WEBFEJLESZTÉS 2. – JAVASCRIPT NYELVI ALAPOK

Horváth Győző

Egyetemi adjunktus

1117 Budapest,
Pázmány Péter sétány 1/C, 2.420
Tel: (1) 372-2500/1816

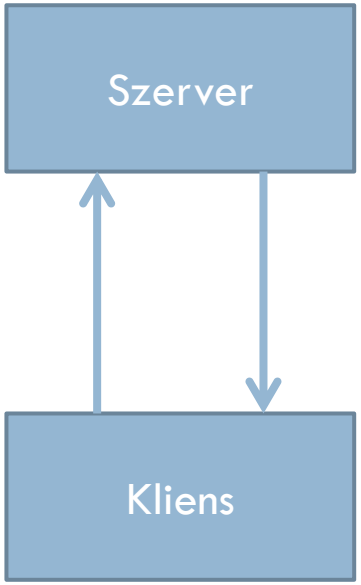
Tartalomjegyzék

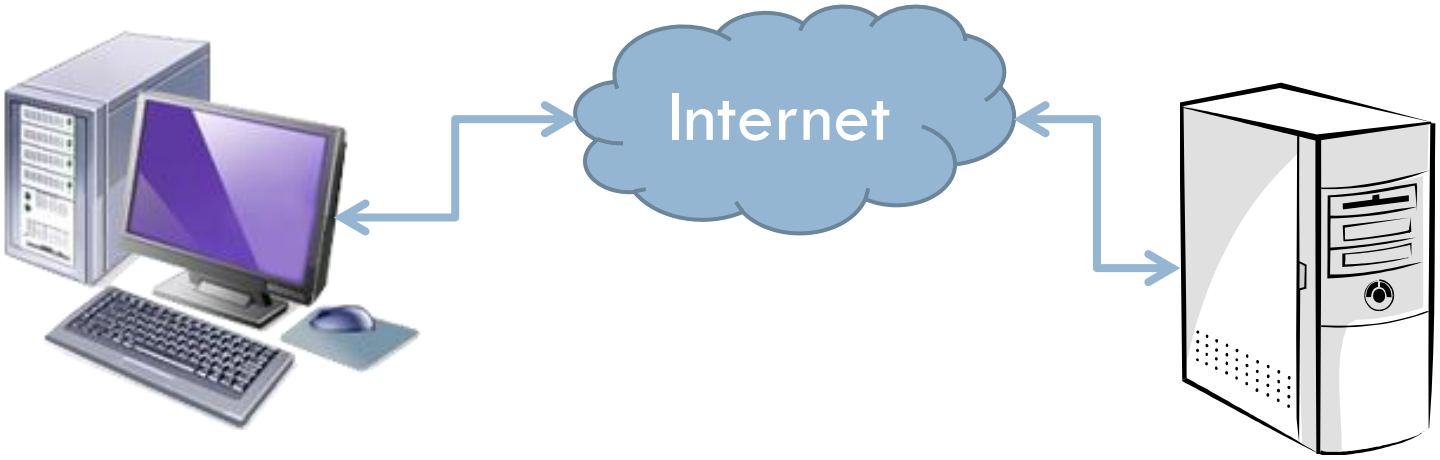
Dinamikus webprogramozás

Statikus vs dinamikus

Kliens-szerver architektúra

- Web: kliens és kiszolgáló kommunikációja
- HTTP: a kommunikáció protokollja
- Kliens kérést intéz a szervernek
- Szerver válaszol
- A kliens feldolgozza a választ





Statikus oldalak

- Szerver szempontjából statikus
  - Kérés pillanatában a szerveren megtalálható az a tartalom, amely leküldésre kerül a válaszban
  - Fájlkiszolgálás
- Kliens szempontjából statikus
  - A letöltött és a létrejött tartalom az oldal élettartamának a végéig ugyanaz
  - Nem változik meg sem a böngésző állapota, sem a betöltött dokumentum szerkezete
  - Nem fut le benne programkód, leíró nyelv, deklaratív

Dinamikus oldalak

- Szerver szempontjából dinamikus
  - A válaszként leküldött tartalmat program állítja elő
  - A kérés pillanatában a válasz még nem létezik a szerveren
- Kliens szempontjából dinamikus
  - A letöltött tartalomban programkód fut le
  - Ez megváltoztathatja a böngésző állapotát és a dokumentum szerkezetét
- → programozás

