Tvorba webových stránek

JavaScript

Martin Trnečka

Katedra informatiky
Univerzita Palackého v Olomouci

Jazyk JavaScript

- plnohodnotný programovací jazyk
- autor Eich Brendan (Netscape)
- velká popularita (nejen v prostředí webu)
- skriptování na straně klienta
  - programový kód je vykonáván klientem (prohlížečem)
  - výhody vs. nevýhody
  - rozšíření možností webových stránek
- v tomto kurzu se omezíme na
  - základní skriptování na straně klienta
  - jednoduché konstrukty jazyka

Historie vývoje jazyka JavaScript

- 1995 – vznik jazyka, původně měl sloužit jako programovací jazyk (LiveScript) pro Netscape prohlížeči (název = marketingový tah)
  - začátek „temného období“ webových stránek, popularita JavaScriptu → odklon od původního účelu webu, experimenty s vizualizací stránky mimo standard → strašlivé následky
  - Microsoft přichází s VBScript (nekompatibilní), později portace JSript pro IE, období největších rozdílů mezi prohlížeči (ActiveX komponenty a problémy s tím spojené) → pokusy o normalizaci
- 1997 – první standartizace ECMAscript
- 2011 – nejrozšířenější verze ECMAScript 5.1.
- 2015 – milník, dokončení ECMAScript 6 (největší update v historii, analogie s HTML, CSS, 4 roky od verze 5.1)
- zaveden nový schvalovací proces = každoroční update (verze označovány ECMAScript YYYY)

Začlenění JS do HTML

1. externí soubor (přípona `.js`)

```
<script src="script.js"></script>

<!-- starší způsob (dnes již netřeba) -->
<script type="text/javascript" src="behavior.js"></script>

<!-- asynchronní načítání -->
<script src="behavior.js" async></script>
```

   - v elementu `head` i mimo něj

2. přímo v HTML, JS kód je obsahem elementu `script`

```
<script>
  document.write("Hello World!")
</script>
```

   - kdekoliv v elementu `body`

- pozdější výskyt = pozdější vykonávání kódu

Možnosti klientského JS

- řízení vzhledu a obsahu webové stránky
  - manipulace s HTML
  - manipulace s CSS
- řízení prohlížeče (manipulace s oknem)
- interakce s formuláři
- interakce s uživatelem (události)
- čtení a zápis cookies

Omezení klientského JS

- JS nemá žádné grafické schopnosti, ale
  - lze generovat HTML
  - element `canvas`
- neumožňuje čtení, zapisování souborů ani zpřístupnění diskového prostoru na klientském PC
  - dnes již není zcela pravda
  - různé možnosti uložení dat v klientovi
  - nad rámec kurzu
- běží v jednom vlákně
  - omezeně lze vykonávat i paralelní kód
  - blokuje vstup od uživatele
- již od počátku: „nedokáže provádět žádné škodlivé operace“ $\rightarrow$ diskutabilní

Základy JS

- case-sensitive, camel case konvence
- příkazy ukončeny středníkem (v některých situacích lze vypustit)
- komentáře: `//` nebo `/* */`
- proměnné $\rightarrow$ klíčové slovo `var`

```
var x = 42;
console.log(x); // zápis do terminálu

var y = 1;
console.log(x + y);

var z;
console.log(z); // nedeklarovaná proměnná "undefined"
```

- základní datové typy: číslo, řetězec, logický typ, `undefined`, `null`

Základy JS

- pole

```
var pole = [1, 2, 3, "42"];
console.log(pole[3]);

pole[3] = 4
console.log(pole); // [1, 2, 3, 4]

// délka pole
console.log(pole.length); // 4
```

- všechny běžně známé operátory

Větvení programu

```
var x = 42

if (x < 42) {
  console.log("x je menší než 42");
} else if (x > 42) {
  console.log("x je větší než 42");
} else {
  console.log("x je rovno 42");
}
```

Základní cykly

```
var pole = [1, 2, 3, 4, 5];

// for cyklus
for (var i = 0; i < pole.length; i++) {
  console.log(pole[i]);
}


// cyklus while
var i = 0
while (i < pole.length) {
  console.log(pole[i]);
    i = i + 1;
}
```

Funkce

```
var pole = [1, 2, 3, 4, 5]

function suma(pole) {
  var soucet = 0;

  for (var i = 0; i < pole.length; i++) {
    soucet += pole[i];
  }

  return soucet;
}

console.log(suma(pole)); // 15
```

