Psaní kódu v ArcGIS API for JavaScript 4

Zdeněk Jankovský, ARCDATA PRAHA, s.r.o.

Tento článek se zaměří na prostředí a nástroje, které nám mohou pomoci s psaním kódu v ArcGIS API for JavaScript 4.x. Místo základních funkcí a teorie kódu vás seznámí s rozšiřujícími tématy o tom, jaké nástroje je možné použít pro psaní robustnějšího kódu.

Probírat budeme následující témata:

- **Editor a pracovní prostředí** – jeho nastavení, doplňky a jak je efektivně využívat.
- **TypeScript** – superset JavaScriptu, jenž přidává rozšířené funkce, které je možné převést zpět na tradiční ECMAScript 5.
- Použití kaskádových stylů pomocí **knihovny Sass** a motivy JavaScript API, pomocí kterých je možné celé aplikaci jednoduše změnit vzhled.
- Nástroj **@arcgis/cli**, pomocí něhož můžeme získat šablonu kódu aplikace i šablonu pro tvorbu widgetu v JavaScript API 4.x.

ShowCoordinates.tsx

src > widgets > ShowCoordinates.tsx > ...

```
19  import ShowCoordinatesViewModel from "./ShowCoordinates/ShowCoordinatesViewModel";
20
21  interface ShowCoordinatesConstructorParams {
22    mapView: MapView
23  }
24
25  const CSS = {
26    base: "esri-widget zj-show-coords"
27  };
28
29  @subclass("app.widgets.ShowCoordinates")
30  export default class ShowCoordinates extends declared(Widget) {
31
32    roundPrec:number = 100;
33
34    wgsToSjtskTrans:GeographicTransformation = new GeographicTransformation({
35      steps: [
36        new GeographicTransformationStep({
37          isInverse: true,
38          wkid: 1623
39        })
40      ]
41    });
42
43    sjtskSpRef:SpatialReference = new SpatialReference({
44      wkid: 5514
45    });
46
47    isProjectionLoaded:boolean = false;
48
49    constructor(params: ShowCoordinatesConstructorParams) {
50      super();
51
52      this.viewModel =  new ShowCoordinatesViewModel({
53        mapView: params.mapView
54      });
```

Obr. 1. Ukázka kódu třídy v TypeScriptu.

NÁSTROJE PRO PRÁCI S JAVASCRIPTEM

Dnešní aplikace používají velké množství závislostí a nejrůznějších knihoven. Pro lepší práci s těmito zdroji se používá balíčkovací systém **npm** obsažený v prostředí **Node.js**.

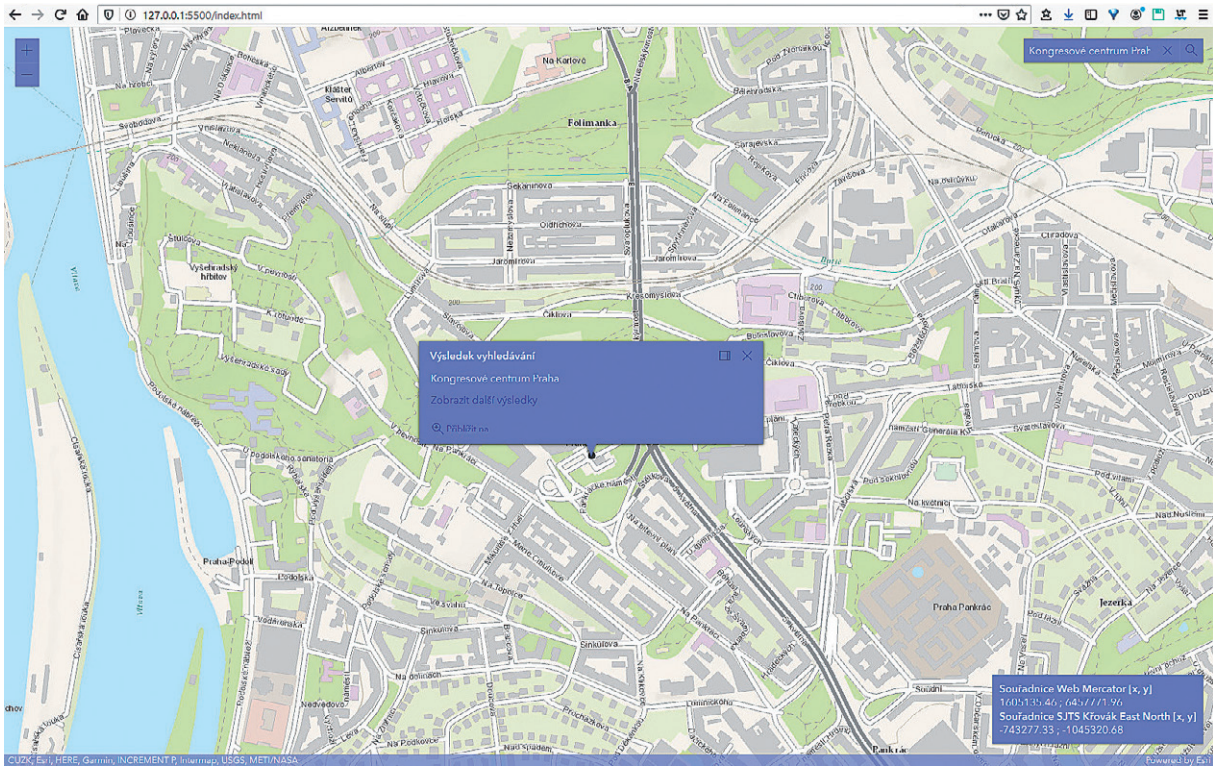
Pro jeho použití se nejprve na počítač nainstaluje prostředí *Node.js* a poté se pomocí balíčkovacího systému npm stahují a instalují závislosti pro aplikaci (JavaScript API, knihovny) i přidružené nástroje používané pro vývoj, jako jsou například *Compiler* pro Typescript, *WebPack* a podobné automatizační nástroje. Informace o těchto závislostech se uchovávají v souboru *config.json* a samotné balíčky jsou uloženy v adresáři *node_modules*.

