

scripts/ 디렉토리 유틸리티들

Pure-Flon 웹사이트 개발 및 관리를 위한 유용한 스크립트들입니다.

디렉토리 구조

```
scripts/  
├── optimize-images.js  # 이미지 최적화  
├── seo-check.js        # SEO 체크  
├── dev-server.js       # 로컬 개발 서버  
├── performance-audit.js # 성능 감사  
├── generate-sitemap.js # 사이트맵 생성  
├── check-links.js      # 링크 검사  
└── deploy-check.js     # 배포 전 검사
```

scripts/optimize-images.js

javascript

```
#!/usr/bin/env node

/**
 * Pure-Flon Website - Image Optimization Script
 * 이미지 파일들을 자동으로 최적화하여 웹 성능을 향상시킵니다.
 */

const fs = require('fs');
const path = require('path');
const sharp = require('sharp');
const glob = require('glob');

const IMAGE_EXTENSIONS = ['.jpg', '.jpeg', '.png', '.webp'];
const OUTPUT_FORMATS = ['webp', 'jpeg'];
const QUALITY_SETTINGS = {
  webp: 85,
  jpeg: 85,
  png: 90
};

class ImageOptimizer {
  constructor() {
    this.processedCount = 0;
    this.savedBytes = 0;
  }

  async optimizeImages() {
    console.log('🖼️ Pure-Flon 이미지 최적화 시작...\n');

    const imageFiles = glob.sync('images/**/*', {
      nodir: true
    }).filter(file =>
      IMAGE_EXTENSIONS.includes(path.extname(file).toLowerCase())
    );
  }
}
```

```
);

if (imageFiles.length === 0) {
  console.log('❌ 최적화할 이미지가 없습니다.');
```

```
  return;
}

console.log('📁 발견된 이미지: ${imageFiles.length}개\n');
```

```
for (const imagePath of imageFiles) {
  await this.processImage(imagePath);
}

this.printSummary();
}

async processImage(imagePath) {
  try {
    const originalStats = fs.statSync(imagePath);
    const originalSize = originalStats.size;

    console.log('📷 처리 중: ${imagePath}');
```

```
    const image = sharp(imagePath);
    const metadata = await image.metadata();

    // 크기 조정 (최대 너비 제한)
    const maxWidth = this.getMaxWidth(imagePath);
    if (metadata.width > maxWidth) {
      image.resize({ width: maxWidth });
    }
  }
}
```

// 포맷별 최적화

```
for (const format of OUTPUT_FORMATS) {  
  const outputPath = this.getOutputPath(imagePath, format);
```

```
  await image  
    .clone()  
    [format]({ quality: QUALITY_SETTINGS[format] })  
    .toFile(outputPath);  
}
```


// WebP 버전 생성

```
const webpPath = this.getOutputPath(imagePath, 'webp');  
await image  
  .webp({ quality: QUALITY_SETTINGS.webp })  
  .toFile(webpPath);
```

```
const optimizedStats = fs.statSync(webpPath);  
const savedBytes = originalSize - optimizedStats.size;
```

```
this.processedCount++;  
this.savedBytes += savedBytes;
```

```
console.log(`  완료: ${this.formatBytes(savedBytes)} 절약`);
```

```
} catch (error) {  
  console.log(`  오류: ${error.message}`);  
}  
}
```

```
getMaxWidth(imagePath) {  
  if (imagePath.includes('hero')) return 1920;  
  if (imagePath.includes('product')) return 800;
```

```

    if (imagePath.includes('icon')) return 512;
    if (imagePath.includes('thumb')) return 300;
    return 1200; // 기본값
}

```

```

getOutputPath(originalPath, format) {
    const parsedPath = path.parse(originalPath);
    return path.join(
        parsedPath.dir,
        `${parsedPath.name}-optimized.${format}`
    );
}

```

```

formatBytes(bytes) {
    if (bytes === 0) return '0 Bytes';
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];
}

