Javascript összefoglaló

Mi is a Javascript?

A JAVASCRIPT NEM JAVA!
A Javascript programozási nyelv egy objektumalapú szkript nyelv, amelyet weblapokon elterjedten használnak. Eredetileg Brendan Eich, a Netscape Communications mérnöke fejlesztette ki; neve először Mocha, majd LiveScript volt, később „Javascript" nevet kapott, és szintaxisa közelebb került a Sun Microsystems Java programozási nyelvéhez. A Javascriptet először 1997–99 között szabványosította az ECMA „ECMAScript" néven. A legújabb szabvány az ECMA-262 Edition 5. Ez a szabvány egyben ISO szabvány is. FONTOS már az elején megjegyezni, hogy a Javascript nem nevezhető igazán objektumorientált nyelvnek. Javascript-ben nincsenek osztályok. A primitív típusok kivételével MINDEN objektum! Még a függvények is!

Mire használhatjuk a Javascript-et?

1. Üzenetablakok segítségével kommunikálhatunk a felhasználóval.
2. Mozgóképeket készíthetünk.
3. Ellenőrizhetjük, hogy az űrlapok tartalma helyesen van-e kitöltve.
4. A teljes oldal frissítése nélkül tudunk változásokat eszközölni az oldalon.
5. Kezelni tudjuk a felhasználói eseményeket.
6. Le tudjuk ellenőrizni a böngésző plugin-jait...
7. ... és még NAGYON sok mindenre használhatjuk.

Hogyan tudjuk futtatni a Javascript kódot?

A Javascript egy interpreteres nyelv, azaz nincs szükségünk fordítóra. A böngésző értelmezi és futtatja a Javascript kódot. A Javascript kód sorról sorra egymás után fut le. Érdemes tudni, hogy mindig a HTML rész töltődik be először. Egy honlapon minden src tulajdonsággal rendelkező tag egy-egy úgynevezett GET kérést eredményez a szerver felé, hiszen külső állományokat kell elkérni. Ezek az állományok sorba, egymás után szekvenciálisan töltődnek be (létezik parallel betöltés is).

Hogyan kapcsolhatjuk hozzá a honlapunkhoz a Javascriptet?

1. Beágyazhatjuk az oldalba:

```
<script type="text/javascript">
   javascript kód
</script>
```

A fenti helyett más megadási módok is léteznek:

```
<script type="application/javascript">
   javascript kód
</script>
```

Megadhatjuk a verziószámot is. Pl.: 1.7, vagy 1.8.1, vagy 1.8.5

```
<script type="application/javascript;version=1.7">
   javascript kód
</script>
```

Az E4X XML kezelési (és egyeb) lehetőséggel egészíti ki a Javascript nyelvet.

```
<script type="text/javascript; e4x=1">
   javascript kód
</script>
```

"Régimódi" megadási lehetőség verziószámmal, vagy anélkül.

```
<script language="Javascript1.2">
   javascript kód
```

```
</script>
```

A beágyazás történhet:

A <head>-be (pl.: függvények esetében)

```
<head>
   <script type="text/javascript">
      javascript kód
</script>
</head>
```

A <body>-ba

```
<body>
   <script type="text/javascript">
      javascript kód
</script>
</body>
```

2. Írhatjuk a kódot külső file-be.

```
<script type="text/javascript" src="kulsoFile.js"> </script>
```

3. Van amikor elhagyhatjuk a *script* tag-et.

```
<img src="kep.jpg" alt="kep" onmouseover="myFunction();">
```

Megjegyzés: Létezik a Javascript-nek egy úgynevezett strict módja is. Amennyiben ezt szeretnénk használni, a script-ünk első sorát a "use strict" utasítással kell kezdenünk. Amennyiben csak függvény szinten szeretnénk a strict módot használni, akkor elég a függvény első sorába beírni a "use strict" sort.

