Effects.lab

GUIDE COMPLET DU DEVELOPPEMENT WEB

Les Bases du Web

HTML, CSS, JavaScript et tout ce qu'il faut savoir pour creer un site web de A a Z

8	45+	700+
Chapitres complets	Exemples de code	Effets sur Effects.lab

effects.lab | 2026 | Version 1.0

HTML - CSS - JavaScript - Responsive - a11y - SEO - Securite

Sommaire

- Core Web Vitals (LCP, FID, CLS)
- Optimisation des images
- Preload, prefetch, preconnect
- Minification et optimisation du code

07 Outils & Workflow

- VS Code - Editeur et extensions
- Git - Versionner son code
- DevTools du navigateur
- Deploiement (GitHub Pages, Netlify, Vercel)
- Domaine et hebergement

08 Securite Web

- HTTPS - Chiffrement obligatoire
- Content Security Policy (CSP)
- Prevention des attaques XSS
- Protection CSRF
- Sanitization des inputs
- Headers de securite

01 HTML - La Structure

Les fondations de chaque page web

01 Structure d'une page HTML5

HTML FONDAMENTAL

ESSENTIEL

Chaque page HTML5 commence par une structure de base obligatoire. Le DOCTYPE indique au navigateur de traiter le document en mode standard. Les balises meta dans le head definissent le charset, le viewport et les informations pour le SEO.

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, initial-scale=1.0">
  <meta name="description"
    content="Description de votre page">
  <title>Titre de la page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <!-- Contenu visible ici -->
  <script src="app.js" defer></script>
</body>
</html>
```

DOCTYPE Head Meta

02 Balises semantiques HTML5

HTML SEMANTIQUE

ESSENTIEL

Les balises semantiques donnent du sens a votre structure. Elles aident les moteurs de recherche, les lecteurs d'ecran et les developpeurs a comprendre l'organisation du contenu. Chaque page devrait avoir un header, main et footer.

```
<header>
  <nav aria-label="Navigation principale">
    <a href="/">Accueil</a>
    <a href="/about">A propos</a>
  </nav>
</header>
<main>
  <article>
    <h1>Titre de l'article</h1>
    <section>
      <h2>Sous-section</h2>
      <p>Contenu...</p>
    </section>
    <aside>Contenu complementaire</aside>
  </article>
</main>
<footer>
  <p>&copy; 2026 MonSite</p>
</footer>
```

Semantique Structure SEO

HTML FORMULAIRES

03 Formulaires HTML

IMPORTANT

Les formulaires sont le principal moyen d'interaction. Chaque input doit avoir un label associe (via for/id). Utilisez les types d'input HTML5 pour la validation native et les claviers mobiles adaptes.

```
<form action="/submit" method="POST">
  <fieldset>
    <legend>Informations personnelles</legend>
    <label for="name">Nom complet</label>
    <input type="text" id="name" name="name"
      required autocomplete="name"
      placeholder="Jean Dupont">
    <label for="email">Email</label>
    <input type="email" id="email" name="email"
      required autocomplete="email"
      inputmode="email">
    <label for="tel">Telephone</label>
    <input type="tel" id="tel" name="tel"
      pattern="[0-9]{10}">
    <button type="submit">Envoyer</button>
  </fieldset>
</form>
```

Formulaires Validation Input

HTML TABLEAUX

04 Tableaux de donnees

UTILE

Les tableaux servent exclusivement a afficher des donnees tabulaires (jamais pour la mise en page). Utilisez thead/tbody pour separer l'en-tete du contenu et scope pour l'accessibilite.

```
<table>
  <caption>Tarifs mensuels</caption>
  <thead>
    <tr>
      <th scope="col">Plan</th>
      <th scope="col">Prix</th>
      <th scope="col">Effets</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Starter</td>
      <td>9 EUR/mois</td>
      <td>100 effets</td>
    </tr>
    <tr>
      <td>Pro</td>
      <td>29 EUR/mois</td>
      <td>742 effets</td>
    </tr>
  </tbody>
</table>
```

Tableaux Accessibilite Donnees

HTML MULTIMEDIA

05 Multimedia (images, video, audio)

IMPORTANT

HTML5 permet d'integrer facilement des medias. Utilisez la balise picture pour servir differents formats d'image. L'attribut loading='lazy' differe le chargement des images hors ecran.

```
<!-- Image responsive avec fallback -->
<picture>
  <source srcset="photo.avif" type="image/avif">
  <source srcset="photo.webp" type="image/webp">
  <img src="photo.jpg" alt="Description"
    loading="lazy" decoding="async"
    width="800" height="600">
</picture>
<!-- Video avec controles -->
<video controls preload="metadata"
  poster="thumbnail.jpg">
  <source src="video.mp4" type="video/mp4">
  <track kind="subtitles" src="fr.vtt"
    srclang="fr" label="Francais">
</video>
```

Images Video Picture

HTML NAVIGATION

06 Liens et navigation

ESSENTIEL

Les liens sont le tissu du web. Utilisez des href descriptifs, target='_blank' pour les liens externes (avec rel='noopener'), et des ancres pour la navigation intra-page. Les liens doivent etre reconnaissables visuellement.

```
<!-- Lien interne -->
<a href="/contact">Contactez-nous</a>
<!-- Lien externe (securise) -->
<a href="https://example.com"
  target="_blank"
  rel="noopener noreferrer">
  Visiter le site
</a>
<!-- Lien ancre (meme page) -->
<a href="#section-2">Aller a la section 2</a>
<!-- Lien telephone / email -->
<a href="tel:+33612345678">Appeler</a>
<a href="mailto:contact@site.fr">Email</a>
<!-- Lien de telechargement -->
<a href="guide.pdf" download>Telecharger</a>
```

Liens Navigation Target

02 CSS - Le Style

Donnez vie a vos pages avec le CSS

07 Selecteurs CSS essentiels

CSS SELECTEURS

ESSENTIEL

Les selecteurs CSS ciblent les elements a styliser. Maitrisez les selecteurs de base (element, classe, id), les combinateurs (descendant, enfant, adjacent) et les pseudo-classes pour un code precis et maintenable.

