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%;
```

```
}

.result-card {
  flex: 1;
  min-width: 320px;
  background: white;
  border-radius: 16px;
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.08);
  transition: all 0.4s cubic-bezier(0.175, 0.885, 0.32, 1.275);
  position: relative;
  overflow: hidden;
  border: 1px solid rgba(230, 230, 230, 0.7);
  display: flex;
  flex-direction: column;
  padding: 0;
}

.result-card::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  height: 4px;
  background: linear-gradient(to right, #34c759, #32ade6);
  opacity: 0.8;
}

.result-card:hover {
  transform: translateY(-8px);
  box-shadow: 0 20px 40px rgba(0, 0, 0, 0.12);
}

.major-card {
  min-height: 380px;
}

.wide-card {
  width: 100%;
  flex-basis: 100%;
}

.card-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 20px 25px;
  border-bottom: 1px solid #f0f0f0;
  background-color: #fcfcfc;
}

.card-header h3 {
  margin: 0;
  font-size: 18px;
  font-weight: 600;
```

```
  color: #333;
  position: relative;
}

.card-icon {
  width: 38px;
  height: 38px;
  border-radius: 10px;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(135deg, #f5f7fa, #e4e7eb);
  color: #32ade6;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.06);
}

.result-card .details,
.result-card .carbon-stats,
.result-card .performance-metrics,
.result-card .suggestion-list,
.result-card .energy-chart,
.result-card .detail-content {
  padding: 25px;
  flex: 1;
  display: flex;
  flex-direction: column;
}

/* 详情项目样式 */
.detail-item {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 12px 0;
  border-bottom: 1px solid #f0f0f0;
}

.detail-item:last-child {
  border-bottom: none;
}

.detail-label {
  color: #666;
  font-weight: 500;
  font-size: 14px;
}

.detail-value {
  font-weight: 600;
  color: #333;
  background: linear-gradient(135deg, #f5f7fa, #e4e7eb);
  padding: 6px 12px;
  border-radius: 6px;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.05);
  font-size: 14px;
```

```
}

.highlight-text {
  color: #34c759;
  font-weight: bold;
}

/* 卡片摘要样式 */
.card-summary {
  padding: 20px 25px;
  background-color: #f9f9f9;
  margin-top: auto;
  border-top: 1px solid #f0f0f0;
}

.card-summary p {
  margin: 10px 0 0;
  color: #666;
  font-size: 14px;
}

.summary-badge {
  display: inline-block;
  padding: 6px 12px;
  border-radius: 20px;
  font-size: 14px;
  font-weight: 600;
  margin-bottom: 5px;
}

.green-badge {
  background-color: rgba(52, 199, 89, 0.15);
  color: #34c759;
}

.standard-badge {
  background-color: rgba(255, 149, 0, 0.15);
  color: #ff9500;
}

/* 能源消耗网格 */
.energy-consumption-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));
  gap: 20px;
  padding: 25px;
}

.energy-consumption-item {
  background: linear-gradient(135deg, #f9f9f9, #f5f5f5);
  border-radius: 12px;
  padding: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
```

```
  text-align: center;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.05);
  transition: transform 0.3s ease;
}

.energy-consumption-item:hover {
  transform: translateY(-5px);
}

.consumption-icon {
  width: 50px;
  height: 50px;
  border-radius: 25px;
  margin-bottom: 15px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: 25px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
}

.data-center-icon {
  background-color: rgba(52, 199, 89, 0.15);
  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>');
}

.network-icon {
  background-color: rgba(90, 200, 250, 0.15);
  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>');
}

.device-icon {
  background-color: rgba(255, 149, 0, 0.15);
  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>');
}

.consumption-value {
  font-size: 18px;
  font-weight: 700;
  color: #333;
  margin-bottom: 5px;
}

```

```
.consumption-label {
  font-size: 14px;
  color: #666;
}

/* 碳排放详情 */
.carbon-impact .carbon-stats {
  padding: 20px 25px;
}

.carbon-annual-impact {
  margin-top: 20px;
  padding: 20px;
  background: linear-gradient(135deg, #f0f9f4, #e6f7eb);
  border-radius: 12px;
  text-align: center;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.05);
}

.impact-title {
  font-size: 15px;
  font-weight: 600;
  color: #333;
  margin-bottom: 10px;
}

.impact-value {
  font-size: 24px;
  font-weight: 700;
  color: #34c759;
  margin-bottom: 5px;
}

.impact-equivalent {
  font-size: 14px;
  color: #666;
  line-height: 1.6;
}

/* 碳排放评分 */
.total-emission-highlight {
  background: linear-gradient(135deg, #f0f9f4, #e6f7eb);
  padding: 25px;
  margin: 20px 25px;
  border-radius: 12px;
  text-align: center;
  animation: pulse 3s infinite alternate;
}

@keyframes pulse {
  0% { box-shadow: 0 4px 15px rgba(52, 199, 89, 0.1); }
  100% { box-shadow: 0 4px 25px rgba(52, 199, 89, 0.25); }
}

.highlight-value {
```

```
  font-size: 28px;
  font-weight: 700;
  color: #34c759;
  margin-bottom: 5px;
}

.highlight-value span {
  font-size: 16px;
  font-weight: 500;
  color: #666;
}

.highlight-label {
  font-size: 14px;
  color: #666;
}

/* 数据源信息 */
.data-source-info {
  padding: 10px 25px;
  background-color: #f9f9f9;
  color: #888;
  font-size: 13px;
  font-style: italic;
  margin-top: auto;
  border-top: 1px solid #f0f0f0;
}

/* 响应式调整 */
@media (max-width: 1024px) {
  .result-row {
    flex-direction: column;
  }

  .result-card {
    width: 100%;
    min-width: 100%;
  }
}

@media (max-width: 768px) {
  .energy-consumption-grid {
    grid-template-columns: 1fr;
  }

  .score-container {
    flex-direction: column;
  }

  .carbon-score-card {
    width: 100%;
    margin-bottom: 15px;
  }
}
```

```
3308 </style>
```

```
/**
 * GreenWeb后端代理服务器
 * 用于处理跨域请求和获取网站性能数据
 */

const express = require('express');
const axios = require('axios');
const cors = require('cors');
const dns = require('dns').promises;
const { performance } = require('perf_hooks');
const { URL } = require('url');
const geoip = require('geoip-lite');
const whois = require('whois-json');
const useragent = require('express-useragent');
const path = require('path');
const https = require('https');
const os = require('os');
const { execSync } = require('child_process');
const puppeteer = require('puppeteer');
const fetch = require('node-fetch'); // 用于API请求

