MŰEGYETEM 1782

Mobile and Web Development

Department of Automation and Applied Informatics

LESS, jQuery

Created: 1/2024/2025

Dr. Mohammad Saleem

msaleem@aut.bme.hu

LESS

Introduction

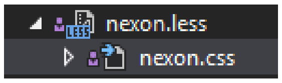
Less (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS. This is the official documentation for Less, the language and Less.js, the JavaScript tool that converts your Less styles to CSS styles.

Why LESS instead of CSS

- The aim of LESS is to create more dynamic and reusable CSS
- LESS extends CSS with new functions:
  - > Variables
  - > Mixins
  - > Rules embending
  - > Funtions and operators

Compiling LESS to CSS

- The souce goes to .less, but we are linking .css to the html page
- Compiling in VS 2015:
  - > With Web Compiler extension.
  - > https://visualstudiogallery.msdn.microsoft.com/3b329021-cd7a-4a01-86fc-714c2d05bb6c
- Under .less, the compiled .css can also be found
- To compile everything into one file, create a nexon.less file, where to import everything

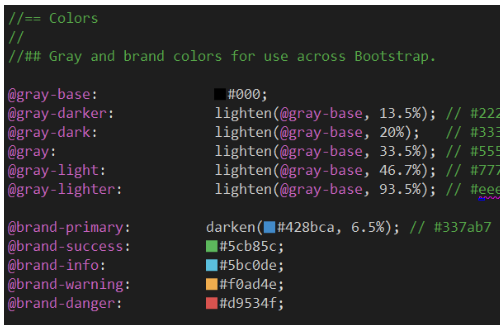


Compiling LESS to CSS involves converting your .less files into standard .css files that can be used by browsers. There are several ways to compile LESS:

- Using the Command Line with Node.js and the LESS Compiler
- Using Task Runners (Gulp/Grunt)
- Using Online LESS Compilers

Bootstrap with CSS or LESS?

- There is .less version with a **variables.less** to customize the theme



```
//== Colors
//
//## Gray and brand colors for use across Bootstrap.

@gray-base:              ■#000;
@gray-darker:            lighten(@gray-base, 13.5%); // #222
@gray-dark:              lighten(@gray-base, 20%);   // #333
@gray:                   lighten(@gray-base, 33.5%); // #555
@gray-light:             lighten(@gray-base, 46.7%); // #777
@gray-lighter:           lighten(@gray-base, 93.5%); // #eee

@brand-primary:         darken(🟦#428bca, 6.5%); // #337ab7
@brand-success:         🟩#5cb85c;
@brand-info:            🟦#5bc0de;
@brand-warning:         🟧#f0ad4e;
@brand-danger:          🟥#d9534f;
```

Using variables

- Values (like colors) can be defined only once and reused several times
- Creating a variable in colors.less:

```
@cyan-bg: #289FA6;
```

- Using it in form.less

```
    .default-value {
        margin-top: 5px;
        color: @cyan-bg;
        padding-left: 0;
```

```
     }
```

Defining

```
    //Layout sizes
    @nxn-page-min-width: 1200px;
    @nxn-page-min-height: 800px;
```

Usage

```
html {
    min-width: @nxn-page-min-width;
    min-height: @nxn-page-min-height;
    height: 100%;
}
```

Mixins

Mixins in LESS are reusable groups of CSS properties that you can include in other rules. They help you write DRY (Don't Repeat Yourself) code by reusing styles across multiple selectors.

```
helpers.less
.nxn-border(@color) {
border: 1px solid @color;
}
table.less
tr {
.nxn-border(@cyan-border);
}
```

```
// Define the mixin
.border-radius(@radius) {
  border-radius: @radius;
}

// Use the mixin
.button {
  .border-radius(10px);
  background-color: blue;
  color: white;
}
```

```
.box-shadow(@x: 0, @y: 0, @blur: 5px, @color: #000) {
  box-shadow: @x @y @blur @color;
}

.card {
  .box-shadow(2px, 4px, 8px, rgba(0, 0, 0, 0.3));
}
```

Embedding rules

- For simpler code, rules can be emended that will compile into a complex selector in CSS.

```
.form-horizontal {
    .form-group {
        padding-top: 5px;}
}
```

```
// Compiled code
.form-horiziontal .form-group{
          padding-top: 5px;
}
```

- In this case we do not use **&** (parent selector)

```
.form-horizontal {
    > form-group {
        padding-top: 5px;
    }
}
// Compiled code
.form-horiziontal > .form-group{
          padding-top: 5px;
}
```

> is a direct child combinator. By using > .form-group, you are styling only the direct children of .form-horizontal that have the .form-group class.

