Semantyczny blog w HTML

Spis treści

> Ten tutorial przeznaczony jest dla tych, którzy już mają jakiekolwiek pojęcie o HTML. Dobrym miejscem na zdobycie podstaw wiedzy jest **kurs HTML (https://www.kurshtml.edu.pl/)**.

Witam wszystkich! Przeszukałem kilka for webmasterskich i nie znalazłem ani jednego tutorialu na temat aktualnego HTML-a. Trza to naprawić!

> Od razu uprzedzam, że słowa „semantyczny” i „HTML” z tematu są rozdzielne i część porad można zastosować w starym dobrym HTML 4. A na upartego słowa „blog” w ogóle można się pozbyć.
>
> Trzeba też zwrócić uwagę na to, że **kiedyś istniały dwa standardy HTML (https://medium.com/content-uneditable/the-great-world-of-open-web-standards-64c1fe53063)**. Ten tutorial opiera się na aktualnej wersji standardu, **HTML LS od WHATWG (https://html.spec.whatwg.org/multipage/)**. W niektórych miejscach wspominam jednak o **HTML 5.2 (https://www.w3.org/TR/html52/)**, ponieważ zawiera często dokładniej określoną semantykę elementów.

Punkt wyjścia

Dobra, do rzeczy. Wyobraźmy sobie, że mamy sobie przykładowy blog (https://tutorials.comandeer.pl/res/html5-blog/start.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
    <html lang="pl" dir="ltr">
        <head>
            <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
            <meta name="description" content="Super hiper wpis na temat czegokolwiek">
            <meta name="keywords" content="Bardzo ważne słowa kluczowe">
            <script type="text/javascript" src="http://bardzo-zajety-serwer.google.com/jquery.js"></script>
            <script type="text/javascript" src="http://mniej-zajety-serwer.blog.pl/scripty.js"></script>
            <link rel="alternate" type="application/rss+xml" title="Wpisy na RSS" href="http://example.net/feed">
            <link rel="index" title="Strona główna" href="http://example.net">
            <link rel="prev" title="Jakiś hiper poprzedni wpis" href="http://example.net/Jakis-hiper-poprzedni-wpis">
            <link rel="next" title="Jakiś super następny wpis" href="http://example.net/Jakis-super-nastepny-wpis">
            <link rel="canonical" href="http://example.net/Super-hiper-wazny-wpis">
            <link rel="pingback" href="http://example.net/xmlrpc.php">
            <link rel="stylesheet" type="text/css" href="http://example.net/css/style.css">
            <title>Super hiper ważny wpis | Example.net - fajowy blog, na którym bloguję</title>
        </head>
        <body>
            <div id="header">
                <h1>
                    <a href="http://example.net">Example.net - fajowy blog, na którym bloguję</a>
                </h1>
                <p>Motto</p>
                <ul id="menu">
                    <li>
```

```
                        <a href="whatever.html">Whatever</a>
                    </li>
                    <li>
                        <a href="whereever.html">Wherever</a>
                    </li>
                    <li>
                        <a href="whenever.html">Whenever</a>
                    </li>
                </ul>
                <form action="search.php" method="post">
                    <p><input type="text" name="q"><button type="su
bmit" name="submit" value="1">Szukaj</button></p>
                </form>
            </div>
            <div id="main">
                <div class="post" id="Super-hiper-wazny-wpis">
                    <h2>
                        <a href="http://example.net/Super-hiper-waz
ny-wpis" rel="bookmark" title="Permalink do Super hiper ważny wpis"
>Super hiper ważny wpis</a>
                    </h2>
                    <p class="post info">Opublikowano 07.01.2011 pr
zez <a href="http://example.net/author">Comandeer</a></p>
                    <img src="obrazek-do-artykulu.png" alt="Ważny o
brazek">
                    <p>Lorem ipsum dolor sit amet, consectetur adip
iscing elit. Proin eu justo enim, ac faucibus massa. Vestibulum in
 elit aliquam purus sollicitudin adipiscing ac et sapien. Curabitur
eleifend justo diam, et viverra nisi. Quisque a ipsum vehicula nunc
vestibulum posuere. Suspendisse potenti. Vestibulum ante ipsum prim
is in faucibus orci luctus et ultrices posuere cubilia Curae; Vesti
bulum porttitor, neque eu congue fermentum, velit ante pellentesque
arcu, a posuere risus tortor vel ante. Donec et neque at odio hendr
erit mattis sed eu tellus. Nullam accumsan leo ut felis suscipit ve
hicula posuere erat eleifend. Donec semper lorem eu nibh tincidunt
 varius.</p>
                    <p>Vivamus leo arcu, convallis id iaculis sit a
met, fringilla et nunc. Donec aliquam, justo at mollis porta, enim
 lorem tempor eros, id iaculis arcu elit ac sem. In et erat eu metu
s ornare vestibulum non at arcu. Integer in nisi massa. Etiam nulla
felis, rhoncus non pharetra sed, vehicula eu risus. Vestibulum soda
```

les nunc nisi, vitae gravida nisl. Quisque lacus ipsum, commodo ac hendrerit vitae, porta dapibus tortor. Mauris sed risus diam. Morbi lacus elit, euismod dapibus interdum id, fermentum et mi. Nam at neque orci. Sed ac hendrerit augue. Vestibulum pharetra mattis mattis. Sed porta turpis dolor, condimentum blandit libero. Aliquam non nisi mi. Nulla nec sem elit. Duis blandit viverra lacus, in convallis mauris dictum et. Ut eget risus enim. Suspendisse potenti. Aenean leo odio, luctus quis eleifend quis, consectetur sit amet ipsum. Curabitur commodo leo ac risus dignissim at porttitor turpis fringilla.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ullamcorper neque a magna sodales sodales. Null a id enim sem, sit amet ultrices dolor. Vestibulum sollicitudin dolor quis quam vehicula egestas. Ut urna erat, gravida a tristique non, congue nec elit. Aenean tincidunt purus sed nisl semper sagittis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Vivamus in orci sit amet ligula posuere mollis a in diam. Nam enim tortor, ultrices quis laoreet nec, egestas non leo. Nam non fringilla turpis. Quisque vitae pellentesque nibh. Etiam dictum tincidunt ultricies. Cras sit amet aliquam odio. Morbi quis urna a nulla egestas placerat sed in lacus. Morbi id urna lacinia nunc bibendum ultricies. In pretium leo non odio feugiat porttitor. Donec aliquet purus in purus pretium aliquam. Etiam aliquet nibh ut nibh semper egestas. Cras in purus quam, nec posuere diam. Aliquam erat volutpat.

Morbi porta blandit nisi at fermentum. Praesent vel sagittis urna. Pellentesque ipsum eros, vestibulum id gravida vitae, dictum vel sapien. Nam placerat lacus non lacus facilisis vitae hendrerit nibh adipiscing. Sed aliquet nisi ligula, eu viverra turpis. Curabitur tincidunt arcu in enim tempus sit amet auctor nunc luctus. Praesent dui turpis, lobortis in euismod quis, fringilla at felis. Nullam ligula purus, mattis nec tristique non, aliquam vitae sem. Quisque sit amet magna magna, eget aliquam orci. Aliquam placerat diam sit amet erat gravida fermentum. Maecenas malesuada nibh quis lorem pellentesque laoreet. Vestibulum pulvinar tincidunt fringilla. Nullam pellentesque felis ac nisi dictum egestas. Nunc metus mi, euismod non convallis nec, varius et felis. Pellentesque vel lacus turpis. In hac habitasse platea dictumst. Morbi sodales mauris a nisi feugiat pulvinar.

Vivamus vitae lectus ut dui luctus iaculis. Pellentesque quis lectus id ipsum venenatis bibendum. Pellentesque

```
non magna nec purus mollis suscipit. Nulla vestibulum interdum ant
e, vestibulum ullamcorper est vulputate vitae. Pellentesque habitan
t morbi tristique senectus et netus et malesuada fames ac turpis eg
estas. Quisque aliquet mollis rhoncus. Phasellus imperdiet posuere
 bibendum. Quisque consequat imperdiet nibh vitae pretium. Suspendi
sse quis eros libero, non luctus mauris. Nulla laoreet nibh at eros
scelerisque condimentum. Cras aliquam velit at orci luctus et tinci
dunt arcu scelerisque. Donec at quam magna. Integer leo leo, ultric
es facilisis bibendum vel, feugiat interdum neque. Fusce vitae nibh
sapien, sit amet consequat velit. Fusce adipiscing tortor id nibh f
ringilla sollicitudin. Donec sodales sapien in sapien rutrum rutru
m. Nam quis enim nec nibh varius mattis nec in ante. Morbi non est
 eu diam elementum interdum. Morbi at odio non libero tempus eleife
nd nec quis ligula. Cras pellentesque mauris sit amet felis vestibu
lum tincidunt.</p>
                </div>
                <div class="tags">
                    <h3>Tagi</h3>
                    <ol>
                        <li>
                            <a href="http://example.net/tag/Tag1">T
ag1</a>
                        </li>
                        <li>
                            <a href="http://example.net/tag/Tag2">T
ag2</a>
                        </li>
                    </ol>
                </div>
                <h3>Komentarze</h3>
                <ol class="comments">
                    <li class="alt" id="comment-25">
                        <small><a href="#comment-25" title="">30.1
2.08, 21:19</a> </small>
                        <cite><a href="http://example.net/"><img al
t="Avatar użytkownika Comandeer" src="http://avatary.gdziekolwiek.c
om/Comandeer" height="48" width="48">Comandeer</a></cite>
                        <div>
                            <p>Ale komentarz!</p>
                        </div>
                    </li>
```

```
                </ol>
                <h3>Dodaj komentarz</h3>
                <form action="http://example.net/comment.php" metho
d="post" id="add_comment">
                    <fieldset>
                        <legend>Napisz komentarz</legend>
                        <dl>
                            <dt>
                                <label for="comment_author">Nick</l
abel>
                            </dt>
                            <dd>
                                <input type="text" name="author" id
="comment_author" tabindex="1">
                            </dd>
                            <dt>
                                <label for="comment_email">E-mail</
label>
                            </dt>
                            <dd>
                                <input type="text" name="email" id=
"comment_email" tabindex="2">
                            </dd>
                            <dt>
                                <label for="comment_url">Strona</la
bel>
                            </dt>
                            <dd>
                                <input type="text" name="url" id="c
omment_url" tabindex="3">
                            </dd>
                            <dt>
                                <label for="comment_comment">Koment
arz</label>
                            </dt>
                            <dd>
                                <textarea name="comment" id="commen
t_comment" cols="48" rows="7" tabindex="5" ></textarea>
                            </dd>
                            <dd>
                                <button name="submit" type="submit"
```

```
id="comment_submit" tabindex="5">Wyślij</button>
                            </dd>
                        </dl>
                        <input type="hidden" name="comment_post_ID"
value="88">
                    </fieldset>
                </form>
            </div>
            <div id="aside">
                <div class="author">
                    <h3>O autorze blogaska</h3>
                    <p>Nazywam się Comandeer, mieszkam w uroczym mi
asteczku Świętochłowice i interesuję się webmasterką.</p>
                </div>
                <div class="tags cloud">
                    <h3>Tagi</h3>
                    <ul>
                        <li>
                            <a href="http://example.net/tag/Tag1">T
ag1</a>
                        </li>
                        <li>
                            <a href="http://example.net/tag/Tag2">T
ag2</a>
                        </li>
                        <li>
                            <a href="http://example.net/tag/Tag3">T
ag3</a>
                        </li>
                        <li>
                            <a href="http://example.net/tag/Tag2">T
ag4</a>
                        </li>
                    </ul>
                </div>
                <div class="archive">
                    <h3>Archiwum</h3>
                    <ol>
                        <li>
                            <a href="http://example.net/archiwum/20
11/01">Styczeń 2011</a>
```

```
                        </li>
                        <li>
                            <a href="http://example.net/archiwum/20
10/12">Grudzień 2010</a>
                        </li>
                    </ol>
                </div>
            </div>
            <div id="footer">Copyright &copy; 2011 by <a href="htt
p://example.net/author">Comandeer</a></div>
        </body>
    </html>
```

Nic nadzwyczajnego – blog jakich dużo w Internecie. Kodu nie będę wyjaśniał, bo mam wrażenie, że powinien być jasny. Ot, zwyczajny wpis na zwyczajnym blogu, poniżej komentarze, formularz do komentowania i jakieś tam dodatkowe info w bocznym panelu. Specjalnie nie dorzucam CSS, bo jestem z tych purystów, co to twierdzą, że strona to treść i kod, a wygląd jest jedynie dodatkiem (https://webroad.pl/inne/3722-progressive-enhancement-zapomniany-fundament).

Nasz cel

Chcemy być cool i porzucamy zgrzybiałego HTML 4 na rzecz najnowszego HTML-a! Poza tym spróbujemy nadać większe znaczenie naszej treści, przede wszystkim dla użytkownika, pamiętając, że co dobre dla użytkownika, jest także (najczęściej) dobre dla Google. Do tego celu wykorzystamy m.in (http://m.in). mikrodane i znaczniki sekcjonujące.

Nowy DOCTYPE

OK, to zaczynamy. Najpierw omówię przejście na aktualny HTML, bo tego jest najwięcej, a później doszlifujemy resztę. Pierwsze, co należy zmienić, to ten cały rozwlekły `DOCTYPE`. W aktualnym HTML mamy króciutki i ładniutki:

```
<!DOCTYPE html>
```

I już – po bólu. Twój blog właśnie stał się cool. Jeśli nie jesteś odważny, migrację możesz zakończyć w tym miejscu. Jeśli chcesz jednak poznać jaką zmianę w semantyce przynosi, pójdź za mną, a pokażę Ci piękno…

Ustawienie kodowania

Kolejnym wrogiem jest ustawienie kodowania. Po nowemu zapisać je można następująco:

```
<meta charset="utf-8">
```

To króciutkie meta obsługuje nawet IE6. Zdziwieni? A powinni, bo ta składnia powstała z… błędu. Na wielu stronach bowiem można było znaleźć potworka typu

