Phone: +91 8882618533
Email: info@geekinstitute.org
Website: www.geekinstitute.org

THE GEEK INSTITUTE OF CYBER SECURITY

Regd. By: E-Max India
Centre Code: EMAX/EK80606
(Recognized By Govt. Of India)

(Building Futures Through Digital Knowledge and Innovation)

Syntactically Awesome Style Sheets (Sass) (01 Month)

Syllabus

1: Introduction to Sass & Modern CSS Workflows
- ✓ Overview of CSS Preprocessing and the need for Sass
- ✓ Sass architecture in large-scale projects
- ✓ Differences between Sass (indented syntax) and SCSS (CSS-like syntax)
- ✓ Real-world use cases: maintainable design systems, component libraries
- ✓ Setup using Dart Sass (recommended compiler)
- ✓ Project structure: base/, layout/, components/, themes/, utils/, vendors/

2: Variables & Dynamic Styling
- ✓ Defining variables for colors, spacing, breakpoints, typography
- ✓ Scoped variables and the !default flag
- ✓ Creating dark/light theme systems using variable maps
- ✓ Global design tokens architecture using @use with !default strategy

3: Nesting Rules for Scalability
- ✓ Logical and semantic nesting techniques
- ✓ Avoiding over-nesting: best practices (max 3 levels)
- ✓ Parent selector (&) usage: states, pseudo-classes, modifiers
- ✓ Combining nesting with BEM (Block Element Modifier) methodology
- ✓ Nesting with media queries, feature queries, and support rules

4: Modularization with Partials, @use, and @forward
- ✓ Creating reusable partials and loading them via @use
- ✓ Using @forward to expose partials as public APIs
- ✓ Building a scalable design system using module hierarchy
- ✓ Overriding variables safely with with clause
- ✓ Namespacing and aliasing modules

5: Mixins & Reusability Patterns
- ✓ Defining dynamic, parameterized mixins
- ✓ Variable arguments (...args) and default fallbacks
- ✓ Creating powerful utility mixins: clearfix, responsive spacing, media breakpoints
- ✓ Mixin composition with @content block injection
- ✓ Difference between mixins and functions in code reuse

6: Inheritance and Code Sharing with @extend
- ✓ Understanding selector inheritance via @extend
- ✓ Limiting specificity and duplication issues
- ✓ Abstract placeholders using %placeholder selectors
- ✓ When to prefer @extend over mixins—and when not to

7: Writing Custom Functions with @function
- ✓ Returning computed values dynamically
- ✓ Creating utility functions: px to rem converter, color contrast calculators
- ✓ Best practices: validation, performance, and naming conventions
- ✓ Function composition: nesting one function within another

8: Advanced Interpolation Techniques
- ✓ Dynamic class names, property names, and values using #{}
- ✓ Combining interpolation with maps and lists
- ✓ Conditional interpolation for theme-specific builds

9: Built-in Color Functions & Design Tokens
- ✓ Advanced color manipulation with:
  - adjust-hue(), saturate(), desaturate()
  - mix(), darken(), lighten(), transparentize()
- ✓ Creating accessible color systems dynamically
- ✓ Using color functions in theme generators

10: Advanced Arithmetic & Operators
- ✓ Logical operators: ==, !=, >, <, >=, <=, and, or, not
- ✓ Arithmetic operators for spacing and layout logic
- ✓ Real-world use cases: grid systems, font scaling, and spacing strategies

11: Core Built-in Functions
- ✓ **Number Functions**
  - abs(), ceil(), floor(), round()
  - max(), min(), comparable()
  - percentage(), random(), unit(), unitless()

- ✓ **String Functions**
  - quote(), unquote()
  - str-index(), str-insert()
  - str-length(), str-slice()
  - to-upper-case(), to-lower-case()
  - unique-id() – ideal for dynamic animations
- ✓ **List Functions**
  - length(), nth(), set-nth()
  - join(), append(), index()
  - list-separator(), is-bracketed()
- ✓ **Selector Functions**
  - selector-nest(), selector-append(), selector-replace()
  - is-superselector(), simple-selector(), selector-extend()
- ✓ **Map Functions**
  - map-get(), map-set(), map-merge()
  - map-remove(), map-keys(), map-values()
  - map-has-key()
- ✓ **Introspection Functions**
  - variable-exists(), global-variable-exists()
  - mixin-exists(), function-exists()
  - type-of(), inspect()

12: Control Flow & Programmatic Logic

- ✓ **@if / @else**
  - Writing logic-based style conditions
  - Complex use cases: accessibility styles, RTL support
- ✓ **@for Directive**
  - Looping through index ranges
  - Creating utility spacing or grid classes
- ✓ **@each Directive**
  - Looping over maps and lists
  - Dynamic theming and modular class creation
- ✓ **@while Directive**
  - Recursive logic and custom framework-like behavior
  - Caution with infinite loops

13: Layout Helpers & Responsive Design

- ✓ Responsive mixins using $breakpoints maps
- ✓ Building container, row, and column helpers
- ✓ Mobile-first media query architecture
- ✓ Fluid typography and spacing with calc() and clamp() integration

14: Media Queries and the @at-root Directive

- ✓ Scoping and structuring nested media queries
- ✓ Avoiding specificity issues using @at-root
- ✓ Creating modular media query systems inside mixins
- ✓ Integration with design tokens for adaptive UIs

15: Final Projects and Professional Integration

- ✓ **SCSS Component Library**: Create a reusable and scalable UI toolkit
- ✓ **Dark/Light Theme Switcher** using maps, functions, and interpolation
- ✓ **Responsive Grid System** powered by mixins and loops
- ✓ **Sass-based Design System** for a web application
- ✓ Integration with **React/Vue projects** (Sass Modules)
- ✓ Git versioning, branch-based development, and SCSS file management

Bonus: Performance Optimization & Best Practices

- Output styles: nested, compressed, expanded
- Reducing compiled file size and unused CSS
- Naming conventions: BEM, SMACSS, or Atomic design
- Maintainable architecture using **7-1 pattern**
- Migrating legacy @import projects to @use