The compiled CSS will target .form-group elements that are direct children of .form-horizontal, ensuring that nested .form-group elements further down the hierarchy are not affected.

```
.form-horizontal {
    &.form-group {
        padding-top: 5px;
    }
}
// Compiled code
.form-horiziontal.form-group{
          padding-top: 5px;
}
```

In LESS, the & symbol refers to the current parent selector.
&.form-group means "append .form-group to .form-horizontal," resulting in the combined selector .form-horizontal.form-group.

```
.form-horizontal {
    &-group {
        padding-top: 5px;
    }
}
// Compiled code
.form-horiziontal-group{
           padding-top: 5px;
}
```

By using &-group, you're appending -group to .form-horizontal, creating a new class called .form-horizontal-group.

This is a common pattern in LESS and other preprocessors when you need to create related classes with consistent naming.

Complex example

```
@nxn-window-space: 20px;
.portal {
   &-sushi {
      display: block;
      width: 100%;
   }
   &-container {
      height: ~"calc( 100% - 282px - 46.5px - @{nxn-window-space} + 50px )";
      margin: @nxn-window-space auto;
      &-single {
         height: ~"calc( 100% - 46.5px - 2 * @{nxn-window-space} )";
      }
   }
}
```

Compiled code

```
.portal-sushi {
  display: block;
  width: 100%;
}
.portal-container {
  height: calc( 100% - 282px - 46.5px - 20px + 50px );
  margin: 20px auto;
}
.portal-container-single {
  height: calc( 100% - 46.5px - 2 * 20px );
}
```

Builtin function

- There are built in function to handle colors easier.

```
//Darker
@sushi-menu-selected-start: darken(@sushi-menu-bg-start, 5%);
// Lighter
@sushi-menu-selected-end: lighten(@sushi-menu-bg-end, 2%);
```

darken(@color, percentage): This function takes a base color and darkens it by the specified percentage.

lighten(@color, percentage): This function takes a base color and lightens it by the specified percentage.

Function names and operators

- You can carry out basic operation (+, -, *, /)

```
@sushi-menu-item-width: 180px;
.generate-positions(@n, @i: 0) when (@i < @n) {
    &.sushi-position-@{i} {
        left: (@i * @sushi-menu-item-width + 18px);
    }
    .generate-positions(@n, (@i + 1));
}
.generate-positions(4);
```

- **@sushi-menu-item-width: 180px;**: This variable sets the width of a sushi menu item.
- **.generate-positions(@n, @i: 0)**: This is a recursive mixin that creates positions for a series of items.
- **@n** is the total number of items for which positions will be generated.
- **@i** is the current iteration index (default is 0).

- **when (@i < @n)**: This guard ensures that the recursion continues until @i reaches @n.
- **left: (@i * @sushi-menu-item-width + 18px);**: This calculates the left position of each item, based on its index.

Compiled code:

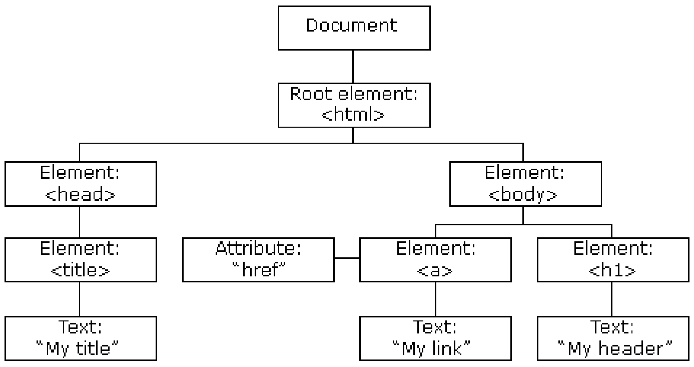
```
.sushi-position-0 {
  left: 18px;
}
.sushi-position-1 {
  left: 198px;
}
.sushi-position-2 {
  left: 378px;
}
.sushi-position-3 {
  left: 558px;
}
```

jQuery

jQuery is a lightweight, "write less, do more", JavaScript library.
The purpose of jQuery is to make it much easier to use JavaScript on your website.