Félév felépítése

- Első felében a kliens oldali dinamikus webprogramozás alapjaival ismerkedünk meg
  - JavaScript
- Második felében a szerver oldali dinamikus webprogramozás alapjai következnek
  - PHP

Motiváció

Trendek

Trendek, jelek

- Egyre több alkalmazás a weben
- Egyre többféle alkalmazás a weben
- Webes ügyintézés, webshopok, felhő technológia
- Mobil platform
- HTML5: Windows 8, Gnome 3, Okostelefon

- → SZERVER
- → WEBES KLIENS

Motiváció

- Állásajánlatok – HTML5

JavaScript története

JavaScript születése, a böngészők háborúja

A kezdet

- 1991: világháló születése
- 1993: első grafikus böngészők
- 1994: Netscape Navigator böngésző
- 1995 áprilisa: Netscape cég felkéri Brendan Eich-et, hogy fektesse le egy olyan programozási nyelv alapjait, amelyekkel interaktívvá tehetők weboldalak
- Cél: Java pluginek elérése nem Javás programozóknak
- 1995 decembere: bejelentik a JavaScriptet

Név és szabvány

- Elnevezések
  - LiveScript
  - JavaScript: marketing miatt a Java programozók átcsábítására (web Visual Basic-je)
- Szabvány: ECMAScript
  - Európai Informatikai és Kommunikációs Rendszerek Szabványosítási Szövetsége (ECMA)
- Microsoft
  - JScript

Böngészők háborúja

- 1996-1999: Böngészők háborúja
  - Netscape Navigator vs. Microsoft Internet Explorer
  - Más-más irányba mennek, kölcsönösen nem fogadják el egymás javaslatait
  - Nyelv és böngészők gyors fejlődése, CSS
  - Szabványosítás elmaradt → inkompatibilitási problémákhoz vezet
- 1999: Sötét középkor
  - Szerveroldali technológiák fejlődése
- 2006: JavaScript újrafelfedezése (AJAX)

Fejlesztőeszközök

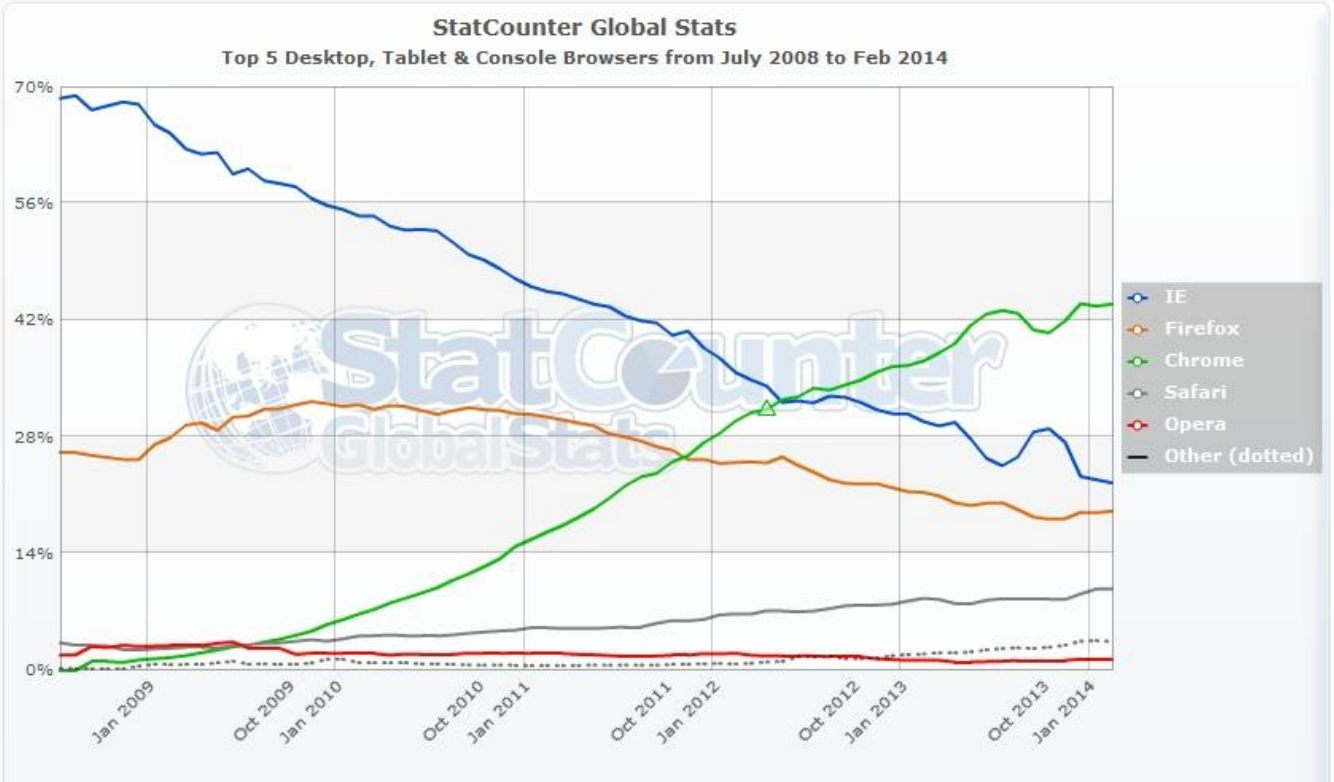
Szerkesztők, böngészők, eszköztárak, dokumentáció

