Primeri uporabe jezika javascript za splošno programiranje

Leon Kos

12. oktober 2000

Povzetek

Jezik Javascript (JS) se tolmači v večini internet brskalnikov in zato ni potrebno za pisanje programov imeti prevajalnika. Kot internetni jezik je tako neodvisen od arhitekture računalnika in je za razvoj programa potreben le urejevalnik besedila in brskalnik. Izvorna koda (program) je del HTML datoteke. Jezik je objektno orientiran in vsebuje večino lastnosti modernih jezikov. Z uporabo FORM lahko implementiramo uporabniški vmesnik in ga povežemo s funkcijami napisanimi v JS. Tako lahko sledimo modelu ločenosti uporabniškega vmesnika od jedra programa.

Naštete lastnosti jezika so primerne za reševanje problemov, ki nimajo zahtev po veliki računski moči. Sem spadajo šolski primeri in enostavni inženirski problemi s preračuni. Za razumevanje JS je potrebno razširiti znanje jezika C z poznavanjem HTML, FORM in objektnega pristopa k programiranju. Namen tega besedila je prikazati študentom primere in osvetliti nekatere strani JS, ki ga pokažejo v luči splošnega programskega jezika za reševanje raznovrstnih inženirskih problemov. Predstavljeni so le pomembni deli jezika, za podrobnejši vpogled in sintakso pa se priporoča literatura, ki je dostopna tudi na internetu.

1 Uvod

Naslednji dve povezavi (PDF) podrobneje predstavljati Javascript:

- Vodič po jeziku Javascript nam predstavi jezik in njegove bistvene dele v pozvezavi z brskalnikom.
- Priročnik Javascript pa do podrobnosti razlaga JS in njegove vgrajene objekte.

Večina knjig, ki predstavljajo nek jezik prične uvod z najmanjšim delujočim programom, s katerim lahko bralec preveri delovanje sistema in dobi vpogled v osnovne dele jezika.

```
1  <html>
2  <head>
3  <script language=javascript>
4  function show(name)
5  {
6    alert("Hello, " + name + '!');
```

```
7   }
8   </script>
9   </head>
10
11  <body>
12  <form>
13  <input type=text size=10 name=tf value="world">
14  <input type=button value="Run me" onclick="show(tf.value)">
15  </form>
16  </body>
17  </html>
```

Datoteka *hello.html* vsebuje vse tri osnovne gradnike Javascript programa. Program je vključen med oznaki `SCRIPT` v jeziku HTML. Program Javascript se običajno vključi v začetek dokumenta med oznaki `HEAD`. V samo telo dokumenta pa se piše ukaze HTML, ki oblikujejo stran. Sem spadajo vsi ukazi HTML in med njimi tudi ukazi za obrazce, ki se pišejo med ukaze `FORM`. Z obrazci običajno izdelamo vnos vhodnih podatkov za program. V polja `text` in `textarea` pa lahko izpisujemo tudi rezultate.

2 Funkcije

Osnova modularnega programiranja je funkcija. Vhodni argumenti funkcije se prenašajo z vrednostjo. To pomeni, da se kot argument prenese kopija vrednosti. To velja za vse osnovne (*core*) objekte (števila, znaki) ne pa tudi za uporabniške objekte, pri katerih se prenese le kopija reference na objekt in ne kopija celotnega objekta. Naslednji primer kaže dva posebna načina uporabe funkcij. Ena od teh je uporaba funkcije za tvorjenje objekta in druga je uporaba podfunkcij (funkcija v funkciji).

```
1   <html>
2   <head>
3   <script language=javascript>
4   function A (a)
5   {
6     this.a = a;
7     function sqr(x) // globalna podfunkcija
8           {
9             return x*x;
10          }
11
12    this.b = function b() // metoda objekta
13          {
14            return sqr(this.a);
15          }
16  }
17
18  function main(a)
19  {
20    var c = new A(a+10);
```

```
21      var b = "abcd";
22      alert(A.sqr(a+20) + ":" + c.b());
23    }
24
25
26    </script>
27    </head>
28
29    <body>
30    <form>
31    <input type=text size=10 name=tf value="1">
32    <input type=button value="Run me" onclick="main(tf.value - 0)">
33    </form>
34    </body>
35    </html>
```

Normalen način uporabe pa je, da v funkcijo prihajajo argumenti, v funkciji se izvrši algoritem in na koncu funkcije se vrne rezultat. Če je funkcija narejena kot podprogram in ne vrača rezultata, potem se funkcija ravno tako vrne v klicoči program.

