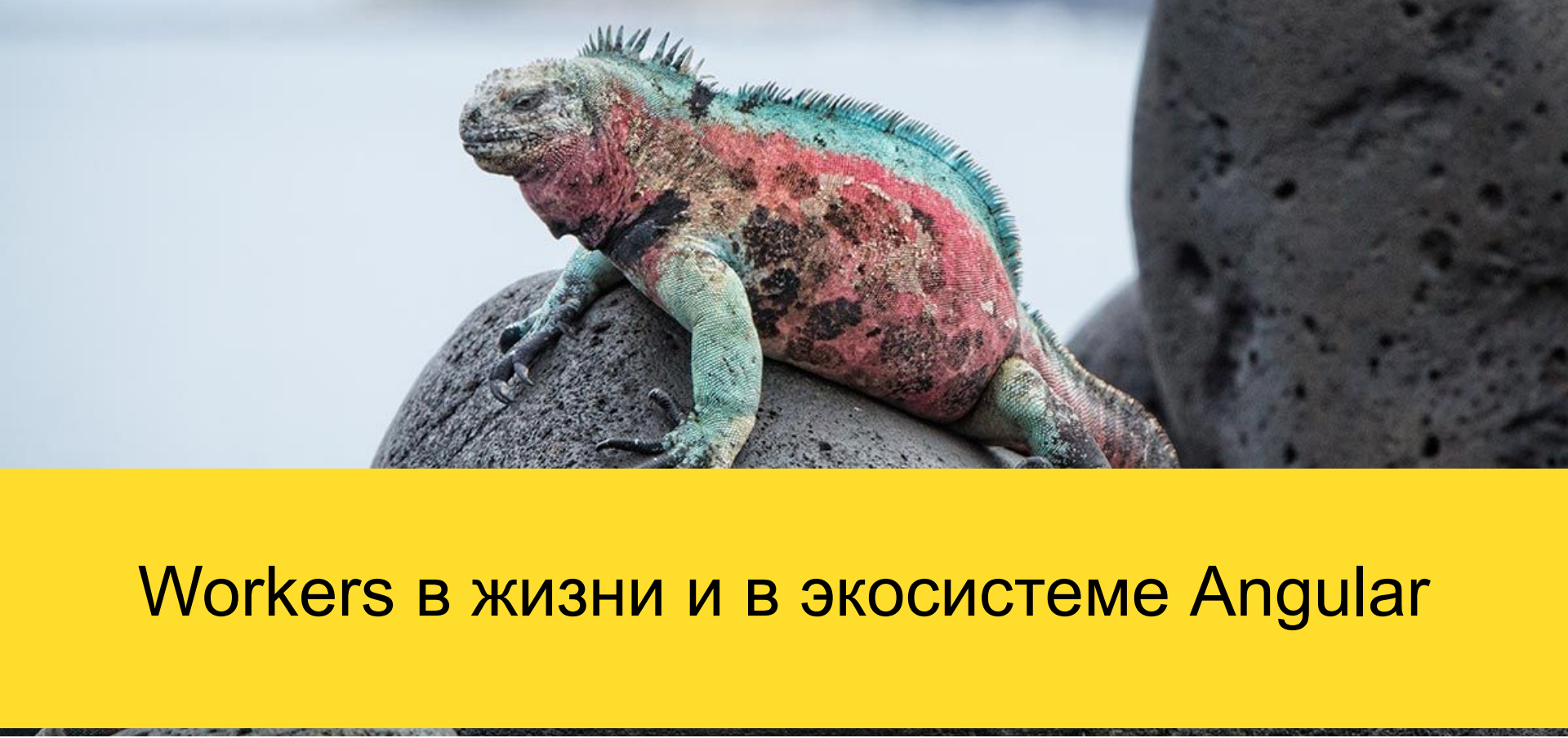


Workers в жизни и в экосистеме Angular

Tinkoff.ru

Обо мне

Духовняк Александр

Старший разработчик, Tinkoff.ru

shanhaichik@gmail.com
https://github.com/shanhaichik

Будем говорить о том:

- Что такое Web Workers и Shared Workers

Будем говорить о том:

- Что такое Web Workers и Shared Workers
- Какие модули есть в экосистеме Angular для работы с Web Workers и как с ними работать

Демо №1

Рассмотрим на примере!

https://github.com/shanhaichik/angular-meetup-worker-demo

Рассмотрим

- Что такое Web Worker и какие задачи решает

Рассмотрим

- Что такое Web Worker и какие задачи решает
- Как создается и конфигурируется Web Worker

Рассмотрим

- Что такое Web Worker и какие задачи решает
- Как создается и конфигурируется Web Worker
- Как происходит взаимодействие с Web Worker и какие типы данных можно в него передавать

Рассмотрим

- Что такое Web Worker и какие задачи решает
- Как создается и конфигурируется Web Worker
- Как происходит взаимодействие с Web Worker и какие типы данных можно в него передавать
- DedicatedWorkerGlobalScop

Web Worker - это

- Инструмент для асинхронной загрузки сторонних скриптов и запуска их выполнения в отдельном потоке

Web Worker - это

- Инструмент для асинхронной загрузки сторонних скриптов и запуска их выполнения в отдельном потоке

- Инструмент решающий проблему однопоточности в JS

Создание обычным программистом

```
const options = { name: 'MyWorker' };

const worker = new Worker('getMyWorker.js', options);
```

https://developer.mozilla.org/en-US/docs/Web/API/Worker/Worker

Создание скучающим программистом

```
const code = new Blob([`(${worker_code.toString()})()`], {type: 'text/javascript'});

const url = URL.createObjectURL(code);

const worker = new Worker(url, options);
```

https://developer.mozilla.org/en-US/docs/Web/API/Worker/Worker

Получение и отправка данных

```
worker.addEventListener('message', event => {
    console.log(event.data);
}, false);
```

https://developer.mozilla.org/en-US/docs/Web/API/Worker/Worker

Получение и отправка данных

```
const params = {
    count: 10,
    size: 300
}

worker.postMessage(JSON.stringify(params));
```

https://developer.mozilla.org/en-US/docs/Web/API/Worker/Worker

Возможные типы данных

- Строки

Возможные типы данных

- Строки
- Любые сериализуемые объекты

Возможные типы данных

- Строки
- Любые сериализуемые объекты
- ArrayBuffer

Важно помнить

Данные передаваемы между потоками КОПИРУЮТСЯ!

Важно помнить

Данные передаваемы между потоками КОПИРУЮТСЯ!

50Mb = 300-500ms

Получение и отправка данных

Передача управления из потока в поток (TransferList)

```
const data = new ArrayBuffer(1000);

worker.postMessage(data, [data]);

console.log(data.byteLength); // 0
```

Важно помнить

50Mb = 10-20ms

Как создается и конфигурируется Web Worker?

```
const someTask = params => {/* just do it */};

self.addEventListener('message', event => {
    const result = someTask(event);

    self.postMessage(result);
}, false);
```

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

```
const someTask = params => {/* just do it */};

self.addEventListener('message', event => {
    const result = someTask(event);

    self.postMessage(result);
}, false);
```

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**
- к объектам **XMLHttpRequest/WebSocket**

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**
- к объектам **XMLHttpRequest/WebSocket**
- **setTimeout / setInterval**

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**
- к объектам **XMLHttpRequest/WebSocket**
- **setTimeout / setInterval**
- **name** – имя воркера

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**
- к объектам **XMLHttpRequest/WebSocket**
- **setTimeout / setInterval**
- **name** – имя воркера
- **importScripts([])** – метод для асинхронной подгрузки скриптов

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope есть доступ к:

- к объекту **location**
- к объекту **navigator**
- к объектам **XMLHttpRequest/WebSocket**
- **setTimeout / setInterval**
- **name** – имя воркера
- **importScripts([])** – метод для асинхронной подгрузки скриптов
- к другим видам сервисов и объектов (см. ссылку)

https://developer.mozilla.org/ru/docs/Web/API/DedicatedWorkerGlobalScope

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope нет доступа:

- к **DOM**

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope нет доступа:

- к **DOM**
- к объекту **document**

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope нет доступа:

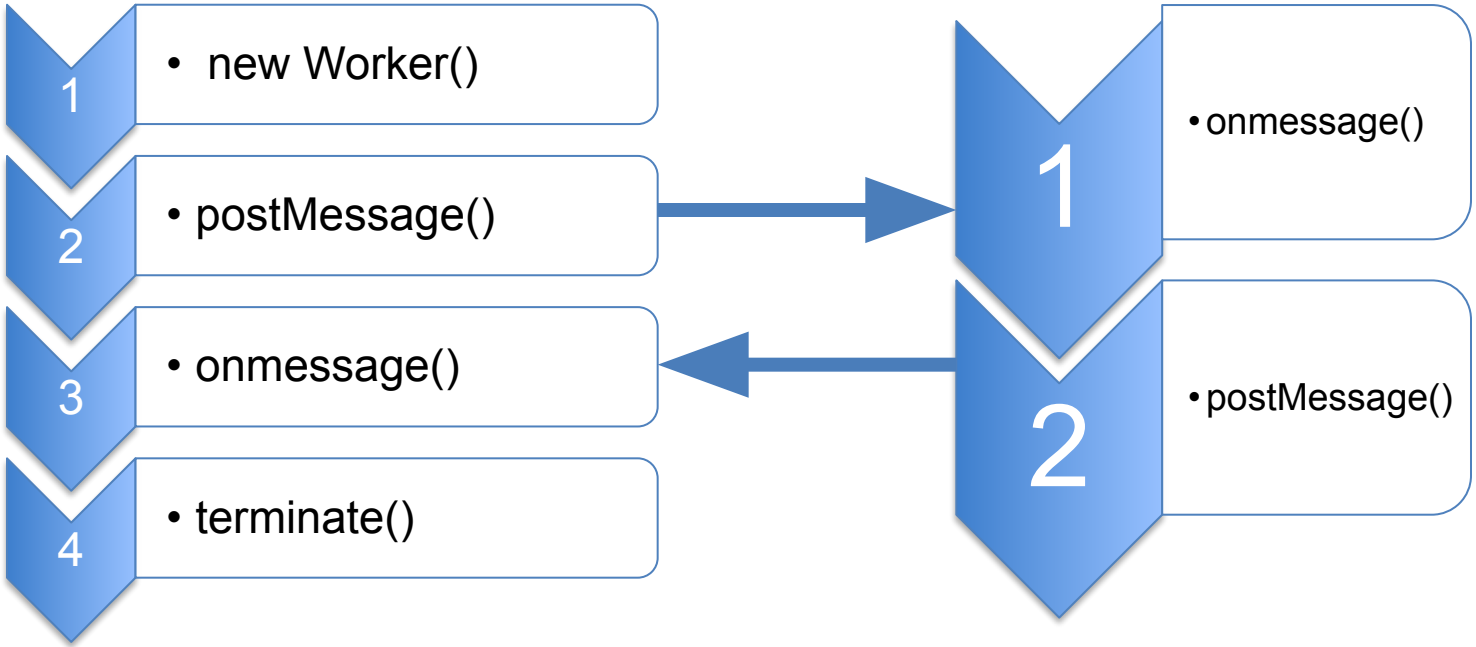
- к **DOM**
- к объекту **document**
- к объекту **window**

Как создается и конфигурируется Web Worker?

В Dedicated Worker Global Scope нет доступа:

- к **DOM**
- к объекту **document**
- к объекту **window**
- к объекту **parent**

Как это выглядит



Выводы

- **Web Worker** - это внешний подгружаемый и исполняемый файл

Выводы

- **Web Worker** - это внешний подгружаемый и исполняемый файл
- Это изолированный процесс со своей областью видимости

Выводы

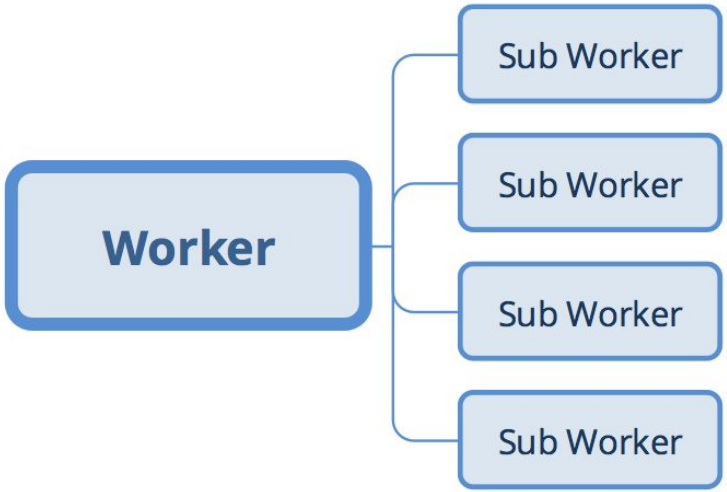
- **Web Worker** - это внешний подгружаемый и исполняемый файл
- Это изолированный процесс со своей областью видимости
- Выполнение кода и задач происходит параллельно

Выводы

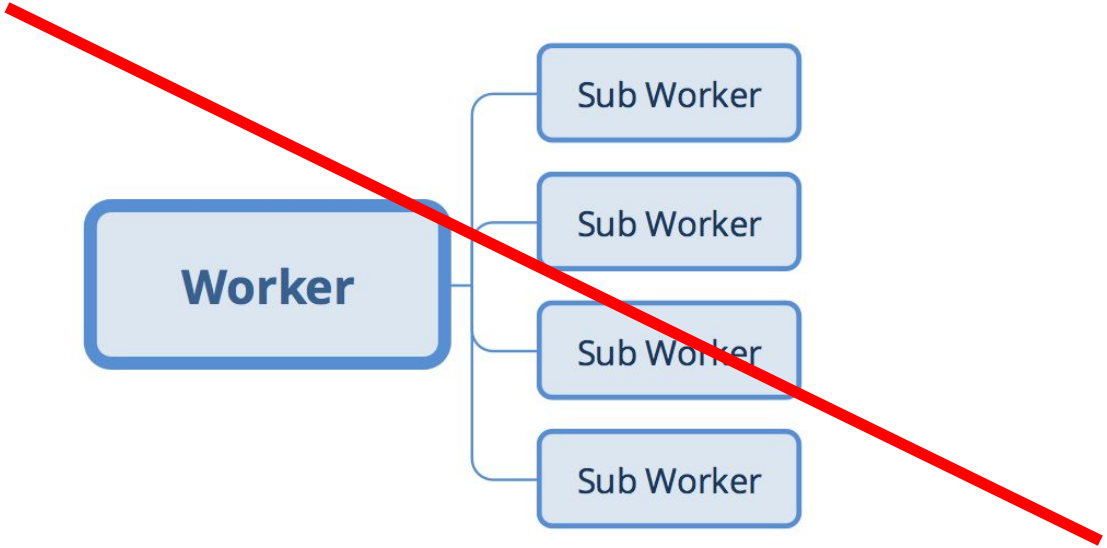
- **Web Worker** - это внешний подгружаемый и исполняемый файл
- Это изолированный процесс со своей областью видимости
- Выполнение кода и задач происходит параллельно
- Коммуникация построена на основе сообщений

- **Это наше настоящее!**

Sub Workers



Sub Workers



Демо №2

Рассмотрим пример с Web Worker-ом!

https://github.com/shanhaichik/angular-meetup-worker-demo

Shared Workers

Преимущества Shared Workers

- Синхронизация между вкладками без костылей

Преимущества Shared Workers

- Синхронизация между вкладками без костылей
- Оптимизация сетевых ресурсов и ресурсов системы

Преимущества Shared Workers

- Синхронизация между вкладками без костылей
- Оптимизация сетевых ресурсов и ресурсов системы
- Борьба с человеком вкладкой
- и многие другие…

Получение и отправка данных

```
const worker = new SharedWorker('getMyWorker.js');

worker.port.onmessage = event => {
    console.log(event.data);
}

worker.port.start();
```

https://developer.mozilla.org/en-US/docs/Web/API/SharedWorker

Получение и отправка данных

```
const worker = new SharedWorker('getMyWorker.js');

worker.port.onmessage = event => {
    console.log(event.data);
}

worker.port.start();
```

https://developer.mozilla.org/en-US/docs/Web/API/SharedWorker

Получение и отправка данных

```
const worker = new SharedWorker('getMyWorker.js');

worker.port.onmessage = event => {
    console.log(event.data);
}

worker.port.start();
```

https://developer.mozilla.org/en-US/docs/Web/API/SharedWorker

Получение и отправка данных

```
onconnect = (e) => {
    const port = e.ports[0];

    port.addEventListener('message', function(e) {
        port.postMessage('hi');
    });

    port.start();
}
```

https://developer.mozilla.org/en-US/docs/Web/API/SharedWorker

Web Workers в Angular

Демо №3

Рассмотрим пример Angular приложения

https://github.com/shanhaichik/angular-workers-in-action

Web Worker в Angular - это

Web Worker в Angular – это механизм для выноса и запуска Angular **приложения** в отдельном потоке.

Решает следующие задачи

- Дает возможность распараллелить работу приложения

Решает следующие задачи

- Дает возможность распараллелить работу приложения
- Вынести сложные части бизнес логики в отдельный поток

Решает следующие задачи

- Дает возможность распараллелить работу приложения
- Вынести сложные части бизнес логики в отдельный поток
- Снять нагрузку с основного потока, тем самым обеспечить наиболее эффективное манипулирование с DOM и работу с анимацией, что улучшает работу UI и увеличивает FPS.

Решает следующие задачи

- Дает возможность распараллелить работу приложения
- Вынести сложные части бизнес логики в отдельный поток
- Снять нагрузку с основного потока, тем самым обеспечить наиболее эффективное манипулирование с DOM и работу с анимацией, что улучшает работу UI и увеличивает FPS.
- Закладывать более гибкую архитектуру приложения

Решает следующие задачи

- Дает возможность распараллелить работу приложения
- Вынести сложные части бизнес логики в отдельный поток
- Снять нагрузку с основного потока, тем самым обеспечить наиболее эффективное манипулирование с DOM и работу с анимацией, что улучшает работу UI и увеличивает FPS.
- Закладывать более гибкую архитектуру приложения
- Переиспользование. Модули запускаемые в Web Worker-е просто запускаются и без него, как обычные модули приложения.

Конфигурация Web Worker в Angular

Шаг 1: Установить нужные зависимости.

npm install -S @angular/platform-webworker

@angular/platform-webworker-dynamic

Конфигурация Web Worker в Angular

Шаг 2: Обновить импорты основного модуля

в файле app.module.ts

заменяем BrowserModule

на WorkerAppModule

из пакета @angular/platform-webworker

Конфигурация Web Worker в Angular

Шаг 3: Добавить загрузчик worker-а

в файле main.ts

заменяем platformBrowserDynamic().bootstrapModule(AppModule);

на bootstrapWorkerUi(’worker.bootstrap.bundle.js').then(afterBootstrap);

из пакета @angular/platform-webworker

Конфигурация Web Worker в Angular

Шаг 4: Создать файл worker-a для запуска приложения

```
import 'polyfills.ts';
import '@angular/core';
import '@angular/common';

import { platformWorkerAppDynamic } from
'@angular/platform-webworker-dynamic';
import { AppModule } from './app/app.module';

platformWorkerAppDynamic().bootstrapModule(AppModule);
```

Вот и все, вы в Треде!

Ограничения

- Такие же ограничения как и обычного Web Worker-а (DOM, window и т.д.)

Ограничения

- Такие же ограничения как и обычного Web Worker-а (DOM, window и т.д.)

Но есть нюанс ☺

Ограничения

- Такие же ограничения как и обычного Web Worker-а (DOM, window и т.д.)

Но есть нюанс ☺

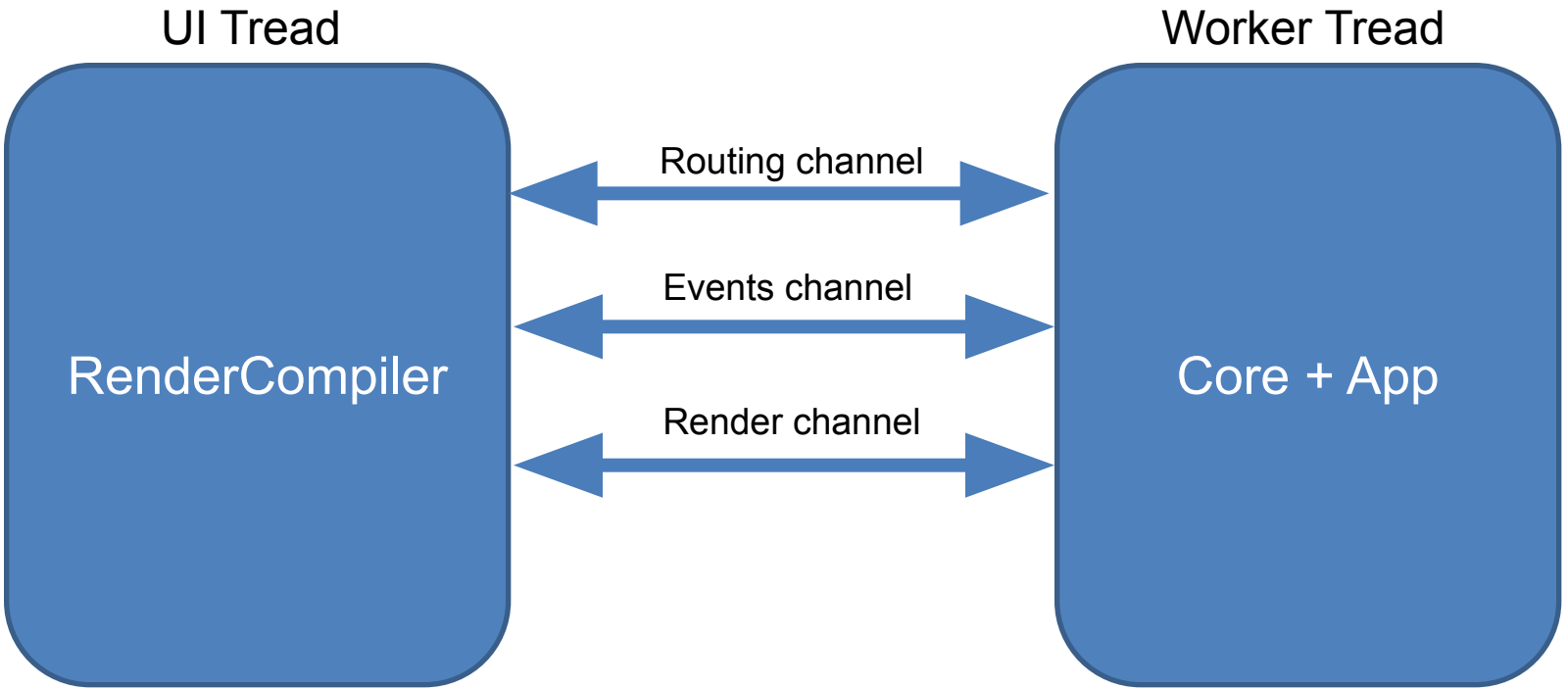
- В приложениях работающих в Web Worker-е есть возможность производить манипуляции с DOM
  - Используя возможность **Renderer/Renderer2**

Ограничения

- Такие же ограничения как и обычного Web Worker-а (DOM, window и т.д.)