```
/* Selecteurs de base */
h1 { }              /* Element */
.card { }           /* Classe */
#hero { }           /* ID (eviter si possible) */
/* Combinateurs */
.card p { }         /* Descendant */
.card > p { }       /* Enfant direct */
.card + p { }       /* Adjacent */
.card ~ p { }       /* General sibling */
/* Pseudo-classes */
a:hover { }         /* Survol */
input:focus { }     /* Focus */
li:first-child { }  /* Premier enfant */
li:nth-child(2n) {} /* Pairs */
/* Pseudo-elements */
p::first-line { }   /* Premiere ligne */
.card::before { content: ""; }
```

Selecteurs | Specificity | CSS

08 Le Box Model

CSS BOX MODEL

ESSENTIEL

Le Box Model est le concept le plus fondamental du CSS. Chaque element est une boite composee de content, padding, border et margin. Utilisez toujours box-sizing: border-box pour que padding et border soient inclus dans la largeur.

```
/* Reset essentiel */
*, *::before, *::after {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
/* Comprendre le box model */
.card {
  width: 300px;       /* Largeur du contenu */
  padding: 20px;      /* Espace interieur */
  border: 2px solid #e5e7eb; /* Bordure */
  margin: 16px;       /* Espace exterieur */
  /* Avec border-box: largeur totale = 300px */
  /* Sans border-box: largeur = 300+40+4 = 344px */
}
/* Raccourci margin/padding */
.box {
  margin: 10px 20px;  /* vertical horizontal */
  padding: 10px 20px 30px; /* top H bottom */
}
```

Box Model | Border-box | Layout

CSS FLEXBOX

09 Flexbox - Mise en page flexible

ESSENTIEL

Flexbox est ideal pour les layouts unidimensionnels (ligne ou colonne). Il simplifie le centrage, la distribution d'espace et le reordonnement des elements. C'est l'outil de base pour la majorite des mises en page.

```
/* Container flex */
.nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: 16px;
}
/* Centrage parfait */
.hero {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}
/* Layout responsive */
.grid {
  display: flex;
  flex-wrap: wrap;
  gap: 24px;
}
.grid > .card {
  flex: 1 1 300px; /* grow shrink basis */
}
/* Colonne avec footer en bas */
.page {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}
.page > main { flex: 1; }
```

Flexbox Layout Centrage

CSS GRID

10 CSS Grid - Grilles bidimensionnelles

ESSENTIEL

CSS Grid excelle pour les layouts 2D complexes. Utilisez fr pour les fractions, repeat() pour simplifier, auto-fit/auto-fill pour le responsive sans media queries, et grid-areas pour des layouts nommes.

```
/* Grille responsive auto */
.gallery {
  display: grid;
  grid-template-columns:
    repeat(auto-fill, minmax(280px, 1fr));
  gap: 24px;
}
/* Layout avec zones nommees */
.page {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 250px 1fr;
  grid-template-rows: auto 1fr auto;
}
.page header  { grid-area: header; }
.page aside   { grid-area: sidebar; }
.page main    { grid-area: main; }
.page footer  { grid-area: footer; }
```

Grid Layout Responsive

CSS VARIABLES

11 Variables CSS (Custom Properties)

IMPORTANT

Les variables CSS centralisent vos valeurs de design (couleurs, espacements, typographie). Definies dans :root, elles sont accessibles partout. Elles permettent le theming, le mode sombre et la maintenance facile.

```
:root {
  /* Couleurs */
  --primary: #6366f1;
  --primary-light: #818cf8;
  --text: #1a1a1a;
  --bg: #ffffff;
  /* Espacements */
  --space-xs: 4px;
  --space-sm: 8px;
  --space-md: 16px;
  --space-lg: 32px;
  --space-xl: 64px;
  /* Typographie */
  --font-sans: system-ui, sans-serif;
  --font-mono: "Fira Code", monospace;
  /* Rayons */
  --radius-sm: 6px;
  --radius-md: 12px;
  --radius-full: 9999px;
}
/* Utilisation avec fallback */
.btn {
  background: var(--primary, #6366f1);
  padding: var(--space-sm) var(--space-md);
  border-radius: var(--radius-md);
  font-family: var(--font-sans);
}
```

Variables Design System Theming

CSS ANIMATIONS

12 Transitions et animations

IMPORTANT

Les transitions animent le passage entre deux etats (hover, focus). Les animations @keyframes permettent des sequences complexes. Privilegiez transform et opacity pour des animations fluides a 60fps.

```
/* Transition (changement d'etat) */
.btn {
  background: var(--primary);
  transition: all 0.3s ease;
}
.btn:hover {
  transform: translateY(-3px);
  box-shadow: 0 10px 30px rgba(0,0,0,0.2);
}
/* Animation keyframes */
@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
.card {
  animation: fadeInUp 0.6s ease forwards;
}
/* Animation au scroll (avec JS) */
.reveal {
  opacity: 0;
  transform: translateY(30px);
  transition: all 0.8s ease;
}
.reveal.visible {
  opacity: 1;
  transform: translateY(0);
}
```

Transitions Animations Keyframes

CSS RESPONSIVE

13 Responsive Design (Media Queries)

ESSENTIEL

Le responsive design adapte votre site a toutes les tailles d'ecran. Utilisez une approche mobile-first, clamp() pour les valeurs fluides, et les container queries pour le responsive par composant.

```
/* Mobile-first approach */
.container {
  width: 100%;
  padding: 16px;
}
/* Tablette */
@media (min-width: 768px) {
  .container {
    max-width: 720px;
    margin: 0 auto;
  }
}
/* Desktop */
@media (min-width: 1024px) {
  .container { max-width: 960px; }
}
/* Typographie fluide avec clamp */
h1 {
  font-size: clamp(1.8rem, 4vw, 3.5rem);
}
/* Container queries */
.card-container {
  container-type: inline-size;
}
@container (min-width: 400px) {
  .card { display: flex; gap: 16px; }
}
```

Responsive | Media Queries | Clamp

03 JavaScript - L'Interaction

Rendez vos pages dynamiques et interactives

14 Variables et types de donnees

JS FONDAMENTAUX

ESSENTIEL

JavaScript utilise let et const pour declarer des variables (evitez var). Const pour les valeurs qui ne changent pas, let pour les autres. Les types principaux : string, number, boolean, array, object, null, undefined.

```
// Constantes (ne change jamais)
const API_URL = "https://api.example.com";
const MAX_ITEMS = 50;
// Variables (peut changer)
let count = 0;
let userName = "Marie";
// Types de donnees
const name = "Effects.lab";    // string
const price = 29.99;           // number
const isActive = true;         // boolean
const colors = ["red", "blue"]; // array
const user = {                  // object
  name: "Jean",
  age: 28,
  isPro: true
};
// Verifier le type
typeof name;    // "string"
typeof price;   // "number"
Array.isArray(colors); // true
```

Variables Types Let/Const

15 JS DOM
Manipulation du DOM

ESSENTIEL

Le DOM (Document Object Model) est la representation de la page en objets JavaScript. querySelector selectionne un element, querySelectorAll en selectionne plusieurs. Modifiez le contenu, les classes et les attributs.

