WEB PROGRAMOZÁS II.
VEMIVIB233WF

1. Hét – ismétlés

Pannon Egyetem
Műszaki Informatikai Kar
Villamosmérnöki és Információs Rendszerek Tanszék

Webprogramozás II. oktatói és elérhetőségeik

- **Dr. Szücs Veronika** egyetemi docens, tárgyfelelős

  szucs.veronika@mik.uni-pannon.hu
- **Nagy Zsuzsanna** tanársegéd, gyakorlatvezető nagy.zsuzsanna@mik.uni-pannon.hu

Tantárgy tematikája

Haladó JavaScript programozás :

- JavaScript objektumai (Array, String, Date, Math, Function, Number stb.)
- JavaScript és az AJAX technológia
- XML és JSON adatleírók
- Adatok kezelése AJAX technológiával
- Szerver oldali JavaScript futtató környezet kialakítása (npm, NodeJS)
- Szerver oldali JavaScript keretrendszerek megismerése (ExpressJS, EmberJS stb.)
- MVC tervezési szemlélet alapú Kliens-szerver alkalmazás fejlesztés JavaScript környezetben
- Beszámolók

JavaScript alapok ismétlése (WP I. tematikából)

- Form (űrlap) létrehozása
- Űrlap elemek és attribútumaik
- Űrlap elemeinek kezelése JavaScript függvényekkel

- **JS ismétlés (diasorból):**
  - **Számok kezelése, hibaforrások (!!!)**
  - **Események kezelése**
- **SEGÉDLET: előző félévben összeállított elem-tulajdonság-példa táblázatuk!**

JavaScript számok, Number objektum

```
var x = 3.14;     // tizedes szám
var y = 34;       // szám tizedesek nélkül

var x = 123e5;    // Nagy számok, tudományos
alakban is felírhatók
var y = 123e-5;   // 0.00123
```

Ellentétben sok más programozási nyelvvel, **a JavaScript nem határoz meg különböző típusú számokat**, például egész számokat, rövid, hosszú, lebegőpontos stb.

A JavaScript számok mindig 64 bites lebegőpontos számok

A JavaScript számok mindig 64 bites lebegőpontos számok

Ebben a formátum tárolja a számokat 64 biten, ahol a szám (a mantissa) a 0-tól 51-ig terjedő biteken van tárolva, az exponens az 52-től 62-ig terjedő biteken, és a 63-as bit az előjel:

Value (aka Fraction/Mantissa)	Exponent	Sign
52 bits (0 - 51)	11 bits (52 - 62)	1 bit (63)

Pontosság:

Egész számok (ismétlődés vagy exponens jelölés nélkül) **legfeljebb 15 jegyig pontosak**.

```
var x = 999999999999999;   // az x  999999999999999
var y = 9999999999999999;  // az y 10000000000000000
```

A tizedesjegyek maximális száma 17, de a lebegőpontos aritmetika nem mindig 100% -os pontosságú.

A probléma megoldását segíti, ha osztás és szorzás műveletek során egészekkel számolunk:

```
var x = (0.2 * 10 + 0.1 * 10) / 10;       // az x  0.3 lesz
```

Műveletek: összeadás

A JavaScript a + operátort használja mind az összeadáshoz, mind összefűzéshez (stringeknél).

- **Számokat összeadja**
- **Sztringeket összefűzi**

```
var x = 10;
var y = 20;
var z = x + y;   // z 30 lesz (szám)
```

```
var x = 10;
var y = "20";
var z = x + y;          // z 1020 ( string)
```

```
var x = „10”;
var y = „20”;
var z = x + y;   // z 1020 lesz (string)
```

Tipikus hibák

Egy általános hiba, ha azt várjuk, hogy ez az eredmény 30 lesz:

```
var x = 10;
var y = 20;
var z = "The result is: " + x + y;
```

Egy általános hiba, ha azt várjuk, hogy ez az eredmény 102030 lesz:

```
var x = 10;
var y = 20;
var z = "30";
var result = x + y + z;
```

Fontos észrevétel: a JS ÉRTELMEZŐ balról jobbra olvas és dolgozik!

