Slovenska univerza za tretje življenjsko obdobje

Sofinancira program
Evropske unije
Erasmus+

JavaScript za uporabo na spletnih straneh – dogodki in nizi

Dogodki (angl. events)

- Dogodki so stvari, ki se zgodijo HTML elementom.
- Dogodek je lahko nekaj, kar naredi brskalnik, ali nekaj, kar naredi uporabnik (spletna stran se naloži, klik na gumb, …).
- Pogosto, ko se nekaj zgodi, želite nekaj narediti. JavaScript nam dovoli izvedbo kode, ko je dogodek prepoznan.
- JavaScript dogodke dodajamo elementom HTML kot atribute:

<element dogodek=‚malo JavaScript kode‘>

ali

<element dogodek=„malo JavaScript kode“>

Dogodki - primeri

- **Vsebino elementa spremenimo z uporabo id=„demo“**

```
<!DOCTYPE html>

<html>

<body>

        <button onclick="document.getElementById('demo').innerHTML=

        Date()">The time is?</button>

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

</body>

</html>
```

Dogodki - primeri

- **V tem primeru spremenimo vsebino istega elementa, z uporabo this.innerHTML**

```
<!DOCTYPE html>

<html>

<body>

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

</body>

</html>
```

Dogodki - primeri

- **Bolj običajno je, da atributi dogodkov prikličejo funkcijo**

```
<!DOCTYPE html>

<html>

<body>

		<p>Klikni na gumb, da se izpiše datum</p>

		<button onclick=„prikaziDatum()">Koliko je ura?</button>

		<script>

			function prikaziDatum() {

			document.getElementById("demo").innerHTML = Date();

		}

		</script>

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

</body>

</html>
```

Pogosto uporabljeni HTML dogodki

Dogodek	Opis
onchange	HTML element je bil spremenjen
onclick	Uporabnik klikne na HTML element
onmouseover	Uporabnik gre z miško čez HTML element
onmouseout	Uporabnik odmake miško s HTML elementa
onkeydown	Uporabnik pritisne tipko na tipkovnici
onload	Brskalnik je naložil stran

Kaj lahko JavaScript naredi?

- Dogodkovne atribute lahko uporabljamo za delo in kontrolo uporabnikovih dejanj, dejanj brskalnika ter za delo s podatki, ki jih uporabnik vnese, ter kontrolo le teh:

  - Stvari, ki morajo biti narejene, vsakič, ko se stran naloži
  - Stvari, ki morajo biti narejene, ko se stran zapre
  - Dejanje, ki naj sledi, ko uporabnik klikne na gumb
  - Vsebina, ki jo je treba preveriti, potem, ko jo je uporabnik vnesel
  - …

Kaj lahko JavaScript naredi?

- Veliko različnih metod lahko uporabimo za delo z dogodki:
  - HTML dogodkovni atributi lahko izvedejo JavaScript kodo direktno
  - HTML dogodkovni atributi lahko prikličejo JavaScript funkcije
  - HTML elementom lahko določimo svoje funkcije dogodkovnega atributa
  - Lahko preprečimo dogodkom, da se zgodijo
  - ….

Nizi (angl. strings)

- Nize uporabljamo za hranjenje in upravljanje besedila.
- Niz predstavlja nič ali več znakov, zapisanih znotraj narekovajev:

  var x = „Neznana oseba“;

- Uporabljamo lahko dvojne ali enojne narekovaje.
- Narekovaje lahko uporabljamo tudi znotraj niza, če se le ne ujemajo s tistimi, ki obkrožajo niz. Torej, če uporabimo dvojne narekovaje za niz, bomo znotraj niza uporabili enojne narekovaje.

Nizi – posebni znaki

- Kadar se ne moremo izogniti dvojnim ali enojnim narekovajem znotraj in zunaj niza, lahko uporabimo tako imenovan **backslash escape znak**.
- Ta znak spremeni posebne znake v znake niza:

Koda	Rezultat	Opis
\'	'	Enojni narekovaj
\"	"	Dvojni narekovaj
\\	\	Leva poševnica

Prelom vrstice kode

- Zaradi boljše berljivosti kode, se programerji pogosto izogibajo vrsticam kode, ki so daljše od 80 znakov. Vrstico kode lahko prekinemo na več načinov:

1. Trditev je najbolje prekiniti za znakom za enačaj:

```
document.getElementById("demo").innerHTML =
„Zdravo, svet!";
```

2. Prekinemo lahko tudi znotraj niza, z uporabo leve poševnice (To velja le znotraj niza, funkcije ne moremo razdeliti na ta način!):

```
document.getElementById("demo").innerHTML = „Zdravo, \
svet!";
```

3. Niz je bolj varno razdeliti z uporabo znaka +:

```
document.getElementById("demo").innerHTML = "Hello " +
"Dolly!";
```

Metode in lastnosti nizov

- Primitivne vrednosti, kot je npr. „Neznana oseba“, ne morejo imeti lastnosti in metod, ker niso objekti.
- Vendar pa so v JavaScriptu lastnosti in metode dosegljive tudi primitivnim vrednostim, saj jih JavaScript obravnava kot objekte, kadar izvaja metode in lastnosti.

