Tworzenie szablonu WordPress w Responsive Web Design

pracabezszefa.pl/tworzenie-szablonu-wordpress-rwd

Marek Duda 30.05.2015

Zobacz, jak przebiega **tworzenie profesjonalnego szablonu WordPress** wykonanego w technologii Responsive Web Design nazywanej w skrócie RWD?

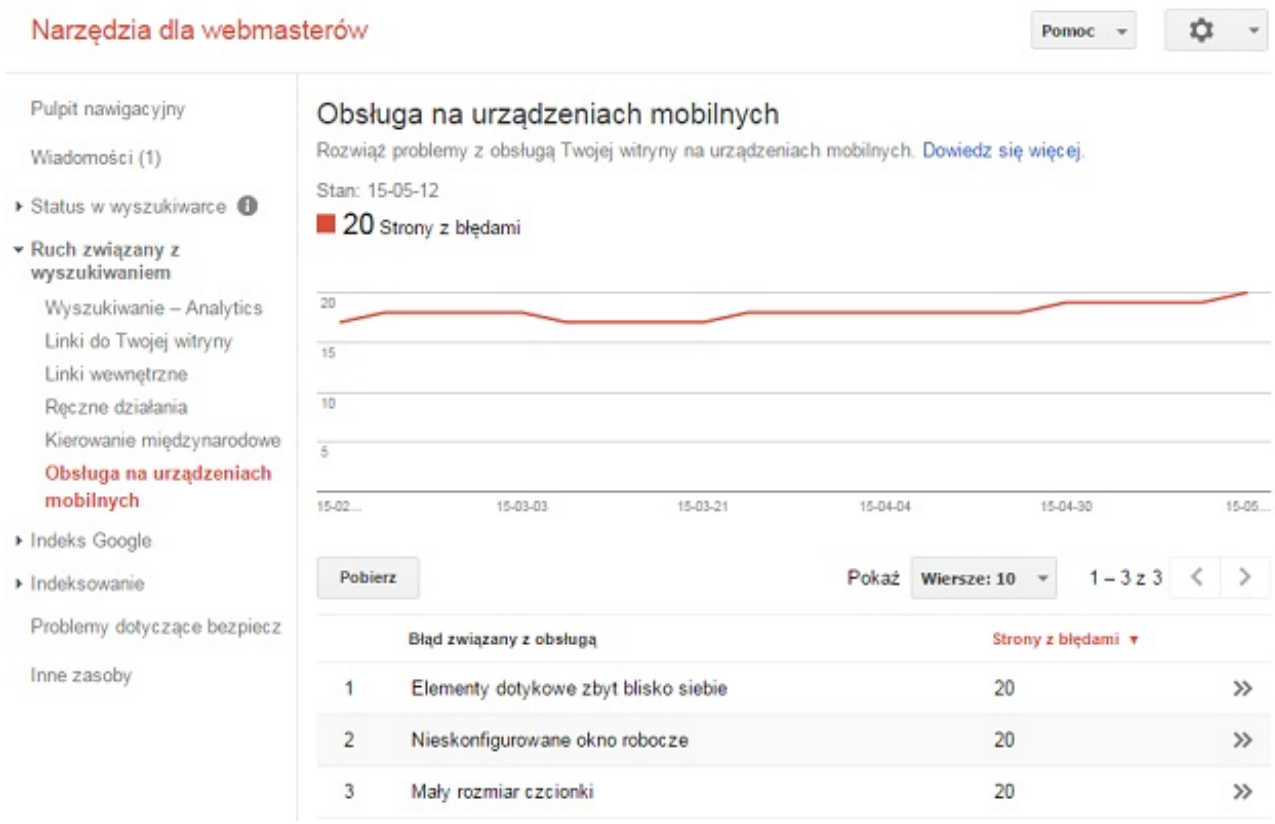
Technologia RWD to nic innego, jak stworzenie responsywnego serwisu przyjaznego technologiom mobilnym. Twój blog, czy strona internetowa musi być przyjazna mobilkom! Dlaczego?

Od 21 kwietnia 2015 jest wdrażany nowy algorytm wyszukiwarki Google dla urządzeń mobilnych. Idea algorytmu w wielkim skrócie jest następująca:

Jeżeli Twój serwis nie jest poprawnie wyświetlany na urządzeniach mobilnych, to musisz się liczyć ze spadkiem Twojej witryny w mobilnych wynikach wyszukiwania.

Mniejszy ruch w serwisie, to mniejsze zyski dla Ciebie 🙂

Sygnały ze strony Google są jak najbardziej poważne, co widać w narzędziu Google Search Console (Narzędzia dla webmasterów) dla stron niezoptymalizowanych pod technologie mobilne.



Liczba użytkowników wykorzystujących urządzenia mobilne do przeglądania zasobów internetu gwałtownie wzrasta, dlatego koniecznym jest zbudowanie szablonu przyjaznego technologiom mobilnym i przeprowadzić test zgodności z urządzeniami przenośnymi, który znajduje się na stronie: https://www.google.com/webmasters/tools/mobile-friendly/

Jeżeli nie masz ochoty, bądź brakuje Ci dostatecznej wiedzy, aby zakodować własną skórkę do WordPressa w

technologii Responsive Web Design polecam Ci wybrać profesjonalny serwis z gotowymi: darmowymi i płatnymi szablonami Template Monster.

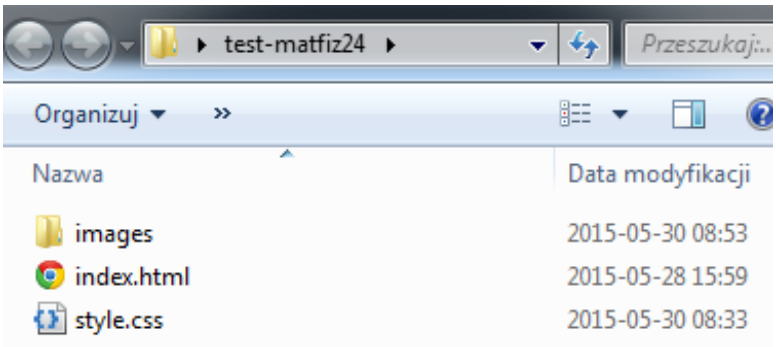
Jak stworzyć szablon WordPress w technologii RWD przyjazny wyszukiwarkom?

Okazuje się, że zbudowanie polskiego, profesjonalnego szablonu przyjaznego wyszukiwarkom jest bardzo proste.

Zbuduj szablon HTML5 i CSS3 w technologi RWD. Możesz go wykonać bezpośrednio na dysku swojego komputera. Jeżeli nie wiesz jak tworzyć szablony stron internetowych zobacz kurs HTML dla zielonych i początkujących.

Zakodowany szablon możesz pobrać tutaj.

Szablon został nazwany: **test-matfiz24** i zawiera strukturę plików tak, jak na grafice poniżej:



Katalog images współtworzy zbiór wszystkich grafik budujących szablon WordPress:

- logo.png – logo testowe serwisu
- ikona-wpisu.png – ikona wpisu

test. matfiz24.pl

IKONA WPISU MATEMATYCZNEGO

WPIS TESTOWY

- punktor_menu.png – zawiera dodatek graficzny dla pionowego menu
- tlo_menu.png – jednopikselowa grafika, która powielana w poziomie przy pomocy HTML utworzy szare tło menu poziomego.
- reklama.png – sens tej grafiki odkryjesz pod koniec wpisu

