Web Workers

Борис Ребров

ЦРИТО
Центр Развития
ИТ-Образования

МФТИ

@mail.ru group

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях
- ДЗ!

Что такое воркеры

- Технология, позволяющая создавать отдельный поток вычислений
- Web Workers API позволяет создавать воркеры и обмениваться с ними данными
- Воркеры не имеют доступ к DOM, большинству API и глобальному объекту страницы
- Все данные при передаче будут скопированы* http://bit.ly/structure-cloning

* за исключением специфичных «transferable objects»

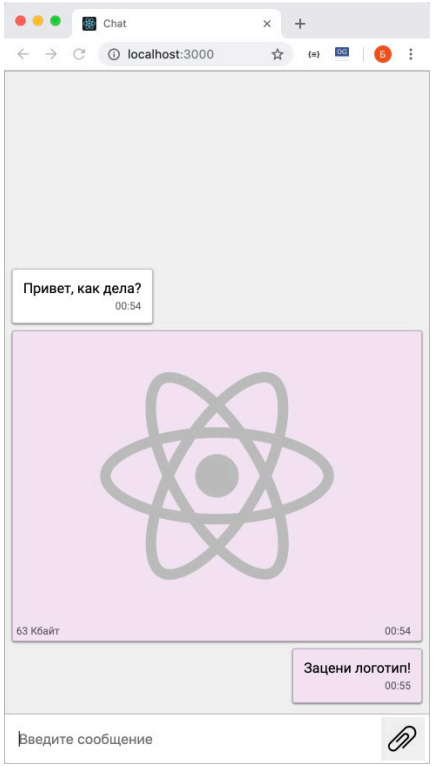
Что доступно воркерам

- fetch/WebSocket/XMLHttpRequest*
- Promise
- Cache
- importScripts()
- Notifications
- FileAPI
- IndexedDB
- WorkerLocation/WorkerNavigator
- http://bit.ly/api-in-worker

Какие бывают воркеры

	Dedicated Worker	Shared Worker	Service Worker
Область взаимодействия	Родительский процесс	Процессы из одного источника (origin)	Ограниченный набор страниц (origin + path)
Управление	postMessage	postMessage	postMessage, события целевых страниц
Завершение	Родительский процесс, worker.terminate(), self.close()	Родительский процесс*	registration.unregister()

Dedicated Worker



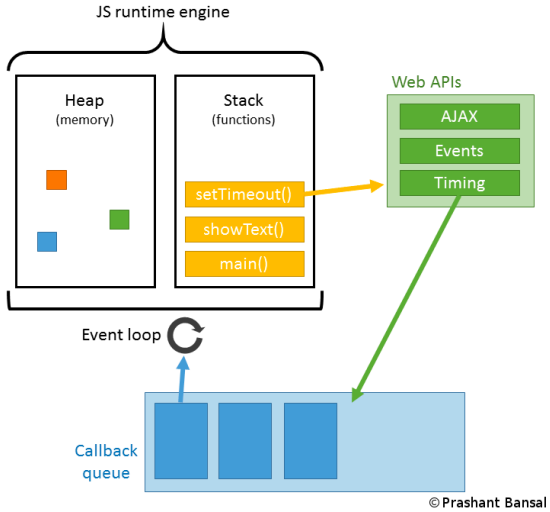
postMessage

Проблема

- JS выполняется в однопоточной среде
- Выполнение блокирует UI

```
1. window.addEventListener('click', (e) => {
2.     new Array(1e9).forEach(() => {
3.         return 1 + 100;
4.     });
5. });
```

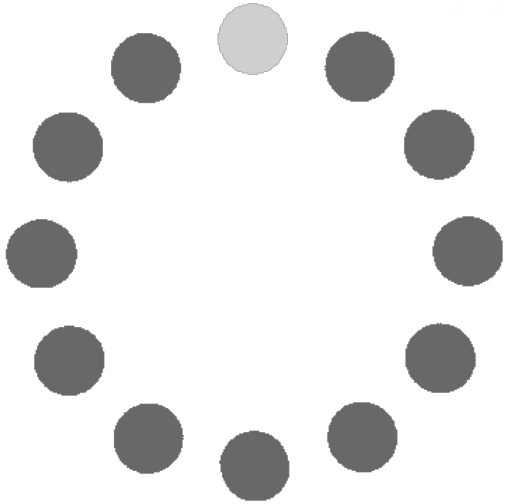
- Тяжелые операции за нас ни кто не сделает



