**JQuery alapok:**A DOM betöltődése és az oldal teljes betöltődése nem feltétlenül egy időpontra esik,hisz az oldal teljes betöltődéséhez minden képnek és médiának be kell töltődnie.A DOM hamarabb betöltődik mint az egész oldal ,és már ekkor elkezdhetjük kijelölni az elemeket módosításra.

pl.

$(document).ready(function(){

alert("Kész a dom");

});

Ez a példaprogramrész a dom betöltődését egy alert megjelenő füllel jelzi.

A ready függvénynek kell értékül adni azt(zokat) a functionokat ,amelyeknek a DOM betöltődésekor le kell futniuk!

Érdemes a stíluslapok és egyéb dolgok beemelése után lefuttatni,hisz akkor bizonyosan minden elemtulajdonság meghatározott lesz ,mielőtt lefutna a javascript kód.

**A JS kódok miért a </body> záró tag előtt legyenek közvetlenül?:**Azért ,mert gyorsabb betöltődést biztosít,ha a böngésző a js kódot a végén tölti be,tehát minden más dolgot a js előtt tölt be.Ez azért hasznos,mert a legtöbb böngésző általában felfüggeszti a betöltést míg a js értelmező-motor le nem fordítja az oldalon található js kódokat.

A következő példában a p tagek számát írjuk ki alertbe.

$(document).ready(function(){

alert($("p").length).text();

});

A következőkben kiválasztási kritériumok megadását taglaljuk:

-Itt megadjuk,hogy azon p tagek számát adja vissza,amelyek div-ben vannak.

alert($("p","div").length).text();

egy ezzel ekvivalens megoldás:

alert($("p",$("div")).length).text();

További JQuery selectors:

$("\*") Selects all elements

$(this) Selects the current HTML element

$("p.intro") Selects all <p> elements with class="intro"

$("p:first") Selects the first <p> element

$("ul li:first") Selects the first <li> element of the first <ul>

$("ul li:first-child") Selects the first <li> element of every <ul>

$("[href]") Selects all elements with an href attribute

$("a[target='\_blank']") Selects all <a> elements with a target attribute value equal to "\_blank"

$("a[target!='\_blank']") Selects all <a> elements with a target attribute value NOT equal to "\_blank"

$(":button") Selects all <button> elements and <input> elements of type="button"

$("tr:even") Selects all even <tr> elements

$("tr:odd") Selects all odd <tr> elements

**Halmaz szűkítés:**Amikor kijelölünk a DOM egy halmazát,azt menetközben szűkíthetjük.

Egy új feltételt adhatunk,és aszerint szűkülhet a halmaz ,amikre módosítunk.

Az alábbi példában használjuk a filter metódust,ami azt csinálja,hogy szűri az aktuális elemhalmazt.

Itt a p tageket úgy szűri,hogy csak azok maradhatnak a halmazban,melyek a cl classban vannak.

alert($("p").filter(".cl").length).text();

Kijelölésre használható a css kijelölés is:

A div p gyerekeit jelöli ki.

alert($("div p").length).text();

**find() és filter() tagfüggvények közti különbség:**A find() –val új elemhalmazt hozhatunk létre az aktuális dom elemeiből vagy annak gyerekeiből.

Az aktuális halmaz gyerekeit jelöli ki és végez rajtuk műveleteket . A filter() csak az aktuális elemhalmazon végez műveletet.

Ha a kijelölt elemek gyerekei közül szeretnénk választani elemeket akkor find, ha az aktuális halmazt szeretnénk szűrni akkor filter.

**Visszatérés az előző kijelöléshez:**Ha elnyomunk findokat és filtereket,de vissza szeretnénk térni az alaphalmazhoz akkor használhatjuk az end() tagfüggvényt.

Itt a diven belüli p-ket kijelöljük,de elnyomunk egy end()-et amivel 1-et visszaugrunk tehát pontosan a dives halmazra.

alert($("div").filter("p").end().length).text();

Természetesen nem csak egyet lehet visszaugrani,hisz nem csak egyszer használhatjuk a findet és a filtert:

alert($("div").filter("p").end().end().length).text();

**Az előző kijelölés befoglalása az aktuálisba:**Itt legyenek divek és azon belül p tagek.Ha a $(”div”).css(formázás) ,akkor a divet formázzuk és a benne lévő p tageket nem.Ha azt szeretnénk,hogy a diveket és a benne lévő p tageket is formázzuk,akkor az andSelf()tagfüggvényt kell használni.

A következő példában,ha csak sima $(”div”).find(”p”).css(formázás); használnánk ,akkor a p tagben definiált italic tagra nem módisítunk.Ha $(”div”).find(”p”).find(”i”)css(formázás); ez pedig csak az italicra vonatkozik.

<div>

<p>DOM is <i>ready</i>.</p>

</div>

Ha azt szeretném,hogy az egész dom is ready mondatot nagyra vegyük, akkor:

$("div").find("p").find("i").andSelf().css("font-size","60px");

Fontos:Az andSelf az aktuális halmazba csak az előző elemhalmazt foglalja bele.

**DOM fa bejárása:**Ugrálni akarunk,hogy a p tagek közül legyen a második stb…

Az első p tag formázása eq() az nullától van indexelve:

$("p:eq(0)").css("font-size","60px");

A kövi elemet választja:

$("p:eq(1)").next().css("font-size","60px");

.prev() az előzőt

.prevAll():Az összes megelőzőt kijelölni.

.prevUntil():

.parent() a szülőt

.parents():az összes szülőt kijelöli.

.nextAll() minden elemet kiválaszt a jelenleg kijelölt után.

.next():Ezzel a következőre hivatkozunk.

.nextUntil(”p”); Jelöld ki a következőket egészen az első előforduló p tageig.

$("p:eq(1)").nextAll().css("font-size","60px"); Tehát itt a második után mindet kijelöljük.

