Optymalizacja rozmiaru witryn

Opis

Obrazki i skrypty JavaScript stanowią znaczący procent rozmiaru witryny. Spora część witryn responsywnych w ogóle nie optymalizuje zasobów. Przegląd technik minimalizacji zasobów. Warunkowe pobieranie obrazków, plików CSS (np. eCSSential), wykonywanie kodu JavaScript.

Plan

1. Problemy
    a. krótkie statystyki
    b. bardzo niewiele osób optymalizuje
    c. wg statystyk, tylko mała część stron RWD optymalizuje zasoby dla mobile!
    d. mobile first vs mobile last i dlaczego optymalizować dla mobile
2. Możliwe usprawnienia
    a. zasady z desktopa jako podstawa (yslow, pagespeed)
    b. Warunkowe zaciąganie assetów (obrazki, cssy, jsy)
        i. lazy loading
        ii. responsywne obrazki
        iii. ładowanie asynchroniczne skryptów
        iv. kolejnosc wykonywania JS
        v. dependencje JS (requirejs itp.)
        vi. pliki CSS (eCSSential*)*
    c. base64 i optymalizacja fontów (FOUT)
    d. wydajność CSS3
    e. onclick delay
    f. RESS
        i. Jak wykrywać?
    g. optymalizacja stron a CDN-y
3. Narzędzia/tooling
    a. robienie spritów - niektóre urządzenia mobilne mają ograniczenia dt sprite
    b. matchMedia
    c. southstreet
    d. grunt i narzędzia w CLI
        i. optymalizacja obrazków
        ii. kompilowanie sass/less
        iii. combining i minifikacja plików
    e. desktopowe aplikacje
        i. Hammer for Mac

        ii. CodeKit
        iii. Prepros
4. Testowanie (PageSpeed Insights, yslow, Chrome DevTools Audits)
    a. trzeba robić benchmarki przed i po dla klientów
    b. zewnętrzne serwisy do testowania
        i. mobitest
5. Zewnętrzne narzędzia do optymalizacji rozmiaru
    a. Akamai
    b. Mobify http://www.mobify.com/mobifyjs/
    c. Foresight http://www.cdnconnect.com/docs/foresightjs
    d. sencha.io

1. Problemy

a. statystyki

2. Możliwe usprawnienia

Dobre praktyki z desktopa

Yslow	Google Page Speed
● Make fewer HTTP requests ● Avoid empty src or href ● Compress components with gzip ● Put CSS at top ● Put JavaScript at bottom ● Avoid CSS expressions ● Reduce DNS lookups ● Minify JavaScript and CSS ● Avoid URL redirects ● Remove duplicate JavaScript and CSS ● Reduce the number of DOM elements ● Avoid HTTP 404 (Not Found) error ● Avoid AlphaImageLoader filter ● Do not scale images in HTML ● Make favicon small and cacheable ~ http://developer.yahoo.com/yslow/	● Leverage browser caching ● Enable compression ● Defer parsing of JavaScript ● Minimize request size ● Specify a cache validator ● Optimize images ● Minify JavaScript ● Minify HTML ● Specify image dimensions ● Specify a character set ● Specify a Vary: Accept-Encoding header ● Reduce request serialization ● Eliminate unnecessary reflows ● Avoid long-running scripts ● Avoid CSS @import ● Avoid bad requests ● Enable Keep-Alive ● Make landing page redirects cacheable ● Minify CSS ● Minimize redirects ● Optimize the order of styles and scripts ● Put CSS in the document head ● Remove query strings from static resources ● Serve resources from a consistent URL ● Serve scaled images

Lazy Loading

Warunkowe zaciąganie assetów (obrazki, cssy, jsy), asynchroniczne ładowanie skryptów

Require.js

Strona:
http://requirejs.org

Opis:
RequireJS is a JavaScript file and module loader. It is optimized for in-browser use, but it can be used in other JavaScript environments, like Rhino and Node.RequireJS is a JavaScript file and module loader. It is optimized for in-browser use, but it can be used in other JavaScript environments, like Rhino and Node.

Matchmedia

Strona:
http://dev.w3.org/csswg/cssom-view/#dom-window-matchmedia

Opis:
Matchmedia służy do wykonywania bloków JavaScript tylko w momencie, gdy spełniony jest warunek mediaquery.
Niestety słabe wsparcie (looking at you - IE) nadal wymaga posiłkowania się polyfillem:
https://github.com/paulirish/matchMedia.js/

responsywne obrazki

srcset.js
Biblioteka, która zaciąga obrazki zoptymalizowane pod dane urządzenie (rozdzielczość)
https://github.com/borismus/srcset-polyfill
<img srcset="picture_320.jpg 320w, picture_768.jpg 768w, picture_1280.png" alt="">

(RESS) ua-parser https://github.com/tobie/ua-parser/
Narzędzie do wykrywania urządzenia, które wykorzystuje user-agent stworzony przez Browserscope http://www.browserscope.org/. Zastosowanie tego narzędzia pozwala z poziomu serwera na wysyłanie zoptymalizowanego kontentu.

window.innerWidth -
if (@screenWidth <= 320) {$imgPath = ‘320’;}
else if (){}
….

Kompresja png
https://tinypng.com/
http://imageoptim.com/pl.html dla mac

Optymalizacja CSS

Używając preprocesorów należy być bardzo czujnym przy zagnieżdzaniu elementów i pamiętać o szybko rosnącym specificity i coraz trudniejszym utrzymaniu kodu, pamiętajmy też o szybkości selektorów.

