05 – JQUERY 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 jQuery osztálykönyvár használatával, ki lehet próbálni az előadáson elhangzottakat a gyakorlatban.

A demók kifejezetten olyan hallgatóknak szólnak, akiknek a jQuery még újdonság, abban tapasztalatuk csekély.

Miért érdemes megismerkedni Ezekkel a technológiákkal?

A jQuery az egyik legelterjedtebb JavaScript keretrendszer, amit egyszerűbb weboldalaknál használnak. Akkor érdemes használni, ha nincs összetett kliens oldali keretrendszerünk, mint például az Angular vagy React, mert kliens oldalon csak minimális DOM manipulációt szeretnénk végezni, illetve szinte csak AJAX kéréseket szeretnénk indítani a kliensből.

A jQuery lényege, hogy a DOM manipulációt és a kliensről indított aszinkron kommunikációt a szerverrel egyszerűen pár sorban meg tudjuk valósítani.

01 – jQuery használata

A demo forráskódja a 12. labor jQueryDemo.zip-ben található 01-Demo-jQuery.html-ban érhető el.

Nyissuk meg a jQueryDemo.html fájlt Visual Studio Code-dal.

Az oldal felépítése viszonylag egyszerű. Az oldalon a tanszék munkatársai jelennek meg, ul – li tagekben, egy-egy viszonylag egyszerű formátumban, ahol a kép mellett megjelenik a munkatárs neve, beosztása, telefonszáma és szobájának a száma.

jQuery linkelése az oldalra

Az oldal alján a </body> előtt találhatjuk meg a jQuery linkelését, amit CDN-ről töltünk be, az alábbi kódrészlettel.

```
	<!-- jQuery 3.2.1 belinkelése CDN-ről -->
	<script src="http://code.jquery.com/jquery-3.2.1.js"></script>
</body>
```

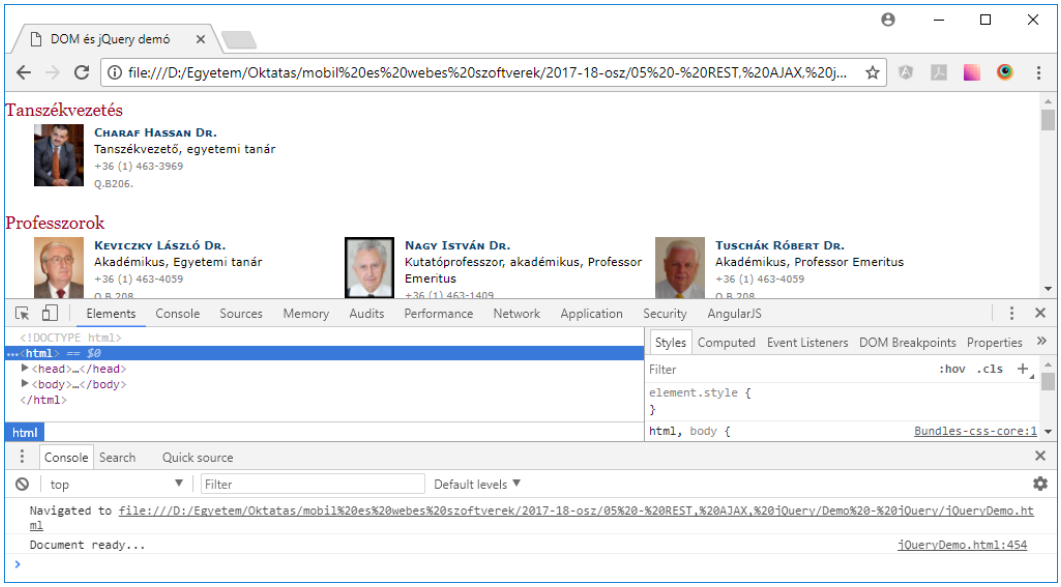
document.ready esemény

Az oldalon akkor tudunk jQuery kódot futtatni, ha már a HTML tagek betöltődtek, azaz ki tudunk választani tageket, amin valamilyen műveletet végre szeretnénk hajtani.