2.1 Rekurzija

Za prikaz delovanja prenosa argumentov z vrednostjo si poglejmo primer rekurzivnega reševanja zaporedja. Želimo preveriti vsoto k recipročnih števil zaporedja

$$\begin{aligned} t_1 &= 2 \\ t_{i+1} &= t_i(t_i - 1) + 1, \quad \text{za } i \geq 2 \end{aligned}$$

Vsota zaporedja k recipročnih števil naj bi ustrezala naslednjemu sklepu:

$$\sum_{i=1}^{k} \frac{1}{t_i} + \frac{1}{t_{k+1} - 1} = 1$$

```
1   <html>
2   <head>
3   <script language=javascript>
4   function t(i)
5   {
6     if (i == 1)
7       return 2;
8     else
9       return t(i-1)*(t(i-1)-1) + 1;
10  }
11
12  function salzer(k)
13  {
14    var sum = 1/(t(k+1) - 1);
15    for (var i = 1; i <= k; i++)
16      {
```

```
17          sum += 1/t(i);
18        }
19      return sum;
20    }
21
22    </script>
23    </head>
24
25    <body>
26    <form>
27    k:<input type="text" size=3 name=tk value="4"><br>
28    <input type=button value="Salzer sequence"
29           onclick="alert(salzer(tk.value-0))">
30    </form>
31    </body>
32    </html>
```

3 Objektni model

Objekt vsebuje podatke in predpisuje metode, ki s podatki operirajo. JS ima možnost gradnje objektov z uporabo funkcij.

Objekt kompleksnega števila ustvarimo z:

```
function Complex(r, i)
{
        this.re = r;
        this.im = i;
}
```

Funkcija Complex je gradnik objekta, ki vsebuje dve spremenljivki `re` in `im`. Argumenta `r` in `i` sta potrebna pri gradnji objekta in nastavljata začetno vred nost kompleksnega števila. Ključna beseda `this` kaže na trenutni objekt.

Tako ukaz `var a = new Complex(0, 1);` ustvari nov objekt Complex z argumentoma nič in ena za realni in kompleksni del števila. Ko ukaz `new` izvede kreiranje objekta, priredi spremenljivki kazalec, ki kaže na novo ustvarjeni objekt. Komplementarni ukaz je `delete`, ki odstrani objekt iz pomnilnika.

3.1 Spremenljivke v JS

V JS so vse spremenljivke v objektih vidne. Spremenljivke se uporabljajo v ohlapnem načinu. Ime spremenljivke je le referenca na objekt. Objekti pa so lahko: numerična vrednost, znaki (*String*), funkcije in ostali objekti, ki so preddefinirani v samem jedru jezika ali uporabniško definirani. Če spremenljivka kaže na funkcijo, ki operira z objektom ji v skladu z objektnim programiranjem rečemo *metoda*.

Naslednji primer kaže uporabo objektov grafičnih elementov. Definirana sta objekta točka in rob. Za pretvorbo izpisa objekta se uporablja rezervirano ime metode *toString* kateri priredimo funkcijo, ki pretvori vrednosti spremenljivk v objektu v znake primerne za izpis. Ker so spremenljivke v objektu ohlapne, jim lahko priredimo karkoli in s tem tudi funkcije.

```
1  <html>
2  <head>
3  <script language=javascript>
4
5  function Point(number, x, y, z)
6  {
7    this.number = number;
8    this.coords = new Array( x, y, z);
9    this.toString = function ()
10   {
11         return "<" + this.number + ":" + this.coords + ">";
12   }
13 }
14
15 function Edge(number, points)
16 {
17   this.number = number;
18   this.points = points;
19   this.add = add_to_edge;
20   this.toString  = function ()
21   {
22      return number + " {" +  points + "}\n";
23   }
24 }
25
26 function add_to_edge(object)
27 {
28   var name = String(object.constructor).substr(10,4);
29   switch (name)
30         {
31         case "Poin":
32           this.points.push(object);
33           break;
34         case "Edge":
35           this.points.push(object.points);
36           break;
37
38         }
39 }
40
41 function main()
42 {
43   var p1 = new Point(1, 0, 0, 0);
44   var p2 = new Point(2, -1, 2, 3);
45   var e1 = new Edge(1,[p1, p2]);
46   var p3 = new Point(3, 0, 1, -2);
47   e1.add(p3);
48   e1.add(p1);
49   p2.coords[0]  = -21;
50   e1.add(new Edge(2, [p2, p3]));
```

```
51      delete p1;
52      alert(e1);
53
54    }
55    </script>
56    </head>
57
58    <body>
59    <form>
60    <input type=button value="Run me" onclick="main()">
61    </form>
62    </body>
63    </html>
```