Szerkesztők

- Tetszőleges text editor használható
- De azért jó, ha a szerkesztő segíti a fejlesztést
  - Szintaxis színezése, kiemelése
  - Kódkiegészítés, függvények ismerete
  - Nyomkövetés
- Megfelelő szerkesztők
  - Notepad++
  - Sublime Text 2
  - Netbeans IDE
  - WebMatrix 2

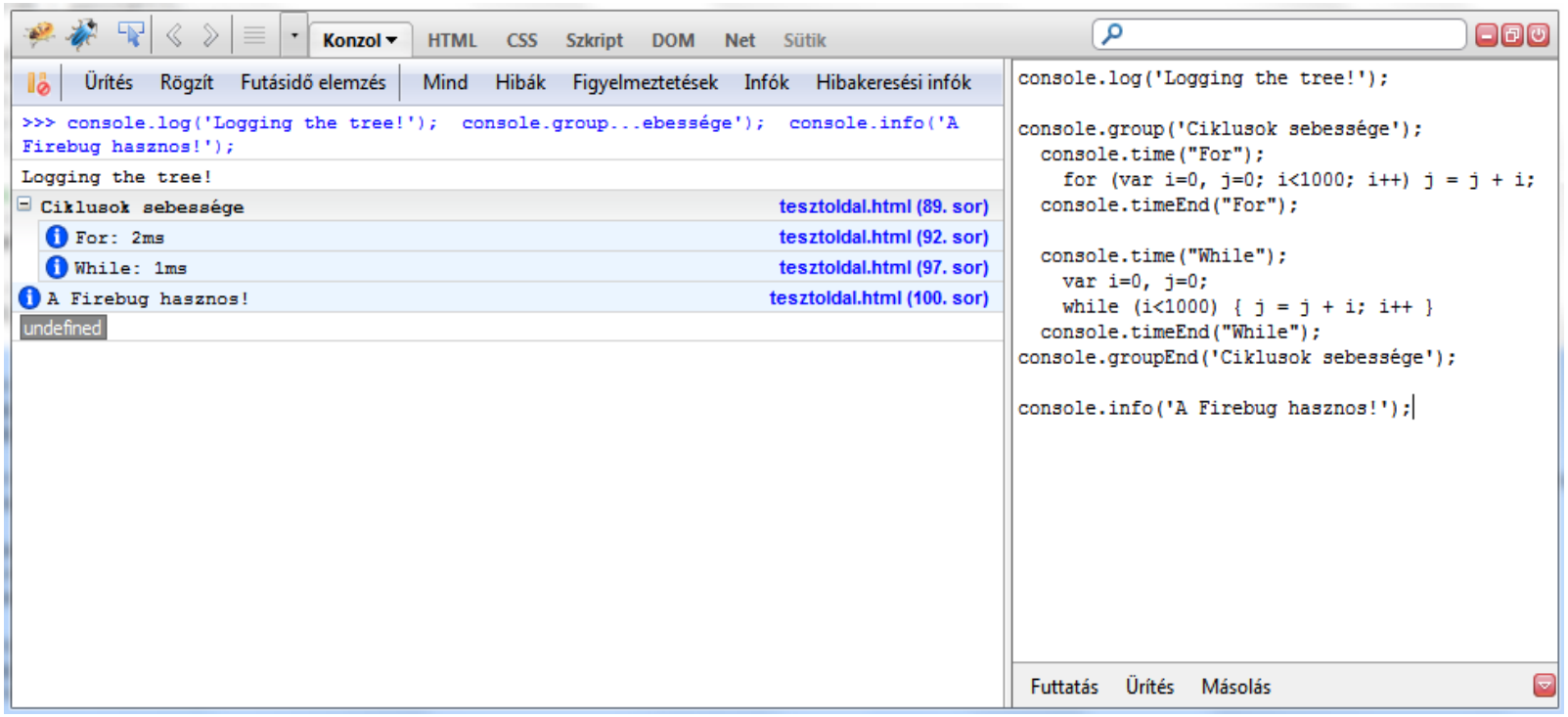
Böngészők

- Mozilla Firefox
- Google Chrome
- Internet Explorer
- Safari
- Opera



Webfejlesztési eszközök

- Webfejlesztő eszközök (FF) – sokféle eszköz
- JavaScript konzol
- console.log()



Dokumentáció

- Hivatalos dokumentáció az ECMAScript szabvány
- Mozilla Developer Network JavaScript része
  - főoldal
  - JavaScript referencia
  - JavaScript guide
- Ezekben sok Firefox specifikus dolog van, de ezt a dokumentáció megfelelően jelzi

Futtató környezet

Futási környezet

- A JavaScriptnek szüksége van egy futtató környezetre.
- Ez kliens oldalon maga a böngésző, amelyben fut.
- A böngészőben lévő értelmező értelmezi a HTML kódban elhelyezett JavaScript kódot.

- Parancssorban/szerveroldalon