// 声明chromeLauncher变量，稍后通过动态导入获取
let chromeLauncher;

// 异步初始化函数用于导入ESM模块
async function initializeESModules() {
  try {
    // 动态导入chrome-launcher
    chromeLauncher = await import('chrome-launcher');
    console.log('Chrome Launcher模块已成功导入');
  } catch (err) {
    console.error('导入ESM模块失败:', err);
  }
}

// 执行初始化
initializeESModules().catch(err => {
  console.error('初始化ESM模块时出错:', err);
});

const app = express();
const PORT = process.env.PORT || 3000;

// 中间件
app.use(cors());
app.use(express.json());
app.use(useragent.express());
app.use(express.static(path.join(__dirname, '../dist')));

// 创建不验证SSL证书的axios实例
const axiosInstance = axios.create({
  httpsAgent: new https.Agent({
    rejectUnauthorized: false
```

```
  }),
  timeout: 10000
});

// 全局常量 - 用于碳排放计算
const GLOBAL_CONSTANTS = {
  // 能源消耗常量
  averageEnergyConsumption: 1.805, // kWh/GB
  greenEnergyCarbonIntensity: 50, // gCO2e/kWh
  averageCarbonIntensity: 475, // gCO2e/kWh

  // 数据中心效率
  averagePUE: 1.67, // 电能使用效率
  bestPUE: 1.1,

  // 传输和设备能耗
  averageTransmissionPerGB: 0.06, // kWh/GB
  averageDevicePerGB: 0.08, // kWh/GB

  // 缓存效率
  cachingEfficiency: 0.2, // 20%的流量被缓存

  // 访问量估算
  averageMonthlyVisits: 10000,

  // 碳中和阈值
  greenEnergyThreshold: 80, // 80%以上可再生能源视为碳中和

  // 每年一棵树可吸收二氧化碳量（kg）
  treeCO2PerYear: 25
};

// 国家碳强度数据（gCO2e/kWh）
const COUNTRY_CARBON_INTENSITY = {
  'US': 383,
  'CN': 554,
  'IN': 739,
  'JP': 478,
  'DE': 344,
  'GB': 231,
  'FR': 56,
  'IT': 331,
  'BR': 87,
  'CA': 135,
  'KR': 415,
  'RU': 351,
  'AU': 656,
  'ES': 200,
  'MX': 428,
  'ID': 736,
  'NL': 358,
  'SA': 523,
  'CH': 24,
  'TR': 461,
  'SE': 13,
```

```
  'PL': 751,
  'BE': 161,
  'TH': 471,
  'AT': 109,
  'IE': 291,
  'SG': 418,
  'IL': 529,
  'DK': 135,
  'FI': 89,
  'NO': 8,
  // 默认值将使用全球平均值
};

// 主要云服务提供商信息
const PROVIDER_INFO = {
  'aws': {
    name: 'Amazon Web Services',
    renewableRange: [60, 85],
    renewableChance: 0.8,
    pueRange: [1.1, 1.4],
    serverEfficiency: 1.2,
    regions: ['us-east', 'us-west', 'eu-west', 'eu-central', 'ap-northeast', 'ap-southeast', 'sa-east']
  },
  'azure': {
    name: 'Microsoft Azure',
    renewableRange: [65, 90],
    renewableChance: 0.85,
    pueRange: [1.1, 1.35],
    serverEfficiency: 1.25,
    regions: ['us-east', 'us-west', 'eu-west', 'eu-north', 'asia-east', 'asia-southeast', 'australia-east']
  },
  'google': {
    name: 'Google Cloud',
    renewableRange: [80, 95],
    renewableChance: 0.9,
    pueRange: [1.1, 1.2],
    serverEfficiency: 1.3,
    regions: ['us-central', 'us-east', 'us-west', 'europe-west', 'europe-north', 'asia-east', 'asia-south']
  },
  'cloudflare': {
    name: 'Cloudflare',
    renewableRange: [70, 90],
    renewableChance: 0.8,
    pueRange: [1.1, 1.3],
    serverEfficiency: 1.2,
    regions: ['global-edge']
  },
  'alibaba': {
    name: 'Alibaba Cloud',
    renewableRange: [45, 70],
    renewableChance: 0.6,
    pueRange: [1.3, 1.6],
```

```
    serverEfficiency: 1.1,
    regions: ['cn-hangzhou', 'cn-shanghai', 'cn-beijing', 'us-west', 'eu-central', 'ap-southeast']
  },
  'tencent': {
    name: 'Tencent Cloud',
    renewableRange: [40, 65],
    renewableChance: 0.5,
    pueRange: [1.35, 1.65],
    serverEfficiency: 1.0,
    regions: ['ap-guangzhou', 'ap-shanghai', 'ap-beijing', 'na-siliconvalley', 'eu-frankfurt', 'ap-singapore']
  },
  'other': {
    name: 'Unknown Provider',
    renewableRange: [30, 60],
    renewableChance: 0.4,
    pueRange: [1.5, 1.9],
    serverEfficiency: 0.9,
    regions: ['unknown']
  }
};

/**
 * 健康检查端点
 */
app.get('/api/health', (req, res) => {
  res.json({ status: 'ok', timestamp: new Date().toISOString() });
});

/**
 * 获取服务器位置信息
 */
app.get('/api/location', async (req, res) => {
  try {
    const { domain } = req.query;

    if (!domain) {
      return res.status(400).json({ error: '缺少域名参数' });
    }

    // 使用DNS查询获取IP地址
    const addresses = await dns.lookup(domain, { all: true });
    const ip = addresses[0]?.address;

    if (!ip) {
      return res.status(404).json({ error: '无法解析域名' });
    }

    // 使用GeoIP获取位置信息
    const geo = geoip.lookup(ip);

