SASS

EJERCICIO 1 (SELECTORES, ANIDACIÓN)

1. Con la adición de Sass a nuestro proyecto, sabemos que `.surveyor h2` y `.surveyor h2` a se pueden simplificar con anidamiento, pero no tenemos tiempo para refactorizarlo todavía. Deja un comentario que no se publicará en nuestro CSS compilado, recordándonos que lo arreglaremos más tarde.

```
.surveyor {
  border: 1px solid #ccc;
  padding: 20px;
}
.surveyor h2 {
  font-size: 18px;
}
.surveyor h2 a {
  color: green;
}
```

2. Nuestro documento tiene algunos comentarios de estilo CSS restantes para la organización. Dado que estos comentarios sólo son útiles para nosotros, vamos a modificarlos para que no se emiten después de compilar.

```
/* Headers */
h1 {
  font-size: 24px;
  font-weight: bold;
}
h2 {
  font-size: 18px;
}
/* Lists */
ol {
  list-style: decimal;
}
ul {
  list-style: disc;
}
```

3. Para modularizar nuestros estilos, hemos separado algunos de ellos en un parcial denominado `_settings.scss`. Importe la configuración parcial en application.scss, recordando que el subrayado y la extensión de archivo no son necesarios con las importaciones de Sass.

```
// application.scss
h1 {
  color: $color-base;
}
```

4. Ahora es el momento de refactorizar la clase `.surveyor` para aprovechar el anidamiento

```
.surveyor {
  border: 1px solid #ccc;
  padding: 20px;
}
.surveyor h2 {
  font-size: 18px;
}
.surveyor h2 a {
  color: green;
}
```

5. Hemos notado un problema con nuestra clase `.notice` - necesitamos anidar la clase `.alert` dentro de una manera que producirá `.notice``.alert` en lugar de `.notice` `.alert`.

```
.notice {
  background: yellow;
  border: 5px solid #000;
  padding: 20px;
}
.notice.alert {
  background: red;
  box-shadow: 0 0 10px red;
}
```

6. Utilizando el selector padre, modifica la propiedad `color: # 313131;` para los `a:hover` en `.notice a`

```
.notice {
  background: yellow;
  border: 5px solid #000;
  padding: 20px;
}
.notice a {
  color: #222;
}
```

7. Hemos encontrado un cambio en `.surveyor`, necesita un poco más de relleno dentro de `.factory`. Si anidamos este cambio en el interior del `.surveyor`, lo controlaríamos mejor.

```
.surveyor {
  border: 1px solid #ccc;
  padding: 20px;
}
.factory .surveyor {
  padding: 30px;
}
```

8. Alguien ha sido exagerado en su anidación: los estilos a deberían aplicarse a cualquier ancla en .notice en vez de anclas en .notice.alert. Ajuste el anidamiento.

```
.notice {
background: yellow;
border: 5px solid #000;
padding: 20px;
&.alert {
 background: red;
 box-shadow: 0 0 10px red;
 a {
   color: #222;
   &:hover {
     color: #313131;
   }
 }
}
```

EJERCICIO 2 (VARIABLES)

1. El valor hexagonal de color `#797979` ha aparecido bastante en nuestra hoja de estilos. Vamos a almacenar el color en una variable y reemplazar todas las instancias de la misma con esa variable.

```
.surveyor {
  border: 1px solid #797979;
  padding: 20px;
}
.surveyor h2 {
  color: #797979;
  font-size: 18px;
}
.notice {
  background: yellow;
  border: 5px solid #797979;
  padding: 10px;
}
```

2. Limpia el siguiente código del valor repetido de padding. Mantenlo modular, la variable (`$mod-padding`) que pongas deberás asegurarte que no sobrescriba una variable del mismo nombre en cualquier lugar.

```
.modal {
  background: #fff;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
  padding: 10px;
}
.modal-title {
  border-bottom: 1px solid #ccc;
  font-size: 24px;
  padding: 10px;
}
.modal-action {
  background: purple;
  display: block;
  padding: 10px;
}
```

3. Un cambio de última hora, nos obliga a cambiar el padding para los distintos componentes es de `20px 10px 15px` en lugar de 10px. Altera el valor de `$mod-padding` para reflejar esta situación.

```
$modal-padding: 10px !default;
.modal {
  background: #fff;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
  padding: $modal-padding;
}
.modal-title {
  border-bottom: 1px solid #ccc;
  font-size: 24px;
  padding: $modal-padding;
}
.modal-action {
  background: purple;
  display: block;
  padding: $modal-padding;
}
```

