[Lesson 1]

Roi Yehoshua 2018

[DAN.IT]
EDUCATION

Our targets for today

- CSS preprocessors
- SASS
- CSS methodologies
- BEM

[CSS preprocessor]

- ➔ Web pages have grown to be complex and big. As such using only what CSS allows is not enough, so a number of solutions were born. One of them is CSS preprocessors that propose a CSS dialect that can be compiled into native CSS

- ➔ The most popular CSS preprocessors are:
  - → SASS
  - → Less
  - → Stylus

- ➔ CSS preprocessor includes tools aimed to improve the creation of webpage styles
- ➔ Such tools as SASS are extended versions of CSS with many new useful features that allow writing styles faster, and at the same time keeping the code clean

[SASS]

- ➔ SASS cannot be run in a browser. Instead, it must be first compiled to CSS
- ➔ SASS can be compiled via a special desktop application, console, or as a part of automation toolkits like Gulp, Grunt, Webpack or Rollup.js

- ➔ SASS comes in two versions
  - → .SASS - uses own syntax
  - → .SCSS - allows CSS-like syntax

```
SCSS syntax:

a.link {
 color: magenta;
}
```

```
SASS syntax:

a.link
 color: magenta
```

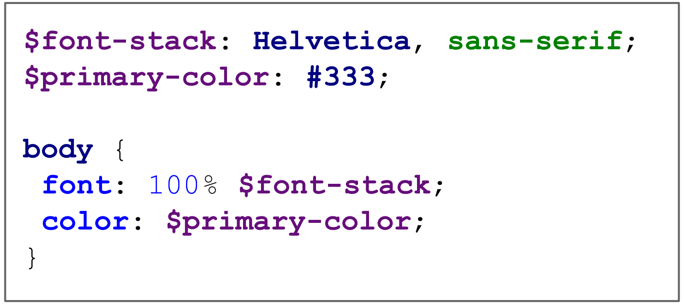
SASS Features

- SASS proposes a number of great features, like
  - Variables
  - Nesting
  - Mixins
  - Partials
  - Inheritance
  - Operators

[Variables]

- ➔ SASS variables are similar to the JavaScript ones
- ➔ You can store things like color, font, padding or margin values, and many other
- ➔ It is possible then to use variables instead of writing the value manually each time it is used

- ➔ Sass uses the $ symbol to declare a variable



```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
 font: 100% $font-stack;
 color: $primary-color;
}
```

[Nesting]

- ➔ SASS allows nesting selector one into another
- ➔ When nested, selectors become part of a hierarchy that will be compiled into one single CSS selector.

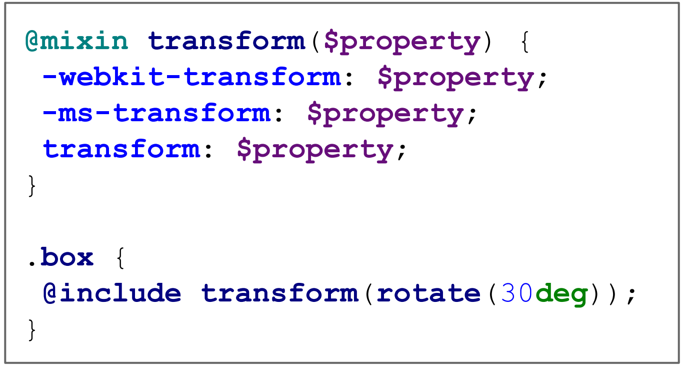
```
nav {
 ul {
   list-style: none;
 }

 li {
   display: inline-block;
 }
}
```

Will be compiled to

```
nav ul {
 list-style: none;
}

nav li {
 display: inline-block;
}
```

[Mixins]

- ➔ Mixins allow to group up CSS declarations to reuse them later in the code
- ➔ Additionally, a value can be passed to the mixins, allowing creation of very flexible reusable mixins



```
@mixin transform($property) {
 -webkit-transform: $property;
 -ms-transform: $property;
 transform: $property;
}

.box {
 @include transform(rotate(30deg));
}
```

Partials

- ➔ Partial is a SASS file named with a leading underscore, for example, _footer.scss
- ➔ Such files won’t be compiled to corresponding CSS files
- ➔ Instead, they are intended to be imported via the @import directive from other SASS files

- ➔ Partials allow dividing one large file into separate smaller files
- ➔ It helps keep code clean and implement the modular approach