Amit a strict mód nyújt nekünk:

- "érzékenyebb" hibajelzés
- teljesítményjavulás

Változók

A programozási nyelvekben a változó úgy képzelhető el, mint egy hely, ami alkalmas egy érték tárolására a számítógép memóriájában.

Változó létrehozása:

Egy változót a var kulcsszóval deklarálunk, azaz hozunk létre.
Ezután megadjuk a változó nevét, majd egy egyenlőségjel után az értékét.
Megjegyzés: A változó neve kis- és nagybetűket, számot, és aláhúzást tartalmazhat csak, valamint nem kezdődhet számmal.

Pl.:

```
var elsoValtozo = 111;
```

Természetesen a változó értékének megadása nem kötelező. Ilyen esetben a változó értéke definiálatlan lesz (undefined).

Pl.:

```
var msodikValtozo;
```

Változókat egymás után is létrehozhatunk, vesszővel elválasztva. Ilyenkor ha akarjuk definiáljuk őket, ha nem, akkor nem.

Pl.:

```
var a, b, c, d =10, e='almafa';
```

Megjegyzés: Vannak bizonyos szavak, melyeket nem adhatunk meg változó- vagy függvénynévnek. Ezeket hívjuk foglalt szavaknak. Ezek a következők:

abstract	boolean	break	byte

case	catch	char	class
const	continue	debugger	default
delete	do	double	else
enum	export	extends	false
final	finally	float	for
function	goto	if	implements
import	in	instanceof	int
interface	long	native	new
null	package	private	protected
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	

Megjegyzés: Ezeken a szavakon kívül az egyes objektumok, azok tulajdonságainak és metódusainak neveit sem ajánlott használni!

Nézzük meg, hogy a Javascript milyen típusú változókat tud kezelni.

Elsőként az úgynevezett primitív típusokat nézzük meg. Ezek NEM objektumok. Javascriptben a következő 5 primitív, avagy elemi típusok állnak rendelkezésünkre:

- number
- string
- boolean
- undefined
- null

Ezeken kívül van még 2 fontos adattípus:

- object
- function

Valamint még 2 speciális típus, melyet a typeof visszaadhat:

- xml
- implementation-dependent

Megjegyzés: A number, string, és boolean primitív adattípusoknak VAN objektum megfelelője!

Nézzük meg sorba az adattípusokat

1. Number

A Javascript szám típusa. Számok tárolására használjuk. Mind egész, mind pedig lebegőpontos számok tárolására alkalmas. A következő módokon adhatunk meg számokat:

- Egész számok:
  - 120
  - -998
- Lebegőpontos számok:
  - 987.112
  - 123.1234
  - 223E10
  - 2E-12
  - 232.122E-23

Megjegyzés: Az *E* (lehet *e* is) az exponenst, azaz 10 hatványát jelenti!
A tört számoknál nem vesszőt hanem pontot használunk! A Javascript alaphelyzetben nem csak a 10-es, hanem a 8-as (oktális) és a 16-os (hexadecimális) számrendszert is ismeri. A számokat megadhatjuk ezekben a számrendszerekben is a következőképp:

- Oktális: 012 (egyenlő 10-el)
- Hexadecimális 0x11 (egyenlő 17-el)

Megjegyzés: Amennyiben azt szeretnénk, hogy egy szám 8-as számrendszerbe legyen, úgy 0-val kell kezdenünk a szám írását!
Amennyiben azt szeretnénk, hogy egy szám 16-os számrendszerbe legyen, úgy 0x-el kell kezdenünk!
Az egész számok 4, a lebegőpontos számok 8 byte-on tárolódnak!

2. String

A String a Javascript szöveg típusa. Szöveg tárolására használjuk.
Egy string a karakterek számától függő méretű helyet foglal a memóriában.
A stringet aposztróf vagy idézőjelek közé tesszük.