    // 尝试使用WHOIS获取额外信息
    let whoisData = {};
    try {
```

```
      whoisData = await whois(domain);
    } catch (whoisError) {
      console.error('WHOIS查询失败:', whoisError);
    }

    res.json({
      ip,
      country: geo?.country || null,
      region: geo?.region || null,
      city: geo?.city || null,
      timezone: geo?.timezone || null,
      coordinates: geo?.ll || null,
      org: geo?.org || null,
      registrar: whoisData.registrar || null,
      registrantCountry: whoisData.registrantCountry || null
    });
  } catch (error) {
    console.error('位置查询错误:', error);
    res.status(500).json({ error: '位置查询失败', message: error.message });
  }
});

/**
 * 获取HTTP响应头信息
 */
app.get('/api/headers', async (req, res) => {
  try {
    const { url } = req.query;

    if (!url) {
      return res.status(400).json({ error: '缺少URL参数' });
    }

    const startTime = performance.now();
    const response = await axiosInstance.head(url, {
      maxRedirects: 5,
      validateStatus: null
    });
    const endTime = performance.now();

    res.json({
      status: response.status,
      statusText: response.statusText,
      headers: response.headers,
      responseTime: endTime - startTime
    });
  } catch (error) {
    console.error('头信息获取错误:', error);
    res.status(500).json({ error: '头信息获取失败', message: error.message });
  }
});

/**
 * 测量页面大小
 */
```

```
app.get('/api/size', async (req, res) => {
  try {
    const { url } = req.query;

    if (!url) {
      return res.status(400).json({ error: '缺少URL参数' });
    }

    const startTime = performance.now();
    const response = await axiosInstance.get(url, {
      maxRedirects: 5,
      validateStatus: null,
      responseType: 'arraybuffer' // 获取二进制响应以精确计算大小
    });
    const endTime = performance.now();

    // 计算页面大小（KB）
    const contentLength = response.headers['content-length'] ||
response.data.length;
    const sizeInKB = Math.round(contentLength / 1024);

    res.json({
      size: sizeInKB,
      contentType: response.headers['content-type'],
      responseTime: endTime - startTime
    });
  } catch (error) {
    console.error('页面大小测量错误:', error);
    res.status(500).json({ error: '页面大小测量失败', message: error.message
});
  }
});

/**
 * 分析服务提供商
 */
app.get('/api/provider', async (req, res) => {
  try {
    const { domain } = req.query;

    if (!domain) {
      return res.status(400).json({ error: '缺少域名参数' });
    }

    // 使用DNS和WHOIS信息尝试确定服务提供商
    const addresses = await dns.lookup(domain, { all: true });
    const ip = addresses[0]?.address;

    if (!ip) {
      return res.status(404).json({ error: '无法解析域名' });
    }

    // 使用GeoIP获取组织信息
    const geo = geoip.lookup(ip);
    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
```

```
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 
```

```
    // 检查其他必要的真实数据
    if (!requestCount || !domainCount) {
      return res.status(400).json({
        error: '缺少网络请求数据',
        measurable: false,
        message: '无法获取网络请求和域名数量，无法准确计算碳排放'
      });
    }

    // 使用实际测量值，不进行估算
    const pageSizeKB = parseInt(pageSize);
    const countryCode = country || null; // 不提供默认值，如果没有则返回错误
    const renewable = parseFloat(renewablePercentage);
    const dataCenterPUE = parseFloat(safeToFixed(pue, 2));
    const actualRequestCount = parseInt(requestCount);
    const actualDomainCount = parseInt(domainCount);
    const actualResponseTime = parseInt(responseTime) || 0;
    const isCompressed = hasCompression === 'true' || hasCompression === true;

    // 检查国家信息
    if (!countryCode) {
      return res.status(400).json({
        error: '缺少国家/地区信息',
        measurable: false,
        message: '无法获取服务器地理位置，无法准确计算碳排放'
      });
    }

    // 检查可再生能源信息
    if (isNaN(renewable)) {
      return res.status(400).json({
        error: '缺少可再生能源使用比例',
        measurable: false,
        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'
      });
    }

    // 检查PUE信息
    if (isNaN(dataCenterPUE)) {
      return res.status(400).json({
        error: '缺少数据中心PUE',
        measurable: false,
        message: '无法获取数据中心PUE，无法准确计算碳排放'
      });
    }

    // 解析资源统计数据 - 只使用真实值
    let resourceStatsData = {};
    if (resourceStats) {
      try {
        resourceStatsData = typeof resourceStats === 'string' ? JSON.parse(resourceStats) : resourceStats;
      } catch (e) {
        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] ||
    null;
576       if (!countryCarbonValue) {
577         return res.status(400).json({
578           error: '无法获取国家电网碳强度',
579           measurable: false,
580           message: `无法获取 ${countryCode} 的电网碳强度数据，无法准确计算碳排放`
581         });
582       }
583
584       // 基于真实测量值计算能源消耗
585       const baseEnergyIntensity = GLOBAL_CONSTANTS.averageEnergyConsumption;
586       const energyIntensity = baseEnergyIntensity * dataCenterPUE;
587
```

```
    // 计算能源消耗 - 确保即使页面很小也有最小值
    // 使用Math.max确保能源消耗有一个最小基准值
    const pageSizeInGBSafe = Math.max(pageSizeInGB, 0.0001); // 确保至少有0.1MB
    const dataCenterEnergy = pageSizeInGBSafe * (energyIntensity / dataCenterPUE);
    const transmissionEnergy = pageSizeInGBSafe * GLOBAL_CONSTANTS.averageTransmissionPerGB;
    const deviceEnergy = pageSizeInGBSafe * GLOBAL_CONSTANTS.averageDevicePerGB;

    // 数据中心碳排放 - 考虑可再生能源比例
    const dataTransferCarbon = dataCenterEnergy * (
      (renewable / 100) * GLOBAL_CONSTANTS.greenEnergyCarbonIntensity +
      ((100 - renewable) / 100) * countryCarbonValue
    );

    // 网络传输和客户设备碳排放
    const networkCarbon = transmissionEnergy * countryCarbonValue;
    const clientCarbon = deviceEnergy * countryCarbonValue;

    // 计算总碳排放量（单位：g CO2e）
    const totalCarbonEmission = dataTransferCarbon + networkCarbon + clientCarbon;

    // 估计月访问量 - 基于固定值，但明确标记这不是测量值
    const monthlyCarbonEmission = (totalCarbonEmission * GLOBAL_CONSTANTS.averageMonthlyVisits) / 1000; // 单位：kg CO2e
    const annualCarbonEmission = monthlyCarbonEmission * 12;

