COESIONE ITALIA 21-27 PIEMONTE
Cofinanziato dall'Unione europea
REGIONE PIEMONTE

Web Developer

HTML, CSS e Strumenti di Digital Marketing (SEO, SEM, SEA)

Docente: Shadi Lahham

Intervento realizzato da
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

SCSS

Dynamic CSS

Shadi Lahham - Web development

Setup SCSS

non sono modificabili tramite JS

SCSS funzionamento: il browser mangia HTML CSS, convertiamo quindi l'SCSS in CSS
e abbiamo bisogno di un programma per farlo abbiamo bisogno di un applicazione di nome SASS

What is SCSS

Sassy Cascading Style Sheets

- Uses the .scss or the .sass extensions
- We will use **only .scss**

Adds additional features

- Enhanced the plain CSS syntax

Reduces the amount of repetition

- DRY - don't repeat yourself

Fully compatible with CSS

- Every valid CSS file is a valid SCSS file

Sass is a preprocessor

- similar in some ways to Less CSS
- .scss files need to be transpiled to .css
- Compiling vs Transpiling

Use SCSS

SassMeister
Use the SassMeister playground to test SCSS to CSS conversion

Codepen
Use codepen but remember to change the language from CSS to SCSS

Playcode
Use the scss playground on playcode directly

Install and run Scss

Install directly

1. Download Sass from the Repository based on your operating system
2. Add it to your PATH

Install with npm from Node.js

```
npm install -g sass
```

Run Sass

```
sass input.scss output.css
OR
sass --watch input.scss output.css
```

per apportare in tempo reale le aggiunte al file.scss, vengono traslate immediatamente dopo aver salvato al fil.css

SCSS Syntax

Comments

```scss
// variables
$primary: darkorange;
$secondary: #bada55;

// this is a Scss comment, it won't appear in the css file
/* this is a CSS comment, it will appear in the css file */
```

```css
/* this is a CSS comment, it will appear in the css file */
```

// non sono visibili nel css
/**/ sono visibili nel css

Variables $

le $variabili ci permettono di memorizzare delle proprietà
si possono combinare anche tra di loro

SCSS

```scss
$primary: darkorange;
$secondary: #bada55;

$special-border: 2px dashed $secondary;

#sample {
  color: $primary;
  background-color: $secondary;
}

.special {
  border: $special-border;
}

.unique {
  border: 2px solid $primary;
}
```

HTML

```html
<body>
    <div id="sample">I am just a sample</div>
    <div class="special">I am special</div>
    <div class="special">I am special too</div>
    <div class="unique">I am unique</div>
</body>
```

le custom PROPIETIES vs $variabili

1. cascading difference
2. le $ variabili NON sono modificabili dai devtools, mentre le custom PROPIETIES si

Variables $

visualizzazione da entrambi le 2 parti di codice (scss, css)

SCSS

```scss
$primary: darkorange;
$secondary: #bada55;

$special-border: 2px dashed $secondary;

#sample {
  color: $primary;
  background-color: $secondary;
}

.special {
  border: $special-border;
}

.unique {
  border: 2px solid $primary;
}
```

CSS

```css
#sample {
  color: darkorange;
  background-color: #bada55;
}

.special {
  border: 2px dashed #bada55;
}

.unique {
  border: 2px solid darkorange;
}
```

Nesting

è una funzione anche introdotta recentemente anche nel CSS (anche se non è compatibile con tutti i browser, evitare di farlo) oltre che al SCSS

UTILE PER INSEREIRE REGOLE DI STILE NON RIPETERE GLI STESSI PEZZI DI CODICE + VOLTE

SCSS

```scss
.special {
  border: $special-border;
  ul {
    li {
      background-color: beige;
      &.selected {
        background-color: brown;
      }
    }
  }
}
```

HTML

```html
<div class="special">
    <ul>
        <li>item1</li>
        <li class="selected">item2</li>
        <li>item3</li>
    </ul>
</div>
```

Nesting

visualizzazione da entrambi le 2 parti di codice (scss, css)

& VA A PRENDERE LA CATENA DEI GENITORI SUPERIORI AD ESSO

SCSS

```scss
.special {
  border: $special-border;
  ul {
    li {
      background-color: beige;
      &.selected {
        background-color: brown;
      }
    }
  }
}
```

CSS

```css
.special {
  border: 2px dashed #bada55;
}
.special ul li {
  background-color: beige;
}
.special ul li.selected {
  background-color: brown;
}
```

Parent Selector &

è utile al nesting per ad esempio evitare lo stesso selettore già definito precedentemente

SCSS

```scss
.warning {
  background-color: red;
  &:hover {
    background-color: orange;
  }
  &--urgent {
    color: purple;
  }
  #footer & {
    // a warning in the footer looks different
    background-color: plum;
  }
  & > & {
    // a warning in a warning
    border: 1px dotted black;
  }
}
```