Before we start: DOM – Document Object Model

- DOM is a model that represents the built-up of a page in a tree hierarchy. The root of the tree is the "Document" object.
- There are multiple versions (aka level) of DOM.
- The versions express the evolution of the DOM model.



DOM Level 1

- DOM Level 1 (October 1, 1998): This was the first standard. A complete model to represent HTML or XML documents
  - > It consists of two parts: Core and HTML
  - > The most important interfaces are introduced in this standard

    - Document,
    - Node,
    - Attribute,
    - Element,
    - Text

DOM Level 2

- DOM Level 2 (November 13, 2000.): Some new useful methods are added to the Core part, e.g. getElementById.
  - Events: methods for event handling (addEventListener, handleEvent)
  - Traversal and Range: Travers the DOM tree; create, insert, modify, delete parts of the tree.
  - Style: Dynamically modify CSS styles (getComputedStyle)

DOM Level 3 and 4

- DOM Level 3 (April 2004)
  - Some minor extensions that are not really supported
    - Keyboard related events
    - XPath 1.0 support
- DOM4 (November 19, 2015)
  - API is simplified
  - E.g.: easier to define events
  - https://www.w3.org/TR/2015/REC-dom-20151119/
- DOM Standard (rev. November 13, 2019)
  - https://dom.spec.whatwg.org/

The content of the Window object

- The window object represents an open window in a browser.
  - This is the HTML page itself that is displayed by the browser.
- Other important objects in the windows
  - frames,
  - location,
  - history,
  - navigator,
  - sreeen

https://www.w3schools.com/jsref/obj_window.asp

Getting the elements of the page

- Get elements by tag name
  - document.getElementsByTagName("img")
- Get elements by CSS class name:
  - document.getElementsByClassName()
- Get element by id:
  - document.getElementById(...)

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbytagname2

Modify the elements of the page

- The properties of an element can also be modified
- Let’s change the background color of the <body> tag

  document.body.style.backgroundColor = "#CCC";

Traversing the DOM

Traversing the DOM involves navigating through the elements and nodes of the Document Object Model (DOM) in a structured way to access, modify, or interact with them.

JavaScript provides several properties and methods to traverse the DOM tree:

- **parentNode**: Gets the parent node of the specified element.
- **childNodes**: Returns a live NodeList of all child nodes, including text nodes (whitespace).
- **children**: Returns an HTMLCollection of only the child elements (ignores text and comments).
- **firstChild / firstElementChild**: Gets the first child node or first element child.
- **lastChild / lastElementChild**: Gets the last child node or last element child.
- **nextSibling / nextElementSibling**: Gets the next node or the next element sibling.
- **previousSibling / previousElementSibling**: Gets the previous node or the previous element sibling.

```
function walkTheDOM(node, func) {
    // The function func is applied to the current node, which allows any operation to be performed
on that node (e.g., logging, modification, data extraction).
    // Execute the parameter function with the actual element
    func(node);
    // Take the first child
    node = node.firstChild;
    while (node) {
        // Recursive call (depth first search)
        walkTheDOM(node, func);
        // Take the next sibling is there is no more child node
        node = node.nextSibling;
    }
}
```

The walkTheDOM function is a recursive function that performs a depth-first traversal of the DOM, visiting each node and executing a callback function on it.