- rozsah platnosti proměnných

Okno prohlížeče

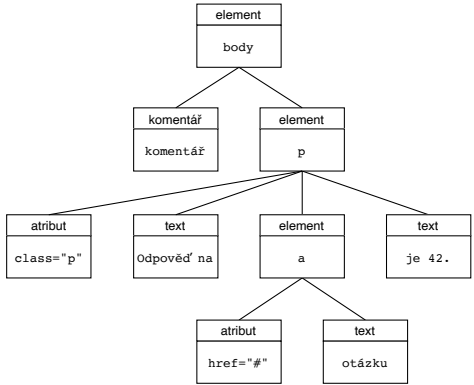
- objekt `window` → napojení na webový prohlížeč
- manipulace s oknem
  - spíše přežitek (např. popup okna)
  - některé funkce již nemají žádný smyl v moderních prohlížečích (například nastavení status baru)
  - jiné se občas hodí (např. zjištění velikosti okna)
- užitečné:
  - `window.alert()`
  - `window.confirm()`
  - `window.scrollTo()`
  - `window.history.back()`, `window.history.forward()`
  - `window.location` – URL, lze i `location`
  - `window.document` – webová stránka, lze i `document`

Webová stránka z pohledu JS

- Document Object Model (DOM)
- specifikace popisující strukturu dokumentu a operace s touto strukturou
- level 4 `http://www.w3.org/TR/dom/` (status doporučení)
- většina běžné funkcionality je podporována:
  `http://quirksmode.org/dom/core/`
- stromová struktura
- elementy HTML $\subseteq$ uzly DOM

Příklad: DOM

```
<body>
  <!-- komentář -->
  <p class="p">Odpověď na <a href="#">otázku</a>je 42.</p>
</body>
```



Výběr elementů v DOM

- základní metody pro výběr elementů
  - `document.getElementById()`
  - `document.getElementsByClassName()`
  - `document.getElementsByTagName()`
  - `document.querySelector()`
  - `document.querySelectorAll()`
- vrací element nebo pole elementů jinak `null`
- základní navigace (pouze elementy) v DOM
  - `.firstElementChild`, `.lastElementChild`, `.children[cislo_uzlu]`
  - `.parentElement`
  - `.nextElementSibling`, `.previousElementSibling`

Metody pro práci s DOM

- vytvoření uzlů
  - `document.createElement()`
  - `document.createAttribute()`
  - `document.createTextNode()`
- modifikace uzlů v DOM
  - `appendChild()`, `removeChild()`, `replaceChild()`, `insertBefore()`, ...
  - `getAtributte()`, `setAtributte()`, `removeAtributte()`
- vlastnosti elementů
  - `.innerHTML`
  - `.innerText`
  - `.style`
  - `.classList`
- ukázka: `https://codepen.io/trnecka/pen/XWpYoJp?editors=1010`

Události

- spuštění (části) skriptu na základě vyvolání události
- např. kliknutí myši (`onclick`), načtení dokumentu (`onload`), ...
- napojení:
  - funkce `addEventListener()`
  - atribut v HTML

```
<div class="tlacitko--A">tlačitko A</div>
<div class="tlacitko--B" onClick="show();">tlačitko B</div>
```

```
function show() {
  console.log("kliknuto");
}

document.querySelector(".tlacitko--A").addEventListener("click", show);
```

Časovače

- vykonání `funkce` po uplynutí časového kvanta
  - `setInterval(funkce, cas)`, periodické opakování
  - `setTimeout(funkce, cas)`, 1 opakování
- čas v milisekundách

```
// jedno opakování
setTimeout(function(){console.log("čas vypršel");}, 3000);

// periodické opakování
var casovac = setInterval(function(){console.log("čas vypršel");}, 3000);

// zastavení zastavení opakování
clearInterval(casovac);
```

(Naivní) Příklady

- hamburger navigace, `https://codepen.io/trnecka/pen/gOgKEJB`
- slideshow, `https://codepen.io/trnecka/pen/yLgXZjv`
- validace formuláře, `https://codepen.io/trnecka/pen/KKaeYOq`
- scroll-up ikona, `https://codepen.io/trnecka/pen/bGgKZLQ`

Ladění JavaScriptu

- console v Google Chrome
- celá řada dalších nástrojů a pluginů
- logování do konzole

```
var promenna = "konec se již blíží";

console.log(promenna);
console.info(promenna);
console.warn(promenna);
console.debug(promenna);
console.error(promenna);
```

Odbočka: Vypnutí JS

- vykonávání JS lze v prohlížeči vypnout → zásadně degraduje webové aplikace
- řada důvodů: chyby v JS, křivdy z dob minulých
- jediná možnost, tag `<noscript>`:

```
<script>
  document.write("Hello World!")
</script>

<noscript>
  Váš prohlížeč nepodporuje JavaScript!
</noscript>
```

Odbočka: JS knihovny a frameworky

- přináší: jednodušší práci s DOM, efekty, běžnou funkcionalitu, ...
- jQuery, Prototype, VanillaJS, DoJo, ...
- vyžadovány pro další knihovny
- obvykle velké $\rightarrow$ dnes odklon k minimalismu

Poslední slide o JS

- JS $\rightarrow$ rozšíření možností webového front-endu
- ukázali jsme pouze naivní možnosti JS
- masivní používání JS na webové stránce $\rightarrow$ ustupující trend (zejména kvůli responzivnímu designu)
- masivní používání JS pro tvorbu webových aplikací $\rightarrow$ vzestupný trend
- obecně CSS utlačuje JS
- stinná stránka JS: neuvážlivé používání = velká zátěž HW