```

```

printSummary() {
    console.log(`\n 🍌 이미지 최적화 완료!`);
    console.log(` 📊 처리된 이미지: ${this.processedCount}개`);
    console.log(` 🗄️ 총 절약 용량: ${this.formatBytes(this.savedBytes)}`);
    console.log(`\n 💡 WebP 이미지를 사용하려면 HTML에서 <picture> 태그를 사용하세요:`);
    console.log(`
<picture>
  <source srcset="image-optimized.webp" type="image/webp">
  
</picture>
`);
}

```

```
}  
}  
  
// 스크립트 실행  
if (require.main === module) {  
  const optimizer = new ImageOptimizer();  
  optimizer.optimizeImages().catch(console.error);  
}  
  
module.exports = ImageOptimizer;
```

scripts/seo-check.js

javascript

```
#!/usr/bin/env node
```

```
/**
```

```
 * Pure-Flon Website - SEO Audit Script
```

```
 * 웹사이트의 SEO 최적화 상태를 체크합니다.
```

```
 */
```

```
const fs = require('fs');
```

```
const path = require('path');
```

```
const cheerio = require('cheerio');
```

```
const glob = require('glob');
```

```
class SEOAuditor {
```

```
  constructor() {
```

```
    this.issues = [];
```

```
    this.successes = [];
```

```
  }
```

```
  async auditSEO() {
```

```
    console.log('🔍 Pure-Flon SEO 감사 시작...\n');
```

```
    const htmlFiles = glob.sync('**/*.html', {
```

```
      ignore: ['node_modules/**', 'dist/**']
```

```
    });
```

```
    for (const file of htmlFiles) {
```

```
      await this.auditFile(file);
```

```
    }
```

```
    this.printReport();
```

```
  }
```

```
  async auditFile(filePath) {
```

```
console.log('📄 검사 중: ${filePath}');
```

```
const content = fs.readFileSync(filePath, 'utf8');  
const $ = cheerio.load(content);
```

```
this.checkTitle($, filePath);  
this.checkMetaDescription($, filePath);  
this.checkHeadings($, filePath);  
this.checkImages($, filePath);  
this.checkLinks($, filePath);  
this.checkStructuredData($, filePath);  
this.checkOpenGraph($, filePath);
```

```
console.log('');  
}
```

```
checkTitle($, file) {  
  const title = $('title').text();  
  
  if (!title) {  
    this.addIssue(file, 'TITLE', '❌ <title> 태그가 없습니다.');  } else if (title.length < 30) {  
    this.addIssue(file, 'TITLE', `⚠️ 제목이 너무 짧습니다 (${title.length}자): ${title}`);  
  } else if (title.length > 60) {  
    this.addIssue(file, 'TITLE', `⚠️ 제목이 너무 깁니다 (${title.length}자): ${title}`);  
  } else {  
    this.addSuccess(file, 'TITLE', `✅ 제목 길이 적절 (${title.length}자)`);  
  }  
}
```

```
checkMetaDescription($, file) {  
  const description = $('meta[name="description"]').attr('content');
```



```
if (!description) {
  this.addIssue(file, 'META_DESC', '❌ meta description이 없습니다.');
```

} else if (description.length < 120) {

```
  this.addIssue(file, 'META_DESC', `⚠️ 설명이 너무 짧습니다 (${description.length}자)`);
} else if (description.length > 160) {
  this.addIssue(file, 'META_DESC', `⚠️ 설명이 너무 길니다 (${description.length}자)`);
} else {
  this.addSuccess(file, 'META_DESC', `✅ 설명 길이 적절 (${description.length}자)`);
}
}
```

```
checkHeadings($, file) {
  const h1Count = $('h1').length;

  if (h1Count === 0) {
    this.addIssue(file, 'HEADINGS', '❌ H1 태그가 없습니다.');
```

} else if (h1Count > 1) {

```
  this.addIssue(file, 'HEADINGS', `⚠️ H1 태그가 여러 개입니다 (${h1Count}개)`);
} else {
  this.addSuccess(file, 'HEADINGS', '✅ H1 태그 개수 적절');
```

