

Bootstrap

VALERIO BELLANDI

Che cos'è il web design reattivo?

Il responsive web design riguarda la creazione di siti web che si adattano automaticamente per avere un bell'aspetto su tutti i dispositivi, dai piccoli telefoni ai desktop di grandi dimensioni.

Bootstrap è il framework HTML, CSS e JavaScript più popolare per lo sviluppo di siti Web responsive e mobile-first.

Bootstrap è completamente gratuito da scaricare e utilizzare!

Cos'è Bootstrap?

Bootstrap è un framework front-end gratuito per uno sviluppo web più rapido e semplice

Bootstrap include modelli di progettazione basati su HTML e CSS per tipografia, moduli, pulsanti, tavelle, navigazione, modali, caroselli di immagini e molti altri, oltre a plug-in JavaScript opzionali

Bootstrap ti dà anche la possibilità di creare facilmente design reattivi

Cos'è Bootstrap

- Libreria open-source per la realizzazione di siti e applicazioni web responsive
- Framework basato su HTML, CSS e JavaScript



Storia del bootstrap

Bootstrap è stato sviluppato da Mark Otto e Jacob Thornton su Twitter e rilasciato come prodotto open source nell'agosto 2011 su GitHub.

Vantaggi di Bootstrap:

- **Facile da usare:** chiunque abbia una conoscenza di base di HTML e CSS può iniziare a utilizzare Bootstrap
- **Funzionalità reattive:** il CSS reattivo di Bootstrap si adatta a telefoni, tablet e desktop
- **Approccio mobile-first:** in Bootstrap 3, gli stili mobile-first fanno parte del framework principale
- **Compatibilità browser:** Bootstrap è compatibile con tutti i browser moderni (Chrome, Firefox, Internet Explorer, Safari e Opera)

Un po' di storia

- 2010 – Nasce con il nome di “Twitter Blueprint”
- 2011 – Rilascio della prima versione ufficiale di Bootstrap
- 2012 – Rilascio della versione 2
- 2013 – Rilascio della versione 3
- 2016 – Annunciata la prima versione alpha di Bootstrap4



Dove ottenerne Bootstrap?

Ci sono due modi per iniziare a usare Bootstrap sul tuo sito web.

- Scarica Bootstrap da getbootstrap.com
 - Se vuoi scaricare e ospitare Bootstrap da solo, vai su getbootstrap.com e segui le istruzioni lì.
- Includi Bootstrap da un CDN
 - Se non desideri scaricare e ospitare Bootstrap da solo, puoi includerlo da un CDN (Content Delivery Network).
 - MaxCDN fornisce supporto CDN per CSS e JavaScript di Bootstrap. Devi includere anche jQuery.

CDN di bootstrap

Devi includere i seguenti CSS, JavaScript e jQuery di Bootstrap da MaxCDN nella tua pagina web.

```
<!-- Ultimo CSS Bootstrap compilato e minimizzato -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

```
<!-- JavaScript Bootstrap più recente compilato -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

```
<!-- ultima libreria jQuery -->
<script src="https://code.jquery.com/jquery-latest.js"></script>
```

Vantaggio dell'utilizzo di Bootstrap CDN:

- Molti utenti hanno già scaricato Bootstrap da MaxCDN visitando un altro sito. Di conseguenza, verrà caricato dalla cache quando visitano il tuo sito, il che porta a tempi di caricamento più rapidi. Inoltre, la maggior parte dei CDN farà in modo che una volta che un utente richiede un file da esso, verrà servito dal server più vicino a loro, il che porta anche a tempi di caricamento più rapidi.

Crea una pagina Web con Bootstrap (1)

Aggiungi il doctype HTML5

- Bootstrap utilizza elementi HTML e proprietà CSS che richiedono il doctype HTML5.
- Includi sempre il doctype HTML5 all'inizio della pagina, insieme all'attributo lang e al set di caratteri corretto:

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
  </head>
</html>
```

Crea una pagina Web con Bootstrap (2)

Bootstrap è mobile-first

- Bootstrap 3 è progettato per rispondere ai dispositivi mobili. Gli stili mobile-first fanno parte del framework principale.
- Per garantire un rendering e uno zoom al tocco corretti, aggiungi il seguente tag `<meta>` all'interno dell'elemento `<head>`:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- La parte `width=device-width` imposta la larghezza della pagina in modo che segua la larghezza dello schermo del dispositivo (che varierà a seconda del dispositivo).
- La parte `initial-scale=1` imposta il livello di zoom iniziale quando la pagina viene caricata per la prima volta dal browser.

Crea una pagina Web con Bootstrap (3)

Contenitori

- Bootstrap richiede anche un elemento contenitore per racchiudere i contenuti del sito.
- Ci sono due classi di contenitori tra cui scegliere:
 - La classe .container fornisce un **contenitore a larghezza fissa reattivo**. ([Vedi campione](#))
 - La classe .container-fluid fornisce un **contenitore a larghezza intera**, che copre l'intera larghezza del viewport. ([Vedi campione](#))

Nota: i contenitori non sono nidificabili (non è possibile inserire un contenitore all'interno di un altro contenitore).

Griglie Bootstrap

Il sistema a griglia di Bootstrap consente fino a 12 colonne nella pagina.

Se non desideri utilizzare tutte e 12 le colonne singolarmente, puoi raggruppare le colonne insieme per creare colonne più larghe:

```
<div class="col-md-12">Si estende su 12 colonne</div>
```

```
<div class="col-md-6">Intervallo 6</div><div class="col-md-6">Intervallo 6</div>
```

```
<div class="col-md-4">Intervallo 4</div><div class="col-md-8">Intervallo 8</div>
```

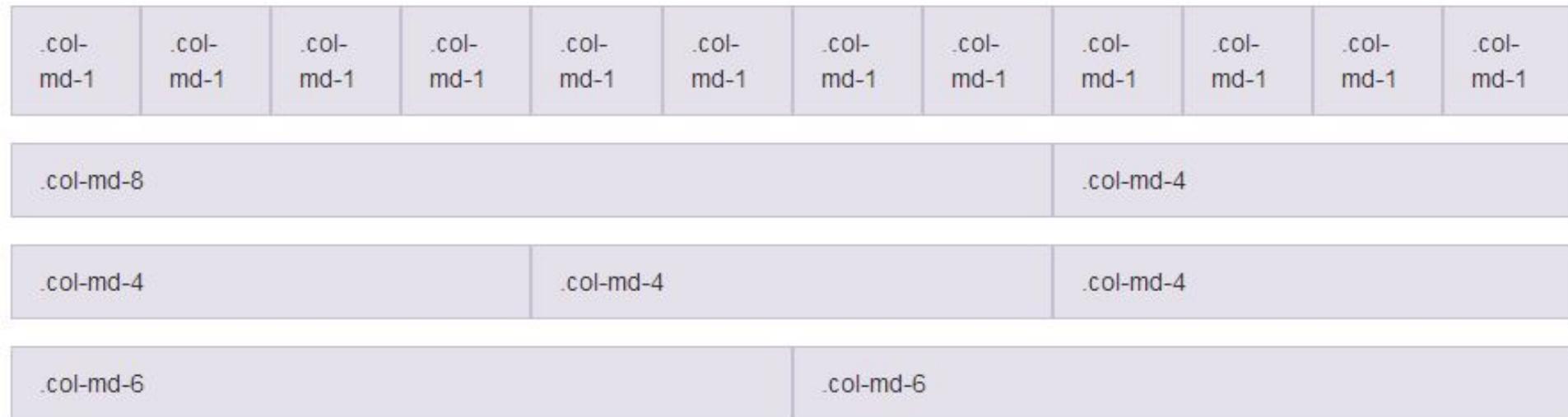
```
<div class="col-md-4">Intervallo 4</div><div class="col-md-4">Intervallo 4</div> <div class="col-md-4">Intervallo 4 </div>
```

Il sistema a griglia di Bootstrap è reattivo e le colonne si riorganizzeranno automaticamente a seconda delle dimensioni dello schermo.

Il grid system di Bootstrap

Grid System basato su dodici colonne innestabili

individuate dalla classe **.col-(nome breakpoint)-***



Il grid system di Bootstrap

E' possibile innestare griglie nelle colonne creando delle .row nelle colonne, reiterando il meccanismo.



Il grid system di Bootstrap

E' possibile gestire i salti di colonna con gli offset col-md-offset-*

.col-md-4

.col-md-4 .col-md-offset-4

.col-md-3 .col-md-
offset-3

.col-md-3 .col-md-
offset-3

.col-md-6 .col-md-offset-3

Il grid system di Bootstrap

Di default è possibile gestire 5 dimensioni del viewport:

Extra small (.col-xs-*), per gli smartphone (< 576px)

Small (.col-sm-*), per i tablet (>= 576px)

Medium (.col-md-*), per i desktop (>= 768px)

Large (.col-lg-*), per i desktop larghi (>=992 px)

Extra Large (.col-xl-*), per i desktop molto larghi (>=1200px)



Nuovo Flexbox grid system di Bootstrap4

Con il nuovo Grid System non dobbiamo più indicare il numero nella classe della colonna, basterà usare solo la classe .col-(nome breakpoint) e si adatteranno in automatico allo spazio disponibile