```
// Selectionner des elements
const btn = document.querySelector(".btn");
const cards = document.querySelectorAll(".card");
const hero = document.getElementById("hero");
// Modifier le contenu
btn.textContent = "Cliquez ici";
btn.innerHTML = "<span>Cliquez</span>";
// Classes CSS
btn.classList.add("active");
btn.classList.remove("hidden");
btn.classList.toggle("open");
btn.classList.contains("active"); // true
// Attributs
btn.setAttribute("disabled", "true");
btn.removeAttribute("disabled");
// Creer et inserer
const div = document.createElement("div");
div.className = "notification";
div.textContent = "Nouveau message";
document.body.appendChild(div);
```

DOM querySelector classList

16 JS EVENEMENTS
Evenements JavaScript

ESSENTIEL

Les evenements permettent de reagir aux actions de l'utilisateur. Utilisez addEventListener (jamais onclick inline). La delegation d'evenements optimise la performance en attachant un seul listener au parent.

```
// Evenement click
const btn = document.querySelector(".btn");
btn.addEventListener("click", () => {
  console.log("Bouton clique !");
});
// Evenement formulaire
const form = document.querySelector("form");
form.addEventListener("submit", (e) => {
  e.preventDefault(); // Empecher le rechargement
  const data = new FormData(form);
  console.log(data.get("email"));
});
// Delegation d'evenements
const list = document.querySelector(".list");
list.addEventListener("click", (e) => {
  if (e.target.matches(".delete-btn")) {
    e.target.closest("li").remove();
  }
});
// Scroll event (optimise)
window.addEventListener("scroll", () => {
  const scrollY = window.scrollY;
  // Logique au scroll...
}, { passive: true });
```

Events Click Delegation

JS ASYNC

17 Fetch API et requetes asynchrones

ESSENTIEL

Fetch permet de communiquer avec des APIs. Utilisez async/await pour un code lisible. Gerez toujours les erreurs avec try/catch. Verifiez response.ok car fetch ne rejette pas les erreurs HTTP.

```
// GET - Recuperer des donnees
async function getUsers() {
  try {
    const response = await fetch("/api/users");
    if (!response.ok) {
      throw new Error(`HTTP ${response.status}`);
    }
    const users = await response.json();
    return users;
  } catch (error) {
    console.error("Erreur:", error);
  }
}
// POST - Envoyer des donnees
async function createUser(data) {
  const response = await fetch("/api/users", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(data),
  });
  return response.json();
}
// Utilisation
const users = await getUsers();
await createUser({ name: "Marie", email: "m@x.fr" });
```

Fetch | Async/Await | API

JS STOCKAGE

18 LocalStorage - Stockage local

UTILE

LocalStorage permet de stocker des donnees cote client qui persistent apres la fermeture du navigateur. Limite a 5-10 MB. Stocke uniquement des strings : utilisez JSON.stringify/parse pour les objets.

```
// Stocker une valeur
localStorage.setItem("theme", "dark");
// Recuperer une valeur
const theme = localStorage.getItem("theme");
console.log(theme); // "dark"
// Supprimer une valeur
localStorage.removeItem("theme");
// Stocker un objet (JSON)
const settings = {
  theme: "dark",
  language: "fr",
  fontSize: 16
};
localStorage.setItem("settings",
  JSON.stringify(settings));
// Recuperer un objet
const saved = JSON.parse(
  localStorage.getItem("settings")
);
console.log(saved.theme); // "dark"
// Tout effacer
localStorage.clear();
```

LocalStorage | JSON | Persistance

JS MODERNE

19 ES6+ - Syntaxe moderne

IMPORTANT

ES6+ a modernise JavaScript avec les arrow functions, le destructuring, les template literals, l'operateur spread, les modules et bien plus. Maitrisez ces features pour ecrire du code concis et lisible.

```
// Arrow functions
const add = (a, b) => a + b;
const greet = name => `Bonjour ${name}`;
// Destructuring
const { name, age } = user;
const [first, ...rest] = items;
// Template literals
const html = `
  <div class="card">
    <h2>${user.name}</h2>
    <p>Age: ${user.age}</p>
  </div>
`;
// Spread operator
const merged = { ...defaults, ...custom };
const allItems = [...oldItems, ...newItems];
// Optional chaining & nullish coalescing
const city = user?.address?.city ?? "Paris";
// Array methods
const names = users.map(u => u.name);
const adults = users.filter(u => u.age >= 18);
const total = prices.reduce((a, b) => a + b, 0);
// Modules
import { formatDate } from "./utils.js";
export const API_URL = "/api";
```

ES6+ Arrow Destructuring

04 Responsive & Mobile

Creez des sites adaptes a tous les ecrans

20 MOBILE CONFIG
Meta viewport et configuration

ESSENTIEL

La meta viewport est obligatoire pour le responsive. Sans elle, les mobiles affichent la version desktop en miniature. Elle indique au navigateur d'adapter la largeur a l'ecran de l'appareil.

```
<!-- Obligatoire dans le <head> -->
<meta name="viewport"
  content="width=device-width, initial-scale=1.0">
<!-- CSS de base responsive -->
html {
  font-size: 16px; /* Base pour rem */
  -webkit-text-size-adjust: 100%;
}
img, video, svg {
  max-width: 100%;
  height: auto;
}
/* Empecher le scroll horizontal */
body {
  overflow-x: hidden;
}
```

Viewport Mobile Config

MOBILE STRATEGY

21 Approche Mobile-First

ESSENTIEL

Le mobile-first signifie ecrire d'abord le CSS mobile, puis enrichir avec des media queries min-width. C'est plus performant car le mobile charge moins de CSS, et force a prioriser le contenu essentiel.