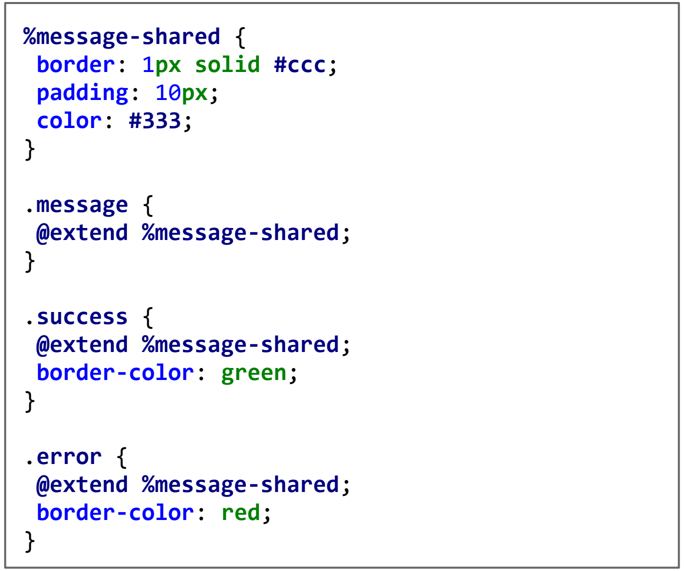
- ➔ A file named app.scss could import such partials like

```
_header.scss
_content.scss
_footer.scss
```

- ➔ And will be compiled into a single file named app.css

Inheritance

- ➔ @extend directive allows you to share a set of CSS properties from one selector with another
- ➔ It goes hand in hand with placeholder classes
- ➔ A placeholder class starts with % symbol and only prints when it is extended



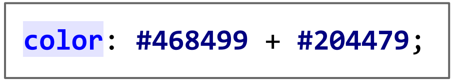
```scss
%message-shared {
 border: 1px solid #ccc;
 padding: 10px;
 color: #333;
}

.message {
 @extend %message-shared;
}

.success {
 @extend %message-shared;
 border-color: green;
}

.error {
 @extend %message-shared;
 border-color: red;
}
```

[Operators]

- Another useful feature of SASS is operators
- They allow using basic math with variables and values
- SASS supports following math operators

+	addition
-	subtraction
*	multiplication
/	division
%	remainder

- They should be used with compatible units only
- 5px + 5px will result in 10px
- but 5px + 2em will result in an compilation error
- It can also be done with colors:

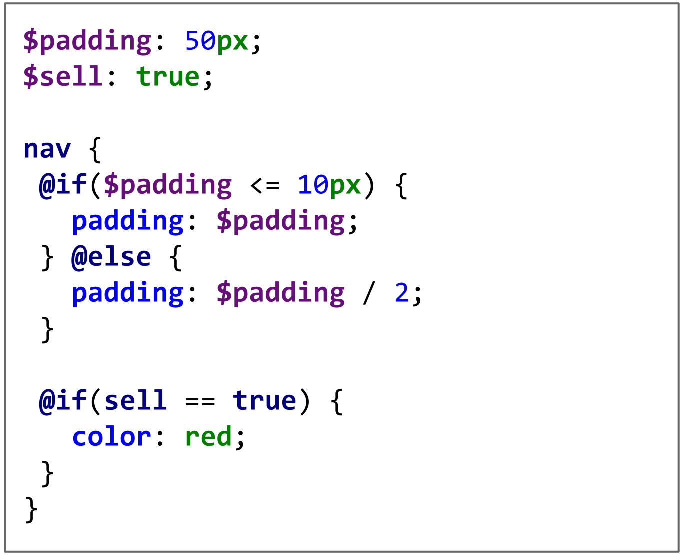


```
color: #468499 + #204479;
```

[Operators]

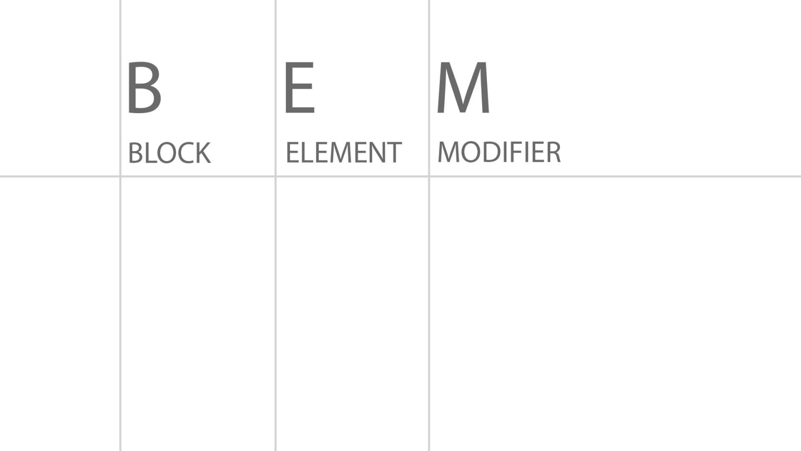
➔ SASS supports following equality, comparison, and logical operators

