
# 03 – BOOTSTRAP, LESS DEMO

## Előadás demó leírása a Mobil- és webes szoftverek c. tárgyhoz

Gincsai Gábor

2017.



**Budapesti Műszaki és Gazdaságtudományi Egyetem**
**Villamosmérnöki és Informatikai Kar**
**Automatizálási és Alkalmazott Informatikai Tanszék**

1117 Budapest, Magyar tudósok körútja 2.
Q. ép. B. szárny 207. • www.aut.bme.hu
Telefon: 463-2870 • Fax: 463-2871

# Bevezetés

Ebben a dokumentumban röviden áttekintjük az előadáson bemutatott demókat, hogy az a későbbiekben otthon önállóan is el lehessen végezni.

## Célkitűzés

A demók elvégzésének segítségével gyakorlatiasan meg lehet ismerkedni a LESS készítésének folyamatával, illetve a Bootstrap és Flexbox használatával, ki lehet próbálni az előadáson elhangzottakat a gyakorlatban.

A demók kifejezetten olyan hallgatóknak szólnak, akiknek a LESS, Bootstrap és Flexbox még újdonság, abban tapasztalatuk csekély.

## Miért érdemes megismerkedni Ezekkel a technológiákkal?

A Bootstrap jelenleg az egyik leggyakoribb CSS keretrendszer, amivel reszponzív layoutot igen egyszerűen lehet készíteni, így annak ismerete a webes fejlesztésben nélkülözhetetlen.

A LESS lényege, hogy a CSS-t generáljuk, nem közvetlenül írjuk, és kihasználjuk azt a változók és mixinek által nyújtott lehetőségeket. Egyébként a Bootstrap 3 is letölthető LESS formátumban, amit sokkal kényelmesebben testreszabhatunk mint a CSS-es verzióját.

A flexbox egy modern layout rendszer megvalósításához nélkülözhetetlen CSS megoldás. A böngésző támogatottsága is egyre jobb. Mivel kényelmesebb mint a float alapú layout ezért mindenképpen érdemes az alapjait megismerni. A Bootstrap 4 már flexbox alapú layoutot fog használni.

# 01 – Bootstrap 3 használata

A HTML-es előadás gyakorlatán elkészített oldalra nem készítettünk design-t viszont most bootstrappel gyorsan formázzuk meg.

## CSS linkelése az oldalra

Az oldal head tagjében linekljük be a bootstrap-et az alábbi módon. A másik CSS-re azért van szükség, hogy a bootstrapes CSS osztályokat felül tudjuk definiálni. Ezt a sorrend is nagyon fontos.

```
<link rel="stylesheet" type="text/css" href="styles/libs/bootstrap-3.3.6/bootstrap.css">
<link rel="stylesheet" type="text/css" href="styles/style.css">
```

## Fejléc és oldal szélesség beállítása

A header-t szeretnénk formázni, azonban mivel HTML tagre nem szoktunk közvetlenül CSS szabályt létrehozni, mert túl általánosak a szabályok, adjunk hozzá egy egyedi osztályt `page-header` névvel és állítsuk be, hogy a kép balra floatolódjon, hogy a szöveg mellé kerüljön. Ez bootstrapben a `pull-left` osztállyal tehetjük meg.

Ezen felül azt kell megoldani, hogy a háttér teljes szélességű legyen, de a szöveg ne legyen teljes szélességű. Ezt bootstrapben a `container` osztállyal tehetjük meg.

```
<header class="page-header">
	<div class="container">
		<img src="images/logo-bme-aut.png" alt="BME AUT logo" class="pull-left">
		<h1>Mobil- és webes szoftverek - MyBlog</h1>
		<p>Demo a HTML és CSS alapokhoz</p>
	</div>
</header>
```

Azért nem tudjuk a teljes body-ra tenni a container osztályt, mert akkor a header-ben a háttérszín nem érne ki az oldal széléig.

Azonban miven a bootstrap igen általános CSS kiinduló osztály egyes szabályokat felül kell benne definiálni, ami egy saját CSS osztályban tudunk megtenni, a style.css-ben.