```
/* 1. Base = Mobile (pas de media query) */
.grid {
  display: flex;
  flex-direction: column;
  gap: 16px;
  padding: 16px;
}
.sidebar { display: none; }
/* 2. Tablette (>= 768px) */
@media (min-width: 768px) {
  .grid {
    flex-direction: row;
    flex-wrap: wrap;
  }
  .grid > * { flex: 1 1 45%; }
  .sidebar { display: block; }
}
/* 3. Desktop (>= 1024px) */
@media (min-width: 1024px) {
  .grid { max-width: 1200px; margin: auto; }
  .grid > * { flex: 1 1 30%; }
}
/* 4. Grand ecran (>= 1440px) */
@media (min-width: 1440px) {
  .grid { max-width: 1400px; }
}
```

Mobile-First Breakpoints Strategy

22 MOBILE BREAKPOINTS

Breakpoints et strategie responsive

IMPORTANT

Choisissez des breakpoints bases sur votre contenu, pas sur des appareils specifiques. Les valeurs courantes : 320px (petit mobile), 768px (tablette), 1024px (desktop), 1440px (grand ecran). Testez sur de vrais appareils.

```
/* Systeme de breakpoints recommande */
:root {
  --bp-sm: 640px;   /* Grand mobile */
  --bp-md: 768px;   /* Tablette */
  --bp-lg: 1024px;  /* Desktop */
  --bp-xl: 1280px;  /* Grand desktop */
  --bp-2xl: 1536px; /* Tres grand ecran */
}
/* Navigation responsive */
.nav-links { display: none; }
.hamburger { display: block; }
@media (min-width: 768px) {
  .nav-links {
    display: flex;
    gap: 24px;
  }
  .hamburger { display: none; }
}
/* Colonnes responsives */
.cols { display: grid; gap: 24px; }
@media (min-width: 640px) {
  .cols { grid-template-columns: 1fr 1fr; }
}
@media (min-width: 1024px) {
  .cols { grid-template-columns: repeat(3, 1fr); }
}
```

Breakpoints Grid Navigation

23 MOBILE IMAGES
Images responsives

IMPORTANT

Servez des images adaptees a chaque ecran avec srcset et sizes. Le navigateur choisit automatiquement la meilleure image. Utilisez picture pour les changements de format ou de cadrage (art direction).

```
<!-- srcset : le navigateur choisit -->
<img
  src="photo-800.jpg"
  srcset="photo-400.jpg 400w,
          photo-800.jpg 800w,
          photo-1200.jpg 1200w"
  sizes="(max-width: 640px) 100vw,
          (max-width: 1024px) 50vw,
          33vw"
  alt="Description de la photo"
  loading="lazy"
>
<!-- picture : art direction -->
<picture>
  <!-- Mobile : image carree -->
  <source
    media="(max-width: 640px)"
    srcset="hero-mobile.webp">
  <!-- Desktop : image paysage -->
  <source
    media="(min-width: 641px)"
    srcset="hero-desktop.webp">
  <img src="hero-fallback.jpg"
    alt="Hero banner">
</picture>
```

Srcset | Picture | Lazy Loading

24 MOBILE UX
Zones tactiles et UX mobile

ESSENTIEL

Sur mobile, les doigts sont imprecis. Apple et Google recommandent des zones tactiles de 44x44px minimum. Espacez les elements cliquables, evitez le hover comme seul indicateur d'interaction.

```
/* Zone tactile minimum */
.btn, a, input, select {
  min-height: 44px;
  min-width: 44px;
}
/* Bouton touch-friendly */
.btn {
  padding: 12px 24px;
  font-size: 16px; /* Evite le zoom iOS */
  touch-action: manipulation;
}
/* Espacement entre elements cliquables */
.nav-links a {
  padding: 12px 16px;
  display: inline-block;
}
/* Input sans zoom sur iOS */
input, select, textarea {
  font-size: 16px; /* Minimum pour iOS */
}
/* Desactiver le tap highlight */
button {
  -webkit-tap-highlight-color: transparent;
}
```

Touch | 44px | UX Mobile

MOBILE TESTING

25 Outils de test responsive

UTILE

Testez toujours sur de vrais appareils en plus des DevTools. Chrome DevTools permet de simuler des tailles d'ecran, le throttling reseau, et les evenements tactiles. Lighthouse teste la compatibilite mobile.

```
/* Test avec media queries de debug */
body::after {
  content: "MOBILE";
  position: fixed;
  bottom: 10px;
  right: 10px;
  background: red;
  color: white;
  padding: 4px 8px;
  font-size: 12px;
  z-index: 9999;
  border-radius: 4px;
}
@media (min-width: 768px) {
  body::after { content: "TABLET"; background: orange; }
}
@media (min-width: 1024px) {
  body::after { content: "DESKTOP"; background: green; }
}
/* Checklist de test :
   - Chrome DevTools (Ctrl+Shift+M)
   - Vrai iPhone + Vrai Android
   - Mode paysage
   - Lighthouse Mobile audit
   - BrowserStack pour les anciens appareils
*/
```

DevTools Testing Debug

05 Accessibilite (a11y)

Un web accessible a tous les utilisateurs

26 HTML semantique pour l'accessibilite

A11Y SEMANTIQUE

ESSENTIEL

Le HTML semantique est la base de l'accessibilite. Les lecteurs d'ecran s'appuient sur la structure HTML pour naviguer. Utilisez les bonnes balises : button pour les actions, a pour les liens, h1-h6 dans l'ordre.

```
<!-- BON : Semantique -->
<button onclick="save()">Sauvegarder</button>
<a href="/contact">Contact</a>
<nav aria-label="Menu principal">
  <ul>
    <li><a href="/" aria-current="page">Accueil</a></li>
    <li><a href="/about">A propos</a></li>
  </ul>
</nav>
<!-- MAUVAIS : Non semantique -->
<div onclick="save()">Sauvegarder</div>
<span class="link">Contact</span>
<!-- Headings dans l'ordre -->
<h1>Titre principal (1 seul par page)</h1>
  <h2>Section</h2>
    <h3>Sous-section</h3>
  <h2>Autre section</h2>
<!-- Jamais sauter de niveau (h1 > h3) -->
```

Semantique ARIA Structure

A11Y ARIA

27 ARIA : roles et attributs

IMPORTANT

ARIA (Accessible Rich Internet Applications) ajoute de la semantique quand le HTML natif ne suffit pas. Regle d'or : n'utilisez ARIA que si aucune balise HTML native ne convient. Trop d'ARIA est pire que pas assez.