.prevAll() Minden előtte lévő azonos tagű elemet kijelöl:

A következő példa nagyon érdekes,hisz kijelöljük a második p-t,majd annak a szülöjét,és annak a gyerekeit,tehát az összes p testvért kijelöljük:

$("p:eq(1)").parent().children().css("font-size","60px");

A burkolóhalmazt end() tagfüggvénnyel pörgethetjük vissza.

**DOM elemek létrehozása,módosítása és beszúrása:**Új DOM elemet akarunk kiválasztani és beszúrni a DOM-ba.

Az appendTo metódussal rakhatjuk hozzá.Az end azért kell,hogy visszaérjünk az eredeti halmazhoz és azt adjuk hozzá a DOM hoz.A find(’a’) azért kell,hogy az a tagre ugorva beállítsunk neki href attribútumot,majd visszaugrunk,és az egész p taget rakjuk bele a body-ba.

$("<p><a>korte</a></p>").find("a").attr("href","http://www.google.hu").end().appendTo("body");

A következő megoldás az összes p elemhez hozzáad egy p taget ami fa stringet tartalmaz.

$("p").append("<p>fa</p>");

A következő a p tagek elé raknak!Az append az a kijelölt elemek mögé rak.

$("p").prepend("<b>fa</b>");

A következő példa arra,hogy az img elé és mögé rakhatunk elemet.

$(document).ready(function(){

$("#btn1").click(function(){

$("img").before("<b>Before</b>");

});

$("#btn2").click(function(){

$("img").after("<i>After</i>");

});

});

**DOM elemek eltávolítása:**Elemeket szeretnénk eltávolítani a DOM-ból.

remove():

Ezzel a megoldással az összes p taget töröljük:

$("p").remove();

Lehet szürni is,hogy ne mindent törötljön:Ekkor csak azokat a p tageket törli,melyek a cl classhez tartoznak.

$("p").remove(".cl");

Fontos:A remove() tagfüggvény a kijelölt elemeket csak a DOM ból törli,a JQuery burkolóhalmazából nem.Azaz elméletben továbbra is végezhetünk rajtuk műveletet ,és akár vissza is rakhatjuk a DOM-ba.A remove() nem csak az elemeket távolítja el a DOMból hanem, minden belső tartalmat amit azok tartalmaznak.

A következő tagfüggvény kiűríti az adott div tartalmát:A div megmarad a beállításaival együtt,csak a leszármazottjainak lőttek.

$("div").empty();

**DOM csomópont cseréje:**

A következő példában az összes p taget lecseréljük:(Még annak gyermekeit is):

$("p").replaceWith("<p>szoveg</p>");

**Klónozás:**Másolat készítés egy DOM halmazról:

Az alábbi feladatban az „a” idjű lista elemeit átmásoljuk a „b” idjű üres listába:

<ul id="a">

<li>list</li>

<li>list</li>

<li>list</li>

<li>list</li>

<li>list</li>

</ul>

<ul id="b">

</ul>

<script>

$("#a li").click(function(){

alert("Rákattintottál!");

}).parent().clone(true).find("li").appendTo("#b").end().end().end().remove();

$("#b").css("background-color","red");

</script>

A fenti megoldás magyarázása:

-$("#a li") kijelöli azt az <ul> elemet,amelynek az idje „a”,majd annak az összes <li> elemét jelöli ki.

- click(function(){

alert("Rákattintottál!");

}) ez a kattintásra állít eseményt a kijelölt halmazon(<li>).

- parent() Bejárja a DOMot és kijelöli az aktuális halmaz szülőjét,ami az <ul> aminek az idje „a”.

- clone(true) Ez lemásolja az aktuálisan kijelölt halmazt a gyerekeivel együtt és minden eseményt,ami azokhoz tartozik.Ezt úgy érjük el hogy a clone() függvényt true ra állítjuk.

- find("li") Itt kijelöli a klónozott <ul> -nek az <li > gyerekeit(akik szintén klónok).

- appendTo("#b") Megfogja a kijelölt <li> klónokat és az üres „b” idjű <ul>be rakja őket.

Majd az end() metódusokkal annyit ugrunk vissza,hogy az eredeti <ul> legyen kijelölve(aminek az idje az „a”) és azt töröljük a remove() metódussal.

**DOM elemjellemző kiolvasása ,módosítása,törlése:**

Itt megadjuk az a tag href attribútumának az értékét.Ha több a tag is van,akkor az elsőt jelöli ki.

$("a").attr("href","http://www.google.hu");

Itt kiirjuk az a tag href attributumának az értékét.

alert($("a").attr("href"));

Több attributúm módosítása:

$("a").attr({"href":"http://www.google.hu","title":"alma"});

Attribútum törlése:

$("a").removeAttr("title");

Mivel a class attribútum több értéket is tartalmazhat,ezért azt különlegesen is kell kezelni:

(class=”class1 class2 class3”)

-addClass(): $("a").addClass("title"); Ez úgy ad hozzá attribútum értéket,hogy a többi megmarad.

-removeClass() Eltávolít egy class értéket,de megtartva a többit.

**HTML tartalom kiolvasása és beállítása:**

Ezzel az al osztályba tartozó elemek egy dőlt a betűt tartalmaznak.(ha van valami bennük,az felülíródik.

$(".al").html("<i>a</i>");

Kiolvasni is kilehet a tartalmát:

alert($(".al").html()); Ezt is a html függvénnyel lehet.

Fontos:A html() függvény a DOM innerHTML –ére alapszik.Illetve a html() XML dokumentumra nem alkalmazható!

Megadhatók tagek is,tehát gyerekeket hozhatunk vele létre.

A .html() tagfüggvény a javascript innerHTML() függvényére támaszkodik,viszont a .html() a végrehajtás előtt még sok előkészületet végez el,ezért lassabb lehet ez a módszer.

Megoldás: $(”#alma p”)[0].innerHTML(); látható,hogy a JQuery objektumot átalakítottuk JS objektummá és arra már meghívható a natív innerHTML() metódus.