Pl.:

```
var firstString = 'ez egy szöveg';
var secondString = "ez is egy szöveg";
```

Ha a szövegben szeretnénk idézőjelet vagy aposztrófot megjeleníteni, azt a következőképp tehetjük

Pl.:

```
var firstString = "'ez egy szöveg aposztrófok között'";
var secondString = '"ez egy szöveg idézőjelek között"';
var thirdString = '\'ez is egy szöveg aposztrófok között\'';
var forthString = "\"ez is egy szöveg idézőjelek között\"";
```

Tehát vagy keverjük az aposztrófos és idézőjeles jelölést, vagy escape-eljük az adott speciális jelentéssel bíró karaktereket. Ha visszapert írunk az aposztróf, vagy idézőjel elé, azzal feloldjuk a speciális jelentését.

Az összes string literálban előforduló speciális karakter:

'	védett karakter
"	védett karakter
\	védett karakter
\b	törlés
\t	tabulátor
\n	új sor
\r	kocsi vissza
\f	lapdobás

Amennyiben ezeket a speciális karaktereket szeretnénk szimpla szövegként megjeleníteni, úgy egy \ karaktert kell eléjük tennünk.

3. Boolean

A Boolean a Javascript logikai típusa. Csak két értéket vehet fel:

- true
- false

Boolean-nál az értéket NEM kell idézőjelek közé tenni.

4. Undefined

Azaz definiálatlan. Ha egy változót deklarálunk, azaz létrehozzuk, de nem definiáljuk, azaz nem adunk neki értéket, akkor undefined lesz.

5. Null

A null egy speciális érték, amely azt jelzi, hogy az adott változó üres. Eleinte zavaros, hogy mi a különbség az undefined és a null között, de a későbbiekben majd nézünk rá példát, és mindenki leszűrheti a konklúziót. Ha valaminek null értéket szeretnénk adni, szintén nem kell idézőjelek közé tenni.

Ha különböző típusú adatokon végzünk aritmetikai műveleteket, akkor a Javascript automatikusan elvégzi a típuskonverziót a következők szerint:

Típuskonverzió

String + Number = String

String + Boolean = String

Boolean + Number = Number

Number - String = Number

Number - Boolean = Number

Operátorok

Összehasonlító operátorok

==	Egyenlőség vizsgálata
!=	Egyenlőtlenség vizsgálata
===	érték- és típusegyenlőség vizsgálata
!==	Típusegyenlőtlenség vizsgálata
>	Nagyobb-e
>=	Nagyobb egyenlő-e
<	Kisebb-e
<=	Kisebb egyenlő-e

Aritmetikai operátorok

+	összeadás
-	Kivonás
*	Szorzás
/	Osztás
%	Maradékos osztás
++	Incrementálás
--	Decrementálás

értékadó operátorok

=	Legyen egyenlő
+=	összevont operátor: a = a + b helyett a += b
-=	összevont operátor: a = a - b helyett a -= b

*= összevont operátor: a = a * b helyett a *= b

/= összevont operátor: a = a / b helyett a /= b

%= összevont operátor: a = a % b helyett a %= b

<<= összevont operátor: a = a << b helyett a <<= b

>>= összevont operátor: a = a ;>> b helyett a >>>= b

>>>= összevont operátor: a = a >>> b helyett a >>>= b

&= összevont operátor: a = a & b helyett a &= b

|= összevont operátor: a = a | b helyett a |= b

^= összevont operátor: a = a ^ b helyett a^= b

[]= összevont operátor: a = b, b = d helyett [a,b]=[c,d]

Logikai operátorok

&& Logikai éS

|| Logikai VAGY

! Tagadás

Bitszintű operátorok

& AND (éS)

| OR (VAGY)

^ XOR (Kizáró VAGY)

~ NOT (Tagadás)

