

# Knihovna pro geometrii v rovině

## Jak se vytvářejí knihovny v jazyce TypeScript

Jiří Cihelka

Gymnázium Christiana Dopplera

5. března 2024



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Cíle mé ročníkové práce

## Cíl práce

Vytvořit a publikovat knihovnu pro jazyk TypeScript, která bude efektivně využívat moderních nástrojů a která bude splňovat běžně očekávané standardy pro takovou knihovnu. Knihovna poskytuje třídy a funkce pro manipulaci s geometrickými objekty v rovině.



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript**
- 3 Co jsou knihovny
- 4 Vývoj knihoven
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Co je TypeScript?

- Rozšíření jazyka JavaScript



# Co je TypeScript?

- Rozšíření jazyka JavaScript
- Přidává statické typování



# Co je TypeScript?

- Rozšíření jazyka JavaScript
- Přidává statické typování
- Transpiluje se do JavaScriptu





## Příklad - JavaScript

```
function filter(array, predicate) {  
    var result = [];  
    for (var i = 0; i < array.length; i++) {  
        if (predicate(array[i])) {  
            result.push(array[i]);  
        }  
    }  
    return result;  
}
```

Obrázek: Příklad implementace funkce `filter[2]` v jazyce JavaScript.



# Příklad - TypeScript

```
function filter<T>(array: Array<T>, predicate: (value: T) =>
boolean): Array<T> {
    var result: Array<T> = [];
    for (var i = 0; i < array.length; i++) {
        if (predicate(array[i])) {
            result.push(array[i])
        }
    }
    return result;
}
```

Obrázek: Přepis funkce z předchozího slidu do TypeScriptu.



## Příklad - TypeScript (moderní syntaxe)

```
function filter<T>(array: Array<T>, predicate: (item: T) =>  
boolean): Array<T> {  
    const result: Array<T> = [];  
    for (const item of array) {  
        if (predicate(item)) {  
            result.push(item)  
        }  
    }  
    return result;  
}
```

Obrázek: Přepis funkce z předchozího slidu za použití moderní syntaxe.



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny**
- 4 Vývoj knihoven
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Co jsou knihovny

## Definice (Softwarová knihovna)

**Knihovna** je souhrn procedur, funkcí, konstant, datových typů a tříd, který může být využíván více počítačovými programy.[3]



# Důležité vlastnosti knihoven

- Jsou znovupoužitelné (nezávislé na konkrétním programu)



# Důležité vlastnosti knihoven

- Jsou znovupoužitelné (nezávislé na konkrétním programu)
- Jsou stabilní (změny API jsou vzácné a verzované)



# Důležité vlastnosti knihoven

- Jsou znovupoužitelné (nezávislé na konkrétním programu)
- Jsou stabilní (změny API jsou vzácné a verzované)
- Jsou dokumentované (API je zdokumentováno)





# Důležité vlastnosti knihoven

- Jsou znovupoužitelné (nezávislé na konkrétním programu)
- Jsou stabilní (změny API jsou vzácné a verzované)
- Jsou dokumentované (API je zdokumentováno)
- Jsou testované



# Důležité vlastnosti knihoven

- Jsou znovupoužitelné (nezávislé na konkrétním programu)
- Jsou stabilní (změny API jsou vzácné a verzované)
- Jsou dokumentované (API je zdokumentováno)
- Jsou testované
- Jsou rozšiřitelné (lze je rozšiřovat bez změny API)



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven**
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Vývoj knihoven

Vývoj knihoven se příliš neliší od vývoje běžných velkých aplikací. Je na něm ale jednodušší ukázat důležité koncepty, které se nám mohou jevit při vývoji menších projektů zbytečné.



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven**
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Verzování - kód

Verzování kódu nám umožňuje:

- Uchovávat historii změn
- Spolupracovat na kódu
- Zpětně se vracet ke starším verzím v případě chyby



# Verzování - Git

K verzování kódu knihovny jsem zvolil nástroj Git s webovou službou GitHub.



# Verzování - Git

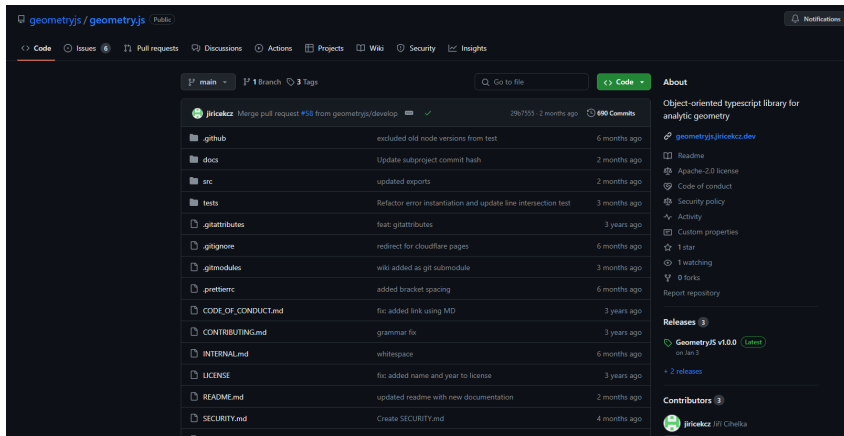
K verzování kódu knihovny jsem zvolil nástroj Git s webovou službou GitHub.  
Tuto kombinaci jsem zvolil převážně protože:

- Git je zdaleka nejpoužívanější nástroj pro verzování kódu na světě
- GitHub nabízí výhodné podmínky pro open-source projekty a vše, co potřebuji, je zdarma
- Projekt je snadno vyhledatelný a má přehlednou stránku repozitáře přímo na GitHubu





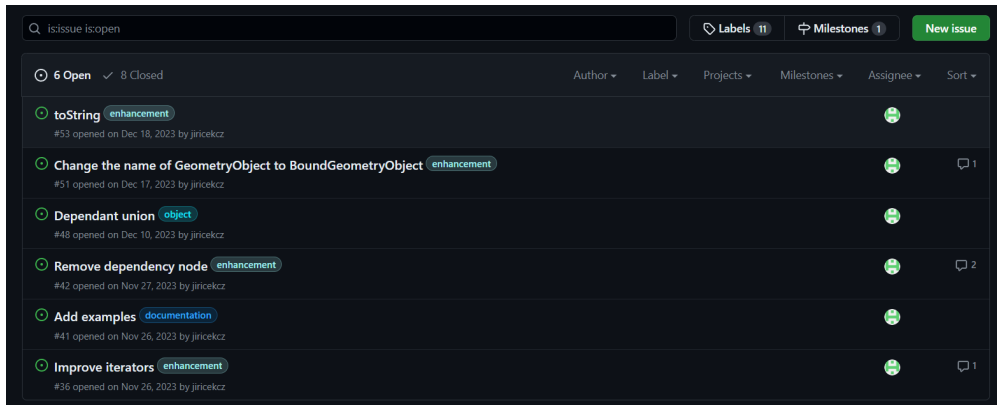
# Verzování - Stránka projektu na GitHubu



Obrázek: Ukázka repozitáře na GitHubu



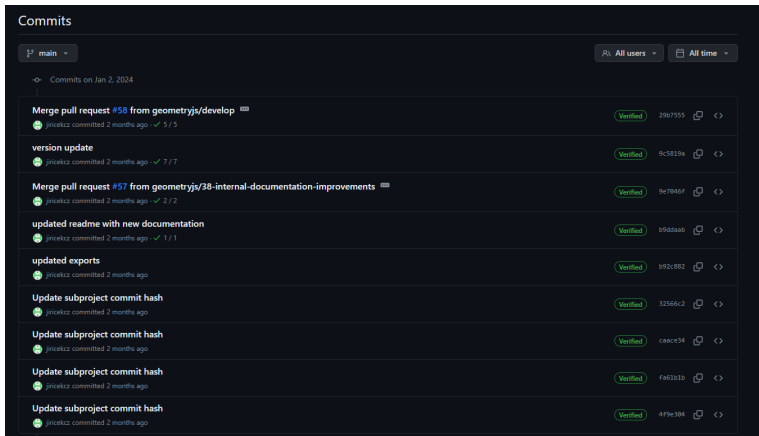
# Verzování - Stránka projektu na GitHubu



Obrázek: Ukázka správy issues na GitHubu



# Verzování - Stránka projektu na GitHubu



Obrázek: Ukázka historie commitů na GitHubu



# Verzování - Vydání

## Definice (Vydání)

**Vydání** (release) je označení určité verze kódu.

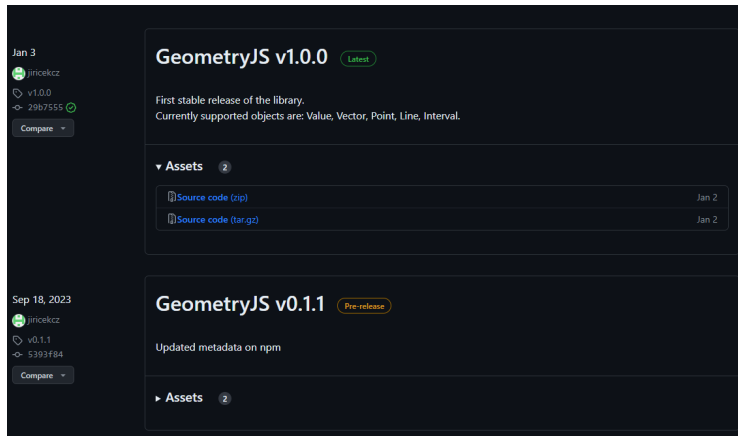
## Číslování vydání

Standardní způsob číslování vydání je pomocí tzv. **Semantic Versioning**[1]

Zjednodušeně: MAJOR.MINOR.PATCH



# Verzování - Ukázka vydání



Obrázek: Ukázka vydání na GitHubu



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven**
  - Verzování
  - **Správa závislostí a publikace**
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Správce balíčků

## Definice (Správce balíčků)

**Správce balíčků** (package manager) je nástroj pro správu softwaru a jeho závislostí.[4]



# Správce balíčků

## Definice (Správce balíčků)


**Správce balíčků** (package manager) je nástroj pro správu softwaru a jeho závislostí.[4]

Správce balíčků umožní publikaci, instalaci a aktualizaci vydání knihovny.





# Správce balíčků - Ukázka NPM

@jiricekcz/geometry.js 

1.0.0 • Public • Published 2 months ago

 Readme

 Code Beta

 0 Dependencies

 0 Dependents

 3 Versions

## Geometry JS

Geometry JS is a JavaScript/TypeScript library for creating and manipulating 2D geometry objects.

### Table of contents

- Geometry JS
  - Table of contents
  - Getting started
  - Documentation
  - Requirements
  - Usage

### Getting started


Install the library using npm:

```
npm install @jiricekcz/geometry-js
```


#### Install

```
> npm i @jiricekcz/geometry.js
```

#### Repository

 [github.com/geometryjs/geometry.js](https://github.com/geometryjs/geometry.js)

#### Homepage

 [geometryjs.jiricekcz.dev](https://geometryjs.jiricekcz.dev)

#### Weekly Downloads

2

#### Version

1.0.0

#### License

ISC

#### Unpacked Size

493 kB

#### Total Files

447

#### Issues

-

#### Pull Requests

-

Obrázek: Ukázka správce balíčků NPM



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven**
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování**
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Automatizované testování

## Automatizované testování

Automatizací testů si ušetříme práci a zároveň získáme jistotu, že naše knihovna funguje tak, jak má. Programy pro automatizované testování nám také mohou poskytnout informaci o pokrytí kódu testy.



# Automatizované testování - Jest

Spouštěné lokálně v příkazové řádce

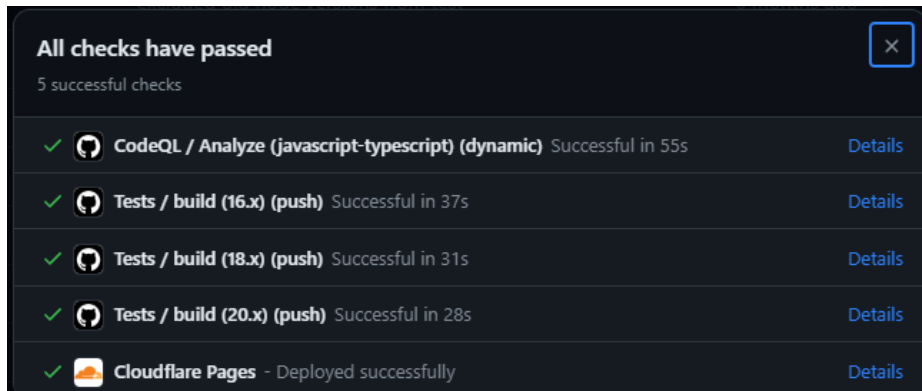
```
Test Suites: 28 passed, 28 total
Tests:       793 passed, 793 total
Snapshots:   0 total
Time:        7.31 s, estimated 9 s
Ran all test suites.
```

Obrázek: Výsledek automatizovaných testů pomocí Jestu.



# Automatizované testování - Jest

Automaticky spouštěné při pushi na GitHub

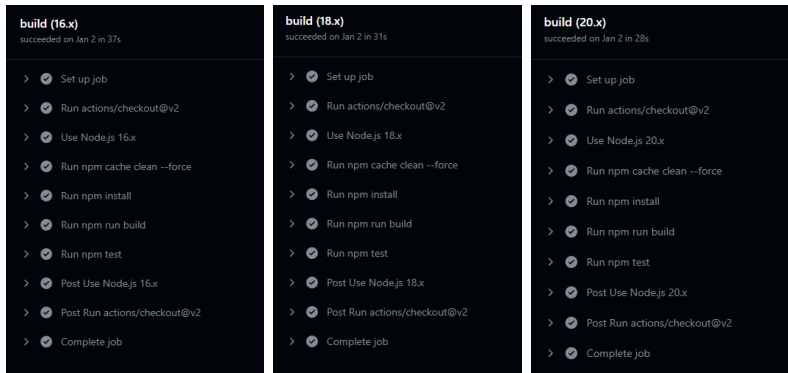


Obrázek: Výsledek „GitHub Actions“ při pushi na GitHub.



# Automatizované testování - Jest

Automaticky spouštěné při pushi na GitHub



Obrázek: Podrobný výsledek testů při pushi na GitHub.



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven**
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Uživatelská dokumentace

## Uživatelská dokumentace

**Uživatelská dokumentace** je důležitá pro uživatele knihovny, kteří se snaží pochopit, jak knihovnu používat. Měla by být jednoduchá a srozumitelná. Měla by zároveň popisovat jednotlivé funkce knihovny.





# Uživatelská dokumentace - TypeDoc

## Generování dokumentace z kódu

### TypeDoc

TypeDoc je nástroj, který umožňuje generovat dokumentaci z kódu. Dokumentace je generována z komentářů v kódu.



# Uživatelská dokumentace - TypeDoc

## Ukázka komentáře v JSDoc syntaxi

```
/**
 * Calculates the n-th number of the fibonacci sequence
 * @param n The number of the fibonacci sequence to calculate
 * @returns The n-th number of the fibonacci sequence
 *
 * @see {@link https://en.wikipedia.org/wiki/Fibonacci_number |
Fibonacci number}
 * @throws {TypeError} If n is not a positive integer
 * @throws {RangeError} If n is greater than 1476
 *
 * @example
 * fibonacci(0); // 0
 * fibonacci(1); // 1
 * fibonacci(2); // 1
 * fibonacci(3); // 2
 * fibonacci(4); // 3
 */
declare function fibonacci(n: number): number;
```

Obrázek: Ukázka komentáře v JSDoc syntaxi.



# Uživatelská dokumentace - TypeDoc

Ukázka metody `normalize` rozhraní `Vector`

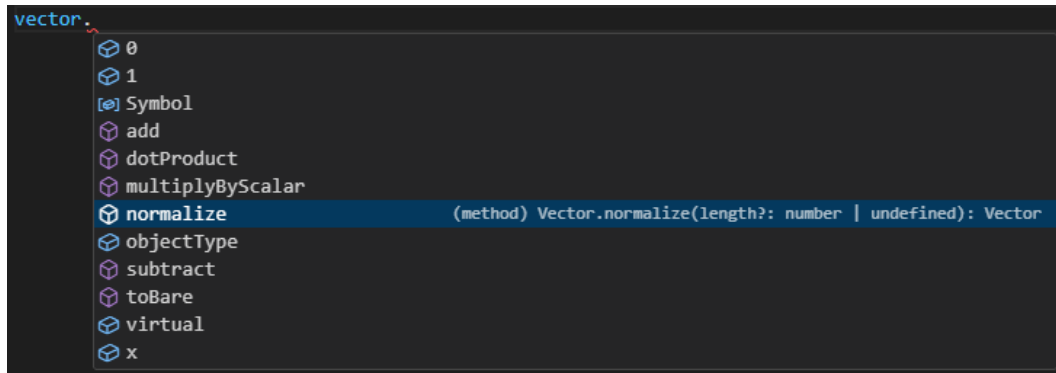
```
/**  
 * Normalizes the vector to a length, defaulting to 1, to create a  
 * new vector.  
 * @param length The length to normalize to.  
 * @returns A new vector with the same direction as this vector and  
 * the given length.  
 */  
normalize(length?: number): Vector;
```

Obrázek: Ukázka komentáře v JSDoc syntaxi z knihovny.



# Uživatelská dokumentace - TypeDoc

Ukázka metody `normalize` rozhraní `Vector`



Obrázek: Ukázka IntelliSense v editoru Visual Studio Code.



# Uživatelská dokumentace - TypeDoc

Ukázka metody `normalize` rozhraní `Vector`

```
normalize(length?: number | undefined): Vector
```

The length to normalize to.

Normalizes the vector to a length, defaulting to 1, to create a new vector.

*@returns* — A new vector with the same direction as this vector and the given length.

Obrázek: Ukázka detailu IntelliSense v editoru Visual Studio Code.



# Uživatelská dokumentace - TypeDoc

Ukázka metody `normalize` rozhraní `Vector`

## `normalize`

`normalize(length?): Interfaces.Vector`

Normalizes the vector to a length, defaulting to 1, to create a new vector.

### Parameters

- Optional **length:** *number*

The length to normalize to.

**Returns** *Interfaces.Vector*

A new vector with the same direction as this vector and the given length.

Defined in [src/interfaces/vector.ts:76](#)

Obrázek: Ukázka webové dokumentace.



# Programátorská dokumentace

GitHub Wiki

## Programátorská dokumentace

**Programátorská dokumentace** je důležitá pro vývojáře, kteří chtějí přispívat do knihovny. Měla by obsahovat informace o tom, jak knihovnu spravovat, jak ji testovat a jak ji publikovat. Také by měla obsahovat informace o tom, jak knihovnu rozšířit.



# Programátorská dokumentace

[GitHub Wiki](#)

## Programátorská dokumentace

**Programátorská dokumentace** je důležitá pro vývojáře, kteří chtějí přispívat do knihovny. Měla by obsahovat informace o tom, jak knihovnu spravovat, jak ji testovat a jak ji publikovat. Také by měla obsahovat informace o tom, jak knihovnu rozšířit.

## [GitHub Wiki](#)

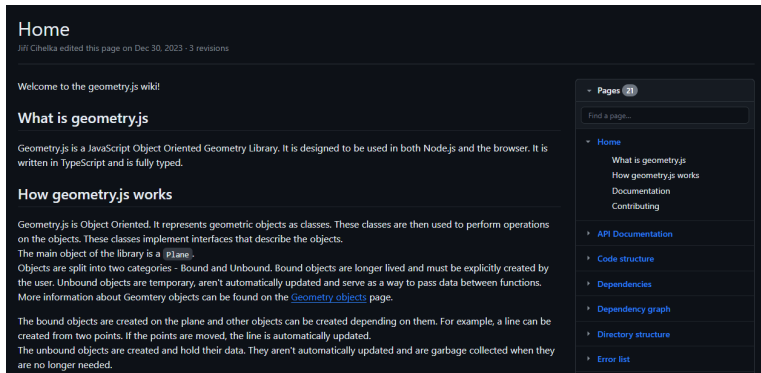
GitHub Wiki je jednoduchý nástroj, který umožňuje vytvářet programátorskou dokumentaci. Je součástí každého repozitáře na GitHubu.





# Programátorská dokumentace

## GitHub Wiki



Obrázek: Ukázka domovské stránky GitHub Wiki.



# Obsah

- 1 Cíle mé ročníkové práce
- 2 Programovací jazyk TypeScript
- 3 Co jsou knihovny
- 4 Vývoj knihoven
  - Verzování
  - Správa závislostí a publikace
  - Automatizované testování
  - Programátorská a uživatelská dokumentace
- 5 Příklady využití knihovny



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';
```



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';  
const plane = GeometryJS.createPlane();
```



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';

const plane = GeometryJS.createPlane();

const x = plane.createValue(1);
const y = plane.createReadOnlyValue(2);
```



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';

const plane = GeometryJS.createPlane();

const x = plane.createValue(1);
const y = plane.createReadOnlyValue(2);

const A = plane.createPointFromTwoValues(x, y);
const B = plane.createPointFromCoordinates(3, 4);
```



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';

const plane = GeometryJS.createPlane();

const x = plane.createValue(1);
const y = plane.createReadOnlyValue(2);

const A = plane.createPointFromTwoValues(x, y);
const B = plane.createPointFromCoordinates(3, 4);

const l = plane.createLineFromTwoPoints(A, B);
```



# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';

const plane = GeometryJS.createPlane();

const x = plane.createValue(1);
const y = plane.createReadOnlyValue(2);

const A = plane.createPointFromTwoValues(x, y);
const B = plane.createPointFromCoordinates(3, 4);

const l = plane.createLineFromTwoPoints(A, B);

const O = plane.createPointFromCoordinates(0, 0);
```





# Příklady využití

## Sestavení minimalistického modelu

```
import * as GeometryJS from '@jiricekcz/geometry.js';

const plane = GeometryJS.createPlane();

const x = plane.createValue(1);
const y = plane.createReadOnlyValue(2);

const A = plane.createPointFromTwoValues(x, y);
const B = plane.createPointFromCoordinates(3, 4);


const l = plane.createLineFromTwoPoints(A, B);


const O = plane.createPointFromCoordinates(0, 0);

const k = plane.constructPerpendicularLineFromPoint(l, O);
```



# Zdroje I

 Tom Preston-Werner.  
Semantic versioning 2.0.0.  
<https://semver.org/spec/v2.0.0.html>.

 Wikipedia contributors.  
Filter (higher-order function) — Wikipedia, the free encyclopedia.  
[https://en.wikipedia.org/w/index.php?title=Filter\\_\(higher-order\\_function\)&oldid=1080346575](https://en.wikipedia.org/w/index.php?title=Filter_(higher-order_function)&oldid=1080346575), 2022.  
[Online; accessed 3-March-2024].



## Zdroje II



Wikipedie.

Knihovna (programování) — wikipedie: Otevřená encyklopedie.

[https://cs.wikipedia.org/w/index.php?title=Knihovna\\_\(programov%C3%A1n%C3%AD\)&oldid=23265368](https://cs.wikipedia.org/w/index.php?title=Knihovna_(programov%C3%A1n%C3%AD)&oldid=23265368), 2023.

[Online; navštíveno 03. 03. 2024].



Wikipedie.

Správce balíčků — wikipedie: Otevřená encyklopedie.

[https://cs.wikipedia.org/w/index.php?title=Spr%C3%A1vce\\_bal%C3%AD%C4%8Dk%C5%AF&oldid=23024887](https://cs.wikipedia.org/w/index.php?title=Spr%C3%A1vce_bal%C3%AD%C4%8Dk%C5%AF&oldid=23024887), 2023.

[Online; navštíveno 03. 03. 2024].



# Odkazy

## Odkazy na součásti projektu

Název	Odkaz
Tato prezentace	<a href="https://geometryjs-showcase.jiricekcz.dev/presentation.pdf">https://geometryjs-showcase.jiricekcz.dev/presentation.pdf</a>
Ročníková práce	<a href="https://geometryjs-showcase.jiricekcz.dev/document.pdf">https://geometryjs-showcase.jiricekcz.dev/document.pdf</a>
Kód	<a href="https://github.com/geometryjs/geometry.js">https://github.com/geometryjs/geometry.js</a>
NPM balíček	<a href="https://www.npmjs.com/package/@jiricekcz/geometry.js">https://www.npmjs.com/package/@jiricekcz/geometry.js</a>
Dokumentace	<a href="https://geometryjs.jiricekcz.dev/">https://geometryjs.jiricekcz.dev/</a>