```
<meta http-equiv="Content-type" content="text/html;charset="UTF-8""
>
```

Przeglądarki stwierdziły bowiem, że nie ma sensu parsować nieprawidłowego znacznika i po prostu znalazły inny sposób: szukały czegoś na wzór `charset=["']*[a-z0-9]["']*` (mam nadzieję, że to dobre wyrażenie regularne…). No i tak już zostało, a później to… ustandaryzowano w ramach HTML5 (https://www.w3.org/TR/html50/infrastructure.html#extracting-character-encodings-from-meta-elements). BTW ten znacznik i tak nie ma większego znaczenia, bo kodowanie jest narzucane przez nagłówki HTTP. Można je kontrolować ręcznie, np. przez PHP:

```
<?php header('Content-Type: text/html;charset=UTF-8'); ?>
```

Warto także zwrócić uwagę, że w specyfikacji HTML napisano, że nazwa kodowania jest „case insensitive”, co oznacza, że wielkość znaków nie ma znaczenia i `UTF-8` to to samo co `utf-8` czy nawet `UtF-8`.

Dołączanie arkusza stylów i skryptów

Teraz słówko o arkuszu stylów: jeśli dołączamy go poprzez `link`, można usunąć zbędny atrybut `[type]` – arkusze i tak zawsze są typu `text/css`.

W przypadku skryptów sprawa jest nieco bardziej skomplikowana. Jeśli mówimy o "normalnym" skrypcie (np. dołączenie jQuery, slider), wówczas `[type]` jest całkowicie zbędne – i tak jest to zawsze `[type=text/javascript]`. W specyfikacji są podane jednak także inne sposoby wykorzystania `script`, m.in (http://m.in). jako systemu szablonów (https://stackoverflow.com/questions/4912586/explanation-of-script-type-text-template-script). Ostatnio także dodano do specyfikacji `[type=module]` (https://html.spec.whatwg.org/multipage/webappapis.html#module-script), które pozwala wczytywać moduły ES (http://exploringjs.com/es6/ch_modules.html), a wkrótce może się także pojawić `[type=importmap]` (https://github.com/WICG/import-maps#installation).

Normalizacja stylów

Pomiędzy poszczególnymi przeglądarkami istnieją różnice w tym, jak domyślnie renderują poszczególne elementy HTML. Co prawda istnieją odgórne zalecenia (https://html.spec.whatwg.org/multipage/#toc-rendering), jednak różne przeglądarki wspierają standardy sieciowe w różnym stopniu, co samo w sobie wpływa także na domyślne style (bo np. przeglądarka nie potrafi obsłużyć właściwości CSS, które zostały użyte w zalecanym, domyślnym stylu). Dlatego też powstała potrzeba stworzenia narzędzi, które będą takie różnice niwelować i sprawiać, że strona będzie wyglądać tak samo w niemal wszystkich przeglądarkach. Jak to zwykle bywa w świecie IT, mamy kilka konkurujących ze sobą rozwiązań (https://xkcd.com/927/):

- `normalize.css` (https://csstools.github.io/normalize.css/) – to obecnie najpopularniejsze rozwiązanie problemu różnych stylów. Jego działanie polega na ustawieniu konkretnych stylów dla tych elementów, które są różnie generowane w przeglądarkach. Tym samym ich wygląd zostaje uspójniony.
- `reset.css` (http://html5doctor.com/html-5-reset-stylesheet/) – to chronologicznie pierwsza z technik, jakie służyły do niwelowania różnic w stylach pomiędzy przeglądarkami. Polegała na usunięciu praktycznie wszystkich domyślnych stylów oraz przywróceniu domyślnego, zgodnego ze specyfikacją sposobu wyświetlania (`display`) poszczególnych elementów.
- `sanitize.css` (https://csstools.github.io/sanitize.css/) – w pewnym sensie narzędzie to rozwija to, co oferuje `normalize.css`, dodając dodatkowe style, które autor tej biblioteki uznał za sensowne.
- CSS Remedy (https://github.com/mozdevs/cssremedy) – to projekt Mozilli, które zadaniem jest dostarczenie sensownych stylów domyślnych dla wszystkich elementów. Style te mają być punktem wyjścia dla wszelkich projektów webowych. To najmłodszy projekt ze wszystkich wymienionych.

Obsługa starszych przeglądarek

Jeśli w przyszłości wzbogacisz swój blog o jakieś nowinki z Web APIs (np. nagrywanie video komentarzy), to warto zapoznać się z pojęciem feature detection (ang. wykrywanie możliwości) i biblioteką Modernizr (https://modernizr.com/). Dodatkowo można też wykorzystać usługę Polyfill.io (https://polyfill.io/v3/), która sama decyduje, jakie polyfille są potrzebne i je dołącza. Warto także poczytać o technice „cut the mustard” (https://responsivenews.co.uk/post/18948466399/cutting-the-mustard).

Natomiast jeśli potrzebujesz wsparcia dla naprawdę antycznych Internet Explorerów (w wersji niższej niż 9), wystarczy dołączyć w `head` jeden skrypt:

```
<!--[if lt IE 9]>
    <script src="https://cdn.jsdelivr.net/npm/html5shiv@3.7.3/dist/
html5shiv-printshiv.min.js"></script>
<![endif]-->
```

Po więcej informacji o tym skrypciku zapraszam do oficjalnego repozytorium na GitHubie (https://github.com/aFarkas/html5shiv), na blog Remy'iego Sharpa (https://remysharp.com/2009/01/07/html5-enabling-script/) i blog Paula Irisha (https://www.paulirish.com/2011/the-history-of-the-html5-shiv/).

Tutaj warto też od razu zwrócić uwagę na fakt, że nieznane przeglądarce elementy są nieostylowane w żaden sensowny sposób i przez to możemy dostać np. liniowe `section`, co raczej jest niezbyt pożądane. Na szczęście problem ten rozwiązuje normalizacja stylów.

> Jeśli się zastanawiasz „a co z użytkownikami IE 9-, którym nie działa JS?”, odpowiem tak: jest ich na tyle mało, że nie sądzę, byś kiedykolwiek na nich trafił. Niemniej istnieją dwie alternatywy dla znaczników HTML5 (na chwilę obecną już bym ich nie polecał): **używanie `div[role]`** lub **(prawie) nieużywanie znaczników HTML5 (https://web.archive.org/web/20170606155631/http://2ndidea.com/en/html5-coding-without-html5shiv/)**. Warto się także zastanowić, czy nie porzucić całkowicie HTML5 Shiva, jeśli mamy pewność, że użytkownicy *naprawdę* starych IE nam się nie trafią.

> Warto także wspomnieć o **komentarzach warunkowych dla IE (https://www.quirksmode.org/css/condcom.html)**. Niemniej w IE 10 już nie działają, a prawdę mówiąc w dzisiejszym webdevie istnieją o wiele lepsze metody (jak wspomniane wyżej feature detection).

Nagłówek i nawigacja

Dobra, `head` na razie zostawmy i przejdźmy do `body`. Pierwsze, co rzuca nam się w oczy, to `div#header`. Nic nie znaczy, prawda? Nawet `[id=header]` tego nie ratuje, ponieważ identyfikatory nie mają żadnego znaczenia semantycznego (pogrubienie moje):

> Identifiers are **opaque strings**. Particular meanings should not be derived from the value of the `id` attribute.
>
> [Identyfikatory są **ciągami nieprzejrzystymi**. Konkretne znaczenia nie powinny być wywodzone z wartości atrybutu `[id]`.]
>
> *— https://html.spec.whatwg.org/multipage/dom.html#the-id-attribute*

HTML rozwiązuje ten problem, wprowadzając znacznik `header`. Jak sama nazwa wskazuje, służy on do oznaczania nagłówków i świetnie się nadaje do oznaczenia nagłówka strony. Zatem zamieńmy nasz `div#header` na `header#header`. Niby mała zmiana, a semantyka kodu rośnie. Ale, ale – to jeszcze nie wszystko! Spójrzmy na nagłówek: mamy tam `h1` z tytułem strony i `p` z mottem. Fajnie by było gdyby motto też było nagłówkiem, ale co z hierarchią nagłówków? Czy przypadkiem się nie zaburzy? Na chwilę obecną musisz wiedzieć, że podtytuły powinny być oznaczane przy pomocy `p`, tudzież `h* > span` (https://www.w3.org/TR/html52/common-idioms-without-dedicated-elements.html#subheadings-subtitles-alternative-titles-and-taglines). Do samej hierarchii nagłówków jeszcze wrócimy.

Jeśli już zmieniamy wszystko, to pewnie pomyślicie, że menu też można zmienić. Oczywiście macie rację! W tym celu posłużymy się znacznikiem `nav`, który powstał specjalnie po to, aby oznaczać elementy nawigacyjne.

> Co prawda w HTML istnieje także element `menu`, ale służy on do tworzenia menu dla aplikacji (menu kontekstowe czy toolbary) i nie należy go mylić z `nav`.

W naszym nagłówku pozostał jeszcze formularz wyszukiwania, ale do niego wrócę później i omówię go razem z formularzem dodawania komentarzy. W chwili obecnej nasz nagłówek powinien wyglądać mniej więcej tak:

```
<header id="header">
    <h1>
        <a href="http://example.net">Example.net - fajowy blog, na
 którym bloguję</a>
    </h1>
    <p>Motto</p>
    <nav id="menu">
        <ul>
            <li>
                <a href="whatever.html">Whatever</a>
            </li>
            <li>
                <a href="whereever.html">Wherever</a>
            </li>
            <li>
                <a href="whenever.html">Whenever</a>
            </li>
        </ul>
    </nav>
    <form action="search.php" method="post">
        <p><input type="text" name="q"><button type="submit" name="
submit" value="1">Szukaj</button></p>
    </form>
</header>
```

Artykuł

Główne ramy

Oki doki, teraz czas na danie główne: treść artykułu! Zapewne zauważyliście, że jest opatulona w dodatkowy `div#main`. Jak już wspomniałem, `div`y nie mają jakiegoś specjalnego znaczenia semantycznego, dlatego istnieje mnóstwo innych znaczników, które mogą `div`y zastąpić. Jednym z nich jest `section`, którym oznacza się pewne części dokumentu możliwe do wyróżnienia ze względu na ich przeznaczenie. W tym wypadku użycie `section` jest dozwolone, gdyż wyznacza obszar artykułu wraz ze wszystkimi "dodatkami" do niego (tj. tagi i komentarze) – całość tworzy pewną całość treściową.

Niemniej tutaj sytuacja jest bardziej specyficzna, bo mamy do czynienia z główną treścią strony. A specjalnie dla niej istnieje znacznik `main` (https://html.spec.whatwg.org/multipage/grouping-content.html#the-main-element). Warto dodać, że ten znacznik – ze względów dostępności –

powinien znaleźć się na stronie **wyłącznie raz** (https://github.com/whatwg/html/issues/100).

Dobra, teraz zatrzymujemy się na `div.post` – to też można ulepszyć. Istnieje znacznik `article`, który służy do oznaczania… artykułów (nie wpadliście na to, co?), a dokładniej „semantycznej całości, która może być prezentowana samodzielnie” (dość swobodna interpretacja specyfikacji). Wpis na blogu pasuje idealnie do tej definicji. No, na co czekasz – wyrzuć tego `div`a!

> W tym miejscu pozwolę sobie na małą dygresję. Przed chwilą oznaczyliśmy główną treść strony przy pomocy `main`. Teraz oznaczamy artykuł przy pomocy `article`. Jest to całkowicie poprawne, jednakże przy stronach, na których znajduje się tylko jeden artykuł (jak w naszym przypadku), można się zastanawiać, czy aż tak dokładny podział jest konieczny. Wówczas można pomyśleć nad rozwiązaniem typu `article[role=main]` (**o `[role]` napisałem ciut niżej**) lub po prostu pozostawieniu samego `main`.

Nagłówek

Znowu natykamy się na nagłówek, a tuż za nim na akapit z informacjami na temat daty i autora postu. Te rzeczy doskonale nadają się na nagłówek, prawda? No to otoczmy to `header`! Nagłówki nie są tylko i wyłącznie dla całych stron – mogą je mieć także sekcje i artykuły. To ma sens, ponieważ praktycznie każdy wpis na blogu ma swój tytuł, swojego autora itd., które wcale nie muszą być zgodne z autorem i tytułem całej strony. Ale to jeszcze nie wszystko: daty też mają swój własny znacznik! Nie, nie jest to `date` wbrew pozorom, lecz `time`. Wraz z nim pojawia się nowy atrybut: `[datetime]`, który przyjmuje datę w formacie ISO-8601 (https://www.w3.org/TR/NOTE-datetime). Ten znacznik doskonale pasuje do oznaczenia daty publikacji wpisu.

Obrazki

Dobra, później mamy obrazek. „No cóż, przynajmniej obrazka tykać nie będzie…” – pomyślą ci, których przytłacza ogrom nowych możliwości. Niestety, obrazek też można ulepszyć! W tym wypadku posłużymy się znacznikiem `figure`. Jego zadaniem jest oznaczanie danych multimedialnych, które są istotne dla danego artykułu (sekcji), lecz mogą także być prezentowane samodzielnie. Na chłopski rozum: masz ilustrację do swojego artykułu, użyj `figure`!

> Co rozumiem pod pojęciem "ilustracja"? Wystarczy wyobrazić sobie książkę, w której mamy obrazek (albo tabelkę, zdjęcie itp.), a pod nim podpis typu "Ryc. 1. Pusta szafa". Dzięki takiemu oznaczeniu można wyciągnąć tego typu obiekt poza pierwotny kontekst (np. do spisu ilustracji na osobnej podstronie), a wciąż będzie mieć te same znaczenie (pusta szafa z podpisem to wciąż pusta szafa). Natomiast wszelkie ozdobniki graficzne, które są w artykule, żeby było ładnie, to po prostu ozdobniki. Ilustracja musi nieść ze sobą jakąś

wartość semantyczną (łatwo to stwierdzić po tym, że da się wymyślić sensowny opis) i nie może zależeć od kontekstu swojego występowania (jeśli tekst traci sens po przesunięciu danego obiektu, to znak, że to nie jest kandydat na `figure`).