Numeric Strings

A JavaScript sztringeknek lehet numerikus tartalma:

```
var x = 100;        // x egy szám
var y = "100";      // y egy string
```

A JavaScript megpróbálja a karakterláncokat numerikus értékké alakítani a numerikus műveletekben alakítani:

Ez működni fog:

```
var x = "100";
var y = "10";
var z = x / y;      // z 10 lesz
```

```
var x = "100";
var y = "10";
var z = x * y;      // z 1000 lesz
```

```
var x = "100";
var y = "10";
var z = x - y;      // z =90
```

Ez nem fog működni:

```
var x = "100";
var y = "10";
var z = x + y;      // z nem 110 (hanem 10010)
```

NaN – Not a Number

A **NaN** a JavaScript-ben egy fenntartott kifejezés, amely azt jelzi, hogy egy szám nem valódi szám.
Ha nem numerikus karakterlánccal szeretnénk számtani műveletet végezni, akkor **NaN** (Nem szám) jelenik meg:

```
var x = 100 / „Tizenkettő";  // x NaN (Not a Number) lesz
```

Ha azonban a karakterlánc tartalmaz egy számértéket, az eredmény egy szám lesz:

```
var x = 100 / "10";     // x 10 lesz
```

Számrendszerek közötti konverzió

Alapértelmezés szerint a JavaScript a számokat 10-es számrendszerben jeleníti meg (decimális számok).

A ***toString ()*** metódus segítségével a kimenet 16-os (hexadecimális), 8 (oktális) vagy 2-es (bináris) számrendszerbeli értékek is lehetnek.

```
var myNumber = 128;
myNumber.toString(16);  // kimenet: 80
myNumber.toString(8);   // kimenet: 200
myNumber.toString(2);   // kimenet: 10000000
```

Soha ne kezdjük a számot nullával (például 07).
Néhány JavaScript verzió a számokat oktálisként értelmezi, ha azok nullával kezdődnek.

toExponential() metódus

Meghívásakor stringet ad vissza!

```
var x = 9.656;
x.toExponential(2);    // kimenet 9.66e+0
x.toExponential(4);    // kimenet 9.6560e+0
x.toExponential(6);    // kimenet 9.656000e+0
```

toFixed() metódus

Meghívásakor stringet ad vissza!

```
var x = 9.656;
x.toFixed(0);          // kimenet 10
x.toFixed(2);        // kimenet 9.66
x.toFixed(4);     // kimenet 9.6560
x.toFixed(6); // kimenet 9.656000
```

Paraméterben megadott pontossággal, tizedes jegy pontossággal adja vissza az eredményt

toPrecision() metódus

Meghívásakor stringet ad vissza!

```
var x = 9.656;
x.toPrecision();      // returns 9.656
x.toPrecision(2);     // returns 9.7
x.toPrecision(4);     // returns 9.656
x.toPrecision(6);     // returns 9.65600
```

A paraméterben megadott számjegynyi számmal adja vissza az eredményt. Ha szükséges, 0-kal feltölti, kiegészíti a sztringet.

parseInt() függvény

A ***parseInt ()*** egy stringet dolgoz fel és egy egész számot ad vissza. White-space-ek megengedettek. Csak az első számot adja vissza:

```
parseInt("10");         // kimenet 10
parseInt("10.33");      // kimenet 10
parseInt("10 20 30");   // kimenet 10
parseInt("10 years");   // kimenet 10
parseInt("years 10");   // kimenet NaN
```

parseFloat() hasonlóan működik, de lebegőpontos számot ad vissza.

```
parseFloat("10.33");    // kimenet 10.33
```

JavaScript függvények

A függvények sajátos kód blokkok, speciális feladatra megírva

Akkor futnak le, amikor valamelyik másik kódrészlet (eseménykezelő, értékadás stb.) meghívja a függvényt

Pl:

```
function myFunction(p1, p2) {
    return p1 * p2;              // A függvény a p1 és p2 szorzatát adja vissza
}
var x = myFunction(4, 3);  //meghívja a függvényt, és x értéke a függvény által visszaadott érték lesz
```

HTML események kezelése – event-ek

- A HTML elemekkel „történik” valami, a JS reagál erre
- Pl.:

<button **onclick**="document.getElementById('demo').innerHTML = Date()">The time is?</button>

<button **onclick**="this.innerHTML = Date()">The time is?</button>

<button **onclick**="displayDate()">The time is?</button>

- Teljes HTML DOM esemény lista:
- https://www.w3schools.com/jsref/dom_obj_event.asp

Esemény	Melyik elemre alkalmazható?	Mikor következik be?	Eseménykezelő neve
abort	képek	A felhasználó megszakítja a kép betöltését	onAbort
blur	ablak, keret, és minden űrlapmezŐ	Az inputfókusz elkerül az elemrŐl (pl. az egérrel az aktuális mezŐn kívülre kattintunk)	onBlur
change	szövegmező, listaablak	Megváltoztatunk egy őrlapbeli értéket	onChange
click	gombok, rádió gomb, csatolások (linkek)	Az űrlap valamely elemére, vagy egy csatolásra (link) kattintunk	onClick
error	ablak, képek	Ha kép vagy dokumentum betöltésekor hiba lép fel	onError
focus	ablak, keret, minden űrlapmező	Az inputfókusz az adott elemhez kerül (pl. kijelöljük az űrlap egy elemét)	onFocus
load	dokumentum törzse (BODY)	Ha az oldal betöltése befejeződött	onLoad
mouseout	linkek, képek	Ha az egérmutató elhagyja a linket vagy a kép területét	onMouseOut
mouseover	linkek, képek	Ha az egérmutató a link felett tartózkodik vagy a kép területén	onMouseOver
reset	űrlapokra	űrlap törlésekor	onReset
select	szövegmező	Új mezőt jelölünk ki	onSelect
submit	submit típusú nyomógomb	űrlap elküldésekor	onSubmit
unload	dokumentum törzse (BODY)	Ha elhagyjuk az oldalt	onUnload

Gyakorló feladat 1.