```
<div class="row">  
    <div class="col-xs"> 1 of 3 </div>  
    <div class="col-xs"> 2 of 3 </div>  
    <div class="col-xs"> 3 of 3 </div>  
</div>
```



Nuovo Flexbox grid system di Bootstrap4

Possiamo allineare le righe in verticale

One of three columns	One of three columns	One of three columns	.row .flex-items-xs-top
One of three columns	One of three columns	One of three columns	.row .flex-items-xs-middle
One of three columns	One of three columns	One of three columns	.row .flex-items-xs-bottom

Nuovo Flexbox grid system di Bootstrap4

...e anche le colonne

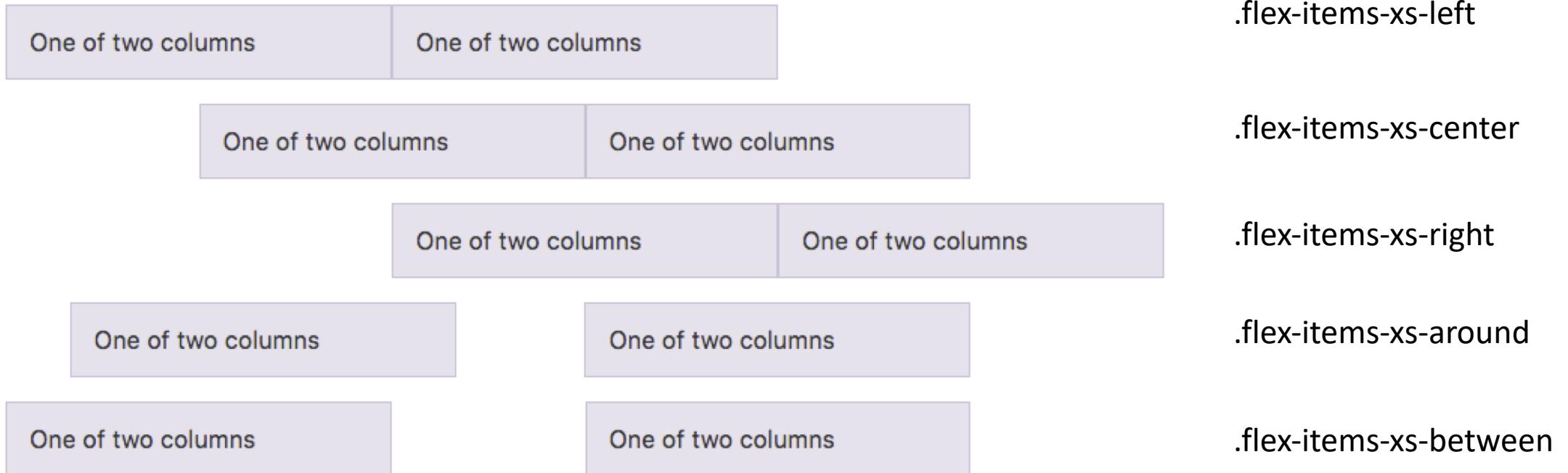
.col-xs .flex-xs-top

.col-xs .flex-xs-middle

.col-xs .flex-xs-bottom

Nuovo Flexbox grid system di Bootstrap4

Riordino orizzontale



Nuovo Flexbox grid system di Bootstrap4

Riordino della posizione

Third, but first

First, but unordered

Second, but last

.flex-xs-unordered

.flex-xs-last

.flex-xs-first

Altre novità di Bootstrap4

- Nuovi Helpers
- Migliorati i “Component” già esistenti e introdotti di nuovi
- Introdotta la nuova unità di misura “REM”



Classi di griglia

Il sistema di griglia Bootstrap ha quattro classi:

- xs (per telefoni)
- sm (per tablet)
- md (per desktop)
- lg (per desktop più grandi)

Le classi di cui sopra possono essere combinate per creare layout più dinamici e flessibili.

Struttura di base di una griglia Bootstrap