    // 计算碳中和所需的树木数量
    const treesNeeded = Math.ceil(annualCarbonEmission / GLOBAL_CONSTANTS.treeCO2PerYear);

    // 碳足迹和能源效率评分 - 基于真实测量值计算
    const carbonFootprintScore = Math.min(100, Math.max(1,
      (totalCarbonEmission / 0.5) * 20 + // 基础碳排放
      (dataCenterPUE / GLOBAL_CONSTANTS.bestPUE - 1) * 20 + // 数据中心效率
      ((100 - renewable) / 100) * 40 // 可再生能源使用
    ));

    const energyEfficiencyScore = Math.min(100, Math.max(1,
      100 - (totalCarbonEmission / 3) * 20 - // 基础能源效率
      (actualDomainCount - 1) * 3 // 域名数量惩罚（实际测量）
    ));

    res.json({
      measurable: true,
      pageSize: pageSizeKB,
      energyIntensity,
      cachingEfficiency: measuredCacheEfficiency,
      compressionEfficiency: isCompressed ? 0.3 : 0, // 压缩节省百分比（实际测量）
      dataCenterEnergy,
      transmissionEnergy,
```

```
      deviceEnergy,
      dataTransferCarbon,
      networkCarbon,
      clientCarbon,
      totalCarbonEmission,
      monthlyCarbonEmission,
      annualCarbonEmission,
      treesNeeded,
      isGreen: renewable ≥ GLOBAL_CONSTANTS.greenEnergyThreshold,
      carbonFootprintScore,
      energyEfficiencyScore,
      resourceBreakdown: resourceStatsData,
      requestCount: actualRequestCount,
      domainCount: actualDomainCount,
      dataSourceInfo: {
        allRealMeasurements: true,
        estimatedValues: ['monthlyCarbonEmission', 'annualCarbonEmission', 'treesNeeded'],
        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。'
      }
    });
  } catch (error) {
    console.error('碳排放计算错误:', error);
    res.status(500).json({
      error: '碳排放计算失败',
      message: error.message,
      measurable: false
    });
  }
});

/**
 * 生成优化建议
 */
app.get('/api/suggestions', async (req, res) ⇒ {
  try {
    const {
      pageSize,
      performance,
      provider,
      renewablePercentage,
      pue,
      totalCarbonEmission,
      carbonFootprintScore,
      energyEfficiencyScore,
      treesNeeded,
      requestCount,
      domainCount
    } = req.query;

    // 提取性能数据
    const performanceData = typeof performance === 'string' ? JSON.parse(performance) : (performance || {});

```

```
    // 准备建议生成的数据
    const data = {
      pageSize: parseInt(pageSize) || 0,
      performance: {
        fcp: parseFloat(performanceData.fcp) || 1.5,
        lcp: parseFloat(performanceData.lcp) || 2.5,
        cls: parseFloat(performanceData.cls) || 0.05,
        fid: parseFloat(performanceData.fid) || 100,
        ttfb: parseFloat(performanceData.ttfb) || 200
      },
      provider: provider || 'other',
      renewablePercentage: parseFloat(renewablePercentage) || 50,
      pue: parseFloat(pue) || 1.67,
      totalCarbonEmission: parseFloat(totalCarbonEmission) || 0,
      carbonFootprintScore: parseFloat(carbonFootprintScore) || 0,
      energyEfficiencyScore: parseFloat(energyEfficiencyScore) || 0,
      treesNeeded: parseInt(treesNeeded) || 0,
      requestCount: parseInt(requestCount) || 1,
      domainCount: parseInt(domainCount) || 1
    };

    // 生成建议
    const suggestions = generateOptimizationSuggestions(data);

    res.json({ suggestions });
  } catch (error) {
    console.error('建议生成错误:', error);
    res.status(500).json({ error: '建议生成失败', message: error.message });
  }
});

/**
 * 综合网站分析
 */
app.get('/api/analyze', async (req, res) => {
  try {
    const { domain, browser = 'auto' } = req.query;

    if (!domain) {
      return res.status(400).json({ error: '缺少域名参数' });
    }

    // 确保域名格式正确
    let url;
    try {
      if (domain.startsWith('http')) {
        url = new URL(domain).href;
      } else {
        url = `https://${domain}`;
      }
    } catch (error) {
      return res.status(400).json({ error: '无效的域名格式' });
    }

    // 并行执行所有分析任务
```

```
    let [locationData, headerData, sizeData, providerData] = [null, null, null, null];
    let performanceData = null;

    try {
      [locationData, headerData, providerData] = await Promise.all([
        axios.get(`http://localhost:${PORT}/api/location?domain=${domain}`).then(resp => resp.data),
        axios.get(`http://localhost:${PORT}/api/headers?url=${encodeURIComponent(url)}`).then(resp => resp.data),
        axios.get(`http://localhost:${PORT}/api/provider?domain=${domain}`).then(resp => resp.data)
      ]);
    } catch (error) {
      console.error('基础数据获取错误:', error);
    }

    // 单独执行性能测量，因为它可能需要更长时间
    try {
      performanceData = await axios.get(`http://localhost:${PORT}/api/performance?url=${encodeURIComponent(url)}&browser=${browser}`)
        .then(resp => resp.data);

      // 如果有性能数据，使用它的页面大小
      if (performanceData && performanceData.pageSize) {
        sizeData = { size: performanceData.pageSize };
      }
    } catch (error) {
      console.error('性能测量错误:', error);
      performanceData = { measurable: false, error: error.message };
    }

    // 如果没有页面大小数据，尝试单独获取
    if (!sizeData) {
      try {
        sizeData = await axios.get(`http://localhost:${PORT}/api/size?url=${encodeURIComponent(url)}`)
          .then(resp => resp.data);
      } catch (error) {
        console.error('页面大小测量错误:', error);
        sizeData = { measurable: false, error: error.message };
      }
    }

    // 尝试计算碳排放（即使部分数据缺失）
    let carbonData = null;
    try {
      const pageSize = sizeData?.size || 0;
      const country = locationData?.country || null;
      const renewablePercentage = providerData?.renewablePercentage || 50;
      const pue = providerData?.pue || GLOBAL_CONSTANTS.averagePUE;