**Szövegtartalom kiolvasása és beállítása:**

$(".al").text("ez egy p tag szövege");

alert($(".al").text());

A text() függvény nemsokban különbözik a html() függvénytől.Annyi a különbség,hogy text()-ben írhatunk < és > jeleket,ezt szimbólumként és nem tagként értelmezi.

$(".al").text("ez egy <p> tag szovege");

**Kijelölés részletesebben:**A css kijelölőelemeket is magába foglalja a jquery,de sajátokat is definiál.

A kijelölt halmazon bármelyik JQuery függvény használható:

$("#dives p").text("ez egy dives <p> tag szovege");

A dives idjű divben lévő összes p-hez hozzárendelünk egy oszt class attribútumértéket.

$("#dives p").addClass("oszt");

div > a ez azokat az a-kat jelöli ki melyek közvetlen gyermekei a divnek!

A következő megoldás ugyan olyan eredményt ad mint a > szimbólum,csak a children() metódus gyorsabb.Gyorsabb mert nem kell kijelölőként értelmezni.

$("#dives").children("p").text("ez egy dives <p> tag szovege"); (gyors)

alert($("#b").children().length); //a b id-jű elem gyerekeinek a száma.

**Konkrét testvér kijelölése:**

Közvetlen rákövetkező testvér kijelölése: $("#dives > p + p").text("testver p");

p + p

$(".al + p").text("testver p"); az al classú elem közvetlen testvére

Ha egy elem minden testvérét kiszeretnénk jelölni,függetlenül attól,hogy közvetlen testvér-e akkor a siblings() tagfüggvény használatos:

$(".al").siblings("p").text("testver p");

A siblingsbe felsoroljuk mely testvéreket jelölnénk ki.

A kijelölt li halmazból az első kiválasztása:

$("#a li:first").css("background-color","red");

létezik :

:last

:even a páros elemekre illeszkedik nullától indulva

:odd a páratlan elemekre illeszkedi

:eq(n) az n-edik sorszámú elemre illeszkedik

:lt(n) mindegyik elemre illeszkedik,melynek a sorszáma n alatti

:dt(n) mindegyik elemre ileszkedi,melynek a sroszáma n feletti

A lista első elemének kiválasztása többféle képpen is történhet:

-li:first

-li:eq(0)

-li:lt(1)

Az indexelés nullától kezdődik!

**Éppen mozgó elemek kijelölése:**

$(”div:animated”);

csak azokra illeszkedi,melyek épp mozognak.(Az összes olyan divet kijelöli,aminek az animációs várakozó sora nem üres.)

A nem animált elemek animálása:

$(”div:not(div:animated)”).animate({height:100});

Néha meg kell vizsgálnunk,hogy egy elem mozog-e:Az is() tagfüggvénnyel végezhető ez a művelet.Egy változóba eltesszük a jqueryvel kiejlölt elemünket.

var myVar=$(”#alma”);

if(myVar.is(”:animated”)){

csinálunk valamit;

}

**Elemek kijelölése a tartalmuk alapján:**

Megoldható az hogy tartalmom alapján jelöljünk ki:

$(”p:contains(’Alma’)”); Azokat a bekezdéseket jelöli ki,melyek tartalmazzák az Alma stringek(kis nagybetű érzékenység van a JQueryben)

Kereshetünk aszerint is,hogy milyen beágyazott elemei vannak a keresett elemnek:Azokat a diveket jelöljük ki,melyek beágyaznak p taget.

$("div:has('p')").css("background-color","red");

A has() tagfüggvényen belül még lehet pontosabb definiíciót megadni!

$("div:has('p > i')").css("background-color","red");

**A nem illeszkedő elemek kijelölése:**

**A**zokat az anchorokat jelöli ki,melyek nem a div.important szakaszon vagy a nav classon belül vannak:

$(”a:not(div.important a, a.nav)”);

Kijelöli az összes a-t kivéve a mostanit(amire pl rákattintottak) és azoknak módosítja az osztályát:

$(”a”).clicked(function(){

$(”a”).not(this).addClass(”not-clicked”);

});

A következő kód azokat az elemeket adja,melyeknek a beállítása display=”none”,de nem visibility=”hidden”:A csillag(\*) az egész DOM-ot kijelöli.

$(”\*”).filter(function(){

return $(this).css(”display”)===”none” && $(this).css(”visibility”)!==hidden;

});

**Elemek kijelölése jellemzők alapján:**

Kijelöli azokat az anchorokat,amelyek a href tulajdonsága a google,és a title tulajdonságuk alma értékű:

$("a[href='http://www.google.hu'][title='alma']").css("background-color","blue");

Kijelöl minden olyan elemet,aminek van title és href attribútuma:

$(”\*[title][href]”);

[jellemző=érték]

[jellemző!=érték]

[jellemző^=érték]

[jellemző$=érték]

**Űrlapelemek szűrése:**A JQuery erre a célra sok eszközt biztosít.

:text

:password

:file

:hidden

:button

:submit

:checkbox

ezek a <input type=”button”>-t jelölik ki

Ezeken kívül létezik egy :input szűrő is ami minden inputelemet kiválaszt és arra akár lehet szűrni is:

$(”:input:not(:hidden)”);

**Egyéni szűrő létrehozása:**Ennek az a lényege,hogy saját szűrőt hozhatunk létre,és ezáltal nem kell mindig lekódolni a szűrési feltételt,hanem meghívható ezzel a kifejezéssel.

$.expr[":"].sajat = function(elem){

return $(this).width() > 100;

}

$("div:sajat").css("background-color","black");

Látható,hogy a :hidden :radio kijelölők is ilyen saját kijelölőknek számítanak.Itt a sajat névvel láttuk el a kijelölőnket,és azon elemeket jelöli ki.melyekek a szélessége nagyobb mint 100 pixel.

**Nagyon pontos kijelölés:**A filter() taggfügvény segítségével függvényt definiálhatunk és amit az visszaad az lehet a halmaz.

Azokat az elemeket jelöli ki,melyek rendelkeznek háttérképpel.

