

10. JavaScript (2)

Obsah

- Premenné
- Práca s reťazcami
- Vlastné funkcie
- Podmienka IF
- Cyklus FOR
- Operátory
- Kontrola hodnôt prvkov formulára

Premenné

- Môžeme vytvárať prakticky kdekoľvek.
- Nemusíme ich deklarovať, stačí ich jednoducho používať.
- Napr. do zdrojového kódu napíšeme **vysl=x+y** a premenná **vysl** sa automaticky vytvorí.
- Premennú môžeme definovať aj pomocou kľúčového slova **var**, napr. **var vysl=x+y**.
- Meno premennej nesmie byť rovnaké ako meno ľubovoľnej funkcie.
- **Premenná platí len na stránke, kde bola vytvorená a len dotedy, kým sa neobnoví (refresh, reload) stránka.**

Funkcie a vlastnosti na prácu s reťazcami (1)

funkcia, vlastnosť	popis
indexOf(znak)	Vráti index znaku od začiatku reťazca . Ak sa nenachádza, vráti hodnotu -1 .
charAt(poz)	Vráti znak na pozícii poz v reťazci. (indexy v reťazci sú od 0)
length	Dĺžka reťazca.
concat(str1, str2, ..., stringN)	Spojí viacero reťazcov.
substr(zaciatok, pocet)	Vráti podreťazec, ktorý začína na pozícii zaciatok a má maximálne pocet znakov .
1) substring(zaciatok, koniec) 2) substring(zaciatok)	1) Vráti podreťazec, ktorý začína na pozícii zaciatok a končí pred pozíciou koniec . 2) Vráti podreťazec, ktorý začína na pozícii zaciatok až do konca reťazca.

Funkcie a vlastnosti na prácu s reťazcami (2)

funkcia, vlastnosť	popis
lastIndexOf(searchvalue, fromindex)	Parameter searchvalue je povinný. Vrátí pozíciu posledného výskytu hľadaného reťazca searchvalue . Vyhľadávanie prebieha od konca reťazca – pozícia je od začiatku.
replace(findstring, newstring)	Oba parametre sú povinné. Nahradí všetky výskyty podreťazca findstring reťazcom newstring .
search(findstring)	Vyhľadá reťazec findstring a vráti jeho pozíciu v reťazci. Ak sa reťazec nenachádza, vráti hodnotu -1 .
toLowerCase()	Prevedie reťazec na malé znaky.
toUpperCase()	Prevedie reťazec na veľké znaky.

Reťazce

■ Reťazce sú objekty

- **Správne použitie: `meno.charAt(0)`**
- Nesprávne použitie: `charAt(meno, 0)`
- **Správne použitie: `meno.toUpperCase()`**
- Nesprávne použitie: `toUpperCase(meno)`

■ Reťazce sú indexované od 0, teda 1. znak získame ako **`meno.charAt(0)`**.

Zápis, použitie a rozdiel

- Zapisujeme:
- `reťazec.funkcia()`, napr. **`email.indexOf('@')`**;
- `reťazec.vlastnosť`, napr. **`meno.length`**;

- `var str = "Tvorba webových stránok";`
- `window.alert(str.substr(7, 8));` // webových
- `window.alert(str.substring(7, 8));` // w

www.w3schools.com/jsref/jsref_obj_string.asp

ukážka: `ukazka-01.html`

Vlastné funkcie

- Vlastnú funkciu definujeme nasledovne:
`function menoFunkcie(parametre) {`
 `telo funkcie`
 `}`
- Ak chceme, aby funkcia vrátila nejakú hodnotu, použijeme príkaz **`return`**. Za týmto príkazom sa už nevykonajú žiadne príkazy.
- V prípade, že funkcia neobsahuje príkaz `return`, stáva sa **`procedúrou`**.
- Ak funkcia nemá parametre, zavoláme ju s prázdnyimi zátvorkami, napr. **`vypis()`**.

