Tartalom

Mire is jó?

A jQuery egy igen népszerű JavaScript library, mely a HTML kód, és a kliensoldali JavaScript közötti kapcsolatot hangsúlyozza. 2006-ban jelentette meg a Mozilla. Lényege, hogy amennyire csak lehetséges leválassza a JavaScript kódot a HTML-ről, és különböző eseményvezérlőkön, és azonosítókon keresztül kommunikáljon a weblap HTML elemeivel. A jQuery könnyebbé teszi a JavaScript használatát, a függvényei segítségével:

- Kezelhetünk eseményeket, pl. amikor a felhasználó egy gombra kattint, jelenjen meg egy üzenet.
- Létre hozhatunk áttűnéseket, mozgásokat és egyéb látványos effekteket
- Hozzá adhatunk elemeket css osztályokhoz, vagy eltávolíthatjuk őket onnan
- Ajax támogatása
- Vagy esetleg meghívhatunk php kódon keresztül SQL lekérdezéseket
- ... és még sok minden más.

Mondhatjuk úgy is, a JQuery a JavaScriptnek olyan, mint a Bootstrap a CSS-nek…

Amit meg lehet valósítani JQuery-ben, azt meg lehet csinálni JavaScriptben, fordítva azért nem mindenre igaz 😊

Előzetese ismeretek: A JQuery használatához, nem árt tisztában lenni a következő nyelvekkel.

- HTML
- CSS
- JavaScript

Alkalmazása

1) Elsőként le kell tölteni magát a JavaScript könyvtárat, az alábbi oldalról.

   https://jquery.com/download/

2) Miután ez megtörtén, másoljuk a mappába, ahova szeretnénk
3) Ezután a fájlon beül hivatkozni kell rá, a weboldalunkon, itt érdemes, a body rész végén hivatkozni rá, mivel a JQuery segítségével általában a weboldalt elemeit szoktuk manipulálni, ehhez pedig, érdemes, ha előtte betölt az összes weboldal elem, ugye…
   vagy közvetlenről hivatkozhatunk az interneten lévő forrásfájlra is! Szintén a body végében.

   ```
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
   ```

Szelektorok (Elemek kiválasztása)

$("p")	Az összes bekezdést kijelöli
$("#test")	test azonosítójú elemeket jelöli ki
$(".test")	test osztályú elemeket jelöli ki
$("*")	Minden elemet kijelöl
$(this)	Az aktuális elemre vonatkozik a tulajdonság
$("p.intro")	Az összes bekezdés elemet kiválasztja az intro osztályból
$("p:first")	Az első p elemet választja ki
$("ul li:first")	A első rendezetlen lista első elemét választja ki
$("ul li:first-child")	Minden rendezetlen lista, első elemét választja ki
$("[href]")	Az összes elemet kiválasztja aminek van href tulajdonsága
$("a[target='_blank']")	Az összes olyan elemet kiválasztja, aminek a target tulajdonsága „_blank”
$("a[target!='_blank']")	Az összes olyan elemet kiválasztja, aminek a target tulajdonsága NEM „_blank”
$(":button")	Kiválasztja az összes input és button elemet, aminek a típusa button
$("tr:even")	Kiválaszt minden páros sort a táblázatban (0-tól indexel)
$("tr:odd")	Kiválaszt minden páratlan sort a táblázatban (0-tól indexel)

További JQuery selectorok(kijelölők) listája

Események, Interakciók

A Jquery-ben ahogy a JavaScriptben is, különböző eseményekhez rendelhetjük a módosítások végrehajtását. Ezek közül a leggyakoribbak a következők.

Egér események

- click – klikkelés egérrel
- double click – dupla kikkelés egérrel
- mouseenter – egérrel az adott elem felé kerülünk
- mouseleave – egérrel elmegyünk az adott elemről

Billentyű események

- keypress – lenyomok egy gombot a billentyűzeten
- keydown – egy gomb lenyomásának folyamata
- keyup – egy gomb felengedése folyamata

Form(Űrlap események)

- submit – űrlap elküldése
- change – megváltoztatjuk egy űrlapelem értékét
- focus – Az elem megkapja a fókuszt (pl.: belekattintok)
- blur – Az adott elemből „kikattintok”.

Document/Window események

- load – betöltődik egy oldal
- resize – átméreteződik egy oldal
- scroll – amikor görgetünk az oldalon
- unload – amikor „bezárunk egy ablakot”

További JQuery események (eventek) listája

Alap JQuery eventek

Nagyon fontos eventeket azután szoktunk csak végrehajtani, miután az egész oldal betöltődött, ehhez vagy a dokumentum végére kell meghívni a script részben a kódunkat, vagy pedig a futtatandó kódot a:

```
$(document).ready(function () {/*ide jön a kód*/});
```

alapvető eventtel kell használunk, ami akkor engedi lefuttatni a kódokat, ha már az oldal betöltődött.