Но есть нюанс ☺

- В приложениях работающих в Web Worker-е есть возможность производить манипуляции с DOM
  - Используя возможность **Renderer/Renderer2**
  - С помощью **Data bindings**


UI Tread
Worker Tread
RenderCompiler
Routing channel
Events channel
Render channel
Core + App

Получение и отправка данных

- **MessageBus** - это абстракция низкого уровня, которая предоставляет API-интерфейс для связи с компонентами приложения в реальном времени. **(**MessageBus стоит использовать если мы хотим контролировать протокол обмена сообщениями)

Получение и отправка данных

- **MessageBus** - это абстракция низкого уровня, которая предоставляет API-интерфейс для связи с компонентами приложения в реальном времени. **(**MessageBus стоит использовать если мы хотим контролировать протокол обмена сообщениями)

- **MessageBroker** - это абстракция сообщений верхнего уровня, которая находится поверх MessageBus. Она используется, когда мы хотим выполнить код в воркере и хотим получить результат.

MessageBroker

MessageBroker в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  const bRef = ref.injector.get(ClientMessageBrokerFactory);

  // код…
  }
```

MessageBroker в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  // код…
  bRef.createMessageBroker(CHANNEL, RUN_IN_ZONE);
  // код…
  }
```

MessageBroker в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  // код…
  const fnArg = new FnArg(data, SerializerTypes.PRIMITIVE);
  const sendData = new UiArguments(channel, [fnArg]);
  // код…
  }
```

MessageBroker в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
   // код…
 appBrokerRef.runOnService(sendData, SerializerTypes.PRIMITIVE)
.then(result => {
    console.log('From worker: ', result);
  });;
  // код…
  }
```

MessageBroker в воркере

```
export class SomeComponent implements OnInit {
  constructor(private broker: ServiceMessageBrokerFactory {  }

  ngOnInit() {
    const broker = this.broker.createMessageBroker(CHANNEL);

    broker.registerMethod(CHANNEL, [SerializerTypes.PRIMITIVE], () => {
      // Код...
      return Promise.resolve('Ответ с воркера');
    }, SerializerTypes.PRIMITIVE);
  }
}
```

MessageBus

MessageBus в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  const bRef = ref.injector.get(MessageBus);

  // код…
  }
```

MessageBus в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  // код…
  bRef.initChannel(CHANNEL);
  // код…
  }
```

MessageBus в основном треде

```
bootstrapWorkerUi('worker.js').then(afterBootstrap);

function afterBootstrap(ref: PlatformRef) {
  // код…
  bRef.to(CHANNEL).emit(sendData);
  // код…
  }
```

MessageBus в воркере

```
export class SomeComponent implements OnInit {
  constructor(private broker: MessageBus){  }

  ngOnInit() {
    this.broker.initChannel(CHANNEL);

    this.broker.from(CHANNEL).subscribe(message => {
       console.log(message);
   });
  }
}
```

Роутинг

Роутинг

```
import {
        bootstrapWorkerUi,
        WORKER_UI_LOCATION_PROVIDERS
} from '@angular/platform-webworker';


bootstrapWorkerUi('webworker.js', [WORKER_UI_LOCATION_PROVIDERS]);
```

Роутинг

```
providers: [
  WORKER_APP_LOCATION_PROVIDERS,
  {provide: APP_BASE_HREF, useValue : '/' }
],
```

Общие выводы

- Использовать Web Worker-ы не только классно, но и полезно. Это наше настоящее и не стоит их бояться.

Общие выводы

- Использовать Web Worker-ы не только классно, но и полезно. Это наше настоящее и не стоит их бояться.
- Web Worker-ы в Angular еще экспериментальны, сложны и многие вещи меняются от версии к версии, но это хорошая почва для экспериментов и новых архитектурных решений.