Варианты решения

- Показывать лоадер

```
1. window.addEventListener('click', (e) => {
2.     document.body.classList.add(‘loading’);
3.     new Array(1e9).forEach(() => {
4.         return 1 + 100;
5.     });
6.     document.body.classList.remove(‘loading’);
7. });
```

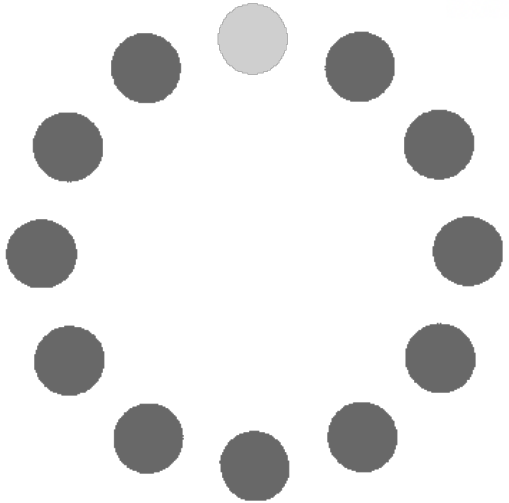


Варианты решения

- Вынести вычисления в воркер

```
self.onmessage = (e) => {
    new Array(1e9).forEach(() => {
        return 1 + 100;
    });
    self.postMessage('success');
});
```

```
const worker = new Worker('/heavy.js');
window.addEventListener('click', (e) => {
    worker.postMessage('test');
});
```



- … показать лоадер

Dedicated Worker

- Создание

```
1. const worker = new Worker('/heavy.js');
2. worker.port.open();
```

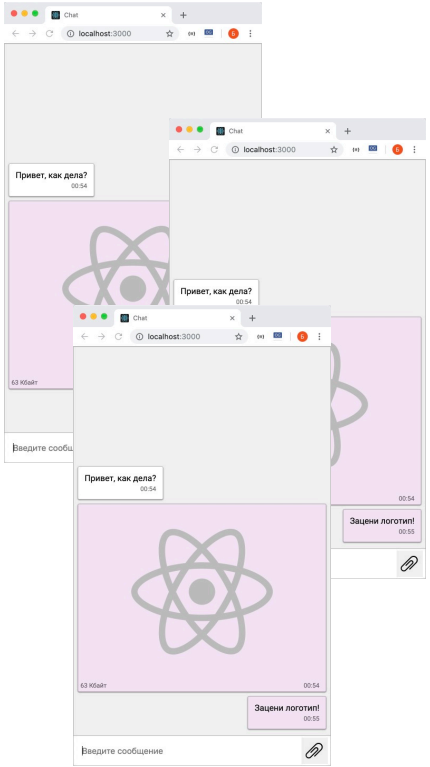
- Обработка событий

```
1. worker.addEventListener('message', (event) => {…});
2. worker.addEventListener('error', (event) => {…});
3. worker.onmessage = (event) => {…};
```

- Взаимодействие

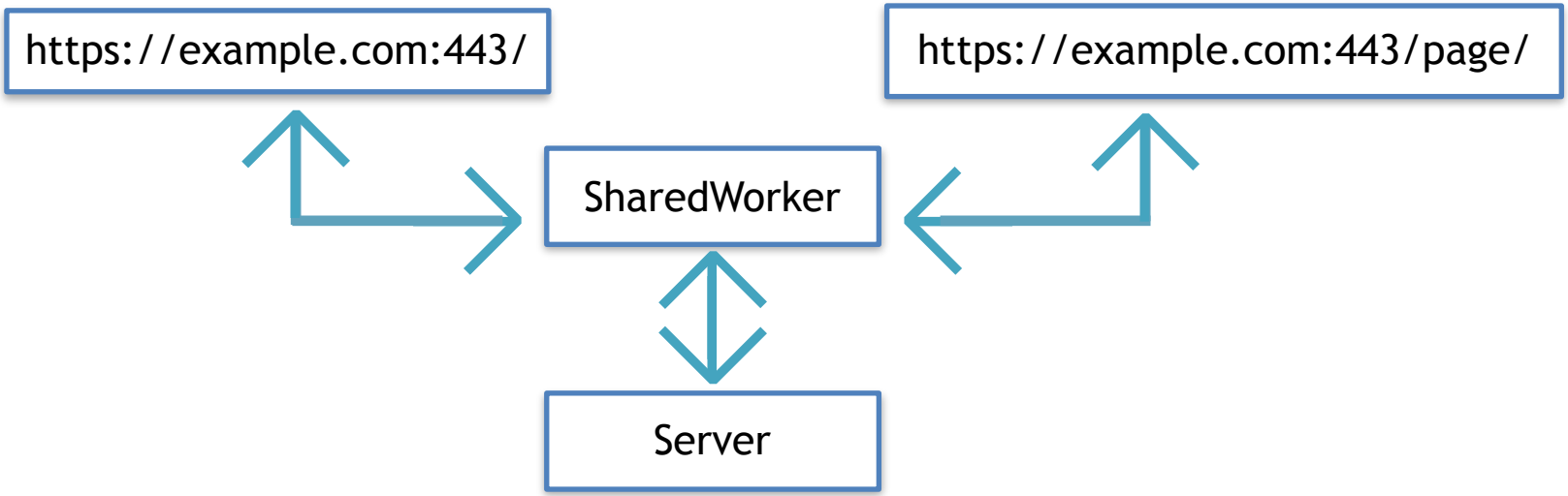
```
1. worker.postMessage(message, [transfer]);
2. worker.terminate();
```