<< Bitszintű balra eltolás

>> Bitszintű jobbra eltolás

>>> Előjel nélküli jobbra eltolás

Különleges operátorok

.	Objektum tulajdonságának, metódusának elérése
[]	Objektum tulajdonságainak "tömbszerű" elérése
delete	elemek törlése
in	Objektum bejárása
instanceof	Példánya-e adott objektumnak
new	új objektum létrehozása
this	Aktuális objektum elérése
?:	Feltételes értékadás
typeof	Típus lekérdezése
void	Visszatérési érték kiküszöbölése

Operátorok precedencia szintjei

1 ()

	[]
	function()
2	!
	~
	-
	++
	--
	new
	typeof
	void
	delete
3	*
	/
	%
4	+
	-
5	<<
	>>>
	>>
6	<
	<=
	>
	>=
7	==
	!=
8	&
9	^
10	|
11	&&
12	||
13	?
14	=
	+=
	-=
	*=
	/=
	%=
	<<=
	>>>=
	>>=
	&=
	^ =
	|=
15	,

Vezérlési szerkezetek

```
//Feltételes szerkezet
if (feltétel) {
   utasításokHaIgaz;
```

```
}

//Feltételes szerkezet else ággal
if (feltétel) {
   utasításokHaIgaz;
}
else {
   utasításokHaHamis;
}

//feltételes szerkezet else if ággal
if (feltétel) {
   utasításokHaIgaz;
}
else if{
   utasításokHaIgaz;
}
...
else {
   utasításokHaHamis;
}

//Feltételes értékadás
result = feltétel ? érték1 : érték2;

//Számlálós ciklus
for ([ciklusváltozó kezdőértéke]; [feltétel]; [lépésköz]) {
   utasítások;
}

//For in ciklus objektumok bejárására
for (var in object) {
   utasítások;
}

//For each ciklus tömbök bejárására
for each ([var] változóNeve in objektumReferencia) {
   utasítások;
}

//Objektumok tulajdonságainak elérése
with (objektumReferencia) {
   utasítások;
}

//Hátultesztelős ciklus
do {
   utasítások;
} while (feltétel)

//Elöltesztelős ciklus
while (feltétel) {
   utasítások;
}

//Visszatérési érték
return [érték];

//Többirányú elágazásos szerkezet
switch (kifejezés) {
   case értékN :
      utasítások;
      [break];
...
```

```
   [default :
      utasítások];
}

//Cimkék létrehozása vezérlés átadásához
label :
continue [cimke];
break [cimke];

//Kivételkezelés OO módra
try {
   utasítások ahol hiba lehet;
}
catch (hibaInformáció) {
   utasítások melyek lefutnak ha hiba volt a try blokkban;
}
[finally {
   utasítások melyek mindenképp lefutnak;
}]

//Kivétel kiváltása
throw érték;

//Függvény szintaxisa
function függvényNeve ([paraméter1], [parametér2], ... [paraméterN]){
   utasítások;
   [return érték];
}
```

Események

Esemény	Hol használhatjuk?
abort	object
blur	window, button, text, password, label, select, textarea
change	text, password, textarea, select
click	Minden elemre
error	window, frameset, object
focus	window, button, text, password, label, select, textarea
keydown	text, password, textarea
keypress	text, password, textarea
keyup	text, password, textarea
load	window, frameset, object
mousedown	Minden elemre
mousemove	Minden elemre
mouseout	Minden elemre
mouseover	Minden elemre
mouseup	Minden elemre
reset	form
resize	window

scroll	window
select	text, password, textarea
submit	form
unload	window, frameset

Array objektum

Tulajdonságok	Metódusok
constructor	concat(tömb2)
length	every(függvény[, objektum])
prototype	filter(függvény[, objektum])
	forEach(függvény[, objektum])
	indexOf(függvény[, objektum])
	join("karakter")
	lastIndexOf(függvény[, objektum])
	map(függvény[, objektum])
	pop()
	push()
	reduce()
	reduceRight()
	reverse()
	shift()
	slice(i,[j])
	some(függvény[, objektum])
	sort(összehasonlító függvény)
	splice(i,j[,értékek])
	toLocaleString()
	toString()
	unshift()