ACTIVITY 1: SASS

Presentation

Sass (Syntactically Awesome Style Sheets) is a powerful CSS extension language that enhances the capabilities of traditional CSS with advanced features like variables, nesting, mixins, and functions. It allows developers to write cleaner, more maintainable, and efficient styles, helping streamline complex styling tasks. By organizing styles in modular and reusable ways, Sass significantly reduces code duplication. It's compatible with all versions of CSS, making it easy to integrate into any project. Sass has become a popular tool among front-end developers for building scalable and robust stylesheets.

Skills

Generals

- Structure the content of a web page and define the visual design of an HTML page and layout

Competencies of the *disseny d'interfices web* subject

- CP 03: Prepare and optimize multimedia content to insert into web interfaces and perform visual animations and interactives

Objectives

The objectives of this activity are:

Understand the use of variables, nesting, and mixins in Sass.

Explore partials and imports for organizing code.

Learn the importance of extending and using functions to improve reusability.

Practice conditional styling with Sass.

Description of the activity

This activity consists of carrying out 4 activities.

Exercise 1. Using variables

Create a new file called “_variables.scss” in your Sass project.
Define variables for common colours (e.g., primary color, secondary color) and font sizes.
Import this file into style.scss and use the variables to style the .name and .role elements in the card.

Exercise 2. Nesting

Open the main “style.scss” file and style the .profile-card and its child elements with nested syntax.

Exercise 3. Creating and using mixins

Create a “mixins.scss” file, define a border-radius mixin, and apply it to the .avatar element.

Exercise 4. Extending Classes

Define a “.text-style” class in “_mixins.scss” with text styling.
Apply the “.text-style” to both .name and .role using @extend.

Exercise 5. Using Partials and Imports.

Organize the existing Sass files into partials (e.g., “_variables.scss, _mixins.scss”).
Import these partials into style.scss using @import.

Exercise 6. Adding Sass Functions.

Define a Sass function to darken or lighten a color based on a parameter.
Use this function to style the “.avatar” border.

Exercise 7. Conditional Styling with @if and @else.

Add conditional styling to the card, e.g., applying different padding for large screens.

Requirements

1. Introduce variables to manage common values.
2. Practice nesting to simplify code.
3. Understand and use mixins to apply reusable styles.
4. Use the @extend directive to inherit styles.
5. Organize code using partials and imports.
6. Use functions to manipulate values dynamically.
7. Add conditional styling to the card, e.g., applying different padding for large screens.

Related readings and resources

Classroom Resources:

- 3.1 SASS

Evaluation criteria

1. Variable Usage
    - Uses variables effectively for colors, font sizes, and other styles; all variables are relevant, organized, and used consistently throughout the project.Use of flexbox
2. Nesting
    - Applies Sass nesting logically, with clean and readable structure; avoids over-nesting and keeps code easy to follow.
3. Mixins and Extending
    - Creates efficient, reusable mixins and applies them correctly; uses @extend appropriately to reduce repetition and improve organization.Visual design & spacing.
4. Code Organization
    - All Sass code is well-organized into partials with logical imports; files are clearly named and each partial has a specific, consistent purpose.
5. Functionality and Conditionals
    - Demonstrates understanding of functions and conditionals; uses custom Sass functions and conditional styling to enhance reusability and responsiveness.

Format and date of delivery.

The final delivery will be made in "Classroom".

Delivery format: in a zip, rar… with the files HTML and CSS

Nomenclature: SurnamesNom_DIW_UT3A1.rar

Note: Intellectual property

It is often unavoidable, when producing a multimedia work, to make use of resources created by third parties people. It is therefore understandable to do it within the framework of a practice of the *desenvolupament d'aplicacions web*, as long as this is clearly documented and does not involve plagiarism in practice.

Therefore, when presenting a practice that makes use of external resources, it must be presented along with it a document detailing all of them, specifying the name of each resource, its author, the place where it was obtained and its legal status: whether the work is protected by copyright or under some other license of use (Creative Commons, GNU license, GPL...). The student must ensure that the license does not prevent specifically its use within the framework of practice. If the information is not found corresponding must assume that the work is protected by copyright.