```
<!-- Roles pour widgets custom -->
<div role="tablist">
  <button role="tab" aria-selected="true"
    aria-controls="panel-1">Onglet 1</button>
  <button role="tab" aria-selected="false"
    aria-controls="panel-2">Onglet 2</button>
</div>
<div role="tabpanel" id="panel-1">
  Contenu onglet 1
</div>
<!-- Live regions (contenu dynamique) -->
<div aria-live="polite" aria-atomic="true">
  <!-- Le lecteur annonce les changements -->
  3 resultats trouves
</div>
<!-- Etats interactifs -->
<button aria-expanded="false"
  aria-controls="menu">
  Menu
</button>
<nav id="menu" hidden>
  <!-- Menu deroulant -->
</nav>
```

ARIA Roles Widgets

28 A11Y COULEURS

Contraste et couleurs (WCAG)

ESSENTIEL

Le contraste insuffisant est le probleme d'accessibilite le plus courant. WCAG AA exige un ratio de 4.5:1 pour le texte normal, 3:1 pour le texte large (18px+ bold ou 24px+). Ne vehiculez jamais l'information uniquement par la couleur.

```
/* WCAG AA - Contraste minimum 4.5:1 */
.text-ok {
  color: #1a1a1a;       /* Texte sombre */
  background: #ffffff;  /* Fond clair */
  /* Ratio: 16.8:1 - Excellent */
}
/* WCAG AAA - Contraste 7:1+ */
.text-aaa {
  color: #000000;
  background: #ffffff;
  /* Ratio: 21:1 - Maximum */
}
/* Ne pas utiliser la couleur seule */
.error {
  color: #dc2626;
  /* Ajouter une icone + texte */
}
.error::before {
  content: "Erreur : ";
  /* L'info n'est pas que la couleur */
}
/* Mode contraste eleve */
@media (prefers-contrast: high) {
  :root {
    --text: #000000;
    --bg: #ffffff;
    --border: 2px solid #000000;
  }
}
```

WCAG Contraste Couleurs

29 A11Y CLAVIER

Navigation au clavier

ESSENTIEL

De nombreux utilisateurs naviguent au clavier (handicap moteur, utilisateurs avances). Tous les elements interactifs doivent etre focusables et utilisables au clavier. Ne supprimez jamais outline:none sans alternative.

```
/* Focus visible et esthetique */
:focus-visible {
  outline: 3px solid var(--primary);
  outline-offset: 2px;
  border-radius: 4px;
}
/* Masquer pour souris uniquement */
:focus:not(:focus-visible) {
  outline: none;
}
/* Skip link (premier element) */
.skip-link {
  position: absolute;
  top: -40px;
  left: 0;
  padding: 8px 16px;
  background: var(--primary);
  color: white;
  z-index: 100;
}
.skip-link:focus {
  top: 0; /* Visible au focus */
}
<!-- Dans le HTML -->
<a href="#main" class="skip-link">
  Aller au contenu principal
</a>
<!-- tabindex pour l'ordre -->
<div tabindex="0">Focusable</div>
<div tabindex="-1">Focus par JS only</div>
```

Clavier Focus Skip Link

30 A11Y LECTEURS

Lecteurs d'ecran

IMPORTANT

Les lecteurs d'ecran (NVDA, VoiceOver, JAWS) lisent le contenu a voix haute. Le texte alternatif des images, les labels des formulaires et les aria-labels sont essentiels pour une experience complete.

```
<!-- Images : alt descriptif -->
<img src="team.jpg"
  alt="Equipe de 5 developpeurs autour d'un ecran">
<!-- Image decorative : alt vide -->
<img src="divider.svg" alt="" role="presentation">
<!-- Formulaires : labels obligatoires -->
<label for="search">Rechercher</label>
<input type="search" id="search"
  aria-describedby="search-help">
<p id="search-help">
  Tapez au moins 3 caracteres
</p>
<!-- Icones avec texte cache -->
<button aria-label="Fermer la modale">
  <svg>...</svg>
</button>
<!-- Texte visible seulement par SR -->
<span class="sr-only">Ouvrir le menu</span>
/* CSS pour sr-only */
.sr-only {
  position: absolute;
  width: 1px; height: 1px;
  clip: rect(0 0 0 0);
  overflow: hidden;
}
```

Alt Text | Labels | Screen Reader

A11Y PREFERENCES

31 Preferences utilisateur

UTILE

Respectez les preferences systeme de l'utilisateur avec les media queries dediees. prefers-reduced-motion desactive les animations, prefers-color-scheme adapte le theme, prefers-contrast augmente les contrastes.

```
/* Reduire les animations si demande */
@media (prefers-reduced-motion: reduce) {
  *, *::before, *::after {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
    scroll-behavior: auto !important;
  }
}
/* Mode sombre automatique */
@media (prefers-color-scheme: dark) {
  :root {
    --bg: #0a0a0f;
    --text: #e5e7eb;
    --card-bg: #1a1a2e;
  }
}
/* Contraste eleve */
@media (prefers-contrast: high) {
  :root {
    --border-width: 2px;
    --text: #000000;
  }
}
/* Transparence reduite */
@media (prefers-reduced-transparency: reduce) {
  .glass-effect {
    backdrop-filter: none;
    background: var(--card-bg);
  }
}
```

Reduced Motion | Dark Mode | Preferences

06 SEO & Performance

Optimisez la visibilite et la vitesse de votre site

32 SEO META TAGS

Meta tags essentiels pour le SEO

ESSENTIEL

Les meta tags informent les moteurs de recherche et les reseaux sociaux sur votre contenu. Title et description apparaissent dans les resultats Google. Les Open Graph tags controlent l'apercu sur les reseaux sociaux.

```
<head>
  <!-- SEO Essentiels -->
  <title>Mon Site - Description courte (60 car.)</title>
  <meta name="description"
    content="Description de 150-160 caracteres
    qui apparait dans Google.">
  <link rel="canonical"
    href="https://monsite.fr/page">
  <!-- Open Graph (Facebook, LinkedIn) -->
  <meta property="og:title" content="Mon Site">
  <meta property="og:description"
    content="Description pour les reseaux">
  <meta property="og:image"
    content="https://monsite.fr/og-image.jpg">
  <meta property="og:url"
    content="https://monsite.fr/page">
  <!-- Twitter Card -->
  <meta name="twitter:card"
    content="summary_large_image">
  <meta name="twitter:title" content="Mon Site">
</head>
```

Meta Tags | Open Graph | Twitter

33 SEO STRUCTURE
Structure des headings (H1-H6)

ESSENTIEL

Les headings structurent votre contenu pour Google et les utilisateurs. Un seul H1 par page (le sujet principal). Les H2 sont les sections, H3 les sous-sections. Ne sautez jamais de niveaux et n'utilisez pas les headings pour le style.