Należy także pamiętać, że w niektórych przypadkach udogodnienia CSS3 nie są najbardziej optymalnym rozwiązaniem. Nie powinniśmy używać gradientów na dużym obszarze (np jako tła strony), ponieważ jest to bardzo niewydajne, zwłaszcza na mobilu.

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Writing_efficient_CSS

Lepszy User Experience

Pomiń 300ms https://github.com/ftlabs/fastclick

RESS

Poruszony w osobnym dokumencie - serwowanie innego kodu na podstawie urządzenia przez back-end.

Korzystanie z CDN

Serwisy takie jak Akamai pozwalają na agresywny caching lub serwowanie treści lub mediów z serwera znajdującego się fizycznie najbliżej użytkownika (na danym kontynencie). Powoduje to czasem problemy związane z wykrywaniem klienta w front- lub back-end, bo agresywne cachowanie powoduje, że CDN serwuje static content, a nie odpytujemy serwera z prawidłowym user agent string.

Na szczęście serwisy takie, jak Akamai dostarczają swoich narzędzi, dzięki którym można np. serwować inne assety na podstawie możliwości łącza lub wielkości viewportu.

Base64

Optymalizacja ttf, otf, etc
za pomocą webfont generatora http://www.fontsquirrel.com/tools/webfont-generator można zakodować plik z rodziną fontu do formatu base64. Przykładowo font BreeSerif-Regular który ma 45.5kb można zmniejszyć do 25.5kb
Ikonki można również wygenerować jako webfont (np za pomocą http://fontello.com), co także ma wpływ na optymalizację strony.

3. Narzędzia/tooling

Use tools, not rules.

Sprites

Należy pamiętać o nie przekraczaniu maksymalnego rozmiaru sprajtów dla urządzeń mobilnych.
Dobrym narzędziem do tworzenia sprajtów dysponuje Compass. Jeżeli używamy w projekcie SASS / Compass jest to idealne rozwiązanie.

Tutorial do Compass Spriting:
http://compass-style.org/help/tutorials/spriting/

Southstreet

Southstreet to zbiór przydatnych, dostępnych skryptów od FilamentGroup. Zawiera:

- eCSSential (konkatenacja plików CSS pod kątem MQs)
- QuickConcat (skrypt w php do konkatenacji plików)
- AjaxInclude (includuje zewnętrzny HTML w front-end)
- Picturefill (responsywne obrazki)
- AppendAround (modyfikacja DOM na podstawie breakpointów)

https://github.com/filamentgroup/Southstreet/

Grunt: The JavaScript Task Runner

Strona:
http://gruntjs.com

Opis:
Głównym zadaniem Grunt'a jest polepszenie i zautomatyzowanie workflow frontendowego projektu. Jest też świetnym narzędziem to optymalizacji assetów.

Lista zadań i pluginów, które mają wpływ na optymalizacje strony RWD:

- concat
- min
- uglify
- grunt-responsive-images
- grunt-contrib-imageoptim
- grunt-uncss

Desktopowe aplikacje

Hammer for Mac

Strona:
http://hammerformac.com

Zalety:
- zoptymalizowany build
    - minifikowane i złączone pliki js/css
- includy w htmlu
- zmienne (np do title)
- automatyczna kompilacja
    - sass, cofeescript, HAML, Markdow
- reload strony
- przygotowywanie templatów

Wady:
- brak kompilacji lessa
- tylko na maca

Cena:
$23.99

Prepros

Strona:
http://alphapixels.com/prepros/

Zalety:

- Compile Everything
  - LESS, Sass, SCSS, Stylus, Jade, Slim, Coffeescript, LiveScript, Haml, Markdown.
- Js Concatenation
- Image Optimization
- Multi device Refresh
- Built-in http server

Wady:

- Brak możliwości wydzielania i includowania szablonow w html

Cena:
darmowy, z płatną opcją Pro $24

CodeKit

Strona:
http://incident57.com/codekit/

Zalety:

- Compile Everything
  - Process Less, Sass, Stylus, Jade, Haml, Slim, CoffeeScript
- Live Browser Reloads
- Combine & Minify
- Optimize Images
- JSHint & JSLint
- Team Collaboration

Wady:

- tylko na maca

Cena:
$28

4. Testowanie

Warto przygotowywać raporty wydajności dla klientów, dzięki czemu mogą zobaczyć postępy w optymalizacji strony. Oprócz tego, warto po opublikowaniu strony zająć się post-optymalizacją, oraz ściśle badać zmiany w wydajności strony celem znalezienia możliwości do zmian.

- PageSpeed Insights
- YSlow
- Chrome DevTools Audits

5. Zewnętrzne narzędzia

Mobify.js
Take control of your DOM.
Adapt your site for any device.
Mobify.js is an open source library for improving responsive sites by providing responsive images, JS/CSS optimization, Adaptive Templating and more.
Mobify.js also provides a Capturing API for manipulating the DOM before any resources have loaded, giving developers the ability to enable the listed features above without changing any backend markup.
http://www.mobify.com/mobifyjs/

Foresight.js

- Request hi-res images according to device pixel ratio
- Estimates network connection speed prior to requesting an image
- Allows existing CSS techniques to control an image's dimensions within the browser
- Implements image-set() CSS to control image resolution variants
- Does not make multiple requests for the same image
- Javascript library and framework independent (ie: jQuery not required)
- Image dimensions set by percents will scale to the parent element's available width
- Default images will load without javascript enabled
- Does not use device detection through user-agents
- Minifies to 7K

http://www.cdnconnect.com/docs/foresightjs

Temp

https://speakerdeck.com/addyosmani/automating-front-end-workflow
http://davidbcalhoun.com/2011/mobile-performance-manifesto