

Offline; HTML značky <svg> a <audio> (9)

Offline + Cache Manifest + Service Worker

Offline API

- webová aplikace má k dispozici JS API, které informuje o stavu připojení
 - `navigator.onLine`
 - `window.addEventListener("online", ...)`
 - `window.addEventListener("offline", ...)`

App Cache Manifest

- webová aplikace poskytuje tzv. **offline manifest**
 - o tj. výčet souborů pro použití offline... `<html manifest="mysite.manifest">`
- pokud je uživatel **offline** a pokusí se přistoupit na adresu webové aplikace, prohlížeč mu poskytne soubory definované v manifestu
- pokud je uživatel **online** a nezměnil se čas modifikace manifestu, opět se použije cache verze
- **při změně** souborů (a manifestu) jsou data jen stažena a příprava k dalšímu použití

```
1. CACHE MANIFEST
2.
3. # soubory určené k použití offline
4. offline/index.html
5. offline/style.css
6. offline/code.js
7.
8. # tyto soubory se nikdy nesmí cachovat
9. NETWORK:
10. remote.cgi
11.
12. # tyto soubory, pokud nedostupné, budou nahrazeny jinými
13. FALLBACK:
14. images/large offline/sorry.jpg
```

- ukázka: <http://ondras.zarovi.cz/demos/morgan/>
- manifest: <http://ondras.zarovi.cz/demos/morgan/morgan.manifest>

applicationCache API

- JS API pro práci s offline cache
- `window.applicationCache` – objekt popisující stav offline cache, též možnost explicitně cache obnovit
- `if (applicationCache.status == applicationCache.UPDATEREADY) { ... }`
- `applicationCache.update()` – pokyn ke kontrole novější verze (jako při reloadu)
- `applicationCache.swapCache()` – přepnutí na aktuální verzi souborů (nutný reload)

Service Worker

- modernější alternativa k AppCache
- konfigurace (AppCache) vs. logika (Service Worker)
- slabší podpora
- **Jak funguje?**
 - o je to skript dodaný autorem stránky
 - o prohlížeč jej vykonává bokem, asynchronně, nezávisle na stránce, pro kterou pracuje
 - o SW může manipulovat s HTTP požadavky stránky, kterou obsluhuje
 - o nemá k dispozici DOM
 - o lze použít jen na https stránkách
- **další využití:**
 - o synchronizace dat na pozadí
 - o cachování
 - o preprocessing zdrojů na klientu
 - o šablonování
 - o pre-fetch
- **ukázka:**

```
1. navigator.serviceWorker.register("/worker.js").then(function(reg) {  
2.     console.log("Service worker ready");  
3. });
```

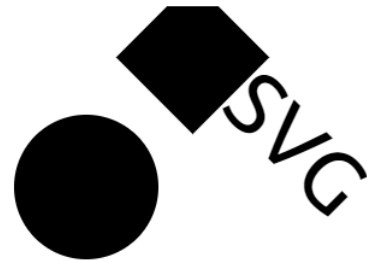
Obrázek 1 - Řízení asynchronního toku kódu je realizováno pomocí vzoru Promise

```
1. self.addEventListener("install", function(event) {  
2.     console.log("SW (či jeho nová verze) nainstalován");  
3. });  
4.  
5. self.addEventListener("activate", function(event) {  
6.     console.log("SW (či jeho nová verze) aktivován");  
7. });  
8.  
9. self.addEventListener("fetch", function(event) {  
10.     console.log("HTTP požadavek ze stránky");  
11.     event.respondWith(new Response("Hello world!"));  
12. });
```

Scalable Vector Graphics (SVG)

- všechny prohlížeče krom IE <= 8
- prehistorický standard z roku 2001
- rozšířený pro vektorovou grafiku
- DOM, události, stylování
- horší výkon v porovnání s <canvas>
- **ukázka (hodiny):** <http://seznam.github.io/CVUT/KAJ/prednasky/09/clock/>
- **ukázka (mapa):** <http://ondras.zarovi.cz/demos/hex/?f=cr.emap>

```
1. <svg width="500" height="300" viewBox="0 0 100 100">
2.   <circle cx="20" cy="50" r="20" />
3.   <g transform="rotate(45) translate(30, -30)">
4.     <rect x="0" y="-10" width="30" height="30" />
5.     <text x="30" y="10">SVG</text>
6.   </g>
7. </svg>
```



SVG je dialekt XML

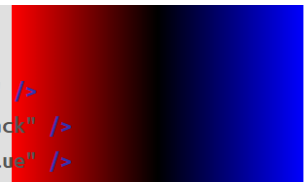
- míchání HTML a SVG:

| | |
|------------------|--|
| HTML4 | <embed>, <object>, <iframe> |
| XHTML (text/xml) | <svg xmlns="http://www.w3.org/2000/svg"> |
| HTML5 | Dedikovaná značka <svg> |
| JS | document.createElementNS() |

