CSS e Tailwind CSS: Guida Completa

Fondamenti di CSS

Selettori CSS

css

```css
/* Selettore di elemento */
h1 {
  color: blue;
}

/* Selettore di classe */
.container {
  max-width: 1200px;
  margin: 0 auto;
}

/* Selettore di ID */
#header {
  background-color: #f5f5f5;
}

/* Selettore di attributo */
input[type="text"] {
  border: 1px solid #ccc;
}

/* Selettori combinati */
.card h2 {
  font-size: 18px;
}

/* Pseudo-classi */
a:hover {
  text-decoration: underline;
}

button:focus {
  outline: 2px solid blue;
}

/* Pseudo-elementi */
p::first-letter {
  font-size: 2em;
}

li::before {
  content: "→ ";
}
```

Box Model

- **Content**: L'area dove viene visualizzato il contenuto
- **Padding**: Lo spazio tra il contenuto e il bordo
- **Border**: La linea che circonda il padding
- **Margin**: Lo spazio esterno che separa l'elemento dagli altri

```css
.box {
  width: 300px;
  height: 200px;
  padding: 20px;
  border: 2px solid black;
  margin: 30px;
  /* box-sizing: border-box; mantiene la width/height includendo padding e border */
  box-sizing: border-box;
}
```

Positioning

```css
/* Static (default) */
.default {
  position: static;
}

/* Relative */
.relative {
  position: relative;
  top: 10px;
  left: 20px;
}

/* Absolute */
.absolute {
  position: absolute;
  top: 0;
  right: 0;
}

/* Fixed */
.fixed-header {
  position: fixed;
  top: 0;
  width: 100%;
}

/* Sticky */
.sticky-nav {
  position: sticky;
  top: 0;
}
```

Flexbox

css

```css
.container {
  display: flex;
  flex-direction: row; /* row | column | row-reverse | column-reverse */
  flex-wrap: wrap; /* nowrap | wrap | wrap-reverse */
  justify-content: space-between; /* flex-start | flex-end | center | space-between |
  align-items: center; /* flex-start | flex-end | center | baseline | stretch */
  gap: 20px; /* spazio tra gli elementi flex */
}

.item {
  flex: 1; /* shorthand per flex-grow, flex-shrink, flex-basis */
  align-self: flex-start; /* sovrascrive align-items per questo elemento */
}
```

CSS Grid

css

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* 3 colonne di uguale larghezza */
  grid-template-rows: auto 200px auto; /* Righe con altezze diverse */
  grid-gap: 20px; /* Spazio tra celle */
}

.grid-item {
  grid-column: 1 / 3; /* Occupa colonne dalla 1 alla 3 */
  grid-row: 2 / 3; /* Occupa la riga 2 */
}

/* Area template */
.dashboard {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main { grid-area: main; }
.footer { grid-area: footer; }
```

Media Queries

```css
/* Mobile first approach */
.container {
  width: 100%;
  padding: 10px;
}

/* Tablet */
@media (min-width: 768px) {
  .container {
    width: 750px;
    padding: 20px;
  }
}

/* Desktop */
@media (min-width: 1024px) {
  .container {
    width: 970px;
  }
}

/* Large Desktop */
@media (min-width: 1200px) {
  .container {
    width: 1170px;
  }
}
```

Animazioni CSS

```css
/* Transition */
.button {
  background-color: blue;
  color: white;
  padding: 10px 20px;
  transition: background-color 0.3s ease, transform 0.2s ease;
}

.button:hover {
  background-color: darkblue;
  transform: scale(1.05);
}

/* Animation */
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.animate-fadeIn {
  animation: fadeIn 0.5s ease forwards;
}
```

CSS Variables (Custom Properties)

```css
:root {
  --primary-color: #3498db;
  --secondary-color: #2ecc71;
  --font-size-base: 16px;
  --spacing-unit: 8px;
}

.header {
  background-color: var(--primary-color);
  padding: calc(var(--spacing-unit) * 2);
}

.button {
  background-color: var(--secondary-color);
  font-size: var(--font-size-base);
  padding: var(--spacing-unit) calc(var(--spacing-unit) * 2);
}
```

Tailwind CSS

Concetti Fondamentali

Tailwind è un framework CSS utility-first che permette di costruire design personalizzati senza lasciare il file HTML. Invece di scrivere CSS tradizionale, applichi classi predefinite direttamente ai tuoi elementi HTML.

Setup Tailwind CSS

bash