  - Node.js

Hova írhatjuk a kódot?

- JavaScript konzolba
  - Az adott oldal kontextusában értelmezésre kerül
  - Kipróbálásra jó
- HTML kódba
  - Inline szkript, <script> tag, bárhova rakhatjuk
  - Külső állományba, <script> tag src attribútumával töltjük be

JavaScript kód helye


```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Webfejlesztés 2.</title>
    <script type="text/javascript">
      //JavaScript kód helye
    </script>
    <script type="text/javascript" src="jskod.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      //JavaScript kód helye
    </script>
    <p>Hello világ!</p>
    <script type="text/javascript" src="jskod2.js"></script>
  </body>
</html>
```

Gyakorlaton

- Üres HTML oldal
- Benne egy inline szkript elem
- Később finomítjuk

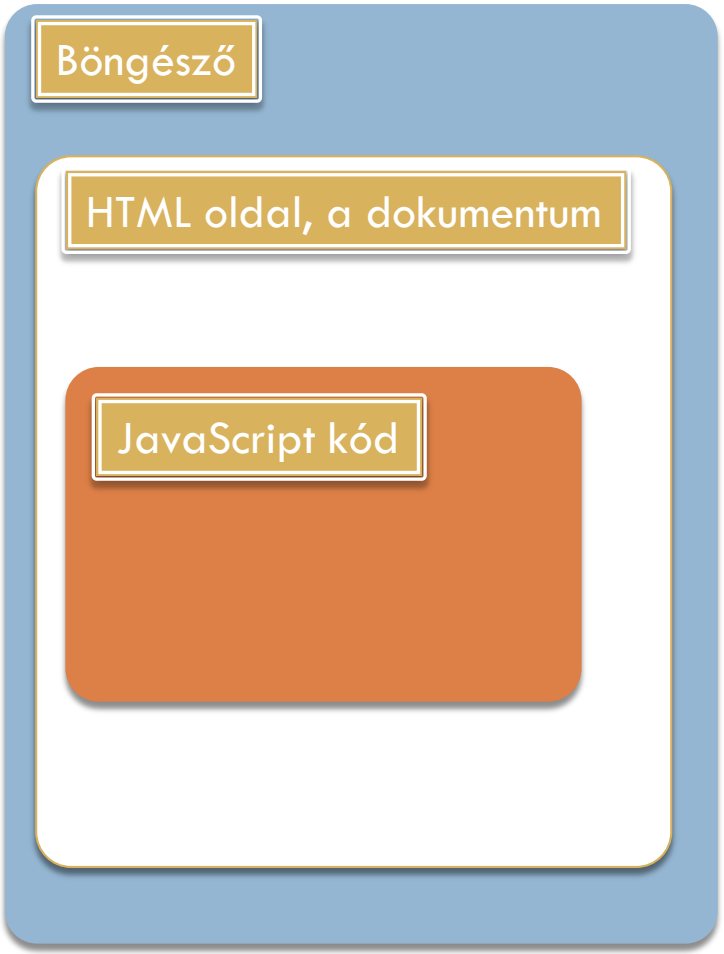
```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Webfejlesztés 2.</title>
  </head>
  <body>
    <script type="text/javascript">
      //JavaScript kód helye
    </script>
</body>
</html>
```