Shared Worker



Shared Worker – Дополнительные возможности

- Результаты работы могут использоваться нескольким контекстам с общим происхождением (origin)
- Простое взаимодействие между контекстами
- Совместное использование сетевых соединений



Shared Worker

- Создание

```
1. const worker = new SharedWorker(‘/shared.js’);
2. worker.port.start();
```

- Обработка событий

```
1. //external context
2. worker.port.addEventListener(‘message’, (event) => {…});
3. worker.onmessage = (event) => {…};
```

```
1. //worker context
2. self.addEventListener(‘connect’, (event) => {
3.     const port = event.source;
4.     port.addEventListener('message', (event) => {…});
5.     port.start();
6. });
```

- Взаимодействие

```
1. worker.postMessage(message, [transfer]);
```

Shared Worker – особенности

- Идентификатор воркера – URL

```
1. new SharedWorker('/test.js');
2. new SharedWorker('/test.js?anticache=ad31');
```

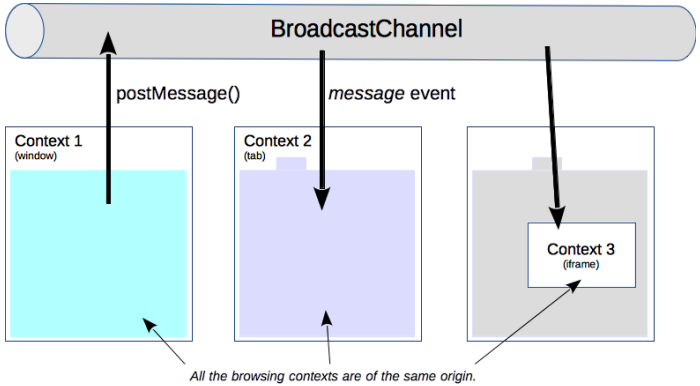
- Воркер не отождествлен с «родительским» контекстом chrome://inspect/#workers
- Нужно контролировать порты в контексте воркера
- Назначение port.onmessage неявно вызывает port.open()

BroadcastChannel

- Использовать SharedWorker как pubsub-эмиттер избыточно

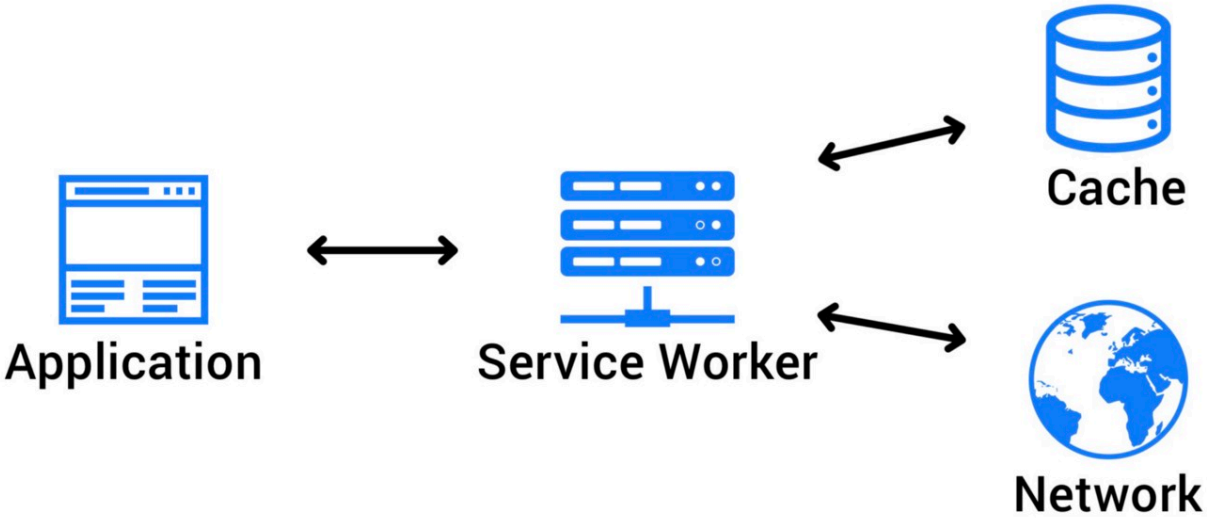
```
1. const channel = new BroadcastChannel('my_channel');
2. channel.postMessage('test');
3. channel.addEventListener('message', () => {…});
4. channel.close()
```

