APUNTES: SCSS, GRID Y RESPONSIVE

SCSS (Sass)

¿Qué es SCSS?

SCSS es un preprocesador CSS que extiende las capacidades de CSS con características de programación.

Variables

```scss
// Definir variables
$color-primary: #3498db;
$font-size-base: 16px;
$spacing-unit: 8px;

body {
  color: $color-primary;
  font-size: $font-size-base;
}
```

Anidamiento (Nesting)

```scss
nav {
  background: #333;

  ul {
    list-style: none;

    li {
      display: inline-block;

      a {
        color: white;
        text-decoration: none;

        &:hover {
          color: $color-primary;
        }
      }
    }
  }
}
```

Mixins (Funciones reutilizables)

scss

```scss
// Definir mixin
@mixin flex-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

@mixin responsive-text($size) {
  font-size: $size;
  line-height: $size * 1.5;
}

// Usar mixin
.container {
  @include flex-center;
  @include responsive-text(18px);
}
```

Funciones

scss

```scss
// Funciones integradas
$color-base: #3498db;
$color-light: lighten($color-base, 20%);
$color-dark: darken($color-base, 20%);

// Función personalizada
@function calculate-rem($pixels) {
  @return $pixels / 16px * 1rem;
}

p {
  font-size: calculate-rem(18px);
}
```

Partials e Imports

scss

```scss
// _variables.scss
$color-primary: #3498db;

// _mixins.scss
@mixin button-style {
  padding: 10px 20px;
  border-radius: 5px;
}

// main.scss
@import 'variables';
@import 'mixins';

button {
  @include button-style;
  background: $color-primary;
}
```

Extend/Herencia

scss

```scss
%button-shared {
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}

.button-primary {
  @extend %button-shared;
  background: blue;
}

.button-secondary {
  @extend %button-shared;
  background: gray;
}
```

CSS GRID

Contenedor Grid Básico

```scss
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px; // 3 columnas fijas
  grid-template-rows: 100px 100px; // 2 filas fijas
  gap: 20px; // espacio entre elementos
}
```

Unidades Flexibles

```scss
.grid {
  display: grid;

  // fr = fracción del espacio disponible
  grid-template-columns: 1fr 2fr 1fr; // 3 columnas proporcionales

  // repeat() para columnas repetidas
  grid-template-columns: repeat(3, 1fr); // 3 columnas iguales

  // minmax() para tamaños mínimos y máximos
  grid-template-columns: repeat(3, minmax(200px, 1fr));
}
```

Auto-fill y Auto-fit

```scss
.responsive-grid {
  display: grid;

  // auto-fill: crea tantas columnas como quepan
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));

  // auto-fit: similar pero colapsa columnas vacías
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

  gap: 20px;
}
```

Grid Areas (Plantillas)

scss

```scss
.layout {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: auto 1fr auto;
  gap: 10px;
  min-height: 100vh;
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main { grid-area: main; }
.footer { grid-area: footer; }
```

Posicionamiento de Items

scss

```scss
.item {
  // Por líneas
  grid-column: 1 / 3; // desde línea 1 hasta línea 3
  grid-row: 2 / 4;

  // Usando span
  grid-column: span 2; // ocupa 2 columnas
  grid-row: span 3; // ocupa 3 filas

  // Shorthand
  grid-area: 2 / 1 / 4 / 3; // row-start / col-start / row-end / col-end
}
```

Alineación en Grid

scss

```scss
.grid-container {
  display: grid;

  // Alinear items dentro de sus celdas
  justify-items: center; // horizontal
  align-items: center; // vertical
  place-items: center; // ambos

  // Alinear el grid completo dentro del contenedor
  justify-content: center; // horizontal
  align-content: center; // vertical
  place-content: center; // ambos
}

.grid-item {
  // Alinear un item específico
  justify-self: end;
  align-self: start;
}
```

RESPONSIVE DESIGN

Media Queries Básicas

scss

```scss
// Mobile First approach
.container {
  padding: 10px;

  // Tablet
  @media (min-width: 768px) {
    padding: 20px;
  }

  // Desktop
  @media (min-width: 1024px) {
    padding: 30px;
  }

  // Large Desktop
  @media (min-width: 1440px) {
    padding: 40px;
  }
}
```

Breakpoints con SCSS

scss

```scss
// Definir breakpoints como variables
$breakpoint-mobile: 480px;
$breakpoint-tablet: 768px;
$breakpoint-desktop: 1024px;
$breakpoint-large: 1440px;

// Mixin para media queries
@mixin responsive($breakpoint) {
  @if $breakpoint == mobile {
    @media (min-width: $breakpoint-mobile) { @content; }
  }
  @else if $breakpoint == tablet {
    @media (min-width: $breakpoint-tablet) { @content; }
  }
  @else if $breakpoint == desktop {
    @media (min-width: $breakpoint-desktop) { @content; }
  }
  @else if $breakpoint == large {
    @media (min-width: $breakpoint-large) { @content; }
  }
}

// Uso
.element {
  font-size: 14px;

  @include responsive(tablet) {
    font-size: 16px;
  }

  @include responsive(desktop) {
    font-size: 18px;
  }
}
```