JavaScript nyelvi alapok

C++ → JavaScript

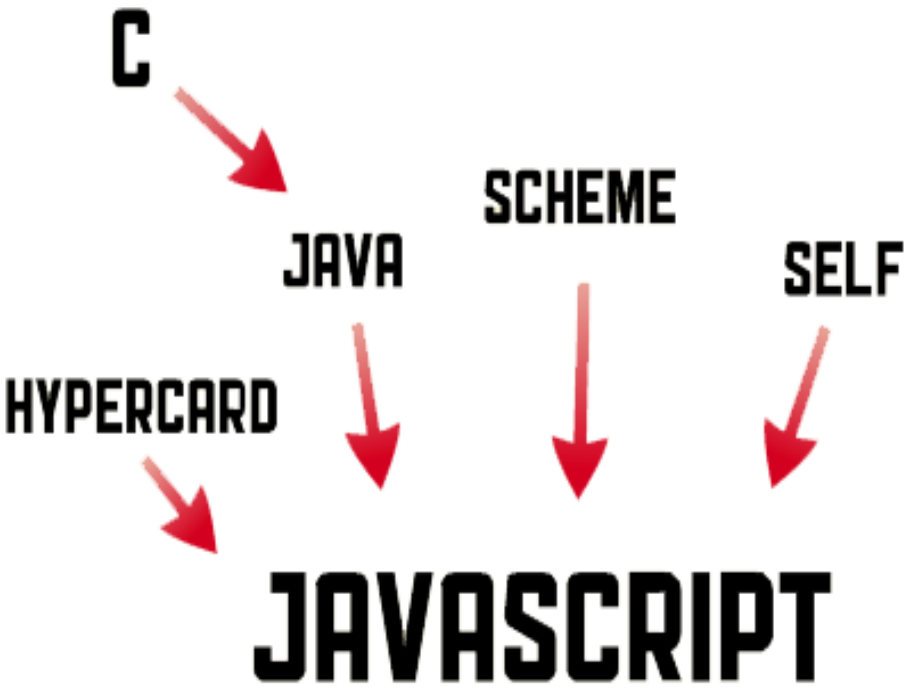
Felépítés

- Böngésző betölti a dokumentumot
- Dokumentumban JavaScript kód van
- Megismerés
  - JavaScript nyelv
  - JavaScript és a dokumentum
  - JavaScript és a böngésző



Mintaként szolgáló nyelvek

- Sokféle minta → sokoldalúság
- Megismerés útja: C++ → JavaScript (hasonlóság).
- Később: különbségek



C++ vs JavaScript

C++

- Erősen típusos
- Fordított
- Általános célú programozási nyelv

JavaScript

- Gyengén típusos
- Interpretált
- Szkriptnyelv

Gyengén típusos

- Dinamikusan tipizált nyelv
- A változók típusa a benne tárolt érték típusától függ
- Vagy másképp: a típusok az értékekhez tartoznak, nem a változókhoz.
- Sok automatikus típuskonverzió kifejezésekben, összehasonlításokban
  - Szövegből szám
  - Bármiből szöveg
  - Stb, stb.

