

# Yass - Manuale utente

Xhorxho Papallazi, Marco Rodolfi, Simone Ronzoni

11 dicembre 2023

# 1 Introduzione

Yass è un linguaggio di stile che estende il CSS andando ad aggiungere funzionalità e feature non disponibili in CSS, come variabili, annidamento, mixin, cicli e molto altro.

Il funzionamento è molto semplice: definisci lo stile in un file `.yass` utilizzando la sintassi definita in seguito in questo documento. Genera il file CSS corrispondente con il comando `yass <input>.yass <output>.css` e nel progetto del tuo sito fai riferimento al file css generato. Attenzione a non modificare direttamente il file CSS generato, ma solamente il file `.yass`.

# 2 Installazione

Passi per l'installazione:

1. Installare Java sul proprio sistema.
2. Scaricare il file Jar di Yass.
3. Utilizza un qualsiasi editor di testo.

## 3 Funzionalità

In questo capitolo si fa riferimento alle funzionalità e alla sintassi da utilizzare in Yass. Essendo un linguaggio che estende CSS, per qualsiasi dubbio non presente in questa guida, consultare la documentazione ufficiale di CSS.

### 3.0.1 Variabili

Le variabili sono uno strumento utilizzabile per memorizzare informazioni da riutilizzare in più parti nel foglio di stile. Possono contenere qualsiasi tipo di valore definito in CSS, come colori, font, padding, ... e stringhe generiche. Sono supportate anche liste e dizionari, ma questi verranno mostrati in seguito. La tipizzazione è dinamica.

Una variabile prima di poter essere utilizzata deve essere inizializzata con un valore, altrimenti verrà generato un errore di compilazione. Inoltre, tutte le variabili sono immutabili, ovvero che una volta inizializzate, il loro contenuto non può essere modificato.

L'utilizzo del valore di una variabile all'interno del codice CSS avviene mediante interpolazione secondo la sintassi `#{<nome_variabile>}`. Questo può avvenire nei selettori di un blocco CSS, come valore di proprietà CSS o nella parte destra dell'assegnazione di un'altra variabile. Attenzione che non è possibile interpolare il valore delle variabili all'interno dei nomi delle proprietà o per definire nomi di altre variabili.

Di seguito si mostra un esempio in cui vengono inizializzate due variabili, la prima di tipo stringa usata per definire il nome di classe nel selettore del blocco CSS, mentre la seconda contiene un colore CSS che viene utilizzato per definire il valore della proprietà `background-color`.

```
Yass
1  bg-color-selector = "blue";
2  bg-color = #E6EED6;
3
4  body .#{bg-color-selector} {
5    background-color: #{bg-color};
6  }
```

```
CSS
1  body .blue {
2    background-color: #E6EED6;
3  }
```

### 3.0.2 Tipi iterabili e cicli

Nella sezione precedente, si era anticipato dell'esistenza di due ulteriori tipi di variabile: i dizionari e i cicli. Questi sono definiti tipi iterabili, in quanto permettono di definire un insieme di valori su cui poter ciclare o accedere direttamente tramite chiave per i dizionari, o tramite indice per le liste.

Ciascun elemento di una lista e di un dizionario può essere di qualsiasi tipo CSS o di tipo stringa, ma non può essere un tipo iterabile. Infatti, ad ora non è possibile definire una lista di liste, lista di dizionari, dizionario di dizionari o dizionario di liste.

Per poter accedere a un determinato elemento di una lista o di un dizionario, non è possibile utilizzare l'interpolazione come descritto nella sezione precedente, ma bisogna utilizzare la funzione di sistema `$get(<nome_variabile_iterabile>, <elemento>)`. Il secondo parametro dovrà essere un numero (l'indice dell'elemento) se la variabile passata è una lista, altrimenti una stringa (la chiave) nel caso del dizionario.

Nell'esempio di seguito, viene dichiarata una lista contenente delle dimensioni di font e un dizionario contenente stringhe. Si utilizza la funzione `get` per accedere puntualmente ai singoli elementi del dizionario e della lista.

## Yass

```

1 font-sizes = [ 12px, 18px, 24px ];
2 types-of-texts = {
3   "paragraph": "p",
4   "subtitle": "h6",
5   "title": "h1"
6 };
7
8 body $get(types-of-texts,
9   ↪ "paragraph") {
10   font-size: $get(font-sizes, 0);
11 }
12 body $get(types-of-texts,
13   ↪ "subtitle") {
14   font-size: $get(font-sizes, 1);
15 }
16 body $get(types-of-texts, "title") {
17   font-size: $get(font-sizes, 2);
18 }

```

## CSS

```

1 body p {
2   font-size: 12px;
3 }
4 body h6 {
5   font-size: 18px;
6 }
7 body h1 {
8   font-size: 24px;
9 }

```

Le variabili iterabili sono state introdotte per poter sfruttare i cicli: è possibile ciclare su una lista o dizionario e concatenare un insieme di proprietà definite nel corpo del ciclo.

Si permettono due sintassi per definire i cicli: la prima accetta come parametro una variabile iterabile e predispone nello scope delle variabili disponibili nel corpo del ciclo **value** che contiene il valore dell'elemento *i*-esimo e **index** l'indice.

