**LESS**

* Dynamic stylesheet language
* Simple language that then compiles down to CSS
* Introduces programming features to CSS, and it just looks like CSS
* Install LESS on server side using  
  + npm install -g less
* Command-line Usage  
  + **lessc less\_filename css\_filename**
  + **Example :** lessc styles.less styles.css
* Importing a less file in another  
  + @import "styles\_new";

**Creating variables**

* Create variables using **“@”** symbol
* @import "styles\_new";  
  @bg-color : #000;  
  @txt-color : #FFF;  
  @padding : 20px;  
  body {  
   background-color: @bg-color;  
  }  
    
  h1 {  
   color: @txt-color;  
  }  
    
  p {  
   color: @txt-color;  
   padding: @padding;  
  }

**Mixins**

* Supports repeatable sections of code
* Almost like functions which takes input parameters
* Can have default values and can be overloaded
* Example  
  + .border-color-radius-p(@size:5px, @color:#000) {  
     border-radius: @size;  
     -webkit-border-radius: @size;  
     -moz-border-radius: @size;  
     border: 1px solid @color;  
    }  
      
    .border-color-radius-p(@size:5px) {  
     border-radius: @size;  
     -webkit-border-radius: @size;  
     -moz-border-radius: @size;  
     border: 10px solid red;  
    }  
      
    p {  
     .border-color-radius-p(10px);  
    }
* **Using guards**  
  + .width(@size) when (isnumber(@size)) {  
     width: @size \* 2;  
    }  
      
    .width(@size) when (ispercentage(@size)) {  
     width: @size;  
    }  
      
    p {  
     border: 1px solid #000;  
     .width(20%);  
    }

**Nested rules**

The example below show how to convert css rules to less nested rules

**CSS without nested rules**

div {

border: 1px solid #000;

width: 20%;

}

div ul {

list-style: none;

}

div ul li {

color: #FF0000;

}

**CSS with nested rules**

div {

border: 1px solid #000;

width: 20%;

ul {

list-style: none;

li {

color: #FF0000;

}

}

}

**Using combinators**

ul {  
 list-style: none;  
}  
  
ul:hover {  
 list-style: circle;  
}

becomes

ul {  
 list-style: none;  
 &:hover {  
 list-style: circle;  
 }  
}

**Scope in less**

@color : blue;

div {

@color:#FF0000;

border: 1px solid @color;

width: 20%;

ul {

list-style: none;

li {

color: #FF0000;

}

}

}

p {

color:@color;

}

The variable @color defined with div will only be available within the block.

**String interpolation**

@img-folder : 'img/';

body {

background: url("@{img-folder}1.jpg");

}

**Using javascript**

@img-folder : 'img/';

@img-folder-upper : `@{img-folder}.toUpperCase()`;

body {

background: url("@{img-folder-upper}1.jpg");

}

**SASS**

* Another dynamic stylesheet language
* Install using npm  
  + npm install sass
* Install compiler using npm  
  + npm install -g node-sass
  + node-sass filename.scss filename.css

**Creating variables**

* Variable names in SCSS starts with $ symbol  
  + $variablename
  + $headerColor : #FF0000;  
    $paraColor : green;  
    $fontSize : 26px;  
      
    h1 {  
     color: $headerColor;  
    }  
      
    p {  
     color:$paraColor;  
     font-size: $fontSize;  
    }
* Adding values to variables  
  + p {  
     color:$paraColor;  
     font-size: $fontSize+20;  
    width : (100% / 2 ) +25%;  
    }
* Adding helper functions with variables  
  + h1 {  
     color: lighten($headerColor, 25%);  
    }
  + Other example darken, invert, fade\_in, fade\_out
* Using variables as placeholders  
  + background-image: url($imgPath+"google\_search\_logo\_summer\_drawing\_26168\_602x339.jpg")
  + OR
  + background-image: url("#{$imgPath}google\_search\_logo\_summer\_drawing\_26168\_602x339.jpg")
* Creating class names using variables  
  + $className : 'cls';  
      
    .#{$className}\_para {  
     border: 1px solid red;  
    }

**Rules**

* Nesting rules just like in LESS  
  + div {  
     border: 1px solid #000;  
     width: 20%;  
     ul {  
     list-style: none;  
     li {  
     color: #FF0000;  
     }  
     &:hover {  
     list-style: circle;  
     }  
      
     }  
    }

**@import**

* Can import both css and scss files  
  + @import "main.css";
  + @import "main.scss";

**@extend**

* Inherit styles from one another  
  + $btnColor : black;  
      
    $btnTxtColor : white;  
      
    .button {  
     background-color: $btnColor;  
     color: $btnTxtColor;  
    }  
      
    .button2 {  
     @extend .button;  
     width: 200px;  
    }

**@mixins**

* Like functions
* Can have parameters, defaults, overloads
* Example  
  + @mixin para\_cls {  
     font : {  
     size : 20px;  
     weight:bold;  
     }  
     border:1px solid red;  
     padding: 20px;  
    }  
      
    p {  
     @include para\_cls;  
    }
* With arguments and default values  
  + @mixin para\_cls($size:20px, $color:red) {  
     font : {  
     size : $size;  
     weight:bold;  
     }  
     border:1px solid $color;  
     padding: $size;  
    }  
      
    p {  
     @include para\_cls(40px, green);  
    }

**@functions**

* Used to create functional representations
* $body-width: 100%;  
    
  @function calculateWidth($cols) {  
   @return ($body-width / $cols);  
  }  
    
  .para\_cls {  
   width: calculateWidth(2);  
  }

**Control directives**

* **@if**  
  + $size : 16px;  
      
    p {  
     @if $size > 14px {  
     color : red;  
     }  
     @else if $size < 14px {  
     color : green;  
     }  
     @else {  
     color : orange;  
     }  
    }
* **@for**
  + $pagewidth : 100%;  
      
    @for $col from 1 through 4 {  
     .col#{$col} {  
     width : $pagewidth / $col;  
     }  
    }
* **@each**
  + @each $item in 25, 50, 75, 100 {  
     .width#{$item}p {  
     width: $item+%;  
     }  
    }
* **@while**
  + $cntr : 1;  
      
    @while $cntr < 4 {  
     .col#{$cntr} {  
     width:25% \* $cntr;  
     }  
     $cntr : $cntr + 1;  
    }