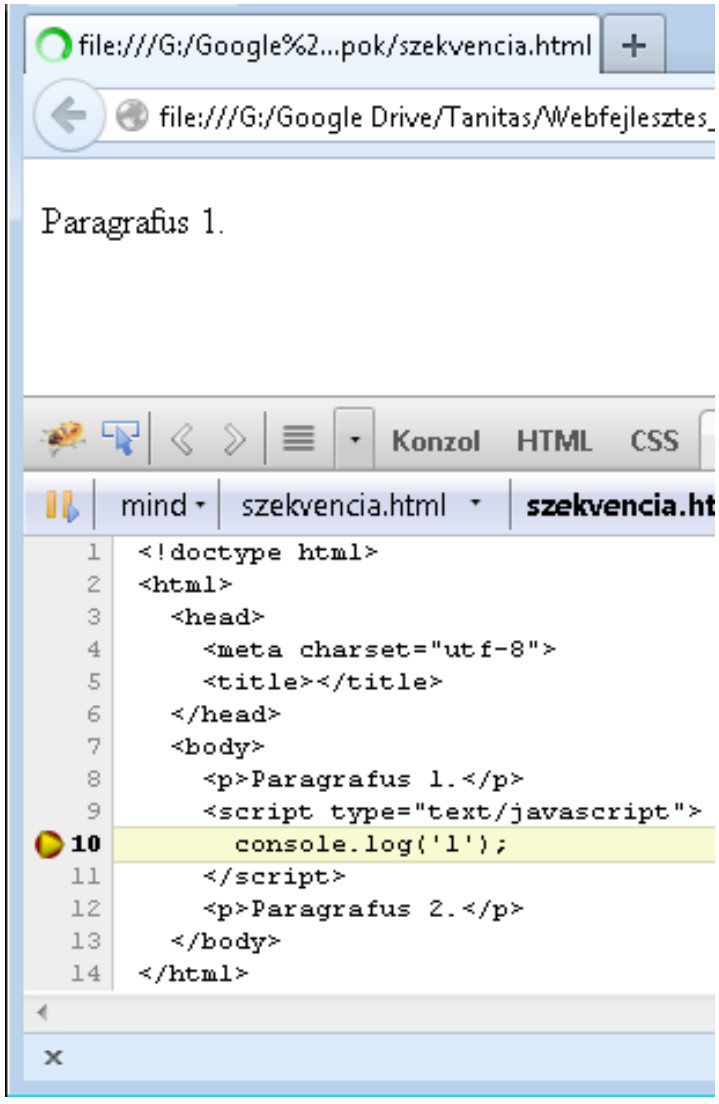
```
a = 'alma';
a = 12;
a == '12'; //true
```

Interpretált

- A böngészőben futó értelmező értelmezi a JavaScript kódsorokat sorról sorra
- Nincs fordítási fázis, nem a lefordított kód fut
- Minimális előfeldolgozás történik
- A kód a hibáig lefut, ott elakad
  - Az adott <script> blokk futása megáll
  - A <script> blokk után folytatódik az oldal betöltése.

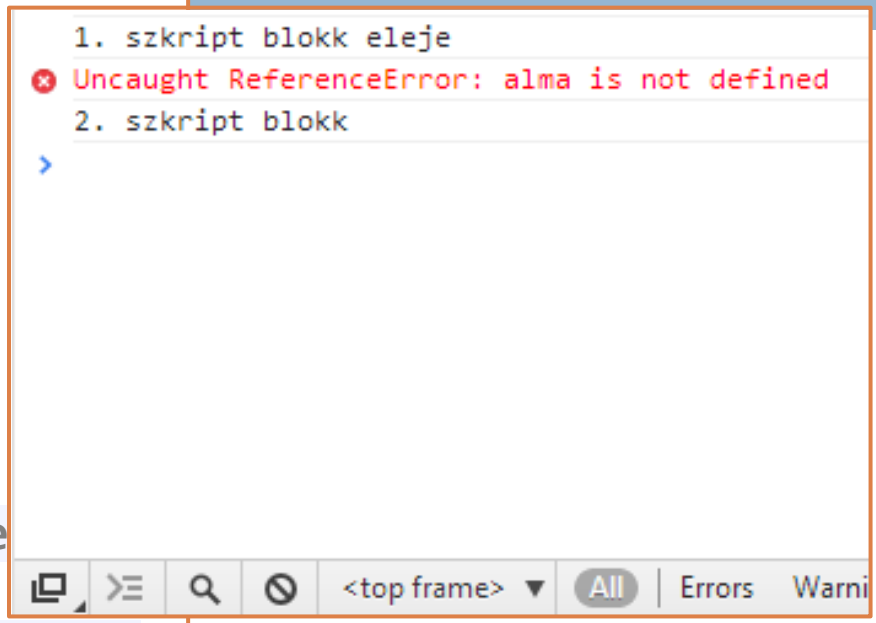
Oldalbetöltési szekvencia


```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <p>Paragrafus 1.</p>
    <script type="text/javascript">
      console.log('1');
    </script>
    <p>Paragrafus 2.</p>
</body>
</html>
```