```
<!-- Structure correcte -->
<h1>Guide Complet CSS Grid</h1>
  <h2>1. Introduction a Grid</h2>
    <h3>Pourquoi utiliser Grid ?</h3>
    <h3>Grid vs Flexbox</h3>
  <h2>2. Syntaxe de base</h2>
    <h3>grid-template-columns</h3>
    <h3>grid-template-rows</h3>
  <h2>3. Layouts avances</h2>
    <h3>Grid areas</h3>
    <h3>Grilles responsives</h3>
<!-- Structure INCORRECTE -->
<h1>Mon Site</h1>
<h3>Section</h3>  <!-- H2 saute ! -->
<h1>Autre titre</h1> <!-- 2 H1 ! -->
<!-- JSON-LD pour Google -->
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "Article",
  "headline": "Guide Complet CSS Grid"
}
</script>
```

Headings | Schema.org | Structure

34 SEO PERFORMANCE
Core Web Vitals (LCP, FID, CLS)

ESSENTIEL

Les Core Web Vitals sont les metriques de Google pour l'experience utilisateur. LCP (plus grand element visible < 2.5s), FID/INP (reactivite < 200ms), CLS (stabilite visuelle < 0.1). Ils impactent directement votre classement Google.

```
<!-- LCP : Optimiser le plus grand element -->
<link rel="preload" as="image"
  href="hero.webp" fetchpriority="high">
<img src="hero.webp" fetchpriority="high"
  width="1200" height="600"
  alt="Hero banner">
<!-- CLS : Reserver l'espace -->
<img src="photo.webp"
  width="800" height="600"
  alt="Photo">
<!-- Dimensions = pas de saut -->
/* CLS : Stabilite des fonts */
@font-face {
  font-family: "Custom";
  src: url("font.woff2") format("woff2");
  font-display: swap;
  /* swap = texte visible immediatement */
}
/* INP : Reactivite */
/* Eviter les long tasks (>50ms) */
/* Utiliser requestAnimationFrame */
/* Deporter le calcul avec Web Workers */
```

LCP | CLS | INP | Vitals

SEO IMAGES

35 Optimisation des images

ESSENTIEL

Les images representent 50%+ du poids des pages. Utilisez les formats modernes (WebP, AVIF), le lazy loading natif, et la compression. Une image hero de 3MB peut devenir 150KB en WebP sans perte visible de qualite.

```
<!-- Lazy loading natif -->
<img src="photo.webp"
  loading="lazy"
  decoding="async"
  width="800" height="600"
  alt="Description">
<!-- Format moderne avec fallback -->
<picture>
  <source srcset="photo.avif" type="image/avif">
  <source srcset="photo.webp" type="image/webp">
  <img src="photo.jpg" alt="Description">
</picture>
/* CSS : aspect-ratio pour eviter CLS */
.image-container {
  aspect-ratio: 16 / 9;
  overflow: hidden;
}
.image-container img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}
/* Outils de compression :
   - Squoosh (en ligne)
   - Sharp (Node.js)
   - ImageOptim (macOS) */
```

WebP AVIF Lazy Loading

SEO HINTS

36 Preload, prefetch, preconnect

IMPORTANT

Les Resource Hints indiquent au navigateur les ressources a charger en avance. Preload charge immediatement (page actuelle), prefetch charge en arriere-plan (pages futures), preconnect etablit la connexion a un domaine tiers.

```
<head>
  <!-- Preconnect : etablir la connexion -->
  <link rel="preconnect"
    href="https://fonts.googleapis.com">
  <link rel="preconnect"
    href="https://fonts.gstatic.com" crossorigin>
  <!-- Preload : charger en priorite -->
  <link rel="preload" href="font.woff2"
    as="font" type="font/woff2" crossorigin>
  <link rel="preload" href="hero.webp"
    as="image">
  <link rel="preload" href="critical.css"
    as="style">
  <!-- Prefetch : precharger pour apres -->
  <link rel="prefetch" href="/about">
  <link rel="prefetch" href="page2-data.json"
    as="fetch">
  <!-- DNS Prefetch : resoudre le DNS -->
  <link rel="dns-prefetch"
    href="https://analytics.example.com">
</head>
```

Preload Prefetch Preconnect

SEO OPTIMISATION

37 Minification et optimisation du code

IMPORTANT

Reduisez la taille de vos fichiers CSS et JavaScript en les minifiant. Combinez les fichiers pour reduire les requetes HTTP. Utilisez la compression gzip/brotli cote serveur pour des gains supplementaires de 60-80%.

```
<!-- Avant : 3 fichiers CSS (3 requetes) -->
<link rel="stylesheet" href="reset.css">
<link rel="stylesheet" href="layout.css">
<link rel="stylesheet" href="theme.css">
<!-- Apres : 1 fichier minifie -->
<link rel="stylesheet" href="styles.min.css">
<!-- Script defer et async -->
<script src="app.js" defer></script>
<!-- defer : execute apres le parsing HTML -->
<!-- async : execute des que telecharge -->
/* .htaccess : Compression Gzip/Brotli */
/* AddOutputFilterByType DEFLATE
   text/html text/css
   application/javascript */
/* Outils de build :
   - Vite (recommande)
   - esbuild (rapide)
   - Terser (JS minification)
   - cssnano (CSS minification)
   npm run build -> dist/ */
```

Minification Gzip Bundling

07 Outils & Workflow

Les outils essentiels du developpeur web

38 OUTILS EDITEUR
VS Code - Editeur et extensions

ESSENTIEL

Visual Studio Code est l'editeur le plus populaire pour le web. Gratuit, leger et extensible, il offre la coloration syntaxique, l'autocompletion, le terminal integre et un ecosysteme d'extensions riche.

```
// Extensions essentielles VS Code
// 1. Live Server - Serveur local avec hot reload
// 2. Prettier - Formatage automatique
// 3. ESLint - Detection d'erreurs JS
// 4. Auto Rename Tag - Renomme les balises
// 5. CSS Peek - Navigation CSS rapide
// 6. GitLens - Historique git avance
// Raccourcis essentiels :
// Ctrl+P        : Ouvrir un fichier
// Ctrl+Shift+P  : Palette de commandes
// Ctrl+D        : Selectionner occurrence
// Alt+Haut/Bas  : Deplacer une ligne
// Ctrl+/        : Commenter/Decommenter
// Ctrl+`        : Terminal integre
// settings.json recommande :
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "emmet.triggerExpansionOnTab": true
}
```

VS Code | Extensions | Productivite

39 OUTILS GIT

Git - Versionner son code

ESSENTIEL

Git est indispensable pour tout developpeur. Il versionne votre code, permet de collaborer et de revenir en arriere. Apprenez ces commandes de base et utilisez des branches pour chaque nouvelle fonctionnalite.

