PROGRAM SZKOLENIA JĘZYKI

Szkolenie JavaScript zaawansowany

Opanujesz prototypy, Proxy, Web Workers i wzorce asynchroniczne, a potem zbudujesz SPA bez frameworka.

CZAS TRWANIA	POZIOM	FORMA
4 dni	zaawansowany	warsztat dla zespołu, zdalnie lub stacjonarnie

Aplikacja działa, ale przy większych danych przycina się interfejs, pamięć rośnie z każdą godziną, a równoległe żądania kończą się w losowej kolejności? To problemy, których nie rozwiąże kolejna biblioteka. Potrzebne jest zrozumienie, jak JavaScript zarządza pamięcią, wątkami i kolejnością zadań.

Na szkoleniu poznasz mniej oczywiste części języka i platformy: prototypy, iteratory i generatory, `Proxy` i `Reflect`, Web Workers, Streams API i narzędzia do mierzenia wydajności. Sprawdzisz też, co wchodzi do standardu w latach 2026-2027, w tym Temporal, który zastępuje obiekt `Date`.

Ostatni dzień poświęcasz na aplikację SPA w czystym JavaScript (vanilla.js). Sam napiszesz routing, stan i komponenty, więc zobaczysz, co robi za Ciebie framework i kiedy go w ogóle nie potrzebujesz.

PROGRAM

4 dni

- Programowanie obiektowe od środka
  - Prototypy i łańcuch prototypów: co naprawdę robi `class`
  - `this` pod kontrolą: `call`, `apply` i `bind`
  - Pola i metody prywatne `#` oraz elementy `static`
  - Bloki statyczne i sprawdzanie `#pole in obiekt`
  - Kompozycja zamiast dziedziczenia i mixiny
- Moduły i ładowanie kodu
  - Dynamiczny `import()` i ładowanie kodu na żądanie
  - Top-level `await`
  - Import maps: moduły w przeglądarce bez bundlera
- Iteratory i generatory
  - Protokół iteracji i `Symbol.iterator`
  - Generatory: `function*`, `yield` i leniwe sekwencje
  - Generatory asynchroniczne i pętla `for await...of`
  - Pomocnicze metody iteratorów i `Array.fromAsync`

- Kolekcje
  - `Map` i `Set` a zwykłe obiekty i tablice: kiedy co wybrać
  - `WeakMap`, `WeakSet` i dane przypięte do obiektu
  - `WeakRef` i `FinalizationRegistry`
  - Nowe metody `Set`: `union`, `intersection`, `difference`
  - Grupowanie do Mapy: `Map.groupBy`, `getOrInsert` i `getOrInsertComputed`
- Wyrażenia regularne
  - Składnia, flagi, `test`, `replace` i `matchAll`
  - Grupy nazwane i lookbehind
  - Flaga `v` i klasy znaków Unicode
  - Wyrażenia, które zawieszają aplikację (ReDoS)
  - Bezpieczne wyrażenia z danych użytkownika: `RegExp.escape`
- Metaprogramowanie
  - Symbole i dobrze znane symbole (well-known symbols)
  - Deskryptory właściwości, `Object.freeze` i `Object.seal`
  - `Proxy`: walidacja, obserwowanie zmian, leniwe obiekty
  - `Reflect` jako para dla pułapek `Proxy`
- Wzorce asynchroniczne
  - Kolejność mikro- i makrozadań w praktyce: `queueMicrotask`
  - `Promise.any` i `Promise.race`: pierwszy wynik wygrywa
  - Ograniczanie współbieżności i kolejki zadań
  - Ponawianie z opóźnieniem (retry z backoffem)
  - Anulowanie: `AbortController`, `AbortSignal.timeout` i `AbortSignal.any`, przekazywanie sygnału w dół
  - `Promise.withResolvers` i `Promise.try`
- Web Workers
  - Przenoszenie obliczeń poza wątek główny
  - `postMessage`, `structuredClone` i obiekty transferowalne
  - Kiedy worker się opłaca, a kiedy nie
- Streams API
  - `ReadableStream`, `WritableStream` i `TransformStream`
  - Przetwarzanie odpowiedzi `fetch` kawałek po kawałku
  - `TextDecoderStream` i przetwarzanie dużych plików
- Pamięć i wydajność
  - Garbage collector i typowe wycieki pamięci
  - Zakładka Memory w DevTools i zrzuty sterty (heap snapshot)
  - Profilowanie w zakładce Performance i Performance API
  - `requestAnimationFrame`, debounce i throttle

- Co nowego w standardzie (ES2026 i ES2027)
  - Temporal: strefy czasowe, kalendarze i czas trwania
  - Zarządzanie zasobami: `using` i `Symbol.dispose`
  - `Error.isError`, `Math.sumPrecise` i Base64 w `Uint8Array`
  - Wsparcie w przeglądarkach (Baseline) i Node.js, kiedy polyfill
- Projekt: aplikacja SPA bez frameworka
  - Architektura: podział na warstwy widoku, stanu i danych
  - Routing na History API: `pushState` i zdarzenie `popstate`
  - Stan aplikacji i ponowne renderowanie widoków
  - Komponenty jako moduły ES z własnym widokiem i zdarzeniami
  - Komunikacja między komponentami: `EventTarget` i `CustomEvent`
  - Testy stanu i routingu w Vitest

DLA KOGO?

- Dla programistów z co najmniej rocznym doświadczeniem w JavaScript. Jeśli dopiero zaczynasz, najpierw przejdź szkolenie JavaScript od podstaw.
- Dla programistów, którzy rozwiązują problemy z wydajnością i pamięcią w dużych aplikacjach.
- Dla autorów bibliotek i narzędzi, którzy potrzebują mniej oczywistych części języka.
- Dla osób, które chcą zrozumieć, jak działa aplikacja SPA pod spodem.

Wzorce oparte na `Proxy` i iteratorach rozwiniesz na szkoleniu Wzorce projektowe, Core Web Vitals na Wydajność aplikacji, a testy na Testy jednostkowe.

CO MUSISZ UMIEĆ?

- Programować w JavaScript: klasy, moduły ES, domknięcia i `this`.
- Pisać kod asynchroniczny z `Promise` oraz `async` i `await`.
- Pracować z DOM, Fetch API i narzędziami deweloperskimi przeglądarki.
- Uruchamiać projekt przez npm w terminalu.

CZEGO SIĘ NAUCZYSZ?

- Zrozumiesz prototypy, ustawisz `this` przez `call`, `apply` i `bind` i użyjesz pól prywatnych oraz elementów `static`.
- Załadujesz moduły na żądanie i użyjesz import maps bez bundlera.
- Napiszesz wyrażenia regularne z grupami nazwanymi, które nie zawieszą aplikacji na złośliwych danych.
- Napiszesz własne iteratory i generatory, także asynchroniczne.
- Dobierzesz właściwą kolekcję do problemu i unikniesz wycieków przez słabe referencje.
- Użyjesz `Proxy`, `Reflect` i symboli do walidacji i obserwowania danych.
- Zastosujesz kompozycję zamiast głębokiego dziedziczenia.
- Opanujesz współbieżność: wyścig obietnic, kolejki, ponawianie i anulowanie zadań.

- Przeniesiesz ciężkie obliczenia do Web Workera.
- Przetworzysz duże dane strumieniowo przez Streams API.
- Znajdziesz wyciek pamięci i wąskie gardło w DevTools.
- Użyjesz nowości ES2026 i ES2027 z Temporal włącznie i sprawdzisz ich wsparcie.
- Zabezpieczysz stan i routing aplikacji testami w Vitest.
- Zbudujesz aplikację SPA z routingiem, stanem i komponentami bez frameworka.

OPINIE UCZESTNIKÓW

> *„Gruntownie, krok po kroku stworzyliśmy aplikację sklepu internetowego, po drodze pisząc testy jednostkowe, e2e, web komponenty, autorski backend oparty na Node.js + Express.js, jak również wykorzystaliśmy narzędzia CI."*
>
> Łukasz, JavaScript Developer (WarsawJS MasterClass)

> *„The whole team was able to learn from the mini-workshops and presentations he regularly shared with us, covering advanced Git tricks, React, JavaScript and more."*
>
> Paul Ngei, współpracownik z zespołu w Box (rekomendacja na LinkedIn)

TRENER

Piotr Kowalski

W branży IT od 2008 roku. Od 2014 jest mentorem dla wielu developerów. Jako trener spędził na sali szkoleniowej kilka tysięcy godzin, a na LinkedIn ma 52 rekomendacje. Każde jego szkolenie to dawka solidnej wiedzy oraz humoru.

Więcej: piecioshka.pl/piotr-kowalski

ZORGANIZUJMY TO SZKOLENIE W TWOIM ZESPOLE

Program dopasuję do Waszego projektu i poziomu grupy. Napisz, z czym pracujecie, a odpowiem z propozycją terminu i zakresu.

KONTAKT	**piecioshka.pl/kontakt**
E-MAIL	**piecioshka+blog [małpka] gmail.com**
SZKOLENIE	**piecioshka.pl/szkolenia/javascript-zaawansowany/**