HTML

```html
<div class="warning">careful</div>
<div class="warning--urgent">please be
careful</div>
<div class="warning">
    <span>some error caused</span><span
class="warning">another error</span>
</div>
<div id="footer">
    <div>some footer text</div>
    <div class="warning">footer warning</div>
</div>
```

Parent Selector &

visualizzazione da entrambi le 2 parti di codice (scss, css)

SCSS

```scss
.warning {
  background-color: red;
  &:hover {
    background-color: orange;
  }
  &--urgent {
    color: purple;
  }
  #footer & {
    // a warning in the footer looks different
    background-color: plum;
  }
  & > & {
    // a warning in a warning
    border: 1px dotted black;
  }
}
```

CSS

```css
.warning {
  background-color: red;
}
.warning:hover {
  background-color: orange;
}
.warning--urgent {
  color: purple;
}
#footer .warning {
  background-color: plum;
}
.warning > .warning {
  border: 1px dotted black;
}
```

Parent Selector &

SCSS

```scss
div, p {
  color: #f5ca0a;

  & & {
    border: 1px solid #f86706;
  }
}
```

CSS

```css
div, p {
  color: #f5ca0a;
}
div div,
div p,
p div,
p p {
  border: 1px solid #f86706;
}
```

Inheritance @extend

permette di utilizzare delle regole all'interno (template), per poi applicarle agli elementi da noi desiderati (EVITA DI RIPETERE DEL CODICE GIA SCRITTO!)

NON VIENE visualizzato nel CSS il %placeholder

SCSS

```scss
// placeholders don't appear in the .css file
%panel {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

.info {
  @extend %panel;
  background-color: wheat;
}
#notification {
  @extend %panel;
  background-color: beige;
}
```

CSS

```css
#notification, .info {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

.info {
  background-color: wheat;
}

#notification {
  background-color: beige;
}
```

usiamo il grouping per produrre codice DRY

Inheritance @extend

CERCARE di ridurre al minimo le ripetizioni inutili di codice

SCSS

```scss
#footer > .special-info {
    @extend .info;
    color: $primary;
}
```

CSS

```css
#notification, .info, #footer > .special-info {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

.info, #footer > .special-info {
  background-color: wheat;
}

#footer > .special-info {
  color: darkorange;
}
```

Mixins @mixin @include

ci permette di duplicare del codice
a differenza dell'@extend NON ha un approccio DRY

quindi non sarà un selettore di grouping!!!!!!!!!

SCSS

```
@mixin panel {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

.info {
  @include panel;
  background-color: wheat;
}

#notification {
  @include panel;
  background-color: beige;
}
```

CSS

```
.info {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
  background-color: wheat;
}

#notification {
  border-radius: 5px;
  border: 1px solid brown;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
  background-color: beige;
}
```

Mixins @mixin @include

possiamo usare delle funzioni che ci permettono di sovrascrivere delle proprietà delle $variabili

SCSS

```scss
@mixin panel($border-color: brown, $bg-color:
wheat, $border-radius: 5px) {
  border-radius: $border-radius;
  border: 1px solid $border-color;
  background-color: $bg-color;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

.info {
  @include panel;
}
#notification {
  @include panel($bg-color:beige,
$border-radius:10px);
}
```

CSS

```css
.info {
  border-radius: 5px;
  border: 1px solid brown;
  background-color: wheat;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}

#notification {
  border-radius: 10px;
  border: 1px solid brown;
  background-color: beige;
  margin: 10px auto;
  box-shadow: 1px 1px 6px -1px black;
}
```

Mixins vs @extend

differenze e punti negativi e positivi di entrambi i metodi

Mixins
- Compiled CSS code is not DRY; same CSS is repeated for every class
- Generated CSS file is larger
+ Flexible: they accept arguments

@extend
- Not flexible: doesn't accept arguments
+ DRY compiled code
+ Creates semantic relationships between selectors
- Couples selectors together

Recommendation
Use @extend for same-for-a-reason
Use @mixin for same-just-because

Partials _ and Modules @use

definiamo un nome simile come a quello di un file, dove possiamo definire una serie di regole su differenti file (come se fosse una libreria)

```
_borders.scss
$round-borders: 5px;
$circle-borders: 50%;

@mixin round-border($border-radius:
$round-borders) {
  border-radius: $border-radius;
  border: 1px solid black;
}
```

```
main.scss
@use 'borders';

.btn {
  @include borders.round-border(10px);
}

.circle {
  background-color: orange;
  width: 20px;
  height: 20px;
  border-radius: borders.$circle-borders;
}
```

```
partials
- filename starts with an _
- won't generate a .css file
- only used by other files
```