Először állítsuk be, hogy a page-header CSS osztállyal ellátott elemen az alábbi tulajdonságokat

- háttérszín legyen bordó #A4001C a kódja
- A betű színe legyen fehér
- Az paddingot legyenek lent és fent 5px jobb és bal oldalon pedig 20px
- Az alapértelmezett margot állítsuk át 0-ra minden irányban.

```
header.page-header{
	background-color: #A4001C;
	color: white;
	padding: 5px 20px;
	margin: 0;
}
```

Ezen felül a fejlécben lévő képen is állítsunk paddingot, hogy vízszintesen középre kerüljön és legyen 20px távolság a kép és szöveg között.

```
header.page-header img{
	padding-top: 5px;
	padding-right: 20px;
}
```

Ezt követően már csak annyi a dolgunk, hogy a h1 margóját is kissebbre vegyük. Fent legyen 5px lent pedig 0.

```
header.page-header h1{
	margin-top: 5px;
	margin-bottom: 0;
}
```

Ha mindent jól adtunk meg a fejlécnek így kell kinéznie.


## Navigációs linkek

Ahhoz, hogy a navigációs linkeket helyesen beállítsuk és vízszintesen jelenjenek meg egy pár osztályt rá kell aggatni a HTML-re bizonyos osztályokat, és esetleg ezeket kiegészíteni egy-egy saját CSS osztállyal.

A `nav` tagre érdemes rátenni a `navbar` és `navbar-inverse` osztályokat.

Ezen felül, hogy a menü elemek is beljebb kezdődjenek be kell tenni egy `<div>`-et a `nav` tag-be, amire rá kell tenni a `container` osztályt.

Ezen felül már csak az ul-re kell rátenni a `nav` és `navbar-nav` osztályokat.

További infók: http://bootstrapdocs.com/v3.0.3/docs/components/#nav

```
<nav class="navbar navbar-inverse">
	<div class="container">
		<h2 class="nocontent" style="display: none">--- Navigáció ---</h2>
		<ul class="nav navbar-nav">
			<li>
				<a href="index.html">Kezdőlap</a>
			</li>
			<li>
				<a href="aboutus.html">Rólunk</a>
			</li>
			<li>
				<a href="contact.html">Kapcsolat</a>
			</li>
		</ul>
	</div>
</nav>
```

Ezen felül még annyi dolgunk van, hogy a bootstrapben beállított lekerekítét vegyük le, illetve a fekete háttérszín legyen bordó, ehhez az alábbi CSS szabályokat kell felvenni.

```
.navbar{
	border-radius: 0;
}

.navbar-inverse{
	background-color: #A4001C;
}
```

Az elkészült fejléc így fog kinézni:


## LAYOUT KIALAKÍTÁSA

A fő tartalmi rész 2 hasábosan szeretnénk megjeleníteni, amihez a bootstrap grid rendszerét tudjuk használni. Ez alapján az `article` tag legyen 8 oszlop széles, az `aside` pedig 4, így kijön a szükséges 12 oszlop és egymás mellé kerülnek.

Itt is figyeljünk, hogy a tartalom ne töltse ki a teljes oldalszélességet, hanem használjuk itt is a `container` osztályt.

```
<main class="container">
…

<article class="col-lg-8">
</article>
…

<aside class="col-lg-4">
</aside>
```

## ŰRLAP FORMÁZÁSA

Már csak egy lépés maradt hátra, hogy a hírlevél feliratkozásra is tegyük rá a megfelelő CSS osztályokat, hogy az űrlap is szépen nézzen ki.

Részletek: http://bootstrapdocs.com/v3.0.3/docs/css/#forms

Az űrlapon a labelek és inputok egymás mellé kell, hogy kerüljenek. Ehhez a `form`-ra a `form-horizontal` osztályt kell rátenni. Ezt követően az egyes label input párokat egy `form-group` osztállyal ellátott `div`-be kell burkolni.

A labelre meg kell adni, hogy milyen széles legyen, ami jelenleg `col-lg-5` és a `control-label`-t kell rátenni.

Az inputra rá kell tenni a `form-control` CSS osztályt viszont itt is kell szélességet állítani, de azt csak úgy tudjuk megtenni, hogy az input köré teszünk egy div-et amin beállítjuk a `col-lg-7`-et.