```
<figure>
    <img src="http://example.net/images/obrazek-do-artykulu.png" al
t="Ważny obrazek">
    <figcaption>Bardzo ważny obrazek[…],który jest bardzo ważny</fi
gcaption>
</figure>
```

W `figcaption` można umieścić dokładny opis danego obiektu. Napisałem "obiektu", bo równie dobrze w `figure` można umieścić filmik (`video`), dźwięk (`audio`) czy tabelę (`table`).

Mała dygresja: `[alt]` jest potrzebny dla obrazka z powodów dostępności. Jednakże czasami… warto pozostawić go pustym, z tych samych powodów. O co chodzi? Otóż nie ma sensu powielać opisu obrazka, jeśli jest on zawarty w otaczającej go treści (np. jeśli w `figcaption` jest dokładny opis tego, co na obrazku, należy zostawić `[alt]` pusty; często zdarza się jednak, że w takim wypadku `[alt]` opisuje co jest na obrazku, a `figcaption` dodaje do tego interpretację/wyjaśnienie; np `[alt]` – „czerwony ptak na gałęzi”, `figcaption` – „Karłowata odmiana papugi”). W takich wypadkach można użyć `[aria-labelledby]`. Specyfikacja Zawiera **kilka wskazówek (https://html.spec.whatwg.org/multipage/images.html#alt)** (zresztą **nie tylko ono (https://webaim.org/techniques/alttext/)**). Ciekawy jest także fakt, że **`[alt]` to nie jedyny sposób na zapewnienie dostępności obrazków (https://developer.paciellogroup.com/blog/2014/04/short-note-alt/)**.

Jeśli chodzi o sam znacznik `img`: jego również można ruszyć, a to za sprawą tagu `picture` (https://html.spec.whatwg.org/multipage/embedded-content.html#the-picture-element). Przykład:

```
<picture>
    <source media="(min-width: 45em)" srcset="large-1.jpg 1x, large
-2.jpg 2x">
    <source media="(min-width: 18em)" srcset="med-1.jpg 1x, med-2.j
pg 2x">
    <source srcset="small-1.jpg 1x, small-2.jpg 2x">
    <img src="small-1.jpg" alt="">
</picture>
```

Ci, którzy bawili się `video`/`audio`, na pewno znają znaczniki `source`. Wskazują one „źródło elementu multimedialnego” (w tym wypadku obrazka). Które `source` ma wybrać przeglądarka? O tym decydują media queries, zawarte w atrybucie `[media]`. Warto zauważyć także, że często podawane są dwa obrazki, dodatkowo oznaczone "1x" i "2x". Są to wersje przystosowane do gęstości pikseli ("1x" dla normalnych monitorów, "2x" dla Retiny). Pojawia się też `img`. Zgodnie z najnowszą wersją specyfikacji HTML, `img` jest jego główną składową – cała otoczka służy jedynie wyborowi odpowiedniego pliku graficznego, który jest następnie ładowany do rzeczonego `img`. Proste i skuteczne, a zarazem dostarcza doskonałego fallbacku dla starszych przeglądarek.

Jaki sens ma stosowanie `picture`? Wyobraźmy sobie, że mamy zdjęcie prezydenta wygłaszającego orędzie do narodu. Owszem, można zastosować tricki w CSS, żeby obrazek nam się ładnie zeskalował na komórce, ale… No właśnie – będziemy mieli małe, rozmazane wiadomo co. Natomiast przy pomocy `picture` komórkom i innym mniejszym ekranom możemy zaserwować odpowiednio mniejszy (zatem i lżejszy) obrazek ze zbliżeniem na twarz prezydenta. Nie dość, że iPhone będzie musiał mniej ściągnąć, to jeszcze dostanie zdjęcie przedstawiające to, co najważniejsze. Oczywiście działa to też w drugą stronę i super wypasionym ekranom panoramicznym o przekątnej 100 cali możemy wysłać zdjęcie prezydenta + całego słuchającego go tłumu.

> Warto zauważyć, że rozwojem technik związanych z responsywnością zajmuje się powołana **community group [grupa społeczna] (https://www.w3.org/community/respimg/)** (która jest teraz częścią **większej inicjatywy (https://www.w3.org/community/wicg/)**). To właśnie ona stoi za powstaniem tagu `picture` oraz jego mniej inwazyjnych odpowiedników: **atrybutów `[srcset]` i `[sizes]` dla tagu `img` (https://html.spec.whatwg.org/multipage/images.html#srcset-attributes)**.

Coraz częściej próbuje się także zrzucić odpowiedzialność za rozwiązanie problemu na serwer (https://www.igvita.com/2013/08/29/automating-dpr-switching-with-client-hints/) (co IMO jest bardzo rozsądnym rozwiązaniem). Jednak najlepszym (ale też najbardziej długotrwałym) rozwiązaniem wydaje się po prostu stworzenie nowego formatu obrazków (https://www.smashingmagazine.com/2013/09/responsive-image-container/). Zresztą nieważne tak naprawdę, jak problem responsywnych obrazków zostanie rozwiązany – najważniejsze, że(by?) został rozwiązany.

Outline

Dobra, akapitów się (jeszcze) zmienić nie da, więc treść mamy z głowy. Przyjrzyjmy się zatem hierarchii nagłówków w HTML. Jest to o tyle ważny temat, że hierarchia nagłówków odzwierciedla całą strukturę treści na stronie (stanowi jej spis treści) i dzięki temu ułatwia

nawigację dla osób korzystających z czytników ekranowych (https://web.archive.org/web/20170603013525/http://internet-bez-barier.com/elementy-naglowkowe-a-dostepnosc/#naglowki-w-nawigacji).

W HTML5 istniał ambitny plan odejścia od tradycyjnych znaczników `h1`–`h6` i przejścia na hierarchię treści opartą na nowych znacznikach sekcjonujących (https://html.spec.whatwg.org/multipage/sections.html#outlines) (`section, article` itd.). Niemniej nikt tego nie zaimplementował (https://www.paciellogroup.com/blog/2013/10/html5-document-outline/) i ostatecznie postanowiono pozbyć się tej fikcji (https://lists.w3.org/Archives/Public/public-html/2016Apr/0032.html). Stąd jedynym poprawnym sposobem tworzenia hierarchii treści na stronie HTML jest połączenie obydwu metod (tagi sekcjonujące + różnicowanie ważności nagłówków), gdyż nowy algorytm prawdopodobnie zostanie zastąpiony nowym rozwiązaniem (https://github.com/whatwg/html/pull/3499) i zastosowanie go w obecnej formie spowoduje, że AT (http://en.wikipedia.org/wiki/Assistive_Technology) dostaną spłaszczoną hierarchię nagłówków (co raczej jest średnim pomysłem).

Jeśli chcesz przetestować swój outline, wystarczy zaznaczyć odpowiednią opcję („Show outline”) w walidatorze (https://validator.w3.org/nu).

> Jako ciekawostkę można tutaj powiedzieć, że umiejętnie korzystając z atrybutu `[role=heading]` i `[aria-level]` można tworzyć hierarchię na samych `h1`, tak, jak zostało to opisane w specyfikacji. Metodę tę stosuje **niestandardowy znacznik `html5-h` (https://github.com/ThePacielloGroup/html5-h)** (przy pomocy **standardu Web Components (https://webroad.pl/javascript/3505-web-components)** i **frameworka Polymer (https://www.polymer-project.org/1.0/)**, z którym **miałem już do czynienia (https://tutorials.comandeer.pl/polymer.html)**). Niemniej warto zauważyć, że to bardziej hack niźli eleganckie rozwiązanie problemu. Jednakowoż `[aria-level]` można użyć w przypadku, gdy potrzebujemy więcej niż 6 poziomów zagłębień nagłówków (do "pogłębienia" `h6` – bardzo edge'owy przypadek, ale warto o tym wiedzieć).

Jak zatem stworzyć sensowną hierarchię nagłówków? Wystarczy trzymać się kilku prostych zasad:

- Nagłówki powinny być po kolei, zatem najpierw `h1`, potem `h2`, `h3` itd. Nie powinna wystąpić sytuacja, gdy hierarchia zaczyna się od np. `h4` albo gdy po `h2` wystąpi od razu `h5`.
- Nagłówek `h1` powinien wystąpić na stronie tylko raz i powinien stanowić główny nagłówek strony (https://blog.comandeer.pl/o-naglowkach-slow-kilka.html#g%C5%82%C3%B3wny-nag%C5%82%C3%B3wek).
- Każda sekcja lub artykuł powinny mieć swój nagłówek. Najlepiej podzielić stronę przy pomocy nagłówków, a dopiero potem dołożyć do tego znaczniki sekcyjne (tzw. Headings First Principle (https://blog.comandeer.pl/headings-first-principle.html)).

Przyjrzyjmy się nieco bliżej kwestii głównego nagłówka. W przypadku bloga tego typu nagłówek będzie się bowiem różnił pomiędzy stroną główną a stronami poszczególnych wpisów. Na stronie głównej bowiem głównym nagłówkiem będzie nazwa strony, zatem kod będzie wyglądał mniej więcej tak:

```
<header id="header">
    <h1>
        <a href="http://example.net">Example.net - fajowy blog, na
 którym bloguję</a>
    </h1>
    <p>Motto</p>
    […]
</header>
```

W przypadku jednak strony konkretnego wpisu głównym nagłówkiem powinien być tytuł wpisu, bo to on informuje, o czym jest dana strona. Dlatego też nagłówek w `#header` nie pełni już swojej funkcji, stając się wyłącznie linkiem nawigacyjnym:

```
<header id="header">
    <a href="http://example.net">Example.net - fajowy blog, na któr
ym bloguję</a>
    <p>Motto</p>
    […]
</header>
```

Natomiast `h1` znajdzie się wówczas w `main > article`:

```
<main>
    <article>
        <header>
            <h1>Super hiper ważny wpis</h1>
            […]
        </header>
        […]
    </article>
    […]
</main>
```

Co uważniejszy czytelnik zapewne zastanowi się teraz, czy aby na pewno `body > header > h1` to to samo co `body > article > h1`. Odpowiedź na to pytanie nie jest jednoznaczna. Na gruncie semantyki nie, ponieważ pierwszy nagłówek jest nagłówkiem strony, a drugi – nagłówkiem konkretnego artykułu. Niemniej na gruncie obecnej implementacji dostępności (czyli braku implementacji algorytmu outline'u) ranga nagłówków **nie zależy** od semantyki. Tym samym `h1` **zawsze** jest najważniejszym nagłówkiem na stronie, bez względu na miejsce wystąpienia. I dopóki algorytm outline'u nie zostanie zaimplementowany, tego typu rozwiązanie (`article > h1` jako główny nagłówek strony) wydaje się najlepsze – pomimo swojej niedoskonałej semantyki.

Można pójść o krok dalej. Heydon Pickering w swojej książce Inclusive Design Patterns (https://www.smashingmagazine.com/inclusive-design-patterns/) stwierdza, że w chwili, gdy logo strony nie jest nagłówkiem a jedynie linkiem, staje się integralną częścią nawigacji. Ma to sens, gdyż logo jest równocześnie odnośnikiem do strony głównej. Tym samym można się pokusić o całkowite przebudowanie nagłówka:

```
<header id="header">
    <nav>
        <ul>
            <li>
                <a href="https://example.net">
                    <p>Example.net - fajowy blog, na którym bloguję
</p>
                    <p>Motto</p>
                </a>
            </li>
            <li><a href="whatever.html">Whatever</a></li>
            <li><a href="wherever.html">Wherever</a></li>
            <li><a href="whenever.html">Whenever</a></li>
        </ul>
    </nav>
    […]
</header>
```

W tej wersji logo strony staje się pierwszym elementem menu.

Jak w praktyce może wyglądać zamiana nagłówków miejscami na stronie głównej i stronach wpisów, można zobaczyć na **WebKrytyku (https://www.webkrytyk.pl/)**. Z kolei wzorzec z `nav` można zobaczyć na **stronie domowej Heydona (https://www.heydonworks.com/)**.

Komentarze

OK, to jeszcze słówko na temat wyświetlania komentarzy: w HTML5 `cite` działał tak, jak nakazuje logika (http://html5doctor.com/cite-and-blockquote-reloaded/) i nie ograniczał sie już tylko i wyłącznie do oznaczania tytułów dzieł, ale także innego rodzaju źródeł, w tym osób. To sprawiało, że komentarze można traktować jako cytaty wypowiedzi konkretnych osób. Co prawda w obecnym standardzie HTML definicja `cite` jest ograniczona wyłącznie do tytułów dzieł, ale powiedzmy sobie szczerze – taka definicja jest mocno ograniczająca i nieprzystająca do rzeczywistości. Stąd osobiście stosuję `cite` tak, jak na to pozwalała specyfikacja HTML5.

Niemniej ważniejszą kwestią jest sposób organizacji komentarzy. Przykład w specyfikacji HTML 5.2 (https://www.w3.org/TR/html52/sections.html#example-86aedc40) jest bardzo sensownym punktem wyjścia. Pokazuje on, że komentarze znajdują się w podsekcji (`article > section`), a dodatkowo są umieszczone w liście uporządkowanej (`ol`):

