Zasady oceniania projektu zaliczeniowego

Technologie internetowe

Responsywny szablon strony internetowej w HTML i CSS

Ocena 3,0	Wykonany responsywny szablon strony charakteryzuje się zastosowaniem stosunkowo prostego układu treści, z niewielkim wykorzystaniem multimediów, ikon. Nie są wykorzystywane skomplikowane układy treści, np. w postaci grupy zdjęć ułożonej do siebie relatywnie, czy tekstu wkomponowanego w ilustrujące go zdjęcia. Strona nie posiada żadnych animowanych treści. Na stronie nie pojawiają się elementy typu wyskakujące okienka (popupy, alerty). Na stronie pojawia się nieskomplikowany do wykonania formularz, w tym jeżeli chodzi o układ treści. Nie są stosowane różnego rodzaju ozdobne elementy. Motyw kolorystyczny wykorzystywany przez szablon jest dość ubogi. Dopuszczalne są drobne pomyłki w realizacji responsywności (tj. zawartość jest np. lekko ucinana, następuje nieestetyczne zawijanie elementów, układ treści np. rozkłada się niesymetrycznie), natomiast interfejs nie zatraca w wyniku tego użyteczności (np. nie znikają przyciski, linki, nie znikają zupełnie obrazki, czy tekst). Szablon ma tendencję do wykorzystywania powtarzalnego rozwiązania (np. powielony wielokrotnie jeden układ gridowy). Ogólnie szablon korzysta z ograniczonej liczby typów znaczników HTML, unikając większej liczby problemów do rozwiązania.
Ocena 4,0	Wykonany responsywny szablon strony charakteryzuje się zastosowaniem średnio złożonego układu treści, z umiarkowanym wykorzystaniem multimediów, ikon. Co najmniej raz pojawia się w szablonie skomplikowany

	układ treści, np. w postaci grupy zdjęć ułożonej do siebie relatywnie, czy tekstu wkomponowanego w ilustrujące go zdjęcia. Strona posiada kilka animacji. Na stronie pojawiają się elementy typu wyskakujące okienka (popupy, alerty). Na stronie pojawia się jeden skomplikowany do wykonania formularz. Nie są stosowane różnego rodzaju ozdobne elementy. Motyw kolorystyczny wykorzystywany przez szablon składa się z przemyślanej palety kolorów. Dopuszczalne są drobne pomyłki w realizacji responsywności (tj. zawartość jest np. lekko ucinana, następuje nieestetyczne zawijanie elementów, układ treści np. rozkłada się niesymetrycznie), natomiast interfejs nie zatraca w wyniku tego użyteczności (np. nie znikają przyciski, linki, nie znikają zupełnie obrazki, czy tekst). Szablon rozwiązuje kilka różnych problemów (np. posiada galerię zdjęć; responsywną szeroką tabelę; kontener, posiadający własny scroll; zwijalną/rozwijalną zakładkę; zakładki z treścią typu tabs). Szablon wykorzystuje większą niż na ocenę 3,0 liczbę typów znaczników HTML i w związku z powyższym rozwiązuje więcej niż jeden problem. Koniecznie w szablonie wykorzystywany jest slider/carousel. Szablon posiada mechanizm, który powoduje, że pomimo wystąpienia niewielkiej zawartości (potencjalnie) na danej podstronie, stopka wyświetla się u dołu viewportu (a nie np. w połowie ekranu).
Ocena 5,0	Wykonany responsywny szablon strony charakteryzuje się zastosowaniem złożonego układu treści, z dużym wykorzystaniem multimediów, ikon. Co najmniej 3 razy pojawia się w szablonie skomplikowany układ treści, np. w postaci grupy zdjęć ułożonej do siebie relatywnie, czy tekstu wkomponowanego w ilustrujące go zdjęcia. Strona posiada kilka animacji, w tym przynajmniej jedną złożoną. Na stronie pojawiają się elementy typu wyskakujące okienka (popupy, alerty). Na stronie pojawia się jeden skomplikowany do wykonania formularz. Na stronie znajduje się jakiś element ozdobny. Motyw kolorystyczny wykorzystywany przez szablon składa się z przemyślanej palety kolorów. Nie są dopuszczalne co do zasady żadne pomyłki w realizacji responsywności szablonu. Szablon rozwiązuje kilka różnych problemów (np. posiada galerię zdjęć; responsywną szeroką tabelę; kontener, posiadający własny scroll; zwijalną/rozwijalną zakładkę; zakładki z treścią typu

	tabs). Szablon wykorzystuje dużą liczbę znaczników HTML i w związku z powyższym rozwiązuje wiele różnych problemów. Koniecznie w szablonie wykorzystywany jest slider/carousel. Szablon posiada mechanizm, który np. utrzymuje pasek nawigacyjny zawsze widoczny na górze strony, lub inne tego typu rozwiązanie. Szablon posiada mechanizm, który powoduje, że pomimo wystąpienia niewielkiej zawartości (potencjalnie) na danej podstronie, stopka wyświetla się u dołu viewportu (a nie np. w połowie ekranu).

Podczas obrony projektu wymaga się opisania stworzonych w projekcie rozwiązań, przykładowo tworząc efekt paralaksy na stronie należy umieć wytłumaczyć w jaki sposób został on uzyskany (jakie właściwości zostały wykorzystane).

