Další JavaScriptová API

Drag and drop

- přetahování věcí ve stránce a do stránky
- **zahájení tažení:**
  - např. soubor do okna prohlížeče
  - HTML prvek s atributem **draggable**
  - událost **dragstart**
  - vlastnost dataTransfer v události obsahuje relativní metadata

- **práce s dataTransfer:**

```
1. div.addEventListener("dragstart", function(e) {
2.     e.dataTransfer.setData("text/plain", "http://www.seznam.cz");
3.     e.dataTransfer.setDragImage(image, offsetX, offsetY);
4.     e.dataTransfer.effectAllowed = "copyMove";
5.     e.dataTransfer.dropEffect = "copy";
6. });
```

- **tažení**:
  - událost **dragenter**, **dragover**, **dragleave**
  - má-li být nad prvkem tažení ukončitelné, je nutné událost preventovat (**preventDefault()**)

- **puštění**:
  - událost **drop**
  - pokud událost zpracujeme, voláme preventDefault
  - vlastnost **e.dataTransfer.files** je pole souborů, jsou-li nějaké

Geolocation

- zjištění polohy uživatele
- asynchronní
- polohu zjišťuje prohlížeč nedefinovanými mechanismy
- GPS, IP, názvy wifi AP,…
- jen po HTTPS

```
1. var ok = function(position) {
2.     alert([position.coords.latitude, position.coords.longitude]);
3. }
4.
5. var error = function(e) {
6.     alert(e.message);
7. }
8.
9. navigator.geolocation.getCurrentPosition(ok, error);
```

- **watchPosition** pro opakované sledování
- **clearPosition** pro přerušení
- nalezená pozice obsahuje také rychlost, směr, výšku, přesnost

Web Storage

- úložiště dat v prohlížeči
- alespoň 5MB pro doménu
- data se nikam neodesílají
- triviální synchronní JS API

```
localStorage.setItem("a", "b");
// též localStorage.a, localStorage["a"]

localStorage.length == 1;
localStorage.key(0) == "a";
localStorage.getItem("a") == "b";

localStorage.removeItem("a");
```

- problém s anonymním režimem (aktuálně Safari)
- ukládají se jen dvojice řetězců (klíč-hodnota)
- storage je sdílena všemi skripty na doméně
    - událost storage při změně, lze využít pro komunikaci tab-tab
- slabé API s ohledem na kapacitu
- **sessionStorage** je oddělené úložiště, do zavření prohlížeče

URL

- globální objekt zastřešující dvě různá API
- statická metoda URL.createObjectURL pro vytvoření URL blobu
- funkce URL pro parsování řetězců

```
var base = "http://www.seznam.cz/";
var url = new URL("/a/b.cde?x=y#123", base);

url.href;         // http://www.seznam.cz/a/b.cde?x=y#123
url.origin;       // http://www.seznam.cz
url.hash;         // #123
url.searchParams; // instanceof URLSearchParams
// ... a mnohé další
```

File API

- přístup k souborům na disku uživatele
- uživatel musí soubor nejdříve „vybrat“ (draganddrop, input,…)
- metadata, čtení, odesílání (XHR2)

- **objekt File:**
  - **name**
  - **size** v bajtech
  - **type** (MIME) – odhad prohlížeče / OS

- **náhled obrázku:**

```
var input = document.createElement("input");
input.type = "file";

input.onchange = function(e) {
    var file = input.files[0];
    var url = URL.createObjectURL(file);
    // url = "blob:........"
    image.src = url;
}
```

- **objekt FileReader:**

```
var fr = new FileReader();
fr.addEventListener("load", function(e) {
    alert(e.target.result);
});

fr.readAsText(file);
/* fr.readAsDataURL(file); */
/* fr.readAsArrayBuffer(file); */
/* fr.readAsBinaryString(file); */
```

History API

- práce s URL stránky
- pro stavové jednostránkové aplikace
- podpora pro zpět/vpřed, záložky a permalinky

- **hash:**
  - práce s tzv. fragment identifikátorem, zkráceně též hashem
  - location.hash = „ahoj“
  - událost hashchange na objektu window