$(”\*”).filter(function(){

return $(this).css(”backgroundImage”);

});

szokványos objetum ra hivatkozás this

JQuery objektumra hivatkozás $(this)

$(”div”).filter(function(){

var width = $(this).width();

return width > 200;

}); //ez azokat a diveket jelöli ki,melyeknek a szélessége nagyobb mint 200 pixel.

**This és $(this) problematika:**A this kulcsszó js-ben sok mindent jelent(objektumon belüli adattag hivatkozás).

Az alábbi sorban a this azt jelenti,hogy a click eseményre meghívódik a clicked() metódus és paraméterként megkapja az ”a” objektumot.Tehát itt a this a DOM egyik elemét adja,amelyiken az esemény bekövetkezett.

<a href="http://www.google.hu" id ="alma" onClick="clicked(this);">Link</a>

Az alábbi kódrészleten magyarázom el a $(this) buktatóját.Látható a clicked() függvényen,hogy a paraméterül kapott változó(ami a kívülről kapott this) értéke „alma” ,viszont a már függvény belsejében $(this) hivatkozásnak az id-je undefined.Igen az undefined,mert a függvény belsejében a this,az a window objektum.Ez ,hogy lehetséges?

<body>

<p id="demo">alma</p>

<a href="http://www.google.hu" id ="alma" onClick="clicked(this);">Link</a>

<script>

function clicked(t){

alert(t.id); //alma

alert(this.id); //undefined

alert(this===window); //true

}

$(document).ready(function(){

$("#demo").click(function(){

$(this).addClass('clicked');

alert($(this).attr("class"));

setTimeout(function(){

$(this).removeClass('clicked');

alert($(this).attr("class"));

},1000);

$(this).unbind();

});

});

</script>

A Javascript a függvényei paramétereinek egy záradékot hoz létre.A JS függvények több mélységig egymásba ágyazhatóak.Egy függvény nem csak a saját paramétereit olvashatja és írhatja,hanem azokét is amelyek őt beágyazták(ez a szabály mindig érvényes,még akkor is ha a külső függvény már visszaadja a vezérlést és a belső függvény meghívására később,esetleg egy esemény bekövetkeztekor kerül sor).

Azzal ha a $(this) értékét egy változóba tesszük,azzal elérhetővé válik a beágyazott függvények számára is.

Tehát a fenti kódban a setTimeout() azért nem működik helyesen,mert amikor a clicked class attribútumot szeretnénk törölni,akkor ő a window objektumra csinálná.

Tehát a helyes megoldás:A $(this) értékét egy változóba tesszük ,és azt ugyebár a beágyazott függvényekből lazán elérhetjük,és tényleg arra fogunk hivatkozni amire szeretnénk,és nem a window objektumra:

$(document).ready(function(){

$("#demo").click(function(){

var $magam = $(this);

$magam.addClass('clicked');

alert($magam.attr("class"));

setTimeout(function(){

$magam.removeClass('clicked');

alert($magam.attr("class"));

},1000);

$(this).unbind();

});

});

Másik péla,ahol azokat az elemeket jelöli ki,amelyeknek 10 vagy 20 gyermeke van:

$(”\*”).filter(function(){

var children = $(this).children().length;

return children ===10 || children ===20;

});

**Környezetként megadott kijelölés:**

$(”p”,”.bekezd”);

$(”.bekezd p”);

A kettő ugyan az,viszont az első gyorsabb!

Talán itt érdemes megemlíteni,hogy a $(”.class”) kijelölés sokkal lassabb ,mint az id-n keresztüli kijelölés, ezért sokat gyorsíthat a dokumentum betöltődésén,ha például $(”#alma.classos”) kijelölést használunk,hisz így leszűkítjük,hogy ennek az id-jű tagnek a leszármazottja az a classu tag,tehát nem kell az egész dokumentumot bejárni.

**Egyéni szűrő kijelölő létrehozása:**

jQuery.expr[”:”].uj=function(elem){

return $(elem).css(”display”)===”inline”;

}

$(”div a:uj”);

**Eredményhalmaz bejárása:**A kapott eredményhalmaz bejárható egyesével az each() tagfüggvény segítségével.

Az alábbi példában deklarálunk egy tombot,aminek nem tudjuk megmondani előre az elemszámát.Majd JQuery segítségével kijelöljük az összes anchort,majd az each() tagfüggvénnyel bejárjuk minden elemét.Az eachben definiálunk egy metódust,ami tomb elemeibe ,az éppen aktuális DOM tag($-val JQuery objektummá alakítottuk,hogy meghívhassunk rá JQuery metódust) href attributumának értékét rakja bele.Az aktuális DOM elemet a this kulcsszóval érjük el:

var tomb = new Array();

$("a").each(function(i){

tomb[i]=$(this).attr("href");

});

alert(tomb[1]);

Minden páros és páratléan listaelem piros és kék hátteret kapjon:

$("#a li").each(function(i){

if(i%2==0){

$(this).css("background-color","blue");

}

else{

$(this).css("background-color","red");

}

});

Az i paraméter reprezentálja a sorszámot.

**Eredményhalmaz egy elemre szűkítése:**Az eq() tagfüggvényként való használtata.

A következő példában az összes li elemnek beállítjuk a háttérszínét pirosra,majd,hogy kiemeljük a második li-t ezért annak kékre állítjuk a színét:

$("li").css("background-color","red").eq(1).css("background-color","blue");

**A JQuery objektum átalakítása nyers DOM objektummá:**Amikor Jqueryvel kijelölünk valamit,akkor az eredményhalmazt JQuery objektumként kapjuk és nem DOM-ként.

Ebből kifolyólag csak JQuery függvényeket használhatunk rajtuk.