      // 获取性能数据中的请求数和域名数
      const requestCount = performanceData?.requestCount || 1;
```

```
      const domainCount = performanceData?.domainCount || 1;

      if (pageSize > 0) {
        carbonData = await axios.get(`http://localhost:${PORT}/api/carbon`, {
          params: {
            pageSize,
            country,
            renewablePercentage,
            pue,
            requestCount,
            domainCount
          }
        }).then(resp => resp.data);
      } else {
        carbonData = { measurable: false, error: '缺少页面大小数据' };
      }
    } catch (error) {
      console.error('碳排放计算错误:', error);
      carbonData = { measurable: false, error: error.message };
    }

    // 只有当性能数据可用时才生成优化建议
    let suggestionsData = { suggestions: [] };
    if (performanceData && !performanceData.error) {
      try {
        suggestionsData = await axios.get(`http://localhost:${PORT}/api/suggestions`, {
          params: {
            pageSize: sizeData?.size || 0,
            performance: JSON.stringify(performanceData),
            provider: providerData?.provider || 'other',
            renewablePercentage: providerData?.renewablePercentage || 50,
            pue: providerData?.pue || GLOBAL_CONSTANTS.averagePUE,
            totalCarbonEmission: carbonData?.totalCarbonEmission || 0,
            carbonFootprintScore: carbonData?.carbonFootprintScore || 0,
            energyEfficiencyScore: carbonData?.energyEfficiencyScore || 0,
            treesNeeded: carbonData?.treesNeeded || 0,
            requestCount: performanceData?.requestCount || 1,
            domainCount: performanceData?.domainCount || 1
          }
        }).then(resp => resp.data);
      } catch (error) {
        console.error('建议生成错误:', error);
        suggestionsData = {
          suggestions: ['由于性能数据获取失败，无法生成针对性优化建议。'],
          measurable: false,
          error: error.message
        };
      }
    } else {
      suggestionsData = {
        suggestions: ['由于性能数据获取失败，无法生成针对性优化建议。'],
        measurable: false
      };
```

```
    }

    // 整合所有数据
    const result = {
      domain,
      url,
      browser,
      timestamp: new Date().toISOString(),

      // 位置信息
      location: locationData || { measurable: false },

      // 页面信息
      pageSize: sizeData?.size || 0,
      contentType: sizeData?.contentType || null,
      headers: headerData?.headers || null,

      // 服务提供商信息
      provider: providerData?.provider || 'unknown',
      providerName: providerData?.providerName || 'Unknown Provider',
      renewablePercentage: providerData?.renewablePercentage || null,
      pue: providerData?.pue || null,

      // 性能指标
      performance: performanceData || { measurable: false },

      // 碳排放信息
      energyIntensity: carbonData?.energyIntensity || null,
      dataCenterEnergy: carbonData?.dataCenterEnergy || null,
      transmissionEnergy: carbonData?.transmissionEnergy || null,
      deviceEnergy: carbonData?.deviceEnergy || null,
      dataTransferCarbon: carbonData?.dataTransferCarbon || null,
      networkCarbon: carbonData?.networkCarbon || null,
      clientCarbon: carbonData?.clientCarbon || null,
      totalCarbonEmission: carbonData?.totalCarbonEmission || null,
      monthlyCarbonEmission: carbonData?.monthlyCarbonEmission || null,
      annualCarbonEmission: carbonData?.annualCarbonEmission || null,
      treesNeeded: carbonData?.treesNeeded || null,
      isGreen: carbonData?.isGreen || false,

      // 优化建议
      suggestions: suggestionsData.suggestions || []
    };

    res.json(result);
  } catch (error) {
    console.error('综合分析错误:', error);
    res.status(500).json({
      error: '综合分析失败',
      message: error.message,
      stack: process.env.NODE_ENV === 'development' ? error.stack :
undefined
    });
  }
});
```

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

```
  }

  if (data.totalCarbonEmission > 1.5) {
    suggestions.push(`当前页面单次访问碳排放(${safeToFixed(data.totalCarbonEmission, 2)}gCO2e)偏高，建议全面优化页面资源`);
    if (data.totalCarbonEmission > 3) {
      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资源加载');
    }
  }

  // 加入碳中和相关建议
  if (data.treesNeeded > 5) {
    suggestions.push(`抵消网站年度碳排放需要种植约${data.treesNeeded}棵树，建议考虑参与碳抵消项目或减少能源消耗`);
  }

  // 通用绿色建议
  if (suggestions.length < 3) {
    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');
    suggestions.push('使用CDN分发静态资源，减少数据传输距离和能耗');
    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');
  }

  // 确保建议数量不会太多
  return suggestions.slice(0, 10);
}

/**
 * 检查系统环境
 */
function checkEnvironment() {
  console.log('=== 系统环境信息 ===');
  console.log(`操作系统: ${os.platform()} ${os.release()}`);
  console.log(`Node.js 版本: ${process.version}`);
  console.log(`CPU 架构: ${os.arch()}`);
  console.log(`可用内存: ${Math.round(os.freemem() / 1024 / 1024)} MB / ${Math.round(os.totalmem() / 1024 / 1024)} MB`);
  console.log('=== 环境信息结束 ===');
}

/**
 * 基础版网页分析 - 不依赖浏览器，使用纯HTTP请求
 * @param {string} url - 目标URL
 * @returns {Promise<Object>} 基础性能指标
 */
async function measurePerformanceBasic(url) {
  try {
    const startTime = Date.now();