Lastnost length (slov. dolžina)

- JavaScript ima že vgrajeno lastnost **length** (slov. dolžina), ki nam pove koliko znakov ima niz.
- Primer:

```
var tekst = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var dolzina_niza = tekst.length;
```

Iskanje niza znotraj niza

- Metoda **indexOf()** prikaže indeks (pozicijo) prve pojavitve posameznega besedila znotraj niza.
- JavaScript pozicije šteje od ničle. Torej, prva pozicija je nič, druga je 1, tretja je 2 itn.
- var niz = “Please locate where 'locate' occurs!";

  var poz = niz.indexOf("locate"); // rezultat je 7

Iskanje niza znotraj niza

- Metoda **lastIndexOf()** prikaže indeks (pozicijo) zadnje pojavitve posameznega besedila znotraj niza:

```
var niz = “Please locate where 'locate' occurs!";
var poz = niz.lastIndexOf("locate");  // rezultat je 21
```

Iskanje niza znotraj niza

- Obe metodi, **indexOf()** in **lastIndexOf()** vrneta -1, če besedilo ni najdeno:

```
var niz = "Please locate where 'locate' occurs!";
var poz = niz.lastIndexOf("John");        // rezultat je -1
```

Obe metodi tudi dovoljujeta drugi parameter, kot začetno točko iskanja:

```
var niz = "Please locate where 'locate' occurs!";
var poz = niz.indexOf("locate",15);       //rezultat je 21
```

Iskanje niza znotraj niza

- Uporabimo lahko tudi metodo **search(),** ki poišče točno določeno vrednost niza in vrne pozicijo ujemanja:

```
var niz = "Please locate where 'locate' occurs!";
var poz = niz.search("locate");    // rezultat je 7
```

Iskanje niza znotraj niza

- Metodi indexOf() in search() sta enaki. Sprejemata iste argumente (parametre) in vrneta enako vrednost, vendar pa metoda search() ne more sprejeti drugega argumenta (začetne pozicije iskanja), metoda indexOf() pa ne sprejema močnih iskalnih vrednosti, kot so regular expressions.
- Več metod za delo z nizi si poglejte na www.w3schools.com.

Niz nizov (angl. Array)

- Niz nizov je posebna spremenljivka, ki lahko hrani več kot eno vrednost naenkrat.
- Če imamo seznam predmetov (recimo avtomobilov), lahko shranimo vsak avto v svojo spremenljivko:

```
var avto1 = "Saab";
var avto2 = "Volvo";
var avto3 = "BMW";
```

Niz nizov (angl. Array)

- Kaj pa, če želimo brskati po seznamu avtomobilov in poiskati točno določenega? In če imamo 300 avtomobilov, ne samo tri?
- Niz nizov je rešitev, ker lahko shranimo več vrednosti pod eno ime in potem dostopamo do teh vrednosti s priklicom indeksne številke (pozicije).
- Sintaksa:

```
var ime_niza_nizov = [predmet1, predmet2, predmet4, …]
```

Niz nizov (angl. Array)

- Primer:

var avtomobili = [„Saab“, „Volvo“, „Fičo“]

ali

var avtomobili = [

"Saab",

"Volvo",

„Fičo"

];

Niz nizov (angl. Array)

- Do elementov niza nizov dostopamo s priklicem indeksne številke (pozicije).
- S to trditvijo dobimo vrednost prvega elementa v avtomobili:

```
var ime = avtomobili [0];
```

- Z naslednjo trditvijo spremenimo ime prvega elementa:

```
avtomobili [0] = „Opel“;
```

Niz nizov (angl. Array)

- Lahko dostopamo tudi do celotnega seznama avtomobili, tako da se sklicujemo na ime niza nizov:

```
var avtomobili = [„Opel", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = avtomobili;
```

Niz nizov (angl. Array)

- Nizi nizov so posebna vrsta objektov. Nizi nizov dostopajo do svojih elementov z uporabo številk, medtem ko objekti dostopajo do svojih članov s sklicevanjem na ime.
- V primeru niza nizov nam **oseba[0]** vrne vrednost Neznana:
- var oseba = [„Neznana“, „oseba“, 46];
- V primeru objekta nam **oseba.ime** vrne vrednost Neznana:
- var oseba = (ime:“Neznana“, priimek:“oseba“, starost:46);

Niz nizov (angl. Array)

- Nizi nizov so posebni tipi objektov, zato lahko vsebujejo različne tipe spremenljivk, od objektov, funkcij do niza nizov:
- mojNiz[0] = Date.now;

  myNiz[1] = mojaFunkcija;

  myNiz[2] = mojiAvtombili;

Dodajanje elementov v niz nizov

1. Najlažji način je **metoda push** (slov. porini):

var sadje = ["Banana", „Pomaranča", „Jabolko", "Mango"];

fruits.push("Limona"); // doda nov element (limono) v sadje

2. Z uporabo **lastnosti length** (slov. dolžina):

var sadje = ["Banana", „Pomaranča", „Jabolko", "Mango"];

sadje[sadje.length] = "Limona"; // doda nov element (limono) v sadje

- 3. Z uporabo **indeksne številke**:
- var sadje = ["Banana", „Pomaranča", „Jabolko", "Mango"];
  fruits[5] = "Limona"; // doda nov element (limono v sadje

Nizi nizov in objekti

- Glavna razlika med nizi nizov in objekti je, da nizi nizov uporabljajo indeksne številke in objekti indeksna imena. Če želite, da so imena elementov besedilo, uporabite objekt in kadar želite, da so imena elementov številke, uporabite niz nizov.
- Niz nizov uporablja indeksne številke:

```
var oseba = [];
oseba[0] = “Neznana";
oseba[1] = „oseba";
oseba[2] = 46;
var x = oseba.length;        // oseba.length vrne rezultat 3
var y = oseba[0];            // oseba[0] vrne rezultat „Neznana"
```

Nizi nizov in objekti

- Ne indeksnih imen, kakor objekti:

```
var oseba = [];
ime[„ime"] = „Neznana";
person[„priimek"] = „oseba";
person[„starost"] = 46;
var x = oseba.length;        // oseba.length vrne rezultat 0
var y = oseba[0];            // person[0] vrne rezultat undefined
```