A következő példában az első div elemet dom objektummá alakítjuk ,és annak lekérjük az innerHTML tulajdonságát egy inner nevezetű változóba:

var inner = $("div")[0].innerHTML;

alert(inner);

[1] ez a második divet jelölte volna ki…stb…

Megoldható egy másik módszerrel is.Ez pedig a JQuery beépített get() metódusa(ez nem az AJAX-os get),ami egy tömbben adja vissza a dom objektumokat.

var tomb = $("div").get();

$.each(tomb,function(key,value){

alert(value.innerHTML);

});

Itt az összes div elemet kiejlöljük,majd a get() metódussal átalakítjuk és tömbbre rakjuk.Ez azért jó,mert a get() álltal visszakapott tömbön használhatunk metódusokat,például .reverse() stb….

**Az elem sorszámának kinyerése a kijelölésből:**

A következő példában egy div elemen belüli p tagekre olyan kattintási eseményt állítunk,hogyha rákattintanak az egyikre,akkor az kiiírja a saját sorszámát a kijelolt halmazon belül.

<div>

<p>DOM is <i>ready</i>.</p>

<p>DOM is <i>ready</i>.</p>

<p>DOM is <i>ready</i>.</p>

<p>DOM is <i>ready</i>.</p>

</div background-color="black">

<script>

$("div p").click(function(){

alert("Rakattoltal a " + $("div p").index(this) + "u elemre");

})

</script>

Ha az index() metódus nem találja az elemt,akkor -1-et ad vissza.

**A kijelölt halmazon belül egy intervallum kiejlölése:**

$("p").slice(1,4).css("background-color","red"); Ez a slice()tagfüggvény a második elemtől az ötödik elemig kijelöl(p tagekre nézve).

**JQuery beállítása a más függvénykönyvtárakkal való ütközés elkerülésére:**Probléma lehet,ha használni szeretnénk Jqueryt és mondjuk Prototype JS függvénykönyvtárakat akkor mind2 használja a $ szimbólumot.

Egyik megoldás:

var j = jQuery.noConflict();

j(”p”).css(formázás); innentől kezdve használhatjuk a saját rövidítésünket.

Másik megoldás:Betokozás átmenetileg elérhetővé teszi a $-t számunkra,de más könyvtárak metódusait amik a $ változóra hivatkoznak innen nem elérhetőek.

(function($){

$("p").css("background-color","blue");

})(jQuery);

**Plug-in használat:**Ugyan úgy script tagek közé kell megadni a forrást(ami lehet más szerveréről is).

<!DOCTYPE html>

<html>

<head>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<script type="text/javascript" src="http://malsup.github.com/jquery.cycle.all.js"></script>

</head>

<body>

<div class="slideshow">

<img src="http://malsup.github.com/images/beach1.jpg" width="200" height="200" />

<img src="http://malsup.github.com/images/beach2.jpg" width="200" height="200" />

<img src="http://malsup.github.com/images/beach3.jpg" width="200" height="200" />

<img src="http://malsup.github.com/images/beach4.jpg" width="200" height="200" />

<img src="http://malsup.github.com/images/beach5.jpg" width="200" height="200" />

</div>

<script>

$(document).ready(function() {

$('.slideshow').cycle({

fx: 'fade' // choose your transition type, ex: fade, scrollUp, shuffle, etc...

});

});

</script>

</body>

</html>

**Tömb bejárása az each() tagfüggvény segítségével:**

A következő feladatban létrehozunk egy 4 elemű tömböt és annak az elemeit hozzáadjuk egy bekezdésként a div tagünkhöz.Az each függvény bejárja a kapott tömböt és arra lefuttat egy functiont amit magunknak határoztunk meg:

<script>

$(document).ready(function(){

var tomb=[1,2,30,40];

$.each(tomb,function(index,value){

$("<p>" + value + " index:" + index + "</p>").appendTo("div");

});

});

</script>

**Lehetséges gyorsítás:**A $.each() helyett használjuk a for ciklust tömb bejárásra,mert gyorsabb!

Az alábbi megoldás hihetetlen gyorsulást jelent.Ez átírható for és do..while ciklusra is.

A ++i művelet eggyes böngészőkben gyorsabb mint az i++ !

Eggyesítjük a ciklusváltozó növelését és ellenőrzését(kevesebb lépés===gyorsaság)!!!!!!!!!!

Ezzel van azomban egy bökkenő:A ciklus csak akkor működik,ha a tömbben nincs különleges érték(null,undefined,0 vagy false)!Ez sajnos gáz,már a nulla érték is elrontja!!!!!!!!!!!!

var tomb = [1,2,3,4,5,6];

var valt=0;

var i = -1;

var n = tomb.length;

while(++i < n){

valt+=tomb[i];

}

**HTML elembeszúrás gyorsítása:**Ha a jquery html() tagfüggvényét használjuk,akkor számolnunk kell azzal,hogy ha sokminden van a myhtml változóba html elemek amiket be akarunk szúrni a DOM ba,akkor az sok időt igénybe vehet,mert a html()tagfüggvény az innerHTML JS függvényre támaszkodik,de mellette sok ellenőrzést is végez.

$("#b").html(myhtml);

Az alábbi megoldás lehet.

$("#b")[0].innerHTML=text;

**Objektum összes jellemzőjén is végigmehetünk az each() tagfüggvénnyel:**

function printCard(){

line1="Name:" + this.name;

line2="Address:" + this.address;

line3="Work:" + this.work;

line4="Home:" + this.home;

document.write(line1 + line2 + line3 + line4);

}

function card(name,address,work,home){

this.name=name;

this.address=address;

this.work=work;

this.home=home;

this.printCard=printCard;

}

var obj = new card("Viktor","Forró","ME","Lakás");

Eddig létrehoztuk az objektumot majd:

A key a változók nevei(name,address,home …)

A value pedig a bennük tárolt érték

$.each(obj,function(key,value){

$("<p>" + value +"</p>").appendTo("div");

});

Vegyük észre hogy ez különbözik a másik each()-től ami $(”div”).each(function(i){

csinál vmit;

}); formában jelenik meg!

Soha nem használjuk a for each ciklust JQuery objektum vagy tömb bejárására,mert ha tartalmaz más objektumot vagy functiont akkor azt is bejárja!!!JQuery objektumokat .each() függvénnyel járhatjuk be.