==	equals
!=	not equals
>	greater than
>=	greater than or equals to
<	less than
<=	less than or equals to
and	logical and. will be true if both operands are true
or	logical or. will be true if one of the operands is true
not	logical not. will be true if the operand is not true



```
$padding: 50px;
$sell: true;

nav {
 @if($padding <= 10px) {
   padding: $padding;
 } @else {
   padding: $padding / 2;
 }

 @if(sell == true) {
   color: red;
 }
}
```

BEM



- ➔ BEM stands for Block, Element, Modifier
- ➔ It is a component-based approach to web development. The idea behind is to divide the user interface into independent blocks. This makes interface development easy and fast even with a complex UI

[BEM]

- ➔ If written right, BEM allows us to connect CSS with HTML in a really understandable tandem
- ➔ Reading HTML code written in BEM should give you a hint how CSS structure looks like
- ➔ And reading CSS code written in BEM should give you a hint on what the HTML code is and what it does
- ➔ This is achieved by naming and structural conventions

- ➔ Three pillars of BEM is
  - → Block - component containers
  - → Element - subparts of a container
  - → Modifier - modifications of container or elements

[Block]

- ➔ BEM Block is defined as a functionally independent page component that can be reused
- ➔ Block is identified by class name, not an ID or tag name
- ➔ It answers the question "what is it?"
- ➔ Examples of such blocks:
    - → button
    - → header
    - → menu
    - → article
- ➔ A block can reside inside another block
- ➔ There can be any number of nesting levels

Element

- BEM Element is a composite part of a block, that can't be used separately from it
- Element selector is written as block-name__element-name. The element name is separated from the block name with a double underscore (__).
- It answers the question "What does this part of block represent?"
- Examples of elements
  - name of the button
  - price of a product in a product view
  - an element of the navigation menu
- Elements can be nested inside each other
- You can have any number of nesting levels
- An element is always part of a block, not another element
- Element names can't define a hierarchy such as block__elem1__elem2, we should use block__elem1 and block__elem2, even if element 2 is nested inside element1

[Modifier]

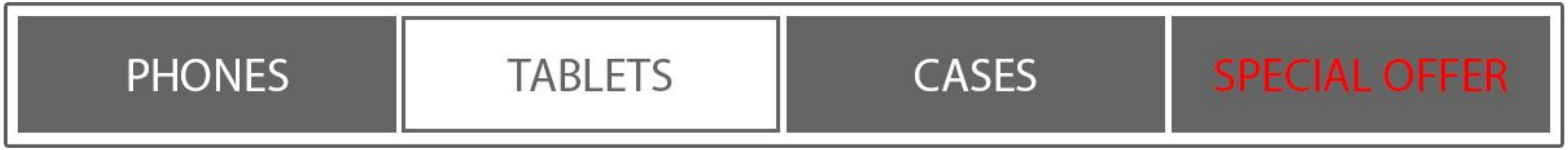
- BEM Modifier is an entity that defines the appearance, state, or behavior of a block or element
- The modifier name is separated from the block or element name by two dashes (--).
- It answers questions like "What size?", "Which theme?", "what color?".
- There are two types of modifiers
  - Boolean - used when only the presence or absence of the modifier is important, and its value is irrelevant, e.g. button--disabled.
  - Key-value - used when the modifier value is important. For example, "a menu with the islands design theme": menu--theme--islands.

Modifier

- ➔ Examples of boolean modifiers
  - → Disabled button
  - → Focused search field
  - → Selected menu item

- ➔ Examples of Key-value modifiers
  - → Size of the button
  - → Type of product in a shop (normal, promoted, sell, etc)
  - → Visual theme of the element (usual, Christmas, promotion, etc)

Example



```html
<nav class="top-menu">
  <a href="/phones" class="top-menu__item">Phones</a>
  <a href="/tablets" class="top-menu__item top-menu__item--selected">Tablets</a>
  <a href="/cases" class="top-menu__item">Cases</a>
  <a href="/special" class="top-menu__item top-menu__item--hot">Special Offer</a>
</nav>
```

Control questions

1. What is a CSS preprocessor?
2. Name issues that SASS can solve
3. How can we install and use SASS?
4. Name six SASS features
5. What is BEM?
6. Name meaning of every letter in BEM abbreviation
7. Using an example of a navigation menu, explain how BEM should be implemented (which class names shall you use for HTML elements)