```bash
# Installazione con npm
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

# Configurazione (tailwind.config.js)
module.exports = {
  content: ["./src/**/*.{html,js,jsx,ts,tsx}"],
  theme: {
    extend: {},
  },
  plugins: [],
}

# Importazione in CSS
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Utility Classes

Typography

```html
<!-- Font Size -->
<p class="text-xs">Extra Small</p>
<p class="text-sm">Small</p>
<p class="text-base">Base</p>
<p class="text-lg">Large</p>
<p class="text-xl">Extra Large</p>
<p class="text-2xl">2XL</p>

<!-- Font Weight -->
<p class="font-thin">Thin</p>
<p class="font-normal">Normal</p>
<p class="font-medium">Medium</p>
<p class="font-bold">Bold</p>

<!-- Text Alignment -->
<p class="text-left">Left</p>
<p class="text-center">Center</p>
<p class="text-right">Right</p>

<!-- Text Color -->
<p class="text-red-500">Red Text</p>
<p class="text-blue-700">Blue Text</p>
```

Spacing

```html
<!-- Margin -->
<div class="m-4">Margin all sides</div>
<div class="mt-4">Margin top</div>
<div class="mr-4">Margin right</div>
<div class="mb-4">Margin bottom</div>
<div class="ml-4">Margin left</div>
<div class="mx-4">Margin horizontal</div>
<div class="my-4">Margin vertical</div>

<!-- Padding -->
<div class="p-4">Padding all sides</div>
<div class="pt-4">Padding top</div>
<div class="pr-4">Padding right</div>
<div class="pb-4">Padding bottom</div>
<div class="pl-4">Padding left</div>
<div class="px-4">Padding horizontal</div>
<div class="py-4">Padding vertical</div>
```

Sizing

```html
<!-- Width -->
<div class="w-full">Full width</div>
<div class="w-1/2">50% width</div>
<div class="w-1/3">33.33% width</div>
<div class="w-1/4">25% width</div>
<div class="w-auto">Auto width</div>
<div class="w-screen">Screen width</div>
<div class="w-96">24rem width</div>

<!-- Height -->
<div class="h-full">Full height</div>
<div class="h-screen">Screen height</div>
<div class="h-64">16rem height</div>
<div class="h-auto">Auto height</div>
```

Flexbox

```html
<!-- Container -->
<div class="flex">Flex container</div>
<div class="flex flex-row">Row (default)</div>
<div class="flex flex-col">Column</div>
<div class="flex flex-wrap">Wrap</div>

<!-- Justify Content -->
<div class="flex justify-start">Start</div>
<div class="flex justify-center">Center</div>
<div class="flex justify-end">End</div>
<div class="flex justify-between">Space Between</div>
<div class="flex justify-around">Space Around</div>
<div class="flex justify-evenly">Space Evenly</div>

<!-- Align Items -->
<div class="flex items-start">Start</div>
<div class="flex items-center">Center</div>
<div class="flex items-end">End</div>
<div class="flex items-stretch">Stretch</div>
<div class="flex items-baseline">Baseline</div>

<!-- Gap -->
<div class="flex gap-4">Gap on all sides</div>
<div class="flex gap-x-4">Horizontal gap</div>
<div class="flex gap-y-4">Vertical gap</div>
```

Grid

```html
<!-- Grid Container -->
<div class="grid grid-cols-1">1 column</div>
<div class="grid grid-cols-2">2 columns</div>
<div class="grid grid-cols-3">3 columns</div>
<div class="grid grid-cols-12">12 columns</div>

<!-- Spanning Columns -->
<div class="grid grid-cols-3">
  <div class="col-span-2">Spans 2 columns</div>
  <div class="col-span-1">Spans 1 column</div>
</div>

<!-- Grid Gap -->
<div class="grid gap-4">Gap all sides</div>
<div class="grid gap-x-4">Column gap</div>
<div class="grid gap-y-4">Row gap</div>
```