**A tömbök szűrése a JQuery.grap tagfüggvény segítségével:**

A következő példában a tomb elemeit szűrjük a grep() függvénnyel.Return után minden tombelemre kiértékelődik a feltétel,és ha az értéke igaz,akkor belekerül az új tömbbe,ha hamis akkor nem:

var tomb=[1,2,30,40];

tomb = $.grep(tomb,function(value,i){

return (value > 10);

});

**Tömbelemek bejárása és módosítása:**

A következő feladatban minden tömbelemnek megnöveljük 100-val az értékét:

var tomb=[1,2,30,40];

tomb = $.map(tomb,function(value,index){

return value+100;

});

A $.map() az a neki átadott tömböt egy másik tömbre képzi.Viszont nem módosítja azt véglegesen(tehát nem pointert kap).Ezért is kell az eredeti tomb tombnek visszaadni a $.map visszatérési tömbjét,mert ha nem adnánk vissza,akkor nem módosítaná végleg.

**Tömbök egyesítése a JQuery.merge tagfüggvény segítségével:**

var tomb=[1,2,30,40];

var masiktomb = $.map(tomb,function(value,i){

return value+100;

});

var egyesitett = $.merge(tomb,masiktomb);

$.each(egyesitett,function(index,value){

$("<p>" + value + " index:" + index + "</p>").appendTo("div");

});

Az új egyesített tömbben az elsőnek megadott tömb elemei szerepelnek elől!

**Eseménykezelés:**

A bind tagfüggvény segítségével adhatunk a kijelölt halmaznak valamiféle eseményre valami történést.Az alábbi példában a kattintásra egy alert válasz érkezik:

$(document).ready(function(){

$("p").bind("click", function(){

alert("The paragraph was clicked.");

});

});

**Adatok hozzákapcsolása a DOM-hoz:**

A lényeg,hogy objektumhoz(kijelölt tagek halmaza)rakhatunk datát.A coolColornak különböző esetekben különböző értéket adunk.Ez a data olyasmi lehet mint az attribútum szerintem:

$("p").each(function(index, el){

if(index % 2 == 0)

$(this).data('coolColor', 'Orange'); // Set the data

else

$(this).data('coolColor', 'Purple'); // Set the data

}).click(function(e){

alert($(this).data('coolColor')); // Retrieve the data

});

**Ha a $(this)-re szeretnénk hivatkozni:**Néhány esetben ez nem működik

$(this).removeClass();

Ezért tegyük bele egy változóba:

var th = $(this);

th.removeClass(); ez így minden esetben működni fog!

**Amit lehet egyszerűsíts:**Ha sokszor kell ugyan azt csinálni,használj function()-t !

Az alábbi példa nagyon érdekes.A bekezdésekre két eseményt is állítunk,és mindkettőre ugyan az történik.A triggerrel rögtön ki is váltunk egy eseményt a kettő közül(itt a clicket),tehát nem is kell kattintanunk,máris kiváltódik az esemény és végrehajtódik aminek végre kell hajtódnia:

$("p").bind("click mouseover",function(){

$(this).css("color","blue");

$("#time").html("<p>beillesztett</p>");

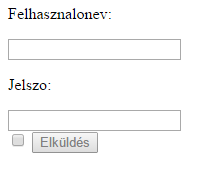
}).trigger("click");

**A tag value értékének módosítása:**

$(”input”).val(’felhnev’); ezzel értéket állítunk neki

var valt=$(’input’).val(); ez visszaadja az értékét

**Olyan regisztrációs ablak készítése,amiben ha nincs bepipálva a licensz elfogadás,akkor nem lehet továbblépni(gombra kattintani):**



<body>

<div id="kulso">

<form>

<p>Felhasznalonev:</p>

<input type="text">

<p>Jelszo:</p>

<input type="text"><br>

<input type="checkbox" id="boxos">

<input type="submit" id="subos" disabled>

</form>

</div>

<script>

$(document).ready(function(){

$("#boxos").click(function(){

if($("#boxos").is(":checked")){

$("#subos").removeAttr("disabled");

}else{

$("#subos").attr("disabled",true);

}

});

});

</script>

A html body részében úgy hoztuk létre a submit gombot,hogy defaultból disabled,azaz nem lehet rákattintani.Majd a checkboxra állítottunk egy kattintási(click)listenert,ami azt csinálja,hogy az is()tagfüggvény segíttségével megvizsgálja,hogy a checkbox a checked,azaz be van-e pipálva vagy sem,és ennek megfelelően állít az attribútumon.

**Belekattintás figyelő:**

Mivel általánosan az egész szövegdobozt kijelöltem,ezért kell a $(this),hogy az aktuálist formázzuk,amire kattintottak:

$(":text").focusin(function(){

$(this).css("background-color","red");

});

**Ha már nem benne vagyunk,és erre szeretnénk eseményt akkor blur():**

$(":text").blur(function(){

$(this).css("background-color","white");

});

**Egynél több selector megadása:**

$(”#alma”,”.oszt”).css(formázás);

**JQuery objektumok gyorstárazása:**Probléma lehet,hogy sokszor kell kiértékelni a $(”#div”) vagy $() akármi halmazkijelölőket.

Mi a baj az alábbi kódrészlettel?A $() kijelölőket minden mousemove eseménykor meghívjuk,ami lehet percenként 100 is,és ez hatalmas belassulást eredményez.Állandóan kiértékelni,bejárni a DOM-ot sok idő.

$(\*).mousemove(function(){

$(".alma").html("<b>alma</b>");

$(".korte").html("<b>körte</b>");

$(".szilva").html("<b>szilva</b>");

$(".barack").html("<b>barack</b>");

$(".narancs").html("<b>narancs</b>");

});

Megoldás:Tegyük változókba a $() kijelölőket,és ezálltal csak egyszer kell kiértékelni őket nem pedig 100 milliószor.

