Guida completa a CSS e Bootstrap

Stefanoo94

Contents

Guida completa e super dettagliata a CSS e Bootstrap

Versione: 1.0
Autore: Stefanoo94

Scopo

Questa guida è pensata per darti una panoramica dettagliata e pratica su CSS moderno e su Bootstrap (v5+): principi fondamentali, layout avanzati, responsive design, best practice, architetture CSS, strumenti di build, prestazioni, accessibilità e come estendere/customizzare Bootstrap con SASS. Include esempi, snippet e suggerimenti operativi per progetti reali.

Indice

  - Media queries moderne
  - Fluid typography e layout (clamp, vw)
- Layout avanzato e pattern
  - Sticky e fixed positioning
  - Multi-column layout
  - CSS Shapes, object-fit, object-position
- Styling avanzato
  - Transitions, animations (keyframes)
  - Filters, backdrop-filter
  - Blend modes
- Architetture CSS e organizzazione del codice
  - BEM
  - SMACSS
  - ITCSS
  - Atomic/Utility-first (Tailwind-like) e quando usarle
- Preprocessori e strumenti
  - Sass/SCSS (parti utili)
  - PostCSS, Autoprefixer, PurgeCSS / UnCSS
  - CSS Modules e CSS-in-JS panoramica
- Performance e distribuzione
  - Minimizzazione, critical CSS, lazy loading
  - Caching e CDN
- Accessibilità e best practice
  - Contrast, focus states, prefers-reduced-motion
  - Form accessibility
- Bootstrap (v5+) approfondito
  - Installazione (CDN, npm, source)
  - Struttura del progetto Bootstrap (SASS files)
  - Grid system dettagliato
  - Utilities API e classi utilitarie
  - Componenti principali: Navbar, Buttons, Forms, Cards, Modals, Tooltips, Popovers, Offcanvas
  - Theming e customizzazione con SASS
  - RTL support, responsive utilities, layout utilities
  - Estendere Bootstrap: plugin, override, mixins
- Testing e manutenzione
  - Visual regression testing, linters, stylelint
  - Documentazione e pattern library (Storybook)
- Esempi pratici e checklist prima della messa in produzione
- Risorse utili

Fondamenti CSS

Sintassi e regole base

CSS è composto da regole: selettore { proprietà: valore; }. Esempio:

```
.button {
  background-color: #06f;
  color: #fff;
  padding: .5rem 1rem;
}
```

Selettori

- Selettori di tipo, ID, classi: `div`, `#id`, `.class`
- Combinatori: `E F` (discendente), `E > F` (figlio diretto), `E + F` (adjacent sibling), `E ~ F` (general sibling)
- Pseudo-classi e pseudo-elementi: `:hover`, `:focus`, `:nth-child()`, `::before`, `::after`
- Attributi: `[attr]`, `[attr="value"]`, `[attr^="start"]`

Specificità e ordine di origine

Specificità determina quale regola vince: - Inline style (1000) - ID selectors (100) - Class/attribute/pseudo-class (10) - Type/pseudo-element (1) ! Important: evita l'abuso di !important — crea debito tecnico.

Box model

- Contenuto, padding, border, margin.
- box-sizing: content-box (default) vs border-box (consigliato per layout prevedibili).

```
* { box-sizing: border-box; }
```

Unità e valori

- Absolute: px
- Relative: em (relativo a font-size dell'elemento), rem (root font-size)
- Viewport: vw, vh
- Moderni/utility: `ch`, `%`, `vmin`, `vmax` Consiglio: usare rem per spaziature coerenti e accessibilità.

CSS moderno

Custom properties (variabili CSS)

Variabili nativamente supportate:

```
:root {
  --primary: #0d6efd;
  --gap: 1rem;
}
.header { background: var(--primary); padding: var(--gap); }
```

- Pro: runtime, cambiano con media query o JS.
- Con: non sostituiscono tutte le esigenze dei preprocessori (calcoli build-time).

Funzioni utili

- calc(): combinazioni dinamiche: `width: calc(100% - 2rem);`
- clamp(): utile per fluid typography: `font-size: clamp(1rem, 2.5vw, 1.5rem);`
- min()/max(): fornisce limiti moderni.

