Webové technológie 7

Aplikácie Webových technológií

Peter Bednár

Objekt `window`

- Poskytuje aplikačné rozhranie prehliadača pre jazyk JavaScript
- Reprezentuje aktuálne okno prehliadača
  - Ak stránka obsahuje rámce, prehliadač vytvorí objekt `window` pre stránku a pre jednotlivé rámce
- Sprístupňuje ďalšie objekty:
  - `window.document` – objektový model HTML dokumentu
  - `window.navigator` – informácie o prehliadači
  - `window.screen` – informácie o obrazovke zariadenia
  - `window.location` – URL zobrazeného dokumentu
  - `window.history` – prístup k histórii prehliadača
- K objektom je možné pristupovať aj priamo (názov `window.` netreba uvádzať)

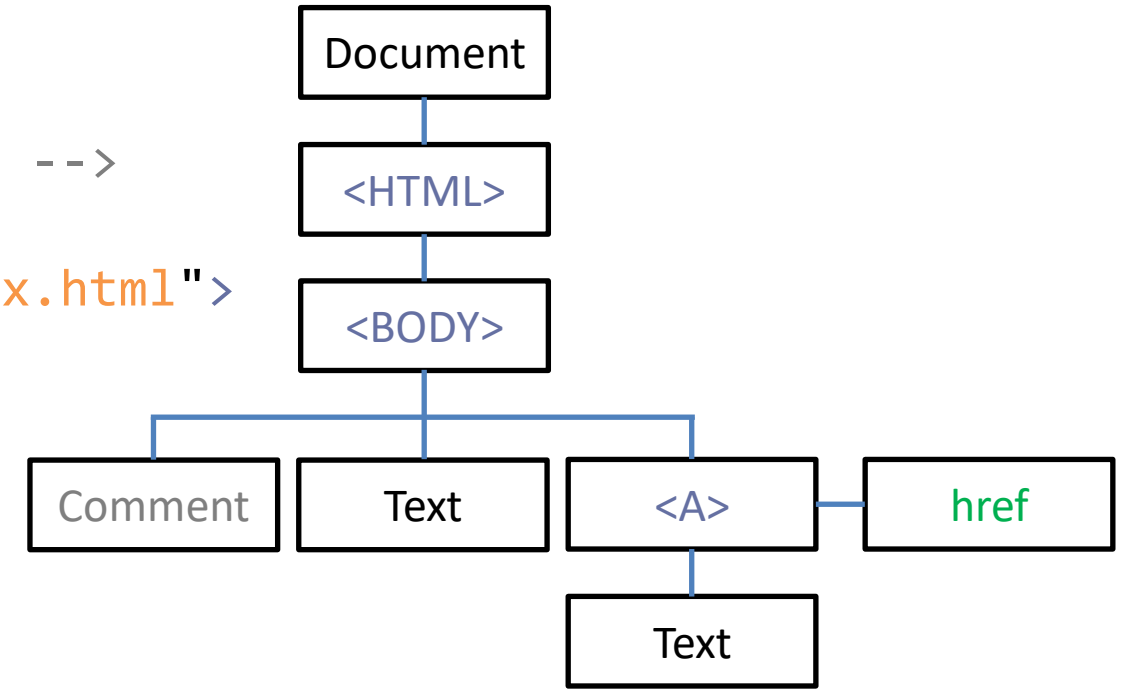
Document Object Model – DOM

- Štandardný objektový model pre reprezentáciu dokumentov značkovacích jazykov ako napr. HTML a XML
- Základné rozhranie, ktoré umožňuje programovo dynamicky meniť obsah HTML na stránke
- Nezávislý na programovacom jazyku – definovaný napr. pre JavaScript, Java, C#, atď.
- Poskytuje objekty pre reprezentáciu štruktúry HTML dokumentu:
  - Document
  - Element
  - Attribute
  - Text
  - Comment

DOM – Príklad

- HTML dokument je reprezentovaný ako stromová štruktúra
- Koreňový objekt je prístupný v premennej `window.document`

```
<html>
<body>
  <!-- komentár -->
  Ahoj,
  <a href="index.html">
    kliknite tu
  </a>
</body>
</html>
```



Prístup k elementom (1)

- Základné premenné objektu `document`:

- `document.documentElement` – odkaz na koreňový element `<html>`
- `document.body` – odkaz na obsah HTML (element `<body>`)
- `document.images` – kolekcia všetkých elementov obrázkov `<img>`
- `document.forms` – kolekcia všetkých formulárov (elementov `<form>`)
- `document.links` – kolekcia všetkých odkazov (elementov `<a>`)
- `document.scripts` – kolekcia všetkých skriptov na stránke (elementov `<script>`)