```
<div class="row">
<div class="col-*-*"></div>
</div>
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
<div class="row">
...
</div>
```

Primo; creare una riga (`<div class="row">`). Quindi, aggiungi il numero desiderato di colonne (tag con classi `.col-*-*` appropriate). Nota che i numeri in `.col-*-*` dovrebbero sempre sommare fino a 12 per ogni riga.

Tre colonne uguali

Tre colonne uguali ([versione desktop](#)):

Tre colonne uguali ([versione tablet](#)):

Tre colonne uguali ([versione smartphone](#)):

Due colonne disuguali

Due colonne disuguali ([versione desktop](#)):

Due colonne disuguali ([versione tablet](#)):

Due colonne disuguali ([versione smartphone](#)):

Tabelle Bootstrap

Un tavolo Bootstrap **di base ha una leggera imbottitura e solo divisorì orizzontali.**

- La classe .table aggiunge uno stile di base a una tabella:

Righe a strisce

- La classe .table-striped aggiunge strisce zebrate a una tabella:

Tavolo bordato

- La classe .table-bordered aggiunge bordi su tutti i lati della tabella e delle celle:

Righe al passaggio del mouse

- La classe .table-hover abilita uno stato hover sulle righe della tabella:

Tabelle reattive

- La classe .table-responsive crea una tabella reattiva. La tabella scorrerà quindi orizzontalmente su dispositivi di piccole dimensioni (meno di 768 px). Quando si visualizza su qualcosa di più largo di 768 px, non c'è differenza:

Immagini bootstrap

Angoli arrotondati

- La classe .img-rounded aggiunge angoli arrotondati a un'immagine (IE8 non supporta gli angoli arrotondati):

Cerchio

- La classe .img-circle modella l'immagine in un cerchio (IE8 non supporta gli angoli arrotondati):

Miniatura