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

```
        supportsCaching,
        securityHeaders,
        serverType,
        supportsHTTPS
      },
      measurementMethod: 'basic-http'
    };
  } catch (error) {
    console.error('基础性能测量错误:', error);
    return {
      success: false,
      error: `基础性能测量失败: ${error.message}`
    };
  }
}

/**
 * 分析HTML内容质量
 * @param {CheerioStatic} $ - Cheerio对象
 * @returns {Object} 内容质量分析结果
 */
function analyzeContentQuality($) {
  if (!$) return { score: 0 };

  try {
    // 计算文本内容量
    const bodyText = $('body').text().trim();
    const textLength = bodyText.length;

    // 计算图像数量和是否有alt属性
    const images = $('img');
    const imageCount = images.length;
    let imagesWithAlt = 0;

    images.each((i, img) => {
      if ($(img).attr('alt')) imagesWithAlt++;
    });

    // 检查标题层次结构
    const h1Count = $('h1').length;
    const h2Count = $('h2').length;
    const h3Count = $('h3').length;

    // 检查链接质量
    const links = $('a');
    const linkCount = links.length;
    let linksWithText = 0;

    links.each((i, link) => {
      if ($(link).text().trim().length > 0) linksWithText++;
    });

    // 检查元数据
    const hasTitle = $('title').length > 0;
    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  /**
```

```
 * 估计资源大小
 * @param {Object} resourceStats - 资源统计
 * @param {number} pageSize - HTML页面大小
 * @returns {number} 估计的总资源大小（字节）
 */
function estimateResourceSize(resourceStats, pageSize) {
  // 不同资源类型的平均大小估计（字节）
  const averageSizes = {
    css: 20000,  // 平均CSS文件约20KB
    js: 80000,   // 平均JS文件约80KB
    images: 200000, // 平均图片约200KB
    fonts: 30000,   // 平均字体约30KB
    other: 10000    // 其他资源约10KB
  };

  let totalSize = pageSize || 0; // HTML大小

  // 计算各类资源的估计总大小
  for (const [type, count] of Object.entries(resourceStats)) {
    if (averageSizes[type]) {
      totalSize += count * averageSizes[type];
    }
  }

  return totalSize;
}

/**
 * HTTP头部分析 - 检查与性能相关的HTTP头
 * @param {string} url - 目标URL
 * @returns {Promise<Object>} 基于头部的指标
 */
async function analyzeHttpHeaders(url) {
  console.log('分析HTTP头部信息...');
  try {
    const response = await axiosInstance.head(url, {
      maxRedirects: 5,
      validateStatus: null
    });

    const headers = response.headers;

    // 检查性能相关的HTTP头
    const hasCompression = headers['content-encoding'] &&
      (headers['content-encoding'].includes('gzip') ||
       headers['content-encoding'].includes('br') ||
       headers['content-encoding'].includes('deflate'));

    const hasCaching = headers['cache-control'] || headers['expires'];

    const securityHeaders = [
      'strict-transport-security',
      'content-security-policy',
      'x-content-type-options',
      'x-frame-options',
```

```
      'x-xss-protection'
    ];

    // 检测存在的安全头
    const presentSecurityHeaders = [];
    securityHeaders.forEach(header => {
      if (headers[header]) {
        presentSecurityHeaders.push(header);
      }
    });

    const securityScore = (presentSecurityHeaders.length / securityHeaders.length) * 100;

    // 分析CDN使用情况
    let usingCDN = false;
    let cdnProvider = null;

    // 常见CDN标识头
    if (headers['server'] &&
        (headers['server'].includes('cloudflare') ||
         headers['server'].includes('akamai') ||
         headers['server'].includes('fastly'))) {
      usingCDN = true;
      cdnProvider = headers['server'].split(' ')[0];
    }

    // 检查CDN特有头
    if (headers['cf-ray'] || headers['cf-cache-status']) {
      usingCDN = true;
      cdnProvider = 'Cloudflare';
    } else if (headers['x-cache'] && headers['x-served-by']) {
      usingCDN = true;
      cdnProvider = 'Fastly/Varnish';
    } else if (headers['x-amz-cf-id']) {
      usingCDN = true;
      cdnProvider = 'Amazon CloudFront';
    } else if (headers['x-azure-ref']) {
      usingCDN = true;
      cdnProvider = 'Azure CDN';
    }

    return {
      hasCompression,
      hasCaching,
      usingCDN,
      cdnProvider,
      securityScore,
      securityHeaders: presentSecurityHeaders,
      server: headers['server'] || 'unknown',
      headers: headers
    };
  } catch (error) {
    console.error('HTTP头分析失败:', error);
    throw error;
```

```
  }
}

// 所有其他路由返回前端应用
const serveIndex = (req, res) => {
  res.sendFile(path.join(__dirname, '../dist/index.html'));
};

app.get('*', serveIndex);

// 启动服务器
app.listen(PORT, async () => {
  console.log(`GreenWeb服务器运行在端口 ${PORT}`);
  console.log(`访问 http://localhost:${PORT} 以使用应用`);
});

/**
 * 从HTML中提取资源URL
 * @param {string} html - HTML内容
 * @param {string} baseUrl - 基础URL
 * @returns {Object} 提取的资源URL和分析结果
 */
function extractResourceUrls(html, baseUrl) {
  try {
    const cheerio = require('cheerio');
    const $ = cheerio.load(html);
    const resourceUrls = [];
    const uniqueDomains = new Set();

    // 获取URL的域名部分
    const getUrlDomain = (url) => {
      try {
        if (!url || typeof url !== 'string') return '';
        const urlObj = new URL(url, baseUrl);
        return urlObj.hostname;
      } catch (e) {
        return '';
      }
    };

    // 提取域名
    const baseDomain = getUrlDomain(baseUrl);

    // 处理URL
    const processUrl = (url) => {
      if (!url) return null;
      try {
        // 处理相对URL
        const absoluteUrl = new URL(url, baseUrl).href;
        const domain = getUrlDomain(absoluteUrl);
        if (domain) {
          uniqueDomains.add(domain);
        }
        return absoluteUrl;
      } catch (e) {
```

```
        return null;
      }
    };

    // 提取CSS链接
    $('link[rel="stylesheet"]').each((_, el) => {
      const url = processUrl($(el).attr('href'));
      if (url) resourceUrls.push({ url, type: 'css' });
    });

    // 提取JavaScript
    $('script[src]').each((_, el) => {
      const url = processUrl($(el).attr('src'));
      if (url) resourceUrls.push({ url, type: 'js' });
    });

    // 提取图片
    $('img[src]').each((_, el) => {
      const url = processUrl($(el).attr('src'));
      if (url) resourceUrls.push({ url, type: 'images' });
    });

    // 提取背景图像从内联样式
    $('[style*="background"]').each((_, el) => {
      const style = $(el).attr('style');
      if (style) {
        const match = style.match(/url\(['"]?([^'")]+)['"]?\)/);
        if (match && match[1]) {
          const url = processUrl(match[1]);
          if (url) resourceUrls.push({ url, type: 'images' });
        }
      }
    });

    // 提取字体
    $('link[rel="preload"][as="font"]').each((_, el) => {
      const url = processUrl($(el).attr('href'));
      if (url) resourceUrls.push({ url, type: 'fonts' });
    });

    return {
      $,
      resourceUrls,
      uniqueDomains
    };
  } catch (error) {
    console.error('提取资源URL错误:', error);
    return {
      $: null,
      resourceUrls: [],
      uniqueDomains: new Set()
    };
  }
}