Feladat:
Készítsen egy weboldalt (HTML, JS fájlok külön!)
A weboldalon kérjen be a felhasználótól egy értéket, egy egész számot, ami egy számokat tartalmazó tömbnek a mérete lesz.
Hozza létre a tömböt JS-ben
Töltse fel a tömböt egész értékekkel, amelyek 0-100 értékeket vehetik csak fel. Az értékeket a Math.random() metódus segítségével állítsa elő!
A tömbben számolja meg, hogy egy szám hányszor fordul elő. Csak azokat számolja meg, amelyek előfordulnak a tömbben. A tömbben nem szereplő számokra nem kell a 0 darab információ.
Írja ki a leggyakrabban előforduló számot, az első és az utolsó elfordulás pozícióját a tömbben!
Írja ki a legritkábban előforduló számot, és annak első és utolsó előfordulási helyét a tömbben!
A több szám is megfelel a leggyakoribb vagy legritkább előfordulás feltételnek, akkor tetszőlegesen döntse el, melyikre íratja ki a fenti információkat!

Gyakorlat feladat 2. (ismétlésként)

Készítsen egy hatvány táblát!
Készítsen táblázatot, mely megjeleníti 0-tól 10-ig a számok négyzetét, köbét, negyedik, és ötödik (tetszés szerint lehet magasabb is) hatványát.
A táblázat elkészítése során a számolást bízzuk JavaScript programra.

	1	2	3	4	5	6	7	8	9	10	11
0	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11
2	1	4	9	16	25	36	49	64	81	100	121
3	1	8	27	64	125	216	343	512	729	1 000	1 331
4	1	16	81	256	625	1 296	2 401	4 096	6 561	10 000	14 641
5	1	32	243	1 024	3 125	7 776	16 807	32 768	59 049	100 000	161 051
6	1	64	729	4 096	15 625	46 656	117 649	262 144	531 441	1 000 000	1 771 561
7	1	128	2 187	16 384	78 125	279 936	823 543	2 097 152	4 782 969	10 000 000	19 487 171
8	1	256	6 561	65 536	390 625	1 679 616	5 764 801	16 777 216	43 046 721	100 000 000	214 358 881
9	1	512	19 683	262 144	1 953 125	10 077 696	40 353 607	134 217 728	387 420 489	1 000 000 000	2 357 947 691
10	1	1 024	59 049	1 048 576	9 765 625	60 466 176	282 475 249	1 073 741 824	3 486 784 401	10 000 000 000	25 937 424 601

Ciklusok: Készítsen ciklust, ami a hiányos számsorokat hiánytalanul kiírja a bennük található számítási logika alapján a böngészőbe:

Számok 1-től 20-ig:1 2 3 4 … … … 18 19 20
Páros számok 2-től 30-ig:2 4 6 8 10 … … … 28 30
Számok 30-tól 100-ig, hetesével:30 37 44 51 … … … 93 100
Számok 112-től 2-ig, tizenegyesével lefelé:112 101 90 … … … 24 13 2
Számok -90-től 90-ig, tizenötösével:-90 -75 … -15 0 15 … 75 90
Azok a kétjegyű számok, amelyek számjegyeinek összege 10:19 28 37 46 55 … … … 91

További gyakorló feladatok ciklusokhoz

Pozitív egész számpárok, ahol a két szám összege 18:1-17 2-16 3-15 ... ... 8-10 9-9
A 8-as szorzótábla:1 * 8 = 8
2 * 8 = 16
3 * 8 = 24
4 * 8 = 32
5 * 8 = 40
6 * 8 = 48
7 * 8 = 56
8 * 8 = 64
9 * 8 = 72
10 * 8 = 809.
Az első 15 pozitív egész szám négyzete:1 4 9 16 25 ... 169 196 225
A elkövetkezendő nem szökőévek 2041-ig:2022 2023 2025 2026 ... 2039 2041
A 144 összes osztója:1 2 3 4 6 ... ... 72 144
A 2 hatványai:1 2 4 8 16 32 64 ... 1048576