Prístup k elementom (2)

- `document.getElementById(id)` – vráti element, ktorý má priradené dané `id`, alebo `null` ak taký element na stránke neexistuje
- `document.getElementByTagName(typ)` – vráti kolekciu elementov daného typu
  - Napr. všetky elementy `<span>`: `document.getElementByTagName("SPAN")`
- `document.getElementByClassName(class)` – vráti kolekciu elementov patriacich do CSS triedy `class`
  - Je možné zadať zoznam tried oddelený medzerou, napr.: `document.getElementByClassName("info main")` vráti kolekciu elementov, ktoré patria do triedy info a main

Kolekcie elementov

- Elementy v kolekcii sú usporiadané v poradí podľa umiestnenia na stránke
- Metódy a premenné kolekcií:
  - `kolekcia.length` – počet prvkov v kolekcii
  - `kolekcia[index]` – prístup k elementom kolekcie (indexovanie od 0, napr. prvý formulár na stránke: `document.forms[0]`
  - `kolekcia.namedItem(id)` – prístup k elementom podľa id (parameter `id` je reťazec)
- Kolekcie sú dynamické, tzn. po ich získaní sa automaticky aktualizujú pri zmenách HTML
- Prevedenie kolekcie na pole JavaScript-u:

```
var pole = Array.from(kolekcia);
```

Vyhľadávanie elementov podľa CSS selektora

- `document.querySelector(selector)` – vráti prvý element ktorý vyhovuje zadanému CSS selektoru, alebo `null` ak taký neexistuje
- `document.querySelectorAll(selector)` – vráti zoznam všetkých elementov, ktoré vyhovujú zadanému CSS selektoru
  - Je možné zadať viacero selektorov oddelených čiarkou, napr.: `"h1, .info"`
  - Zoznam elementov je statický, tzn. na rozdiel od kolekcií nie je po získaní aktualizovaný pri zmene HTML
- Príklad:

```
var elms = document.querySelectorAll("div > p");
```

Prístup k elementom (3)

- Metódy `getElementByTagName`, `getElementByClassName`, `querySelector` a `querySelectorAll` sú definované aj pre elementy

- Príklad:

```
// vráti element s id = "info"
var elm = document.getElementById("info");
// vráti kolekciu všetkých vnorených elementov span
var spans = elm.getElementByTagName("SPAN");
for (var i = 0; i < spans.length; i++) {
    var span = spans[i];
    ...
}
```

Potomkovia, rodičia a súrodenci

- Nasledujúce vlastnosti objektu elementu sú iba na čítanie, a ak daný element neexistuje, majú hodnotu `null`

- `elm.parentElement` – rodič elementu `elm`
- `elm.childElementCount` – počet priamych potomkov
- `elm.children` – kolekcia priamych potomkov
- `elm.firstElementChild` – prvý priamy potomok
- `elm.lastElementChild` – posledný priamych potomkov
- `elm.nextSibling` – nasledujúci súrodenec
- `elm.previousSibling` – predchádzajúci súrodenec
  - súrodenec `elm` = element, ktorý je priamym potomkom toho istého uzla ako `elm`

Obsah elementov

- `elm.innerHTML` – reťazec s celým obsahom elementu
- `elm.textContent` – reťazec s textovým obsahom bez HTML značiek (vnorený obsah je zreťazený do jedného reťazca)

- Obe vlastnosti sú na čítanie a zápis a je možné pomocou nich dynamicky meniť obsah HTML stránky, napr.:

```
var elm = document.getElementById("info");
elm.innerHTML = "Zmenený <span>obsah</span>";
var text = elm.textContent; // = "Zmenený obsah"
// DOM je automaticky aktualizovaný podľa
// zmeneného obsahu
var span = elm.firstElementChild;
```

Zmena atribútov elementu

- `elm.getAttribute(attr)` – vráti reťazec s hodnotou atribútu `attr` elementu `elm`, ak hodnota neexistuje, vráti `null` alebo prázdny reťazec (pre staršie prehliadače)
- `elm.hasAttribute(attr)` – vráti `true` ak element `elm` má atribút `attr`, inak `false`
- `elm.setAttribute(attr, val)` – nastaví hodnotu atribútu `attr` elementu `elm` na hodnotu `val`
- `elm.removeAttribute(attr)` – odstráni z elementu `elm` atribút `attr`

- Niektoré hlavné atribúty sú priamo prístupné ako vlastnosti objektu elementu, napr. `elm.id` a `elm.value` pre prvky formulárov