- La classe .img-thumbnail modella l'immagine in una miniatura:

Immagini reattive

- Le immagini sono disponibili in tutte le dimensioni. Così fanno gli schermi. Le immagini reattive si adattano automaticamente alle dimensioni dello schermo.
- Crea immagini reattive aggiungendo una classe .img-responsive al tag . L'immagine verrà quindi ridimensionata in modo corretto rispetto all'elemento genitore.
- La classe .img-responsive applica display: block; e larghezza massima: 100%; e altezza: auto; all'immagine:

Bottoni Bootstrap

Stili dei pulsanti

- Bootstrap fornisce sette stili di pulsanti con le seguenti classi:

.btn-predefinito

.btn-primario

.btn-successo

.btn-info

.btn-avviso

.btn-pericolo

collegamento .btn

Elementi del pulsante Bootstrap

Le classi di pulsanti possono essere utilizzate sui seguenti elementi:

- <a>
- <pulsante>
- <input>

Dimensioni dei pulsanti

Bootstrap fornisce quattro dimensioni di pulsante con le seguenti classi:

.btn-lg

.btn-md

.btn-sm

.btn-xs

Pulsanti a livello di blocco

Un pulsante a livello di blocco copre l'intera larghezza dell'elemento genitore.

- Aggiungi la classe .btn-block per creare un pulsante a livello di blocco:

Pulsanti attivi/disabilitati

Un pulsante può essere impostato su uno stato attivo (appare premuto) o disabilitato (non cliccabile):

- La classe `.active` fa apparire premuto un pulsante e la classe `.disabled` rende un pulsante non cliccabile:

Personalizzazione

Il codice sorgente di Bootstrap può essere personalizzato:

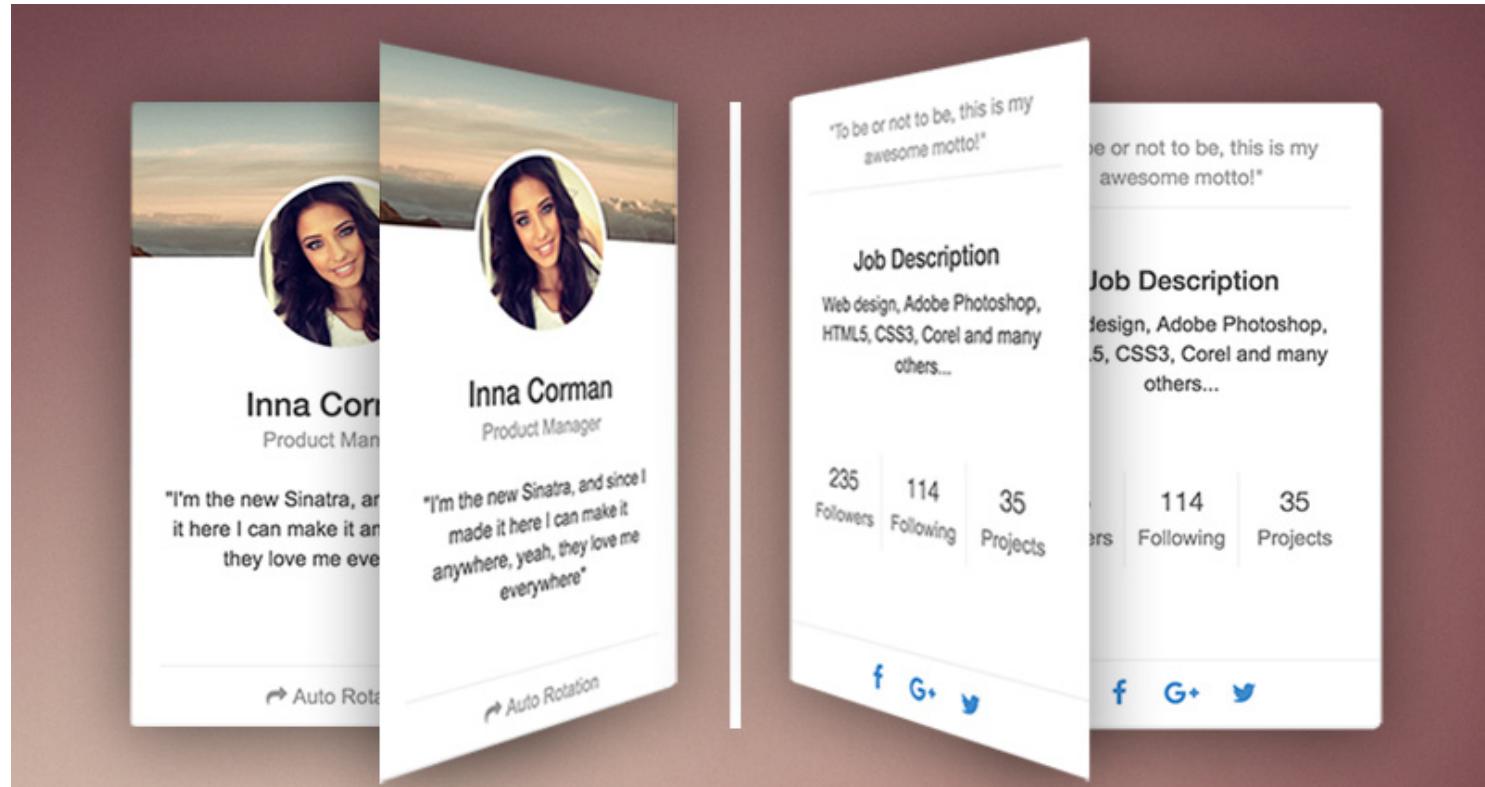
- Attraverso un tool online (V3)
- Less (V3)
- Sass (V3 e V4)

{less} *Sass*



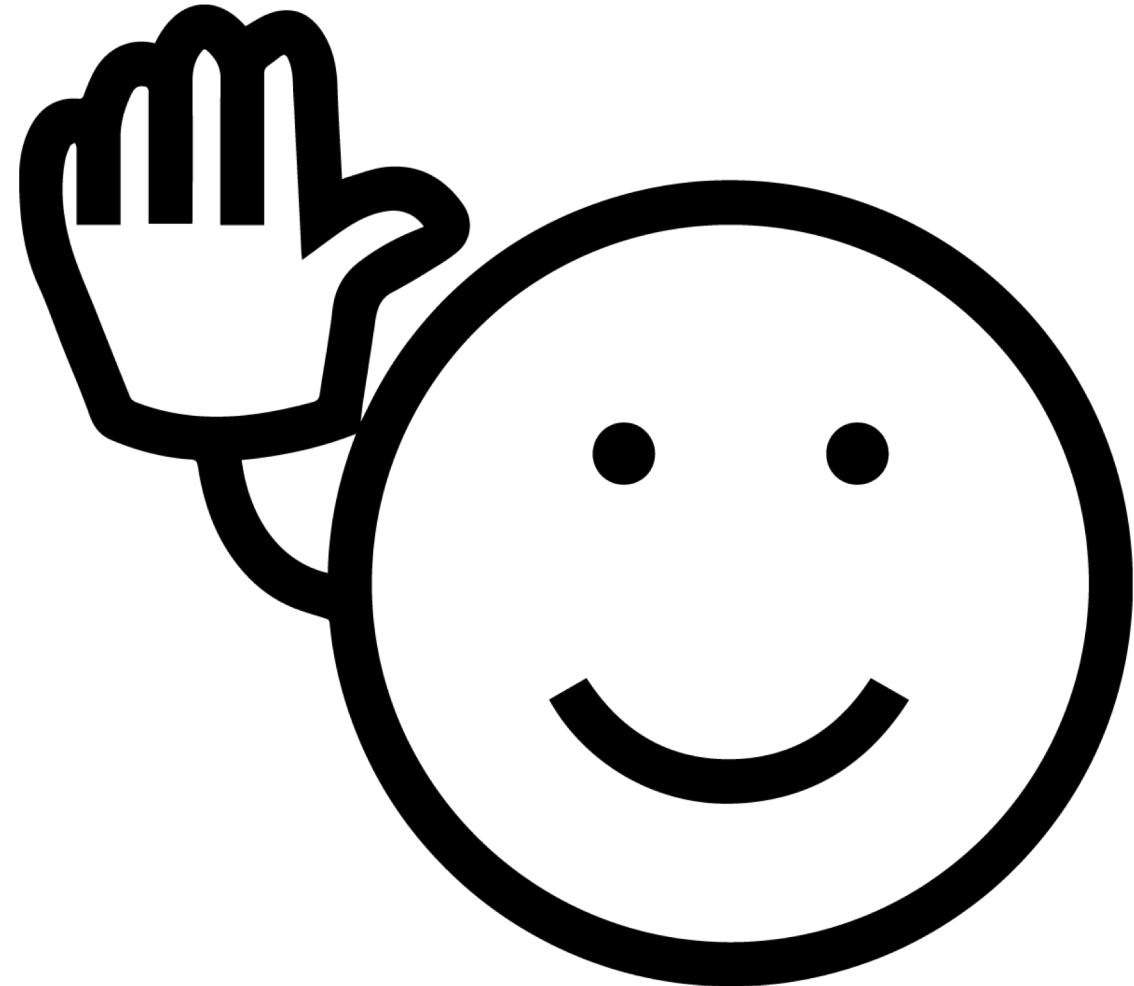
Cards

Wells, thumbnails
