BO·TT·EGA
IT minds

Program szkolenia:

Web Performance Optimisation - szybsze strony internetowe

Informacje:

Nazwa:	**Web Performance Optimisation - szybsze strony internetowe**
Kod:	**web-performance**
Kategoria:	Web Platform
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	20% wykłady / 80% warsztaty

Dziesiątki ćwiczeń praktycznych pokazujących niemal każdy aspekt optymalizacji.

Czego się nauczę?

- Znajdować wąskie gardła podczas ładowania strony oraz po jej załadowaniu w trakcie animacji
- Optymalizować HTML, JS, CSS, obrazki oraz czcionki
- Mierzyć rezultaty wprowadzanych optymalizacji
- Efektywnie korzystać z narzędzi tj. Chrome DevTools pod kątem optymalizacji stron internetowych
- Automatyzować powtarzalne optymalizacje

Dlaczego warto się tego nauczyć?

- Zarabiaj więcej pieniędzy na swoich stronach. Szybsze serwisy to lepszy UX oraz wyższe rankingi SEO prowadzące do większej ilości konwersji
- Oszczędzaj własne pieniądze. Zoptymalizowane strony zużywają mniej zasobów serwerowych
- Oszczędzaj pieniądze swoich użytkowników. Zoptymalizowane strony zużywają mniej zasobów tj. mobilny plan internetowy czy bateria w telefonie
- Szanuj czas swoich użytkowników. Szybsze strony nie każą czekać swoim użytkownikom
- Bądź dumny ze swojej pracy. Szybsze strony internetowe to źródło satysfakcji dla ich autorów

Forma zajęć
Na starcie uczestnicy dostają stronę, która ładuje się ponad 10 sekund w sieci 3G. W trakcie kolejnych ćwiczeń stopniowo przyspieszamy stronę, tak aby ładowała się poniżej 1 sekundy.
Następnie usprawniamy wszystkie animacje tak aby strona renderowała się płynnie nawet na starszych telefonach. Gdy poznamy już klasyczne techniki optymalizacji, robimy analizę aplikacji zbudowanych z użyciem nowoczesnych frameworków do aplikacji typu Single Page App. Dodatkowo robimy audyt jednego z publicznie dostępnych serwisów internetowych pod kątem poznanych optymalizacji.

Zalety szkolenia:

- Przykłady z realnych systemów
- Sprawdzone techniki
- Trener jest praktykiem z wieloletnim doświadczeniem

BOTTEGA IT minds

Szczegółowy program:

1. Ogólne techniki optymalizacji ładowania

- 1.1. Performance golden rule
- 1.2. Theory of Constraints
- 1.3. Optymalizacje czasu ładowania
- 1.4. Optymalizacje runtime
- 1.5. Perceived performance

2. Optymalizacja na poziomie sieci

- 2.1. Bandwidth vs latency
- 2.2. HTTP cache (Cache-Control, ETag, strategie cachowania)
- 2.3. Resource hints (preload, prefetch)
- 2.4. Service Workers - implementacja przykładowej strategii cache'owania
- 2.5. HTTP/2 - wpływ na techniki optymalizacji
- 2.6. HTTP/2 push
- 2.7. Reguła 14kB
- 2.8. Kompresja (gzip, brotli)
- 2.9. CDN

3. Rendering Pipeline

- 3.1. Critical Rendering Path
- 3.2. DOM
- 3.3. CSSOM
- 3.4. Render Tree
- 3.5. Layout
- 3.6. Paint

BOTTEGA IT minds

4. Ładowanie CSS

- 4.1. Blokowanie renderowania
- 4.2. Minifikacja (cssnano, csso)
- 4.3. Sekcja head
- 4.4. Skrócony zapis CSS
- 4.5. Płaskie selektory
- 4.6. Usuwanie nieużywanego CSS
- 4.7. Usuwanie duplikacji
- 4.8. Mobile first jako technika optymalizacji
- 4.9. @import vs link
- 4.10. Krytyczny CSS (above the fold)

5. Ładowanie JS

- 5.1. Blokowanie parsowania
- 5.2. SPOF (Single Point of Failure)
- 5.3. Minifikacja (UglifyJS2)
- 5.4. Łączenie skryptów (bundlers)
- 5.5. defer vs async
- 5.6. defer vs skrypty na końcu body
- 5.7. Lekkie alternatywy dla popularnych bibliotek
- 5.8. Vanilla JS

6. Optymalizacja Single Page Apps

- 6.1. Client Side Render vs Server Side Render
- 6.2. Porównanie czasu ładowania popularnych frameworków JS
- 6.3. Czas parsowania JS
- 6.4. Koszt transpilacji (babel, babel-preset-env)

BOTTEGA IT minds

- 6.5. Code splitting
- 6.6. Optymalizacje renderowania (Virtual DOM)
- 6.7. Memoization
- 6.8. Microbenchmarking (benchmark.js)
- 6.9. Tryb deweloperski vs tryb produkcyjny
- 6.10. HTML streaming vs JSON streaming
- 6.11. App shell

7. Ładowanie czcionek

- 7.1. Wybór optymalnej czcionki
- 7.2. Konwersja typów czcionek (ttf2eot, ttf2woff, ttf2woff2)
- 7.3. Optymalna kaskada @font-face
- 7.4. Font subsetting (fontmin)
- 7.5. Flash of Unstyled Text vs Flash of Invisible Text
- 7.6. Nieblokujące ładowanie (font-display, Font Loading API)

8. Ładowanie obrazków

- 8.1. Raster vs vector
- 8.2. Obrazki w HTML vs obrazki w CSS
- 8.3. img@srcset
- 8.4. picture
- 8.5. Optymalizacja JPEG (imagemin)
- 8.6. WebP (imagemin-webp)
- 8.7. Lazy loading (blazy, IntersectionObserver)
- 8.8. Sprites (svg-sprite)

9. Optymalizacje renderowania (scroll, animacje)

- 9.1. 60fps

BOTTEGA IT minds

9.2. Jank i analiza wolnych ramek

9.3. Forced synchronous layout/Layout thrashing

9.4. requestAnimationFrame

9.5. Paint flashing

9.6. Animacje w osobnej warstwie

9.7. Animacje w CSS vs animacje w JS

10. Metryki

10.1. Time to First Byte

10.2. Time to First Paint

10.3. DOM Content Loaded

10.4. Time to First Meaningful Paint

10.5. Time to Interactive

10.6. Load Time

10.7. Speed Index

10.8. Custom metrics

10.9. Performance budget

10.10. Analiza wykresów typu waterfall i flame chart

10.11. Timing APIs (Navigation, Resource, Paint, User Timing)

11. Narzędzia

11.1. Chrome DevTools (Network, Performance, Audits, Rendering)

11.2. npm i Parcel jako narzędzia do budowania i automatyzacji optymalizacji

11.3. Narzędzia oparte o reguły (np. PageSpeed Insights, Lighthouse)

11.4. WebPageTest - synthetic testing

11.5. Continuous Web Performance Testing z webpagetest-api

11.6. Performance monitoring: Speedtracker, SpeedCurve