Modify DOM elements

```
window.onload = function () {
    // Traverse the elements the body node
    walkTheDOM(document.body, function (node) {
        if (node.className != null
              && node.className.indexOf("meta") != -1) {
            // Subscribe to the onclick event
            node["onclick"] = function (e) {
                e.target.style.backgroundColor = "yellow";
            }
        }
    });
};
```

It traverse the entire DOM tree starting from document.body once the window has loaded. It looks for any element that has a class name containing the substring "meta" and subscribes those elements to an onclick event that changes their background color to yellow when clicked.

Add html elements to the DOM – JS

```
function addHeading(text) {
        // #staff is the item that we append the new content to.
        var sc = document.getElementById('staff');
        // Dynamically create tags <h1>
        var newHeading = document.createElement('h1');
        var textNode = document.createTextNode(text)
        newHeading.appendChild(textNode)
        // Add to the DOM
        sc.appendChild(newHeading);
        //sc.insertBefore(newHeading, sc.firstChild);
}
```

onLoad()

Subscribe to the event that fires when the page is loaded

```
window.onload = function(){
    addHeading("Staff" );
}
```

DOM manipulation with JS

- The DOM can be traversed and manipulated purely with JS without any external class libraries as one can see in the previous examples
- Though, it requires a lot of coding

DOM manipulation tools, libraries

- **JQuery**
- Vue.js
- React
- Angular (typescript-based)

What is jQuery?

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code.
jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods

- Effects and animations
- AJAX

much simpler with an easy-to-use API that works across a **multitude of browsers**. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Document ready

- Document ready is one of the most important events of jQuery
- Document ready is a jQuery-specific event that is fired when the document got loaded
  - Not the whole window
  - Before the browser would start downloading the images and all other external resources
  - The same as DOMContentLoaded
    - Emulates DOMContentLoaded if it is not supported by the browser

How to link the jQuery JavaScript class library (from CDN)

```
<script src="http://code.jquery.com/jquery-2.2.4.js"> </script>
```

- If you remove the version from the URL it will always download the latest version)
  - Don't do this in production
    - breaking changes
  - Consider to use CDN or not
    - Some fallback mechanism is needed if CDN is not available

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

```
$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".
```

Handle the document ready event and log some text to the console

```
$(document).ready(function(){
    console.log("Document ready...");
});
```

Simple selectors

- Every element
  $("*")
- The element where the id is “staff”
  $("#staff")
  - The first one if there are multiple
- Elements that has a CSS class named “meta”:
  $(".meta")
- All the image tags
  $("img")

Compound selectors

- Spacing is important!
  - Tag and CSS:
    $("span.meta")
  - Tag and ID:
    $("div#staff")
  - ID or CSS:
    $("#staff, .meta")

```
$("div.highlight, span.note, p.special").css("border", "1px solid red");
```

This applies the border to <div> elements with the highlight class, <span> elements with the note class, and <p> elements with the special class.

Hierarchical selectors

- Descendents ( ):
  $("ul img")
  - This selects all <img> elements that are descendants of any <ul> element. It could be nested at any level within the <ul>.