Írjuk a gyorstárba a kiértékeléseket:

var $alma = $(".alma"),

$korte = $(".korte"),

$szilva = $(".szilva"),

$barack = $(".barack"),

$narancs = $(".narancs");

$(\*).mousemove(function(){

$alma.html("<b>alma</b>");

$korte.html("<b>körte</b>");

$szilva.html("<b>szilva</b>");

$barack.html("<b>barack</b>");

$narancs.html("<b>narancs</b>");

});

Felmerülhet a kérdés,hogy miért rakok a változók neveinek elejére $ jelet.Azért ,hogy jelezzem magamnak,hogy ez JQuery objektummal machinál.Nem történne semmi hiba,ha nem tennék oda $ jelet,de ez megszokás.

Fontos:Az id-s kijelölés sokkal gyorsabb mint a classos.

$(”#alma”) gyors

$(”.korte”) lassú

A classos kijelölőnek listát kell készítenie a DOM objektumairól,és kiértékelni melyik objektumnak az a classa(nagyon időigényes munka).

Ha minden féle képpen classos kijelölésre van szükségünk,akkor is gyorsathatunk.A classos DOM objektumokat egy divbe rakhatjuk vagy bármibe,de annak adunk egy id-t és már nem is kell az egész DOM-ból listát készíteni:

$(”#key .alma”);

**Hide és show függvények:**

A css display : none –t használjuk.Megadható időegység is.

.hide(1000,function(){ });

.show(1000,function(){ });

**Hover event:**A felémenés és elhagyás egyben kivitelezése.

$(”#dives”).hover(function(){

Ha felémegyünk mi történjen

},function(){

Ha elhagyjuk mi történjen

});

Érdemes legörfülő menüknél ezt használni.

**Window documentes eventek:**

.resize()

.scroll() amikor görgetünk az ablakon

**Effektek:**

$(”div”).fadeIn(3000,function(){}); a 3000 milisecundum alatt és a functiont még végrehatja.

fadeOut();

fadeToggle(); kaphat időértéket is functiont is.Itt az a lényeg ha eltüntetett akkor előhozza majd eltünteti,ha előhozott akkor eltünteti majd előhozza.

fadeTo():Ez egy nagyon hasznos függvény.A képet elhalványíthatjuk(mint naruto-arenan azok a karakterek képei,akik még nincsenek megszerezve).

Az alábbi program azt csinálja,hogy ha a képre kattintunk akkor 0,1 másodperc alatt(100 milisec) elhalványul,majd a szin dataját beállítja szürkére(classt nem sikerült hozzáadni).Majd minden képre állítunk egy click eseménykezelőt,ami ellenőrzi,hogy az épp kattintott kép szind data adata szurke-e,mert ha igen akkor alert figyelmeztetést írunk.

<img src="Banner3.PNG" id="alma">

<script>

$(function() {

$("#alma").click(function(){

$(this).fadeTo(100,"0.2",function(){

$(this).data('szin', 'szurke');

});

});

$("img").bind("click",function(){

if($(this).data("szin")=="szurke"){

alert($(this).data("szin"));

}

});

});

</script>

A fadeTo()-nak kötelező megadni,hogy mennyi idő alatt menjen végbe a halványulás,illetve azt is,hogy milyen mértékben halványuljon(0,2 a példámban)!

slideDown():Itt is megadható idő és function.Ez a képet vagy akármit fentről lefelé engedett függönyként jeleníti meg.Fentről lefelé megjelenít.

slideUp():Lentről felfelé eltünteti a képet.

**animate() használata:**Ezzel a függvénnyel megadott idő alatt animációk készíthetők.

Fontos,hogyha valamennyivel el akarjuk tolni,akkor absolute position kell és a megadott mértékben a saját pozíciójához képest eltoljuk amit el kell tolnunk.Az animate() metódusba kapcsos jelek között megadhatjuk,hogy milyen css változások legyenek,utána milisecbe,hogy ezek mennyi idő alatt menjenek végbe,illetve harmadik paraméterként,amit én nem adtam lehet egy function() .

<style>

#alma{

position:absolute;

}

</style>

</head>

<body>

<img src="Banner3.PNG" id="alma">

<script>

$(function() {

$("#alma").click(function(){

$(this).animate({left:'250px',top:'150px'},3000);

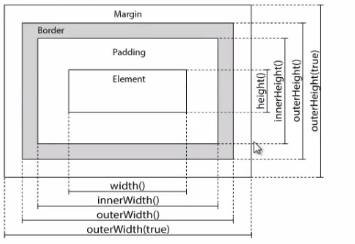
});

});

stop():Ezzel megállítható az animáció.Slideok stb…(Egy adott időpillanatnál megállíthatunk)

finish():Ezzel teljesen leállítjuk az animációt.

var hossz=$(”img”).queue().length; (Ezzel azt a számot adjuk a változónak értékül,ami azt határozza meg,hogy hány animáció van alkalmazva azon az elemen.)



Ezekkel a függvényekkel állíthatóak, és visszakaphatóak az értékei.

**Késleltetés:**A delay() függvénnyel.

3 másodperc késéssel kezdi el csinálni:

$(this).delay(3000).fadeTo(100,"0.2",function(){

$(this).data('szin', 'szurke');

});

**Eredeti figyelő beállítása:**Minden objektumra létezik valami default eseménykezelő,például az anchorokra a kattintásra navigálás.De mi van ha ezt valamikor megszeretnénk szüntetni,és más eseményt szeretnénk kiváltani?

$("a").click(function(event){

event.preventDefault();

$("#news").html("<p>alma</p>");

});

Az anchorokra a kattintás eseményt defaultoljuk(azaz töröljükl),és egy más történést állítunk rá.

**Az ablak és a dokumentum:**A dokumentum szélessége és magassága sok esetben eltér az ablak szélességétől és magasságától.A legtöbb esetben a dokumentum magassága nagyobb mint az ablak magassága(hisz a dokumentum legörgethető és lejjebb is van,míg az ablak mérete nem lehet akármilyen hosszú,bele kell férnie a képernyőbe.).