Betöltés hiba esetén

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <p>Paragrafus 1.</p>
    <script type="text/javascript">
      console.log('1. szkript blokk ele
      alma.szin = 'piros';
      console.log('1. szkript blokk vége');
    </script>
    <p>Paragrafus 2.</p>
    <script type="text/javascript">
      console.log('2. szkript blokk');
    </script>
    <p>Paragrafus 3.</p>
  </body>
</html>
```



További jellemzők

- Kis és nagybetűk különböznek
- Nincs főprogram (main)
- Nincs input/output
- Nincs fájlkezelés
- Objektumorientált
- Prototípusos
- Automatikus pontosvessző beszúrás

Típusok

- Egyszerű típusok
  - Szám
  - Szöveg
  - Logikai
  - null
  - undefined
- Összetett típusok
  - Tömb
  - Objektum

Literálformák

- Így jelennek meg kifejezésekben, pl. értékadásban

```
//Szám literál
12
12.34

//Szöveg literál
'Szöveg'
"Szöveg"
'Idézőjelben "így"
macsakörmölök'
"Macskakörömben 'így' idézek"
'Idézőjelben \' idézőjel'
"Macskakörömben \" macskaköröm"
'Escape: \t \n \\ '

//Logikai literál
true
false
```

Változók

- var kulcsszóval deklarálunk új változót
  - Függvényen belül lokális
  - Függvényen kívül globális
- var elhagyásával → globális változó – KERÜLENDŐ!!!
- Ha nincs kezdőérték → undefined

```
var nev = 'Győző';
var masik;  //undefined
```

Operátorok

- Aritmetikai operátorok
  - +, -, *, /, %, ++, --, unáris -, unáris +
- Értékadás operátorai
  - =, *=, /=, %=, +=, -=, stb.
- Összehasonlító operátor
  - ==, !=, ===, !==, >, >=, <, <=
  - == és != *érték* szerint vizsgál (automatikus konverziók)
  - === és !== *érték és típus* szerint

Operátorok

- Logikai operátorok
  - &&, ||, !
- Szövegösszefűzés operátorai
  - +, +=
- Bitenkénti operátorok
  - &, |, ^, ~, <<, >>, >>>
- Speciális operátorok
  - ? : feltételes operátor
  - , több kifejezés végrehajtása egy utasításban, visszatérési értéke az utolsó kifejezés

Vezérlési szerkezetek

- Teljesen hasonló a C++-ban megismertekhez

```
if (felt) {
  utasítások
}

if (felt) {
  utasítások
} else {
  utasítások
}
```

```
switch(kifejezés) {
  case érték1:
    utasítások
    break;
  case érték2:
    utasítások
    break;
  default:
    utasítások
}
```

```
while (felt) {
  utasítások
}

do {
  utasítások
} while (felt);

for (var i = 1; i <= n; i++) {
  utasítások
}

for (var tul in obj) {
  utasítások
}
```

Tömbök


```
//Létrehozás és hivatkozás
var uresTomb = [];
var tomb = [12, 'alma', true];
tomb[0]; // => 12;
tomb[1]; // => 'alma';
tomb[2]; // => true;
tomb.length // => 3

//Módosítás
tomb[0] = 13;
tomb[0]; // => 13
```

```
//Bővítés
tomb[tomb.length] = 'uj';
tomb[100] = 'messze';
tomb.length; // => 101
tomb[99]; // => undefined

//Törlés (méret nem változik)
delete tomb[1];
tomb[1]; // => undefined
tomb.length; // => 101
```

```
//Mátrix
var m = [[1, 2, 3], [4, 5, 6]];
m[1][2]; // => 6
```

Tömbök


```
var gyumolcsok = [
  'alma',
  'korte',
  'szilva'
];
//A gyümölcsök kiírása a konzolra
for (var i = 0; i < gyumolcsok.length; i++)
{
  console.log(gyumolcsok[i]);
}
```

Objektumok

- Objektum
  - Kulcs-érték párok gyűjteménye
  - Asszociatív tömbhöz hasonlít (hash)
  - Rekord, osztálypéldány szimulálható
- JavaScriptben nagyon fontos szerepük van
- Majdnem minden objektum
- Literálforma: { … }
- Ha az érték függvény → metódus

Objektumok


```
//Létrehozás
var uresObj = {};
var obj = {
  mezo1: 12,
  'mezo2': 'alma'
};

//Hivatkozás
obj.mezo1; // => 12
obj['mezo1']; // => 12

//Módosítás
obj.mezo2 = 'korte';
```

```
//Bővítés
obj.mezo3 = true;

