

Интеграция Rust в Node

Николай Шувалов

О себе

- Senior Full Stack Developer
- Beeline



- <https://t.me/evelas>
- @evelas



План

- Зачем Rust понадобился в Node
- Немного затронем влияние Rust на сообщество JS
- Какие существуют методы внедрения С-подобных языков в Node
- Посмотрим Rust фреймворк для работы с внедрением в Node
- Расскажу про дистрибуцию
- Разберем примеры использования

Зачем понадобился Rust в приложении на JS?

- Более эффективный код
- Инструмент по назначению
- Влияние Rust на JS



Speedy Web Compiler

SWC можно использовать как для компиляции, так и для сборки.

- Используется в Next.js
- В 20 раз быстрее Babel
- В 2 раза быстрее Webpack

При помощи чего можно интегрировать С-подобный язык в Node?

- Native Abstractions for Node.js (NAN). Реализован с использованием прямых вызовов Chrome V8. Необходимо обновлять каждый раз, когда обновляется движок V8, используемый Node.
- Node API (N-API). Набор API не зависит от базовой среды выполнения JavaScript (например, V8) и поддерживается как часть самого Node.js. Неудобно пользоваться на прямую

NAPI

- Набор API
- Поддерживается как часть node js
- Все API возвращают napi-status
- Не нужно пересобирать под обновление версии node

NAPI-RS

Фреймворк для создания pre-compiled Node.js addons в Rust.

- Совместим с CommonJS/ESM
- Удобно работать с NAPI
- Автоматически сгенерированный файл .d.ts
- Параллелизм в несколько строк
- Дистрибьюция на разные платформы

Дистрибуция

```
✓ npm
  > darwin-x64
  > linux-x64-gnu
  ✓ win32-x64-msvc
    {} package.json
    ⓘ README.md
```

```
npm > win32-x64-msvc > {} package.json > ...
```

```
1  {
2    "name": "piterjs-rust-examples-win32-x64-msvc",
3    "version": "0.0.6",
4    "os": [
5      "win32"
6    ],
7    "cpu": [
8      "x64"
9    ],
10   "main": "piterjs-rust-examples.win32-x64-msvc.node",
11   "files": [
12     "piterjs-rust-examples.win32-x64-msvc.node"
13   ],
14   "license": "MIT",
15   "engines": {
16     "node": ">= 10"
17   }
18 }
```

piterjs-rust-examples TS

0.0.6 • Public • Published 20 hours ago



Readme



Code

Beta



3 Dependencies



Dependencies (3)

piterjs-rust-examples-win32-x64-msvc piterjs-rust-examples-darwin-x64

piterjs-rust-examples-linux-x64-gnu

Какие примеры посмотрим?

- Типы
- Функции
- Многопоточность

Более подробно посмотреть примеры по ссылке:

<https://github.com/evelas/piterjs-rust/tree/master>



```
#[napi(object)]
pub struct Pet {
    pub name: String,
    pub age: Option<u32>,
}
```

```
#[napi]
pub fn print_pet(pet: Pet) {
    println!("{}", pet.name);
}
```

```
#[napi]
pub fn create_cat() -> Pet {
    Pet {
        name: "Vaska".to_string(),
        age: Some(2),
    }
}
```

9

10

11

12

13

14

15

16



```
export interface Pet {
    name: string
    age?: number
}
```

```
export function printPet(pet: Pet): void
export function createCat(): Pet
```

```
9      #[napi(  
10          ts_args_type = "callback: (  
11              processed: number,  
12              total: number,  
13              fileIteration: number,  
14          ) => void",  
15          ts_return_type = "Promise<string>"  
16      )]  
17      pub fn loading_files_info_napi(  
18          env: Env,  
19          callback: JsFunction,  
20      ) -> Result<JsObject> {
```

```
21 let callback_thread_safe: ThreadsafeFunction<(u64, u64, usize), ErrorStrategy::Fatal> =
22     callback
23     .create_threadsafe_function(0, |ctx: ThreadSafeCallContext<(u64, u64, usize)>| {
24         let mut v: Vec<JsNumber> = Vec::new();
25         v.push(ctx.env.create_uint32(ctx.value.0 as u32).unwrap());
26         ...
27         Ok(v)
28     })
29     .unwrap()
30 ;
```

```
31
32     env.execute_tokio_future(async {
33         let result = rust_mod::loading_files_with_progress(
34             Box::new(move |p, t, f| {
35                 callback_thread_safe.call((p, t, f), ThreadsafeFunctionCallMode::NonBlocking);
36             })),
37         );
38         Ok(result)
39     }, |_env, data| Ok(data))
40 }
41
```