Hide/Show (Megjelenítés/Elrejtés)

```
 //hide függvény, elrejti az elemet.
            $("#hideButton").click(function () {
                $("#elrejtendoElem").hide();
            });

//show függvény megjeleníti az elemet.
            $("#showButton").click(function () {
                $("#elrejtendoElem").show();
            });

//toggle button, a show és a hide parancs váltogatását végzi, állapottól
függően
            $("#toggleButton").click(function () {
                $("#elrejtendoElem").toggle();
            });
```

Slide (Beúszás)

```
//SLIDEDOWN/SLIDEUP/SLIDETOGGLE - Bezáródó lenyíló elem
            $("#slideUpButton").click(function () {
                $("#slidePanel").slideUp("slow");
            });

            $("#slideDownButton").click(function () {
                $("#slidePanel").slideDown("fast");
            });

            $("#slideToggleButton").click(function () {
                $("#slidePanel").slideToggle(3000);
            });
```

Egyszerű animáció

```
//ANIMATE - Animáció készítése
            $("#alapAnimacioGomb").click(function () {
                $("#animacioPanel").animate({
                    left:'200px'
                });
            });
            $("#egyszeruAnimacioGomb").click(function () {
                $("#animacioPanel").animate({
                    left:'200px',
                    color:'#ff0000',
                    fontSize: '+=2'
                });
            });
```

Callback függvény

```
//CALLBACK függvények
//A callback függvények végrehajtása azután a függvény után   történt ami után
meghívtuk paraméterként, érdemes a "aszinkron műveletek esetén használni"
            $("#hideCallbackkelGomb").click(function () {
                $("#elrejtendoElem").hide("slow",
                    function () {
                        alert("Az elemünk elrejtésre került!")
                    });
            });
//Nem muszáj megírjunk egyből a függvényt azt külön is meghívhatjuk
paraméterként lásd a következő példában!
            $("#showCallbackkelGomb").click(function () {
                $("#elrejtendoElem").show("slow", CallBackAlert);
            });

            function CallBackAlert() {
                alert("Callback üzenet külső függvényből.")
            }
```

JQuery Chain függvényhívás

```
//CHAIN - A függvényeket láncban is meghívhatjuk, JQuery esetében is, párszor
már láttunk JavaScriptban is rá példát, pl szövegfüggvény kezelésénél...
            $("#chainFadeGomb").click(function () {
                $("#slidePanel").fadeIn(3000).fadeOut(3000);
            });

            $("#chainSlideGomb").click(function () {
                $("#slidePanel").slideDown(3000).slideUp(3000, ChainUzenet);
            });
            function ChainUzenet() {
                alert("A chain animáció véget ért!.")
            }
```

Értékek kiolvasása HTML elemekből

```
//ÉRTÉKEK KIOLVASÁSA HTML KÓDBÓL
            //text(): Az adott elem szöveges tartalmával tér vissza
            $("#textErtekGomb").click(function () {
                alert("Szöveges tartalom: [" + $("#leiras").text() + "]");
            });

//html(): Az adott elem html kódjával tér vissza
            $("#htmlErtekGomb").click(function () {
                alert("HTML tartalom: [" + $("#leiras").html() + "]");
            });

//val(): Az adott elem (ebben az esetben input mező) értékével tér vissza
            $("#valErtekGomb").click(function () {
                alert("Input mező értéke: [" + $("#szamBeviteliMezo").val() +
"]");
            });

//attr("type"): AZ adott elem (ebben az esetben type) attibútum típusával tér
vissza
            $("#attrErtekGomb").click(function () {
                alert("Input mező típusa: [" +
$("#szamBeviteliMezo").attr("type") + "]");
            });
```

Értékek írása webes elemekbe

```
//text(): Az adott elem szöveges tartalmát módosíthatjuk vele
            $("#textErtekModositoGomb").click(function () {
                $("#leiras").text("<b>Módosított</b> szöveg");
            });

//html(): Az adott elem html kódját módosíthatjuk vele
            $("#htmlErtekModositoGomb").click(function () {
                $("#leiras").html("<b>Módosított</b> szöveg");
            });

//val(): Az adott elem (ebben az esetben input mező) értékével tér vissza
            $("#valErtekModositoGomb").click(function () {
                $("#szamBeviteliMezo").val(42);
            });

//attr("type"): AZ adott elem (ebben az esetben type) attibútum értékét
módosítjuk
            $("#attrErtekModositoGomb").click(function () {
                $("#szamBeviteliMezo").attr("type", "password");
            });
```