## Yass

```

1 bg-colors = {
2   "blue": #0000FF,
3   "white": #FFFFFF,
4   "green": #00FF00,
5   "yellow": #FFFF00,
6   "black": #000000
7 };
8
9 $foreach(bg-colors) {
10   body .${index} {
11     background-color: ${value};
12   }
13 }

```

## CSS

```

1
2 body .blue {
3   background-color: #0000FF;
4 }
5
6 body .white {
7   background-color: #FFFFFF;
8 }
9
10 body .green {
11   background-color: #00FF00;
12 }
13
14 body .yellow {
15   background-color: #FFFF00;
16 }
17
18 body .black {
19   background-color: #000000;
20 }

```

Nella seconda sintassi i parametri da passare sono `$foreach(index, item : list)`

**Yass**

```

1
2 bg-colors = {
3   "blue": #0000FF,
4   "white": #FFFFFF,
5   "green": #00FF00,
6   "yellow": #FFFF00,
7   "black": #000000
8 };
9 $foreach(colorName, color :
10  ↪ bg-colors) {
11   body .${colorName} {
12     background-color: ${color};
13   }
14 }

```

**CSS**

```

1
2 body .blue {
3   background-color: #0000FF;
4 }
5
6 body .white {
7   background-color: #FFFFFF;
8 }
9
10 body .green {
11   background-color: #00FF00;
12 }
13
14 body .yellow {
15   background-color: #FFFF00;
16 }
17
18 body .black {
19   background-color: #000000;
20 }

```

Sono supportati cicli annidati e la possibilità di richiamare mixin e variabili definite esternamente. E' inoltre permesso utilizzare gli annidamenti all'interno di cicli.

### 3.0.3 Annidamento

Questa funzionalità permette di specificare stili annidati andando a definire all'interno di un blocco di stile, altri blocchi CSS. Il risultato sarà la creazione di blocchi CSS con selettori la concatenazione dei selettori padre con quelli del figlio.

E' stato introdotto l'operatore & per riferirsi ai selettori della classe padre, potendo specificare la posizione in cui inserirli, ad esempio, per applicare alla classe padre un modificatore di classe (e.g. `btn` e `btn-primary`) o una pseudo classe (e.g. `:hover`).

**Yass**

```

1 body {
2   font-size: 1rem;
3   font-family: Arial;
4
5   &:hover {
6     cursor: pointer;
7   }
8
9   h1 {
10    font-size: 3rem;
11  }
12 }

```

**CSS**

```

1
2 body {
3   font-size: 1rem;
4   font-family: Arial;
5 }
6
7 body:hover {
8   cursor: pointer;
9 }
10
11 body h1 {
12   font-size: 3rem;
13 }

```

### 3.0.4 Mixin

I mixin permettono di definire delle proprietà CSS riutilizzabili, permettendo di parametrizzare determinati valori.

## Yass

```
1 button = (bg-color) {
2   padding: 12px;
3   background-color: ${bg-color};
4 };
5
6 button-primary-color = #2236e1;
7 button-secondary-color = #0a1043;
8
9 button .btn-primary {
10  $button(button-primary-color);
11  font-size: 1.5rem;
12 }
13
14 button .btn-secondary {
15  $button(button-secondary-color);
16 }
```

## CSS

```
1 button .btn-primary {
2   padding: 12px;
3   background-color = #2236e1;
4   font-size: 1.5rem;
5 }
6
7 button .btn-secondary {
8   padding: 12px;
9   background-color: #0a1043;
10 }
```

### 3.1 Mettendo tutto insieme

## Yass

```
1 primary-color = "blue";
2 colors = [ ${primary-color}, "red"];
3 font-sizes = {
4   "small": 0.5em,
5   "normal": 1em,
6   "big": 2em
7 };
8 buttons = {
9   "primary": ${primary-color},
10  "secondary": "gray"
11 };
12 button = (color, font-size) {
13   color: ${color};
14   font-size: ${font-size};
15 };
16 $foreach(colors) {
17   .color-${index} {
18     color: ${value};
19   }
20 }
21 $foreach(button-name, button-color: buttons) {
22   $foreach(font-name, font-size: font-sizes) {
23     .btn-${button-name}__${font-name} {
24       $button(button-color, font-size);
25       &:hover {
26         border: red;
27         border-size: 2px;
28       }
29       &:focus {
30         border: blue;
31         border-size: 1px;
32       }
33     }
34   }
35 }
```

```
1 .color-0 {
2   color: blue;
3 }
4 .color-1 {
5   color: red;
6 }
7 .btn-primary__small {
8   color: blue;
9   font-size: 0.5em;
10 }
11 .btn-primary__small:hover {
12   border: red;
13   border-size: 2px;
14 }
15 .btn-primary__small:focus {
16   border: blue;
17   border-size: 1px;
18 }
19 .btn-primary__normal {
20   color: blue;
21   font-size: 1em;
22 }
23 .btn-primary__normal:hover {
24   border: red;
25   border-size: 2px;
26 }
27 .btn-primary__normal:focus {
28   border: blue;
29   border-size: 1px;
30 }
31 .btn-primary__big {
32   color: blue;
33   font-size: 2em;
34 }
35 .btn-primary__big:hover {
36   border: red;
37   border-size: 2px;
38 }
39 .btn-primary__big:focus {
40   border: blue;
41   border-size: 1px;
42 }
43 .btn-secondary__small {
44   color: gray;
45   font-size: 0.5em;
46 }
47 .btn-secondary__small:hover {
48   border: red;
49   border-size: 2px;
50 }
51 .btn-secondary__small:focus {
52   border: blue;
53   border-size: 1px;
54 }
55
56 ...
```