```
<section id="komentarze" class="comments">
    <h2>Komentarze</h2>
    <ol>
        <li id="comment-25" itemprop="comment" itemscope itemtype="
http://schema.org/Comment">
            <div class="comment-meta">
                <a href="#comment-25">
                    <time datetime="2011-01-07T20:41:06+00:00" item
prop="dateCreated">07.01.2011, 20:41</time>
                </a>
                <b itemprop="author" itemscope itemtype="http://sch
ema.org/Person"><a href="http://example.net/" itemprop="name url"><
img alt="" src="http://avatary.gdziekolwiek.com/Comandeer" height="
48" width="48" itemprop="image">Comandeer</a> skomentował</b>
            </div>
            <p itemprop="text">Ale komentarz!</p>
        </li>
    </ol>
    […]
</section>
```

Wyjaśnienie znaczenia atrybutów `[itemprop], [itemscope], [itemtype]` znajduje się w **sekcji o mikrodanych**.

Aktualna specyfikacja sugeruje oznaczanie komentarzy przy pomocy `article` (https://html.spec.whatwg.org/multipage/sections.html#article-example). Dzięki temu powstaje swoista hierarchia artykułów (tzn. każdy komentarz jest "podartykułem" wpisu). Jest to logiczne, ponieważ każdy komentarz może być wyświetlany "samodzielnie" (wszak opinia o dziele A może istnieć odrębnie od tego dzieła). Przywodzi to na myśl stare, dobre gazety, w których odpowiedzią na konkretny artykuł jest… inny artykuł.

Jeśli jednak już zgodzimy się na oznaczanie komentarzy przy pomocy artykułu, czekają nas kolejne wyzwania. Otóż regułą kciuka jest, by każdy artykuł miał odpowiedni nagłówek. Lecz jaki nagłówek powinien mieć komentarz? Na ForumWeb.pl (http://ForumWeb.pl) wywiązała się gorąca dyskusja na ten temat (https://www.forumweb.pl/komentarze-kurshtml-edu-pl/html-html5/496603#496603), która skłoniła mnie do refleksji na temat kodu komentarzy. Na większości stron w Sieci, a także w specyfikacji, komentarze są traktowane jako wyjątek, który nie posiada nagłówka (bo są to artykuły zagnieżdżone). Niemniej na chwilę obecną uważam, że dodanie jako nagłówka nazwy użytkownika nie zaszkodzi – pozwoli to nam w łatwiejszy sposób śledzić komentarze danego użytkownika (np. naszego ulubionego hejtera!). Pomysł, by w nagłówku czy etykietce `[aria-label]` dodawać pierwsze słowa komentarza ostatecznie odrzuciłem – z tego samego powodu, z jakiego niekiedy warto zostawić puste `[alt]`: nie ma sensu dublować treści.

> Nie oznacza to jednak, że np. w RSS nie warto zamieścić tego typu opisu. W przypadku jednak strony internetowej jest to zbędne.

Ostatecznie kod komentarzy w oparciu o `article` wygląda tak:

```
<article id="comment-25">
    <header class="comment-meta">
        <a href="#comment-25">
            <time datetime="2011-01-07T20:41:06+00:00">07.01.2011,
 20:41</time>
        </a>
        <h3>
            <b><a href="http://example.net/"><img alt="" src="htt
p://avatary.gdziekolwiek.com/Comandeer" height="48" width="48">Coma
ndeer</a> skomentował</b>
        </h3>
    </header>
    <p>Ale komentarz!</p>
</article>
```

Warto sobie zatem zadać pytanie: który sposób jest lepszy? Osobiście optuję bardziej za wersją z `ol`, gdyż niepotrzebnie nie komplikuje prostej sprawy. Lista uporządkowana dodatkowo jasno pokazuje, że komentarze są spójną całością i tworzą dyskusję nierozerwalnie związaną z konkretnym artykułem. Wersja z `article` z kolei wskazuje na fakt, że każdy komentarz jest tak naprawdę niezależnym utworem i można go prezentować oddzielnie. Wydaje się zatem, że wybór zależy od sposobu patrzenia na rolę komentarzy.

Nagłówki jako punkty nawigacyjne

Jedną z praktyk poprawiających użyteczność strony internetowej jest bez wątpienia możliwość łatwego nawigowania po jej zawartości. W przypadku tradycyjnych stron internetowych można to osiągnąć dzięki kotwicom (http://www.kurshtml.edu.pl/html/do_etykiety,odsylacze.html). Najlepiej je przypiąć dla każdej sekcji lub nagłówka (jeśli nie mamy sekcji). Ma to bardzo proste uzasadnienie: sekcje/nagłówki dzielą stronę na sensowne części. Gdyby porównać stronę do książki, poszczególne nagłówki oznaczałyby rozdziały. Jeśli chcemy komuś powiedzieć, żeby przeczytał sobie o skorupiakach, podajemy mu nazwę konkretnego rozdziału (czy wręcz numer strony, na której się zaczyna!) a nie tytuł grubej encyklopedii, prawda? Tak samo jest w przypadku stron internetowych – zamiast podać odnośnik do całej strony, często o wiele lepiej jest podać odnośnik do konkretnego fragmentu. Stąd zachęcam do dodawania `[id]` do każdej sekcji/nagłówka.

Choć cały kod strony najlepiej pisać w języku angielskim, `[id]` służące do nawigacji są wyjątkiem i **powinny** być w języku, w jakim jest napisana cała strona internetowa. Wystarczy sobie pomyśleć, który adres sprawia więcej sensu: `strona.pl/artykul#section` czy jednak `strona.pl/artykul#sekcja`.

Zawartość nawigacyjnego `[id]` najlepiej stworzyć przez stworzenie **sluga (https://www.wpbeginner.com/glossary/post-slug/)** nagłówka, do którego chcemy odesłać czytelnika, np. "Moje piórka" dadzą nam `[id=moje-piorka]`.

Całość złożona do kupy

A oto szkielet naszej treści (spokojnie, na końcu podam całość):

```
<main>
    <article id="tytul">
        <header>
            <h1>
                <a>Tytuł</a>
            </h1>
            <p>Autor <time datetime="2011-01-07T20:40:06+00:00">07.
01.2011</time></p>
        </header>
        <div>
        treść
        </div>
        <footer>
            <section>Tagi</section>
        </footer>
        <section>Komentarze
            <ol></ol>
            <form>Dodanie komentarza</form>
        </section>
    </article>
</main>
```

Warto zauważyć, że sama treść artykułu znajduje się po prostu w `div`. Wiadomo, że jest to treść artykułu (jest w `article` i nie jest ani w nagłówku, ani w stopce), dlatego co najwyżej można umieścić ją w `div` w celu ułatwienia stylizacji. `section` również tutaj nie pasuje, bo nie da się utworzyć żadnego sensownego nagłówka ("Treść artykułu" brzmi jak wymyślony na poczekaniu). Niemniej `section` w artykule jak najbardziej pojawić się może, ale w przypadku, gdy artykuł zawiera w sobie podtytuły (jak np. ten tutorial – nie, nie wzorujcie się na jego kodzie). Wówczas każdy podtytuł z odpowiadającą mu treścią może być traktowany jako osobna `section`. Z kolei tagi można potraktować jako `footer` (dodatkowe, poboczne informacje do artykułu) a komentarze – sekcję (mają sensowny tytuł i stanowią spójną całość). Oczywiście można też zrobić inaczej: zarówno tagi, jak i komentarze potraktować jako sekcje i włożyć je w `footer`. Wypada jednak pamiętać, że definicja stopki sugeruje, że jest ona przeznaczona na zawarcie metadanych:

> A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.
>
> [Stopka zwyczajowo zawiera informacje o sekcji ją zawierającej, takie jak nazwa autora, linki do powiązanych dokumentów, prawa autorskie itp.]
>
> — *https://html.spec.whatwg.org/multipage/sections.html#the-footer-element*

Tagi w tym kontekście pasują o wiele bardziej (po prostu opisują dany artykuł jako słowa kluczowe), podczas gdy komentarze już niespecjalnie. Owszem, można je potraktować jako "powiązane dokumenty", jednak są one dość autonomicznymi tworami i część z nich z powodzeniem mogłaby żyć bez łączności ze swoim "rodzicem". Stąd wydzielenie komentarzy do sekcji i pozostawienie w stopce samych tagów wydaje się bardziej sensownym rozwiązaniem.

Tak to mniej więcej wygląda. Formularz se opuszczę i powrócę do niego później, bo formularze w HTML5 to bardzo ciekawy temat warty więcej niż szybkiego spojrzenia.

Sidebar

Przejdźmy do `div#aside`. Tworzy on tzw. "boczny pasek" z informacjami luźno związanymi z główną treścią (i tutaj mała dygresja: gdyby chodziło o luźno związane informacje z samym artykułem, mielibyśmy do czynienia z `article > aside`; w tym wypadku chodzi o całą zawartość `main`; inna sprawa, że tutaj faktycznie jest to jeden artykuł): autor, tagi, archiwa… Do oznaczania takich rzeczy powstał `aside`. Oznacza się nim właśnie luźno związane z główną treścią rzeczy, które można zaprezentować samodzielnie (to chyba ulubione słówko w WHATWG).

Idąc dalej, `div`y z tagami, autorem i archiwami można potraktować jako artykuły. I choć początkowo taka interpretacja specyfikacji wydaje się ciutkę naciągana, to po dogłębniejszym wczytaniu okazuje się (chyba raczej na pewno) właściwa.

> The `<article>` element represents **a complete, or self-contained, composition in a document**, page, application, or site. This could be a magazine, newspaper, technical or scholarly article, an essay or report, a blog or other social media post.
>
> [Element `<article>` reprezentuje **kompletną, niezależną i samodzielną kompozycję w dokumencie**, stronie, aplikacji lub witrynie. Może to być magazyn, gazeta, techniczny lub szkolny artykuł, esej lub raport, blog lub inny wpis z sieci społecznych.]
>
> *— https://html.spec.whatwg.org/dev/sections.html#the-article-element*

Najważniejszy jest pogrubiony fragment. Ba, tylko co on znaczy? Idąc za radą doktorów (http://html5doctor.com/the-article-element/) trza pomyśleć, czy dany fragment wyciągnięty z kontekstu wciąż ma jakiś sens. Notka o autorze pozbawiona sąsiedztwa wpisu raczej nie stanie się bełkotem. Lista archiwalnych wpisów również pozostanie listą archiwalnych wpisów, tak jak lista tagów – listą tagów. Dlatego, moim skromnym zdaniem, chociaż na pierwszy rzut oka wydaje się to pewnym nadużyciem, `article` pasują tutaj idealnie. Rozważania te wspiera wyżej cytowany opis `article`, jeszcze bardziej zwracający uwagę na kwestię niezależności danego fragmentu treści.

Szkielet:

```
<aside>
    <article>
        <h3>O autorze</h3>
    </article>
    <article>
        <h3>Tagi</h3>
    </article>
    <article>
        <h3>Archiwum</h3>
    </article>
</aside>
```

Kolega (https://kacperkozak.pl/) zwrócił mi uwagę, że tagi i archiwum można potraktować też w inny sposób. Nie da się ukryć, że są to *de facto* dwie listy odnośników. Dlatego też można je potraktować jako dodatkowe elementy nawigacyjne (otoczyć `nav`). Moim zdaniem obydwie interpretacje tych elementów są poprawne, a różnica wynika po prostu z innej perspektywy spojrzenia na sprawę. Moja interpretacja wynika z analizy samej natury treści (czy jest ona

możliwa do zaprezentowania niezależnie, czy stanowi poboczną informację do głównej treści itd.), natomiast interpretacja kolegi – z analizy wzajemnych relacji i funkcji (czy ta lista odnośników może być potraktowana jako nawigacja, czy prowadzi do innej, powiązanej treści w obrębie witryny itd.). Gdyby ktoś był ciekawy, w specyfikacji pojawia się zastosowanie `aside > nav` właśnie w takim kontekście.

> `nav` nie służy tylko do oznaczania głównego menu strony. Jest znacznikiem do oznaczania elementów nawigacyjnych w ogóle. Jednak nie każda lista odnośników jest elementem nawigacyjnym. Zbiór odnośników do różnych artykułów w Internecie jest po prostu zbiorem odnośników (lub artykułem, w zależności od kontekstu). Natomiast już breadcrumbs czy też kotwice w obrębie jednej strony można potraktować jako element nawigacyjny.

> Jeśli używamy na swojej stronie więcej niż jednego `nav`, powinniśmy zadbać o to, aby każde z nich miało **inną nazwę** – najlepiej reprezentowaną przez nagłówek, połączony z elementem `nav` przy pomocy atrybutu `[aria-labelledby]`, w ostateczności atrybut `[aria-label]`. Pozwoli to użytkownikom, zwłaszcza korzystającym z czytników ekranowych, na zrozumienie, co dany element nawigacyjny im umożliwia (np. czy jest to menu strony, czy spis treści artykułu itd.). Niepisaną zasadą jest także ograniczenie liczby elementów `nav` do 3–4.

Stopka

No, to lwia część za nami – została tylko stopka. Tak, ją też zmieniamy. Tym razem div zamienia się w `footer`. Uff, blog jest już w całości przerobiony na aktualny HTML, ale to nie koniec – to dopiero początek!

> P.S. dla ekstrawaganckich: `html`, `head` i `body` są opcjonalne. To bardzo ciekawy sposób na **dodatkową mikrooptymalizację (https://meiert.com/en/blog/html-performance/)**.

Formularze

Wyszukiwarka

W HTML istnieje pełno typów pól i atrybutów formularzy. My zastosujemy raptem ułamek z nich.

Zacznijmy od formularza wyszukiwania. Dla pól wyszukiwania jest przewidziany typ – `[type=search]`. Choć dzięki temu nasz kod staje się bardziej zrozumiały, to jednak z punktu widzenia użytkownika nie zmienia się absolutnie nic – to pole wciąż zachowuje się jak stare dobre pole tekstowe (`[type=text]`).

Typy pól

Oczywiście `search` to nie jedyny typ pola. Z innych warto wspomnieć:

- `email` – dla e-maili
- `url` – dla URL-i
- `number` – dla liczb
- `date` – dla dat
- `color` – dla kolorów
- `range` – taki fajny slider

Inne bajery i dostępność

Ale, ale. Można jeszcze bardziej podbajerować dzięki atrybutowi `[placeholder]`. Robi on to, co tysiące tych badziewnych JS: wyświetla pomocniczy tekst, który znika po kliknięciu w pole (i nie kasuje przy okazji już wprowadzonego zapytania…).

> **Nie powinien być jednak traktowany jako zastępstwo dla `label`. Wspomina o tym nawet specyfikacja (https://html.spec.whatwg.org/multipage/input.html#the-placeholder-attribute)**.

A jeśli już chcesz się nazywać mistrzem formularzy, to dodaj sobie jeszcze atrybut `[required]`, który oznacza, że formularz nie może zostać wysłany, jeśli dane pole jest puste. Po przeróbce formularz szukania wygląda następująco:

```
<form action="search.php" method="post">
    <p>
        <label for="search-input">Szukaj:</label>
        <input type="search" id="search-input" name="q" placeholder
="Wpisz szukaną frazę…" required>
        <button type="submit" name="submit" value="1">Szukaj</butto
n>
    </p>
</form>
```

`label` do pola wyszukiwania można wstawić techniką ukrywania przed graficznymi przeglądarkami (https://adaptivethemes.com/using-css-clip-as-an-accessible-method-of-hiding-content/), używaną m.in (http://m.in). w HTML5 Boilerplate (https://html5boilerplate.com/). Userzy z wizualnymi przeglądarkami nie zobaczą wówczas etykiety (a pole wyszukiwania jest na tyle

charakterystyczne, że na pewno je rozpoznają), natomiast ci, którzy korzystają z czytników ekranowych, nie poczują się zagubieni, gdy natrafią na pole bez etykiety. **Nie można** jednak całkowicie usunąć etykiety (https://www.filamentgroup.com/lab/a11y-form-labels.html)!

PAMIĘTAJ! CSS (jak i obrazki, JS, inne dziwne rzeczy typu Flash, Java, Silverlight) są tylko dodatkami, które nie muszą być dostępne!

Oznacza to, że możesz być pewny tylko tego, że końcowy user zobaczy treść i HTML (patrz: użytkownicy **Lynxa (https://lynx.browser.org/)**). Stąd ważne jest zrozumienie podstaw tworzenia czystego, semantycznego kodu, który – nawet bez CSS – tworzy sensowną i czytelną całość. Dlatego też osobiście wolę tworzyć najpierw HTML dla strony, a dopiero później dorabiać CSS i JS. HTML to kartka papieru, na którą nakładam kolejne warstwy folii (CSS, JS). Podejście to znane jest jako **Progressive Enhancement (https://webroad.pl/inne/3722-progressive-enhancement-zapomniany-fundament)**.

Kompatybilność

Teraz zapewne się zapytasz: „a gdzie te wszystkie typy pól formularzy są obsługiwane?”. Zamiast podawać tak szybko deaktualizujące się informację, odsyłam do odpowiedniego źródła (https://caniuse.com/#feat=forms). Brak wsparcia nie oznacza jednak, że trza rezygnować z nowoczesnych formularzy. HTML jest pisany z myślą o kompatybilności wstecznej, co znaczy, że jeśli dany typ pola nie jest obsługiwany, pojawi się stare, dobre pole tekstowe. Resztę zachowań – włącznie z walidacją wprowadzonych danych – we wszystkich przeglądarkach można osiągnąć dzięki JS (https://www.thecssninja.com/javascript/h5f).

Przeróbmy zatem formularz dodawania komentarzy:

```
<form action="https://example.net/comment.php" method="post" id="do
daj-komentarz">
    <h3>Dodaj komentarz</h3>
    <p>
        <label for="comment-author">Nick</label>
        <input type="text" name="author" id="comment-author" requir
ed>
    </p>
    <p>
        <label for="comment-email">E-mail</label>
        <input type="email" name="email" id="comment-email" require
d>
    </p>
    <p>
        <label for="comment-url">Strona</label>
        <input type="url" name="url" id="comment-url">
    </p>
    <p>
        <label for="comment-comment">Komentarz</label>
        <textarea name="comment" id="comment-comment" cols="48" row
s="7" required></textarea>
    </p>
    <p>
        <button name="submit" type="submit" id="comment-submit">Wyś
lij</button>
    </p>
    <input type="hidden" name="comment_post_ID" value="88">
</form>
```

Tutaj warto wspomnieć o jeszcze jednym atrybucie pól formularza: `[pattern]`, który przyjmuje jako wartość wyrażenie regularne (https://www.regular-expressions.info/), dzięki czemu – jeśli żadne z wbudowanych typów pól formularza nie spełnia naszych wymagań – można wymusić kokretny format danych, np. kodu pocztowego:

```
<input type="text" name="zipcode" pattern="\d{2}\-\d{3}">
```

Naprawdę zachęcam do używania wszystkich dostępnych typów pól formularzy. Dzięki nim można uprościć walidację danych po stronie serwera (a przynajmniej nie martwić się, że w e-mailu dostajemy JS-a).

Uprościć **nie znaczy** zaniechać. Niewalidowanie danych po stronie serwera jest jak wpuszczanie nieznajomego do domu, dając mu do tego klucz do sejfu w sypialni. Sama specyfikacja wspomina, że **walidacja po stronie klienta ma jedynie podnosić UX (https://html.spec.whatwg.org/multipage/form-control-infrastructure.html#security-forms)**.

Przy okazji: **natywna walidacja w HTML jest ledwo działająca (https://www.quirksmode.org/blog/archives/2017/12/native_form_val.html)** i dlatego powinna być wykorzystywana głównie jako najbardziej podstawowa warstwa walidacji po stronie klienta, na której **nadbudujemy własną walidację w JS (https://hiddedevries.nl/en/blog/2017-04-04-how-to-make-inline-error-messages-accessible)**.

`[rel]` czyli dokąd to prowadzi

Atrybut `[rel]` dla linków oznacza ich przeznaczenie, a dokładniej do jakiego zasobu się odnoszą. Warto stosować atrybut `[rel]` na wszystkich linkach, które mają jakiekolwiek znaczenie semantyczne, np. jeśli link prowadzi do strony o autorze, niech będzie miał odpowiedni `[rel]`. Jakie `[rel]` można zatem stosować?

Istnieją też dwie wartości atrybutu `[rel]`, które znacząco podwyższają bezpieczeństwo strony: `[rel=noreferrer]` oraz `[rel=noopener]`. Najlepiej obydwa te atrybuty dodawać do linków prowadzących do zewnętrznych stron, dzięki czemu **nie będziemy narażać naszych użytkowników na ataki phishingowe oraz śledzenie (https://www.jitbit.com/alexblog/256-targetblank---the-most-underestimated-vulnerability-ever/)**.

Lista dozwolonych `[rel]`

- `index` – strona główna;
- `author` – coś o autorze;
- `prev` – poprzedni wpis;
- `next` – następny wpis;
- `canonical` – oryginalny URI do zasobu (czyli ten, który chcemy zaindeksować w Google);
- `nofollow` – BlackSEO;
- `shortlink` – oficjalny, skrócony adres strony;
- `tag` – do oznaczania tagów.

Pamiętaj, że wszystkie `[rel]` dotyczą aktualnej strony. Tym samym np. niepoprawne jest oznaczenie tagów w chmurce tagów przy pomocy `[rel=tag]` (gdyż nie odnoszą się one do tej konkretnej strony). Natomiast oznaczenie tak tagów, które występują wewnątrz głównego artykułu (w nagłówku czy stopce), jest już poprawne (bo odnoszą się do treści danej strony).

Rozszerzalność

Poza tym można tworzyć własne `[rel]`, chociaż większego sensu w tym nie widzę, bo wszystkie potrzebne już istnieją i są na tyle elastyczne, że można je nagiąć do własnych potrzeb. Na szczęście radosne tworzenie jest monitorowane i żeby jakiś `[rel]` został uznany za poprawny w HTML, należy go zgłosić (http://microformats.org/wiki/existing-rel-values).

A teraz, drogi blogerze, linki czekają na semantyczne wsparcie! Naprzód!

WAI-ARIA

Role

W HTML istnieje atrybut `[role]`, który określa rolę danego elementu na stronie. W przypadku bloga można je rozdzielić przykładowo tak:

- `header` (strony) → `banner` – ogólne informacje o stronie;
- `main` → `main` – główna część strony;
- `article` → `article` – artykuł, względnie `document` – dokument;
- `article > header, article > footer` → `group` (z tym, że nie wnosi to nic ciekawego tak naprawdę);
- `aside` → `complementary` – coś uzupełniającego;
- wyszukiwarka → `search` – nie trza tłumaczyć;
- `footer` strony → `contentinfo` – metadane;
- `nav` → `navigation` – element nawigacyjny.

Warto także zauważyć, że dla większości elementów przeglądarki same nadają odpowiednie `[role]` i dublowanie ich w kodzie jest niezalecane (https://w3c.github.io/using-aria/#rule1). ARIA przydaje się wówczas, gdy tworzymy pewne struktury od podstaw np. w przypadku wykorzystywania Web Components czy podmienianiu natywnych pól formularza własnymi odpowiednikami.

Ale po co to?

Dobra, `[role]` jest jasne, ale co to WAI-ARIA? Otóż WAI to inicjatywa wewnątrz W3C poświęcona dostępności stron internetowych, a ARIA to jej najnowszy standard. ARIA to skrót od Accessible Rich Internet Applications i, jak nazwa wskazuje, powstał po to, by podnieść dostępność takich zaawansowanych aplikacji, jak Gmail czy Google Docs. Nic jednak nie szkodzi, aby nasz blog potraktować jak webappa.

Dla tworzących bardziej rozbudowane strony, z wykorzystaniem JS, głównie XHR, mogę polecić jeszcze takie własności jak `[aria-live]` i `[aria-atomic]`. Przy ciężkim Ajaksie (czyt. wszystko na **History API (https://developer.mozilla.org/en-US/docs/Web/API/History_API/Working_with_the_History_API)**) wręcz ratują życie.

Przy walidacji formularzy również warto użyć ARIA, zwłaszcza oznaczając niepoprawnie wypełnione pola przy pomocy `[aria-invalid]`.

Po więcej info o WAI-ARIA zapraszam do specyfikacji *ARIA in HTML* (https://w3c.github.io/html-aria/). Natomiast o dostępności najlepiej pisze Heydon Pickering (https://www.smashingmagazine.com/inclusive-design-patterns/).

Dane semantyczne

Mikroformaty

Mikroformaty powstały po to, by w dobie HTML 4 niesemantycznym divom nadać semantyczne znaczenie. Mimo że aktualnie mamy pełno semantycznych znaczników, to jednak mikroformaty wciąż są niezwykle przydatne. Pozwalają nam dokładnie opisać wydarzenia, osoby czy miejsca, używając do tego HTML-owego zapisu ogólnie znanych standardów, takich jak np. vcard (format wizytówek, obsługiwany przez większość telefonów komórkowych). Zastosowanie mikroformatów wcale nie jest trudne, wystarczy dodać do elementu odpowiednią klasę. Jako przykład zastosuję h-card (odpowiednik vcard) (http://microformats.org/wiki/h-card) i "oznaczę" boczne info o autorze:

```
<article class="h-card">
    <h3>O autorze blogaska</h3>
    <p>Nazywam się <a href="https://example.net/author" class="p-na
me u-url">Comandeer</a>, mieszkam w uroczym miasteczku <span class=
"p-locality">Świętochłowice</span> i interesuję się webmasterką.</p
>
</article>
```

Prawda, że łatwe? W podobny sposób można oznaczać wydarzenia (h-event (http://microformats.org/wiki/h-event)), przepisy (h-recipe (http://microformats.org/wiki/h-recipe)), a także upodobnić HTML do ATOM (h-feed (http://microformats.org/wiki/h-feed) + h-entry (http://microformats.org/wiki/h-entry)).

Microdata

Microdata (mikrodane) to z kolei rozszerzenie składni HTML, które działa podobnie do mikroformatów. Jedyna różnica polega na tym, że korzysta z nowych atrybutów – `[itemscope]`, `[itemtype]` i `[itemprop]` oraz z przestrzenii nazw. Najprościej pokazać to na przykładzie:

```
<article itemscope itemtype="http://schema.org/Person">
    <h3>O autorze blogaska</h3>
    <p>Nazywam się <a href="http://example.net/author" itemprop="na
me url">Comandeer</a>, mieszkam w uroczym miasteczku <span itemprop
="address" itemscope itemtype="http://schema.org/PostalAddress"><sp
an itemprop="addressLocality">Świętochłowice</span></span> i intere
suję się webmasterką.</p>
</article>
```

Wygląda i działa bardzo podobnie do mikroformatów (chociaż jest bardziej skomplikowane i – teoretycznie – pozwala odwzorowywać bardziej skomplikowane struktury danych). Co ważne, oba sposoby są znane Google'owi, który korzysta z nich do wyciągania danych ze stron. Korzystanie więc podnosi dostępność treści z poziomu wyszukiwarki.

> Google, Yahoo i Bing stworzyły standard oznaczania treści o nazwie **Schema.org (https://schema.org/)**, który ujednolica mikrodane na stronach WWW. Dzięki temu przedsięwzięciu można oznaczyć prawie każdy typ danych. Wydaje mi się, że takie podejście ma szansę wyprzeć mikroformaty, a przynajmniej **zmusić je do ewolucji (http://microformats.org/wiki/microformats-2)**.

Inne formaty danych semantycznych

Tutaj warto wspomnieć o głównym konkurencie microdata – RDFa (http://rdfa.info/) – który wygląda dość podobnie, ale jest bardziej XML-friendly i… ładniejszy. Osobiście preferuję go zamiast mikrodanych.

```
<article vocab="http://schema.org" typeof="Person">
    <h3>O autorze blogaska</h3>
    <p>Nazywam się <a href="http://example.net/author" property="na
me url">Comandeer</a>, mieszkam w uroczym miasteczku <span property
="address" typeof="PostalAddress"><span property="addressLocality">
Świętochłowice</span></span> i interesuję się webmasterką.</p>
</article>
```

Chociaż obecnie zarówno on, jak i microdata/mikroformaty, są zagrożone przez najnowszy format danych semantycznych, JSON LD (http://www.w3.org/TR/json-ld/), który jest obecnie preferowany przez Google (https://developers.google.com/search/docs/guides/intro-structured-data#structured-data-format). Osobiście uważam jednak, że JSON LD nadaje się najbardziej jako format danych zwracanych przez APIs, nie jako format opisywania stron WWW. W tym względzie RDFa wydaje się bardziej elastyczne.

Dublin Core

Dublin Core jest międzynarodowym standardem metadanych (ISO 15836-2003) i służy do opisywania dokumentów. Jest rozwijany i upowszechniany przez organizację Dublin Core Metadata Initiative, mającą na celu stworzenie semantycznej Sieci, w której z łatwością można będzie znaleźć dokumenty (pieśń przyszłości). Standard ustala 15 elementów, dzięki którym można dokładniej opisać stronę, lecz nie tylko – praktycznie każdy dokument cyfrowy. Może dzięki temu stanowić zamiennik dla "standardowych" `meta`. Najczęściej służy jako zamiennik dwóch metatagów: `keywords` i `description`. Ich "dublinowe" odpowiedniki to: `dcterms.subject` i `dcterms.description`.

