HTML/SEO 표준 가이즈

- 메타 태그 설정
- Owner Key 활용하여 외부 API 호출 예시
- 제목 태그 (<h1>~<h6>)
- aria-label 속성
- lang 속성
- label 태그
- alt 속성
- Apache를 통한 보안설정
  - [SECURITY] http header의 cookie
  - [Security] Content-Security-Policy
  - [SECURITY] Header 보안설정하기

메타 태그 설정

> “
>
> ## 메타 태그 설정
>
> 본 페이지는 퍼블리싱 작업중 HTML의 표준화를 위해 작성되었습니다.

<title> 태그

검색 결과에서 웹페이지의 제목으로 표시되며, 사용자에게 페이지 내용을 요약해 알려주는 중요한 요소입니다. 브라우저 탭에도 제목으로 표시됩니다

- 50~60자 이내로 간결하게 작성하며, 페이지의 핵심 내용을 포함합니다.
- 페이지마다 고유한 제목을 설정해 중복을 피하고, 중요한 키워드를 자연스럽게 포함합니다.

<meta name="description"> 태그

페이지에 대한 요약 정보를 제공합니다. 검색 결과에 표시되며, 사용자에게 페이지 내용을 미리 알려주는 역할을 합니다.

- 150~160자 이내로 작성해 검색 엔진에 잘 표시되도록 합니다.
- 매력적이고 클릭을 유도할 수 있는 문구를 포함합니다.
- 페이지의 주요 키워드를 포함하되, 내용과 부합해야 합니다.

<meta name="robots"> 태그

검색 엔진이 페이지를 크롤링하고 인덱싱할 때의 지침을 제공합니다.

- `index` / `noindex` : 페이지를 검색 엔진 인덱스에 포함할지 여부를 지정.
- `follow` / `nofollow` : 페이지 내 링크를 따라갈지 여부를 설정.
- 예를 들`noindex, nofollow`설정합니다.

<meta name="keywords"> 태그

페이지와 관련된 키워드를 명시합니다. 현재는 대부분의 검색 엔진에서 무시되지만, 여전히 일부 **SEO** 도구에서 분석 요소로 사용됩니다.

- 5~10개의 키워드를 쉼표로 구분하여 간략하게 작성합니다.
- 페이지의 핵심 주제와 관련된 키워드를 포함하되, 중복 키워드나 스팸성 키워드는 피합니다.

<meta charset="UTF-8"> 태그

웹페이지의 문자 인코딩 방식을 지정합니다. **UTF-8**은 대부분의 언어와 특수 문자를 지원하기 때문에 널리 사용됩니다.

<meta name="viewport"> 태그

모바일 기기에서 화면의 초기 크기를 설정합니다. 반응형 웹 디자인을 구현할 때 필수적입니다.

- `width=device-width` : 디바이스의 가로 폭에 맞게 화면 너비를 설정.
- `initial-scale=1.0` : 기본 확대 비율을 설정하여 페이지가 사용자 디바이스에 맞게 표시되도록 설정.

<meta property="og:title"> 태그

페이지가 소셜 미디어에서 공유될 때 표시될 정보를 설정합니다. **Facebook, Twitter**와 같은 플랫폼에서 미리보기 정보로 활용됩니다.

- **og**: 소셜 미디어에서 표시될 제목.
- **og**: 짧고 간결한 설명.
- **og**: 고해상도 이미지를 사용하여 시각적 매력을 높임.
- **og**: 해당 페이지의 정확한 URL을 사용.

<link rel="canonical"> 태그

동일한 콘텐츠가 여러 **URL**로 접근 가능한 경우, 검색 엔진에 대표 **URL**을 명시하여 중복 콘텐츠 문제를 방지합니다.

- 대표 콘텐츠로 인식되길 원하는 URL을 지정해 중복 페이지가 생성되는 것을 방지합니다.
- 특히 필터 또는 정렬 옵션을 통해 동일한 페이지가 여러 URL로 접근 가능한 경우 유용합니다.

<meta http-equiv="X-UA-Compatible"> 태그

페이지가 **Internet Explorer**에서 특정 모드로 렌더링되도록 지시합니다.