4. Nuestra variable `$font-base` es útil cuando se establece la familia de fuentes en el cuerpo, pero estamos recibiendo un error al intentar utilizarlo en otro lugar. Refactorice los estilos a continuación para que $font-base se pueda utilizar en todas partes.

```
body {
  $font-base: "Helvetica Neue", Helvetica, Arial, sans-serif;
  background: #fff;
  font: normal 16px/1.5 $font-base;
  margin: 0;
  padding: 0;
}
.blueprint {
  font: bold 24px/1.3 Georgia, serif;
  text-align: center;
  span {
    font-family: $font-base;
  }
}
```

5. Nos gustaría añadir una forma sencilla de definir en qué bordes laterales aparecen. Crea una variable con el valor actual (izquierda) e interpola ese valor en cada ocurrencia de propiedad de borde.

```
.girder {
  border-left: 4px solid #ccc;
  h2 {
    font-size: 24px;
  }
}
.notice {
  border-left: 8px solid #797979;
  a {
    color: #222;
  }
}
```

EJERCICIO 3(INTERPOLACIÓN Y MIXIN)

1. Hemos identificado un conjunto de propiedades que comúnmente aparecen en nuestra hoja de estilos (a veces con pequeños ajustes). Vamos a simplificar el proceso de uso de estos valores añadiendo las líneas comentadas a un mixin llamado constructor, llamando a ese mixin en `.factoria` y `.rascacielo`.

```
// background: #fff;
// border: 1px solid #ccc;
// padding: 10px;
.factoria {
}
.rascacielo {
}
```

2. Modifica el anterior minix que hemos creado para cambiar el background color . Introduce en el minix un argumento que se llame `$bg` y pasale el valor `#fff` en `.factoria y #797979 .rascacielo`.
3. Hemos descubierto que el color `#fff` para el background se utiliza muy frecuentemente. Añade este valor, al anterior mixin, por defecto a `$bg` y refactoriza el código de los `@include`.
4. Con el código anterior, crea una nueva variable `$pad` con el fin de utilizar el minix para asignar `10 px` (valor por defecto) en `.factoria y 20px en .rascacielo`.
5. Una nueva petición acaba de entrar. El borde sólo debe aplicarse un lado. Añada un argumento `$lado` al minix del ejercicio anterior e interpolalo con la propiedad border. La clase `.factoria` a la izquierda y `.rascacielo` a la derecha.
6. En el siguiente ejemplo tenemos que pasar un argumento a bos-shadow que está separado por coma, pero tenemos un error con el número de argumentos. Modifica $shadow para que acepte el valor.

```
@mixin modal($shadow) {
  box-shadow: $shadow;
  border: 1px solid #ccc;
}
.modal {
```

```
    @include modal(inset 0 0 5px #000, 0 2px 5px #000);
  }
```

EJERCICIO 4 (AMPLIACIONES)

1. Parece que `.blueprint` y `.surveyor` tienen una serie de propiedades comunes. Recodifica `.surveyor` para que esta sección sea más eficiente.

```
.blueprint {
  background: blue;
  border-radius: 5px;
  margin-bottom: 15px;
  padding: 10px;
}
.surveyor {
  background: blue;
  border-radius: 5px;
  color: #fff;
  margin-bottom: 15px;
  padding: 10px;
}
```

2. Los desarrolladores están utilizando .notice en algunos lugares de la aplicación, `.error` en otros, y son incapaces de utilizar solamente uno o el otro. Extender los estilos .notice en una declaración `.error`.

```
.notice {
  background: yellow;
  border: 5px solid #000;
  padding: 20px;
  &.alert {
    background: red;
    box-shadow: 0 0 10px red;
  }
  a {
    color: #222;
    &:hover {
      color: #313131;
```

```
    }
  }
}
```

3. Refactorizar el código siguiente para extender el uso de las propiedades de la clase `.socket`, en caso de que tengamos que añadir los elementos más tarde.

```
.wrench .bolt {
  border-radius: 50%;
  padding: 15px;
  width: 30px;
}
.wrench {
  width: 100px;
}
.bolt {
  padding: 14px;
}
```

4. `.group` (comúnmente conocida como `clearfix`) se utiliza en toda nuestra aplicación para la limpieza de los floats. En el siguiente código, hemos extendido sus propiedades. Sólo queremos utilizar su código en las clases que tenemos. Refactoriza el código.

```
.group {
  zoom: 1;
  &:before,
  &:after {
    content: "";
    display: table;
  }
  &:after {
    clear: both;
  }
}
.factory {
  @extend .group;
  background: #fff;
}
```

5. Hemos descubierto una alteración a `.blueprint` en nuestra hoja de estilo y extender `.blueprint` con `.surveyor` está creando selectores adicionales en `.factory` que no son necesarios. Cree un placeholder llamado container para contener la propiedades compartidas y extiéndalo a `.blueprint` y `.surveyor` para eliminar el selector de `.factory` `.surveyor` extra.

```
.blueprint {
  background: blue;
  border-radius: 5px;
  margin-bottom: 15px;
  padding: 10px;
}
.surveyor {
  @extend .blueprint;
  color: #fff;
}

.factory {
  background: #fff;
  .blueprint {
    margin-bottom: 20px;
  }
}
```