Background & Borders

```html
<!-- Background Color -->
<div class="bg-blue-500">Blue background</div>
<div class="bg-green-200">Light green background</div>
<div class="bg-gray-800">Dark gray background</div>
<div class="bg-white">White background</div>

<!-- Border -->
<div class="border">Border all sides</div>
<div class="border-t">Border top</div>
<div class="border-r">Border right</div>
<div class="border-b">Border bottom</div>
<div class="border-l">Border left</div>

<!-- Border Width -->
<div class="border border-2">Width 2px</div>
<div class="border border-4">Width 4px</div>
<div class="border border-8">Width 8px</div>

<!-- Border Color -->
<div class="border border-red-500">Red border</div>
<div class="border border-blue-300">Light blue border</div>

<!-- Border Radius -->
<div class="rounded">Small radius</div>
<div class="rounded-md">Medium radius</div>
<div class="rounded-lg">Large radius</div>
<div class="rounded-full">Full radius (circle/pill)</div>
<div class="rounded-t-lg">Top corners rounded</div>
<div class="rounded-r-lg">Right corners rounded</div>
```

Responsive Design

```html
<!-- Responsive Classes -->
<div class="hidden sm:block">Visibile solo da sm in su</div>
<div class="block md:hidden">Visibile solo fino a md</div>

<!-- Responsive Layout -->
<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3">
  <!-- Contenuto -->
</div>

<!-- Responsive Typography -->
<h1 class="text-xl md:text-2xl lg:text-3xl">Titolo responsivo</h1>
```

State Variants

```html
<!-- Hover, Focus, Active states -->
<button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">
  Hover me
</button>

<input class="border border-gray-300 focus:border-blue-500 focus:ring-1 focus:ring-blu

<button class="bg-green-500 active:bg-green-700">
  Click me
</button>

<!-- Other States -->
<input class="border invalid:border-red-500" type="email" />
<div class="odd:bg-gray-100 even:bg-white">Row with alternating background</div>
<a class="text-blue-500 visited:text-purple-500">Link</a>
```

Personalizzazione Tailwind

javascript

```javascript
// tailwind.config.js
module.exports = {
  content: ["./src/**/*.{html,js,jsx,ts,tsx}"],
  theme: {
    extend: {
      colors: {
        primary: '#ff6b6b',
        secondary: '#4ecdc4',
      },
      fontFamily: {
        sans: ['Inter', 'sans-serif'],
        heading: ['Montserrat', 'sans-serif'],
      },
      spacing: {
        '72': '18rem',
        '84': '21rem',
        '96': '24rem',
      },
      borderRadius: {
        'xl': '1rem',
        '2xl': '2rem',
      },
      screens: {
        'xs': '475px',
      }
    },
  },
  plugins: [
    require('@tailwindcss/forms'),
    require('@tailwindcss/typography'),
  ],
}
```

Component Extraction

Approccio con @apply

```css
/* In un file CSS con Tailwind importato */
@layer components {
  .btn {
    @apply px-4 py-2 rounded font-semibold text-sm text-white focus:outline-none focus
  }

  .btn-primary {
    @apply bg-blue-500 hover:bg-blue-600 focus:ring-blue-500;
  }

  .btn-secondary {
    @apply bg-gray-500 hover:bg-gray-600 focus:ring-gray-500;
  }

  .card {
    @apply bg-white rounded-lg shadow-md overflow-hidden;
  }
}
```

Approccio con React Components

```jsx
function Button({ variant = 'primary', children, ...props }) {
  const baseClasses = "px-4 py-2 rounded font-semibold text-sm text-white focus:outlin

  const variantClasses = {
    primary: "bg-blue-500 hover:bg-blue-600 focus:ring-blue-500",
    secondary: "bg-gray-500 hover:bg-gray-600 focus:ring-gray-500",
    danger: "bg-red-500 hover:bg-red-600 focus:ring-red-500"
  };

  return (
    <button
      className={`${baseClasses} ${variantClasses[variant]}`}
      {...props}
    >
      {children}
    </button>
  );
}
```