Plik index.html – pełna struktura HTML5 strony głównej szablonu


```
<!DOCTYPE html>
<html>
<head>
    <title>Test bloga matematycznego
Matfiz24.pl</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="style.css"
type="text/css" />
    <meta name="viewport" content="width=device-
width, initial-scale=1, maximum-scale=1"/>
    <script>
        for(var e,l='footer header nav article
section aside figure'.split('
');e=l.pop();document.createElement(e));
    </script>
</head>
<body>
    <header class="web">
```

```
        <div class="logo">
            <a href="/">
                <img src="images/logo.png"
alt="logo Matfiz24.pl" />
            </a>
        </div>


        <form action="/" method="get" accept-
charset="UTF-8">
            <fieldset>
                <input name="s" type="text"
placeholder="Szukaj..." value="" /><input
type="submit" value="Szukaj" />
            </fieldset>
        </form>
        <div class="clearfix"></div>


        <nav>
            <ul>
                <li>
                    <a href="/">Start</a>
                </li>
                <li>
                    <a href="#">O mnie</a>
                </li>
                <li>
                    <a href="#">Sekcje</a>
                </li>
                <li>
                    <a href="#">Zapytaj</a>
                </li>
            </ul>
        </nav>
    </header>
    <div class="clearfix"></div>

    <div class="web">
        <section>

            <article>
                <h1>
                    <a href="#">
                        Link do wpisu 1
                    </a>
                </h1>
                <a href="#">
                    <img src="images/ikona-
wpisu.png" alt="" />
                </a>
                <p>
                    Treść artykułu matematycznego.
Formatowanie tekstu, dodawanie obrazków,
                    wideo i prezentacji, które
```

```
wpłyną na unikalność całego serwisu, co zostanie
                    docenione przez wyszukiwarkę.
                </p>
                <p>
                    Vestibulum dapibus, mauris nec
malesuada fames ac turpis velit, rhoncus eu,
                    luctus et interdum adipiscing
wisi. Aliquam erat ac ipsum. Integer aliquam
                    purus.
                </p>
                <a class="more-link"
href="#">Czytaj Dalej</a>
            </article>


            <article>
                <h1>
                    <a href="#">
                        Link do wpisu 2
                    </a>
                </h1>
                <a href="#">
                    <img src="images/ikona-
wpisu.png" alt="" />
                </a>
                <p>
                    Lorem ipsum dolor sit amet
enim. Etiam ullamcorper. Suspendisse a
pellentesque
                    dui, non felis. Maecenas
malesuada elit lectus felis, malesuada ultricies.
                    Curabitur et ligula. Ut
molestie a, ultricies porta urna. Vestibulum
commodo
                    volutpat a, convallis ac,
laoreet enim. Phasellus fermentum in, dolor.
                    Pellentesque facilisis. Nulla
imperdiet sit amet magna.
                </p>
                <p>
                    Vestibulum dapibus, mauris nec
malesuada fames ac turpis velit, rhoncus eu,
                    luctus et interdum adipiscing
wisi. Aliquam erat ac ipsum. Integer aliquam
                    purus.
                </p>
                <a class="more-link"
href="#">Czytaj Dalej</a>
            </article>
        </section>


        <aside>
            <h2>SEKCJE</h2>
            <nav>
                <ul>
```

```
                    <li>
                        <a href="#">Bryły</a>
                    </li>
                    <li>
                        <a href="#">Funkcja</a>
                    </li>
                    <li>
                        <a href="#">Funkcja
Liniowa</a>
                    </li>
                    <li>
                        <a href="#">Geometria</a>
                    </li>
                    <li>
                        <a href="#">Konstrukcja</a>
                    </li>
                    <li>
                        <a href="#">Matury</a>
                    </li>
                    <li>
                        <a href="#">Testy
Gimnazjalne</a>
                    </li>
                </ul>
            </nav>
        </aside>
        <div class="clearfix"></div>
    </div>


    <footer class="web">
        &copy; 2015 Test.MatFiz24.pl
    </footer>
</body>
</html>
```

Plik style.css zawiera wszystkie kaskadowe arkusze stylów odpowiadające za wygląd szablonu w wyszukiwarce.

Stworzyłem Media Quariers, czyli zbiór osobnych stylów CSS skalujących serwis pod technologiami mobilnymi. Musisz tak zakodować style, aby szablon przeszedł pomyślną weryfikację w narzędziu Mobile Friendly.

Zaznaczam, że **kod pliku style.css ma charakter poglądowo – testowy**.

```
*
{
    margin: 0;
    padding: 0;
}

body
{
    font-family: Arial, Tahoma, sans-serif;
    font-size: 15px;
}
```

```
header,footer,nav,section,article,aside,hgroup
{
    display: block;
}

fieldset, img
{
    outline: none;
    border: none;
    margin: 0;
    padding: 0;
}

a
{
    color: #2c5c88;
    text-decoration: none;
}

a:hover
{
    color: #F39130;
}

p
{
    margin: 0 0 10px 0;
}

.clearfix
{
    clear: both;
}

.web
{
    width: 960px;
    margin: 0 auto;
}

    header .logo
    {
        float: left;
        padding: 10px 0;
    }

    header form
    {
        float: right;
        padding: 25px 0;
    }

        header form input[name=s]
        {
            height: 35px;
```

```
        width: 160px;
        padding-left: 5px;
        outline: none;
        border: 1px solid #58585A;
        border-radius: 7px 0 0 7px;
        color: #31323d;
    }

    header form input[type=submit]
    {
        height: 39px;
        outline: none;
        width: 80px;
        font-weight: 700;
        background: #58585A;
        color: #FFF;
        border-radius: 0 7px 7px 0;
        cursor: pointer;
        border: none;
    }

header nav
{
    background: url(images/tlo_menu.png) repeat-x;
    height: 46px;
}

    header nav ul,
    .web aside ul
    {
        list-style: none;
    }

        header nav ul li a
        {
            color: black;
            display: block;
            float: left;
            padding: 0 18px;
            line-height: 46px;
        }

        header nav ul li a:hover
        {
            color: #FFF;
            background: #F39130;
        }

    .web section
    {
        width: 700px;
        float: left;
        margin-top: 10px;
    }

        .web section article
```

```
    {
        border: 1px solid #e9e9e9;
        padding: 20px;
        border-radius: 20px;
        margin-bottom: 10px;
    }

        .web section article h1
        {
            margin: 0 0 10px 0;
            padding: 5px 10px;
            font-size: 25px;
        }

        a.more-link
        {
            display: block;
            background: #2c5c88;
            color: #FFF;
            width: 100px;
            padding: 10px 25px;
            border-radius: 5px;
            text-align: center;
        }

        a.more-link:hover
        {
              background: #F39130;
        }

.web aside
{
    width: 240px;
    margin-top: 10px;
    float: right;
}

    .web aside h2
    {
        background: #2a669e;
        color: white;
        padding: 10px;
        font-size: 15px;
    }

    .web aside nav
    {
        margin-bottom: 10px;
    }

        .web aside nav ul li:first-child a
        {
            border-top: 1px solid #e9e9e9;
        }

        .web aside nav ul li a
```

```
                {
                    border-left: 1px solid #e9e9e9;
                    border-right: 1px solid #e9e9e9;
                    border-bottom: 1px solid #e9e9e9;
                    display: block;
                    background: url(images/punktor_menu.png) 20px center no-repeat;
                    padding: 10px 35px;
                }

                .web aside nav ul li a:hover
                {
                    text-decoration: underline;
                }

    footer
    {
        color: #FFF;
        background: grey;
        line-height: 40px;
        text-align: center;
    }

@media all and (max-width: 959px)
{
    .web
    {
        width: 100%;
    }

        header .logo
        {
            float: none;
            text-align: center;
        }

        header form
        {
            display: none;
        }

        .web section
        {
            float: none;
            width: 100%;
        }

            .web section img
            {
                width: 100%;
                height: 100%;
            }

        .web aside
        {
            float: none;
            width: 100%;
```

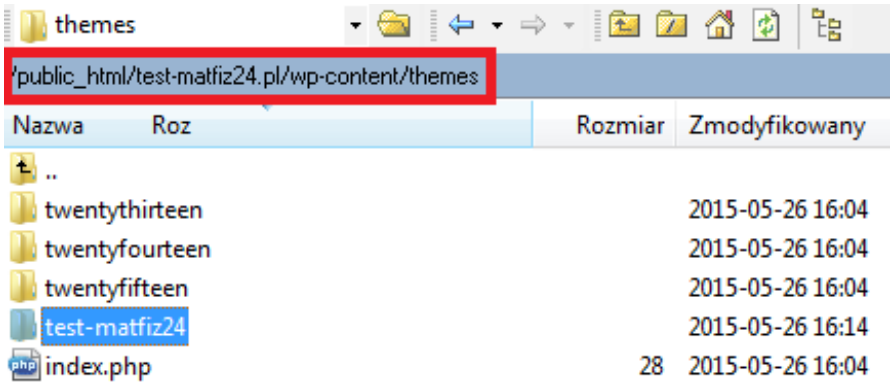
```
        }
}
```

Podłożenie metod WordPress w szablonie HTML5 i CSS3

Po zakodowaniu szablonu należy skonfigurować kod HTML i CSS w taki sposób, abyś mógł zarządzać treścią poprzez kokpit WordPress.

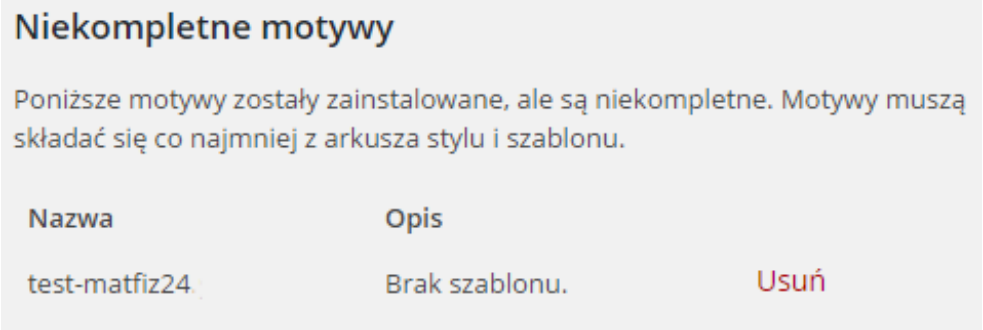
Krok 1 – wgraj szablon na serwer

Umieść zakodowany przed chwilą szablon test-matfiz24 na serwerze w **katalogu themes**. W tym katalogu znajdziesz wszystkie aktualne i domyślne motywy WordPressa: twentythirteen, twentyfourteen, twentyfifteen.