Partials _ and Modules @use

main.css

```css
.btn {
  border-radius: 10px;
  border: 1px solid black;
}

.circle {
  background-color: orange;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}
```

index.html

```html
<div class="btn">click me</div>
<div class="circle"></div>
```

note: @import is the old way of doing **@use**
Don't use @import

Operators

permettono di modificare i valori degli attributi facendo delle operazioni

la comodità e che i calcoli vengono fatti una volta nella macchina dello sviluppatore andando a mettere i valori istantaneamente, e il browser degli utenti evitano di fare il calcolo ogni volta che accedono alla pagina

SCSS

```
$container-width: 800px;
$fraction: 1/3;

.container {
  width: $container-width;
  margin: 0 auto;
  .left {
    float: left;
    background-color: gold;
    width: $container-width * $fraction;
  }
  .right {
    float: right;
    background-color: darkkhaki;
    width: $container-width * (1-$fraction);
  }
}
```

CSS

```
.container {
  width: 800px;
  margin: 0 auto;
}

.container .left {
  float: left;
  background-color: gold;
  width: 266.6666666667px;
}

.container .right {
  float: right;
  background-color: darkkhaki;
  width: 533.3333333333px;
}
```

Built-In Modules

SCSS

```scss
@use 'sass:color';
.strange {
  $mixed: color.mix($primary, $secondary, $weight: 50%);
  background-color: $mixed;
  &:hover {
    background-color: lighten($mixed, 20%);
  }
  p {
    background-color: color.adjust($mixed, $hue: 35);
  }
}
```

CSS

```css
.strange {
  background-color: #ddb32b;
}

.strange:hover {
  background-color: #ebd383;
}

.strange p {
  background-color: #9fdd2b;
}
```

Built-In Modules

scss
```scss
@use "sass:map";

$colors: (
  primary: #007bff,
  secondary: #6c757d,
  success: #28a745,
  danger: #dc3545,
  warning: #ffc107
);

$primary: map.get($colors, primary); // $color will be #007bff

.advertising {
  color: $primary; // will set color to the primary color defined in the map
}
```

Functions

SCSS

```scss
// converts pixel values to rem values based
// on a base font size (default: 16px)
@function px-to-rem($px, $base-font-size: 16px) {
  @return $px / $base-font-size * 1rem;
}

.element {
  font-size: px-to-rem(24px);
  margin: px-to-rem(32px) 0;
}
```

CSS

```css
.element {
  font-size: 1.5rem;
  margin: 2rem 0;
}
```

@function
custom functions for reusable style logic

Control directives

@if
allows conditional styling based on specified conditions

@else
used with @if to provide an alternative styling when the condition is not met

@for
generates CSS rulesets dynamically based on a specified range or list

@each
iterates over lists or maps and applies styles accordingly

@while
executes a block of styles repeatedly while a condition is true

Control directives

SCSS

```scss
// iterating a list
$colors: red, green, blue;
@each $color in $colors {
  .color-#{$color} {
    color: $color;
  }
}
```

CSS

```css
.color-red {
  col0or: red;
}

.color-green {
  color: green;
}

.color-blue {
  color: blue;
}
```

Control directives

SCSS

```scss
// iterating a map
$colors: (
  primary: #3498db,
  secondary: #2ecc71,
  accent: #e74c3c
);

@each $name, $color in $colors {
  .color-#{$name} {
    color: $color;
  }
}
```

CSS

```css
.color-primary {
  color: #3498db;
}

.color-secondary {
  color: #2ecc71;
}

.color-accent {
  color: #e74c3c;
}
```

Your turn

1.Mix it up

- Write a mixin that uses another mixin that uses yet another mixin
- All 3 mixins should accept parameters and do something useful
- Create a complete page with a few SCSS features and variables
- Use the 3 mixins that you created in a useful way in the page
- Submit your SCSS, CSS and HTML files as well as any files used for generating the SCSS

Bonus

2.Of light and darkness

- Create a complete page with at least 3 styled page elements
  - e.g a page with a list, a table and a form with inputs and buttons
- Use SCSS variables, the parent selector, mixins or @extend and color functions to do the following
  - The page should have two 'themes', light and dark
  - You may use the classes on body 'light' or 'dark' to change themes
  - Use SCSS to generate the themes dynamically
  - Try to generate as much as possible changing only 2 or 3 main color values
- Submit your SCSS, CSS and HTML files as well as any files used for generating the SCSS

References: sass:color, Theming with Sass tutorial

References

SassMeister - The Sass Playground

Install Sass

Sass Basics

References

Placeholder Selectors

@extend

@mixin and @include

@use

Built-In Modules

References

An Introduction to Sass and SCSS

Intro to Sass. DRY up CSS with variables

Introduction to Sass/SCSS and Less

Less CSS