Javascript

Domonkos Sándor Balázs Phd hallgató

Programozási alapismeretek felzárkóztató kurzus

Jelen tananyag a Szegedi Tudományegyetemen készült az Európai Unió támogatásával.

Projekt azonosító: EFOP-3.4.3-16-2016-00014

Szegedi Tudományegyetem
Cím: 6720 Szeged, Dugonics tér 13.
www.u-szeged.hu
www.szechenyi2020.hu

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Oktató: **Domonkos Sándor Balázs**	**Programozási alapismeretek**	Becsült olvasási idő: **15 perc**

Javascript Harmadik Lecke

- Változók
- Adat típusok
- Operátorok
- Objektumok
- Eventek

Változók

Legegyszerűbb tárolók adat értékek tárolására a javascriptben. A javascript gyenge típusos nyelv ami azt jelenti,hogy változóknak az értékadása és a rajtuk végrehajtott oprerációk lazán vannak csak megkötve.

Egy változó **létrehozása** a **var** keyworddel történik

var pi = 3.14;

Ahol a **var** a kulcsszó a létrehozásra a pi a **változó neve** a = az **operátor** ami megadja ,hogy a pi értéke legyen a 3.14 numerikus értékként.

var person = "John Doe";

Itt pedig a “ “ használatával **szövegként** tároljuk le a John Doe szavakat a person nevű változóba.

var carName = "Volvo";
.../valami kód/
carName = "VW";

Ezzel a konstrukcióval pedig a **carName**-t létrehozzuk és beletöltjük a Volvo szót és valahol később a kódban ugyanezt a carName változót **felülírhatjuk** más **szövegre** sőt más **adattípusra** is akár.

var person = "John Doe", carName = "Volvo", price = 200;

Tömegesen is létrehozhatunk változókat a var szó után , jellel elválasztva őket. Értékadás nélkül is lehet változót létrehozni ez esetben undefined lesz a visszatérési értéke amíg nem töltjük fel adattal.

Adat típusok

```
var length = 16;                               // Szám
var lastName = "Johnson";                      // Szöveg
var x = {firstName:"John", lastName:"Doe"};    // Objektum
```

A különböző **típusoknak** azért kell létezniük,hogy amikor **műveleteket** hajtunk végre a változókkal akkor **egyértelmű legyen**,hogy milyen **eredményt** szeretnénk kapni. A JS a kiértékelések folyamán **balról jobbra hajtja végre** a kiértékelést így előfordulhat,hogy más sorrend más és más eredményt adhat mint az alábbi példáknál:

var x = 16 + 4 + "Volvo";
Eredménye 20Volvo

var x = "Volvo" + 16 + 4;
Eredménye Volvo164

Itt látható,hogy az első esetben a JS **számként** kezeli a változókat a 16 + 4 számokta amíg a “volvo” -ig el nem ér azt pedig már **szövegként** fogja odarakni az érték végére.
A második esetben pedig mivel egy **szöveggel** indítunk így a 16 és 4 számokat is **szövegként** fogja kezelni, erre láthatunk még példát az **operátorok** résznél is.

A **javascript változók dinamikusok** ami azt jelenti,hogy menetközben tudnak adat típust váltani ahogy mindig új és új értéket kapnak :

```
var x;          // x még nincs meghatározva
x = 5;          // x szám
x = "John";     // x már szöveg
```

Tömbök

Több változó tárolására használt változók amik több értéket is tudnak egyszerre tárolni, a JS tömbök 0-tól vannak **indexelve** ami azt jelenti,hogy az első elemet [0] ként tudjuk hivatkozni.

A tömb eleje és vége a [ ] jelekkel jelöljük az elemeket , jellel választjuk el.
Var i = [“VW”,”Mercedes”,”Volvo”]
Hozzáférés adott elemhez i[elemszáma] módon pl var x = i[0] esetén az x értéke “VW” lesz mivel az i tömb első eleme ezt tartalmazza.

Objektumok

Hasonlóan a tömbökhöz egy **speciális változó** ami több változót (tulajdonságok) és ezek értékeit tudja magába foglalni, illetve **kód blokkokat** (functions) is tudunk definiálni egy objektumon belül.

Egyszerű példából kiindulva:
```
var car = {type:"Fiat", model:"500", color:"white"};
```

Itt a car az **objektum** neve és a benne lévő **tulajdonságok** illetve ezek **értékeik** láthatóak.
Ezeket külön külön is **el tudjuk érni** az alábbi módon illetve így tudjuk őket **módosítani** is.

2 módon **érhetőek** el ezek az elemek:

objectName.propertyName	vagyis	car.type
objectName["propertyName"]	vagyis	car[“type”]

Következő példában láthatjuk ahogy egy kód blokkot is megadhatunk objektum elemnek.