Krok 2 – spróbuj włączyć szablon poprzez kokpit WordPress

W kokpicie WordPress spróbuj włączyć szablon test-matfiz24, ale jest on nieaktywny.



Krok 3 – nadaj nazwę przyszłego szablonu

Na początku pliku style.css w pierwszej linijce obowiązkowo dopisz fragment kodu:

```
/* Theme Name: test-matfiz24
*/
```

Powyższy fragment kodu odpowiada za utworzenie nazwy szablonu w kokpicie WordPress.

Krok 4 – index.php

Plik index.html zamieniamy na plik index.php. Wystarczy zmienić rozszerzenie plików **.html na .php**

Dzięki zamianie rozszerzeń plików będziesz mógł swobodnie korzystać z języka PHP oraz wszystkich metod charakterystycznych dla technologii WordPress.

Krok 5 – włącz szablon

Odśwież kokpit WordPress i spróbuj ponownie włączyć szablon, który tym razem powinien być aktywny.

Krok 6 – odśwież stronę internetową

Po włączeniu szablonu i odświeżeniu strony widać, że cała strona się posypała jak domek z kart 🙂 Jest to zjawisko jak najbardziej naturalne, ponieważ straciliśmy poprawne ścieżki na serwerze do zaczytania stylów CSS oraz grafik. W praktyce każda strona internetowa w tym momencie wyświetla tylko zawartość kodu HTML.

Krok 7 – dodaj funkcje wp_head() i wp_footer()

Musisz zająć się podłożeniem odpowiednich metod WordPress w pliku index.php, dlatego rozpocznij od samego początku.

Szablon WordPress prawidłowo zakodowany powinien obsługiwać poprawnie wtyczki, które będziesz mógł instalować i konfigurować w kokpicie.

W tym celu w pliku index.php wystarczy umieścić dwie funkcje:

- wp_head() przed znacznikiem
- wp_footer() przed znacznikiem

W praktyce będzie to wyglądało następująco:

```
<?php wp_head(); ?
>
```

oraz

```
<?php wp_footer(); ?
>
```

Krok 8 – dynamiczny tytuł

Statyczny kod znacznika title:

```
<title>Test bloga matematycznego
Matfiz24.pl</title>
```

zamień na:

```
<title><?php wp_title(); ?> <?php bloginfo('name'); ?>
</title>
```

Przypomnę, że znacznik title ma wyjątkowe znaczenie dla wyszukiwarek internetowych i wszystkich speców od SEO. Powyższy kod spowoduje wyświetlenie tytułu z kokpitu WordPress. Do manipulacji tytułem i najważniejszymi meta tagami dla wyszukiwarek lepszym rozwiązaniem są instalacje pluginów (wtyczek) WordPress, ale o tym kiedy indziej 🙂

Krok 9 – zaktywuj arkusze stylów CSS3

W pliku index.php zaktywuj plik style.css. Wystarczy wykonać bardzo proste działanie!

Kod zamieszczony poniżej:

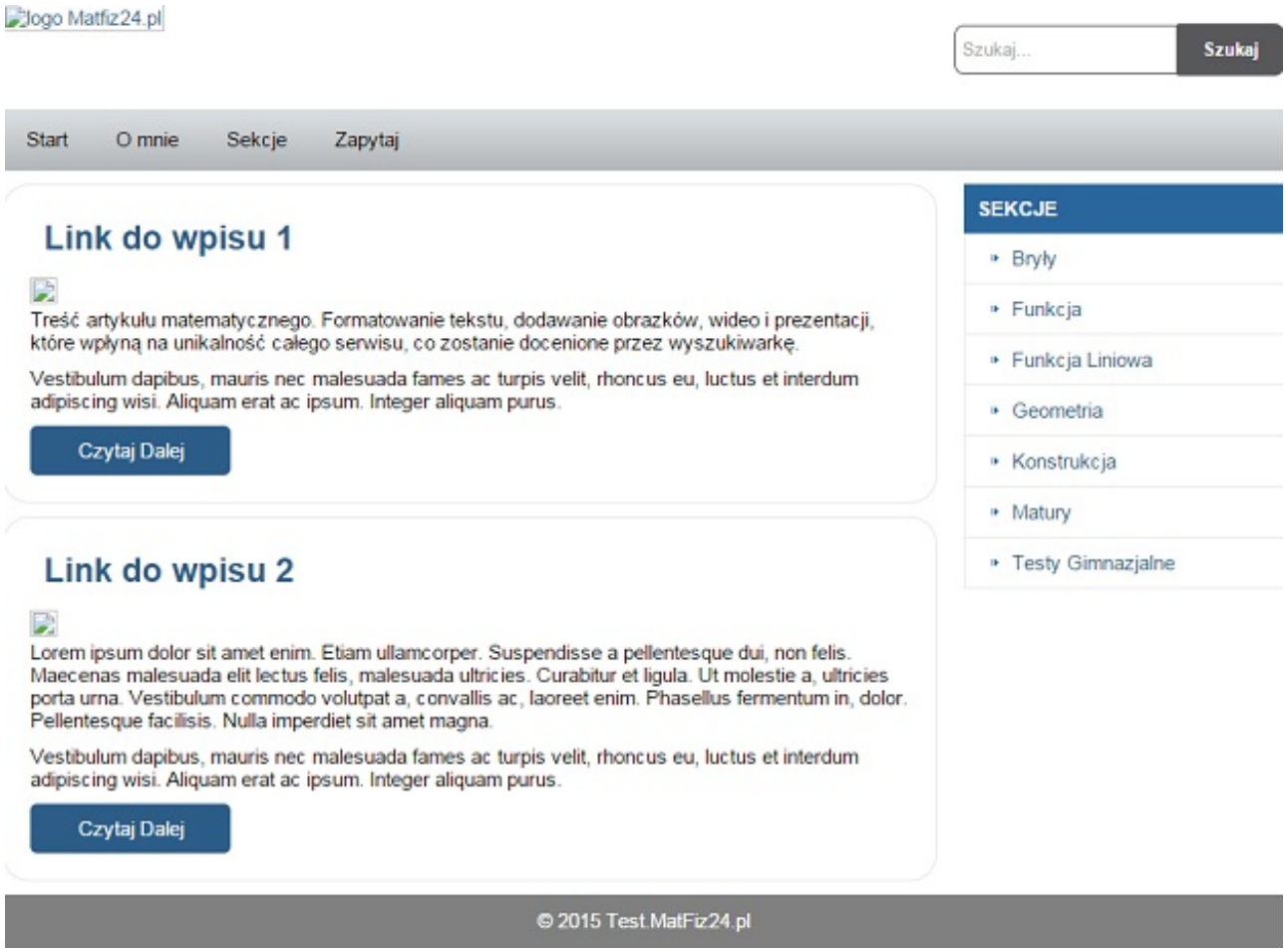
```
<link rel="stylesheet" href="style.css" type="text/css" />
```

zamień na:

```
<link rel="stylesheet" href="<?php bloginfo( 'stylesheet_url' ); ?>"
type="text/css" />
```

Krok 10 – odśwież szablon ze stylami CSS3

Odśwież stronę w przeglądarce. Zauważysz, że szablon poprawnie wczytuje plik style.css odpowiedzialny za warstwę prezentacji strony internetowej. Do pełnego wczytania szablonu strony w przeglądarce w pliku index.php musisz poprawnie wczytać obrazki z serwera, to jest: logo oraz ikony wpisu.



Krok 11 – wczytaj grafiki, nadając odpowiednią ścieżkę na serwerze

Musisz teraz poprawnie zaczytać z serwera wszystkie grafiki, które znajdują się w pliku index.php.

Poniższy fragment odpowiedzialny za wyświetlenie grafiki logo:

```
<div class="logo">
    <a href="/">
        <img src="images/logo.png" alt="logo Test.Matfiz24.pl" />
    </a>
</div>
```

---

zamień na:

```
<div class="logo">
    <a href="<?php bloginfo( 'home' ); ?>">
        <img src="<?=get_template_directory_uri(); ?>/images/logo.png" alt="logo
Test.Matfiz24.pl" />
    </a>
</div>
```

Krok 12 – wczytaj grafiki przyszłych ikon wpisu

Identyczną czynność wykonujemy dla grafik przyszłych ikon wpisu.