```
<meta name="dcterms.description" lang="pl" content="Super hiper wpi
s na temat czegokolwiek">
<meta name="dcterms.subject" lang="pl" content="Bardzo ważne słowa
 kluczowe; rozdzielone średnikiem">
```

Chciałbym tutaj zwrócić uwagę, że warto także podać atrybut `[lang]` wskazujący na język użyty w metadanych. W końcu to międzynarodowy standard i niekoniecznie opis dokumentu musi być po polsku.

Kilka uwag o dostępności

Pisząc ten poradnik, skupiałem się głównie na kwestiach semantyki. Skutkiem ubocznym tego jest automatyczne zwiększenie dostępności tak stworzonej strony WWW. Na chwilę obecną nasza strona powinna być już naprawdę dobrze dostosowana dla osób niepełnosprawnych (zwłaszcza po dodaniu kilku rzeczy z opisanego wyżej standardu WAI-ARIA), lecz kilka rzeczy można jeszcze poprawić.

Na początku `body` warto dodać link odsyłający nas bezpośrednio do `main` danej strony (przy użyciu opisanych wyżej kotwic) – tzw. skip link (https://web.archive.org/web/20170419061424/http://internet-bez-barier.com/skip-linki-czym-sa-i-do-czego-sluza/):

```
<a href="#tresc" class="sr-only">Przejdź do treści</a>
```

Zwiększy to użyteczność dla tych, którzy korzystają z przeglądarek głosowych lub mają zaburzenia psychomotoryczne i są zmuszeni używać jedynie klawiatury. Dla innych ukryjemy to w CSS robiąc np. tak:

```
.sr-only {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;
  white-space: nowrap; /* 1 */
}
.sr-only:active,
.sr-only:focus {
  clip: auto;
  height: auto;
  margin: 0;
  overflow: visible;
  position: static;
  width: auto;
  white-space: inherit;
}
```

Oczywiście link będzie czuły na TAB.

Warto także dodać, że niektóre przeglądarki co prawda przeskoczą do odpowiedniego elementu, ale zachowają starą kolejność TAB-owania elementów (czyli będzie TAB-ować menu, które chcieliśmy przeskoczyć) – dlatego dla elementów, do których mamy zamiar linkować, warto dorzucić `[tabindex=-1]` (https://www.viget.com/articles/skip-link-primer/).

Przy okazji warto zauważyć, że bezpośrednia zabawa z atrybutami `[tabindex]` czy `[accesskey]` jest średnio przydatna. Narzucony z góry `[tabindex]` może zrobić więcej szkody niż pożytku (https://bitsofco.de/how-and-when-to-use-the-tabindex-attribute/) i w gruncie rzeczy powinien służyć wyłącznie do uczynienia focusowalnymi tych elementów, które natywnie nie są (czyli *de facto* winien być używany tylko i wyłącznie w połączeniu z JS i ARIA). Tak samo wydaje mi się, że skróty klawiaturowe lepiej i wygodniej dla użytkownika zrobić jest w JS (chociaż istnieje propozycja naprawienia `[accesskey]` (https://chaals.github.io/accesskey/index.src.html)).

Dla większości elementów sekcjonujących (`section, article, aside`) należy dodać nagłówki. Oczywiście nie ma sensu na chamca wtryniać nagłówków dla każdej sekcji – czasami "Untitled section" jest dozwolone. Lepszy jest brak nagłówka niźli bezsensowny nagłówek. Inna rzecz, że jeśli coś nie ma nagłówka, to prawdopodobnie nie jest sekcją. A jeśli już nagłówki *naprawdę* nie pasują do layoutu i nie możemy go poprawić tak, aby pasowały, to same nagłówki ukryć można przed wizualnymi przeglądarkami podobną techniką, jak ta dla już wspominanego linku `.sr-only`. Warto pamiętać o sensownych nagłówkach dla sekcji, bo tworzą swoisty schemat nawigacyjny po stronie, który można wykorzystać do szybkiego przeskakiwania między częściami strony i identyfikowania ich. Tutaj warto jeszcze raz przypomnieć, że każda sekcja powinna mieć odpowiednie `[id]` w języku strony. Dzięki temu będzie można linkować do każdej sekcji, co znacząco zwiększa użyteczność strony. Podobna technika jest stosowana w przypadku tego tutorialu, gdzie można zauważyć najczęściej stosowany wzorzec dla tworzenia identyfikatorów: slug z treści nagłówka.

Czasami zdarza się też, że musimy zapewnić jeszcze większy poziom dostępności. Tutaj na pomoc przychodzi nam standard WCAG 2.1 (https://w3c.github.io/wcag21/guidelines/), dokładnie opisujący wszelkie aspekty dotyczące dostępności stron WWW. Jest on podzielony na 3 poziomy (A, AA i AAA), określające stopień dostępności (od minimum do sensownego maksimum). Standard ten opisuje **naprawdę wszystko**, co webmaster powinien wiedzieć o dostępności, stąd można go traktować jako swoistą biblię. Istnieje także oficjalny katechizm (https://www.w3.org/WAI/WCAG21/quickref/).

> WCAG powinien znać **KAŻDY** szanujący się programista sieciowy. Zasady określone w tym standardzie nie są trudne w zastosowaniu (większość sprowadza się do poprawnego użycia HTML-a), a przynoszą spore korzyści w dziedzinie dostępności.

Jeśli przestrzegasz zasad semantyki HTML i nie zapominasz o zasadzie Progressive Enhancement, wówczas prawdopodobnie nie będziesz musiał osobno zajmować się kwestiami dostępności strony. Wyjątkiem będą tu kwestie związane z kontrastem, gdyż te bardzo łatwo

przeoczyć.

Ostatnie poprawki

To już prawie koniec! Co jeszcze warto zmienić?

Wydajność i bezpieczeństwo

- Zacznijmy od tytułu strony: warto się upewnić, że jest w formacie `<Nazwa podstrony> <separator> <Nazwa witryny>`, np. „Projekty @ Comandeer's Homepage”. Ma to duże znaczenie zarówno z punktu widzenia użyteczności (widać od razu, co dana karta przeglądarki zawiera), jak i dostępności (czytnik ekranowy przeczyta najpierw nazwę podstrony, a dopiero potem całą resztę).
- Wszystkie skrypty, które nie muszą blokować renderowania strony (https://addyosmani.com/blog/script-priorities/) (w naszym wypadku wyłącznie HTML5 Shiv powinien), warto wyposażyć w atrybut `[defer]` (https://calendar.perfplanet.com/2016/prefer-defer-over-async/), dzięki któremu zostaną wykonane dopiero po wczytaniu się strony. Jeżeli są to naprawdę mało istotne skrypty (np. reklamy), dodatkowo można przenieść je na koniec `body`. Jeśli mamy bardzo dużo JS, ze złożonymi relacjami pomiędzy poszczególnymi plikami, warto rozważyć architekturę modułową (https://github.com/umdjs/umd).
- Można się w ogóle pokusić o serwowanie CSS, JS i obrazków z CDN (https://en.wikipedia.org/wiki/Content_delivery_network). Jest to jedno z zaleceń Google odnośnie szybkości wczytywania stron: rozłożenie wczytywania na 2 paralelne domeny. Jedna serwuje dynamiczną stronę (czyli PHP i generujemy blogaska), a druga serwuje wszystko, co statyczne. Jednak rozłożenie wczytywania strony to tylko część zalet i bardziej rozbudowane CDN-y korzystają choćby z geolokalizacji, żeby zasysać zasoby z serwera jak najbliżej użytkownika, aby czas wczytywania był jeszcze krótszy. Istnieją także darmowe CDN-y, np. jsDelivr (https://www.jsdelivr.com/). Bardzo ważnym przeciwskazaniem dla CDN-ów jest fakt, że nie mamy nad nimi kontroli, co sprawia, że *de facto* uzależniamy swoje bezpieczeństwo od bezpieczeństwa zewnętrznej usługi. Warto mieć to na uwadze. Innym przeciwskazaniem jest fakt, że każda strona ma oddzielny cache (https://blog.josephscott.org/2019/10/16/prepare-for-fewer-cache-hits-as-chrome-partitions-their-http-cache/), a więc zalety korzystania z CDN są obecnie zdecydowanie mniejsze niż kiedyś – zwłaszcza w przypadku HTTP/2+.
- A jeśli już mówimy o bezpieczeństwie zasobów pobieranych z zewnętrznych źródeł, obowiązkową lekturą jest instrukcja użycia Subresource Integrity (https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity).
- Warto też stosować build process, podczas którego będziemy minifikować kod HTML, łączyć i minifikować pliki CSS i JS (być może nawet z przygotowywaniem paczek dla poszczególnych podstron) oraz kompresować obrazki (np. przy pomocy ImageOptim (https://imageoptim.com/) czy Squoosh (https://squoosh.app/)). Bardzo prymitywny

przykład takiego rozwiązania można zobaczyć w repozytorium mojej strony domowej (https://github.com/Comandeer/comandeers-homepage).
- Jak lubimy eksperymentować, to warto przejść całkowicie na HTTPS i protokół HTTP/2 (https://http2.github.io/) czy nawet HTTP/3 (https://blog.cloudflare.com/http3-the-past-present-and-future/). Zwiększy to zarówno bezpieczeństwo, jak i wydajność naszej strony.
- Jak już jesteśmy przy HTTPS, to warto wspomnieć o HSTS (https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security).
- Kontynuując ten temat, jest jeszcze kilka innych pomocnych nagłówków (https://www.owasp.org/index.php/List_of_useful_HTTP_headers), z czego najwięcej uwagi warto poświęcić `Content-Security-Policy` (https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy), które w swojej najnowszej odsłonie pozwoli nam zabezpieczyć się nawet przed dziurawymi CDN-ami (https://speakerdeck.com/mikispag/making-csp-great-again-michele-spagnuolo-and-lukas-weichselbaum).
- Warto też dodać plik zawierający politykę bezpieczeństwa (https://securitytxt.org/).

Integracja z zewnętrznymi usługami

- Warto dodać web app manifest (http://html5doctor.com/web-manifest-specification/), co pozwoli "uaplikacjowić" się naszemu blogowi. To doskonały wstęp do stworzenia z naszej strony pełnoprawnego Progressive Web Application (https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/).
- Oprócz RSS można dodać także Atom (https://en.wikipedia.org/wiki/Atom_(standard)), bo to o wiele lepszy format i lepiej się z nim pracuje. Ostatnio powstał też JSON Feed (https://jsonfeed.org/).
- Dla obrazków w treści artykułu warto stosować ścieżki bezwzględne (te z `https://` na początku), bo treść może być udostępniana także przez RSS i Atom i może pojawić się problem z odnalezieniem właściwego obrazka. Gdy to tylko możliwe, należy wymuszać HTTPS (https://twitter.com/paul_irish/status/588502455530311680). Zapewnia to większe bezpieczeństwo, a w przyszłości umożliwi korzystanie z HTTP/2+.
- Jeśli masz stronę domową/firmową, rozważ umieszczenie `link[rel=author]`:

```
<link rel="author" href="https://www.comandeer.pl">
```

- Można dodatkowo zastosować humans.txt (http://humanstxt.org/).
- Powstał też standard Open Graph Protocol (https://ogp.me/). Facebook posiada swoje metatagi (https://davidwalsh.name/facebook-meta-tags) a Twtiter nie pozostaje mu dłużny (https://dev.twitter.com/docs/cards).
- Warto też serwować ikonkę strony. Najlepiej mieć podstawową – `favicon.ico` – w głównym folderze witryny. Wtedy, nawet jeśli nie zamieścimy `link[rel=icon]` w kodzie, przeglądarki sobie ją pobiorą (tak, przeglądarki po prostu na chama szukają ikonki, śląc

requesty pod `http://naszastrona.pl/favicon.ico`). Mając taką podstawową w zapasie, można próbować wcisnąć browserom np coś w PNG właśnie przy pomocy `link[rel=icon]`. IE < 9 i tak tego nie rozumie, uparcie szukając `link[rel=shortcut]`, zatem ono dostanie ico a reszta ładny PNG (albo nawet animację w GIF). Do tego dochodzą jeszcze np. ikonki dotykowe dla iUrządzeń (https://mathiasbynens.be/notes/touch-icons) i inne takie, które też można (a jeśli strona ma być dla mobilnych też – nawet trzeba) zamieścić. Są od tego odpowiednie narzędzia (https://realfavicongenerator.net/). Uwaga! Generowany kod to prawdziwa kobyła. Tak się to kończy, gdy nie istnieje jeden, powszechny standard (https://css-tricks.com/favicon-quiz/).
- Istnieje także generator PWA (https://www.pwabuilder.com/), powstały z dawnego projektu Manifold.js.
- Chrome na mobilnych urządzeniach pozwala również zmienić kolor paska adresu (https://developers.google.com/web/updates/2014/11/Support-for-theme-color-in-Chrome-39-for-Android?hl=en):

```
<meta name="theme-color" content="#db5945">
```

Kompatybilność

- Warto używać grida, flexboxa oraz media-queries, aby strona sama się dostosowywała do urządzenia użytkownika (nurt responsive webdesign). Opłaca się, bo Google lubi strony mobile friendly (https://www.google.com/webmasters/tools/mobile-friendly/) (a mój telefon to popiera!).
- Dla IE wypada słać nagłówek `X-UA-Compatible` (https://github.com/h5bp/html5-boilerplate/blob/b5d6e7b1613fca24d250fa8e5bc7bcc3dd6002ef/dist/doc/html.md#x-ua-compatible) ustawiony na `IE=edge`. Wymusza to renderowanie strony przy pomocy najnowszych standardów w IE >= 8. Możesz też naprawdę kulturalnie poinformować użytkownika o tym, jak bardzo przestarzałą przeglądarkę używa (https://browser-update.org/pl/).
- Na końcu warto poświęcić chwilkę i przetestować stronę, poczynając od walidatora (https://validator.w3.org/nu), przechodząc do testu dostępności (https://wave.webaim.org/) a na teście szybkości kończąc (https://www.webpagetest.org/). Sprawdzenie strony na kilku różnych urządzeniach lub BrowserStack (https://www.browserstack.com/)/Sauce Labs (https://saucelabs.com/) też jest dobrym pomysłem.
- A już na naprawdę samym końcu naprawdę zachęcam, aby odpalić JAWS-a (https://www.freedomscientific.com/Products/software/JAWS/), NVDA (https://www.nvaccess.org/), VoiceOver (https://www.apple.com/accessibility/mac/vision/) lub jakiś inny czytnik ekranowy (chociaż bardziej bym ufał tej trójce, w kolejności wymieniania) i *posłuchać* swojej strony. Bo może się okazać, że nie za bardzo jest czego słuchać…

Więcej grzechów nie pamiętam.

Nasz blog

Oto i pełny przerobiony kod z paroma dodatkami (https://tutorials.comandeer.pl/res/html5-blog/final.html):