}

// 헤딩 계층 구조 검사

```
const headings = $('h1, h2, h3, h4, h5, h6').map((i, el) => {
  return parseInt($(el).prop('tagName').replace('H', ''));
}).get();
```

```
for (let i = 1; i < headings.length; i++) {
  if (headings[i] - headings[i-1] > 1) {
    this.addIssue(file, 'HEADINGS', `⚠️ 헤딩 계층 구조가 올바르지 않습니다.`);
    break;
  }
}
```

```
}  
}  
}
```

```
checkImages($, file) {  
  const imagesWithoutAlt = $('img:not([alt])').length;  
  const totalImages = $('img').length;  
  
  if (imagesWithoutAlt > 0) {  
    this.addIssue(file, 'IMAGES', `❌ alt 속성이 없는 이미지: ${imagesWithoutAlt}/${totalImages}개`);  
  } else if (totalImages > 0) {  
    this.addSuccess(file, 'IMAGES', `✅ 모든 이미지에 alt 속성 있음 (${totalImages}개)`);  
  }  
}
```

```
checkLinks($, file) {  
  const externalLinksWithoutRel = $('a[href^="http"]:not([rel="noopener"])').length;  
  
  if (externalLinksWithoutRel > 0) {  
    this.addIssue(file, 'LINKS', `⚠️ rel="noopener" 없는 외부 링크: ${externalLinksWithoutRel}개`);  
  }  
}
```

```
checkStructuredData($, file) {  
  const jsonLd = $('script[type="application/ld+json"]').length;  
  
  if (jsonLd === 0) {  
    this.addIssue(file, 'STRUCTURED_DATA', `❌ JSON-LD 구조화 데이터가 없습니다.`);  
  } else {  
    this.addSuccess(file, 'STRUCTURED_DATA', `✅ JSON-LD 구조화 데이터 있음 (${jsonLd}개)`);  
  }  
}
```

```
checkOpenGraph($, file) {
  const ogTitle = $('meta[property="og:title"]').attr('content');
  const ogDescription = $('meta[property="og:description"]').attr('content');
  const ogImage = $('meta[property="og:image"]').attr('content');

  if (!ogTitle || !ogDescription || !ogImage) {
    this.addIssue(file, 'OPEN_GRAPH', '❌ Open Graph 메타 태그가 불완전합니다.');
```

}

```
  } else {
    this.addSuccess(file, 'OPEN_GRAPH', '✅ Open Graph 메타 태그 완료');
```

}

```
  }
}
```

addIssue(file, category, message) {

```
  this.issues.push({ file, category, message });
}
```

addSuccess(file, category, message) {

```
  this.successes.push({ file, category, message });
}
```

printReport() {

```
  console.log('\n📊 SEO 감사 보고서\n');
  console.log('=' .repeat(50));

  console.log('\n✅ 성공 항목: ${this.successes.length}개');
  console.log('❌ 문제 항목: ${this.issues.length}개\n');
```

if (this.issues.length > 0) {

```
  console.log('🚨 발견된 문제들:\n');
  this.issues.forEach(issue => {
    console.log(`📄 ${issue.file}`);
```

```
        console.log(` ${issue.message}\n`);
    });
}

console.log('\n💡 SEO 개선 권장사항:');
console.log('• 모든 페이지에 고유한 title과 meta description 설정');
console.log('• 이미지에 적절한 alt 텍스트 추가');
console.log('• 구조화 데이터(JSON-LD) 구현');
console.log('• Open Graph 메타 태그 완성');
console.log('• 내부 링크 구조 최적화');
console.log('• 사이트맵 및 robots.txt 업데이트');
}
}

// 스크립트 실행
if (require.main === module) {
    const auditor = new SEOAuditor();
    auditor.auditSEO().catch(console.error);
}

module.exports = SEOAuditor;
```

 **scripts/dev-server.js**

javascript

```
#!/usr/bin/env node
```

```
/**
```

```
 * Pure-Flon Website - Enhanced Development Server
```

```
 * 기능이 강화된 로컬 개발 서버입니다.
```

```
 */
```

```
const express = require('express');
```

```
const path = require('path');
```

```
const chokidar = require('chokidar');
```

```
const { createProxyMiddleware } = require('http-proxy-middleware');
```

```
const compression = require('compression');
```

```
class DevServer {
```

```
  constructor() {
```

```
    this.app = express();
```

```
    this.port = process.env.PORT || 3000;
```

```
    this.host = process.env.HOST || 'localhost';
```

```
  }
```

```
  setupMiddleware() {
```

```
    // Gzip 압축
```

```
    this.app.use(compression());
```

```
    // CORS 설정
```

```
    this.app.use((req, res, next) => {
```

```
      res.header('Access-Control-Allow-Origin', '*');
```

```
      res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
```

```
      res.header('Access-Control-Allow-Headers', 'Content-Type, Authorization');
```

```
      next();
```

```
    });
```

```
    // 보안 헤더
```

```
this.app.use((req, res, next) => {  
  res.header('X-Content-Type-Options', 'nosniff');  
  res.header('X-Frame-Options', 'DENY');  
  res.header('X-XSS-Protection', '1; mode=block');  
  next();  
});
```

// 정적 파일 서빙

```
this.app.use(express.static('.', {  
  setHeaders: (res, filePath) => {  
    if (filePath.endsWith('.html')) {  
      res.header('Cache-Control', 'no-cache');  
    } else if (filePath.match(/\.(css|js|png|jpg|jpeg|gif|ico|svg)$/)) {  
      res.header('Cache-Control', 'public, max-age=31536000');  
    }  
  }  
}));
```

// API 프록시 (향후 사용)


```
this.app.use('/api', createProxyMiddleware({  
  target: 'http://localhost:3001',  
  changeOrigin: true,  
  pathRewrite: {  
    '^/api': ''  
  }  
}));
```


// SPA 라우팅 (404 처리)

```
this.app.get('*', (req, res) => {  
  res.sendFile(path.join(__dirname, './404.html'));  
});  
}
```









```

setupFileWatcher() {
  const watcher = chokidar.watch([
    '**/*.html',
    '**/*.css',
    '**/*.js',
    '!node_modules/**',
    '!dist/**'
  ]);

  watcher.on('change', (filePath) => {
    console.log( 파일 변경됨: ${filePath});
    // 여기에 라이브 리로드 로직 추가 가능
  });

  console.log( 파일 변경 감시 시작...);
}

start() {
  this.setupMiddleware();
  this.setupFileWatcher();

  this.app.listen(this.port, this.host, () => {
    console.log( Pure-Flon 개발 서버 시작!);
    console.log( URL: http://${this.host}:${this.port});
    console.log( 루트: ${process.cwd()});
    console.log("");
    console.log( 유용한 URL들:);
    console.log( 메인 페이지: http://${this.host}:${this.port}/);
    console.log( 의료용: http://${this.host}:${this.port}/products/medical.html`);
    console.log( 반도체용: http://${this.host}:${this.port}/products/semiconductor.html`);
    console.log( 화학용: http://${this.host}:${this.port}/products/chemical.html`);
  });
}

```

```
    console.log("");  
    console.log("❏ 서버 중지: Ctrl+C");  
  });  
}  
}
```

```
// 스크립트 실행  
if (require.main === module) {  
  const server = new DevServer();  
  server.start();  
}
```

```
module.exports = DevServer;
```