```

```
/**
 * 统计不同类型的资源
 * @param {Array} resourceUrls - 资源URL数组
 * @returns {Object} 资源类型统计
 */
function countResourceTypes(resourceUrls) {
  const stats = {
    css: 0,
    js: 0,
    images: 0,
    fonts: 0,
    other: 0
  };

  resourceUrls.forEach(resource => {
    if (stats[resource.type] !== undefined) {
      stats[resource.type]++;
    } else {
      stats.other++;
    }
  });

  return stats;
}

/**
 * 检查CDN使用情况
 * @param {Set} uniqueDomains - 唯一域名集合
 * @param {Object} headers - HTTP响应头
 * @returns {Object} CDN使用信息
 */
function checkCdnUsage(uniqueDomains, headers) {
  const cdnProviders = {
    'cloudflare': ['cloudflare', 'cloudflare-nginx', 'cloudfront.net'],
    'akamai': ['akamai', 'akamaiedge.net', 'akamaized.net'],
    'fastly': ['fastly'],
    'cloudfront': ['cloudfront.net'],
    'vercel': ['vercel-edge', 'vercel.app'],
    'netlify': ['netlify', 'netlify.app']
  };

  // 检查响应头中是否包含CDN信息
  let cdnProvider = 'unknown';
  let usesCdn = false;

  // 检查常见CDN响应头
  if (headers['server']) {
    for (const [provider, keywords] of Object.entries(cdnProviders)) {
      if (keywords.some(keyword =>
headers['server'].toLowerCase().includes(keyword))) {
        cdnProvider = provider;
        usesCdn = true;
        break;
      }
    }
```

```
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' },
```

```
    'x-frame-options': { score: 10, name: '框架选项' },
    'x-xss-protection': { score: 10, name: 'XSS保护' },
    'referrer-policy': { score: 10, name: '引用策略' },
    'permissions-policy': { score: 10, name: '权限策略' },
    'feature-policy': { score: 10, name: '功能策略' }
  };

  for (const [header, info] of Object.entries(securityHeaders)) {
    if (headers[header]) {
      score += info.score;
      details[header] = {
        present: true,
        value: headers[header],
        name: info.name
      };
    } else {
      details[header] = {
        present: false,
        name: info.name
      };
    }
  }

  return {
    score: Math.min(93, score),
    details
  };
}

// 添加安全的toFixed函数，避免对null/undefined调用toFixed
function safeToFixed(value, digits = 2) {
  if (value === null || value === undefined) return '0.00';
  return Number(value).toFixed(digits);
}
```

```javascript
/**
 * 网站分析服务
 * 用于向目标网站发送请求并获取性能指标和位置信息
 */

import axios from 'axios';

// 获取当前环境的基础URL
function getBaseUrl() {
  // 在生产环境中，API和前端在同一域下运行
  // 在开发环境中，API在localhost:3000上运行
  const isProd = import.meta.env.PROD;
  return isProd ? '/api' : 'http://localhost:3000/api';
}

// 代理服务器URL
const API_URL = getBaseUrl();

/**
 * 解析URL获取完整域名
 * @param {string} url - 输入的URL或域名
 * @returns {string} 处理后的完整URL
 */
function parseUrl(url) {
  if (!url) return '';

  // 如果没有协议，添加https://
  if (!url.startsWith('http://') && !url.startsWith('https://')) {
    url = 'https://' + url;
  }

  try {
    const parsedUrl = new URL(url);
    return parsedUrl.href;
  } catch (error) {
    console.error('URL解析错误:', error);
    return '';
  }
}

/**
 * 获取网站服务器位置
 * @param {string} domain - 目标域名
 * @returns {Promise<Object>} 服务器位置信息
 */
export async function getServerLocation(domain) {
  try {
    const response = await axios.get(`${API_URL}/location`, {
      params: { domain }
    });
    return response.data;
  } catch (error) {
    console.error('获取服务器位置失败:', error);
```

```
    // 失败时返回错误信息
    return {
      error: '无法获取服务器位置信息',
      details: error.message,
      measurable: false
    };
  }
}

/**
 * 获取网站HTTP响应头信息
 * @param {string} url - 目标URL
 * @returns {Promise<Object>} HTTP响应头信息
 */
export async function getHttpHeaders(url) {
  try {
    const fullUrl = parseUrl(url);
    const response = await axios.get(`${API_URL}/headers`, {
      params: { url: fullUrl }
    });
    return response.data;
  } catch (error) {
    console.error('获取HTTP头信息失败:', error);
    return {
      error: '无法获取HTTP头信息',
      details: error.message,
      measurable: false
    };
  }
}

/**
 * 测量网站页面大小
 * @param {string} url - 目标URL
 * @returns {Promise<number>} 页面大小（KB）
 */
export async function measurePageSize(url) {
  try {
    const fullUrl = parseUrl(url);
    const response = await axios.get(`${API_URL}/size`, {
      params: { url: fullUrl }
    });
    return response.data;
  } catch (error) {
    console.error('测量页面大小失败:', error);
    // 失败时返回错误信息
    return {
      error: '无法测量页面大小',
      details: error.message,
      measurable: false
    };
  }
}

/**
```

```
 * 测量网站性能指标
 * @param {string} url - 目标URL
 * @param {string} [browser='auto'] - 使用的浏览器，可选值：'auto'(自动)、'chrome'、'edge'
 * @returns {Promise<Object>} 性能指标
 */
export async function measurePerformance(url, browser = 'auto') {
  let retries = 0;
  const maxRetries = 2;
  const retryDelay = 3000; // 重试间隔3秒

  const attemptMeasure = async () => {
    try {
      const fullUrl = parseUrl(url);
      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}`);

      const response = await axios.get(`${API_URL}/performance`, {
        params: {
          url: fullUrl,
          browser: browser // 将浏览器参数传递给后端
        },
        timeout: 120000 // 增加超时时间到120秒，因为Lighthouse分析可能需要较长时间
      });

      console.log(`性能指标测量成功 (使用${response.data.measuredBy || '未知方法'}):`, response.data);
      return response.data;
    } catch (error) {
      if (axios.isAxiosError(error)) {
        console.error(`性能测量失败 (尝试 ${retries + 1}/${maxRetries + 1}):`, error.message);

        if (error.response) {
          // 服务器返回了错误状态码