```
<!DOCTYPE html>
    <html lang="pl" dir="ltr" vocab="http://schema.org" typeof="Blo
g">
        <head>
            <meta charset="UTF-8">

            <meta property="url" content="https://example.net/Super
-hiper-wazny-wpis">
            <meta name="description" content="Super hiper wpis na t
emat czegokolwiek">
            <meta name="dcterms.description" lang="pl" content="Sup
er hiper wpis na temat czegokolwiek">
            <meta name="keywords" content="Bardzo ważne słowa klucz
owe">
            <meta name="dcterms.subject" lang="pl" content="Bardzo
 ważne słowa kluczowe; rozdzielone średnikiem">
            <meta name="application-name" content="Super hiper blog
">
            <meta name="msapplication-tooltip" content="Niesamowity
blog o niesamowitych sprawach">
            <meta name="msapplication-starturl" content="http://exa
mple.net">
            <meta name="msapplication-window" content="width=1024;h
eight=768">
            <meta property="og:site_name" content="Example.net - fa
jowy blog, na którym bloguję">
            <meta property="og:url" content="https://example.net">
            <meta property="og:title" content="Super hiper ważny wp
is | Example.net - fajowy blog, na którym bloguję">
            <meta property="og:image" content="https://example.net/
images/favicon.png">

            <link rel="author" href="https://plus.google.com/108791
847143656151689/">
            <link rel="alternate" type="application/rss+xml" title=
"Wpisy na RSS" href="http://example.net/feed">
            <link rel="alternate" type="application/atom+xml" title
="Wpisy na ATOM" href="http://example.net/atom">
            <link rel="index" title="Strona główna" href="https://e
xample.net">
```

```
            <link rel="prev" title="Jakiś hiper poprzedni wpis" hre
f="https://example.net/Jakis-hiper-poprzedni-wpis">
            <link rel="next" title="Jakiś super następny wpis" href
="https://example.net/Jakis-super-nastepny-wpis">
            <link rel="canonical" href="https://example.net/Super-h
iper-wazny-wpis">
            <link rel="pingback" href="https://example.net/xmlrpc.p
hp">
            <link rel="icon" href="https://example.net/images/favic
on.png" type="image/png">
            <link rel="apple-touch-icon" href="https://example.net/
images/apple-touch-icon.png">
            <link rel="stylesheet" href="https://example.net/css/st
yle.css">

            <title property="name">Super hiper ważny wpis | Exampl
e.net - fajowy blog, na którym bloguję</title>

            <!--[if lt IE 9]>
                <script src="https://cdn.jsdelivr.net/npm/html5shiv
@3.7.3/dist/html5shiv-printshiv.min.js"></script>
            <![endif]-->
        </head>
        <body>
            <a href="#tresc" class="focus-only">Przejdź do treści</
a>

            <header id="naglowek">
                <nav id="menu" vocab="http://schema.org" typeof="Si
teNavigationElement">
                    <ul>
                        <li>
                            <a href="https://example.net" rel="inde
x">
                                <p>Example.net - fajowy blog, na kt
órym bloguję</p>
                                <p>Motto</p>
                            </a>
                        </li>
                        <li><a href="whatever.html">Whatever</a></l
i>
```

```
                        <li><a href="wherever.html">Wherever</a></l
i>
                        <li><a href="whenever.html">Whenever</a></l
i>
                    </ul>
                </nav>
                <form action="search.php" method="post" role="searc
h">
                    <p>
                        <label class="visuallyhidden" for="search-i
nput">Szukaj:</label>
                        <input type="search" id="search-input" name
="q" placeholder="Wpisz szukaną frazę…" required>
                        <button type="submit" name="submit" value="
1">Szukaj</button>
                    </p>
                </form>
            </header>

            <main id="tresc" class="hfeed" property="mainEntityOfPa
ge" tabindex="-1">
                <article class="post h-entry" property="blogPost" v
ocab="http://schema.org" typeof="BlogPosting" id="super-hiper-wazny
-wpis">
                    <header>
                        <h1 class="p-name" property="headline">
                            <a href="https://example.net/Super-hipe
r-wazny-wpis" class="bookmark u-uid u-url" rel="bookmark" title="Pe
rmalink do Super hiper ważny wpis">Super hiper ważny wpis</a>
                        </h1>
                        <p class="post-info">Opublikowano <time dat
etime="2011-01-07T20:40:06+00:00" class="dt-published" property="da
tePublished">07.01.2011</time> przez <a href="https://example.net/a
uthor" property="author" vocab="http://schema.org" typeof="Person"
rel="author" class="author p-author h-card"><span property="name">C
omandeer</span></a></p>
                    </header>
                    <div class="e-content" property="articleBody">
                        <figure property="image">
                            <picture>
                                <source media="(min-width: 45em)" s
```

```
rcset="large-1.jpg 1x, large-2.jpg 2x">
                                <source media="(min-width: 18em)" s
rcset="med-1.jpg 1x, med-2.jpg 2x">
                                <source srcset="small-1.jpg 1x, sma
ll-2.jpg 2x">
                                <img src="small-1.jpg" alt="Kwadrat
owa ramka z czerwonym krzyżykiem w środku">
                            </picture>
                            <figcaption>Bardzo ważny obrazek[…],któ
ry jest bardzo ważny</figcaption>
                        </figure>
                        <p>Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Proin eu justo enim, ac faucibus massa. Vestibulu
m in elit aliquam purus sollicitudin adipiscing ac et sapien. Curab
itur eleifend justo diam, et viverra nisi. Quisque a ipsum vehicula
nunc vestibulum posuere. Suspendisse potenti. Vestibulum ante ipsum
primis in faucibus orci luctus et ultrices posuere cubilia Curae; V
estibulum porttitor, neque eu congue fermentum, velit ante pellente
sque arcu, a posuere risus tortor vel ante. Donec et neque at odio
 hendrerit mattis sed eu tellus. Nullam accumsan leo ut felis susci
pit vehicula posuere erat eleifend. Donec semper lorem eu nibh tinc
idunt varius.</p>
                        <h2 id="podtytul">Podtytuł</h2>
                        <p>Vivamus leo arcu, convallis id iaculis s
it amet, fringilla et nunc. Donec aliquam, justo at mollis porta, e
nim lorem tempor eros, id iaculis arcu elit ac sem. In et erat eu m
etus ornare vestibulum non at arcu. Integer in nisi massa. Etiam nu
lla felis, rhoncus non pharetra sed, vehicula eu risus. Vestibulum
 sodales nunc nisi, vitae gravida nisl. Quisque lacus ipsum, commod
o ac hendrerit vitae, porta dapibus tortor. Mauris sed risus diam.
 Morbi lacus elit, euismod dapibus interdum id, fermentum et mi. Na
m at neque orci. Sed ac hendrerit augue. Vestibulum pharetra mattis
mattis. Sed porta turpis dolor, condimentum blandit libero. Aliquam
non nisi mi. Nulla nec sem elit. Duis blandit viverra lacus, in con
vallis mauris dictum et. Ut eget risus enim. Suspendisse potenti. A
enean leo odio, luctus quis eleifend quis, consectetur sit amet ips
um. Curabitur commodo leo ac risus dignissim at porttitor turpis fr
ingilla.</p>
                        <p>Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Nullam ullamcorper neque a magna sodales sodales.
Nulla id enim sem, sit amet ultrices dolor. Vestibulum sollicitudin
```

dolor quis quam vehicula egestas. Ut urna erat, gravida a tristique non, congue nec elit. Aenean tincidunt purus sed nisl semper sagitt is. Vestibulum ante ipsum primis in faucibus orci luctus et ultrice s posuere cubilia Curae; Vivamus in orci sit amet ligula posuere mo llis a in diam. Nam enim tortor, ultrices quis laoreet nec, egestas non leo. Nam non fringilla turpis. Quisque vitae pellentesque nibh. Etiam dictum tincidunt ultricies. Cras sit amet aliquam odio. Morbi quis urna a nulla egestas placerat sed in lacus. Morbi id urna laci nia nunc bibendum ultricies. In pretium leo non odio feugiat portti tor. Donec aliquet purus in purus pretium aliquam. Etiam aliquet ni bh ut nibh semper egestas. Cras in purus quam, nec posuere diam. Al iquam erat volutpat.

Morbi porta blandit nisi at fermentum. P raesent vel sagittis urna. Pellentesque ipsum eros, vestibulum id g ravida vitae, dictum vel sapien. Nam placerat lacus non lacus facil isis vitae hendrerit nibh adipiscing. Sed aliquet nisi ligula, eu v iverra turpis. Curabitur tincidunt arcu in enim tempus sit amet auc tor nunc luctus. Praesent dui turpis, lobortis in euismod quis, fri ngilla at felis. Nullam ligula purus, mattis nec tristique non, ali quam vitae sem. Quisque sit amet magna magna, eget aliquam orci. Al iquam placerat diam sit amet erat gravida fermentum. Maecenas males uada nibh quis lorem pellentesque laoreet. Vestibulum pulvinar tinc idunt fringilla. Nullam pellentesque felis ac nisi dictum egestas. Nunc metus mi, euismod non convallis nec, varius et felis. Pellent esque vel lacus turpis. In hac habitasse platea dictumst. Morbi sod ales mauris a nisi feugiat pulvinar.

Vivamus vitae lectus ut dui luctus iacul is. Pellentesque quis lectus id ipsum venenatis bibendum. Pellentes que non magna nec purus mollis suscipit. Nulla vestibulum interdum ante, vestibulum ullamcorper est vulputate vitae. Pellentesque hab itant morbi tristique senectus et netus et malesuada fames ac turpi s egestas. Quisque aliquet mollis rhoncus. Phasellus imperdiet posu ere bibendum. Quisque consequat imperdiet nibh vitae pretium. Suspe ndisse quis eros libero, non luctus mauris. Nulla laoreet nibh at e ros scelerisque condimentum. Cras aliquam velit at orci luctus et t incidunt arcu scelerisque. Donec at quam magna. Integer leo leo, ul trices facilisis bibendum vel, feugiat interdum neque. Fusce vitae nibh sapien, sit amet consequat velit. Fusce adipiscing tortor id nibh fringilla sollicitudin. Donec sodales sapien in sapien rutrum rutrum. Nam quis enim nec nibh varius mattis nec in ante. Morbi non est eu diam elementum interdum. Morbi at odio non libero tempus ele

```
ifend nec quis ligula. Cras pellentesque mauris sit amet felis vest
ibulum tincidunt.</p>
                    </div>
                    <footer>
                        <section id="tagi" class="tags" property="a
rticleSection">
                            <h2>Tagi</h2>
                                <ul>
                                    <li>
                                        <a href="https://example.ne
t/tag/Tag1" rel="tag" class="p-category">Tag1</a>
                                    </li>
                                    <li>
                                        <a href="https://example.ne
t/tag/Tag2" rel="tag" class="p-category">Tag2</a>
                                    </li>
                                </ul>
                        </section>
                    </footer>

                    <section id="komentarze" class="comments">
                        <h2>Komentarze</h2>
                        <ol>
                            <li id="comment-25" property="comment"
vocab="http://schema.org" typeof="Comment">
                                <div class="comment-meta">
                                    <a href="#comment-25">
                                        <time datetime="2011-01-07T
20:41:06+00:00" property="dateCreated">07.01.2011, 20:41</time>
                                    </a>
                                    <b property="author" vocab="htt
p://schema.org" typeof="Person"><a href="http://example.net/" prope
rty="name url"><img alt="" src="http://avatary.gdziekolwiek.com/Com
andeer" height="48" width="48" property="image">Comandeer</a> skome
ntował</b>
                                </div>
                                <p property="text">Ale komentarz!</
p>
                            </li>
                            <li id="comment-26" property="comment"
vocab="http://schema.org" typeof="Comment">
```

```
                                <div class="comment-meta">
                                    <a href="#comment-26">
                                        <time datetime="2017-01-08T
20:08:30+01:00" property="dateCreated">07.01.2017, 20:08</time>
                                    </a>
                                    <b property="author" vocab="htt
p://schema.org" typeof="Person"><a href="http://example.net/" prope
rty="name url"><img alt="" src="http://example.net/avatary/Anonim"
height="48" width="48" property="image">Anonim</a> skomentował</b>
                                </div>
                                <p property="text">Inny komentarz!
</p>
                            </li>
                        </ol>

                        <form action="https://example.net/comment.p
hp" method="post" id="dodaj-komentarz">
                            <h3>Dodaj komentarz</h3>
                                <p>
                                    <label for="comment-author">Nic
k</label>
                                    <input type="text" name="author
" id="comment-author" required>
                                </p>
                                <p>
                                    <label for="comment-email">E-ma
il</label>
                                    <input type="email" name="email
" id="comment-email" required>
                                </p>
                                <p>
                                    <label for="comment-url">Strona
</label>
                                    <input type="url" name="url" id
="comment-url">
                                </p>
                                <p>
                                    <label for="comment-comment">Ko
mentarz</label>
                                    <textarea name="comment" id="co
mment-comment" cols="48" rows="7" required></textarea>
```

```
                                </p>
                                <p>
                                    <button name="submit" type="sub
mit" id="comment-submit">Wyślij</button>
                                </p>
                                <input type="hidden" name="comment_
post_ID" value="88">
                        </form>
                    </section>
                </article>
            </main>