Poniższy fragment kodu odpowiedzialny za wyświetlenie przyszłych ikon wpisu:

```
<img src="images/ikona-wpisu.png" alt="" />
```

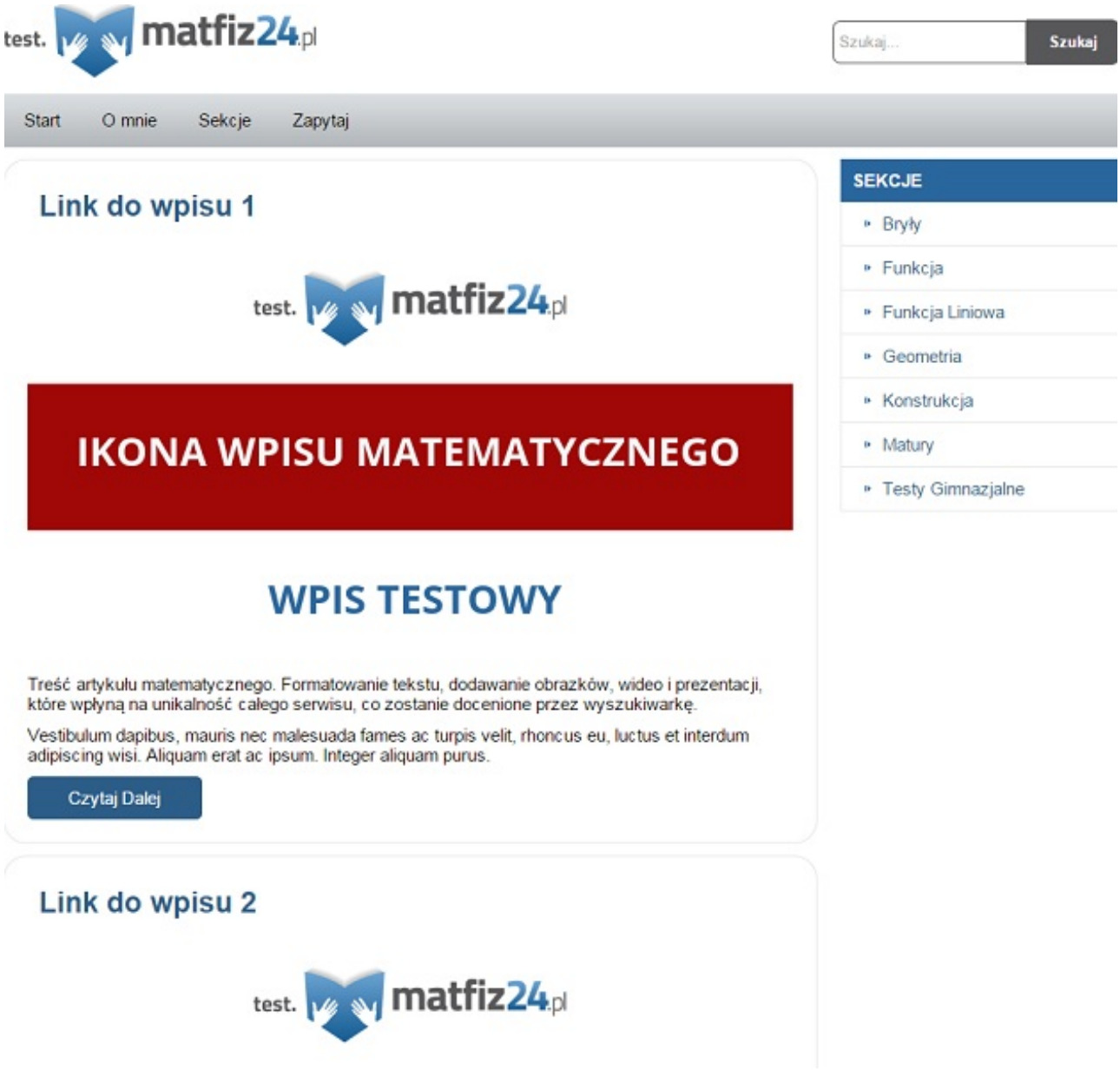
zamień na:

```
<img src="<?=get_template_directory_uri(); ?>/images/ikona-wpisu.png" alt=""
/>
```

Uwaga: W szablonie znajdują się aktualnie dwie przykładowe ikony wpisu. Pamiętaj, aby operację podmiany kodu wykonać w dwóch miejscach 🙂

Krok 13 – odśwież szablon ze stylami i grafikami

Odśwież stronę internetową w przeglądarce. Zauważ, że szablon od strony warstwy prezentacji jest już kompletny. Style CSS3 i grafiki zaczytywane są poprawnie z serwera.



Krok 14 – uruchom wyszukiwarkę treści na blogu

Po prawej stronie logo znajduje się wyszukiwarka, która nie działa. W związku z tym wyszukiwarkę WordPress należy odpowiednio uruchomić.

Poniższy fragment kodu:

```
<form action="/" method="get" accept-charset="UTF-8">
    <fieldset>
        <input name="s" type="text" placeholder="Szukaj..." value="" /><input
type="submit" value="Szukaj" />
    </fieldset>
</form>
```

---

zamień na:

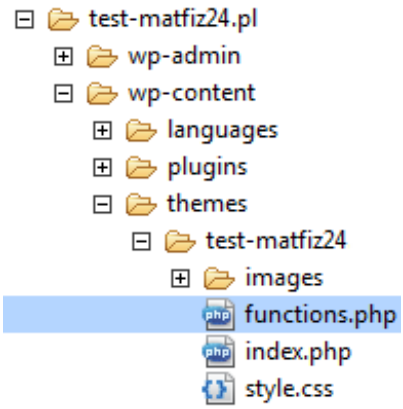
```
<form class="szukaj" action="<?php bloginfo('url'); ?>" method="get" accept-
charset="utf-8">
    <fieldset>
        <input name="s" type="text" placeholder="Szukaj..." value="<?php
the_search_query(); ?>"/><input type="submit" value="Szukaj" />
    </fieldset>
</form>
<div class="clearfix"></div>
```

Krok 15 – aktywacja i dodanie poziomego menu

Czas pokazać, jak zrobić poziome menu w szablonie? Będzie zawierać cztery podstrony: Start, O mnie, Sekcje, Zapytaj.

Stwórz w szablonie nowy, pusty plik o nazwie: functions.php. Platforma WordPress będzie wywoływać z tego pliku funkcje szablonu odpowiedzialne między innymi za tworzenie menu.

W pliku functions.php musisz stworzyć nowe poziome menu, dlatego do wnętrza pustego pliku wklej następujący kod:




```
<?php

function register_my_menus() {

  register_nav_menus(

    array(

      'poziome-menu-test' => __( 'Poziome menu testowe'
)

    )

  );

}

add_action( 'init', 'register_my_menus' );
?>
```

Powyższa funkcja jest odpowiedzialna za rejestrację w WordPress menu, które w kokpicie będzie wyświetlane pod nazwą: **Poziome menu testowe**.

Do szczęścia potrzeba nam jeszcze wywołać menu w pliku index.php podmieniając fragment kodu:

```
<nav>
    <ul>
        <li>
            <a href="/">Start</a>
        </li>
        <li>
            <a href="#">O mnie</a>
        </li>
        <li>
            <a href="#">Sekcje</a>
        </li>
        <li>
            <a href="#">Zapytaj</a>
        </li>
    </ul>
</nav>
```