EJERCICIO 5 (DIRECTIVAS, FUNCIONES)

1. Necesitamos encontrar el `height` de un `div` dado su width para tener un aspecto 16:9. Crea una función que se llame aspecto que se le pase el `width` y que devuelva el `height`.

```
// $ancho * 9 / 16

.intro {
  background: #000;
  width: 500px;
}
```

2. Crea una condición que según sea el tamaño en la clase `.cambio`. Si el tamaño es de `12px el font-family: Arial, sans-serif;` en otro caso `font-family: Helvetica, sans-serif;`.

```
$tamaño: 18px;

.cambio {
  font-size: $size;
}
```

3. Añade al anterior código otra condición de tal forma que si el tamaño es `24px, font-family: Georgia, serif;`
4. La directiva @if puede ser útil fuera de la clase `.cambio`, expande el ámbito. Par lo cual, inserta el código anterior en un mixin'in llamado familia al que se le pase `$tamaño` desde la clase `.cambio`.
5. El código que presentamos abajo es muy poco reutilizable. Recodificalo con la directiva `@each`.

```
// Simplificalo con @each

.tool-socket {
  background: url("socket.png") no-repeat;
}
.tool-wrench {
  background: url("wrench.png") no-repeat;
}
.tool-bolt {
  background: url("bolt.png") no-repeat;
}
```

6. El siguiente código funciona perfectamente para conseguir los números impares del 1 al 7. Vamos a simplificarlo, mediante `@while`.

```
@for $i from 1 through 7 {
  @if $i % 2 != 0 {
    .row-#{$i} {
```

```
        background: #ccc;
        height: $i * 10px;
      }
    }
  }
```

EJERCICIO 6 (OPERACIONES MATEMÁTICAS Y COLOR)

1. Para asegurar un espaciado entre las columnas, nuestra hoja de estilos tiene una variable `$gutter` reutilizable. Resta ese valor al ancho de `.factory` y `.highrise`.

```
$gutter: 20px;

.factory {
  background: #fff;
  margin-right: $gutter;
  width: 600px;
}
.highrise {
  background: #797979;
  margin-right: $gutter;
  width: 300px;
}
```

2. Su compañero de trabajo ha estado tratando de establecer el tamaño de fuente de elementos span dentro `.sign` a la mitad del tamaño de la fuente `.sign`, sin éxito.

```
.sign {
  font-size: 3.75em;
  font-weight: bold;
  padding: 20px 40px;
  span {
    font-size: 3.75em;
  }
}
```

```

```

3. El valor, del código anterior, `3,75em` es un valor que se utiliza en todo nuestra hoja de estilos. Utiliza una variable para guardar su valor y use esa variable dónde sea posible con el código de la solución anterior.
4. Modifica el siguiente código con el fin que `$font` tenga un valor entre comillas dobles (“ ”).

```
$theme: modern;
$font: "serif";
@if $theme == modern {
  $font: sans- + $font;
}

.sign {
  font-family: $font;
}
```

5. Redondea al número entero más próximo el tamaño de la fuente del `span`.

```
$size: 3.75em;

.sign {
  font-size: 3.75em;
  font-weight: bold;
  padding: 20px 40px;
  span {
    font-size: $size / 2;
  }
}
```

6. Las clases `.factory` y `.highrise` carecen de adaptación al contexto. Usa la siguiente formula target / context para dar a cada `width` un valor en porcentaje.

```
$context: 960px;

.factory {
```

```
  background: #fff;
  width: 600px;
}
.highrise {
  background: #797979;
  width: 300px;
}
```

7. Hemos guardado el color en hexadecimal en la variable `$color-base`, pero el equivalente (121,121,121) está usándose en la clase .modelo. Usa el atajo `rgba()` para hacer uso de la variable.

```
$color-base: #797979;

.modal {
  background: rgba(121, 121, 121, 0.75);
  border: 1px solid $color-base;
  padding: 20px;
}
```

8. Añade una pseudoclase hover al link con un 15% más brillante que `$color-link`

```
$color-link: #3097b4;

a {
  color: $color-link;
  text-decoration: underline;
}
```

9. Además, al código anterior añade al link una pseudoclase `:active` de tal forma que desatures un 25% un color un 15% más oscuro que el color-link. Y satures un 20% el color que has obtenido en el ejercicio anterior en la pseudoclase `:hover`.
10. No nos gusta el anterior resultado sobre la pseudoclase hover. Borra todas las funciones y utiliza la función `invert`.