Yass - Manuale utente

Xhorxho Papallazi, Marco Rodolfi, Simone Ronzoni

20 febbraio 2024

1 Introduzione

Yass è un linguaggo di stile che estende il CSS andando ad aggiungere funzionalità quali variabili, cicli, annidamenti e mixin.

CSS (acronimo di Cascading Style Sheets) è un linguaggio usato per definire la formattazione e lo stile visuale di documenti HTML, XHTML o XML.

Fornisce un controllo completo sulla presentazione di pagine web. Le regole per costruire un CSS sono contenute in un insieme di direttive (le recommendations) emanate a partire dal 1996 dal W3C.

Un file CSS è costituito da:

- *regole* che sono coppie costituite da un selettore e un blocco di dichiarazioni, racchiuso tra parentesi graffe;
- *selettore* è un predicato che individua certi elementi del documento HTML;
- *dichiarazioni*, separate con un punto e virgola dalle altre, e a loro volta costituite da una proprietà, ovvero un tratto di stile (come il colore del testo) e un valore da assegnare a quest'ultimo (per esempio blu), separati dai due punti.

Yass mantiene la stessa struttura di regole del CSS ma ne estende le funzionalità per renderlo più potente.

È sufficiente definire lo stile in un file `.yass`, secondo la sintassi del linguaggio definita nel paragrafo 2. Utilizzando il comando `java -jar yass.jar <input>.yass <output>.css` viene generato il file `.css`, che contiene la traduzione da Yass a CSS e che può essere referenziato all'interno della pagina web in cui lo si vuole utilizzare. Attenzione a non modificare direttamente il file CSS generato, ma solamente il file `.yass`.

1.1 Istruzioni per l'uso

Passi per l'installazione e l'utilizzo:

1. Installare Java sul proprio sistema.
2. Scaricare il file Jar di Yass.
3. Scrivere un file di stile in formato Yass.
4. Compilare il file in CSS.
5. Utilizzare il file su un sito web.

2 Funzionalità

In questo paragrafo 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 è possibile consultare la documentazione ufficiale di CSS.

2.1 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, etc. 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 una volta inizializzate non possono essere riassegnate.

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 variabile 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

```
bg-color-selector = "blue";
bg-color = #E6EED6;

body .${bg-color-selector} {
  background-color: ${bg-color};
}
```

CSS

```
body .blue {
  background-color: #E6EED6;
}
```

2.2 Variabili iterabili e cicli

Nella sezione precedente, si era anticipato dell'esistenza di due ulteriori tipi di variabile: i dizionari e i cicli. Questi definiscono delle variabili iterabili ossia un insieme di valori su cui poter iterare 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(<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

```
font-sizes = [ 12px, 18px, 24px ];
types-of-texts = {
    "paragraph": "p",
    "subtitle": "h6",
    "title": "h1"
};

body $get(types-of-texts,
↪  "paragraph") {
    font-size: $get(font-sizes, 0);
}
body $get(types-of-texts,
↪  "subtitle") {
    font-size: $get(font-sizes, 1);
}
body $get(types-of-texts, "title") {
    font-size: $get(font-sizes, 2);
}

```

CSS

```
body p {
    font-size: 12px;
}
body h6 {
    font-size: 18px;
}
body h1 {
    font-size: 24px;
}
```

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

1. `$foreach(<variabile iterabile>)` accetta come parametro una variabile iterabile e predispone nello scope delle variabili locali del ciclo la variabile `value` che contiene il valore e `index` l'indice dell'elemento i-esimo della variabile iterabile;
2. `$foreach(<nome indice>, <nome valore> : <variabile iterabile>)` è necessaria nel caso di cicli annidati per ridefinire i nomi delle variabili definite nello scope delle variabili locali del ciclo con una sintassi simile a quella di Java. Il primo identificativo consentirà l'accesso alla chiave, in caso di dizionari, o all'indice, in caso di liste, dell'iterazione corrente. Il secondo identificativo si riferirà al valore all'iterazione corrente e dopo il carattere `:` viene specificata la variabile iterabile.

Sono supportati cicli annidati e la possibilità di richiamare mixin e variabili definite esternamente.

Di seguito viene riportato un esempio di definizione di un ciclo con la prima sintassi.