Készítsünk egy olyan kódrészletet, ami a console-ra kiírja ha a dokumentum készen áll. A példa egyszerűségéért a JS kódot most nem külön fájlba készítjük hanem az oldal végén egy script tagben adjuk meg.

```
	<script>
		// Esemény, ami akkor fut le, ha a HTML elemek már létrejöttek.
		// Note: Ide érdemes tenni minden jQuery kódot.
		$(document).ready(function(){
			console.log("Document ready...");
		});
	</script>
</html>
```

Mentés után ha megnyitjuk az oldalunkat és F12-vel megnyitjuk a Developer consolet, akkor az alábbi kell látnunk:



Készítsük el, hogy hogy a *meta* CSS osztállyal rendelkező elemek háttérszíne legyen sárga.

Ehhez a documentum.ready eseménykezelőben kell kiválasztani a .meta osztállyal rendelkező elemeket és annak a css tulajdonságát beállítani.

```
	<script>
		$(document).ready(function(){
			// Háttérszín módosítása
			$(".meta").css("backgroundColor", "yellow");
		});
	</script>
</html>
```

Iteráció a selectált elemeken

Ettől egy kicsivel bonyolultabb az a feladat, hogy minden második ul-elemhez adjunk hozzá egy box CSS osztályt. Ehhez iteráljunk végig az összes UL tagen és egy if-ben döntsük el, hogy rá kell-e tenni a box CSS osztályt vagy sem.

```
// Minden 2. sorhoz css class adása
$("ul").each( function(i){
	if (i % 2 == 1 ) {
		$(this).addClass("box");
	}
});
```

A fenti példában vegyük észre, hogy a $(this)-t használunk. Itt valójában a this egy DOM elem lesz, konkrétan az ul tag és ezt fogjuk a $()-kal jQuery elemmé konvertálni, hogy az addClass függvényt tudjuk haszánlni.

:odd és :even használata

A másik jQuery-hez közelebb álló megoldás kulcsa az :odd vagy :even selector.

```
$("ul").filter(":odd").addClass("box");
```

A fenti példában először kiválasztjuk az összes UL-t, majd tovább szűrjük, hogy csak minden második elemmel szeretnénk valami kezdeni.

```
$("ul:odd").addClass("box");
```

Eseménykezelők

Eseményeket is lehet jQuery-vel készíteni. A következő példában az LI tagek kattintás eseményére tűnjön el az egész LI tag.

```
$("li").click( function( event ) {
	$(this).hide();
});
```

A modernebb megoldása mindennek az alábbi

```
$("li").on( "click", function( event ) {
	console.log("Hide li");
	$(this).hide();
});
```

Események lefutásának sorrendje

Készítsünk egy másik eseménykezelőt is, hogy meg tudjuk nézni az események lefutásának sorrendjét.

Regisztáljunk be az li tagekre még egy eseménykezelőt, ami az li háttérszínét állítja pirosra.

```
// Ha több eseménykezelő van fontos a sorrend.
$("li").on( "click", function( event ) {
	console.log("Background li");
	$(this).css("backgroundColor", "red");
} );
```

Ha futtatjuk a fenti kódot, azt fogjuk látni, hogy az LI elemünk eltűnt, de a konzolon látjuk, hogy mind a két eseménykezelő lefutott a regisztráció sorrendjében.

Eseménykezelő delegálása

Ne iratkozzuk fel egyesével minden LI-re hanem az UL-nél regisztráljuk be az eseménykezelőnket úgy, hogy az csak az LI-kre fusson le.

```
$("ul").on("click", "li", function(event){
	$(this).hide();
});
```

Ezen felül kommentezzük ki a korábbi click eseménykezelő függvényeinket, hogy csak ez fusson. És próbáljuk is ki.

stopPropagation

Ezen felül regisztráljunk be az LI-re is egy eseménykezelőt, ami az LI hátterét módosítja pirosra és letiltja az esemény propagálását!

```
$("li").on( "click", function( event ) {
	$(this).css("backgroundColor", "red");
	event.stopPropagation();
} );
```

preventDefault

Egy eseménykezelőben egyszerűen megtehetjük, hogy egy-egy tagnek megakadályozzuk, hogy az alapértelmezett működése lefusson.