e panels ci lasciano
in favore delle cards.



Addio icone

La libreria Glyphicon
non sarà più inclusa

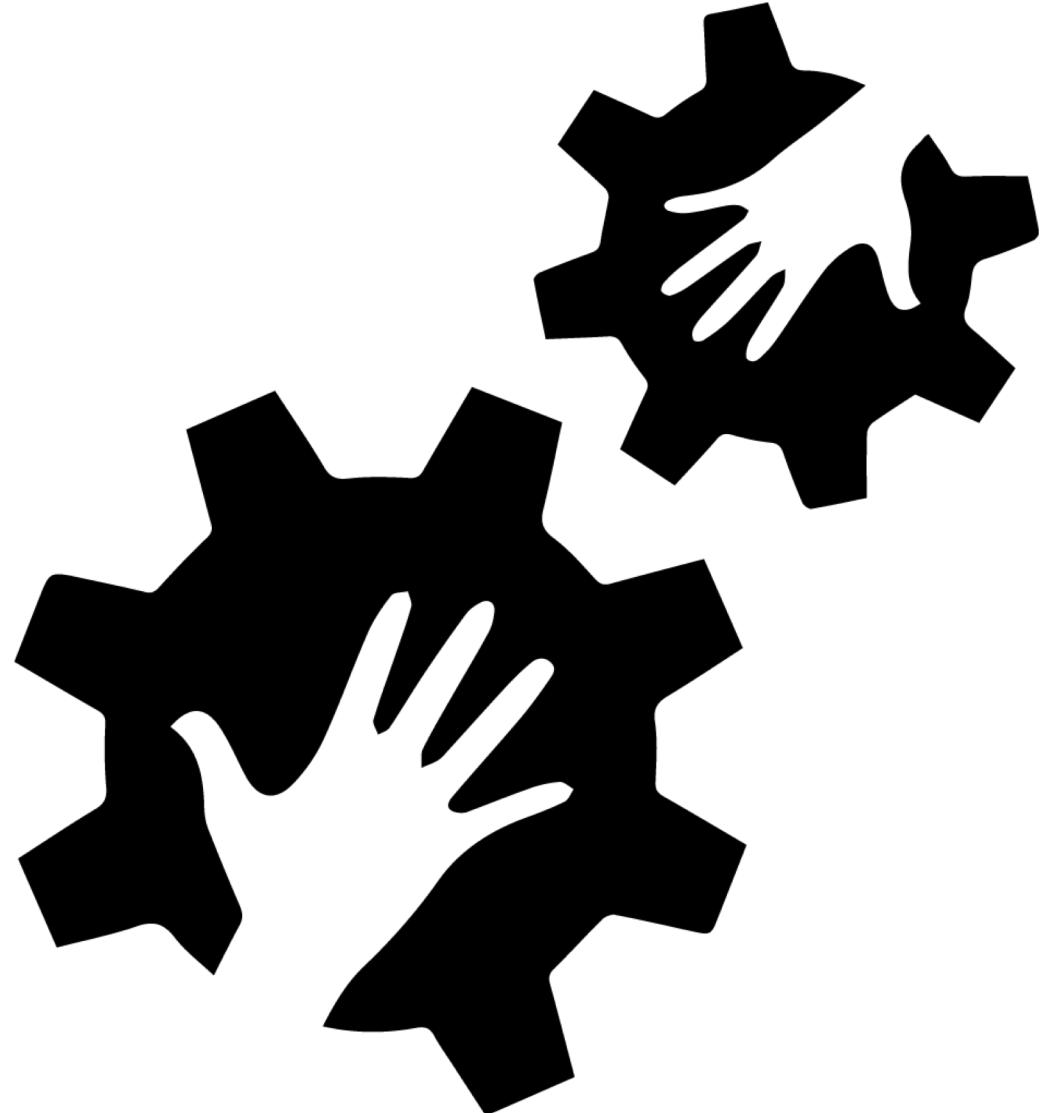


Nuovi Helpers

Nuove classi di supporto per gestire
margin, padding e proprietà display

.pb-0 (padding-bottom: 0;)
.p-3 (padding: 3rem;)

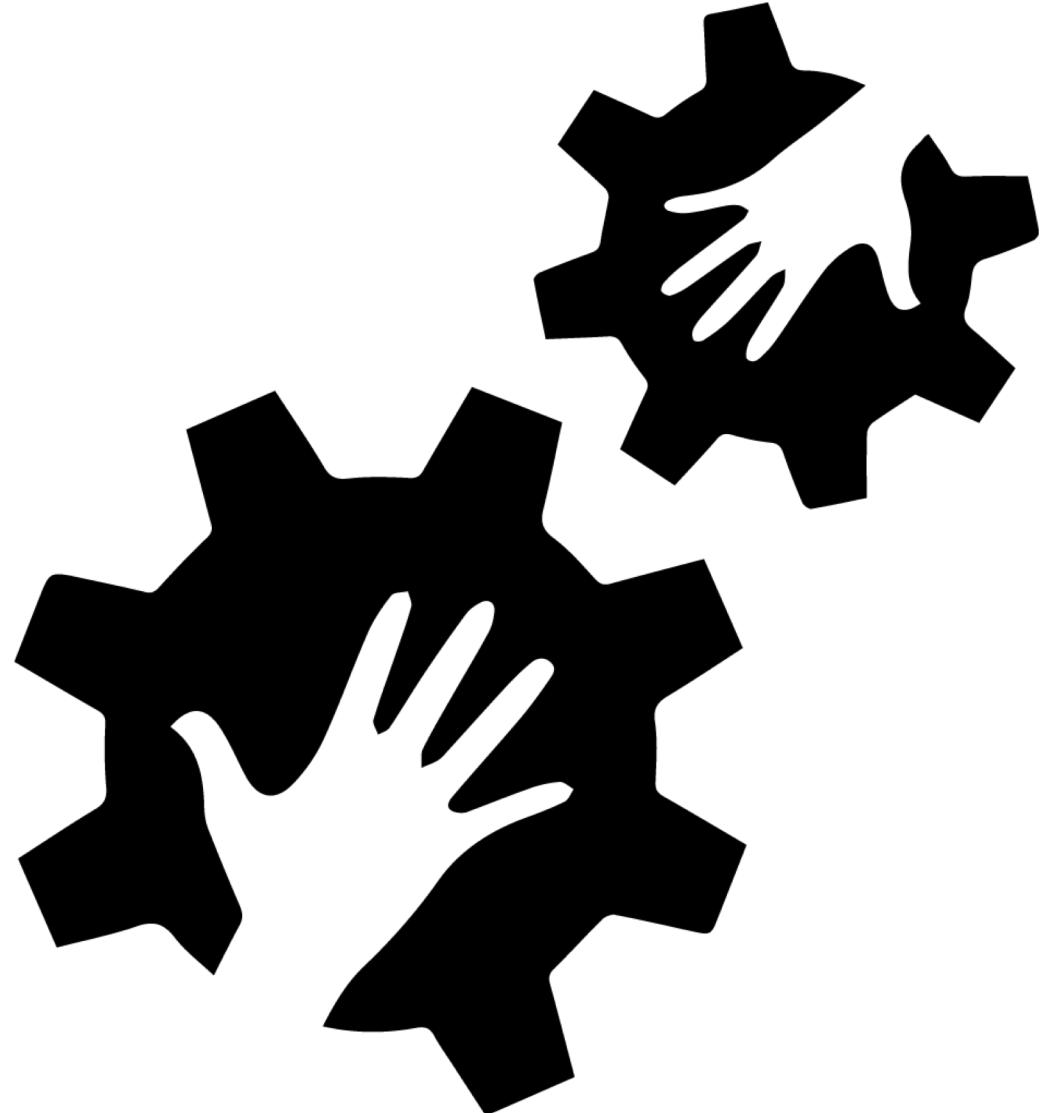
.mt-0 (margin-top: 0;)
.ml-1 (margin-left: 1rem;)
.mx-auto (margin-left: auto;
margin-right: auto;)



Nuovi Helpers

Migliorate le classi di supporto per allineare il testo in base al breakpoint

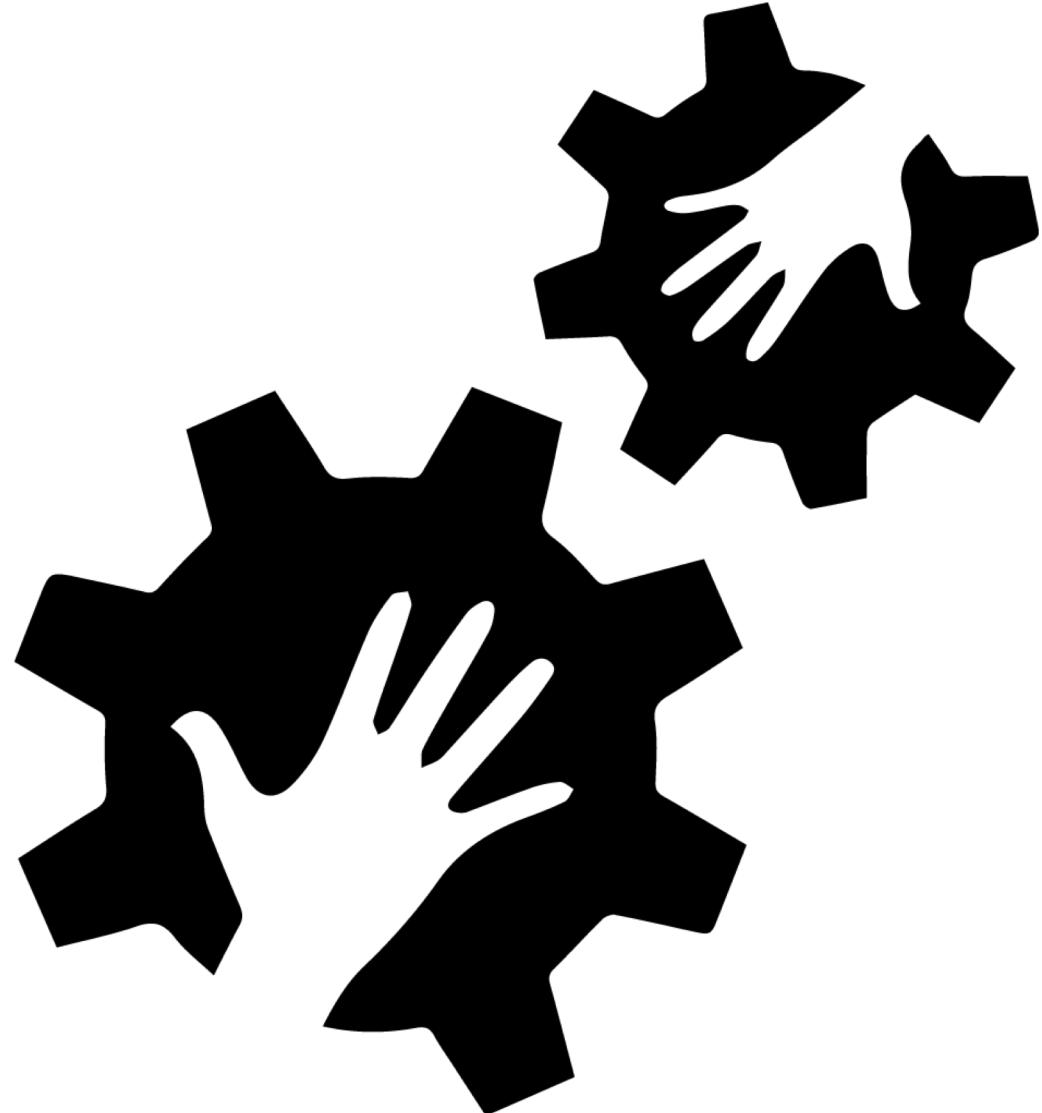
- `.text-xs-left (-right, -center)`
- `.text-sm-left (-right, -center)`
- `.text-md-left (-right, -center)`
- `.text-lg-left (-right, -center)`
- `.text-xl-left (-right, -center)`



Nuovi Helpers

Migliorate le classi di supporto per allineare gli elementi in base al breakpoint

- .float-xs-left (-right)**
- .float-sm-left (-right)**
- .float-md-left (-right)**
- .float-lg-left (-right)**
- .float-xl-left (-right)**

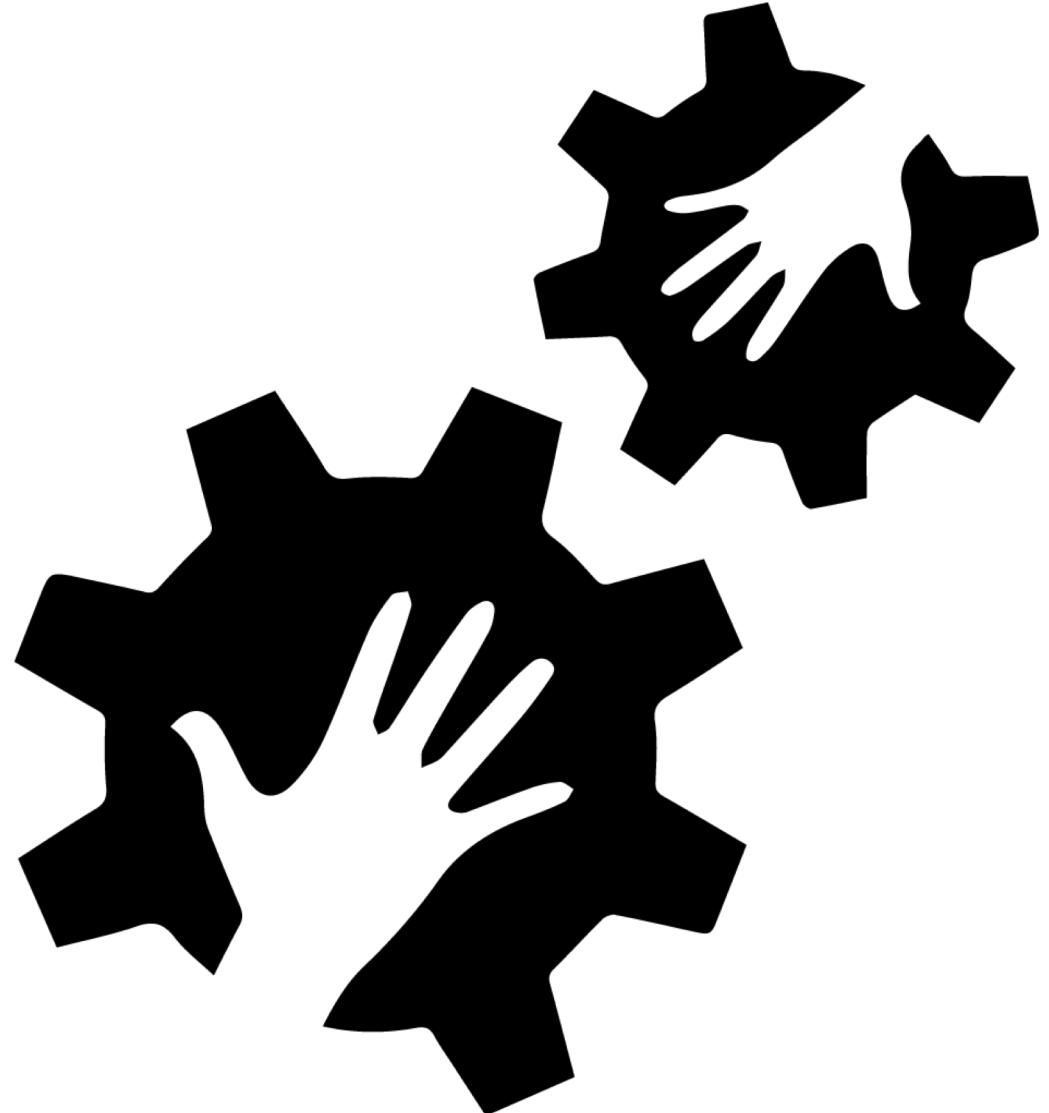


Nuovi Helpers

Migliorate le classi per nascondere o mostrare elementi in base al breakpoint

.hidden-xs diventa .hidden-xs-up

Ad es.: .hidden-md-up
nasconderà tutti gli elementi dal
breakpoint md in su



Introduzione a SASS

SASS: è un pre-processore CSS, un meta-linguaggio che ci permette di usare una sintassi più simile a quella dei linguaggi di programmazione offrendo moltissimi vantaggi tra cui una maggiore manutenibilità del codice



Ma non bastava CSS?

- La **struttura e la sintassi** dei fogli di stile è rimasta invariata fin dal 1996 l'anno della sua prima introduzione.
- I CSS hanno una struttura semplice ed efficiente ma presentano alcune debolezze.