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Itt a **fullName** mint kód blokk fog szerepelni ami a benne lévő kódot **lefuttatja** és annak az **értékével tér vissza**.
Következő módon hívható meg : person.fullname() , illetve ha paraméter listás lenne a function akkor a ( ) között azok is megadhatóak lennének.

Az előző esetre nézve:
Var name = person.fullname() kódra a name értéke a “John Doe” szöveg lenne, mivel a kód blokk a this.firstName és a this.lastNamet **adja össze szövegesen**.

Fontos még a This szó mindig az adott objektumra vonatkozik így tudunk objektumon belül legegyszerűbben hivatkozni saját magára.

Aritmetikai Operátorok

Mivel **gyengén típusos a nyelv** így az algebrai operátorokat + - / * is **lazán** kezeli, igazából bármelyik változót bármelyik másik változóval összehozhatunk és kapunk egy eredményt, ami vagy az lesz amit szeretnénk vagy máshogy vonódik össze.

Áttekintés szintjén a legfontosabb operátorok :
Összeadás , - kivonás , / osztás , * szorzás , ++ érték növelés , -- érték csökkentés

```
var x = 1 + 2 + 3;
```

Legegyszerűbb példa a csak szám elemek összeadása ez esetben numerikusan 6 lesz az eredményünk.

```
var x = "5" + 2 + 5;
```

Pl ez a kód az x értékének a 525-et fogja adni, mivel a “5” karakterként szerepel innentől a 2 és az 5 hozzáadását is karakterként értelmezi.

var x = 2 + 5 + "5";

Viszont, ha így kezdjük,hogy az első értékadás szám akkor számként fog végrehajtódni a művelet így a 2+5 = 7 és mivel a “5” egy karakter így a 7 mögé fogja rakni így az eredmény 75 lesz.

var y = 2+ 2 +"5" + 6 + 3;

Ez egy **komplikáltabb** eset itt az elején a 2+2 az 4 lesz eddig számként kezeli a változót viszont utána a “5”-nél már szövegként így a +6 +3 is szövegként fog íródni a sorozat végére szóval az eredmény 4563 lesz.

var x = "John" + " " + "Doe";

Ez esetén az x értéke mivel az első elem is szöveg így egymás után írja és John Doe lesz az eredmény.

Ezekre az érdekes dolgokra fontos **odafigyelni**, mert más nyelvekkel ellentétben **nem figyelmeztet**,hogy nem egyforma adattípusokkal akarunk műveleteket végrehajtani. Illetve már a **kód elején érdemes létrehozni és adattal feltölteni minden változót** a könnyebb átláthatóság miatt.

Összehasonlító Operátorok

Ciklus vezérlések feltételének megadásához használjuk őket leginkább.

==	egyenlőség értékre
===	egyenlő érték és egyenlő adattípus
!=	nem egyenlő érték
!==	nem egyenlő érték vagy nem egyenlő adattípus
>	nagyobb
<	kisebb
>=	nagyobb egyenlő
<=	kisebb egyenlő

Logikai operátorok

Az előző operátorokat lehet felfűzni a ciklus feltétel vizsgálatának kibővítéséhez.

&&	és
||	vagy
!	logikai tagadás

Példa:
```
var x = 6;
var y = 3;
document.getElementById("demo").innerHTML =
(x < 10 && y > 1) + "<br>" +
(x < 10 && y < 1  + "<br>"
+ (x < 10 || y < 1 );
```

Ez a kód **visszatérési értéke** az első sorban igaz lesz mivel a x kissebb mint 10 és y nagyobb mint 1 így kiírja,hogy true azaz igaz, a második viszont hamis lesz mivel y nem kissebb mint 1, az utolsó sor értéke viszont újból true mivel a && ÉS operátort kicseréltem || vagy operátorra így elég ha csak az egyik vizsgált feltétel teljesül ami esetünkben a x<10 true értéke elég lesz ahhoz,hogy a vizsgálat true értékkel térjen vissza.

Event

A leggyakoribb legtöbbször használt eventeket nézzük át következőleg, ezek adott események észlelésére illetve azokra történő reakciókra valók, így ha valami történik a weblapon vagy a futás során akkor arra tudunk egy úgymond válasz reakciót adni.

onclick Event
```
<button onclick="myFunction()">Click me</button>
```
Adott gombra vagy bármilyen html elemre rákötve lefuttathatunk egy adott kód blokkot

```
<body>

<p id="demo" onclick="myFunction()">Click me to change my text color.</p>

<script>
function myFunction() {
  document.getElementById("demo").style.color = "red";
}
</script>

</body>
```

Adott kód például a szövegre kattintás után **kicseréli** annak a színét pirosra.