Akadályozzuk meg, hogy a likekre kattintva betöltődjön a hivatkozott oldal.

```
$("a").on("click", function( event ) {
	alert("Ez már nem link!");
	// Megakadályozza az elem default viselkedésének lefutását. Most pont a navigációt.
	event.preventDefault();
});
```

Elemek dinamikus létrehozása

jQuery segítségével is tudunk dinamikusan létrehozni elemeket.

```
// Elemek dinamikus létrehozása
var s = $("<h1>Dinamikusan létrehozott elem.</h1>");
// Hozzáfűzi a #staff Id-jú elemhez az új elemet.
$("#staff").append(s);
// Az újonnan létrehozott elemhez hozzáadja a meta CSS osztályt.
s.addClass( "meta" );
```

Ha az első elemnek szeretnénk az új tag-et beszúrni, akkor az append helyett használjuk a prepend-et.

Eseménykezelő készítése dinamikusan létrehozott elemekhez

Ha már létezik az eleme, tehát hozzáfűztük a DOM-hoz az append vagy prepend függényekkel, akkor minden gond nélkül fel tudunk rá iratkozni.

```
$("#staff h1").click( function() {
	alert("Click");
});
```

Azonban ha a fenti kódot a <h1> létrehozása elé tesszük, akkor nem fog működni. Ilyenkor a delegálás tud segíteni, hogy magát az eseménykezelőt egy már létező elemhez regisztráljuk be pl.: a document elemhez, és az on függvény paramétereként adjuk meg, hogy melyik elemre kell lefutnia.

```
// Ilyenkor a document tárolja, hogy milyen feliratkozás van és a 2. paraméter az,
// hogy valójában mire iratkozunk fel.
$(document).on("click", "#staff h1", function() {
	alert("Click");
});
```

jQuery validate

A demo forráskódja a 12. labor jQueryDemo.zip-ben található 02-Demo-jQuery-validate.html-ban érhető el.

Számos esetben előfordul, hogy a HTML5-ös validáció nem elég számunkra, mert

- lokalizálni, vagy testre szeretnénk szabni a hibaüzeneteket,
- egyedi validációs szabályt szeretnénk megadni
- a hibaüzenetek megjelenítését szeretnénk befolyásolni.

Ilyen esetben a jQuery validate plugin tud segíteni, aminek a használatát az alábbi példán keresztül nézzük meg.

Nyissuk meg a validacio.html-t.

Az oldalon található egy regisztrációs űrlap. Ami lényeges benne, hogy az email címre meg van adva a required HTML attibútum és minden inputhoz meg van adva a name attibútum is.

Mivel még csak HTML5-ös validáció van megadva, ezért ha a Regisztrálok gombra kattintunk, akkor az alábbi hibaüzenetet kapjuk.

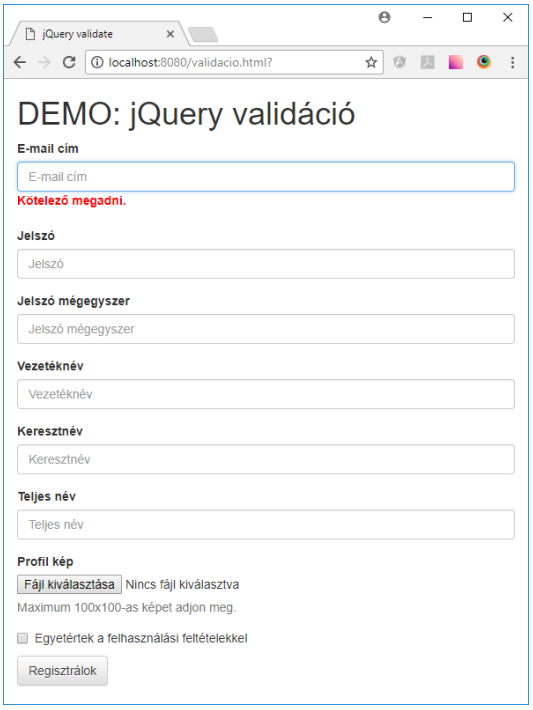
HTML5-ös validációk megjelenítése jQuery validate-tel

Ahhoz, hogy a hibaüzenetek ne a fenti formában jelenjenek meg, csak annyi dolgunk van, hogy a document.ready eseménykezelőben megadjuk, hogy az űrlaponkat jQuery validate-tel szeretnénk validálni. Ehhez a jQuery-t és a jQuery validate-et kell linkelni.

```
<!-- jQuery 3.2.1 belinkelése CDN-ről -->
<script src="https://code.jquery.com/jquery-3.2.1.js"></script>
<!-- jQuery validate belinkelése -->
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/jquery.validate.js"></script>
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/additional-methods.js"></script>
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/localization/messages_hu.js"></script>
<script>
	$().ready(function() {
		$("#registrationForm").validate();
	});
</script>
```

Ahogy a fenti kódrészleten is látható csak annyi dolgunk van a szükséges JS fájlok megfelelő sorrendben történő linkelése után, hogy meghívjuk az űrlapon a **validate()** függvényt.