Új webes elemek létrehozása

```
//prepend(): A kijelölt elemek elejére illeszt be tartalmat
            $("#listaElejetBovitoGomb").click(function () {
                $("ul").prepend("<li>Lista elejére beillesztett elem</li>");
            });
//append(): A kijelölt elemek végére illeszt be tartalmat
            $("#listaVegetBovitoGomb").click(function () {
                $("ul").append("<li>Lista végére beillesztett elem</li>");
            });

//before(): A kijelölt elem elé illeszt be tartalmat
            $("#listaEleElemBeszuroGomb").click(function () {
                $("ul").before("<br>Ez egy szöveg a lista előtt
            });

//after(): A kijelölt elem mögé illeszt be tartalmat
            $("#listaMogeElemBeszuroGomb").click(function () {
                $("ul").after("Ez egy szöveg a lista után...<br>");
            });
```

Elemek törlése

```
//empty(): Eltávolítja a gyermek elemeket a kijelölt elemből
            $("#listaKiuritese").click(function () {
                $("ul").empty();
            });

//remove(): Eltávolítja a kijelölt elemet és annak teljes tartalmát!
(VESZÉLYES!)
            $("#listaTorlese").click(function () {
                $("ul").remove();
            });
```

Osztályok értékeinek módosítása

```
//addClass(): Egy vagy több osztályt adhatunk a választott elemhet
            $("#nagybetusSzovegGomb").click(function () {
                $("#leiras").addClass("nagyBetu");
            });

//removeClass(): Egy vagy több osztályt távolíthatunk el a választott elemről
            $("#simaSzovegGomb").click(function () {
                $("#leiras").removeClass("nagyBetu");
            });
```

```
//toggleClass(): Vált az osztályok felvétele vagy eltávolítása között a
kijelölt elemen
            $("#valtozoSzovegGomb").click(function () {
                $("#leiras").toggleClass("nagyBetu");
            });
```

CSS tulajdonságok módosítása:

```
//css(): CSS tulajdonság kiolvasása
            $("#cssKiolvasasaGomb").click(function () {
                alert("A aktuális színe:" + $("#leiras").css("color"));
            });

//css(): CSS tulajdonság módosítása
            $("#cssModositasaGomb").click(function () {
                $("#leiras").css("color", "red");
            });
```

Ajax

Mire használjuk?

Az **Ajax** (Asynchronous JavaScript and XML) interaktív webalkalmazások létrehozására szolgáló webfejlesztési technika. A weblap kis mennyiségű adatot cserél a szerverrel a háttérben, így a lapot nem kell újratölteni minden egyes alkalommal, amikor a felhasználó módosít valamit. Ez növeli a honlap interaktivitását, sebességét és használhatóságát.

Előnyei

Felhasználói felület

Az Ajax használat közben az oldal képes arra, hogy az oldal csak egy adott részét frissítse be, így az interaktivitás sebessége megnő, szinte már olyan mintha asztali alkalmazást használnánk.

Letöltés sebesség szerver tehermentesítése

Mivel kliensoldali nyelvről beszélünk, ugye ami a gépünkön fut, erről már volt korábban volt szó, így növelhető az oldal letöltésének sebessége, valamint a műveletek esetén a szerver tehermentesítése, hiszen a legtöbb folyamat a kliens gépen fut majd le. Tehát az oldal használati sebessége, csupán a felhasználó eszközének sebességétől függ majd.

Tartalom funkció és forma

Az Ajax segítségével az oldal tartalmát, annak funkcióit, és a megjelenésének formáját teljes mértékben el tudjuk különíteni a kódunkban. Az adatok tárolására általában az XML vagy hasonló struktúrákat használjuk

FETCH API

A Fetch API az XMLHttpRequest egyszerűbb, könnyen használható verziója az erőforrások aszinkron felhasználására. A Fetch lehetővé teszi a REST API-k használatát további lehetőségekkel, például az adatok gyorsítótárazásával, a streaming válaszok olvasásával és egyebekkel.

A fő különbség az, hogy a Fetch ígéretekkel dolgozik, nem pedig visszahívásokkal. A JavaScript fejlesztői az ígéretek bevezetése után eltávolodtak a visszahívásoktól.

Bonyolult alkalmazások esetén könnyen megszokhatja a visszahívások írását, amelyek visszahívási pokolhoz vezetnek.

Felhasznált források:

JQuery letöltő honlap

JQuery Wikipédia(magyar)

JQuery Wikipédia(angol)

W3Schools JQuery

JQuery Script(kódgyűjtemény)

Ajax Wikipédia

animate.css letöltő oldal

hover.css tutorial oldal

hover.css letöltő oldal