Objekt *Edge* ima prirejeno tudi metodo add kateri lahko kot argument damo objekt točke ali robu. V funkciji *add_to_edge* pa ugotavljamo ime objekta in nato dodajamo točke glede na vrsto objekta. Rezervirano ime metode `toString` nam pomaga pri izpisu vsebine objekta.

4 Matrične operacije

Vgrajeni objekt *Array* se lahko uporabi tudi za gradnjo večdimenzionalnih polj. Primer uporabe z objektom kaže množenje matrike z vektorjem.

```
1   <html>
2   <head>
3   <script language=javascript>
4
5
6   function matrix_A()
7   {
8     this[0] = [1, 4,-1, 4];
9     this[1] = [3, 1, 0, 4];
10    this[2] = [2, 0, 1, 0];
11    this[3] = [0, 0, 0, 1];
12    this.toString = function ()
13    {
14          var s = new String();
15          for (var i=0; i < 4; i++)
16            s += '|' + this[i] + "|\n";
17          return s;
18    }
19  }
20
21  function multiply(A, x)
22  {
23    var r = [0, 0, 0, 0];
24    for (var i = 0; i < 4; i++)
25          for(var j = 0; j < 4; j++)
26            r[i] += A[i][j]*x[j];
```

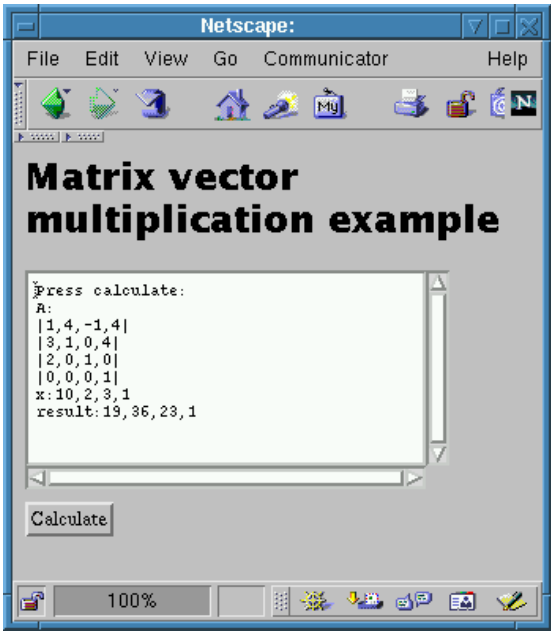
```
27      return r;
28    }
29
30
31    function main(data_form)
32    {
33      var A = new matrix_A();
34      var x = [10, 2, 3, 1];
35
36      var r = multiply(A, x);
37
38      data_form.result.value += "A:\n" +  A
39            + "x:" + x + "\nresult:" + r + '\n';
40
41    }
42    </script>
43    </head>
44
45    <body>
46    <h1>Matrix vector multiplication example</h1>
47    <form name=data>
48    <textarea name=result rows=10 cols=40>
49    Press calculate:
50    </textarea>
51    <br>
52    <input type=button value="Calculate" onclick="main(data);" >
53    </form>
54    </body>
55    </html>
```

Posebnost pri gradnji izpisa je še izpisovanje rezultata v tekstovno polje *textarea*, ki ga kaže slika

5 Razhroščevanje

Za zahtevnejše programe je uporaba izpisnih ukazov `alert` neprimerna. Zato je bolje uporabiti izpis z uporabo sistemske java metode println. Nasledni primer kaže problem podvsote. Iz danega nabora celih števil, moramo poiskati podmnožico, katere seštevek se najbolj približa zahtevani vsoti. Delovanje programa spremljamo v oknu *Java Console*.

```
1   <html>
2   <head>
3   <script language="javascript">
4
5   function sum(a) // size the array
6   {
7     var s = 0;
8     for(var i = 0; i < a.length; i++)
9       s += a[i];
```