Vlastné funkcie – príklady

```
function nazov() {  
  telo funkcie  
  return true;  
}
```

```
function sucet(x, y) {  
  return x + y ;  
}
```

```
function sucet(x, y) {  
  var vysl = x + y;  
  return vysl;  
}
```

Podmienka if ... else

```
if (vyraz) {  
  // zoznam príkazov, ktoré sa vykonajú, ak hodnota výrazu  
  je TRUE  
} else {  
  // zoznam príkazov, ktoré sa vykonajú, ak hodnota výrazu  
  je FALSE  
}
```

- Ak zoznam príkazov obsahuje len 1 príkaz, môžeme zátvorky { } vynechať.
- Vetva else je nepovinná.

Podmienka if ... else – ukážka

```
function spravny_email(mail) {  
  if (mail.indexOf("@") < 1) { return false; }  
  else { return true; }  
}
```

ukážka: ukazka-02.html

Cyklus for

```
for (zaciatok; koniec; operacia) {  
  ... zoznam príkazov  
}
```

- **zaciatok**: Vykoná sa na začiatku celého cyklu – tzv. inicializačná podmienka.
- **koniec**: Podmienka, ktorá sa vykonáva (kontroluje) pred začiatkom každého prechodu cyklom. V prípade, že podmienka neplatí, cyklus končí.
- **operácia**: Príkaz(y), ktoré sa vykonajú na konci každého prechodu cyklom. Zvyčajne tu uvádzame príkaz na zvyšovanie premennej cyklu.

```
suma = 0;  
for (i = 1; i <= 10; i++) {  
  suma = suma + i;  
}
```

Operátory

&&	Logické AND
	Logické OR
==	Rovnosť, porovnanie
!=	Nerovnosť
<, <=, >=, >	Operátory porovnania
!	Logické NOT
++	Zvýšenie hodnoty premennej o 1 (napr. i++ znamená i = i + 1)
--	Zníženie hodnoty premennej o 1 (napr. i-- znamená i = i - 1)

Logické operátory – ukážka

```
function spravny_email(mail) {  
  if ( (mail.indexOf("@") == -1) ||  
        (mail.length <= 2) ) { return false; }  
  else { return true; }  
}
```

ukážka: ukazka-03.html

Doplnok ku kontrole formulárov

- Odovzdávanie vlastnej hodnoty v prvku formulára – **this.value**
- `<input name="email" type="text" onchange="spravny_email(this.value);" />`
- Bez použitia `this.value` by sme museli napísať `<input name="email" type="text" onchange="spravny_email(document.form1.email.value);" />`

Radiobutton

- Prvky v skupine radiobuttonov sú uložené v poli.
- Položky sú indexované od 0 a sú do pol'a pridávané v poradí, ako sú definované na stránke.
- **Zvolenie/označenie** konkrétneho radiobuttonu zistíme:
document.meno_formulara.nazov_radiobuttonu[index_polozky].checked
- **Hodnotu** konkrétneho radiobuttonu zistíme:
document.meno_formulara.nazov_radiobuttonu[index_polozky].value

ukážka: ukazka-04.html

Checkbox

- **Zvolenie/označenie/zaškrtnutie** konkrétneho checkboxu zistíme:
document.meno_formulara.nazov_checkboxu.checked
- **Hodnotu** konkrétneho checkboxu zistíme:
document.meno_formulara.nazov_checkboxu.value

ukážka: ukazka-04.html

Select (najmä pre NEmultiple)

- **Index** aktuálne vybratej položky vo výberovej ponuke (čísľuje sa od 0):
document.meno_formulara.nazov_vyberu.selectedIndex
- **Text** aktuálne vybratej položky vo výberovej ponuke:
document.meno_formulara.nazov_vyberu.options[index_položky].text
- **Hodnota** výberovej ponuky:
document.meno_formulara.nazov_vyberu.value

ukážka: ukazka-04.html

**Ďakujem za
pozornosť 😊**

Priestor na Vaše otázky