Ha így futtatjuk a validációt, akkor már a jQuery validate plugin fogja a hibákat megjeleníteni az alábbi módon



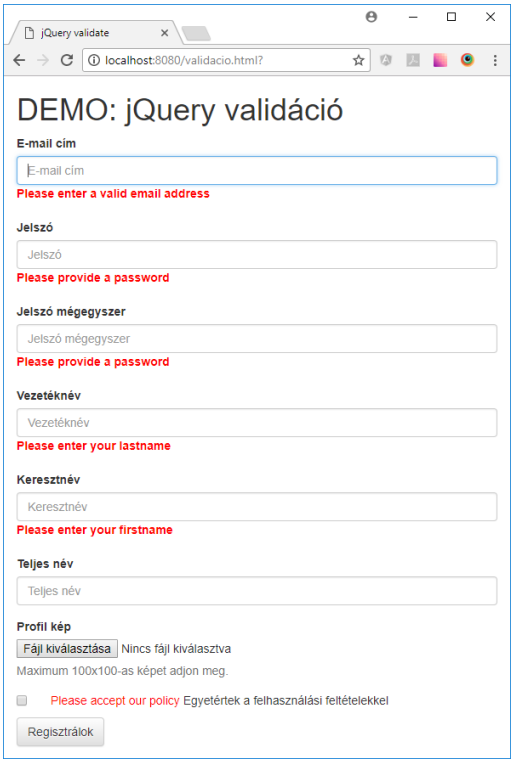
VALIDÁCIÓS SZABÁLYOK ÉS HIBAÜZENETEK MEGADÁSA A JQUERY VALIDATE-NEK.

Arra is van lehetőség, hogy a validációs szabályokat ne HTML-ben attibútumokkal adjuk meg, hanem a validate függvény paramétereként az alábbi módon.

```
<script>
$().ready(function() {
	// jQuery validáció regisztrálása a formhoz.
	$("#registrationForm").validate({
		// Szabályok megadása az egyes mezőkre (name alapján)
		rules: {
			firstName: "required",
			lastName: "required",
			password: { required: true, minlength: 5 },
			confirmPassword: { required: true, minlength: 5, equalTo: "#password" },
			email: { required: true, email: true },
			agree: "required"
		},
		// Hibaüzenetek megadása az egyes mezőkre (name alapján)
		messages: {
			firstName: "Please enter your firstname",
			lastName: "Please enter your lastname",
			password: {
				required: "Please provide a password",
				minlength: "Your password must be at least 5 characters long"
			},
			confirmPassword: {
```

```
                        required: "Please provide a password",
                        minlength: "Your password must be at least 5 characters long",
                        equalTo: "Please enter the same password as above"
                    },
                    email: "Please enter a valid email address",
                    agree: "Please accept our policy"
                }
        });
</script>
```

Ezt követően már látható, hogy minden mezőhöz szépen megjelenik az egyedi hibaüzenet.



Egyedi sumbit kezelő függvény megadása

Arra is lehetőségünk van, hogy a sikeres validáció után, egyedileg adjuk meg, hogy hogyan szeretnénk az űrlapot továbbítani a szerver felé, akár AJAX-al is. Ehhez az alábbi kódot kell elkészíteni, amiben a validátor alapértelmezett beállításai között megadhajuk a saját `submitHandler` függvényünket.

```
$.validator.setDefaults({
        submitHandler: function() {
                alert("submitted!");
        }
});
```

AJAX KÉRÉS JSONP-VEL

A demo forráskódja a 12. labor jQueryDemo.zip-ben található 03-Demo-JSONP.html-ben érhető el.

jQuery segítségével egyszerűen tudunk AJAX kéréseket is indítani a $.ajax segítségével.