최신 `IE=edge`로 설정하여 페이지가 최신 브라우저 표준에 맞게 표시되도록 합니다.

Owner Key 활용하여 외부 API 호출 예시

Opticalpos가 Host일 때

- **Owner Key:** Opticalpos의 DB 정보(`.env` 파일에 작성)
- **Main (Host):** Opticalpos
- **Guest:** Eyemsg

요청 주체	API 호출 대상	사용 DB
Opticalpos	Eyemsg API	**Opticalpos DB**
Eyemsg	Opticalpos API	**Opticalpos DB**

Eyemsg가 Host일 때

- **Owner Key:** Eyemsg의 DB 정보(`.env` 파일에 작성)
- **Main (Host):** Eyemsg
- **Guest:** Opticalpos

요청 주체	API 호출 대상	사용 DB
Eyemsg	Opticalpos API	**Eyemsg DB**
Opticalpos	Eyemsg API	**Eyemsg DB**

핵심:

- **Host**의 **DB** 정보를 **Owner Key**로 관리
- **Guest**는 **Owner Key**를 이용해 **Host**의 **API**를 호출
- **API** 호출 시 항상 **Host**의 **DB** 정보를 사용

제목 태그 (<h1>~<h6>)

제목 <h1> ~ <h6>)

콘텐츠의 구조와 중요도에 따라 제목을 지정하여 계층 구조를 만듭니다.

- <h1> 태그는 페이지 당 하나로 설정.
- <h2> ~ <h6> 태그는 중요도에 따라

```
<h1>페이지 메인 제목</h1>
<h2>부제목 1</h2>
<h3>소제목 1.1</h3>
<h2>부제목 2</h2>
```

aria-label 속성

aria-label 속성이 필요한 경우

1. 레이블 텍스트가 없는 경우

<input> 는 <label> 요소가 aria-label 에 필요합니다. 예를 들어, 검색 상자, 텍스트 입력 창 등에 추가하여 필드의 목적을 설명할 수 있습니다.

```
<input type="text" aria-label="Search">
```

2. 레이블을 커스터마이징하고 싶은 경우

레이블은 aria-label 을 사용하여 필드의 의미를 더 잘 전달하도록 할 수 있습니다.

3. 아이콘만 있는 입력 필드

aria-label 은 화면에 보이는 레이블이 없고 아이콘으로만 이루어진 입력 필드에 유용합니다.

```
<input type="text" aria-label="Phone number">
<i class="icon-phone"></i>
```

aria-label 속성이 필요하지 않은 경우

1. 명확한 레이블이 있는 경우

`<label>` 요소로 `aria-label` 이 작성되어 있어 필요없음

aria-label [illegible]

aria-label [illegible]

aria-label [illegible]

1. [illegible]

`<input>` [illegible] `<label>` [illegible] `aria-label` [illegible]. [illegible], [illegible], [illegible].

```
<input type="text" aria-label="Search">
```

2. [illegible]: [illegible]

[illegible] `aria-label` [illegible].

3. [illegible]

`aria-label` [illegible].

```
<input type="text" aria-label="Phone number">
<i class="icon-phone"></i>
```

aria-label [illegible]

1. [illegible]

`<label>` [illegible] `aria-label` [illegible]

lang 속성

lang 속성

lang 속성은 <html> 태그에 설정하는 것이 일반적입니다. 예를 들어, 문서가 한국어로 작성되었으면 lang="ko"를 지정하고, 영어로 작성된 경우 lang="en"을 지정합니다.

```
<html lang="ko-KR">
```

lang 웹 SEO 성능을 높이는 중요한 [<html>] 태 lang 속 설정하는 것이 좋습니다.

label [illegible]

label [illegible]

<label> [illegible] <input>, <textarea>, <select>[illegible]. <label>[illegible] for[illegible] id [illegible]

```
<form>
  <label for="email">Email Address</label>
  <input type="email" id="email" name="email">
</form>
```

label [illegible]

- [illegible] `<label>` [illegible].[illegible]
- [illegible] `<input type="text">` [illegible] `<label>` [illegible],[illegible]

label [illegible]

[illegible] `<label>` [illegible] `<label>` [illegible]

- [illegible] `` ,[illegible] `` [illegible] `<label>` [illegible].
- [illegible] `<label>` [illegible] `<label>` [illegible].[illegible]

alt [illegible]

alt [illegible]

`alt` [illegible] `<img>` [illegible].

[illegible].

```
<img src="logo.png" alt="Company Logo">
```

alt [illegible]

- [illegible]: [illegible] `alt` [illegible] `alt=""` [illegible].
- [illegible] `alt` [illegible] `alt` [illegible],

Apache□ □□ □□□□

Apache를 통한 보안설정

[SECURITY] http header의 cookie

쿠키 이름	역할	**Secure** 플래그	**HttpOnly** 플래그
PHPSESSID	PHP 기본 세션 쿠키	적용 안됨	적용 안됨
XSRF-TOKEN	CSRF 보호를 위한 토큰	적용됨	적용 안됨
Laravel 세션 쿠키:	Laravel 세션 쿠키	적용됨	적용됨

XSRF-ROKEN과 Laravel 세션 쿠키는 프론트에서 처리 가능

> “
>
> **PHPSESSID**의 **Secure** 플래그 추가 방법

- **php.ini** 설정 수정

  session.cookie_secure = On

- **PHP** 코드에서 설정

  session_start()를 호출하기 전에 아래와 같이 설정합니다:

```
ini_set('session.cookie_secure', 1);
ini_set('session.cookie_httponly', 1); // HttpOnly 추가
session_start();
```

- **.htaccess** 파일에서 설정

- `.htaccess`를 사용하여 PHP 설정을 수정할 수 있습니다:

  ```
  php_value session.cookie_secure On
  php_value session.cookie_httponly On
  ```

Apache를 통한 보안설정

[Security] Content-Security-Policy

Content-Security-Policy header is not set. It restricts resources the page can load and prevents XSS attacks.

seo 진단시 위와 같은 진단내용이 잡힌다면 아래 내용들을 확인해주세요.

> “
> **Content-Security-Policy(CSP)란?**

CSP는 HTTP 응답 헤더 중 하나로, 브라우저가 페이지에 어떤 리소스를 로드할 수 있는지 제어하는 보안 정책을 정의합니다.
이를 통해 아래와 같은 보안 위협을 방지할 수 있습니다:

1. **XSS(**크로스 사이트 스크립팅**)**
   공격자가 악성 JavaScript를 주입하는 것을 방지합니다.
2. 데이터 주입 공격
   악성 리소스(URL, CSS, 폰트 등)의 로드를 제한합니다.
3. **Clickjacking**
   악성 iframe 삽입 등을 방지합니다.

> “
> **Content-Security-Policy**가 중요한 이유

CSP가 없으면 다음과 같은 문제가 발생할 수 있습니다:

- 외부에서 [illegible]를 주입하여 사
- 비인가된 리소스가 로드되어 정보 유출 가능.
- HTTPS 환경에서도 보안이 취약한 HTTP 리소스가 로드될 가능성.

>

해결 방법: Content-Security-Policy 헤더 설정

Apache 설정을 통해 CSP 추가

Apache[Content-Security-Policy 헤 .htaccess 파일[httpd.conf , apache2.conf)에 다음 규칙을 추가합니다.

```
<IfModule mod_headers.c>
Header set Content-Security-Policy "default-src 'self'; script-src 'self' https://www.googletagmanager.com
https://d3js.org https://cdnjs.cloudflare.com https://cdn.jsdelivr.net 'unsafe-inline' 'unsafe-eval'; style-src 'self'
https://fonts.googleapis.com https://cdn.jsdelivr.net 'unsafe-inline'; font-src 'self' https://fonts.gstatic.com
https://cdn.jsdelivr.net data:; connect-src 'self' https://www.google-analytics.com; img-src 'self' data:
https://cdn.jsdelivr.net;"
</IfModule>
```

default-src 'self'

현재 사이트 외의 다른 출처에서 리소스(스크립트, 스타일, 이미지)를 불러오지 못하게 방지합니다.

self

현재 사이트 자체에서 제공되는 스크립트만 허용합니다.

unsafe-inline

인라인 스크립트를 허용합니다 (보안상 추천되지 않음).

unsafe-eval

eval() [setTimeout(string) 같은 동적 코드 실행을 허용합니다.

data:

데이터 URI를 통해 폰트, 이미지 등의 데이터를 허용합니다.

* 줄바꿈이 포함되면 500 server error가 발생할 수 있습니다.

2. 프론트에서 미들웨어 추가 (라라벨 기준)

1. app/Http/Middleware에 새로운 class를 생성합니다.

2. 아래와 같이 handle function을 추가합니다. $csp 변수 안에 내용은 직접 변경하셔야합니다.

```
public function handle(Request $request, Closure $next)
    {
        $response = $next($request);



        $csp = "default-src 'self'; " .
       "script-src 'self' https://www.googletagmanager.com https://d3js.org https://cdnjs.cloudflare.com
https://cdn.jsdelivr.net 'unsafe-inline' 'unsafe-eval'; " .
       "style-src 'self' https://fonts.googleapis.com https://cdn.jsdelivr.net 'unsafe-inline'; " .
       "font-src 'self' https://fonts.gstatic.com https://cdn.jsdelivr.net data:; " .
       "connect-src 'self' https://www.google-analytics.com; " .
       "img-src 'self' data: https://cdn.jsdelivr.net;";



        $response->headers->set('Content-Security-Policy', $csp);



        return $response;
    }
```

3. `app/Http/Kernel.php` 에 미들웨어를 추가합니다.

```
protected $middleware = [ \App\Http\Middleware\ContentSecurityPolicy::class, ];
```

권장사항

엄격한 정책으로 전환

- [`unsafe-inline` 을 사용할 수 있지만, 최종 배포시 제거하는 것이 좋습니다.
- `nonce` [`hash` 기반 CSP를 사용하여 특정 스크립트만 허용하도록 설정하는 것이 좋습니다.

Nonce를 사용 (권장)

아래 코드는 nonce 적용에 대한 예시코드입니다.

- apache 설정

```
Header set Content-Security-Policy "default-src 'self'; script-src 'self' https://www.googletagmanager.com
https://d3js.org https://cdnjs.cloudflare.com 'nonce-abc123'; style-src 'self' https://fonts.googleapis.com
https://cdn.jsdelivr.net 'unsafe-inline'; font-src 'self' https://fonts.gstatic.com; connect-src 'self'
https://api.example.com; img-src 'self' data:;"
```

- script 사용부분

```
<script nonce="abc123"> console.log('Inline script with nonce'); </script>
```

- 동적 Nonce 생성

```
<?php
// php + javascript
$nonce = base64_encode(random_bytes(16));
header("Content-Security-Policy: script-src 'self' 'nonce-$nonce'");
?>

<script nonce="<?php echo $nonce; ?>">
  console.log('This script uses a nonce');
</script>
```

```
<script>
  // only javascript
  // Generate a random nonce
  const array = new Uint8Array(16);
```

```
  window.crypto.getRandomValues(array);
  const nonce = btoa(String.fromCharCode.apply(null, array));

  // Set the Content-Security-Policy header
  const meta = document.createElement('meta');
  meta.setAttribute('http-equiv', 'Content-Security-Policy');
  meta.setAttribute('content', `script-src 'self' 'nonce-${nonce}'`);
  document.head.appendChild(meta);

  // Dynamically create a script element with the nonce
  const script = document.createElement('script');
  script.setAttribute('nonce', nonce);
  script.textContent = "console.log('This script uses a nonce')";
  document.body.appendChild(script);
</script>
```