- **HTML5 History API**
  - možnost změny celého URL (bez načtení stránky)
  - funkce **history.pushState()** a **history.replaceState()**
  - history.pushState(date, title „/nove/URL“)
  - událost popstate na objektu window vzniká při uživatelově interakci s historií

Web Workers

- snaha o zpřístupnění dalších vláken JS
- JS vykonáván vždy jen v jednom vlákně!
- možnost spuštění kódu v izolovaných boxech
- problematika komunikace a předávání dat

```
1. var worker = new Worker("script.js");
2.
3. /* obsah souboru je nyní vykonáván v odděleném vlákně */
4.
5. worker.terminate();
```

- **komunikace s workerem:**

```
1. var worker = new Worker("script.js");
2.
3. worker.postMessage({nejaka:"data"});
4.
5. worker.onmessage = function(e) {
6.     alert(e.data);
7. }
```

- **komunikace z workeru:**

```
1. onmessage = function(e) {
2.     /* přišla data zvenčí */
3.
4.     postMessage(/* ... */); /* posílám data ven */
5. }
```

- **izolace**:
  - žádné window, žádný document
  - inportScripts, XMLHttpRequest/fetch, Worker
  - setTimeout, setInterval

- výpočet ve workeru může trvat libovolně dlouho
- data jsou při předávání klonována (netřeba řešit synchronizaci)
    - alternativa: transferrable
- vhodné tam, kde jsou třeba náročné výpočty
- iděálně málo vstupních i výstupních dat

Další JS API

Page Visibility API

- Je stránka vidět?
- událost **visibilitychange** na objektu document
- document.hidden
- document.visibilityState – „visible“, „hidden“, „prerender“

Fullscreen API

- možnost zobrazit konkrétní HTML prvek přes celou obrazovku
- pouze z posluchače události
- podpora dobrá, ale s různými názvy/prefixy
- pseudotřída **:full-screen** či **:fullscreen**
- document.fullscreenElement, document.fullscreenEnabled

```
var elem = document.querySelector("#myvideo");

if (elem.requestFullscreen) {
    elem.requestFullscreen();
} else if (elem.msRequestFullscreen) {
    elem.msRequestFullscreen();
} else if (elem.mozRequestFullScreen) {
    elem.mozRequestFullScreen();
} else if (elem.webkitRequestFullscreen) {
    elem.webkitRequestFullscreen();
}
```

```
if (document.exitFullscreen) {
    document.exitFullscreen();
} else if (document.msExitFullscreen) {
    document.msExitFullscreen();
} else if (document.mozCancelFullScreen) {
    document.mozCancelFullScreen();
} else if (document.webkitExitFullscreen) {
    document.webkitExitFullscreen();
}
```

FormData

- automatiká serializace key-value dat tvaru jako z HTML formuláře
- možnost předat do XMLHttpRequest::send()
- možnost postavit nad existujícím formulářem

```
var fd = new FormData( [form] );

fd.append("key1", "value");
fd.append("key2", file, "filename");

var blob = new Blob([data...]);
fd.append("key3", blob);
```

Indexed DB

- transakční key-value databáze v prohlížeči
- pro libovolné množství dat
- zcela asynchronní API
- databáze je persistentní per-origin

```
var r = indexedDB.open("mojeDB", 1);
r.onupgradeneeded = function(e) {
    var db = e.target.result;
    var r = db.createObjectStore("tabulka", {autoIncrement:true});
}

r.onsuccess = function(e) {
    var db = e.target.result;
    var t = db.transaction(["tabulka"], "readwrite");
    var r = t.objectStore("tabulka").add({some:"data"});
}
```

```
var r = indexedDB.open("mojeDB", 1);

r.onsuccess = function(e) {
    var db = e.target.result;
    var t = db.transaction(["tabulka"], "readonly");
    var r = t.objectStore("tabulka").get(1);
    r.onsuccess = function(e) {
        console.log(e.target.result); // {some: "data"}
    }
}
```