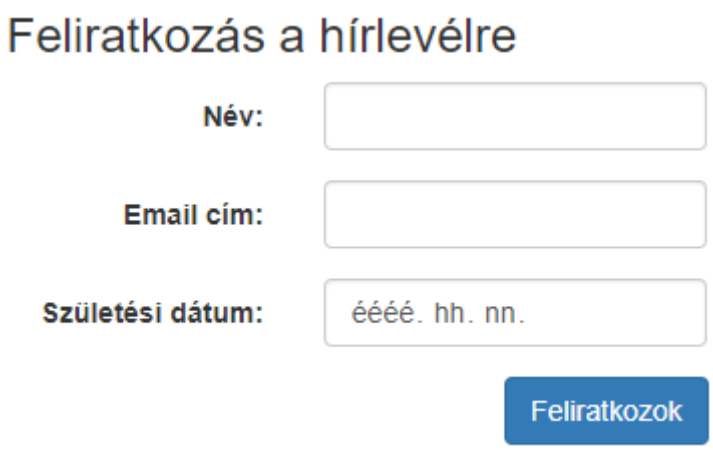
Végül a gombra adjuk meg, hogy gomb kinézete legyen az elsődleges színnel, amit a `btn btn-primary` osztállyal tudjuk megtenni.

```
<form class="form-horizontal">
  <div class="form-group">
    <label for="txtName" class="control-label col-lg-5">Név:</label>
    <div class="col-lg-7">
      <input class="form-control" id="txtName" name="name" type="text" required>
    </div>
  </div>

  <div class="form-group">
    <label for="txtEmail" class="control-label col-lg-5">Email cím:</label>
    <div class="col-lg-7">
      <input class="form-control" id="txtEmail" name="email" type="email" required>
    </div>
  </div>
  <div class="form-group">
    <label for="txtBirthDate" class="control-label col-lg-5">Születési dátum:</label>
    <div class="col-lg-7">
      <input class="form-control" id="txtBirthDate" name="birthDate" type="date">
    </div>
  </div>

  <button class="btn btn-primary pull-right" type="submit">Feliratkozok</button>
</form>
```

Az elkészült formnak így kell kinéznie:



# 02 – LESS A GYAKORLATBAN

Egy pár egyszerű példán kereszül végignézzük, hogy egy-egy LESS kódrészlet milyen CSS-t fog generálni, hogy megértük a pontos működését

Ahhoz, hogy VS Code-ban is használhassul a LESS-t fel kell telepíteni a less node module-t, az alábbi paranccsal a Terminal ablakból (Ctrl + ö )

```
npm i -g less
```

Ezt követően ha fordítani szeretnénk a LESS-t az alábbi paranccsal tehetjük meg, ahol az első paraméter, hogy mit szeretnénk fordítani, a második pedig az output file.

```
lessc styles/demo.less styles/demo.css
```

## A LESS-BEN EGYSZERŰ CSS IS MEGADHATÓ

A LESS fájlokat CSS-re fordítjuk így minden további nélkül megadhatunk sima CSS szabályokat is, hiszen az is értelmezhető lesz. E miatt minden CSS fájl egyben LESS fájlnak is tekinthető, bár ilyenkor nem használjuk ki a LESS nyújtotta előnyöket.

**LESS kód**

```
div {
      padding: 20px;
      font-size: 20px;
}
```

**Generált CSS kód**

```
div {
      padding: 20px;
      font-size: 20px;
}
```

## VÁLTOZÓK

Változók használatával karbantarthatóbbá tehetjük a LESS fájlunkat. Például a színeket felvehetjük egy-egy változóban és a szabályok megadásakor a változó nevére hivatkozunk, így ha megváltozik egy szín, akkor elegendő egy helyen módosítani azt.

**LESS kód**

```
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;

#header {
      color: @light-blue;
}
```

**Generált CSS kód**

```
#header {
	color: #6c94be;
}
```

Mint látható a kimenti CSS fájlban a változó értéke közvetlenül behelyettesítésre kerül, így magát a változót a CSS-ben már nem látjuk.

## Mixinek

Mixinek segítségével több szabály beállítását tudjuk összefogni és arra egy sorral hivatkozni.

A lenti példában azt láthajuk, hogy a .bordered mixin egy alsó és felső bordert állít be, amit később egy sorral tudunk használni.

**LESS kód**

```
.bordered {
	border-top: dotted 1px black;
	border-bottom: solid 2px black;
}

nav a {
	color: #111;
	.bordered;
}
```

**Generált CSS kód**

```
.bordered {
	border-top: dotted 1px black;
	border-bottom: solid 2px black;
}
nav a {
	color: #111;
	border-top: dotted 1px black;
	border-bottom: solid 2px black;
}
```

A generált kódban azt láthatjuk, hogy a mixin is megmaradt mint CSS osztály és ezen felül ahol használtuk oda behelyettesítődött a mixin összes szabálya.

Mixint úgy is létrehozhatunk, hogy ()-et teszünk a végére.

**LESS kód**

```
.yellow-everything(){
	color: orange;
	background-color: yellow;
}

a.current {
	color: red;
	.yellow-everything();
```

```
}
```

**Generált CSS kód**

```
a.current {
	color: red;
	color: orange;
	background-color: yellow;
}
```

Ebben az esetben az történik, hogy maga a mixin mint CSS szabály nem jelenik meg, hanem csak behelyettesítődik oda ahol azt használjuk. Érdemes a sorrendet is megfigyelni, mert jelen esetben a mixinben megadott narancssárga szín fog érvényre jutni, hiszen azt adtuk meg később.

A mixinek készítésénél paramétereket is várhatunk, és azokra default értékeket is megadhatunk.

**LESS kód**

```
.bar (@bg, @color) {
	background: @bg;
	color: @color;
}
.foo (@bg: #15f5f5, @color: #900) {
	.bar (@bg, @color);
}
.unimportant {
	.foo(white, gray);
}

.important {
	.foo() !important;
}
```

**Generált CSS kód**

```
.unimportant {
	background: white;
	color: #900;
}
.important {
	background: #15f5f5 !important;
	color: #900 !important;
}
```

Figyeljük meg a fenti kódban, hogy a .bar mixin két paramétert vár, a @bg-t és a @color-t, amit majd a háttérszínnek és a betű színének állít be. Itt nincs default érték definiálva a paraméterekre.

A .foo mixinben ugyanezt a két paramétert várjuk, azonban adunk meg alapértelmezett színt (fehér és szürke), viszont a kódja csak annyi, hogy meghívjuk a .foo() mixin-t a két paraméterrel. Fontos hogy itt egy mixinből hívunk egy másik mixint.

Használat esetén még az lehet fontos, hogy az !important-ot használhatjuk és az a mixinben lévő minden egyes sorhoz hozzáadódik.

## Beágyazott szabályok

CSS esetén elég macerás leírni azt, ha például a .header osztályra szeretnénk szabályt megadni, majd a .header-ön belüli .logo osztályra hiszen mindet új szabályként kell felvenni és mindig kiírni a selectorba, hogy .header.

LESS esetén ezt a szabályok egymásba ágyazásával meg tudjuk tenni.

**LESS kód**

```
.header {
	.logo {
		width: 300px;
	}
	&.logo {
		width: 300px;
	}
	&-logo {
		width: 300px;
	}
	&&-logo {
		width: 300px;
	}
	>.logo {
		width: 300px;
	}
	&::after, &::before {
		content:'';
	}
}
```

**Generált CSS kód**

```
.header .logo {
	width: 300px;
}
.header.logo {
	width: 300px;
}
.header-logo {
	width: 300px;
}
.header.header-logo {
	width: 300px;
}
.header > .logo {
	width: 300px;
}
.header::after,
.header::before {
	content: '';
}
```

Mint látható szépen kiírtékelésre kerül az egymásba ágyazás és a CSS-ben már beágyazás nélküli szabályok láthatóak. Figyeljük meg a &, && és > használatát is a kódban, hogy mikor mire fordulnak.