- **stavební kameny SVG:**
 - o <svg>, <g> (group)
 - o <rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon>
 - o <path d="M100,200 C100,100 250,100 250,200">
 - o barevné přechody (gradient), vzory (pattern), značky (marker)
- **SVG <path> detailněji:**
 - o **M x y** – posun na absolutní pozici
 - o **m dx dy** – posun o relativní pozici
 - o **L x y (l dx dy)** – rovná čára
 - o **Q x1 y1, x y** – kvadratická Bézierova křivka
 - o **C x1 y1, x2 y2, x y** – kubická Bézierova křivka
 - o **A rx ry x-axis-rotation large-arc-flag sweep-flag x y** – eliptická křivka
 - o **Z** – spojit se začátkem

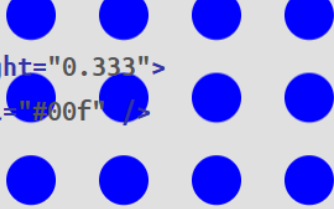
- **SVG barevné přechody:**

```
1. <defs>
2.   <linearGradient id="gradient">
3.     <stop offset="0%" stop-color="red" />
4.     <stop offset="50%" stop-color="black" />
5.     <stop offset="100%" stop-color="blue" />
6.   </linearGradient>
7. </defs>
8. <rect fill="url(#gradient)" x="0" y="0" width="100%" height="100%" />
```



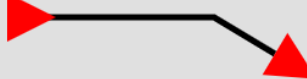
- SVG <pattern>:

```
1. <defs>
2.   <pattern id="pattern" width="0.25" height="0.333">
3.     <circle cx="25" cy="25" r="20" fill="#00f" />
4.   </pattern>
5. </defs>
6. <rect fill="url(#pattern)" x="0" y="0" width="100%" height="100%" />
```



- SVG <marker>:

```
1. <defs>
2.   <marker
3.     id="marker" markerWidth="13" markerHeight="13"
4.     refX="2" refY="6" orient="auto">
5.     <path d="M2,2 L2,10 L10,6 Z" fill="red" />
6.   </marker>
7. </defs>
8. <path d="M30,20 L200,20 L250,50"
9.   marker-start="url(#marker)"
10.  marker-end="url(#marker)" />
```



- stylování SVG:

- o pomocí atributů (stroke, color, opacity,...)
- o pomocí CSS

```
1. circle {
2.   fill: yellow; stroke: red;
3. }
4. rect {
5.   fill: none; stroke: black;
6.   stroke-width: 2;
7. }
8. text {
9.   text-anchor: middle; font-size: 10px;
10. }
```

- SVG a Javascript:

- o práce stejně jako s DOM stránky
- o tvorba prvků přes **document.createElementNS("http://www.w3.org/2000/svg", "...")**
- o změna hodnot pomocí **setAttribute**
- o běžná práce s událostmi pomocí **addEventListener**

- in-line SVG:

- o je možné mixovat přímo s HTML5
- o v mnoha ohledech flexibilnější než ``
- o styly sdílené s dokumentem stránky, ideální pro (barvené) ikonky

HTML5 <audio>



- vložení zvukového souboru do stránky
- ovládací prvky vyrábí prohlížeč (jsou-li požadovány)
- různé prohlížeče rozumí různým formátům zvukových souborů
- možnost specifikovat více formátů
- možnost zadat alternativní obsah pro nekompatibilní prohlížeče

- ukázka: <http://seznam.github.io/CVUT/KAJ/prednasky/09/#25>

```
1. <audio controls>
2.   <source src="sound.ogg" type="audio/ogg" />
3.   <source src="sound.mp3" type="audio/mpeg" />
4.   <source src="sound.wav" type="audio/wave" />
5.   Smůla :-(
6. </audio>
```

- javascriptové API:
`new Audio("song.mp3").play();`

```
1. var a = new Audio();
2. a.canPlayType("audio/mpeg"); /* "", "maybe", "probably" */
3. a.src = "song.mp3";
4. a.addEventListener("timeupdate", function() { console.log(a.currentTime)
5. a.play();
```



Web Audio API

- poměrně moderní JS API
- konstrukce zvukového grafu (
- uzly = transformace zvuku
- přístup k právě přehrávaným datům (vizualizace)

- generování zvukového signálu:

```
1. var ctx = new AudioContext();
2.
3. var oscillator = ctx.createOscillator();
4.
5. oscillator.frequency.value = 440;
6. oscillator.connect(ctx.destination);
7.
8. oscillator.start();
```

- ukázky: <http://seznam.github.io/CVUT/KAJ/prednasky/09/#28>