//Törlés
delete obj.mezo1;
obj.mezo1; // => undefined

//Metódus
obj.metodus = function () {
  console.log('Meghívtak');
}
obj.metodus(); // => Meghívtak
```

Objektumok


```
//Tömb az objektumban
var zsofi = {
  kor: 7,
  kedvencEtelek: [
    'krumplipüré',
    'rántott hús',
    'tejberizs'
  ]
};
//Elem elérése
zsofi.kedvencEtelek[0];

// => 'rántott hús'
```

```
//Objektum az objektumban
var david = {
  kor: 4,
  cim: {
    iranyitoszam: '1241',
    varos: 'Budapest',
    utca: 'Holvoltholnemvolt utca',
    hazszam: 63
  }
};
//Elem elérése
david.cim.utca;
// => 'Holvoltholnemvolt utca'
```

Objektumok


```
//Feldolgozás a for..in ciklussal
var matyi = {
  kor: 1.5,
  fiu: true,
  cuki: true
}
//Elemek kilistázása a konzolra
for (var i in matyi) {
  console.log(i, matyi[i]);
}
//Eredmény
// => kor 1.5
// => fiu true
// => cuki true
```

Adatszerkezetek modellezése


```
//C++ vector --> JavaScript tömb
var kutyuk = [
  'telefon',
  'fülhallgató',
  'pendrive',
  'e-könyv olvasó'
];
```

```
//C++ struct --> JavaScript objektum
var hallgato = {
  nev: 'Mosolygó Napsugár',
  neptun: 'kod123',
  szak: 'Informatika BSc'
};
```

```
//Rekordok tömbje
var hallgatok = [
  {
    nev: 'Mosolygó Napsugár',
    neptun: 'kod123',
    szak: 'Informatika BSc'
  },
  {
    nev: 'Kék Ibolya',
    neptun: 'kod456',
    szak: 'Informatika BSc'
  }
];
```

Adatszerkezetek modellezése

- JSON
- JavaScript Object Notation

```
//Tömböt tartalmazó rekordok tömbje
var hallgatok = [
  {
    nev: 'Mosolygó Napsugár',
    neptun: 'kod123',
    szak: 'Informatika BSc',
    targyak: [
      'Programozás',
      'Webfejlesztés 2.',
      'Számítógépes alapismeretek'
    ]
  },
  {
    nev: 'Kék Ibolya',
    neptun: 'kod456',
    szak: 'Informatika BSc',
    targyak: [
      'Programozás',
      'Webfejlesztés 2.',
      'Diszkrét matematika',
      'Testnevelés'
    ]
  }
];
```

Függvények


```
//Függvény általános formája
function fvnev(par1, par2) {
  var valt; //lokális változó
  //függvénytörzs
  return visszatérési_érték;
}

//Például
function osszead(a, b) {
  return a + b;
}
```

- Visszatérési érték tetszőleges elemi vagy összetett érték
- Ha nincs return → undefined a visszatérési érték

Tétel példa

- Konzolba vagy HTML-be helyezve

```
function osszegzes(tomb) {
  var s = 0;
  for (var i = 0; i < tomb.length; i++) {
    s = s + tomb[i];
  }
  return s;
}

var x = [1, 3, -2, 8];
console.log('Az összeg: ' + osszegzes(x));
```

Alapvető kommunikáció

Beolvasás, kiírás

Kommunikáció – input, output

- Beépített felugró ablakokkal
  - alert(szöveg) – kiírás
  - confirm(szöveg): logikai – beolvasás: logikai érték
  - prompt([szöveg, [érték]]) – beolvasás: szöveges érték
- HTML dokumentumon keresztül
  - document.writeln(html szövegként) – kiírás
  - Űrlapelemek – beolvasás (következő előadás)

Kommunikáció példák


```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Webfejlesztés 2.</title>
  </head>
  <body>
    <script type="text/javascript">
      alert('Hello világ!');

      var kerdes = confirm('Szereted a csokit?');

      var nev = prompt('Mi a neved?');
      var nev = prompt('Mi a neved?', 'Senki bácsi');

      document.writeln('<p>Hello világ!</p>');
    </script>
  </body>
</html>
```

Összefoglalás

- C++ → JavaScript
- Adatszerkezetek
  - elemi
  - összetett: tömb, objektum
  - JSON
- Programozási tételek függvényekben
- Ld. gyakorlat