Egy direktívába media selectorokat is ágyazhatunk be az alábbi módon:

**LESS kód**

```
.screen-color {
	@media screen {
		font-weight: bold;
		color: green;

		@media( min-width: 768px) {
			color: red;
		}
	}

	@media tv {
		color: black;
	}
}
```

**Generált CSS kód**

```
@media screen {
	.screen-color {
		font-weight: bold;
		color: green;
	}
}
@media screen and (min-width: 768px) {
	.screen-color {
		color: red;
	}
}
@media tv {
	.screen-color {
		color: black;
	}
}
```

Figyeljük meg, hogy ebben az esetben a LESS-ben egy szabályba ágyazzuk be a @media selectort, amit a CSS-ben már fordított sorrendben fogunk látni, hiszen a CSS-ben a @media selectort kell először megadni és ebben hivatkozni az egyes szabályokra. Tény, hogy a LESS-ben ezt egyszerűbben és kézre állóbban tudjuk megadni.

## Műveletek

A LESS-ben egyszerű műveleteket is végezhetünk úgy, hogy a különböző mértékegységeket automatikusan tudja konvertálni.

**LESS kód**

```
// Ugyanabba a mértékegységbe konvertálja.
@conversion-1: 5cm + 10mm; // 6cm
@conversion-2: 2 + 3cm - 5mm; // 4.5cm

// Nem lehet konvertálni
@incompatible-units: 2 + 5px + 1cm; // 4px

.result-class1{
	height: @conversion-1;
	width: @conversion-2;
	height: @incompatible-units;
}
```

**Generált CSS kód**

```
.result-class1 {
	height: 6cm;
	width: 4.5cm;
	height: 44.79527559px;
}
```

## Karakterek escapeelése

A LESS-ben a szabályok nagy része így például a calc is azonnal kiértékelődе és a CSS-be a dinamikus számolás helyett egy konkrét szám kerülne be. Ezt az escapeléssel tudjuk megakadájozni, amihez a ~ karaktert kell használnunk.

**LESS kód**

```
.weird-element {
	height: calc(100% - 10px);
	width: ~"calc(100% - 10px)";
	min-width: calc(~"100% - 10px");
}
```

**Generált CSS kód**

```
.weird-element {
	height: calc(90%);
	width: calc(100% - 10px);
	min-width: calc(100% - 10px);
}
```

## Beépített függvények

A LESS-ben található számos előre beépített függvény, amivel például színeket tudunk manipulálni. Halvínytani, sötétíteni…

**LESS kód**

```
@base: #f04615;
```

```
@width: 0.5;

.class {
	width: percentage(@width); // `50%`
	color: fade(saturate(@base, 15%), 50%);
	background-color: spin(lighten(@base, 30%), 8);
}
```

**Generált CSS kód**

```
.class {
	width: 50%;
	color: rgba(255, 62, 6, 0.5);
	background-color: #f9c3a5;
}
```

## CSS SZABÁLYOK GENERÁLÁSA

LESS-ben lehetőség van rekurzív függvényekkel CSS osztályokat generálni, amivel egyszerűen tudunk például olyan szabályokat létrehozni, ami 25%, 50%, 75%, 100%-es szélességet állít be.

**LESS kód**

```
.generate-columns(@n, @i: 1) when (@i =< @n) {
	.column-@{i}-@{n} {
		width: (@i * 100% / @n);
	}
	.generate-columns(@n, (@i + 1));
}
.generate-columns(4);
```

**Generált CSS kód**

```
.column-1-4 {
	width: 25%;
}
.column-2-4 {
	width: 50%;
}
.column-3-4 {
	width: 75%;
}
.column-4-4 {
	width: 100%;
}
```

Figyelejük meg, hogy a CSS-ben már csak a legenerált osztályok találhatóak.

A LESS kódban a when -el adhatjuk meg a bent maradási feltételt, rekurzívan hívhatjuk a függvényünket, de ne felejtsük el a kódban egyszer meghívni, mert egyébként nem fog lefutni és nem generálja le a szükséges osztályokat.

## EXTEND

A LESS-ben arra is lehetőség van, hogy egy másik szabályban (lenti példában az .inline osztályban) lévő beállításokat a saját selectorunkhoz is hozzáadjuk.

**LESS kód**

```
nav ul {
      &:extend(.inline);
      background: blue;
}
.inline {
      color: red;
}
```

**Generált CSS kód**

```
nav ul {
      background: blue;
}
.inline,
nav ul {
      color: red;
}
```