```
# Initialiser un projet
git init
git add .
git commit -m "Premier commit"
# Workflow quotidien
git status              # Voir les changements
git add index.html      # Ajouter un fichier
git add .               # Ajouter tout
git commit -m "Ajout navbar responsive"
git log --oneline       # Historique
# Branches (une par feature)
git branch feature/navbar  # Creer
git checkout feature/navbar # Basculer
git checkout -b fix/bug    # Creer + basculer
git merge feature/navbar   # Fusionner
# Remote (GitHub)
git remote add origin URL
git push -u origin main
git pull origin main
# Annuler
git checkout -- fichier.html  # Annuler modif
git reset HEAD fichier.html   # Desindexer
```

Git Versionning Branches

40 OUTILS DEVTOOLS

DevTools du navigateur

ESSENTIEL

Les DevTools (F12 ou Cmd+Option+I) sont l'outil de debug le plus puissant. L'onglet Elements inspecte le DOM/CSS, Console debug le JS, Network analyse les requetes, et Lighthouse audite la performance.

```
// Onglet Console - Debug JavaScript
console.log("Variable:", myVar);
console.table(users);  // Affiche un tableau
console.time("fetch");
await fetch("/api");
console.timeEnd("fetch"); // Temps ecoule
// Onglet Elements
// - Clic droit > Inspecter sur un element
// - Modifier le CSS en temps reel
// - Voir le box model (margin/padding)
// - Forcer les etats (:hover, :focus)
// Onglet Network
// - Voir toutes les requetes HTTP
// - Taille des fichiers transferes
// - Temps de chargement par ressource
// - Simuler connexion lente (3G)
// Onglet Lighthouse
// - Performance (score sur 100)
// - Accessibilite
// - Bonnes pratiques
// - SEO
// - PWA
```

DevTools Console Debug

OUTILS DEPLOY

41 Deploiement (GitHub Pages, Netlify, Vercel)

IMPORTANT

Mettez votre site en ligne gratuitement. GitHub Pages pour les sites statiques, Netlify et Vercel pour plus de fonctionnalites (formulaires, serverless, previews). Le deploiement continu publie automatiquement a chaque push.

```
# GitHub Pages - Le plus simple
# 1. Creer un repo GitHub
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/user/site
git push -u origin main
# 2. Settings > Pages > Source: main > /root
# URL: https://user.github.io/site
# Netlify - Plus de features
# 1. Connecter le repo GitHub
# 2. Build command: (vide si HTML pur)
# 3. Publish directory: / ou dist/
# 4. Deploy automatique a chaque push
# Vercel - Ideal pour frameworks
# 1. npm i -g vercel
# 2. vercel
# 3. Suivre les instructions
# 4. Deploiement preview par branche
# Domaine personnalise
# 1. Acheter un domaine (Namecheap, OVH)
# 2. DNS : CNAME -> site.netlify.app
# 3. HTTPS gratuit automatique
```

Deploy | GitHub Pages | Netlify

OUTILS HOSTING

42 Domaine et hebergement

UTILE

Un domaine personnalise (.fr, .com) donne une identite professionnelle. Les hebergeurs modernes offrent HTTPS gratuit, CDN, et deploiement continu. Pour un site statique, les hebergeurs gratuits suffisent largement.

```
# Structure de prix typique
# Domaine .fr : ~8 EUR/an (OVH, Gandi)
# Domaine .com : ~12 EUR/an
# Hebergement statique : GRATUIT
#   (GitHub Pages, Netlify, Vercel)
# Hebergement avec backend :
#   VPS : ~5 EUR/mois (OVH, DigitalOcean)
#   Mutualise : ~3 EUR/mois (o2switch)
# Configuration DNS basique
# Type A     : @ -> 185.199.108.153
# Type CNAME : www -> user.github.io
# Type MX    : @ -> mail.provider.com
# Fichier _redirects (Netlify)
/old-page  /new-page  301
/*         /index.html 200
# HTTPS obligatoire en 2026
# - Let's Encrypt (gratuit)
# - Netlify/Vercel = auto
# - .htaccess: RewriteRule ^(.*)$
#   https://%{HTTP_HOST}/$1 [R=301,L]
```

Domaine | DNS | Hebergement

08 Securite Web

Protegez votre site et vos utilisateurs

43 HTTPS - Chiffrement obligatoire

SECURITE HTTPS

ESSENTIEL

HTTPS chiffre les communications entre le navigateur et le serveur. C'est obligatoire en 2026 : Google penalise les sites HTTP, les navigateurs affichent des avertissements, et les APIs modernes exigent un contexte securise.

```
<!-- Forcer HTTPS -->
<meta http-equiv="Content-Security-Policy"
  content="upgrade-insecure-requests">
<!-- .htaccess (Apache) -->
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$
  https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
<!-- Headers HSTS (serveur) -->
Strict-Transport-Security:
  max-age=63072000;
  includeSubDomains;
  preload
/* Certificat SSL gratuit :
   - Let's Encrypt (certbot)
   - Netlify/Vercel = automatique
   - Cloudflare = proxy gratuit
   Verifier : https://www.ssllabs.com
   Le cadenas dans la barre d'URL
   confirme que HTTPS est actif */
```

HTTPS SSL HSTS

SECURITE CSP

44 Content Security Policy (CSP)

IMPORTANT

La CSP est un header HTTP qui controle quelles ressources le navigateur peut charger. Elle bloque les scripts inline malveillants, limite les origines des ressources et previent la majorite des attaques XSS.

```
<!-- CSP via meta tag (basique) -->
<meta http-equiv="Content-Security-Policy"
  content="
    default-src 'self';
    script-src 'self' https://cdn.trusted.com;
    style-src 'self' 'unsafe-inline';
    img-src 'self' data: https:;
    font-src 'self' https://fonts.gstatic.com;
    connect-src 'self' https://api.monsite.fr;
    frame-src 'none';
">
<!-- CSP via header serveur (recommande) -->
# Content-Security-Policy:
#   default-src 'self';
#   script-src 'self';
#   object-src 'none';
#   base-uri 'self';
#   form-action 'self';
/* Directives importantes :
   default-src : fallback
   script-src  : JavaScript
   style-src   : CSS
   img-src     : Images
   frame-src   : iframes */
```

CSP Headers XSS

SECURITE XSS

45 Prevention des attaques XSS

ESSENTIEL

Le XSS (Cross-Site Scripting) injecte du code malveillant dans votre page. Echappez toujours le contenu utilisateur, n'utilisez jamais innerHTML avec des donnees non fiables, et activez la CSP.