További gyakorló feladatok ciklusokhoz

Mindig 1-gyel nagyobb különbség:1 2 4 7 11 16 22 29 … 301
Az előző két szám összege:1 2 3 5 8 13 21 34 … 4181
9 időpont óránként, reggel negyed 9-től:08:15 09:15 10:15 … … … 16:15
Időpontok 20 percenként, délelőtt:08:00 08:20 08:40 … … … 12:00
Napi menetrend, 50 percenként induló járatokkal:08:00
08:50
09:40
…
…
18:00

További gyakorló feladatok ciklusokhoz(JS+ciklusok)

Csengetési rend sorszámmal, 45 perces tanórákkal, 10 perces szünetekkel:
1. 08:30 - 09:15
2. 09:25 - 10:10
3. 10:20 - 11:05
...
...
8. 14:55 - 15:40

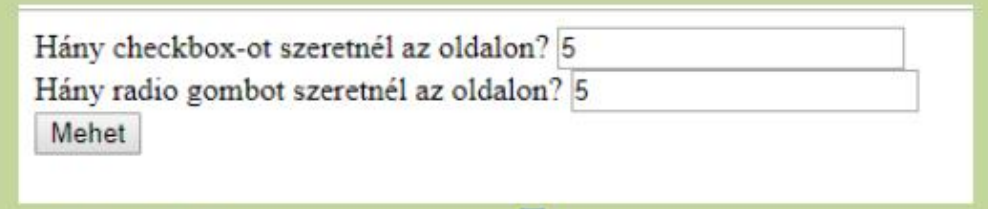
Feladat

Készítsen egy olyan weboldalt, amelyen megadható, hogy hány radio gombot és checkbox-ot szeretne létrehozni!

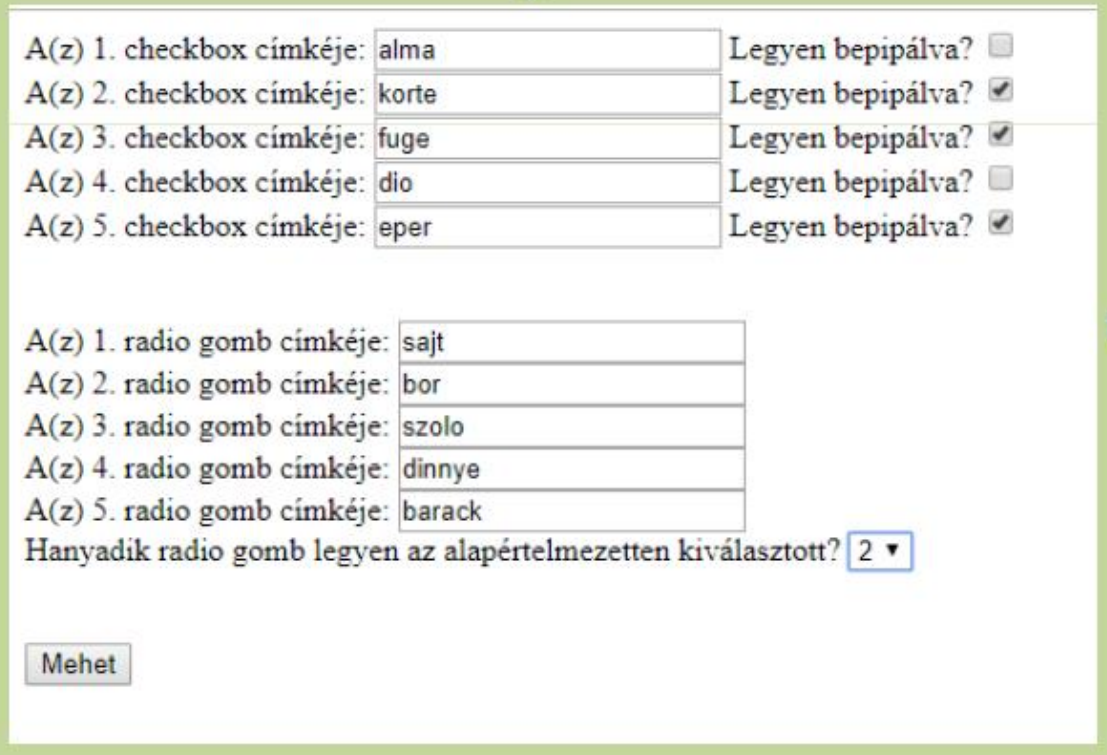
A megadott értékek alapján generáljon egy olyan oldalt, ahol a létrehozandó gomboknak címke adható, valamint beállítható az, hogy mely gombok legyenek alapértelmezetten bejelölve.

A második oldalon megadott tartalom alapján generálja le a 3. oldalt!

Kiinduló oldal



2. oldal



3. oldal