

Samoobnavljajuće aplikacije preko Javascripta

autor: zli vukodlak iv43141

Do sada ste koristili XAJAX za svoj rad koji je djelomice izvođen na klijentu, a djelomice na poslužitelju. No, ukoliko želimo da aplikacije kontinuirano obnavljaju svoje podatke, očito je da se barem dio procesiranja mora napisati na klijentu; odnosno, moramo se zadubiti u ono slovo J u riječi AJAX, to jest u Javascript. Ne brinite, toga ćemo činiti vrlo malo, čisto da bismo postigli odgođeno obnavljanje, odnosno kontinuirano obnavljanje.

Prvo ćemo uvesti novi tag u HTMLu u koji se pišu klijentske skripte. Taj tag je `<script>`.

```
<script type="text/javascript">
// kod ide ovdje
</script>
```

Atribut `type` označava tip podataka naveden unutar taga `<script>`. Nekada se koristio atribut `language="javascript"` i još ćete ga često vidjeti, no službeno je "umirovljen" (tj. *deprecated*). Čisto povijesno, zanimljivo je da Internet Explorer zna pričati još jedan jezik, a to je VBScript. No nemojte pisati u njemu iz očitih razloga :-)

Ovaj tag se meće unutar `<body>` taga.

I kakav kod ide unutar tog taga? Evo kratak *hello world* primjer koda koji će se izvršiti odmah pri pozivu stranice.

```
<script type="text/javascript">
  alert('Zdravo svijete!');
</script>
```

Uvedimo i funkcije:

```
<script type="text/javascript">
  function nasha_f() {
    alert('Zdravo svijete!');
  }
  nasha_f();
</script>
```

Pozovimo funkciju preko `javascript: URLa`, odnosno preko `onclick`:

```
<a href="javascript:nasha_f();">Klikni me!</a>
<button onclick="nasha_f();">I mene isto!</button>
```

XAJAX pomoću `$ajax->printScript()` ustvari ispisuje ovakav poziv na funkciju unutar našeg `onclick`. To znači da možemo ovu XAJAXovu funkciju koristiti da ubacimo takav poziv i unutar neke Javascript funkcije:

```
<script type="text/javascript">
  function neka_druga() {
    <?
      $xf->printScript();
    ?>
  }
</script>
<button onclick="neka_druga();">Super</button>
```

Konačno uvedimo Javascript funkcije za ponavljanje komada koda!

```
<script type="text/javascript">
  setTimeout("alert('Hello world');", 500);
</script>
```

setTimeout() prima **string** s komadom Javascript koda, te **broj milisekundi** na koliko se odgađa izvođenje. Kod naveden u stringu se izvršava **samo jednom**.

Ovako kako smo napisali, setTimeout() se izvršava još za vrijeme učitavanja stranice (čim dođe do zatvorenog </script> taga), što nama za sada paše. Ponekad *browser* može odlučiti da se izvrši tek na kraju učitavanja cijele stranice, no to općenito nije slučaj ako mu to posebno ne napomenete, no to vas generalno ne treba zabrinjavati... posebice ako stavite komad javascripta skroz na kraj :-)

Ako se XAJAXu rekli da koristi *single quoteove*, odnosno apostrofe, slobodno napišete nešto u ovom stilu:

```
<script type="text/javascript">
  setTimeout("<? $xf->printScript(); ?>", 500);
</script>
```

To znači: učita se stranica, i 500ms (0.5s) nakon učitavanja, izvrši se ono navedeno u stringu. Moramo koristiti *single quoteove* pri ispisu funkcije iz istog razloga iz kojeg ih koristimo u *onclick handleru*.

E sad, mi smo postavili da 30s nakon učitavanja stranice uklonimo komad *interfacea*. Ali korisnik je kliknuo na neki button, pa sad ipak želimo ostaviti *interface*, odnosno poništiti postojeći *timeout*! Ništa, idemo se malo s varijablama igrati.

```

<script type="text/javascript">
  // u cisto edukativne svrhe stavljamo id na 0
  // naime setTimeout() ne moramo izvorsavati u glavnom kodu, nego
  // u nekom onclickku ili funkciji pozvanoj iz onclickka, pa
  // bi bilo fino imati default.
  var idTimera = 0;

  // ali za ovaj primjer idemo odmah dodijeliti "pravu"
  // vrijednost, s obzirom da nama timer kreće odmah nakon
  // učitavanja ;)
  idTimera = setTimeout("<? $xf->printScript(); ?>", 30*1000);

  // trpamo sve u odvojenu funkciju. naime idemo biti pravi
  // programeri(tm) i pocistiti idTimera i ne ponavljati
  // ciscenje ako je vec pocisceno.
  // nadajmo se da neki browser neće vratiti 0 za idTimera :-)
  function pocistiTimer()
  {
    if (idTimera) // znaci, i razlicit od 0 i razlicit od undef
    {
      clearTimeout(idTimera);
      idTimera = 0;
    }
  }
</script>
<button onclick="pocistiTimer();">Ponisti timer</script>

```

Što vidimo iz ovog primjera?

- varijable mogu imati bilo koji tip podataka
- deklariraju se s `var nesto;` ili `var nesto = difoltna_vrijednost;`
- `setTimeout()` ustvari vraća jedan *id* koji možemo koristiti u `clearTimeout()`

E sad, dosta ljudi koji čuju za `setTimeout()` misle – to je to. Autor ovih slova bješe jedan od takvih. Naime, možete vrlo lako izvesti ponašanje sljedeće funkcije pomoću `setTimeout()`, no zašto ako ne morate.

Stoga dozvolite da vam deda pokaže ... `setInterval()` :-)

```
<div id="okvir"></div>
<script type="text/javascript">
  // ova funkcija dodaje nesto teksta na kraj diva 'okvir'
  function dodajTeksta()
  {
    // nije preporucljivo pisati izravno po innerHTML
    // naime ako bismo ovo radili u for petlji, browser pri
    // svakoj izmjeni innerHTML ide ponovno generirati neke
    // interne strukture.
    // stoga idemo iskopirati u privremenu varijablu

    var privremeno = document.getElementById('okvir').innerHTML;
    privremeno += 'Kmeee!!!!<br>';
    document.getElementById('okvir').innerHTML = privremeno;
  }
  var idTimera = setInterval("dodajTeksta();", 1000);
</script>
<button onclick="clearInterval(idTimera);">Daj stani vise</button>
```

To je to! Na vama je da sami otkrijete sve čari kombiniranja `setTimeout()` i `setInterval()` s XAJAXom. Sretno!

Napomena: Ovaj tekst sadrži loš Javascript kod; u skorije vrijeme će biti objavljen dodatak kako se inače rade real-time aplikacije.