Pro editaci kódu JavaScriptového API se osvědčil editor **VSCode**. Je rychlý a je dobře integrován s prostředím GIT i s příkazovou řádkou. Lze jej proto použít pro příkazy *npm*, což se nám bude hodit později. Pro *VSCode* existuje mnoho rozšíření. Z nich lze doporučit například rozšíření pro zvýraznění kódu (například zvýraznění závorek) a rozšíření pro integraci s GITem (*Git Graph*).

TYPESCRIPT

TypeScript je superset neboli nadmnožina JavaScriptu. To mimo jiné znamená, že jakýkoliv kód JavaScriptu ES5 je také kódem *TypeScript*. Použití *TypeScriptu* při psaní kódu umožňuje využít prvky, které vývojáři usnadňují práci. Před spuštěním aplikace dojde k překladu kódu zpět do JavaScriptu. Během tohoto procesu je provedena kontrola správnosti zápisu a vazeb, následně dojde k odstranění zápisů *TypeScriptu* a kód se stane surovým *ES5*. *TypeScript* při použití nikterak neupravuje jména proměnných ani neminifikuje kód (neodstraňuje nepotřebné znaky, jako jsou mezery apod.), k tomu mohou následně sloužit další nástroje.

ArcGIS API for JavaScript 4.x je v *TypeScriptu* přímo vyvíjeno. Proto také Esri poskytuje předpis objektů pro JavaScript, což nám pomáhá s psaním kódu a s odhalováním překlepů a základních chyb.



Obr. 2. Aplikace využívající upravený motiv Sass a vlastní widget.

Podíváme-li se na zápis kódu v *TypeScriptu*, nalezneme na první pohled některé rozdíly: soubor má například koncovku TS, čímž je označeno, že se jedná o *TypeScript*. Druhým rozdílem je import, který je zapsán pomocí notace *ES6*. *TypeScript* obecně používá notaci z *ES6*, což naznačují i deklarace proměnných pomocí klíčových slov *const* a *let*. *const* označuje konstantu, tedy proměnnou, která se v kódu nebude dále měnit, slovo *let* označuje proměnnou, která má oproti tradičnímu označení *var* (funkční *scope*) *scope* v rámci bloku kódu.

Následující ukázka demonstruje zápis několika proměnných pomocí notace *ES6*:

```
let centerPoint:number[] = [14.43, 50.062];
const view = new MapView({
  map: map,
  container: "viewDiv",
  center: centerPoint,
  zoom: 12
});
centerPoint = [14, 50];
view.center = new Point({x: centerPoint[0], y: centerPoint[1]});
```

Velkou výhodou *TypeScriptu* je využití syntaxe *ES6* při tvorbě tříd – *TypeScript* nás tak nutí psát objektově orientovaný kód, a tím přispívat k robustnosti a recyklaci komponent.

KASKÁDOVÉ STYLY A MOTIVY

Dále bychom rádi představili **motivy**, anglicky themes. Při vytváření JavaScriptového API použila Esri pro psaní CSS technologii **Sass**. *Sass* můžeme považovat za rozšíření jazyka CSS nebo za nadmnožinu jazyka CSS, podobně jako *TypeScript* u JavaScriptu. *Sass* také můžeme nazvat jako preprocesor, tedy nástroj, který převede kód *Sass* do CSS, který je pak ve standardní podobě k dispozici prohlížeči.