src > TS test.ts >  bootstrap >  infoCallback

```
1  import { loadingFilesInfoNapi } from 'piterjs-rust-examples';
2
3  async function bootstrap() {
4      const infoCallback = (
5          processed: number,
6          total: number,
7          fileIteration: number,
8      ) => {
9          console.log(`Loading file #${fileIteration}. ${processed} / ${total}`);
10     };
11
12     try {
13         await loadingFilesInfoNapi(infoCallback);
14     } catch (err) {
15         console.error(err);
16     }
17 }
18 bootstrap();
19
```


Loading file #0. 0 / 2438455
Loading file #1. 0 / 3297868
Loading file #0. 426 / 2438455
Loading file #1. 426 / 3297868
Loading file #0. 1507 / 2438455
Loading file #1. 1507 / 3297868
Loading file #0. 2533 / 2438455
Loading file #1. 2533 / 3297868
Loading file #0. 3597 / 2438455
Loading file #1. 3597 / 3297868
Loading file #0. 4647 / 2438455
Loading file #1. 4647 / 3297868
Loading file #0. 5675 / 2438455
Loading file #1. 5675 / 3297868
Loading file #0. 6750 / 2438455
Loading file #1. 6750 / 3297868
Loading file #1. 7799 / 3297868
Loading file #0. 7799 / 2438455
Loading file #1. 8855 / 3297868
Loading file #0. 8855 / 2438455
Loading file #1. 9899 / 3297868
Loading file #0. 9899 / 2438455
Loading file #1. 10994 / 3297868
Loading file #0. 10994 / 2438455

TS test.ts 1. U ●

src > TS test.ts > bootstrap

```
1 import { loadingFilesInfoNapi } from 'piterjs-rust-examples';
2
3 async function bootstrap() {
4     const wrongCallback = (
5         stage: string,
6     ) => {
7         console.log(`Stage is: ${stage}`);
8     };
9
10    try {
11        await loadingFilesInfoNapi(wrongCallback);
12    } catch (err) {
13        console.error(err);
14    }
15 }
16 bootstrap();
17
```

Argument of type '(stage: string) => void' is not assignable to parameter of type '(processed: number, total: number, fileIteration: number) => void'.

Types of parameters 'stage' and 'processed' are incompatible.

Type 'number' is not assignable to type 'string'. ts(2345)

const wrongCallback: (stage: string) => void

[View Problem \(Alt+F8\)](#) No quick fixes available

Что внутри фреймворка?

env.rs



C: > Users > evela > .cargo > registry > src > index.crates.io-6f17d22bba15001f > napi-2.13.2 > src > env.rs > {} impl Env > create_uint32

```
87 pub fn create_uint32(&self, number: u32) -> Result<JsNumber> {
88     let mut raw_value: *mut napi_value__ = ptr::null_mut();
89     check_status!(unsafe { sys::napi_create_uint32(self.0, number, &mut raw_value) })?;
90     Ok(unsafe { JsNumber::from_raw_unchecked(env: self.0, raw_value) })
91 }
```

Результат

- Сократили количество кода
- Легко поддерживать код
- Увеличилась скорость обработки данных

Выводы

- Посмотрели как влияет Rust на сообщество JS
- Узнали про способы внедрения C-подобных языков в Node
- Увидели фреймворк, который помогает с внедрением в действии
- Затронули несколько Node API на примерах
- Внедрили приложение на Rust в Node

Вопросы