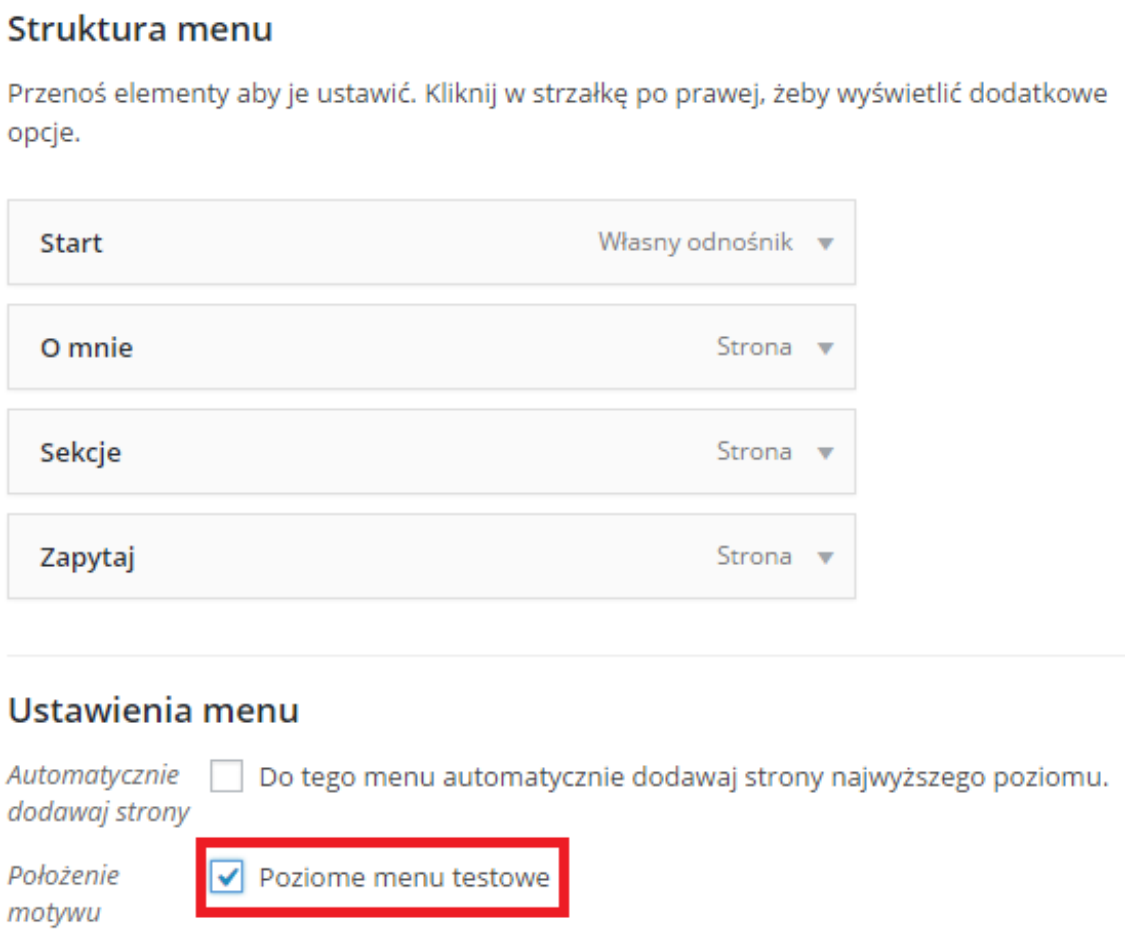
---

na kod:

```
<nav>
    <?php wp_nav_menu( array( 'theme_location' => 'poziome-menu-test', 'depth' =>
2)); ?>
</nav>
```

Zauważ, że po odświeżeniu szablonu – WordPress już domyślnie pokazał w menu przykładową podstronę. Jest to domyślna strona, która została utworzona **świeżo po instalacji WordPress**.

Teraz należy w kokpicie WordPress dodać nowe strony: Start, O mnie, Sekcje, Zapytaj, a także przypisać je do „Poziomego menu testowego” 🙂



Po odświeżeniu strony internetowej menu powinno zawierać wyjściowe podstrony, które są już dodane z kokpitu WordPress.

Krok 16 – wyświetlanie wpisów i stron

Wyświetlanie wpisów i stron z kokpitu jest możliwe przy pomocy Pętli WordPress, określanej fachowo **WordPress Loop.**

W pliku index.php aktualnie umieściłem dwa przykładowe wpisy z ikonami wpisu. Zrobiłem to celowo, aby pokazać Ci dynamikę tworzenia wpisów z kokpitu WordPressa. Będziesz mógł pisać wpisy do woli, bez kodowania HTML tak jak poniżej.

Tak na prawdę z dwóch artykułów znajdujących się w pliku index.php

```
<!--WPISY-->
<article>
    <h1>
        <a href="#">
            Link do wpisu 1
        </a>
    </h1>
    <a href="#">
        <img src="<?=get_template_directory_uri();?>/images/ikona-wpisu.png" alt=""
/>
    </a>
    <p>
        Treść artykułu matematycznego. Formatowanie tekstu, dodawanie obrazków,
        wideo i prezentacji, które wpłyną na unikalność całego serwisu, co
zostanie
        docenione przez wyszukiwarkę.
    </p>
    <p>
        Vestibulum dapibus, mauris nec malesuada fames ac turpis velit, rhoncus
eu,
        luctus et interdum adipiscing wisi. Aliquam erat ac ipsum. Integer aliquam
        purus.
    </p>
    <a class="more-link" href="#">Czytaj Dalej</a>
</article>

<!--WPISY-->
<article>
    <h1>
        <a href="#">
            Link do wpisu 2
        </a>
    </h1>
    <a href="#">
        <img src="<?=get_template_directory_uri();?>/images/ikona-wpisu.png" alt=""
/>
    </a>
    <p>
        Lorem ipsum dolor sit amet enim. Etiam ullamcorper. Suspendisse a
pellentesque
        dui, non felis. Maecenas malesuada elit lectus felis, malesuada ultricies.
        Curabitur et ligula. Ut molestie a, ultricies porta urna. Vestibulum
commodo
        volutpat a, convallis ac, laoreet enim. Phasellus fermentum in, dolor.
        Pellentesque facilisis. Nulla imperdiet sit amet magna.
    </p>
    <p>
        Vestibulum dapibus, mauris nec malesuada fames ac turpis velit, rhoncus
eu,
        luctus et interdum adipiscing wisi. Aliquam erat ac ipsum. Integer aliquam
        purus.
    </p>
    <a class="more-link" href="#">Czytaj Dalej</a>
</article>
```

pozostaw tylko jeden wpis, tak jak poniżej:

```
<!--WPISY-->
<article>
    <h1>
        <a href="#">
            Link do wpisu 2
        </a>
    </h1>
    <a href="#">
        <img src="<?=get_template_directory_uri(); ?>/images/ikona-wpisu.png"
alt="" />
    </a>
    <p>
        Lorem ipsum dolor sit amet enim. Etiam ullamcorper. Suspendisse a
pellentesque
        dui, non felis. Maecenas malesuada elit lectus felis, malesuada ultricies.
        Curabitur et ligula. Ut molestie a, ultricies porta urna. Vestibulum
commodo
        volutpat a, convallis ac, laoreet enim. Phasellus fermentum in, dolor.
        Pellentesque facilisis. Nulla imperdiet sit amet magna.
    </p>
    <p>
        Vestibulum dapibus, mauris nec malesuada fames ac turpis velit, rhoncus
eu,
        luctus et interdum adipiscing wisi. Aliquam erat ac ipsum. Integer aliquam
        purus.
    </p>
    <a class="more-link" href="#">Czytaj Dalej</a>
</article>
```

Teraz należy uruchomić Pętlę WordPress zastępując powyższy kod na poniższy 🙂

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<article>
    <h1>
        <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
    </h1>
    <a href="#">
        <img src="<?=get_template_directory_uri(); ?>/images/ikona-wpisu.png"
alt="" />
    </a>
    <?php the_content('czytaj dalej'); ?>
</article>

<?php endwhile; else: ?>

<p><?php _e('Sorry, no posts matched your criteria.'); ?></p>

<?php endif; ?>
```

Po odświeżeniu strony głównej został wyświetlony domyślny wpis **Witaj, świecie!** Jeżeli masz więcej wpisów

napisanych w kokpicie, one też zostaną wyświetlone.

Tak naprawdę tutaj znajdują się najważniejsze funkcje szablonu:

- the_permalink() – funkcja pobiera adres URL wpisu
- the_title() – funkcja pobiera tytuł wpisu
- get_template_directory_uri() – funkcja pobiera właściwą ścieżkę z serwera, np: do zaczytania grafik szablonu
- the_content() – funkcja pobiera treść wpisu – zobacz jak alternatywnie działa funkcja the_excerpt()

Krok 17 – tworzenie ikony wpisu

W szablonie utworzę ikony wpisu o wymiarach 658px szerokości i 370px wysokości.

Umieść w pliku functions.php trzy linie kodu.