Flexbox

- Container: `display: flex; flex-direction; justify-content; align-items; gap; flex-wrap;`
- Item: `flex: 1 1 auto` (grow, shrink, basis) Esempio:

```
.header { display: flex; justify-content: space-between; align-items: center; gap: 1rem; }
```

CSS Grid

- Potente per layout bidimensionali:

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 1rem;
}
@media (max-width: 768px) {
  .grid { grid-template-columns: 1fr; }
}
```

- Subgrid (se supportato) e auto-placement (`grid-auto-flow`).

Logical properties

Supportano writing-mode e internationalization: - `margin-inline-start`, `padding-block-end`, `inset-inline-start`, ecc. Preferire per progetti con supporto RTL.

Containment e isolation

`contain: layout style paint;` può aiutare performance in componenti isolati. `isolation: isolate;` evita blend di z-index tra stacking contexts.

Responsive design

Mobile-first

Dichiara stili base per mobile, poi usa media queries min-width per breakpoints. Esempio breakpoints comuni: - xs: 0 - sm: 576px - md: 768px - lg: 992px - xl: 1200px

Media queries moderne

```
@media (min-width: 768px) { ... }
@media (prefers-reduced-motion: reduce) { ... }
@media (min-resolution: 2dppx) { ... }
```

Fluid layout e typography

Usa `clamp()` e `vw` per testi e larghezze fluide:

```
h1 { font-size: clamp(1.5rem, 3vw, 2.5rem); }
```

Layout avanzato e pattern

Sticky & fixed

- `position: sticky; top: 0;` per header che rimangono nel flusso ma fermi in viewport
- `position: fixed;` per elementi fuori dal flusso

Multi-column

- `column-count`, `column-gap` per layout tipo magazine.

object-fit / object-position

Per immagini e media:

```
img.cover { width: 100%; height: 200px; object-fit: cover; object-position: center; }
```

Styling avanzato

Transitions e animations

Transitions per cambi di stato:

```
.btn { transition: background-color .2s ease, transform .15s ease; }
.btn:hover { transform: translateY(-2px); }
```

Animations:

```
@keyframes fadeIn { from { opacity: 0 } to { opacity: 1 } }
.modal { animation: fadeIn .3s ease forwards; }
```

Preferisci `prefers-reduced-motion` per rispettare utenti sensibili.

Filters e backdrop-filter

Esempio blur:

```
.modal-backdrop { backdrop-filter: blur(6px); }
```

Architetture CSS e organizzazione

BEM (Block Element Modifier)

- Convenzione: `.block`, `.block__element`, `.block--modifier`
- Vantaggio: leggibilità e isolamento semantico.

SMACSS

- Organizza in categorie: Base, Layout, Module, State, Theme.

ITCSS

- From generic to specific: Settings → Tools → Generic → Elements → Objects → Components → Utilities.

Scegli uno schema e mantienilo coerente nel team.

Preprocessori e strumenti

Sass / SCSS

- Variabili, mixin, nesting, partials, import. Esempio:

```
$primary: #0d6efd;
@function rem($px) { @return #{$px/16}rem; }
.button { padding: rem(12); }
```

- Usa partials `_variables.scss`, `_mixins.scss`, `_components.scss`.

PostCSS e Autoprefixer

- Autoprefixer aggiunge vendor prefixes a build-time.
- PostCSS plugin ecosystem per minificazione, cssnano, etc.

PurgeCSS / UnCSS

Rimuove CSS non usato per ridurre bundle size. Attenzione a classi dinamiche (utilities generate at runtime).

CSS Modules / CSS-in-JS

- CSS Modules: isolamento a livello di file (class name hashing).
- CSS-in-JS (styled-components, emotion): utile in component-driven stacks React, ma valuta runtime cost.

Performance e deploy

Critical CSS

- Estrai e inietta il CSS critico per above-the-fold per ridurre First Contentful Paint.

Minificazione e caching

- Usa gzip o brotli. Imposta long-cache headers per assets fingerprinted.

Caricamento condizionale

- Lazy-load CSS per parti non-critical (es. admin panel styles).
- Usa preload/prefetch per font e CSS critico:

```
<link rel="preload" href="/styles/critical.css" as="style">
<link rel="stylesheet" href="/styles/main.css">
```

Accessibilità (a11y) e best practice

Contrast e leggibilità

- WCAG contrast ratio >= 4.5:1 per testo normale.
- Usa strumenti come Lighthouse, axe-core.

Focus states

- Fornisci focus styles visibili:

```
:focus { outline: 3px solid #0d6efd; outline-offset: 2px; }
```

prefers-reduced-motion

Rispetta:

```
@media (prefers-reduced-motion: reduce) {
  * { animation: none !important; transition: none !important; }
}
```

Form accessibility

- label associati a input, aria-describedby per messaggi di errore, role e aria-live per aggiornamenti dinamici.

Bootstrap (v5+) approfondito

Introduzione e installazione

Tre modi principali: 1. CDN (veloce, non custom)

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.x/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.x/dist/js/bootstrap.bundle.min.js"></script>
```

2. NPM (per custom SASS)

```
npm install bootstrap
```

3. Scaricare source e compilare (consigliato per theming)

SASS source e struttura

Bootstrap fornisce file SASS modulari: - `_variables.scss` — tutte le variabili per colori, spacings, breakpoints - `_mixins.scss` — mixin utili - componenti in `_components/` e utilities in `_utilities/`

Per customizzare, crea un file `custom.scss` che importa le parti necessarie e sovrascrive variabili prima di `@import "bootstrap";`:

```
// custom.scss
$primary: #ff5722;
@import "node_modules/bootstrap/scss/bootstrap";
```

Grid system (dettagli)

- 12 colonne, classi `.container`, `.row`, `.col`, `.col-md-6`, ecc.
- Breakpoints: `sm`, `md`, `lg`, `xl`, `xxl`.
- Column ordering: `.order-md-1`
- Auto-layout columns: `.col` con larghezza dinamica
- Gutter spacing: `g-0` … `g-5` o utilities `gx-3`, `gy-2`.

Esempio:

```
<div class="container">
  <div class="row">
    <div class="col-md-4">Col1</div>
    <div class="col-md-8">Col2</div>
  </div>
</div>
```

Utilities API e classi utilitarie

Bootstrap 5 introduce Utilities API per generare classi utilitarie. Utile per: - Spacing: `m-`, `p-` (es. `mb-3`, `px-2`) - Display: `d-flex`, `d-none`, `d-md-block` - Flex utilities: `flex-row`, `justify-content-between` - Sizing: `w-100`, `h-50` - Colors: `text-primary`, `bg-success` - Responsive utilities: aggiungi breakpoint: `d-md-none`, `p-lg-4`

Puoi abilitare/creare nuove utilities tramite SASS config.

Componenti principali

- Buttons
  - `.btn`, `.btn-primary`, `.btn-outline-secondary`
  - Sizing: `.btn-sm`, `.btn-lg`
- Navbar
  - Responsive collapse con `.navbar-expand-md` e `.navbar-toggler`
- Forms
  - `.form-control`, `.form-group`, validation styles `.is-invalid`, `.invalid-feedback`
  - Floating labels `.form-floating`
- Cards
  - `.card`, `.card-body`, `.card-title`
- Modals
  - JavaScript initialization via `bootstrap.Modal` o data attributes
- Tooltips / Popovers
  - Richiesto Popper (in bundle)
  - Inizializzazione JS: `new bootstrap.Tooltip(element)`
- Offcanvas
  - Drawer moderno con markup semplice
- Toasts
  - Piccoli messaggi transitori

Esempio: modal

```
<button data-bs-toggle="modal" data-bs-target="#myModal">Apri</button>
<div class="modal fade" id="myModal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">...</div>
  </div>
</div>
```

Theming e customizzazione con SASS

- Sovrascrivi variabili in `_custom.scss` prima di importare `bootstrap.scss`.
- Variabili comuni: `$primary`, `$secondary`, `$body-bg`, `$body-color`, `$spacers`, `$font-family-base`.
- Mappe di colori e palette: puoi ridefinire `$theme-colors`.