A dokumentum szélessége mindig eléri legalább az ablak szélességét,de lehet nagyobb is.

Az alábbi módon kérdezhető le a dokumentum és az ablak szélessége és magassága:

alert($(document).height() + " " + $(document).width());

alert($(window).height() + " " + $(window).width());

Minden elemhez meghívható a .width() és a .height() metódus,viszont ez nem mindig elég:

<style>

#b{

width:200px;

height:300px;

padding:1px;

border:1px solid black;

background-color:red;

}

</style>

</head>

<body>

<div id="b">

</body>

Ha meghívjuk a width() vagy height() metódusokat akkor az alap beállítás értékét kapjuk:

alert($(”b”).width()); // ez 200

alert($(”b”).innerWidth()) // ez 202 mivel beleszámolja már a padding értéket is mind2 oldalról 1 px-et.

alert($(”b”).outerWidth()) // ez 204 ,hisz itt már a bordert is beleszámolja

Ha lenne is margin,azt akkor sem számolná hozzá.

Az innerWidth/innerHeight és az outerWidth/outerHeight függvények document és window objektumra NaN értékkel térnek vissza:

Taház Nan-t eredményez: $(document).innerWidth();

**Bind:**

$(”p”).bind(”click keydown”,function(){

});

Több esemény is átadható egyszerre,ez azért kényelmes,mert ha kattintásra és billentyű lenyomásra is ugyan az történik,akkor itt együtt kezeljük a kódot és rövidebb is lesz.

$(”p”).unbind(”click keydown”);

$(”p”).unbind(); //ez minden eseményt levesz róla

Névtér is képezhető az eseményeknek:

$("#demo").bind("click mouseover.alma",function(){

alert("alma");

});

$("#demo").unbind(".alma");

Egyszerűen csak odaírjuk .névtér és erre hivatkozhatunk törléskor és azok fognak törlődni.

**event.target:** Ez a tulajdonság arra a konkrét elemre hivatkozik,amelyikre az esemény vonatkozik.

$("div").bind("click",function(e){

alert(e.target);

});

Lehetséges több kezelő is egy elemre:

$("div").bind("click",function(e){

alert(e.target);

/\*e.stopImmediatePropagation();\*/

}).bind("click",function(){

alert("almaalma");

});

$(document).bind("click",function(){

alert("alma");

});

A szabványos DOM események legtöbbje felúszik.Ez azt jelenti,hogy miután az esemény elindult a célobjektumon, felúszik a szülőcsomópontba és ott is kiváltja ezt az eseményt.Ez a folyamat addig folytatódik míg el nem érjük a document objektumot vagy eseménykezelőben nem hívjuk meg az event.stopImmediatePropagation() metódust.

Tehát ha a fenti kódban a kikommentezett e.stopImmediatePropagation() részt mégis bennehadjuk,akkor az almaalma szöveg nem fog kiíródni,hisz megszüntetjük a további kezelők lefutását.Így a többi befoglaló kezelése sem történik meg,tehát a documenthez tartozó alma kiírás sem fog így meghívódni,amúgy ha marad kikommentezve akkor még az is meghívódik a document click kezelője.

**Load():**Ezzel a függvénnyel bármit becsavarhatunk.Akár txt akár html minden átjön gond nélkül.

<body>

<div id="demo">

</div>

<input type="button" name="Gomb" value="Gomb">

<script>

$(document).ready(function(){

$("input[type='button']").click(function(){

$("#demo").load("szov.txt");

});

});

</script>

</body>

A demo id-s divbe belecsavarrjuk a szov.txt file tartalmát.

**$.get():**Ajax egyik eszköze.

HTML kód:A feedback divbe pakoljuk bele amit a php ad vissza.

A $.get() függvénynek 3 paramétere van.Az első,hogy melyik php-nek küld,a második,hogy mit küld át(Ez bizony JSON),a harmadik pedig egy data paraméteres function.A data paraméter az,amit a php vissza echozik.És mi azt beletesszük a feedback divünkbe.Ez futás időben történik(csodálatos).

<body>

<input type="text" id="user"><br>

<input type="text" id="pw"><br>

<input type="button" id="submit" value="Submit">

<div id="feedback"></div>

<script>

$(document).ready(function(){

$("input[type='button']").click(function(){

$.get("index.php",{

username:$("#user").val(),

password:$("#pw").val()

},function(data){

$("#feedback").html(data);

});

});

});

</script>

</body>

PHP kód:Változókba belevágjuk a megkapott JSON adatokat,majd string konkatenációt küldünk vissza a data paraméterbe,és az kerül a divbe.

<?php

$username=$\_GET['username'];

$password=$\_GET['password'];

echo "User:".$username." "."pw: ".$password;

?>

**$.post():**Biztonságosabb és nagyobb adatokat is tologathatunk ,mint a getnél,a szintaxisa ugyan az!

**Error() Success() :**

$(document).ready(function(){

$("input[type='button']").click(function(){

$.post("index.php",{

username:$("#user").val(),

password:$("#pw").val()

},function(data){

$("#feedback").html(data);

}).error(function(){

$("#feedback").html("<p>hiba</p>");

}).succes(function(data){

alert("Minden oké");

});

});

Az a lényeg,ha sikertelenség van akkor azt az error() függvényben kezelhetjük,és azt is kezelhetjük ha sikerült a küldés és a fogadás.

**Ajax() függvény:**

$(document).ready(function(){

$("input[type='button']").click(function(){

$.ajax({

type: 'POST',

url: 'index.php',

data :{

username:$("#user").val(),

password:$("#pw").val()

},

statusCode:{

404: function(){

$("#feedback").html("Page not found");

},

505: function(){

$("#feedback").html("Gase");

}

},

success:function(data){

$("#feedback").html(data);

}

});

});

});

Az ajax() függvény egész sokat tud.Megadhatjuk milyen küldés történjen.url-be hogy melyik phpnak.

a data az a JSON.

satusCodeba kezelhetjük különböző fellépő hibáinkat.