            <aside id="aside">
                <h2>Dodatkowe informacje</h2>
                <article class="author h-card" vocab="http://schem
a.org" typeof="Person">
                    <h3>O autorze blogaska</h3>
                    <p>Nazywam się <a href="https://example.net/aut
hor" class="fn p-name u-url" property="name url">Comandeer</a>, mie
szkam w uroczym miasteczku <span class="adr p-locality" property="a
ddress" vocab="http://schema.org" typeof="PostalAddress"><span prop
erty="addressLocality">Świętochłowice</span></span> i interesuję si
ę webmasterką.</p>
                </article>
                <article class="tags cloud">
                    <h3>Tagi</h3>
                    <ul>
                        <li>
                            <a href="https://example.net/tag/Tag1">
Tag1</a>
                        </li>
                        <li>
                            <a href="https://example.net/tag/Tag2">
Tag2</a>
                        </li>
                        <li>
                            <a href="https://example.net/tag/Tag3">
Tag3</a>
                        </li>
                        <li>
                            <a href="https://example.net/tag/Tag2">
```

```
Tag4</a>
                        </li>
                    </ul>
                </article>
                <article class="archive">
                    <h3>Archiwum</h3>
                    <ol>
                        <li>
                            <a href="https://example.net/archiwum/2
011/01">Styczeń 2011</a>
                        </li>
                        <li>
                            <a href="https://example.net/archiwum/2
010/12">Grudzień 2010</a>
                        </li>
                    </ol>
                </article>
            </aside>

            <footer id="stopka">Copyright &copy; 2017 by <a href="h
ttps://example.net/author" rel="author">Comandeer</a></footer>

            <script src="https://example.net/jquery.js" defer></scr
ipt>
            <script src="https://example.net/scripty.js" defer></sc
ript>
        </body>
    </html>
```

Przydatne linki

Standardy i oficjalne materiały

- HTML Living Standard (https://html.spec.whatwg.org/)
- Specka dla developerów (https://html.spec.whatwg.org/dev/)
- Specyfikacja HTML 5.2 (https://www.w3.org/TR/html52)
- Specyfikacja WCAG 2.1 (https://w3c.github.io/wcag21/guidelines/) – i tu anegdota: warto ufać specyfikacjom W3C publikowanym na GitHubie, ponieważ te pod adresami zaczynającymi się od `https://www.w3.org/TR` to śmieci („"/TR/ stands for trash" ["/TR/ oznacza śmieci"]” (https://github.com/nolanlawson/html5workertest/issues/6#issue-

169542408)), czyli specyfikacje często nieaktualne j niezmieniane po ostatecznym opublikowaniu
- Oficjalny poradnik WCAG 2.0 (https://www.w3.org/WAI/WCAG21/quickref/)
- Specyfikacja WAI-ARIA (https://w3c.github.io/aria/)
- Oficjalny poradnik dla chcących bawić się WAI-ARIA (https://w3c.github.io/aria-practices/)
- Specyfikacje i oficjalne materiały o mikroformatach (http://microformats.org/wiki/Main_Page)
- Informacje o Dublin Core (https://www.dublincore.org/)
- Oficjalna część specyfikacji HTML, opisująca użycie atrybutu `[alt]` (https://html.spec.whatwg.org/multipage/images.html#alt)
- Oficjalna informacja o responsywnych obrazkach (https://usecases.responsiveimages.org/)
- Specyfikacja HTTP/2 (https://tools.ietf.org/html/rfc7540)
- Specyfikacja Content Security Policy (https://w3c.github.io/webappsec-csp/)
- Opis standardu daty (https://www.w3.org/TR/NOTE-datetime)

Poradniki

- Perfekcyjny przykład bloga w HTML5 (https://www.paciellogroup.com/blog/)
- Microdata (http://diveintohtml5.info/extensibility.html)
- Doktorzy o microdata (http://html5doctor.com/microdata/)
- DublinCore (https://kurs.browsehappy.pl/HTML/DublinCore)
- Techniki dobierania odpowiedniego atrybutu `[alt]` (https://webaim.org/techniques/alttext/)
- WAI-ARIA (https://alistapart.com/article/waiaria)
- Krótka uwaga dla nadużywających ARIA (http://html5doctor.com/on-html-belts-and-aria-braces/)
- Trochę o tagu `time` (http://html5doctor.com/the-time-element/)
- Lista najczęstszych błędów (http://html5doctor.com/avoiding-common-html5-mistakes/)
- Prosty schemat pomagający dobrać odpowiedni tag (http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf)
- Progressive Enhancement (https://webroad.pl/inne/3722-progressive-enhancement-zapomniany-fundament)
- Krótki artykuł o walidacji (https://webroad.pl/html5-css3/3925-validate-or-not-to-validate-that-is-the-question)

Narzędzia

- Oficjalny walidator (https://validator.w3.org/nu)
- Najpełniejszy audyt strony (https://web.dev)
- Narzędzie do sprawdzania dostępności strony (https://wave.webaim.org/)
- Narzędzie do mierzenia wydajności strony (https://www.webpagetest.org/)

- Inne narzędzie do mierzenia wydajności strony (https://developers.google.com/speed/pagespeed/insights/)
- Narzędzie do testowania microdata/mikroformatów/RDFa (https://search.google.com/structured-data/testing-tool/)
- Prosty sposób na sprawdzenie wsparcia HTML i Web APIs w różnych przeglądarkach (https://caniuse.com/)
- Testowanie strony w różnych przeglądarkach (https://www.browserstack.com/)
- Inne testowanie strony w różnych przeglądarkach (https://saucelabs.com/)
- Pokaz możliwości Sieci (https://whatwebcando.today/)
- Walidator reguł CSP (https://csp-evaluator.withgoogle.com/)
- Miernik "otyłości" strony (https://www.webbloatscore.com/) (radzę nie traktować za poważnie!)
- Prosty walidator formularzy (https://formlinter.com/)
- Walidator dostępności strony (https://www.gewoontoegankelijk.nl/en)
- Czytnik ekranowy JAWS (https://www.freedomscientific.com/Products/software/JAWS/)
- Czytnik ekranowy NVDA (https://www.nvaccess.org/)
- Czytnik ekranowy VoiceOver (https://www.apple.com/accessibility/mac/vision/)

Poprawki i takie tam

▼ changelog

- **07.11.2020**:
  - Uaktualnienie linków do kursów HTML.
- **02.01.2020**:
  - Przejście na HTML LS i porzucenie terminu HTML5 na rzecz HTML.
  - Odejście od sformułowań typu "nowe znaczniki".
  - Dodanie sekcji o normalizacji stylów.
  - Przeredagowanie fragmentu o wsparciu dla starszych przeglądarek.
  - Przejście na `normalize.css` od `csstools`.
  - Uaktualnienie linków.
  - Uaktualnienie listy dozwolonych `[rel]`.
  - Usunięcie kilku historycznych ciekawostek.
  - Uaktualnienie wykorzystania ARIA w przykładach.
  - Uaktualnienie dobrych rad.
- **23.10.2019**:
  - Poprawienie linku do książki Ferrante'a.
- **12.05.2019**:
  - Poprawienie niewłaściwej struktury nagłówków wewnątrz artykułu.
- **12.03.2019**:
  - Podlinkowanie do lokalnej kopii książki Ferrante.
  - Przeredagowanie kilku fragmentów, łącznie z usunięciem tych wprowadzających niepotrzebny szum informacyjny.
  - Dodanie info o remedium i sanityzacji CSS.

    - Poprawienie formatowania kodu.
    - Przeniesienie informacji o dwóch standardach HTML na początek tekstu.
    - Dodanie informacji o HFP.
    - Dodanie informacji o `[rel=noreferrer]` oraz `[rel=noopener]`.
    - Aktualizacja linków.
    - Aktualizacja zalecenia co do wczytywania skryptów.
    - Dodanie informacji o security.txt oraz humans.txt.
    - Przepisanie kodu na RDF-a.
- **16.12.2017** – zmiana struktury nagłówków, uaktualnienie opisu outline'u, zmiana struktury komentarzy, uaktualnienie opisu komentarzy, polskie tłumaczenia cytatów ze specyfikacji, przejście na cytowanie najnowszej wersji specyfikacji W3C, aktualizacja linków, liczne doprecyzowania
- **08.01.2017** – przeredagowanie; dodanie informacji o `script[type=module]`; dodanie informacji o bibliotekach has.js i polyfill.io (http://polyfill.io) oraz teście „cut the mustard”; dodanie informacji o alternatywach dla HTML5 shiv oraz komentarzach warunkowych; dodanie informacji o dwóch standardach HTML; usunięcie odwołań do opisu hierarchii nagłówków opartej na sekcjach; dodanie informacji o `[id]` nagłówków/sekcji; dodanie informacji o dostępności `nav`; zmiana kodu formularza; aktualizacja informacji o ARIA, `meta[name]` i mikroformatach; dodanie informacji o czytnikach ekranowych; dodanie informacji o wzorcowym formacie tytułu strony; dodanie linków do kilku narzędzi i czytników ekranowych; aktualizacja informacji o HTML5 Shiv; dodanie informacji o SRI
- **21.11.2016** – dodanie linków do dwóch oficjalnych walidatorów
- **24.10.2016** – aktualizacja finalnego kodu
- **23.07.2016** – dodanie info o HTML 5.1 jako CR + dodanie linku do najnowszej wersji specki HTML 5.x; zmiana przykładu perfekcyjnego bloga
- **28.05.2016** – poprawienie linków do sekcji o `[alt]` i `picture` w specyfikacji HTML 5.1
- **17.04.2016** – dodanie informacji o aktualizacji algorytmu outline'u w specyfikacji HTML5
- **09.04.2016** – dodanie linku do algorytmu wyszukiwania deklaracji kodowania w `meta` oraz linku do opisu `X-UA-Compatible` w H5BP
- **16.03.2016** – dopisanie krótkiego wyjaśnienia w fragmencie o `main` zachowującym się jak `div`
- **16.02.2016** – poprawienie linków o `[alt]`; odwrócenie changeloga; poprawienie notki o `figure`
- **03.02.2016** – lekkie uszczegółowienie opisu `figure`; przebudowa i aktualizacja sekcji "Ostatnie poprawki"; dopisanie kilku informacji o bezpieczeństwie; utworzenie sekcji o dostępności; przebudowa i aktualizacja sekcji z linkami; uaktualnienie stopki bloga
- **11.12.2015** – uaktualnienie obrazków z outlinem blogów; dodanie notki o możliwości testowania outline w walidatorze; dodanie linka do walidatora
- **26.11.2015** – mała poprawka we fragmencie o adresach zasobów na stronie
- **31.07.2015** – wyjęcie komentarzy poza stopkę artykułu
- **04.06.2015** – opisanie komentarzy, informacja o relative protocol, uzupełnienie informacji o `figure`, poprawienie wykorzystania `[rel=tag]`, notka przy `[rel]` o ich połączeniu z

konkretną stroną oraz uaktualnienie kodu bloga
- **26.04.2015** – podział tutorialu na mniejsze partie (zwłaszcza część poświęconą HTML5)
- **23.04.2015** – krótka notka na temat wystrzegania się niepotrzebnego stosowania ARIA; poprawiony kod bloga
- **09.03.2015** – ulepszenie skip linków, uaktualnienie informacji o atrybutach `[rel]`, poprawienie kodu `picture` oraz krótki dopisek o formacie JSON LD
- **19.11.2014** – poprawienie zapisu atrybutów, dodanie linku do CanIUse.com (http://CanIUse.com) i aktualizacja informacji o tagu `picture`
- **05.11.2014** – kilka poprawek związanych głównie z ARIA
- **18.08.2014** – dodanie linku do artykułu o favicons
- **04.08.2014** – poprawnie kodu formularzu (wywalenie `[tabindex]` i wyjęcie przycisku poza `dl`)
- **13.07.2014** – dodanie kilku informacji o ARIA, faviconach oraz przeredagowanie kilku fragmentów; rozróżnienie między `nav` a `menu`
- **16.04.2014** – uwaga początkowa + podlinkowanie książki Ferrante i Kursu HTML + uwagi o `[alt]`; pewna zmiana w stosunku do `[tabindex]` i `[accesskey]`
- **10.04.2014** – dodanie kilku linków, aktualizacja fragmentu o `picture`, dodanie części o podstawowych stylach i porady o testowaniu strony
- **21.11.2013** – dodanie fragmentu o `[pattern]` i kilka dodatkowych uwag na temat `aside`
- **20.11.2013** – zmiana sposobu generowania tutorialu (kod powinien być czystszy), zmiana kodu artykułu (głównie `section, footer`), wyjaśnienie wątpliwości z `cite`, aktualności związane z responsywnymi obrazkami oraz nagłówkami
- **20.09.2013** – dodałem nagłówki w kilku miejscach ostatecznego kodu bloga oraz usunąłem z niego `hgroup`
- **10.03.2013** – oddanie sprawiedliwości tagowi `main`, dodanie kilku linków, troszkę poprawek
- **10.01.2013** – dodano kilka linków, dopisano kilka informacji (m.in (http://m.in). o OpenGraph wspomniano)
- **06.01.2013** – gryzło mnie sumienie o `label` na wyszukiwarce. Toteż dopisałem, opisałem i przesadziłem ;) również dodałem/zmieniłem/poprawiłem kilka linków i doaktualizowałem pewne informacje. No i przekroczyłem limit znaków ;) kod HTML początkowy i końcowy walnięte na serwer :P
- **29.12.2012** – dopisek o elemencie `main`
- **28.11.2012** – małe poprawki w kilku miejscach, dodanie kilku linków
- **17.09.2012** – pojawił się akapicik o responsive images, zmiany w linkach
- **25.07.2012** – kilka poprawek wymuszonych zmianami w specyfikacji i trochę więcej o Schema.org (http://Schema.org) + kilka takich tam uwag o różnych elementach
- **20.10.2011** – poprawiłem kilka linków i literówek, lepiej sformatowałem tekst, uzupełniłem informację o mikrodanych, mała zmiana w markupie artykułu
- **09.01.2011** – pierwsza wersja