Esempio minimale:

```
$primary: #0077cc;
$body-bg: #f8f9fa;
@import "bootstrap";
```

RTL support

Bootstrap fornisce build RTL. Quando necessario, costruisci o includi la versione RTL.

Estendere Bootstrap

- Evita di sovrascrivere troppe classi; preferisci estendere e creare componenti custom che usano utilities.
- Crea mixins custom che riutilizzano variabili Bootstrap.
- Per plugin JS custom, usa le convenzioni di data attributes o inizializza manualmente con l'API JS.

Performance con Bootstrap

- Importare solo i partials necessari per ridurre CSS.
- Usa PurgeCSS per rimuovere classi non usate (attenzione a classi dinamiche come `d-${breakpoint}-...`).

- Tree-shake JS: importa solo i moduli JS che servono (es. modal, tooltip).

Testing e manutenzione

Visual regression testing

- Percy, Chromatic, BackstopJS per catturare regressioni visive.

Linting & stylelint

- Configura stylelint con regole adatte (no deep nesting, no id selectors se policy)
- Esempio .stylelintrc:

```
{
  "extends": "stylelint-config-standard",
  "rules": {
    "max-nesting-depth": 3
  }
}
```

Storybook / Pattern Library

- Documenta componenti con Storybook per component-driven development.

Esempi pratici e snippet utili

Layout tipico header + hero + grid responsive

```
<header class="bg-light py-3">
  <div class="container d-flex justify-content-between align-items-center">
    <a class="navbar-brand" href="#">Brand</a>
    <nav class="d-none d-md-flex gap-3">
      <a href="#" class="text-decoration-none">Home</a>
      <a href="#" class="text-decoration-none">Docs</a>
    </nav>
    <button class="btn btn-outline-primary d-md-none">Menu</button>
  </div>
</header>

<section class="hero py-5">
  <div class="container">
    <div class="row align-items-center">
      <div class="col-md-6">
        <h1 class="display-4">Titolo</h1>
        <p class="lead">Sottotitolo significativo...</p>
        <a class="btn btn-primary btn-lg" href="#">Inizia</a>
      </div>
      <div class="col-md-6">
        <img src="/hero.png" class="img-fluid" alt="">
      </div>
    </div>
  </div>
</section>
```

Form con validazione client

```
<form id="login" novalidate>
  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="email" id="email" class="form-control" required>
    <div class="invalid-feedback">Inserisci una email valida</div>
  </div>
  <button class="btn btn-primary" type="submit">Accedi</button>
</form>

<script>
document.getElementById('login').addEventListener('submit', function(e) {
  if (!this.checkValidity()) {
    e.preventDefault();
    e.stopPropagation();
    this.classList.add('was-validated');
  }
});
</script>
```

Checklist prima della messa in produzione

- ☐ CSS minificato e gzip/brotli abilitati
- ☐ Critical CSS iniettato per above-the-fold
- ☐ Font ottimizzati, preload per font principali
- ☐ PurgeCSS configurato (attenzione a classi dinamiche)
- ☐ Test contrasto e accessibility checks (axe, Lighthouse)
- ☐ Visual regression test baselines aggiornate
- ☐ Documentazione componenti aggiornata (Storybook)
- ☐ Variabili e temi centralizzati per manutenzione

Risorse utili

- MDN CSS: https://developer.mozilla.org
- CSS-Tricks: https://css-tricks.com
- Bootstrap docs: https://getbootstrap.com
- A11y resources: https://a11yproject.com, https://deque.com
- Performance & web vitals: https://web.dev/vitals

Come estendere questa guida

Se vuoi, posso: - Generare versioni separate (es. cheat-sheet PDF, snippets directory con file HTML/CSS demo) - Fornire un file SCSS di base preconfezionato che importa solo i partials di Bootstrap che ti servono - Creare GitHub Action per generare automaticamente i PDF ad ogni push - Preparare un esempio completo (repo scaffold) con build pipeline (Vite/Webpack), linters e Storybook

Dimmi quale di queste preferisci e preparo i file o gli script.