Strategie di Design CSS

Atomic CSS vs BEM vs Tailwind

BEM (Block Element Modifier)

```html
<div class="card">
  <div class="card__header">
    <h2 class="card__title">Titolo</h2>
  </div>
  <div class="card__body">
    <p class="card__text">Contenuto</p>
  </div>
  <div class="card__footer">
    <button class="card__button card__button--primary">Action</button>
  </div>
</div>
```

```css
.card {
  border: 1px solid #ddd;
  border-radius: 8px;
}

.card__header {
  padding: 16px;
  border-bottom: 1px solid #ddd;
}

.card__title {
  margin: 0;
  font-size: 18px;
}

.card__body {
  padding: 16px;
}

.card__footer {
  padding: 16px;
  border-top: 1px solid #ddd;
}

.card__button {
  padding: 8px 16px;
  border-radius: 4px;
}

.card__button--primary {
  background-color: blue;
  color: white;
}
```

Tailwind (Utility-First)

```html
<div class="border border-gray-200 rounded-lg">
  <div class="p-4 border-b border-gray-200">
    <h2 class="text-lg font-semibold m-0">Titolo</h2>
  </div>
  <div class="p-4">
    <p class="text-gray-700">Contenuto</p>
  </div>
  <div class="p-4 border-t border-gray-200">
    <button class="px-4 py-2 bg-blue-500 text-white rounded">Action</button>
  </div>
</div>
```

Design Systems con Tailwind CSS

Un design system è una collezione di componenti riutilizzabili, guidati da standard chiari che possono essere assemblati per costruire applicazioni.

Creazione di un Design System

1. **Definire Token di Design**:

```javascript
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        brand: {
          50: '#f0f9ff',
          100: '#e0f2fe',
          // ...
          900: '#0c4a6e',
        },
        // ...
      },
      spacing: {
        xs: '0.25rem',
        sm: '0.5rem',
        md: '1rem',
        lg: '1.5rem',
        xl: '2rem',
        // ...
      },
      // Definire tipografia, ombre, breakpoint, ecc.
    }
  }
}
```

2. **Creare Componenti Base**:

```jsx
// Button.jsx
export function Button({ variant = 'primary', size = 'md', children, ...props }) {
  const baseClasses = "font-medium rounded focus:outline-none focus:ring-2";

  const variantClasses = {
    primary: "bg-brand-600 text-white hover:bg-brand-700 focus:ring-brand-500",
    secondary: "bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500",
    // ...
  };

  const sizeClasses = {
    sm: "py-1 px-2 text-sm",
    md: "py-2 px-4 text-base",
    lg: "py-3 px-6 text-lg",
  };

  return (
    <button
      className={`${baseClasses} ${variantClasses[variant]} ${sizeClasses[size]}`}
      {...props}
    >
      {children}
    </button>
  );
}
```

3. **Documentare i Componenti**: Creare un catalogo dei componenti con storybook o una semplice pagina di documentazione

Best Practices CSS e Tailwind

Performance

- Minimizzare CSS in produzione
- Purge CSS non utilizzato (Tailwind lo fa automaticamente con la configurazione corretta)
- Utilizzare specificity bassa per evitare cascate complesse
- Preferire classi a ID per la riutilizzabilità

Organizzazione

- Seguire una convenzione di denominazione coerente (BEM, SMACSS, ecc.)
- Organizzare CSS per componenti
- Per progetti grandi, considerare CSS Modules o CSS-in-JS

Accessibilità

- Usare contrasti adeguati (Tailwind ha classi come `text-gray-900`)
- Non disabilitare outline su focus (usare `focus:ring-2` di Tailwind invece)
- Assicurarsi che i testi siano leggibili (dimensioni e spaziatura adeguate)
- Utilizzare attributi ARIA quando necessario

Responsive Design

- Adottare approccio "mobile-first"
- Testare su diversi dispositivi e browser
- Usare unità relative (rem, em, %) invece di pixel fissi
- Impostare viewport meta tag appropriato

Esercizi Pratici

1. **Ricrea un componente UI esistente**:
   - Scegli un componente da un sito popolare (pulsante, card, navbar)
   - Ricrealo prima con CSS puro, poi con Tailwind
2. **Crea un mini design system**:
   - Definisci colori, tipografia, spaziatura
   - Crea componenti base (button, card, input)
   - Assembla una pagina completa con questi componenti
3. **Responsive Layout Challenge**:
   - Crea un layout che cambi significativamente tra mobile, tablet e desktop
   - Usa Grid o Flexbox
   - Implementa sia con CSS puro che con Tailwind