```
color: #696969;
font-weight: 500;
font-family: Arial;
```

```
color: #696969;
font-weight: 500;
font-family: Arial;
```

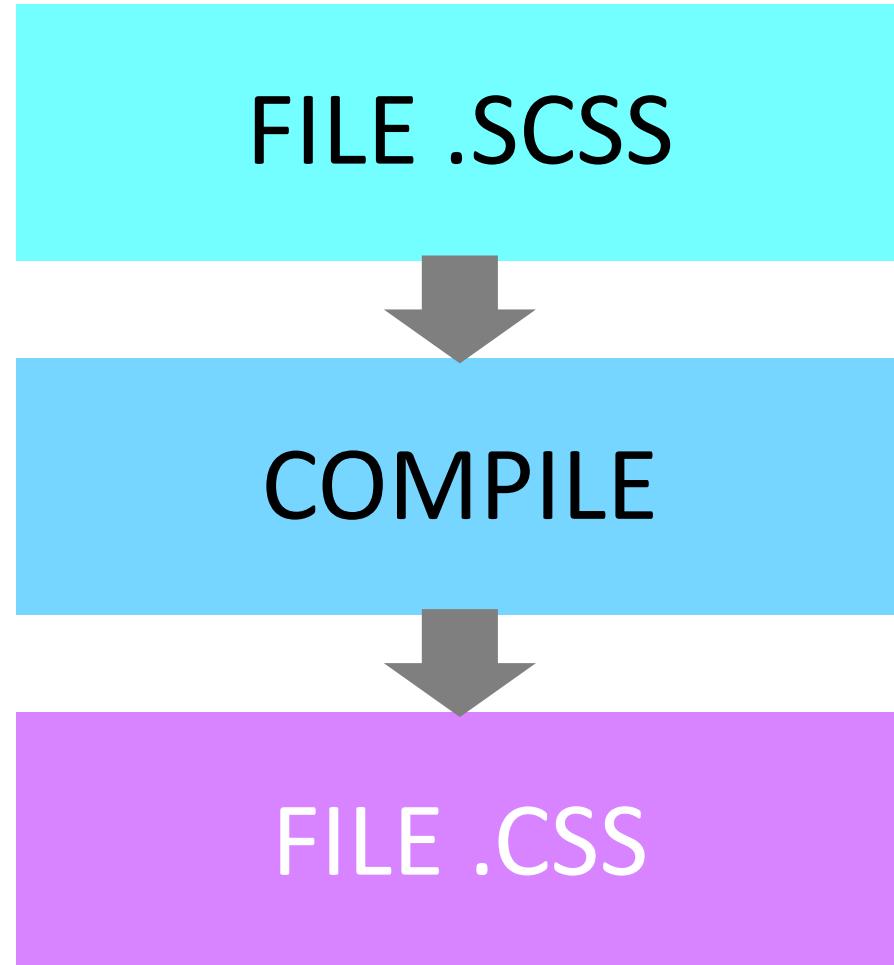
```
color: #4F4F4F;
font-weight: 400;
font-family: Arial;
```

Perché è nato Sass

- Per rimediare a disordine e ripetizioni
- Per introdurre una struttura formale nei CSS
- Migliorare la scrittura del codice e renderlo più manutenibile
- Ottimizzare la gestione delle media queries e andare incontro alle esigenze del mercato in crescita di dispositivi mobili



Come funziona



Caratteristiche di Sass

Nesting - possibilità di nidificare i selettori ottenendo un codice pulito

Variabili – come in altri linguaggi è un insieme di dati modificabili situati in una porzione di memoria;

Importazione – permette di importare altri file scss o css senza generare un grosso numero di richieste HTTP che rallenterebbero inevitabilmente il sito

Funzioni – in Sass esistono dei set di funzioni preconfezionate per effettuare diverse operazioni, è inoltre possibile sviluppare funzioni personalizzate;

Mixin - permette di realizzare dei set riutilizzabili di regole CSS;

Logica condizionale – @if, @else, @else if

Cicli - @for, @each, @while

Compass, scout, ecc... – framework opensource che offrono un'ampia gamma di mixin, funzioni e variabili preconfezionati



Nesting

- Con il **nesting** basta dichiarare una sola volta il nome dell'elemento radice e innestare le altre regole figlie all'interno

```
#box{  
    background-color: red;  
  
    header{  
        background-color: yellow;  
  
        h1{ color: black; }  
    }  
  
    section h1{ color: blue; }  
}
```

```
#box{  
    background-color: red;  
}  
  
#box header{  
    background-color: yellow;  
}  
  
#box header h1{  
    color: black;  
}  
  
#box section h1{  
    color: blue;  
}
```

Nesting

- SASS fornisce il selettore speciale “&” che, utilizzato all’interno di un **nesting** farà riferimento all’elemento radice

```
#box {  
  &-main {  
    width: 100%;  
  }  
  &.small {  
    width: 100%;  
  }  
}
```



```
1  ▼ #box-main {  
2    width: 100%;  
3  }  
4  ▼ #box.small {  
5    width: 100%;  
6  }  
7  
8  
9
```

Nesting

- “&” Può essere utilizzato per definire le pseudoclassi di un elemento

```
a {  
    color: #000;  
    &:hover{  
        color: #aaa;  
    }  
}
```

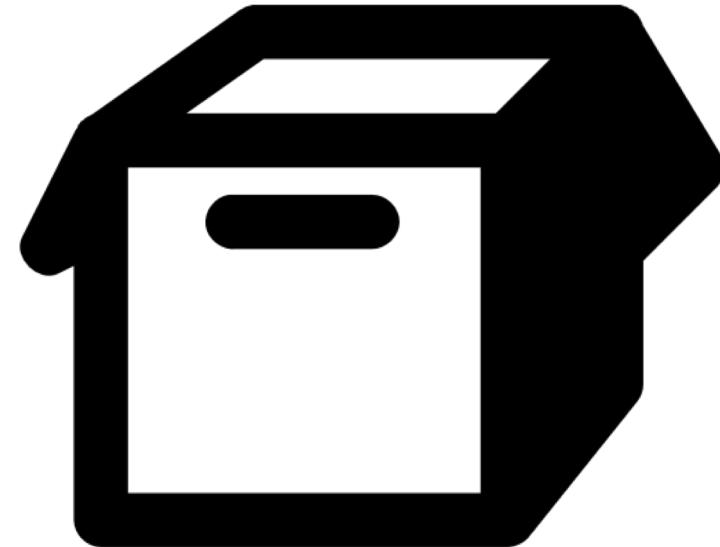