```
add_theme_support( 'post-thumbnails' );
set_post_thumbnail_size( 658, 370 );
add_image_size( 'post-icon', 658, 370, true
);
```

Pierwsza funkcja inicjuje miniaturki w WordPress.

Druga metoda ustawia domyślne wartości miniaturek.

Trzecia funkcja odwołuje się do nazwy ustawienia ikony wpisu **post-icon** o podanych powyżej wymiarach.

Naszą ikonę wpisu musisz uruchomić w pliku index.php, a dokładnie w Pętli WordPress.

Odszukaj fragment kodu:

```
<a href="#">
    <img src="<?=get_template_directory_uri(); ?>/images/ikona-wpisu.png" alt=""
/>
</a>
```

i zamień na:

```
<a href="<?php the_permalink(); ?>">
    <?php if ( has_post_thumbnail() ) { the_post_thumbnail( 'post-icon') ; }
?>
</a>
```

W tym momencie podczas dodawania wpisów w kokpicie masz możliwość dodawania obrazka wyróżniającego – ikony wpisu.

Uwaga 1!

Jeżeli wgrasz ikonę wpisu mniejszą od zadeklarowanych wymiarów miniaturki, wówczas **ikona wpisu nie zostanie utworzona.**

Uwaga 2!

Jeżeli wgrasz ikonę wpisu równą lub większą od zadeklarowanych wymiarów miniaturki, wówczas **ikona wpisu zostanie utworzona**.

Uwaga 3!

Jeżeli będziesz modyfikował wielkość ikon wpisu w pliku functions.php musisz zainstalować plugin Regenerate Thumbnails i zregenerować wszystkie grafiki. Aktualnie jest to niezbędna operacja do wygenerowania nowych miniaturek wpisu.

Krok 18 – tworzenie pionowego menu

Sekcje matematyczne znajdujące się w prawej kolumnie współtworzą menu pionowe szablonu, które należy obsłużyć przez CMS.

W tym celu rozwiniemy funkcję w pliku functions.php, którą utworzyliśmy w celu **zarejestrowania Poziomego menu testowego**.

Do poniższego fragmentu kodu:

```
function register_my_menus() {
  register_nav_menus(
    array(
      'poziome-menu-test' => __( 'Poziome menu testowe'
)
    )
  );
}
```

musisz dodać przecinek i dokładnie jedną linijkę kodu tak jak poniżej:

```
function register_my_menus() {

  register_nav_menus(

    array(

      'poziome-menu-test' => __( 'Poziome menu testowe'
),
      'pionowe-menu-test' => __( 'Pionowe menu testowe' )
    )

  );

}
```

W tym momencie zarejestrowałeś w pliku functions.php nowe menu, które w kokpicie WordPressa będzie się wyświetlane jako: **Pionowe menu testowe**.
Nowo powstałe menu pionowe należy wywołać w pliku index.php podmieniając fragment kodu:

```
<nav>
    <ul>
        <li>
            <a href="#">Bryły</a>
        </li>
        <li>
            <a href="#">Funkcja</a>
        </li>
        <li>
            <a href="#">Funkcja Liniowa</a>
        </li>
        <li>
            <a href="#">Geometria</a>
        </li>
        <li>
            <a href="#">Konstrukcja</a>
        </li>
        <li>
            <a href="#">Matury</a>
        </li>
        <li>
            <a href="#">Testy
Gimnazjalne</a>
        </li>
    </ul>
</nav>
```

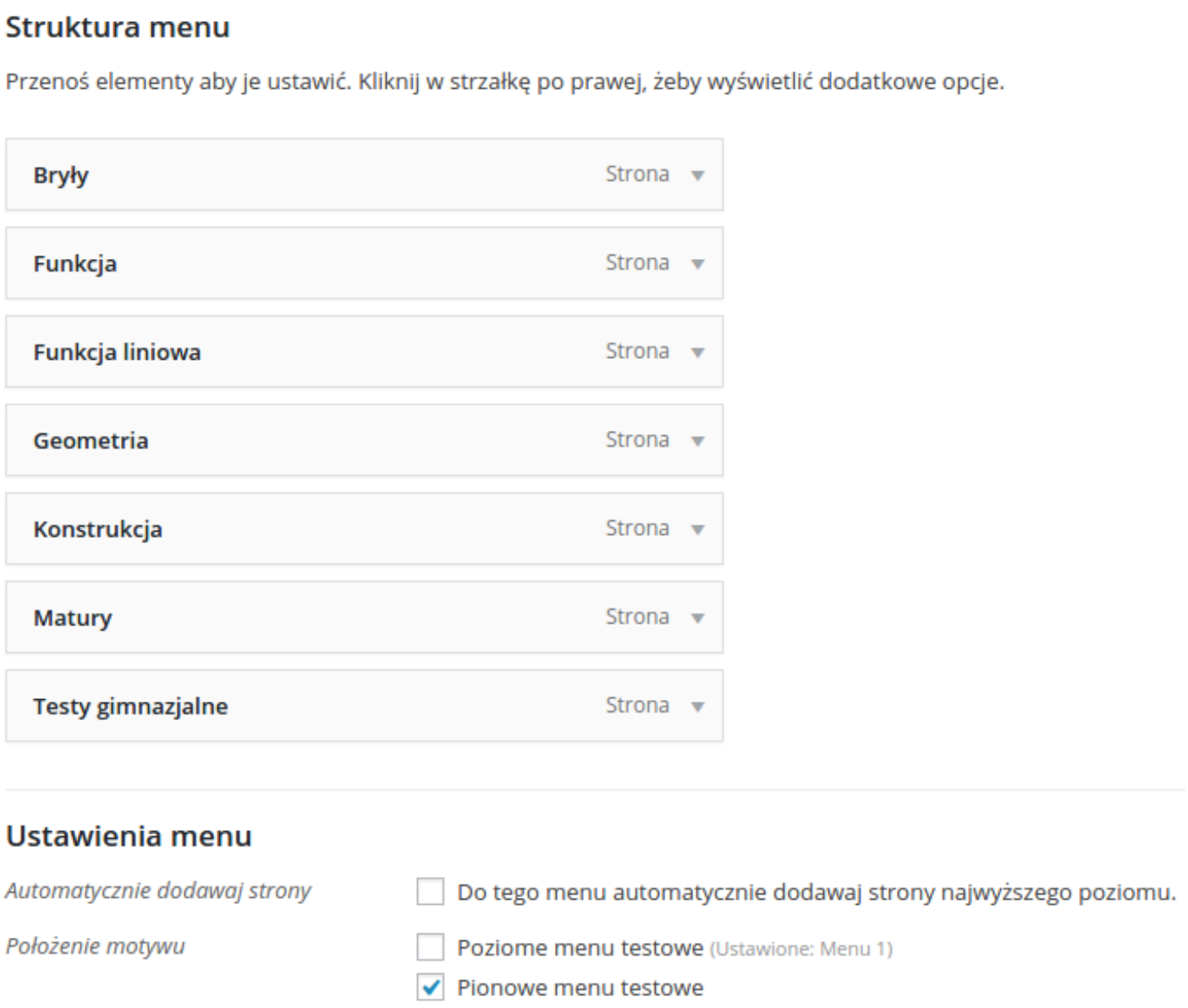
na kod:

```
<nav>
    <?php wp_nav_menu( array( 'theme_location' => 'pionowe-menu-test', 'depth' =>
2)); ?>
</nav>
```

Teraz nie pozostaje nic innego jak stworzyć nowe podstrony w kokpicie WordPress:

- Bryły
- Funkcja
- Funkcja Liniowa
- Geometria
- Konstrukcja
- Matury
- Testy Gimnazjalne

W kokpicie WordPress tworzymy nowe menu -> Menu 2 i przypisujemy tam nowo powstałe podstrony działów matematycznych. Pamiętaj, aby menu przypisać tym razem do: **Pionowego menu testowego**.



Odśwież stronę internetową. Zauważ, że otrzymałeś dokładnie wyjściowy szablon HTML5 i CSS3, który jest w pełni obsługiwany przez kokpit WordPress.

Krok 19 – rozbudowa szablonu o widgety

W czasie blogowania zawsze dochodzi do sytuacji, w którym aktualnie użytkowany szablon wymaga rozbudowy.

Przypuśćmy, że nieoczekiwanie zgłosił się do Ciebie potencjalny Klient i chciałby umieścić odpłatnie **baner o wymiarach 240px x 240px w prawej kolumnie pod sekcjami matematycznymi**.

W tym celu uruchom funkcjonalność widgety, które ponadto udostępniają szereg innych ciekawych modułów z kokpitu całkowicie za darmo:

- Najnowsze wpisy
- Wyszukiwarka
- Własne menu
- Kategorie
- Sam potestuj 🙂

W pliku functions.php wklej następujący kod, który zaktywuje widgety w kokpicie WordPress.

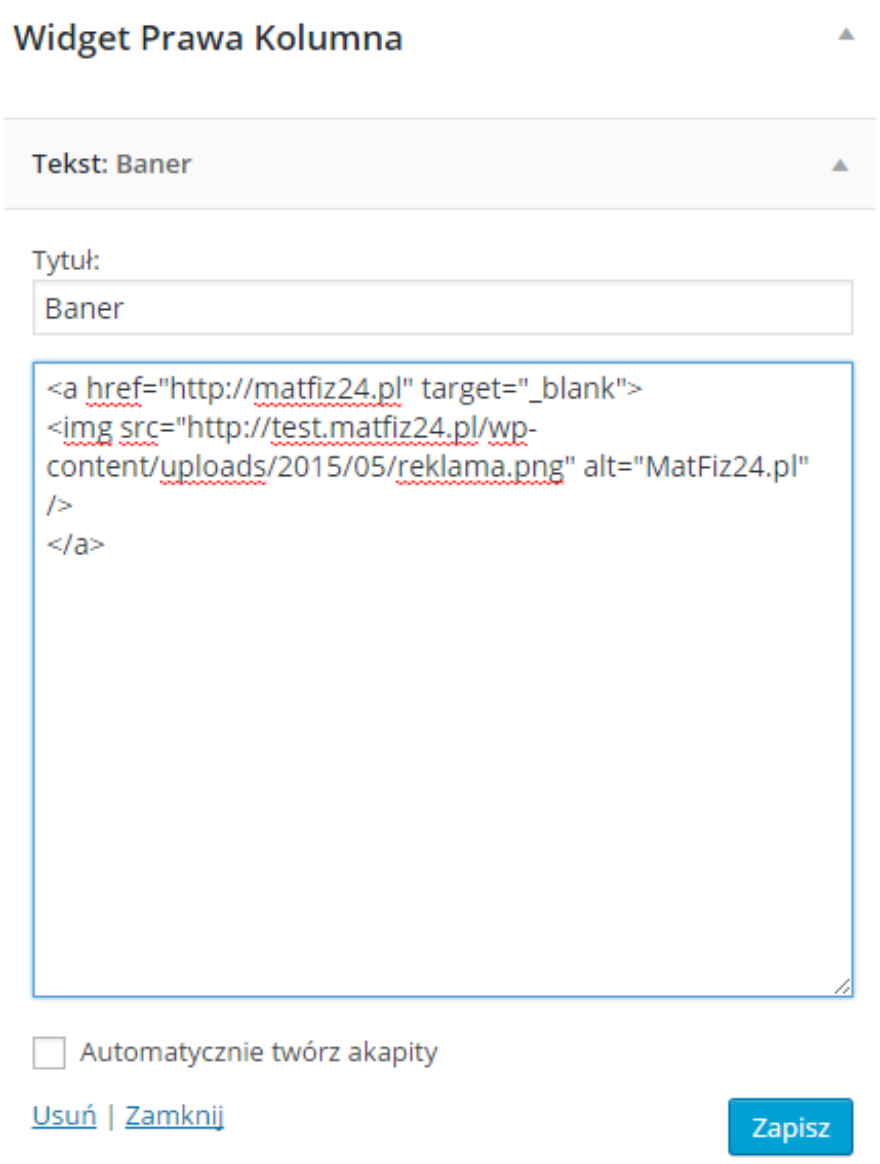
```
if ( function_exists ('register_sidebar')) {
   register_sidebar(array(
           'name' => 'Widget Prawa Kolumna',
            'before_widget' => '<div class="new-widget
%2$s">',
            'after_widget' => '</div>',
            'before_title' => '<h2>',
            'after_title' => '</h2>',
        ));
}
```

Widget należy jeszcze wywołać w pliku index.php. Możesz go wywołać w dowolnym miejscu szablonu. Ja swój widget uruchomię przed </aside>.

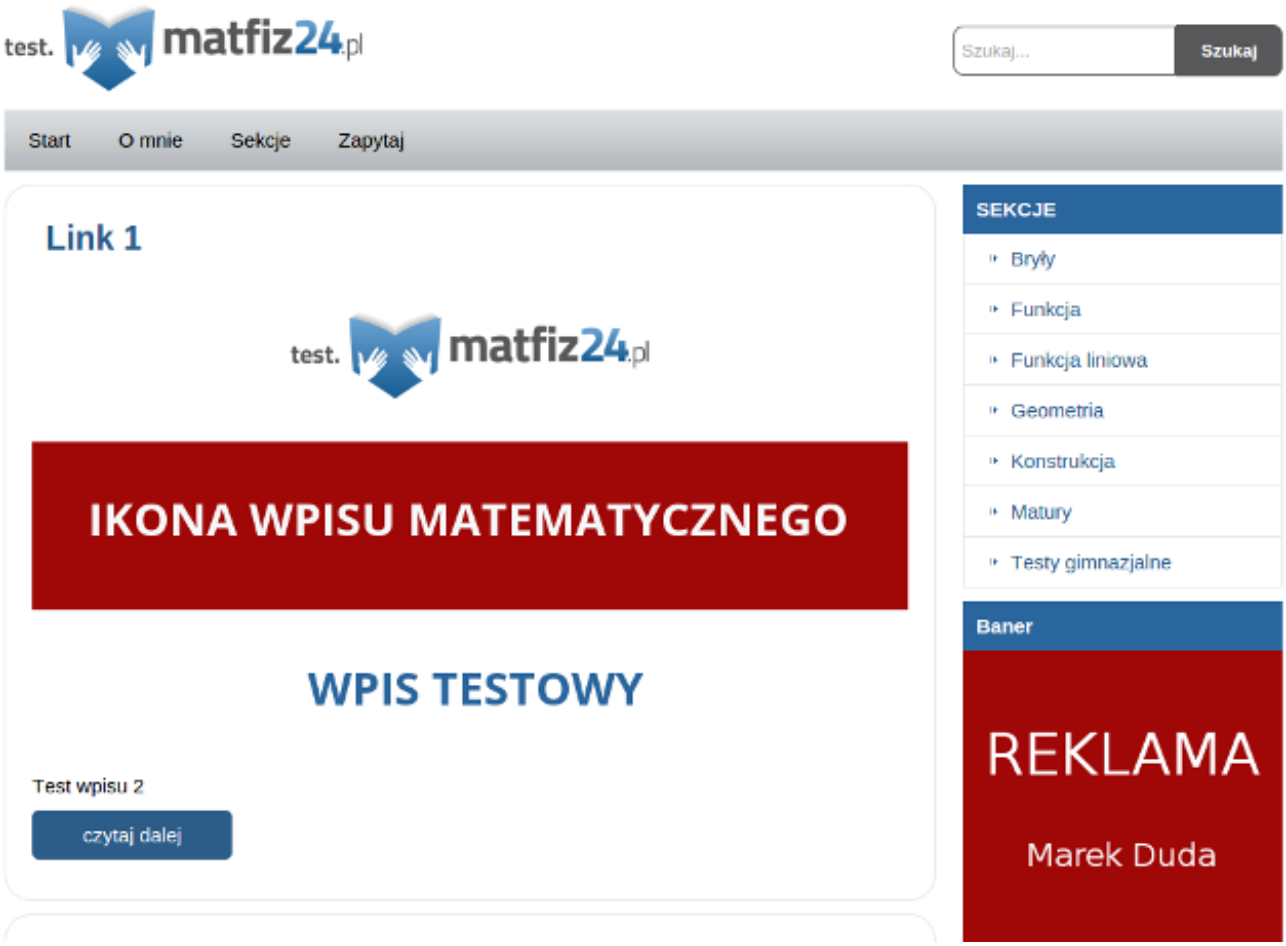
```
<?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar() ) : ?
>

<?php endif; ?>
```

W tym momencie po odświeżeniu kokpitu możesz już dodawać nowe widgety z zakładki Wygląd.

Moja reklama została zaimplementowana w widgecie: Tekst



Tak wygląda teraz szablon – test.matfiz24 po uruchomieniu widgetu Tekst z reklamą.



Krok 20 – demontaż pliku index.php na mniejsze pliki

Wykonując powyższe kroki implementowania szablonu możesz przekonać się, że szablon test-matfiz24 działa poprawnie pod najpopularniejszymi przeglądarkami: Chrome, Mozilla, Opera, czy IE. Sprawdź!

Platforma WordPress udostępnia logiczną hierarchię plików, która pozwala rozwinąć szablon zarówno funkcjonalnie, jak i wizualnie od strony CSS.

W tym momencie rozdzielisz razem ze mną plik index.php na mniejsze pliki.

Tam, gdzie zawartość kodu zostanie przeniesiona do innego pliku, w index.php będziesz musiał zaczytać zawartość tego pliku przy pomocy konkretnych funkcji.

Zobacz, jak to zrobić w praktyce?

Utwórz w szablonie nowe pliki:

Plik searchform.php

```
<!--WYSZUKIWARKA-->
<form class="szukaj" action="<?php bloginfo('url'); ?>" method="get" accept-
charset="utf-8">
    <fieldset>
        <input name="s" type="text" placeholder="Szukaj..." value="<?php
the_search_query(); ?>"/><input type="submit" value="Szukaj"/>
    </fieldset>
</form>
<div class="clearfix"></div>
```

Plik header.php