ondblclick Event
Double-click me
Hasonló az előzőhöz csak a dupla kattintásra reagálva fog **lefuttatni** egy kód blokkot.

ondrag Event
Drag me!
Adott html elemre engedélyezi az **egérrel húzást** és a húzás közben megtudunk hívni egy kód blokkot. Erre nézünk majd példát az ötödik olvasó leckében részletesebben is.

ondrop Event

Az előzőhöz csatlakozó event, ez akkor fut le amikor a **draggelt elemet elengedjük** egy droppolható elem felett. Erre is nézünk majd bővebb példát,hogy lehet őket kombinálni.

onfocus Event
<input type="text" onfocus="myFunction()">
Akkor hajtódik végre ha egy html elemre rákattintunk vagy a tab billentyűval belelépünk ez lehet akár gomb vagy beviteli mező is .

```
<!DOCTYPE html>
<html>
<body>

Írd be a neved: <input type="text" id="myInput" onfocus="focusFunction()"
onblur="blurFunction()">

<script>
function focusFunction() {
  // Focus = Changes the background color of input to yellow
  document.getElementById("myInput").style.background = "yellow";
}

function blurFunction() {
  // No focus = Changes the background color of input to red
  document.getElementById("myInput").style.background = "red";
}
</script>

</body>
</html>
```

Írd be a neved:

Adott kód például ha belekattintunk a beviteli mezőbe sárgára váltja a beviteli mezőt ha kikattintunk belőle akkor pedig pirosra.

oninput Event

<input type="text" oninput="myFunction()">

Beviteli mezőkre rakható rá, ha elkezdünk begépelni valamit az adott mezőben, akkor fog végrehajtódni.

```
<!DOCTYPE html>
<html>
<body>

<input type="range" oninput="myFunction(this.value)">
<p id="demo"></p>

<script>
function myFunction(val) {
  document.getElementById("demo").innerHTML = val;
}
</script>

</body>
</html>
```

58

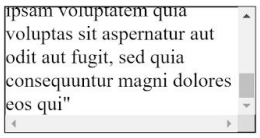
Adott példánál a beviteli mező egy slider és ahogy megmozdítjuk (beviszünk egy új értéket) rögtön frissíti a lent megjelent számot ami a slider értéke.

onscroll Event

Div-re rákötve az egér görgő mozgásának hatására fut le. Például 50 sor görgetés után történik valami.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  width: 200px;
  height: 100px;
  overflow: scroll;
}
</style>
</head>
<body>
<div onscroll="myFunction()">"Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo
inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim
ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur
magni dolores eos qui"
</div>
<p>Scrolled <span id="demo">0</span> times.</p>
<script>
var x = 0;
function myFunction() {
  document.getElementById("demo").innerHTML = x += 1;
}
</script>

</body>
</html>
```



Scrolled 12 times.

Adott div boxon belül számolja,hogy mennyit görgettünk le illetve fel.

MouseEvent clientX / clientY Property

Az egérmutató pozícióját adja meg a képernyő

```html
<!DOCTYPE html>
<html>
<body>

<h2 onclick="showCoords(event)">Erre a szövegre kattintva megmutatja,hogy hol
kattintottál pontosan rá az egérrel.</h2>


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

<script>
function showCoords(event) {
  var x = event.clientX;
  var y = event.clientY;
  var coords = "X coords: " + x + ", Y coords: " + y;
  document.getElementById("demo").innerHTML = coords;
}
</script>

</body>
</html>
```

Erre a szövegre kattintva megmutatja,hogy hol kattintottál pontosan rá az egérrel.

X coords: 140, Y coords: 35

Ellenőrző kérdések

Mi a változó létrehozásnak a kulcsszava?
Milyen karaktert kell használnod ahhoz,hogy a változóhoz adat kötéskor a változó adat típusa szöveges legyen?
Javascriptben,hogy hivatkozhatom az i tömb első elemét?
!== összehasonlító operátor mit jelent?
Mi a különbség a logikai || , && között?

Válaszok

Mi a változó létrehozásnak a kulcsszava?
Var

Milyen karaktert kell használni ahhoz,hogy a változóhoz adat kötéskor a változó adat típusa szöveges legyen?
“xyz” vagy ‘xyz’

Javascriptben,hogy hivatkozhatom az i tömb első elemét?
Javascriptben a tömb indexelés 0-tól indul így i[0]

!== összehasonlító operátor mit jelent?
Adattípus szinten nem egyenlő vagy nem egyenlő érték

Mi a különbség a logikai || , && között?
Az egyik a logikai ÉS a másik a logikai vagy az egyik megköveteli minden feltétel igaz vagy hamis értékét egyformán a másik pedig legalább 1 igaz feltétel esetén is már igazzal tér vissza.