- Сообщения будут доставлены всем подписчикам канала, за исключением отправителя



Service Worker

http://bit.ly/service-workers-intro



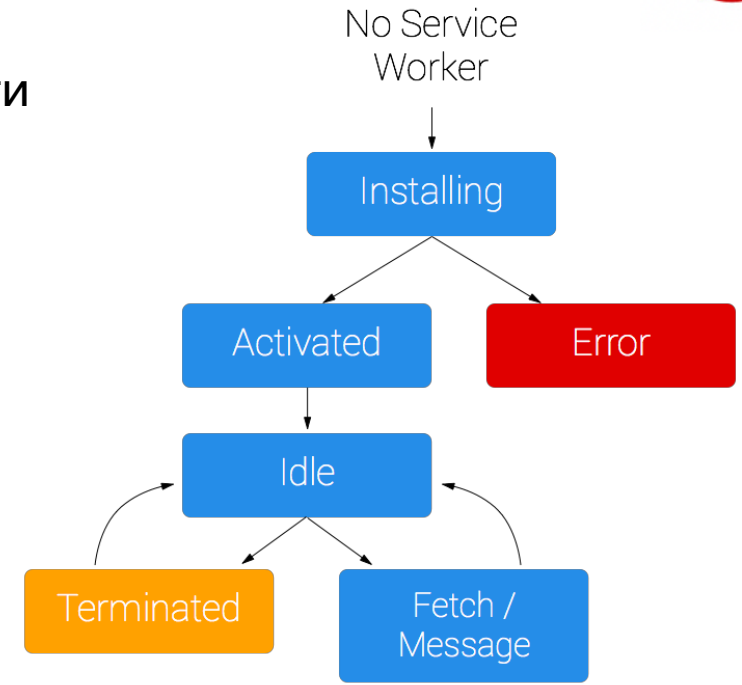
Service Worker – регистрация

```
1.  if ('serviceWorker' in navigator) {
2.    window.addEventListener('load', function() {
3.      navigator.serviceWorker.register('/sw.js').then(function(registration) {
4.        // Registration was successful
5.        console.log('Registration successful with scope:', registration.scope);
6.      }, function(err) {
7.        // registration failed :(
8.        console.log('ServiceWorker registration failed:', err);
9.      });
10.   });
11. }
12.
```

- Воркер регистрируется для определенного пространства страниц (scope)
- Scope по-умолчанию зависит от URL файла воркера (origin + path)
- Работает только с https (или localhost)

Service Worker – жизненный цикл

- Воркер живет вечно*
- При простое он выгружается из памяти
- При поступлении события вновь загружается



* пока не упадет или не будет принудительно остановлен

Service Worker

- Обработка событий

```
1. self.addEventListener('install', (event) => {…});
```

- События
  - install
  - activate
  - fetch
  - message
  - notificationclick
  - notificationclose
  - push
  - pushsubscriptionchange

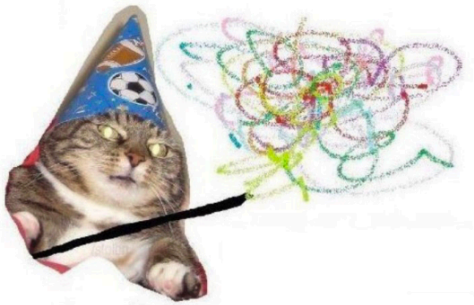
Service Worker – пример

- Перехват сетевого запроса и ответ из кеша

```
1.  self.addEventListener('fetch', function(event) {
2.    event.respondWith(
3.      caches.match(event.request)
4.        .then(function(response) {
5.          // Cache hit - return response
6.          if (response) {
7.            return response;
8.          }
9.          return fetch(event.request);
10.       }
11.     )
12.   );
13. });
```

Домашнее задание № 1

- Перенести взаимодействие с сервером в SharedWorker
- Любое другое преобразование изображений с использованием OffsceenCanvas



Срок сдачи

- до 7 марта

Спасибо за внимание!