Zmena CSS

- Hodnoty CSS vlastností sú priamo prístupné vo vlastnosti `style` objektu elementu a je možné ich priamo meniť, napr.:

```
var elm = document.getElementById("info");
elm.style.color = "red";
```

- `window.getComputedStyle(elm)` – vráti objekt s vypočítanými hodnotami CSS vlastností pre zadaný element tak ako sú nastavené prehliadačom (vypočítané hodnoty sa môžu líšiť od deklarovaných, napr. v jednotkách)

Vytvorenie, vloženie a zmazanie elementu

- `document.createElement(typ)` – vytvorí objekt reprezentujúci element daného typu, napr. pre `<span>`:

```
var span = document.createElement("SPAN");
```

- `elm.appendChild(child)` – vloží element `child` ako posledného potomka elementu `elm`
- `parent.insertBefore(newElm, elm)` – vloží element `newElm` ako potomok elementu `parent` pred element `elm` (`elm` a `newElm` budú súrodenci)
- `elm.removeChild(child)` – odstráni potomka `child` elementu `elm`

DOM – prehľad API (1)

- Prístup k elementom
  - `documentElement`, `body`, `images`, `forms`, `links`, `scripts`
  - `getElementById()`, `getElementByTagName()`, `getElementByClassName()`, `querySelector()`, `querySelectorAll()`
  - `parentElement`, `children`, `firstElementChild`, `lastElementChild`, `nextSibling`, `previousSibling`
- Obsah elementov
  - `innerHTML`, `textContent`

DOM – prehľad API (2)

- Zmena atribútov
  - `getAttribute()`, `hasAttribute()`, `setAttribute()`, `removeAttribute()`
  - `style`, `getComputedStyle()`
  - `id`, `value`
- Vytvorenie, vloženie a zmazanie elementov
  - `createElement()`, `appendChild()`, `insertBefore()`, `querySelector()`, `removeChild()`

Udalosti

- Udalosti umožňujú reagovať na akcie používateľa (napr. po kliknutí myšou a pod.), alebo zmeny v prehliadači (napr. pri zmene veľkosti okna)
- Rozhranie prehliadača umožňuje zaregistrovať funkciu, ktorá sa zavolá pri výskyte udalosti, funkcia potom vykoná požadovanú akciu

- Zoznam podporovaných udalostí:
  - https://developer.mozilla.org/en-US/docs/Web/Events

Priradenie funkcie pre obsluhu udalosti (1)

- Priamo v HTML kóde
  - Hlavné udalosti sa dajú priradiť pomocou atribútov elementu v tvare on[typ udalosti], napr. pre kliknutie myši:

```
<div onclick="onClick();">
    Kliknite tu
</div>
...
<script>
    function onClick() {
        // obsluha udalosti
    }
</script>
```

Funkciu pre obsluhu udalosti je potrebné definovať v skripte načítaného na stránke

Priradenie funkcie pre obsluhu udalosti (2)

- V JavaScript kóde
  - Metóda `elm.addEventListener(typ, f)`
  - Preferovaný spôsob priradenia, jedna funkcia `f` môže byť priradená obsluhe viacerým elementom, alebo typom udalostí
- Príklad:

```
function onClick(e) {
    // obsluha udalosti
}
var elm = document.getElementById("nadpis1");
elm.addEventListener("click", onClick);
```

- Zrušenie obsluhy:
  - `elm.removeListener(typ)`

Obsluha udalostí (1)

- Obsluha udalosti je funkcia s jedným parametrom typu Event
- Najdôležitejšie vlastnosti objektu Event:
  - `type` – typ udalosti (napr. "click")
  - `target` – odkaz na objekt elementu nad ktorým bola udalosť vyvolaná

- Príklad:

```
function onClick(e) {
    if (e.target.id == "nadpis1") {
        // používateľ klikol na element s ID "nadpis1"
    }
}
```

Obsluha udalostí (2)

- Väčšina typov udalostí sa šíri od vnorených elementov na nadradené, napr.:

```
<div onclick="onClick();">
  Text <p>Odstavec</p>
</div>
```

  - onClick() sa zavolá aj keď sa klikne na odstavec

```
<div onclick="onClick1();">
  <div onclick="onClick2();">
    <p onclick="onClick3();>Odstavec</p>
  </div>
</div>
```

  - Funkcie sa volajú v poradí onClick3, onClick2, onClick1

Prehľad typov udalostí (1)

- `click` – kliknutie myšou na elemente
- `dblclick` – dvojkliknutie
- `touchstart` – stlačenie s miestom dotyku na elemente
- `touchend` – uvoľnenie s miestom dotyku na elemente
- `pointerdown`, `pointerenter`, `pointerleave`, atď. – zjednotené rozhranie pre dotykové zariadenie a myši

- `keydown` – stlačenie klávesu na aktívnom elemente
- `keyup` – uvoľnenie klávesu

Prehľad typov udalostí (2)

- `load` – po načítaní celého dokumentu
- `focus` – po získaní focusu na elemente (pre formulárové prvky)
- `resize` – po zmene veľkosti okna prehliadača (viewportu)
- `scroll` – po skrolovaní obsahu okna

Dialógové okná

- `window.alert("správa")`
  - Zobrazí jednoduché okno so správou a tlačidlom "OK"
- `window.confirm("správa")`
  - Zobrazí jednoduché okno so správou a tlačidlami "OK", a "Zrušiť",
  - Vracia `true` ak používateľ stlačil "OK" a `false` ak okno uzavrel, alebo stačil "Zrušiť"
- `window.prompt("správa", "prednastavená odpoveď")`
  - Zobrazí jednoduché okno so správou, editovacím poľom a tlačidlami "OK", a "Zrušiť",
  - Vracia zadaný reťazec ak používateľ stlačil "OK", alebo `null` ak okno uzavrel, alebo stačil "Zrušiť"
- Po zobrazení okna je vykonávanie skriptu pozastavené až kým ho používateľ neuzavrie

Výpis do konzoly prehliadača

- Konzola je prístupná cez Vývojárske nástroje
- `window.concole.log("správa")`
  - Zapíše správu do konzoly prehliadača
- `window.concole.dir(obj)`
  - Zobrazí štruktúru JavaScript objektu, ak je `obj` objekt HTML elementu, zobrazí jeho DOM štruktúru

Objekt `window.navigator`

- Objekt reprezentujúci prehliadač a softvérové prostredia v ktorom je spustený
- Základné vlastnosti a metódy
  - `navigator.appName` – názov prehliadača
  - `navigator.appVersion` – verzia prehliadača
  - `navigator.language` – preferovaný jazyk (podľa nastavenia používateľa)
  - `navigator.geolocation.getCurrentPosition()` – vráti aktuálnu polohu zariadenia

Objekt `window.location`

- Objekt reprezentujúci URL aktuálne zobrazenej stránky
- Základné vlastnosti a metódy
  - `location.href` – celá adresa URL
  - `location.protocol`, `host`, `hostname`, `port`, `pathname`, `search` – jednotlivé časti
  - `location.assign(url)` – v okne (resp. na karte) sa zobrazí dokument s danou URL adresou
  - `location.reload()` – znova načíta aktuálny dokument

Objekt `window.history`

- Objekt pre prístup k histórii prehliadača
- Základné vlastnosti a metódy
  - `history.length` – počet odkazov v histórii prehliadača
  - `history.back()` – navigácia ako pri stlačení tlačidla "Dozadu" v prehliadači
  - `history.forward()` – navigácia ako pri stlačení tlačidla "Dopredu" v prehliadači

Objekt `window.screen`

- Objekt reprezentujúci obrazovku fyzického zariadenia
- Základné vlastnosti a metódy
  - `screen.width` – rozlíšenie obrazovky vodorovne v pixeloch
  - `screen.height` – rozlíšenie obrazovky zvislo v pixeloch
  - `screen.colorDepth` – farebné rozlíšenie obrazovky (v bitoch)
  - `screen.orientation` – orientácia obrazovky (hodnoty: `"portrait-primary"`, `"landscape-primary"`, `"portrait-secondary"`, `"landscape-secondary"`)
- Pre ošetrenie zmeny orientácie je možné na objekte `screen` nastaviť obsluhu udalosti `orientationchange`

Vývojárske nástroje v prehliadači (1)

- Integrované v hlavných prehliadačov
  - Chrome – menu *Tools*/*Developer Tools*
  - Explorer – menu *Tools*/*Developer Tools*
  - Safari – povoliť vývojárske nástroje v *Preferences*/*Advanced*, menu *Develop*
- DOM prehliadač
  - Štruktúra HTML (aj dynamicky generovaných elementov), CSS vlastnosti
- JavaScript konzola
  - Logovanie

Vývojárske nástroje v prehliadači (2)

- JavaScript debugger
  - Zobrazenie zdrojového kódu skriptov, nastavenie bodov prerušenia (breakpoint),inšpekcia hodnôt premenných
- Komunikácia na sieti
  - Sťahované zdroje (HTML stránka, CSS súbory, skripty, obrázky, atď.), odosielané a prijímané dáta