Yass

```
bg-colors = {
    "blue": #0000FF,
    "white": #FFFFFF,
    "green": #00FF00,
    "yellow": #FFFF00,
    "black": #000000
};

$foreach(bg-colors) {
  body .${index} {
    background-color: ${value};
  }
}
```

CSS

```
body .blue {
    background-color: #0000FF;
}

body .white {
    background-color: #FFFFFF;
}

body .green {
    background-color: #00FF00;
}

body .yellow {
    background-color: #FFFF00;
}

body .black {
    background-color: #000000;
}
```

Di seguito viene riportato un esempio di definizione di un ciclo con la seconda sintassi.

Yass

```
bg-colors = {
    "blue": #0000FF,
    "white": #FFFFFF,
    "green": #00FF00,
    "yellow": #FFFF00,
    "black": #000000
};
$foreach(colorName, color :
↪  bg-colors) {
  body .${colorName} {
    background-color: ${color};
  }
}
```

CSS

```
body .blue {
    background-color: #0000FF;
}

body .white {
    background-color: #FFFFFF;
}

body .green {
    background-color: #00FF00;
}

body .yellow {
    background-color: #FFFF00;
}

body .black {
    background-color: #000000;
}
```

2.3 Annidamento di regole CSS

Questa funzionalità permette di specificare stili annidati andando a definire all'interno di un blocco CSS altre regole CSS. Il risultato sarà la creazione di regole CSS che hanno come selettore la concatenazione dei selettori padre con quelli del figlio.

È stato introdotto l'operatore & per riferirsi ai selettori della classe padre, potendo specificare la posizione in cui inserirli, (ad esempio per generare da classi "base", come `btn`, classi specifiche, come `btn-primary`, o pseudo-classi, come `btn:hover`).

Yass

```
body {
  font-size: 1rem;
  font-family: Arial;

  &:hover {
    cursor: pointer;
  }

  h1 {
    font-size: 3rem;
  }
}
```

CSS

```

body {
    font-size: 1rem;
    font-family: Arial;
}

body:hover {
    cursor: pointer;
}

body h1 {
    font-size: 3rem;
}
```

2.4 Mixin

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

Yass

```
button = (bg-color) {
  padding: 12px;
  background-color: ${bg-color};
};

button-primary-color = #2236e1;
button-secondary-color = #0a1043;

button .btn-primary {
  $button(button-primary-color);
  font-size: 1.5rem;
}

button .btn-secondary {
  $button(button-secondary-color);
}
```

CSS

```
button .btn-primary {
  padding: 12px;
  background-color = #2236e1:
  font-size: 1.5rem;
}

button .btn-secondary {
  padding: 12px;
  background-color: #0a1043;
}
```

2.5 Riassumendo

Di seguito viene riportato un esempio che include tutte le funzionalità esposte precedentemente.

Yass

```
primary-color = "blue";
colors = [ ${primary-color}, "red"];
font-sizes = {
  "small": 0.5em,
  "normal": 1em,
  "big": 2em
};
buttons = {
    "primary": ${primary-color},
    "secondary": "gray"
};
button = (color, font-size) {
  color: ${color};
  font-size: ${font-size};
};
$foreach(colors) {
  .color-${index} {
    color: ${value};
  }
}
$foreach(button-name, button-color: buttons) {
  $foreach(font-name, font-size: font-sizes) {
    .btn-${button-name}__${font-name} {
      $button(button-color, font-size);
      &:hover {
        border: red;
        border-size: 2px;
      }
      &:focus {
        border: blue;
        border-size: 1px;
      }
    }
  }

```

CSS

```
.color-0 {
color: blue;
}
.color-1 {
color: red;
}
.btn-primary__small {
color: blue;
font-size: 0.5em;
}
.btn-primary__small:hover {
border: red;
border-size: 2px;
}
.btn-primary__small:focus {
border: blue;
border-size: 1px;
}
.btn-primary__normal {
color: blue;
font-size: 1em;
}
.btn-primary__normal:hover {
border: red;
border-size: 2px;
}
.btn-primary__normal:focus {
border: blue;
border-size: 1px;
}
.btn-primary__big {
color: blue;
font-size: 2em;
}
.btn-primary__big:hover {
border: red;
border-size: 2px;
}
.btn-primary__big:focus {
border: blue;
border-size: 1px;
}
.btn-secondary__small {
color: gray;
font-size: 0.5em;
}
.btn-secondary__small:hover {
border: red;
border-size: 2px;
}
.btn-secondary__small:focus {
border: blue;
border-size: 1px;
}

...
```