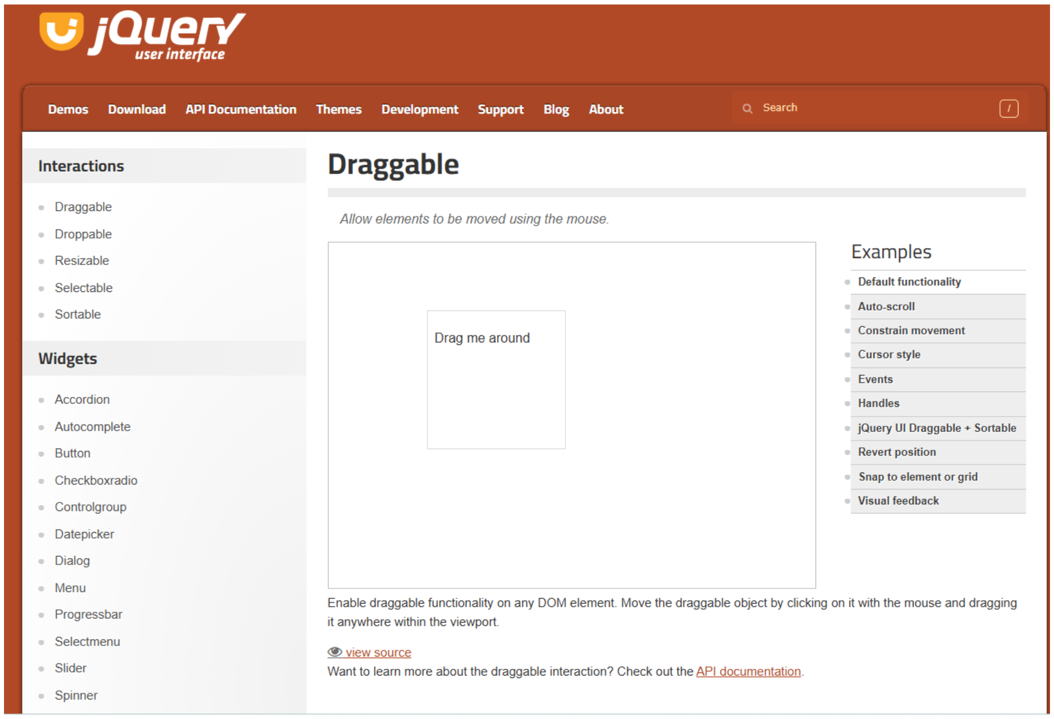
- Children (>):
  $("ul > li > div > img")
  - This selects <img> elements that are direct children of <div>, which are direct children of <li>, which are direct children of <ul>. It does **not** select nested <img> elements that are deeper than one level within <div>.

- Next (+):
  $("img + a")
  - This selects the first <a> element that immediately follows an <img> element. It only targets the next sibling element.
- Siblings (~):
  $("#staff ~ h2")

> This selects all <h2> elements that are siblings of the element with the id="staff". It finds all <h2> siblings at the same level, not just the immediate one.

More:

https://jqueryui.com/



Allow elements to be moved using the mouse.

Enable draggable functionality on any DOM element. Move the draggable object by clicking on it with the mouse and dragging it anywhere within the viewport.

Want to learn more about the draggable interaction? Check out the API documentation.

jQuery vs vanilla JavaScript

Here's a comparison of how to perform the same tasks using both vanilla JavaScript and jQuery.

vanilla JavaScript	jQuery

`// Select all <p> elements`
`var paragraphs = document.querySelectorAll("p");`
`// Change the text content of each <p>`
`paragraphs.forEach(function(p) { p.textContent = "This text has been changed with vanilla JavaScript."; });`
	`// Select all <p> elements and change their text content`
	`$("p").text("This text has been changed with jQuery.");`

`// Select all elements with the class 'box'`
`var boxes = document.querySelectorAll(".box"); // Add the 'highlight' class to each element`
`boxes.forEach(function(box) {`
`box.classList.add("highlight"); });`
	`// Select all elements with the class 'box' and add 'highlight' class`
	`$(".box").addClass("highlight");`

`// Select a button with the ID 'myButton'`
`var button = document.getElementById("myButton");`
`// Attach a click event listener`
`button.addEventListener("click", function() {`
`alert("Button clicked using vanilla JavaScript!"); });`
	`// Select a button with the ID 'myButton' and attach a click event`
	`$("#myButton").click(function() { alert("Button clicked using jQuery!"); });`

`// Select an element with the class 'content'`
` var content = document.querySelector(".content");`
`// Hide the element`
`content.style.display = "none";`
	`// Hide an element with the class 'content'`
	`$(".content").hide();`
	`// Show the element with the class 'content'`
	`$(".content").show();`

`// Show the element`
`content.style.display = "block";`	

Useful links and references:

https://lesscss.org/
https://www.w3schools.com/jquery/jquery_intro.asp
https://www.w3schools.com/jquery/jquery_syntax.asp
https://www.w3schools.com/jsref/obj_window.asp
jQuery website: http://jquery.com
jQueryUI website : http://jqueryui.com
jQuery tutorials:

    http://learn.jquery.com/about-jquery/
    http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery

jQuery UI elements detailed introduction website: http://jqueryui.com/demos/