```
1 ▼ a {  
2     color: #000;  
3 }  
4 ▼ a:hover {  
5     color: #aaa;  
6 }  
7
```

Variabili

Variabili: come nei linguaggi di programmazione il concetto di variabile è pressoché identico, una variabile **SASS** è un contenitore virtuale di:

- Stringhe di testo
- Valori numerici
- Numeri in virgola mobile
- Valori booleani



Variabili e set di colori

Esempio pratico: possiamo creare dei set di colori

```
$base-color: #abcdef;  
  
button {  
  background: $base-color;  
}  
a {  
  color: $base-color;  
}  
header {  
  border-bottom: 5px;  
  border-color: $base-color;  
}
```



```
button {  
  background: #abcdef;  
}  
  
a {  
  color: #abcdef;  
}  
  
header {  
  border-bottom: 5px;  
  border-color: #abcdef;  
}
```

Variabili e Media Queries

Esempio pratico: possiamo gestire facilmente le media queries

```
$smartphone: 320px;  
$hd-monitor: 1200px;  
  
@media screen and (min-width: $smartphone) {  
    .box {  
        float: left;  
    }  
}  
  
@media screen and (min-width: $hd-monitor) {  
  
    .box {  
        float: right;  
    }  
}
```

Variabili e Media Queries

Esempio pratico: possiamo gestire facilmente le media queries

```
$landscape: "(max-width: 767px)"; @media #{$landscape} { }

$xsmall: "(max-width: 480px)"; @media #{$xsmall} { }

$xxsmall: "(max-width: 320px)"; @media #{$xxsmall} { }

$small: "(min-width: 768px) and (max-width: 991px)"; @media #{$small} { }

$medium: "(min-width: 992px) and (max-width: 1199px)"; @media #{$medium} { }

$large: "(min-width: 1200px)"; @media #{$large} { }
```

Operazioni

Esempio pratico: possiamo gestire facilmente le media queries

```
$wrapper-width: 1120px;  
$aside-width: 230px;  
  
aside {  
  width: $aside-width / $wrapper-width * 100%;  
}
```



```
aside {  
  width: 20.53571%;  
}
```

Extend

- Con la direttiva **@extend** stiamo dicendo a **SASS** che vogliamo che l'elemento selezionato erediti le regole da un'altra regola o da un placeholder

```
.message {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
}  
.error {  
  @extend .message;  
  color: red;  
  border-color: red;  
}  
.alert {  
  @extend .message;  
  color: orange;  
  border-color: orange;  
}
```



```
.message, .error, .alert {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
}  
.error {  
  color: red;  
  border-color: red;  
}  
.alert {  
  color: orange;  
  border-color: orange;  
}
```

Extend placeholder

```
%message {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
}  
  
.error {  
  @extend %message;  
  color: red;  
  border-color: red;  
}  
  
.alert {  
  @extend %message;  
  color: orange;  
  border-color: orange;  
}
```



```
.error, .alert {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
}  
  
.error {  
  color: red;  
  border-color: red;  
}  
  
.alert {  
  color: orange;  
  border-color: orange;  
}
```

Mixin

Mixin: serve a realizzare un set di proprietà con la possibilità di replicare la stessa struttura generale con alcune piccole, ma significative, variazioni.

```
@mixin message {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
}  
.error {  
  @include message;  
  color: red;  
  border-color: red;  
}  
.alert {  
  @include message;  
  color: orange;  
  border-color: orange;  
}
```



```
.error {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
  color: red;  
  border-color: red;  
}  
.alert {  
  font-weight: bold;  
  padding: 1em;  
  border-width: 2px;  
  color: orange;  
  border-color: orange;  
}
```

Mixin

@Mixin – la direttiva che servirà a dire a **SASS** che stiamo creando un mixin

nome – il nome servirà a richiamare il mixin (nell'esempio è **alert_box**)

\$color – è una variabile, il parametro che andremo ad utilizzare all'interno del nostro codice e che verrà sostituito dal valore passato nel momento della chiamata al mixin.

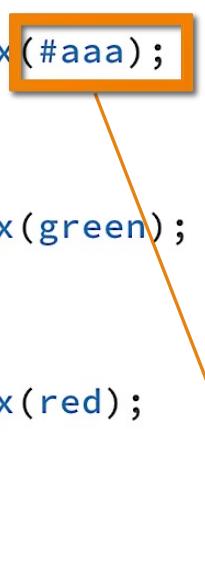
```
@mixin alert_box($color){  
    border: 2px solid;  
    border-radius: 10px;  
    padding: 15px;  
    margin: 0 auto 10px;  
    background-color: $color;  
    border-color: border_color($color);  
}
```

Mixin

Per richiamare un mixin va utilizzata la direttiva `@include nome_mixin(parametri)`

```
@mixin alert_box($color){  
  border: 2px solid;  
  border-radius: 10px;  
  padding: 15px;  
  margin: 0 auto 10px;  
  background-color: $color;  
  border-color: border_color($color);  
}
```

```
.info{  
  @include alert_box(#aaa);  
}  
  
.success{  
  @include alert_box(green);  
}  
  
.error{  
  @include alert_box(red);  
}
```

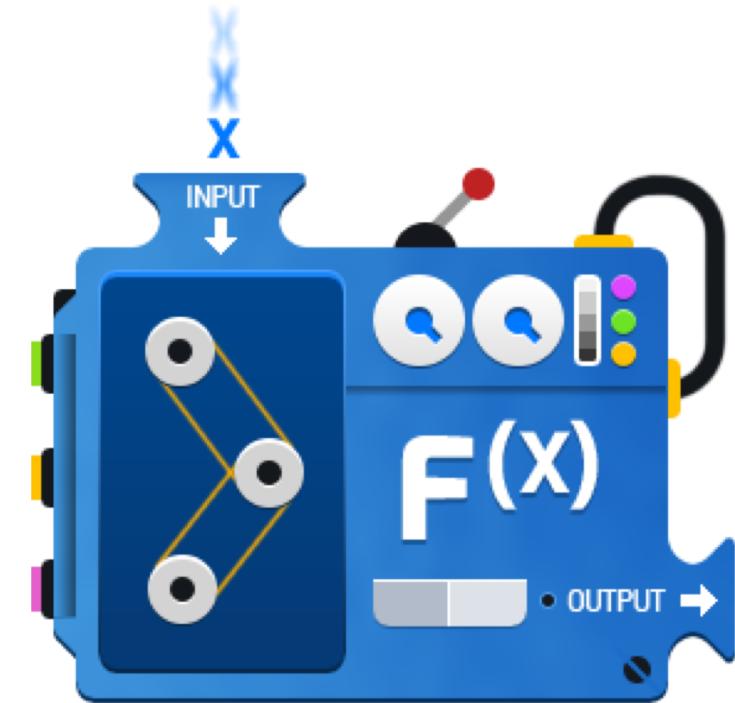


il parametro che sostituirà la variabile `$color`
definita all'interno del mixin.

Funzioni

Funzione: può essere considerata una generalizzazione di `@mixin`.

Una `@function` accetta dei parametri in ingresso, effettua una elaborazione e ritorna un valore attraverso la direttiva `@return`.



Funzioni

@function - la direttiva che servirà per dire a **SASS** che stiamo creando una funzione

nome – il nome servirà a richiamare il mixin (nell'esempio è **add**)

\$value – è una variabile, rappresenta il parametro che viene passato alla funzione e che verrà sostituito con il valore da utilizzare;

@return – è il valore restituito dalla funzione

```
@function add( $value ) {
    $value: $value + 25;
    @return $value;
}
```

```
.box {
    width: add(15px);
}
```



Funzioni preimpostate in Sass

```
$btn-color: #abcdef;  
  
button {  
  background: $btn-color;  
  &:hover {  
    background-color: saturate($btn-color, 50%)  
  }  
  &.disabled {  
    background-color: desaturate($btn-color, 50%);  
  }  
}
```