          console.error('服务器错误:', error.response.status, error.response.data);
          return {
            error: `性能测量失败: ${error.response.data.message || '服务器错误'}`,
            details: error.response.data,
            measurable: false
          };
        } else if (error.request) {
          // 请求已经发出，但没有收到响应
          console.error('未收到响应:', error.request);
          if (retries < maxRetries) {
            retries++;
            console.log(`等待 ${retryDelay}ms 后重试...`);
            await new Promise(resolve => setTimeout(resolve, retryDelay));
            return attemptMeasure();
          }
          return {
            error: '性能测量超时，请检查网络连接或网站可访问性',
            details: '请求已发出但未收到响应',
```

```
            measurable: false
          };
        } else {
          // 设置请求时发生了错误
          if (retries < maxRetries) {
            retries++;
            console.log(`等待 ${retryDelay}ms 后重试...`);
            await new Promise(resolve => setTimeout(resolve, retryDelay));
            return attemptMeasure();
          }
          return {
            error: '无法测量性能指标',
            details: error.message,
            measurable: false
          };
        }
      } else {
        // 非Axios错误
        console.error('性能测量过程中发生非网络错误:', error);
        return {
          error: '无法测量性能指标',
          details: error.message,
          measurable: false
        };
      }
    }
  };

  return attemptMeasure();
}

/**
 * 分析网站服务器提供商
 * @param {string} domain - 目标域名
 * @returns {Promise<Object>} 服务提供商信息
 */
export async function analyzeProvider(domain) {
  try {
    const response = await axios.get(`${API_URL}/provider`, {
      params: { domain }
    });
    return response.data;
  } catch (error) {
    console.error('分析服务提供商失败:', error);
    return {
      error: '无法分析服务提供商',
      details: error.message,
      measurable: false
    };
  }
}

/**
 * 综合分析网站（使用后端的单一接口）
 * @param {string} domain - 目标域名
```

```
 * @param {string} [browser='auto'] - 使用的浏览器，可选值：'auto'(自动)、'chrome'、'edge'
 * @returns {Promise<Object>} 分析结果
 */
export async function analyzeWebsite(domain, browser = 'auto') {
  try {
    // 使用后端的综合分析端点
    const response = await axios.get(`${API_URL}/analyze`, {
      params: {
        domain,
        browser // 添加浏览器参数
      },
      timeout: 120000 // 增加超时时间到120秒
    });

    return response.data;
  } catch (error) {
    console.error('网站分析失败:', error);
    return {
      error: '网站分析失败',
      details: error.message,
      measurable: false
    };
  }
}

/**
 * 单独调用，不使用综合接口（用于测试和调试）
 * @param {string} domain - 目标域名
 * @param {string} [browser='auto'] - 使用的浏览器，可选值：'auto'(自动)、'chrome'、'edge'
 * @returns {Promise<Object>} 分析结果
 */
export async function analyzeWebsiteSeparately(domain, browser = 'auto') {
  try {
    // 准备完整URL
    const url = parseUrl(domain);
    if (!url) throw new Error('无效的域名');

    // 并行执行所有请求以提高性能
    const [locationData, headersData, sizeData, performanceData, providerData] = await Promise.all([
      getServerLocation(domain),
      getHttpHeaders(url),
      measurePageSize(url),
      measurePerformance(url, browser), // 传递浏览器参数
      analyzeProvider(domain)
    ]);

    // 检查是否有任何请求失败
    if (locationData.error || headersData.error || sizeData.error ||
        performanceData.error || providerData.error) {
      const errors = [];
      if (locationData.error) errors.push(locationData.error);
      if (headersData.error) errors.push(headersData.error);
```

```
      if (sizeData.error) errors.push(sizeData.error);
      if (performanceData.error) errors.push(performanceData.error);
      if (providerData.error) errors.push(providerData.error);

      return {
        url,
        domain,
        browser, // 添加使用的浏览器信息
        error: '部分数据获取失败',
        details: errors.join('; '),
        measurable: false,
        location: locationData,
        provider: providerData.provider || 'unknown',
        pageSize: sizeData.size,
        performance: performanceData,
        headers: headersData.headers
      };
    }

    return {
      url,
      domain,
      browser, // 添加使用的浏览器信息
      provider: providerData.provider,
      location: locationData,
      pageSize: sizeData.size,
      performance: performanceData,
      headers: headersData.headers
    };
  } catch (error) {
    console.error('网站分析失败:', error);
    return {
      error: '网站分析失败',
      details: error.message,
      measurable: false
    };
  }
}

/**
 * 分析网站碳排放
 * @param {Object} params - 碳排放分析参数
 * @param {number} params.pageSize - 页面大小(KB)
 * @param {string} params.country - 服务器所在国家代码
 * @param {number} params.requestCount - 请求数量
 * @param {number} params.domainCount - 域名数量
 * @param {number} [params.renewablePercentage] - 可再生能源使用百分比
 * @param {number} [params.pue] - 数据中心PUE
 * @returns {Promise<Object>} 碳排放分析结果
 */
export async function analyzeCarbonEmission(params) {
  try {
    const response = await axios.get(`${API_URL}/carbon`, { params });
    return response.data;
  } catch (error) {
```

```
    console.error('碳排放分析失败:', error);
    return {
      error: '无法分析碳排放',
      details: error.message,
      measurable: false,
      totalCarbonEmission: 0,
      monthlyCarbonEmission: 0,
      annualCarbonEmission: 0,
      isGreen: false
    };
  }
}

export default {
  analyzeWebsite,
  analyzeWebsiteSeparately,
  getServerLocation,
  measurePageSize,
  measurePerformance,
  analyzeProvider,
  getHttpHeaders,
  analyzeCarbonEmission
};
```