scripts/performance-audit.js

javascript


```
#!/usr/bin/env node
```

```
/**
```

```
 * Pure-Flon Website - Performance Audit Script
```

```
 * 웹사이트 성능을 자동으로 측정하고 보고서를 생성합니다.
```

```
 */
```

```
const lighthouse = require('lighthouse');
```

```
const chromeLauncher = require('chrome-launcher');
```

```
const fs = require('fs');
```

```
const path = require('path');
```

```
class PerformanceAuditor {
```

```
  constructor() {
```

```
    this.results = [];
```

```
  }
```

```
  async auditPerformance() {
```

```
    console.log('⚡ Pure-Flon 성능 감사 시작...\n');
```

```
    const urls = [
```

```
      'http://localhost:3000/',
```

```
      'http://localhost:3000/products/medical.html',
```

```
      'http://localhost:3000/products/semiconductor.html',
```

```
      'http://localhost:3000/products/chemical.html'
```

```
    ];
```

```
    for (const url of urls) {
```

```
      await this.auditURL(url);
```

```
    }
```

```
    this.generateReport();
```

```
  }
```

```
async auditURL(url) {
  console.log('🔍 감사 중: ${url}');

  try {
    const chrome = await chromeLauncher.launch({chromeFlags: ['--headless']});

    const options = {
      logLevel: 'info',
      output: 'json',
      onlyCategories: ['performance'],
      port: chrome.port,
    };

    const runnerResult = await lighthouse(url, options);
    await chrome.kill();

    const result = this.parseResults(url, runnerResult.lhr);
    this.results.push(result);

    console.log('✅ 완료: 점수 ${result.score}/100');

  } catch (error) {
    console.log('❌ 오류: ${error.message}');
  }
}

parseResults(url, lhr) {
  const performance = lhr.categories.performance;
  const metrics = lhr.audits;

  return {
```

```

url,
score: Math.round(performance.score * 100),
metrics: {
  fcp: metrics['first-contentful-paint'].displayValue,
  lcp: metrics['largest-contentful-paint'].displayValue,
  fid: metrics['max-potential-fid'].displayValue,
  cls: metrics['cumulative-layout-shift'].displayValue,
  si: metrics['speed-index'].displayValue,
  tti: metrics['interactive'].displayValue
},
opportunities: lhr.audits['diagnostics'] || [],
timestamp: new Date().toISOString()
};
}

```

```

generateReport() {
  console.log(`\n📊 성능 감사 보고서\n`);
  console.log('=' .repeat(60));

  this.results.forEach(result => {
    console.log(`\n📄 ${result.url}`);
    console.log(`  성능 점수: ${result.score}/100`);
    console.log(`  FCP: ${result.metrics.fcp}`);
    console.log(`  LCP: ${result.metrics.lcp}`);
    console.log(`  FID: ${result.metrics.fid}`);
    console.log(`  CLS: ${result.metrics.cls}`);
  });
}

```

// JSON 보고서 저장

```

const reportPath = path.join(__dirname, './performance-report.json');
fs.writeFileSync(reportPath, JSON.stringify(this.results, null, 2));

```

```

console.log(`\n📄 상세 보고서 저장됨: ${reportPath}`);

this.printRecommendations();
}

printRecommendations() {
  console.log(`\n💡 성능 개선 권장사항:`);
  console.log(`• 이미지 최적화 (WebP 포맷 사용)`);
  console.log(`• CSS/JS 파일 압축`);
  console.log(`• 불필요한 JavaScript 제거`);
  console.log(`• CDN 사용 고려`);
  console.log(`• 캐싱 전략 개선`);
  console.log(`• 폰트 로딩 최적화`);
}
}

// 스크립트 실행
if (require.main === module) {
  const auditor = new PerformanceAuditor();
  auditor.auditPerformance().catch(console.error);
}

module.exports = PerformanceAuditor;

```

📄 package.json에 추가할 스크립트들

json

```
{
  "scripts": {
    "optimize:images": "node scripts/optimize-images.js",
    "audit:seo": "node scripts/seo-check.js",
    "audit:performance": "node scripts/performance-audit.js",
    "dev:enhanced": "node scripts/dev-server.js",
    "check:all": "npm run audit:seo && npm run audit:performance"
  },
  "devDependencies": {
    "sharp": "^0.32.6",
    "cheerio": "^1.0.0-rc.12",
    "glob": "^10.3.10",
    "lighthouse": "^10.4.0",
    "chrome-launcher": "^0.15.2",
    "express": "^4.18.2",
    "chokidar": "^3.5.3",
    "http-proxy-middleware": "^2.0.6",
    "compression": "^1.7.4"
  }
}
```

사용 방법

bash

의존성 설치

`npm install`

이미지 최적화

`npm run optimize:images`

SEO 검사

`npm run audit:seo`

성능 검사 (개발 서버 실행 후)

`npm run dev:enhanced`

`npm run audit:performance`

전체 검사

`npm run check:all`

이 스크립트들을 사용하면 Pure-Flon 웹사이트의 품질을 지속적으로 관리하고 개선할 수 있습니다! 🎯