```
button {  
  background: #abcdef;  
}  
  
button:hover {  
  background-color: #9bcdff;  
}  
  
button.disabled {  
  background-color: #c4cdd6;  
}
```

@if

La direttiva `@if` permette di generare, interi blocchi di dichiarazioni al verificarsi di una condizione. Nell'esempio è stata impiegata all'interno del mixin colors:

```
$template: dark;  
  
@mixin colors {  
  @if $template == dark {  
    background-color: $bg-dark;  
    color: $fg-bright;  
  }  
  @if $template == bright {  
    background-color: $bg-bright;  
    color: $fg-dark;  
  }  
}  
body {  
  @include colors;  
}
```



```
body {  
  background-color: black;  
  color: #eeee99;  
}
```

@else

In questo secondo esempio, testeremo una sola condizione (`$background == dark`), utilizzando la clausola `@else` nel caso in cui questa non sia verificata:

```
$template2: darkness;

@mixin colors2 {
  @if $template2 == dark {
    background-color: $bg-dark;
    color: $fg-bright;
  } @else {
    background-color: $bg-bright;
    color: $fg-dark;
  }
}
body {
  @include colors2;
}
```



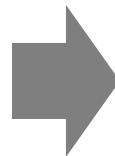
```
body {
  background-color: #eeee99;
  color: #999;
}
```

@else if

Abbiamo generato due possibili set di proprietà a seconda del valore di \$template.

```
$template3: other;

@mixin colors3 {
  @if $template3 == dark {
    background-color: $bg-dark;
    color: $fg-bright;
  } @else if $template3 == bright {
    background-color: $bg-bright;
    color: $fg-dark;
  } @else {
    background-color: $bg-green;
    color: $fg-green;
  }
}
body {
  @include colors3;
}
```



```
body {
  background-color: #00ae00;
  color: #004800;
}
```

Cicli

SASS mette a disposizione dei **costrutti iterativi**, i cosiddetti **cicli**, che consentono di gestire definizioni ripetitive.

@for	Ripete le istruzioni un numero specificato di volte.
@each	Itera tra gli elementi di una lista o di una mappa.
@while	Cicla finché la dichiarazione restituisce false.

@for

La direttiva **@for** genera ripetutamente set di dichiarazioni in un numero finito di cicli, in base al valore assunto da un contatore.

La direttiva può essere definita secondo due strutture:

@for \$var from <start> through <end>
(viene considerato anche il valore finale)

@for \$var from <start> to <end>
(viene escluso il valore finale)



@for

In questo esempio il **mixin** level accetta come argomento un numero intero che stabilisce il numero di ripetizioni del ciclo **@for** in esso contenuto. Ad ogni ciclo, viene prodotto un nome di selettore, grazie alla funzionalità dell'**interpolazione**.

```
@mixin level($levels) {  
  @for $i from 1 through $levels {  
    .levels-#{$i} {  
      z-index: #{$i};  
    }  
  }  
  
@include level(3);  
  .levels-1 {  
    z-index: 1;  
  }  
  .levels-2 {  
    z-index: 2;  
  }  
  .levels-3 {  
    z-index: 3;  
  }
```

@each

Grazie alla **direttiva @each** possiamo iterare tra gli elementi di una lista o di una mappa.

```
@each $icon in home, contact, links, about, blog {  
  .#$icon-icon {  
    padding: 10px;  
    background: url(images/icn-#${icon}.png) no-repeat center center;  
  }  
}
```

```
.home-icon {  
  padding: 10px;  
  background: url(images/icn-home.png) no-repeat center center;  
}  
.contact-icon {  
  padding: 10px;  
  background: url(images/icn-contact.png) no-repeat center center;  
}  
.links-icon {  
  padding: 10px;  
  background: url(images/icn-links.png) no-repeat center center;  
}  
.about-icon {  
  padding: 10px;  
  background: url(images/icn-about.png) no-repeat center center;  
}  
.blog-icon {  
  padding: 10px;  
  background: url(images/icn-blog.png) no-repeat center center;  
}
```

@each

@each accetta anche l'assegnazione di variabili multiple, ad ogni variabile corrisponde una lista. Ad ogni ripetizione, ad ogni variabile sarà assegnato il suo valore tra quelli elencati tra parentesi.

```
@each $icon, $bg-color, $border-color in
  ..... (home, #ffd5d5, #ab0a0a),
  ..... (contact, #cdffbe, #1c6605),
  ..... (links, #bdc5ff, #081787) {
  .... .#$icon-icon {
  ....   padding-left: 16px;
  ....   background-image: url(icons/#{$icon}.png) center left no-repeat;
  ....   background-color: $bg-color;
  ....   border: 1px solid $border-color;
  .... }
  }

.home-icon {
  padding-left: 16px;
  background-image: url(icons/home.png) center left no-repeat;
  background-color: #ffd5d5;
  border: 1px solid #ab0a0a;
}

.contact-icon {
  padding-left: 16px;
  background-image: url(icons/contact.png) center left no-repeat;
  background-color: #cdffbe;
  border: 1px solid #1c6605;
}

.links-icon {
  padding-left: 16px;
  background-image: url(icons/links.png) center left no-repeat;
  background-color: #bdc5ff;
  border: 1px solid #081787;
}
```

@each

Le mappe sono liste di coppie \$key: \$value

```
@each $highlight, $color in (primary: white, user: red, info: cyan) {  
  .btn-#${highlight} {  
    background-color: $color;  
  }  
}
```

```
.btn-primary {  
  background-color: □white;  
}  
  
.btn-user {  
  background-color: □red;  
}  
  
.btn-info {  
  background-color: □cyan;  
}
```

@while

La direttiva `@while` ripete un ciclo finché l'espressione non genera il risultato false, di solito viene utilizzata per cicli più complessi rispetto a `@for`.

```
$i: 4;  
@while $i > 0 {  
    ... h#{5 - $i} { font-size: ($i)+em; }  
    → $i: $i - 1;  
}
```

```
h1 {  
    ... font-size: 4em;  
}  
h2 {  
    ... font-size: 3em;  
}  
h3 {  
    ... font-size: 2em;  
}  
h4 {  
    ... font-size: 1em;  
}
```

@import

Importazione: Importare dei file con i CSS genera un grosso numero di richieste HTTP che rallentano inevitabilmente il sito.

In SASS i file importati vengono compilati e viene generato un unico foglio di stile.



@import

Ai nomi di tutti i file da importare dobbiamo **anteporre il carattere “_”** serve a dire al compilatore che deve ignorarli, **tutti i file con tale prefisso non saranno compilati**

`_miofile.scss`

`_base.scss`

`_default.scss`



@import

Per importare i file basta utilizzare la direttiva “**@import**” seguita dal nome del file senza “**_**”

```
@import "base.scss";  
@import "default.scss"
```

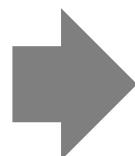
Possiamo anche evitare di scrivere l'estensione .scss, SASS riconoscerà il file in ogni caso

```
@import "base";  
@import "default"
```



@import

```
1 @import "_variables";
2 @import "_reset";
3 @import "_fonts";
4 @import "_header";
5 @import "_footer";
6 @import "_forms";
7 @import "_base";
8 @import "_helpers";
```



```
/* _resets.scss */
html, body, div, span, h1,
h2, h3, h4, h5, h6 {
  margin: 0;
  padding: 0;
}

/* _fonts.scss */
@font-face {
  font-family: myFirstFont;
  src: url(myFont.woff);
}

/* ... */
```

Debug

SOURCE MAPS

COMMENTI DI RIFERIMENTO

Demo



Perché usarlo

- Per utilizzare Variabili, Nesting, Mixins, Operatori, Funzioni, Logica condizionale e cicli.
- Fa risparmiare molte righe di codice.
- Permette di dividere il foglio di stile in file parziali.
- Rende i file CSS più leggibili, riutilizzabili e dinamici.
- Esistono delle librerie di Mixin già pronti come compass, che miglioreranno ulteriormente la nostra esperienza di sviluppo.