Slika 1: textarea uporabljen za izpis rezultatov

```
10    return s;
11  }
12
13
14  function subsum(seq, n) // subset sum problem
15  {
16    java.lang.System.out.println("SEQ = " + seq);
17
18    if (seq.length == 0)
19      return [];
20
21    var r;
22    var last = seq.pop();
23
24    if (last == n)
25      return [n];
26
27
28    r = subsum(seq, n); // search for sum without last
29
30    var delta = n - last;
31
32    if (delta > 0 ) // search for more
33      {
34        var r1 = subsum(seq, delta);
```

```
35          if(sum(r1)+last > sum(r))
36            r = r1.concat([last]);
37        }
38
39      seq.push(last);
40      java.lang.System.out.println("r = " + r);
41      return r;
42    }
43
44    function main(data_form)
45    {
46      var seq = data_form.seq.value.split(',');
47      for (var i = 0; i < seq.length; i++)
48        seq[i]  -= 0; // convert strings to numbers
49      var n = data_form.n.value - 0;
50      var r = subsum(seq, n);
51      alert(sum(r) +  ' = sum{' + r + "}");
52    }
53
54    </script>
55    </head>
56
57    <body>
58    <h1>Subset Sum problem</h1>
59    <form name=data>
60    nondecreasing list:<input type="text" size=60 name=seq
61    value="1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233"><br>
62    sum:<input type="text" size=3 name=n value="100"><br>
63    <input type=button value="Calculate" onclick="main(data);" >
64    </form>
65    </body>
66    </html>
```

6 Odpiranje novega okna

Pri daljših preračunih je ugodno strniti rezultate in vhodne podatke v novem oknu, ki ga lahko nato uporabnik shrani ali iztiska. Naslednji primer kaže odpiranje novega okna. Spremenljivki `w` se priredi prazno okno, ki da odpremo kot *HTML* besedilo in priredimo spremenljivki `d`. Dokument gradimo z ukazi `write` in ga na koncu zapremo, da ga bo lahko brskalnik interpretira in prikaže. Dokument bi lahko odprli tudi kot `text/plain`, kar bi pomenilo drugačen način gradnje izpisa bolj primeren za shranjevanje v ASCII datoteko.

```
1   <html>
2   <head>
3   <script language=javascript>
4   function show(name)
5   {
6     var w = window.open("", null, "width=300,height=400,status=no,location=no,"
7     + "toolbar=no,menubar=yes,scrollbars=yes,resizable=yes");
```

```
 8     var d =  w.document.open("text/html");
 9     d.writeln("<html><head><title>Example</title></head>");
10     d.writeln("<body>Hello, World!</body></html>");
11     d.close();
12   }
13   </script>
14   </head>
15   <body>
16   <form>
17   <input type=button value="Run me" onclick="show()">
18   </form>
19   </body>
20   </html>
```

7 Vaje

7.1 Predstavitev OS, uporaba brskalnika in urejevalnika, jezik HTML

Obljuba Kot študent pri PK/OPK/RPK obljubljam, da bom prizadevno delal na vajah, izdelal naročeno delo in upošteval naslednje pogoje:

1. Ne bom zlorabljal uporabniškega imena na računalniški opremi
2. Ne bom kopiral programov ali pisnih izdelkov drugih oseb. Priporoča pa se izmenjava mnenj z drugimi študenti in pomoč pri razhroščevanju.
3. Ne bom dopustil, da bi drugi kopirali moje delo.
4. Citiral bom vir vseh delov programske kode, ki jih uporabljam in jih nisem napisal sam.

Podpis študenta

7.2 Forme. Hello, World! Vrednosti, spremenljivke, izrazi in operatorji

Osnovni gradniki form, kot vhodno/izhodni uporabniški vmesnik za JS programe. Osnovni tipi spremenljivk in njihove operacije. Uporaba lokalnih/globalnih spremenljivk. Enostavne računske operacije in posebni izrazi.

7.3 Stavki, funkcije, objektni model

Krmilni stavki. Uporaba funkcij in prenos argumentov. Uporabniški objekti in metode. Logično združevanje elementov v objekte in operacije nad njimi.

7.4 Delo z objekti, polja, matrike, asociativne tabele

Manipulacija in povezave objektov in gradnja kompleksnejših baz. Polja in matrike.

7.5 Povezava z java appleti, osnove Phigs

Homeogene geometrijske transformacije. Risanje z JSPhigs. Raster applet.