```
// DANGEREUX : injection XSS possible
element.innerHTML = userInput;
document.write(userInput);
// SECURISE : echapper le contenu
element.textContent = userInput;
// Fonction d'echappement HTML
function escapeHTML(str) {
  const div = document.createElement("div");
  div.textContent = str;
  return div.innerHTML;
}
// Utilisation securisee
const safe = escapeHTML(userInput);
element.innerHTML = `<p>${safe}</p>`;
// Sanitize les URLs
function safeURL(url) {
  try {
    const parsed = new URL(url);
    if (["http:", "https:"].includes(parsed.protocol)) {
      return parsed.href;
    }
  } catch {}
  return "#"; // URL invalide
}
// Template literals securises
const html = `<div>${escapeHTML(name)}</div>`;
```

XSS Echappement Sanitize

SECURITE CSRF

46 Protection CSRF

IMPORTANT

Le CSRF (Cross-Site Request Forgery) force un utilisateur authentifie a executer des actions non voulues. Protegez vos formulaires avec des tokens CSRF, l'attribut SameSite sur les cookies, et la verification du header Origin.

```
<!-- Token CSRF dans les formulaires -->
<form method="POST" action="/api/update">
  <input type="hidden" name="csrf_token"
    value="abc123randomToken456">
  <input type="text" name="email">
  <button type="submit">Mettre a jour</button>
</form>
// Verification cote serveur (Node.js)
app.post("/api/update", (req, res) => {
  const token = req.body.csrf_token;
  if (token !== req.session.csrfToken) {
    return res.status(403).json({
      error: "Token CSRF invalide"
    });
  }
  // Traiter la requete...
});
// Cookie SameSite (protection navigateur)
// Set-Cookie: session=abc123;
//   SameSite=Strict;
//   Secure;
//   HttpOnly
// Verifier le header Origin
const origin = req.headers.origin;
if (origin !== "https://monsite.fr") {
  return res.status(403).send("Refuse");
}
```

CSRF Tokens Cookies

47 SECURITE INPUT

Sanitization des inputs

ESSENTIEL

Ne faites jamais confiance aux donnees utilisateur. Validez cote client ET cote serveur. Sanitizez les inputs pour prevenir les injections SQL, XSS et autres attaques. Le front-end valide pour l'UX, le back-end pour la securite.

```
<!-- Validation HTML5 (front-end) -->
<input type="email" required
  pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$"
  maxlength="254">
<input type="number" min="0" max="100" step="1">
<input type="url" required>
// Validation JavaScript
function validateEmail(email) {
  const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return re.test(email);
}
function sanitizeInput(str) {
  return str
    .trim()
    .replace(/[<>]/g, "") // Supprimer < >
    .slice(0, 500);       // Limiter longueur
}
// Validation serveur (toujours !)
if (!validateEmail(req.body.email)) {
  return res.status(400).json({
    error: "Email invalide"
  });
}
// Requetes SQL parametrees (anti-injection)
db.query(
  "SELECT * FROM users WHERE email = ?",
  [email] // Parametre echappe
);
```

Validation Sanitize Injection

48 SECURITE HEADERS

Headers de securite

IMPORTANT

Les headers HTTP de securite ajoutent des couches de protection cote navigateur. Ils previennent le clickjacking, le MIME sniffing, et controlent les informations envoyees aux sites tiers. Testez avec securityheaders.com.

```
<!-- Headers de securite (serveur) -->
# .htaccess ou configuration serveur
# Anti clickjacking
X-Frame-Options: DENY
# Anti MIME sniffing
X-Content-Type-Options: nosniff
# Controle referrer
Referrer-Policy: strict-origin-when-cross-origin
# Permissions navigateur
Permissions-Policy:
  camera=(),
  microphone=(),
  geolocation=(self)
# HSTS (forcer HTTPS)
Strict-Transport-Security:
  max-age=31536000; includeSubDomains
<!-- Netlify : fichier _headers -->
/*
  X-Frame-Options: DENY
  X-Content-Type-Options: nosniff
  Referrer-Policy: strict-origin-when-cross-origin
  Permissions-Policy: camera=(), microphone=()
  Content-Security-Policy: default-src 'self'
/* Tester : securityheaders.com
   Objectif : note A ou A+ */
```

Headers X-Frame Permissions

Checklist recapitulative

Verifiez ces points essentiels avant de publier votre site

HTML & Structure

- [ ] DOCTYPE et meta viewport
- [ ] Balises semantiques (header, main, footer)
- [ ] Alt text sur toutes les images
- [ ] Labels sur tous les inputs
- [ ] Liens avec rel='noopener'
- [ ] Structure h1-h6 logique

CSS & Design

- [ ] box-sizing: border-box global
- [ ] Variables CSS centralisees
- [ ] Mobile-first media queries
- [ ] Flexbox/Grid pour les layouts
- [ ] Animations GPU-accelerees
- [ ] clamp() pour la typographie fluide

JavaScript

- [ ] const/let (jamais var)
- [ ] addEventListener (pas onclick)
- [ ] async/await avec try/catch
- [ ] Echappement du contenu utilisateur
- [ ] Delegation d'evenements
- [ ] Modules ES6 (import/export)

Responsive & Mobile

- [ ] Meta viewport configuree
- [ ] Zones tactiles >= 44px
- [ ] font-size >= 16px sur inputs
- [ ] Test sur vrais appareils
- [ ] Pas de scroll horizontal
- [ ] Images responsives (srcset)

Accessibilite (a11y)

- [ ] Contraste WCAG AA (4.5:1)
- [ ] Navigation clavier complete
- [ ] Skip link vers le contenu
- [ ] ARIA uniquement si necessaire
- [ ] prefers-reduced-motion respecte
- [ ] Test lecteur d'ecran

SEO & Performance

- [ ] Title + meta description uniques
- [ ] Open Graph tags (og:image, etc.)
- [ ] Core Web Vitals OK
- [ ] Images WebP + lazy loading
- [ ] Preload ressources critiques
- [ ] HTTPS + headers de securite

Merci d'avoir telecharge ce guide !

Vous avez maintenant toutes les bases pour creer
des sites web professionnels et performants.

HTML	Structure semantique et accessible
CSS	Flexbox, Grid, variables et animations
JavaScript	DOM, events, fetch et ES6+
Et plus	Responsive, a11y, SEO et securite

Decouvrir Effects.lab

742 effets | 26 templates | 20 categories