[SECURITY] Header □□□□□

Header	OK	Notice	Warning	Critical	Recommendation
X-Frame-Options	0	0	77	0	X-Frame-Options header is not set. It prevents clickjacking attacks when set to 'deny' or 'sameorigin.'
X-Content-Type-Options	0	0	77	0	X-Content-Type-Options header is not set. It stops MIME type sniffing and mitigates content type attacks.
Referrer-Policy	0	0	77	0	Referrer-Policy header is not set. It controls referrer header sharing and enhances privacy and security.
Feature-Policy	0	0	77	0	Feature-Policy header is not set. It allows enabling/disabling browser APIs and features for security. Not important if Permissions-Policy is set.
Permissions-Policy	0	0	77	0	Permissions-Policy header is not set. It allows enabling/disabling browser APIs and features for security.
Server	0	0	77	0	Server header is set to known 'Apache.' It is better not to reveal used technologies.
Set-Cookie	70	65	70	0	Set-Cookie header for 'PHPSESSID' does not have 'SameSite' flag. Consider using 'SameSite=Strict' or 'SameSite=Lax.'
					Set-Cookie header for 'XSRF-TOKEN' does not have 'HttpOnly' flag. Attacker can steal the cookie using XSS. Consider using

'HttpOnly' when cookie is not used by JavaScript.

> “
> X-Frame-Options

문제:

- X-Frame-Options 헤더가 설정되지 않음.

해결:

Apache 설정 파일에 다음을 추가:

```
Header always set X-Frame-Options "DENY"
```

- **DENY**: iframe 로딩을 완전히 차단.
- **SAMEORIGIN**: 같은 도메인에서만 iframe 로딩 허용.

> “
> **X-Content-Type-Options**

문제:

- X-Content-Type-Options 헤더가 설정되지 않음.

해결:

Apache 설정 파일에 다음을 추가:

```
Header always set X-Content-Type-Options "nosniff"
```

- MIME 타입 스니핑 방지.

> # Referrer-Policy

문제:

- Referrer-Policy 헤더가 설정되지 않음.

해결:

Apache 설정 파일에 다음을 추가:

```
Header always set Referrer-Policy "no-referrer"
```

- **no-referrer**: 참조 헤더를 완전히 숨김.
- 다 `strict-origin`, `strict-origin-when-cross-origin`, `same-origin` 등.

> “
> # Feature-Policy / Permissions-Policy

문제:

- Feature-Policy 또는 Permissions-Policy 헤더가 설정되지 않음.

해결:

Apache 설정 파일에 다음을 추가:

```
Header always set Permissions-Policy "geolocation=(), camera=(), microphone=()"
```

- **Permissions-Policy**: 특정 브라우저 기능(API) 사용을 제한.
- 예: `geolocation`, `camera`, `microphone` 등.

Server

문제:

- Server 헤더가 Apache로 설정되어 서버 정보가 노출됨.

해결:

Apache 설정 파일에 다음을 추가:

```
ServerTokens Prod
ServerSignature Off
Header unset Server
```

- **ServerTokens Prod**: 최소한의 정보만 노출.
- **ServerSignature Off**: 오류 페이지에서 Apache 정보 숨김.
- **Header unset Server**: Server 헤더를 제거.

> “
Set-Cookie

문제:

- SameSite 및 HttpOnly 플래그가 없음.

해결:

Apache 설정 파일에 다음을 추가:

```
Header always edit Set-Cookie ^(.*)$ "$1; SameSite=Strict; HttpOnly; Secure"
```

- **SameSite=Strict**: 쿠키를 외부 요청에 전송하지 않음.
- **HttpOnly**: JavaScript에서 쿠키 접근을 차단.
- **Secure**: HTTPS 요청에서만 쿠키를 전송.

□□ □□□

```
# X-Frame-Options □□
Header always set X-Frame-Options "DENY"

# X-Content-Type-Options □□
Header always set X-Content-Type-Options "nosniff"

# Referrer-Policy □□
Header always set Referrer-Policy "no-referrer"

# Permissions-Policy □□
Header always set Permissions-Policy "geolocation=(), camera=(), microphone=()"

# Server □□ □□□
ServerTokens Prod
ServerSignature Off
Header unset Server

# Set-Cookie □□ □□
Header always edit Set-Cookie ^(.*)$ "$1; SameSite=Strict; HttpOnly; Secure"
```