Grid Responsivo

scss

```scss
.grid-responsive {
  display: grid;
  gap: 20px;

  // Mobile: 1 columna
  grid-template-columns: 1fr;

  // Tablet: 2 columnas
  @media (min-width: 768px) {
    grid-template-columns: repeat(2, 1fr);
  }

  // Desktop: 3 columnas
  @media (min-width: 1024px) {
    grid-template-columns: repeat(3, 1fr);
  }

  // Large: 4 columnas
  @media (min-width: 1440px) {
    grid-template-columns: repeat(4, 1fr);
  }
}
```

Grid Automático Responsivo

scss

```scss
.auto-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(min(100%, 300px), 1fr));
  gap: 20px;

  // Se adapta automáticamente sin media queries
}
```

Tipografía Fluida

scss

```scss
// Con clamp() - tamaño fluido entre min y max
.title {
  font-size: clamp(1.5rem, 5vw, 3rem);
  // min: 1.5rem, preferred: 5vw, max: 3rem
}

// Con función SCSS
@function fluid-size($min, $max, $min-width: 320px, $max-width: 1200px) {
  $slope: ($max - $min) / ($max-width - $min-width);
  $base: $min - $slope * $min-width;

  @return clamp(#{$min}px, #{$base}px + #{$slope * 100}vw, #{$max}px);
}

h1 {
  font-size: fluid-size(24, 48);
}
```

Container Queries (moderno)

scss

```scss
.card-container {
  container-type: inline-size;
  container-name: card;
}

.card {
  padding: 1rem;

  @container card (min-width: 400px) {
    display: grid;
    grid-template-columns: 1fr 2fr;
    padding: 2rem;
  }
}
```

Layout Completo Responsivo

scss

```scss
.page-layout {
  display: grid;
  gap: 20px;
  padding: 20px;

  // Mobile
  grid-template-areas:
    "header"
    "nav"
    "main"
    "sidebar"
    "footer";
  grid-template-columns: 1fr;

  // Tablet
  @media (min-width: 768px) {
    grid-template-areas:
      "header header"
      "nav nav"
      "main sidebar"
      "footer footer";
    grid-template-columns: 2fr 1fr;
  }

  // Desktop
  @media (min-width: 1024px) {
    grid-template-areas:
      "header header header"
      "nav main sidebar"
      "footer footer footer";
    grid-template-columns: 200px 1fr 250px;
  }
}

.header { grid-area: header; }
.nav { grid-area: nav; }
.main { grid-area: main; }
.sidebar { grid-area: sidebar; }
.footer { grid-area: footer; }
```

Utilidades Responsive

scss

```scss
// Ocultar/mostrar elementos
.hide-mobile {
  display: none;

  @media (min-width: 768px) {
    display: block;
  }
}

.show-mobile {
  display: block;

  @media (min-width: 768px) {
    display: none;
  }
}

// Espaciado responsivo
@mixin spacing-responsive($property, $mobile, $tablet, $desktop) {
  #{$property}: $mobile;

  @media (min-width: 768px) {
    #{$property}: $tablet;
  }

  @media (min-width: 1024px) {
    #{$property}: $desktop;
  }
}

.section {
  @include spacing-responsive(padding, 20px, 40px, 60px);
  @include spacing-responsive(margin-bottom, 30px, 50px, 80px);
}
```

TIPS Y MEJORES PRÁCTICAS

SCSS

- Organiza tu código en partials (_variables, _mixins, _components)
- No anides más de 3-4 niveles de profundidad
- Usa variables para colores, tamaños y breakpoints
- Prefiere mixins sobre extends para mayor flexibilidad

Grid

- Usa auto-fit/auto-fill para grids verdaderamente responsivos
- Considera minmax() para evitar elementos muy pequeños
- Grid es excelente para layouts de página, flexbox para componentes
- Usa grid-template-areas para layouts complejos y legibles

Responsive

- Enfoque Mobile First (desde pantallas pequeñas hacia grandes)
- Usa clamp() para tipografía fluida
- Considera container queries para componentes verdaderamente modulares
- Testea en dispositivos reales, no solo en el navegador

Combinando Todo

scss

```scss
// Variables
$breakpoints: (
  'mobile': 480px,
  'tablet': 768px,
  'desktop': 1024px,
  'large': 1440px
);

// Mixin responsive con map
@mixin respond-to($breakpoint) {
  @media (min-width: map-get($breakpoints, $breakpoint)) {
    @content;
  }
}

// Grid component responsivo
.product-grid {
  display: grid;
  gap: clamp(1rem, 3vw, 2rem);
  grid-template-columns: repeat(auto-fill, minmax(min(100%, 280px), 1fr));
  padding: clamp(1rem, 5vw, 3rem);

  .product-card {
    background: white;
    border-radius: 8px;
    padding: 1.5rem;

    @include respond-to('tablet') {
      padding: 2rem;
    }
  }
}
```