```
<!DOCTYPE html>
<html>
<head>
    <title><?php wp_title(); ?> <?php bloginfo('name'); ?></title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="<?php bloginfo( 'stylesheet_url' ); ?>"
type="text/css" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1"/>
    <script>
        for(var e,l='footer header nav article section aside figure'.split('
');e=l.pop();document.createElement(e));
    </script>
    <?php wp_head(); ?>
</head>
<body>
    <header class="web">
        <!--LOGO-->
        <div class="logo">
            <a href="<?php bloginfo( 'home' ); ?>">
                <img src="<?=get_template_directory_uri(); ?>/images/logo.png"
alt="logo Test.Matfiz24.pl" />
            </a>
        </div>

        <?php get_search_form(); ?>

        <!--MENU POZIOME-->
        <nav>
            <?php wp_nav_menu( array( 'theme_location' => 'poziome-menu-test',
'depth' => 2)); ?>
        </nav>
    </header>
    <div class="clearfix"></div>
```

Plik sidebar.php

```
<!--SIDEBAR-->
<aside>
    <h2>SEKCJE</h2>
    <nav>
        <?php wp_nav_menu( array( 'theme_location' => 'pionowe-menu-test', 'depth'
=> 2)); ?>
    </nav>
    <?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar() ) : ?>

    <?php endif; ?>

</aside>
<div class="clearfix"></div>
```

Plik footer.php

```
<!--FOOTER-->
    <footer class="web">
        &copy; 2015
Test.MatFiz24.pl
    </footer>
    <?php wp_footer(); ?>
</body>
</html>
```

Plik index.php

```
<?php get_header(); ?>
    <div class="web">
        <section>
            <!--WPISY-->
            <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

            <article>
                <h1>
                    <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
                </h1>
                <a href="<?php the_permalink(); ?>">
                    <?php if ( has_post_thumbnail() ) { the_post_thumbnail( 'post-
icon' ); } ?>
                </a>
                <?php the_content('czytaj dalej'); ?>
            </article>

            <?php endwhile; else: ?>

            <p><?php _e('Sorry, no posts matched your criteria.'); ?></p>

            <?php endif; ?>
        </section>

        <?php get_sidebar(); ?>
    </div>

<?php get_footer(); ?>
```

W pliku header.php wykorzystałem funkcję get_search_form() do pobrania pliku searchform.php, gdzie umieściłem wyszukiwarkę.

W pliku index.php wykorzystałem trzy funkcje:

- get_header() – do pobrania pliku header.php, gdzie wyświetlone jest logo, wyszukiwarka i poziome menu
- get_sidebar() – do pobrania pliku sidebar.php, gdzie wyświetlone jest menu pionowe i baner reklamowy
- get_footer() – do pobrania pliku footer.php, gdzie wyświetlona jest stopka serwisu

Krok 21 – test mobile friendly

W narzędziu https://www.google.com/webmasters/tools/mobile-friendly/ sprawdź, czy szablon jest przyjazny technologiom mobilnym?

Okazuje się, że zakodowany szablon jest przyjazny pod urządzenia mobilne.

W tym momencie nie pozostaje Ci nic innego, jak spakować szablon w formacie .zip.

Szablon test-matfiz24 możesz pobrać tutaj i modyfikować kod bez ograniczeń 🙂

Utworzony szablon możesz zainstalować na każdym blogu WordPress z zakładki Wygląd -> Motywy.

Struktura plików szablonu WordPress

Do pisania rozbudowanych szablonów WordPress niezbędna jest znajomość innych plików WordPress.

Poniżej przedstawiam po dwa najpopularniejsze pliki odpowiedzialne za wyświetlanie różnych sekcji w serwisie.

Pliki strony głównej:

- home.php – plik dedykowany tylko dla strony głównej. Możesz w nim stworzyć osobną strukturę i wygląd strony głównej.
- index.php – plik praktycznie niezbędny w szablonie, tworzymy listę wpisów wyświetlanych na stronie głównej.

Pliki nagłówka:

- header-{name}.php – plik zawierający dedykowany nagłówek bloga. Możesz go uruchomić w szablonie przy pomocy funkcji.
- header.php – plik zawierający główny nagłówek bloga. Znajdują się tutaj wszystkie elementy, które znajdują się w górnej części serwisu: meta tagi strony, pliki .css .js, kod śledzenia Google Analytics, a także menu poziome. Możesz go uruchomić w szablonie przy pomocy funkcji.

Pliki stopki:

- footer-{name}.php – plik zawierający dedykowaną stopkę serwisu. Możesz go uruchomić przy pomocy funkcji.
- footer.php – wyświetlenie stopki głównej bloga. Możesz go uruchomić przy pomocy funkcji.

Pliki wyświetlające pojedynczy wpis:

- single.php – plik odpowiedzialny za wyświetlanie pojedynczego wpisu na blogu.
- index.php

Pliki strony statycznej:

- page.php – plik wyświetlający pojedynczą stronę.
- index.php

Pliki kategorii:

- category.php – plik wyświetlający pojedynczą kategorię.
- index.php

Pliki tagów:

- tag.php – plik wyświetlający pojedynczy tag
- index.php

Pliki autora:

- author.php – plik wyświetlający pojedynczego autora
- index.php

Plik komentarzy:

- comments.php – plik obsługujący i wyświetlający formularz do komentarzy. Możesz go uruchomić przy pomocy funkcji.

Pliki wyszukiwarki:

- search.php – plik zawierający sekcję wyświetlania wyników wyszukiwania na stronie
- searchform.php – plik obsługujący i wyświetlający wyszukiwarkę WordPress. Możesz go uruchomić przy pomocy funkcji.

Pliki błędu 404:

- 404.php – plik uruchamiany w WordPress w wyniku wystąpienia błędu 404.

- index.php

Pliki kolumny bocznej:

- sidebar-{name}.php – dedykowany pasek boczny. Możesz go uruchomić przy pomocy funkcji.
- sidebar.php – plik odpowiedzialny za możliwość uruchomienia widgetów na blogu. Możesz go uruchomić przy pomocy funkcji.

Plik funkcji:

- functions.php – w tym pliku należy umieszczać zbiór wszystkich funkcji napisanych przez twórcę szablonu.

Plik stylów css:

- style.css – plik musi się znajdować w katalogu głównym szablonu (na tym samym poziomie, co pliki Twojego szablonu). Plik ten jest miejscem tworzenia oryginalności szablonu pod względem wizualnym. Niezbędna jest podstawowa znajomość kaskadowego arkusza stylów CSS.

Podsumowanie:

Nie musisz się przejmować ilością plików tworzących rozbudowany szablon WordPressa. W rzeczywistości do zrobienia funkcjonalnego motywu wystarczy znajomość zaledwie kilku plików WordPressa oraz kilku funkcji, co udowodniłem Ci podczas kodowania szablonu test-matfiz24.

Zbuduj własny szablon WordPress RWD i skończ z problemami!

Po zainstalowaniu mojego matematycznego bloga MatFiz24.pl na WordPress.org popełniłem wiele błędów odnośnie szablonów, ponieważ:

1. Nie zaplanowałem wyglądu i funkcjonalności mojej skórki – w efekcie straciłem wiele czasu na testowanie darmowych i płatnych motywów.
2. Nie miałem zielonego pojęcia o optymalizacji i pozycjonowaniu stron internetowych. Okazało się, że struktura szablonu przyszłej witryny ma kluczowe znaczenie do zdobywania wysokich pozycji w wyszukiwarce! Więcej dowiesz się w osobnym kursie tworzenia zarabiającego szablonu WordPress.
3. Nie wiedziałem, jak tworzyć szablony WordPress?

Zalety tworzenia własnych motywów do WordPressa

Jeżeli potrafisz stworzyć szablon WordPress od zera jesteś wielki, ponieważ:

1. Możesz zrobić własnego bloga bez żadnych ograniczeń funkcjonalnych i wizualnych.
2. Posiadasz niezbędną wiedzę do szybkiej edycji motywu i rozwijania bloga.
3. Jesteś profesjonalistą i wiesz jak pomagać innym ludziom w tworzeniu szablonów, udzielając się na forach tematycznych, budując własną markę.
4. Możesz pracować jako freelancer, tworząc autorskie szablony WordPressa na zlecenie.

Tworzenie autorskich szablonów WordPress na zlecenie to świetny pomysł na własny biznes, ponieważ technologia WordPress jest bardzo na czasie.

Warunkami koniecznymi do stworzenia responsywnego szablonu WordPress, przyjaznego wyszukiwarkom są:

- Zbudowanie szablonu strony w HTML i CSS
- Poznanie metod PHP i WordPress – wystarczy zaledwie kilka metod do zbudowania własnego motywu
- Wiedza na temat hierarchii plików w technologii WordPress
- Niezbędne kwalifikacje z zakresu SEO
- Odkrycie niezbędnych narzędzi do zbudowania i testowania szablonu

Jeżeli poznałeś podstawy budowania szablonów, zapraszam Cię na wyższy poziom tworzenia szablonów WordPress.