Az alábbi példa a diplomaterv portálról kérdezi le egy adott konzulensek a témáit. Ehhez szerver oldalon egy OData protokoll által kiajánlott végpont van, ahol az URL-ben lehet keresési feltételeket megadni és JSON választ fog visszaadni.

A HTML oldalon egy egyszerű űrlap található, ahol meg lehet adni a konzulens nevét.

A HTML fájl az alábbiak szerint alakul

```
<!DOCTYPE html>

<html lang="en">
<head>
	<meta charset="utf-8" />
	<title>Demo: JSONP kérés jQueryvel.</title>
	<link rel="stylesheet" href="app.css" type="text/css" />
</head>
<body>
	<h1>Diplomaterv kereső</h1>

	<div id="content">
		<label for="supervisor">Konzulens neve:</label>
		<input id="supervisor" type="text" />
		<input id="btnSearch" type="button" value="Keresés" />

		<h2>Találatok</h2>
		<ul id="result"></ul>
	</div>

	<script src="http://code.jquery.com/jquery-3.2.1.js"></script>
	<script src="app.js"></script>

	<script type="text/javascript">
		$(function() {
			// Feliratkozás a gomb kattintás eseményre.
			$("#btnSearch").on("click", search);
		});
	</script>
</body>
</html>
```

Az oldal alján hivatkozzuk be a jquery-t a CDN-ről és egy saját JS fájlt app.js névvel.

Ezt követően egy külön script tagben iratkozunk fel a gomb kattintás eseményére, amit a search függvény fog kezelni. A keresés gombra kattintva kell letölteni egy speciálisan összeállított URL-ről a választ, aminek a kódja a search függvénybe kerül az app.js fájlban.

Az ajax kérést a http://diplomaterv.vik.bme.hu/OData/V1/Theses oldalra kell irányítani. Ezt követően a query stringben kell megadni, hogy pontosan mire szeretnénk keresni. Jelen esetben a konzulens nevére, amit a

`Supervisor/DisplayName eq 'Gincsai Gábor'` pararméter ad meg. Ezen felül az `&$expand=Supervisor, Student` paraméterrel pedig azt tudjuk megmondani, hogy a konzulens és hallgató adatait is le szeretnénk kérdezni.

Ezt követően az AJAX kérés then függvényében tudjuk a választ feldolgozni, hiszen a $.ajax egy promisse-al fog visszatérni. A then()-nek az alábbi 3 függvény paramétere lehet

- success handler: Sikeres kérés esetén fut le, itt tudjuk az eredményt a HTML oldalra kitenni.
- error handler: Hiba esetén ez fut le.
- always handler: Minden esetben lefut, függetlenül attól, hogy sikeres vagy sikertelen volt a kérés.

A keresés megvalósító függvényünk az alábbiak szerint fog alakulni:

```
function search() {
	console.log("Keresés a  diplomaterv portálon");
	$.ajax({
		dataType: "jsonp",
		url: "http://diplomaterv.vik.bme.hu/OData/V1/Theses
			?$filter=Supervisor/DisplayName eq '" + $("#supervisor").val()
			+ "'&$expand=Supervisor, Student",
		jsonp: '$callback',
	}).then(
		// success handler.
		function (response, status, xhr) {
			var list = response.d.results;

			// A megkapott listát írjuk ki a konzolra (debughoz jó lesz).
			console.log(list);

			// Hallgató és cím kiírása.
			for (var i = 0; i < list.length; i++) {
				// <li> dinamikus létrehozása.
				var item = $("<li/>");
				// <a> dinamikus létrehozása, href beállítása.
				var link = $("<a/>", { href: list[i].PortalUrl });
				link.text(list[i].Student.DisplayName + ": " + list[i].TitleHu);
				// <li>-hez <a> adása és az egész hozzáfűzése a #result-hoz.
				item.append(link).appendTo("#result");
			}
		}
		/*, Ide jöhet az error handler */
		/*, Ide jöhet az always ami siker és sikertelenség esetén is lefut. */
	);
}
```

Érdemes megfigyelni, hogy a success handlerben a válasz alapján dinamikusan generáljuk az <li> tagekbe a linkeket, amiben megjelenik a hallgató neve és a téma címe.

Ha kipróbáljuk a kódot és rákeresünk Gincsai Gábor -ra, akkor az alábbi eredményt kell látnunk: