20. Dynamické webové stránky

Obsah

Dynamické webové stránky umožňují interaktivitu a personalizaci obsahu na základě akcí uživatele, na rozdíl od statických stránek, které zobrazují vždy stejný obsah.

Skriptovací jazyky pro webové aplikace

HTML a CSS slouží pouze k definici struktury a vzhledu webových stránek, ale neumožňují přidání funkcionality. Pro vytvoření dynamického obsahu potřebujeme skriptovací jazyky, které se dělí na:

Frontend (klientská strana)

Kód se spouští v prohlížeči uživatele:

- **JavaScript** - Nejrozšířenější skriptovací jazyk pro webové prohlížeče
- **TypeScript** - Nadstavba JavaScriptu s typovou kontrolou
- **WebAssembly** - Nízkoúrovňový bytekód, který umožňuje spouštět kód napsaný v C/C++, Rust a dalších jazycích v prohlížeči

Backend (serverová strana)

Kód se spouští na serveru:

- **PHP** - Specializovaný jazyk pro webové aplikace
- **Node.js** - JavaScript na straně serveru
- **Python** (Django, Flask) - Univerzální jazyk s mnoha frameworky
- **Ruby** (Ruby on Rails) - Dynamický, objektově orientovaný jazyk
- **Java** (Spring, JSP) - Silně typovaný, objektově orientovaný jazyk
- **C#** (ASP.NET) - Objektově orientovaný jazyk od Microsoftu
- **Go** - Moderní jazyk vytvořený Googlem

Vložení skriptu do HTML

JavaScript lze do HTML dokumentu vložit několika způsoby:

1. Externí soubor

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

Výhody: - Oddělení obsahu od funkcionality - Možnost cachování skriptu prohlížečem - Lepší údržba a přehlednost kódu

2. Inline skript

```
<script type="text/javascript">
    function pozdrav() {
        alert("Ahoj světe!");
    }
</script>
```

Výhody: - Nemusí se stahovat další soubor - Vhodné pro krátké skripty specifické pro danou stránku

3. Inline atribut události

```
<button onclick="alert('Ahoj!')">Klikni na mě</button>
```

Tato metoda se však nedoporučuje, protože míchá HTML s JavaScriptem.

Aktivace skriptu

Skripty se obvykle spouští při načtení stránky nebo jako reakce na události:

Načítání skriptů

- **Standardní načítání** - Skripty v hlavičce jsou stahovány a spouštěny v pořadí, v jakém jsou uvedeny, což může blokovat vykreslování stránky
- **Atribut** `async` - Umožňuje asynchronní stahování skriptu, který se spustí ihned po stažení

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

- **Atribut** `defer` - Stáhne skript asynchronně, ale spustí ho až po načtení celého HTML dokumentu

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

Události (Event Listeners)

```
// Moderní přístup
document.getElementById("tlacitko").addEventListener("click", function() {
    alert("Tlačítko bylo stisknuto!");
});

// Po načtení DOMu
document.addEventListener("DOMContentLoaded", function() {
    console.log("DOM je připraven!");
});

// Po kompletním načtení stránky včetně všech zdrojů
window.addEventListener("load", function() {
    console.log("Stránka je plně načtena!");
});
```

Interaktivní uživatelské rozhraní

Dynamické webové stránky umožňují interaktivní rozhraní pomocí několika technik:

Manipulace s DOM (Document Object Model)

```
// Změna obsahu elementu
document.getElementById("demo").innerHTML = "Nový text";

// Změna stylu
document.getElementById("demo").style.color = "red";
```

```javascript
// Vytvoření nového elementu
var novyElement = document.createElement("p");
novyElement.textContent = "Nový odstavec";
document.body.appendChild(novyElement);
```

AJAX (Asynchronous JavaScript and XML)

Umožňuje asynchronní komunikaci se serverem bez nutnosti přenačíst celou stránku:

```javascript
// Pomocí moderního Fetch API
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Chyba:', error));
```

Animace a vizuální efekty

```javascript
// Jednoduchá animace pomocí čistého JavaScriptu
function animace() {
    const element = document.getElementById("animovany");
    let pozice = 0;
    const id = setInterval(frame, 10);

    function frame() {
        if (pozice == 350) {
            clearInterval(id);
        } else {
            pozice++;
            element.style.left = pozice + "px";
        }
    }
}
```

Datové typy a správa paměti

JavaScript - dynamické typování

JavaScript používá dynamické typování, což znamená, že datový typ proměnné se určuje za běhu:

```javascript
// Deklarace proměnných
var staraProměnná = 10;        // Funkční rozsah
let modernějšíProměnná = 20;   // Blokový rozsah
const konstantníProměnná = 30; // Konstantní hodnota s blokovým rozsahem

// Datové typy
let číslo = 42;              // Number
let text = "Hello";          // String
let pravda = true;           // Boolean
let nic = null;              // Null
let nedefinováno = undefined;// Undefined
let objekt = {jméno: "Jan"}; // Object
let pole = [1, 2, 3];        // Array (speciální druh objektu)
let funkce = function() {};  // Function (speciální druh objektu)
let symbol = Symbol("id");   // Symbol (ES6)
let velké = BigInt(9007199254740991n); // BigInt (ES2020)
```

Správa paměti

JavaScript používá automatickou správu paměti (garbage collection):

- Objekty jsou udržovány v paměti, dokud na ně existuje reference
- Garbage collector automaticky uvolňuje paměť nedostupných objektů
- Problémy mohou nastat při cyklických referencích nebo nevyčištěných event listenerech

Knihovny a frameworky

Knihovny a frameworky usnadňují vývoj dynamických webových aplikací:

Frontend (JavaScript)

- **jQuery** - Knihovna usnadňující manipulaci s DOM, AJAX a animace
- **React** - Knihovna pro tvorbu uživatelských rozhraní pomocí komponent
- **Vue.js** - Progresivní JavaScript framework pro tvorbu UI
- **Angular** - Komplexní framework od Googlu
- **Svelte** - Moderní přístup ke kompilaci komponentů

Backend

- **Express.js** (Node.js) - Minimalistický webový framework
- **Laravel** (PHP) - Plnohodnotný MVC framework
- **Django** (Python) - "Batteries included" framework
- **Ruby on Rails** (Ruby) - Konvence před konfigurací
- **Spring Boot** (Java) - Framework pro enterprise aplikace

Způsoby importu knihoven

Frontend:

```
<!-- CDN (Content Delivery Network) -->
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>

<!-- Lokálně uložený soubor -->
<script src="js/vue.js"></script>

<!-- ES6 moduly -->
<script type="module">
  import Vue from './js/vue.js';
</script>
```

Backend (Node.js):

```
// CommonJS
const express = require('express');

// ES6 moduly
import express from 'express';
```

PHP:

```
// Vložení souboru
include('knihovna.php');
require('důležitá-knihovna.php'); // Vyvolá chybu, pokud soubor neexistuje

// Autoloading (PSR-4, Composer)
use App\Knihovna\TriduKnihovny;
```

Generování dynamického obsahu

Serverové generování (SSR - Server-Side Rendering)

Obsah stránky je generován na serveru před odesláním klientovi:

```php
<!-- PHP příklad -->
<!DOCTYPE html>
<html>
<head>
    <title>Dynamická stránka</title>
</head>
<body>
    <h1>Seznam uživatelů</h1>
    <ul>
    <?php
        // Připojení k databázi
        $db = new PDO('mysql:host=localhost;dbname=testdb', 'username', 'password');
        $uzivatele = $db->query('SELECT * FROM uzivatele');

        // Dynamické generování obsahu
        foreach ($uzivatele as $uzivatel) {
            echo "<li>" . htmlspecialchars($uzivatel['jmeno']) . "</li>";
        }
    ?>
    </ul>
</body>
</html>
```

Klientské generování (CSR - Client-Side Rendering)

Obsah je generován JavaScriptem přímo v prohlížeči:

```javascript
// Načtení dat a jejich zobrazení
fetch('/api/uzivatele')
  .then(response => response.json())
  .then(uzivatele => {
    const seznam = document.getElementById('seznamUzivatelu');
    uzivatele.forEach(uzivatel => {
      const polozka = document.createElement('li');
      polozka.textContent = uzivatel.jmeno;
      seznam.appendChild(polozka);
    });
  });
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Dynamická stránka (klientská)</title>
    <script src="script.js" defer></script>
</head>
<body>
    <h1>Seznam uživatelů</h1>
    <ul id="seznamUzivatelu">
        <!-- Obsah bude vygenerován JavaScriptem -->
    </ul>
</body>
</html>
```

Hybridní přístupy

- **Statická stránka s hydrogenací (SSG + hydrogenace)** - Staticky generované stránky s klientskou interaktivitou
- **Inkrementální statická regenerace (ISR)** - Statické stránky s periodickou regenerací

- **Serverové komponenty** - Kombinace serverových a klientských komponent (např. React Server Components)

Zpracování dat formuláře

HTML formulář

```
<form id="kontaktFormular" action="/zpracovat.php" method="POST">
    <label for="jmeno">Jméno:</label>
    <input type="text" id="jmeno" name="jmeno" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <label for="zprava">Zpráva:</label>
    <textarea id="zprava" name="zprava"></textarea>

    <button type="submit">Odeslat</button>
</form>
```

Zpracování na serveru (PHP)

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Sanitizace vstupu
    $jmeno = htmlspecialchars($_POST["jmeno"]);
    $email = filter_var($_POST["email"], FILTER_SANITIZE_EMAIL);
    $zprava = htmlspecialchars($_POST["zprava"]);

    // Validace
    if (empty($jmeno) || empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Chyba: Neplatný vstup!";
        exit;
    }

    // Zpracování dat (např. uložení do databáze)
    $db = new PDO('mysql:host=localhost;dbname=testdb', 'username', 'password');
    $stmt = $db->prepare('INSERT INTO zpravy (jmeno, email, zprava) VALUES (?, ?, ?)');
    $stmt->execute([$jmeno, $email, $zprava]);

    // Odpověď
    echo "Děkujeme za vaši zprávu!";
}
?>
```

Zpracování na klientovi (JavaScript)

```
document.getElementById('kontaktFormular').addEventListener('submit', function(event) {
    event.preventDefault(); // Zabránění standardnímu odeslání formuláře

    // Získání dat z formuláře
    const formData = new FormData(this);

    // Odeslání dat pomocí AJAX
    fetch('/zpracovat', {
        method: 'POST',
        body: formData
    })
```

```
    .then(response => response.json())
    .then(data => {
        alert(data.message); // Zobrazení odpovědi
    })
    .catch(error => {
        console.error('Chyba:', error);
    });
});
```

Bezpečnostní aspekty

- **Validace vstupu** - kontrola formátu a obsahu vstupních dat
- **Sanitizace vstupu** - odstranění potenciálně nebezpečných znaků
- **Cross-Site Scripting (XSS)** - prevence vkládání škodlivých skriptů
- **Cross-Site Request Forgery (CSRF)** - ochrana pomocí tokenů
- **SQL Injection** - používání prepared statements
- **CAPTCHA** - ochrana proti automatizovaným botům

Relace (Sessions)

Relace umožňují uchovávat stav mezi jednotlivými požadavky, což je důležité pro funkce jako přihlášení uživatelů, nákupní košíky atd.

Cookies

Malé textové soubory uložené v prohlížeči:

```
// JavaScript - Nastavení cookie
document.cookie = "uzivatel=jnovak; expires=Thu, 18 Dec 2023 12:00:00 UTC; path=/";

// JavaScript - Čtení cookie
let cookies = document.cookie;
```

```
// PHP - Nastavení cookie
setcookie("uzivatel", "jnovak", time() + 3600, "/");

// PHP - Čtení cookie
$uzivatel = $_COOKIE["uzivatel"];
```

Sessions (Relace)

Serverové úložiště spojené s klientem pomocí cookie:

```
// PHP - Zahájení relace
session_start();

// PHP - Uložení dat
$_SESSION["uzivatel_id"] = 123;
$_SESSION["uzivatel_jmeno"] = "Jan Novák";

// PHP - Čtení dat
echo "Přihlášen: " . $_SESSION["uzivatel_jmeno"];

// PHP - Zničení relace (odhlášení)
session_destroy();
```

Web Storage API (localStorage, sessionStorage)

Modernější alternativy k cookies pro ukládání dat na straně klienta:

```
// localStorage - trvalé úložiště
localStorage.setItem("uzivatel", "jnovak");
let uzivatel = localStorage.getItem("uzivatel");
localStorage.removeItem("uzivatel");

// sessionStorage - dočasné úložiště (do zavření prohlížeče)
sessionStorage.setItem("kosik", JSON.stringify({produkty: [1, 2, 3]}));
let kosik = JSON.parse(sessionStorage.getItem("kosik"));
```

JWT (JSON Web Tokens)

Moderní způsob autentizace a předávání dat mezi klientem a serverem:

```
// Příjem tokenu po autentizaci
fetch('/login', {
    method: 'POST',
    body: JSON.stringify({username: 'jnovak', password: 'heslo'})
})
.then(response => response.json())
.then(data => {
    // Uložení tokenu
    localStorage.setItem('jwt_token', data.token);
})

// Použití tokenu pro autentizované požadavky
fetch('/chraneny-zdroj', {
    headers: {
        'Authorization': 'Bearer ' + localStorage.getItem('jwt_token')
    }
})
```

Uživatelské přizpůsobení

Dynamické webové stránky umožňují personalizaci obsahu podle preferencí uživatele:

Personalizace obsahu

```
// Načtení preferencí uživatele
let userPreferences = JSON.parse(localStorage.getItem('preferences')) || {
    theme: 'light',
    fontSize: 'medium',
    language: 'cs'
};

// Aplikace preferencí
document.body.classList.add(`theme-${userPreferences.theme}`);
document.body.classList.add(`font-${userPreferences.fontSize}`);
document.documentElement.lang = userPreferences.language;

// Změna preferenčního nastavení
function changeTheme(theme) {
    document.body.classList.remove(`theme-${userPreferences.theme}`);
    userPreferences.theme = theme;
    document.body.classList.add(`theme-${theme}`);
    localStorage.setItem('preferences', JSON.stringify(userPreferences));
}
```

Responsivní design a adaptivní obsah

```
/* CSS Media Queries pro různá zařízení */
@media (max-width: 768px) {
    .desktop-only {
        display: none;
    }
    .mobile-friendly {
        font-size: 14px;
    }
}
```

```
// Detekce zařízení a přizpůsobení obsahu
if (window.innerWidth < 768) {
    // Mobilní zobrazení
    document.getElementById('menu').classList.add('hamburger');
} else {
    // Desktopové zobrazení
    document.getElementById('menu').classList.add('horizontal');
}

// Detekce preferencí uživatele (dark mode)
if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {
    document.body.classList.add('dark-theme');
}
```

Moderní trendy v dynamických webových stránkách

Single-Page Applications (SPA)

Jednostránkové aplikace, kde se obsah mění bez nutnosti přenačtení celé stránky:

```
// Jednoduchý router pro SPA
window.addEventListener('hashchange', function() {
    const hash = window.location.hash.substring(1);
    showPage(hash);
});

function showPage(page) {
    // Skrytí všech stránek
    document.querySelectorAll('.page').forEach(p => p.style.display = 'none');

    // Zobrazení požadované stránky
    const activePage = document.getElementById(page) || document.getElementById('home');
    activePage.style.display = 'block';
}
```

Progressive Web Apps (PWA)

Webové aplikace s funkcionalitou nativních aplikací:

```
// Registrace service workeru pro offline funkcionalitu
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js')
        .then(registration => {
            console.log('Service Worker registrován:', registration);
        })
        .catch(error => {
            console.error('Chyba při registraci Service Workeru:', error);
```

```
        });
}
```

Serverové API a mikroslužby

```
// REST API v Express.js (Node.js)
const express = require('express');
const app = express();
app.use(express.json());

// Definice koncových bodů (endpoints)
app.get('/api/uzivatele', (req, res) => {
    // Získání seznamu uživatelů z databáze
    res.json([{id: 1, jmeno: 'Jan'}, {id: 2, jmeno: 'Eva'}]);
});

app.post('/api/uzivatele', (req, res) => {
    // Uložení nového uživatele
    console.log(req.body);
    res.status(201).json({id: 3, ...req.body});
});

app.listen(3000, () => console.log('Server běží na portu 3000'));
```