Sass přináší ulehčení při psaní komplexních kaskádových stylů – například v podobě proměnných, vnořených definic, matematických výrazů či mixinů. Může jednoduše přepsat proměnné prostředí, kód zkompilovat, a tím získat kompletně změněný vizuální vzhled aplikace.

BEM

S kaskádovými styly v ArcGIS API for JavaScript souvisí pojem **BEM**. Je to zkratka pro *Block – Element – Modifier*, což přímočaře vyjadřuje, o co se jedná: BEM není technologie nebo programovací jazyk, je to sada doporučení, jak pojmenovávat jednotlivé selektory CSS. Názvosloví BEM souvisí s objektovým psaním kaskádových stylů (OOCSS), přičemž BEM tento koncept rozšiřuje.

Styly se dle BEM vytvářejí samostatně, bez závislosti vzhledu na struktuře. Není proto doporučeno používat modifikátory standardních tagů. CSS se píše s co nejmenší specifičností, tzn. například bez klauzule !*important*,

selectoru potomků nebo bez kombinovaných selectorů. Vývoj je tímto způsobem zaměřen na komponenty a na jejich opakované použití. Například pokud vývojář pozmění hierarchii HTML, nemělo by být potřeba měnit předpis kaskádových stylů.

Malá praktická ukázka: za *blok* se považuje funkční, samostatný blok aplikace. U nás si pod tím můžeme představit například *widget*. *Element* je pak část *bloku*, například název souřadnicového systému. Dle předpisu BEM je oddělen pomocí dvou podtržítek. *Modifikátor*, oddělený pomocí dvou pomlček, konkretizuje nějaký stav elementu, i třeba *bloku*. Předpis *modifikátoru* neopakuje kód, v aplikaci se tedy použije dohromady předpis jak původní komponenty, tak *modifikátoru*.

ArcGIS CLI – COMMAND LINE INTERFACE

Nástroj **ArcGIS CLI (Command Line Interface)** obsahuje skripty v prostředí *Node.js* pro tvorbu šablon vybraných aplikací. Pomocí nástroje je možné rychle vygenerovat základ aplikace nebo základ widgetu.

CLI se používá velice jednoduše. Ve vybraném adresáři provedeme instalaci pomocí

```
npm install @arcgis/cli
```

Npm tím stáhne balíček a všechny závislosti. Následně použijeme globálně instalovaný balíček *npm-run* a pomocí něj spustíme nástroj na tvorbu a vytvoření widgetu.

```
npm-run arcgis widget muj-widget
```

Vytvořila se základní struktura kódu widgetu: vlastní složka, základní soubor *TypeScriptu*, *ViewModel* widgetu a také soubor stylů pro *Sass*.

WIDGETY

Widget je komponenta dědící od třídy *Widget* JS API. Tím ji můžeme snadno začlenit do uživatelského prostředí aplikace JS API 4.x; stačí vložit funkci *view.ui.add()* před označení rohu okna MapView. Engine widgety na obrazovce automaticky uspořádá.

Na struktuře widgetu je důležité rozdělení **View** a **ViewModelu**. *View* slouží pro zobrazení komponenty, tedy jejího vykreslení v aplikaci, *ViewModel* slouží pro vytvoření business logiky. Esri dává k dispozici neminifikovaný kód *View* všech widgetů – je tedy možné část widgetů pro vlastní potřebu modifikovat nebo alespoň prostudovat jejich zdrojový kód.

SHRNUTÍ

V článku jsme se seznámili s prostředím *Node.js* a jeho balíčkovacím systémem. Pomocí něho je možné stáhnout závislosti pro vývoj aplikace a použít automatizaci pro sestavení kódu a stylů aplikace. Představili jsme si editor Visual Studio Code, dále byla řeč o *TypeScript*. Pro JavaScriptové API jsme využili zejména deklaraci tříd, syntaxi JSX a typovost jazyka.

Řeč byla o preprocesoru *Sass*, změně motivu aplikace pomocí proměnných v *Sassu* a také o metodologii pojmenování tříd CSS BEM. Na závěr jsme představili nástroj *arcgis/cli* pro generování šablon aplikací (popřípadě adresářů widgetu) a ve spojitosti s tímto nástrojem jsme si ukázali také základní body frameworku pro widgety, tak jak je nastaven v rámci JavaScriptového API. «

Ing. Zdeněk Jankovský, ARCDATA PRAHA, s.r.o.
Kontakt: zdenek.jankovsky@arcdata.cz

Odkazy

Ukázkový kód aplikace JS API 4.x: http://bit.ly/2rbotdb
Node.js a npm: https://nodejs.org/en/
Záznam přednášky z Konference GIS Esri 2019: https://youtu.be/Sc7Rtkoglgw