Aplikacja w języku JavaScript

Ocena 3,0	• Aplikacja wykorzystuje podstawowe konstrukcje języka JavaScript, takie jak pętle, funkcje, sterowanie przepływem programu, tablice. • Aplikacja wykorzystuje co najmniej jeden raz DOM, w tym do manipulacji zawartością dokumentu. • Aplikacja wykorzystuje proste wywołanie asynchroniczne w JavaScript. • Aplikacja co najmniej raz wykorzystuje JSON. • Aplikacja potrafi odczytać wartość elementu formularza i w jakiś sposób ją przetworzyć (np. wyświetlić w output wartość obliczoną na podstawie wartości pobranej z input).
Ocena 4,0	• Aplikacja wykorzystuje podstawowe konstrukcje języka JavaScript, takie jak pętle, funkcje, sterowanie przepływem programu, tablice. • Aplikacja wykorzystuje biernie (tj. wyłącznie wykorzystuje zdefiniowane zewnętrznie, np. w bibliotece, w tym wbudowane w język) bardziej złożone konstrukcje języka JavaScript, takie jak: klasy, moduły. • Aplikacja wykorzystuje DOM kilkukrotnie, w tym do dynamicznego utworzenia fragmentu dokumentu. • Aplikacja modyfikuje za pomocą języka JavaScript właściwości zdefiniowane w CSS.

	• Aplikacja wykorzystuje API zdarzeń, w więcej niż jednym kontekście. • Aplikacja wykorzystuje co najmniej raz Promise. • Aplikacja co najmniej raz wykorzystuje JSON. • Aplikacja wykorzystuje localStorage. • Aplikacja potrafi odczytać wartość elementu formularza, zwalidować ją i przetworzyć.
Ocena 5,0	• Aplikacja wykorzystuje podstawowe konstrukcje języka JavaScript, takie jak pętle, funkcje, sterowanie przepływem programu, tablice. • Aplikacja wykorzystuje zaawansowane konstrukcje języka JavaScript, takie jak klasy, moduły, w tym samodzielnie napisane. • Aplikacja wykorzystuje DOM wielokrotnie, w tym do dynamicznego utworzenia fragmentu dokumentu. • Aplikacja modyfikuje za pomocą języka JavaScript właściwości zdefiniowane w CSS, w tym również programowo tworzone są animacje, niemożliwe do zrealizowania w CSS. • Aplikacja wykorzystuje API zdarzeń, w tym: nadpisujące domyślną akcję formularza, wykorzystuje mysz i klawiaturę. • Aplikacja wykorzystuje localStorage oraz stosuje ciasteczka do przechowywania globalnych, trwałych ustawień. • Aplikacja co najmniej raz wykorzystuje JSON. • Aplikacja wykorzystuje co najmniej raz Promise i co najmniej raz wywołanie async/await. • Aplikacja co najmniej raz wykorzystuje BOM (browser object model). • Aplikacja potrafi odczytać wartość elementu formularza, zwalidować ją i przetworzyć.

Dokumentacja

Ponadto należy wykonać dokumentację projektu. Dokumentacja zawiera:

1. Stronę tytułową
2. Zdefiniowanie problemu do realizacji (czyli np. szablon jakiej strony przygotowujemy, jaką aplikację w JavaScript zamierzamy napisać - tutaj przygotowujemy opis słowny, który powinien być w miarę wyczerpujący).

3. Zaproponowany przez studenta sposób na rozwiązanie tego problemu (co, w jakiej kolejności należy wykonać i w jaki sposób będzie to wykonane, w tym za pomocą jakiej technologii problem będzie rozwiązywany; w przypadku aplikacji w JS mogą być wykorzystane schematy blokowe algorytmów; w przypadku szablonu responsywnego można opisać sposób w jaki będzie rozmieszczona zawartość, np. że zostanie w danym miejscu zastosowany suwak, slider, carousel, hamburger menu by poradzić sobie z brakiem miejsca na ekranie smartfonów by wyświetlić wszystkie elementy menu itp.).
4. Testowanie (Przygotowanie scenariuszy użycia aplikacji i następnie sprawdzenie czy realne działanie aplikacji jest zgodne z założonym teoretycznie; np. zakładamy wpisanie błędnych danych i czy aplikacja radzi sobie z tą sytuacją. Sprawdzone powinny być **kluczowe** funkcjonalności aplikacji. Odrębnie powinno być przetestowana responsywność szablonu.) Przykłady:
    a. https://inetum.pl/scenariusze-testowe/
    b. https://qualityisland.pl/jak-napisac-dobry-scenariusz-testow/
    c. https://www.softwaretestingclass.com/how-to-test-responsive-website-sample-testcases-and-examples/
    d. https://www.softwaretestinghelp.com/responsive-web-design-testing-guide/
    e. https://caniuse.com/ - sprawdzenie kompatybilności znaczników/właściwości z przeglądarkami
5. Dokumentacja kodu źródłowego w postaci JSDoc https://code.visualstudio.com/docs/languages/javascript i KSS https://github.com/kneath/kss . Przy czym należy tutaj zwrócić uwagę na kilka kwestii:

- Istnieje kilka poziomów na jakich możemy dokumentować kod źródłowy https://swimm.io/learn/code-documentation/code-documentation-examples-and-lessons-learned
- Zaproponowane wyżej dwie biblioteki służą raczej do tworzenia dokumentacji kodu skierowanej do zewnętrznego odbiorcy (tzn. kogoś, kto wykorzystuje nasz kod, a niekoniecznie kogoś, kto sam miałby go dalej rozwijać). Oznacza to, że w przypadku języka JavaScript np. podajemy ogólny opis co robi dana funkcja, klasa, jakie może przyjmować argumenty, parametry, możemy podać przykłady użycia. W przypadku dokumentowania CSS opisujemy ogólne przeznaczenie danych klas, względnie możemy udokumentować pewne konwencje, jakie chcemy, aby były wykorzystywane (np. jednolita kolorystyka, zaokrąglenia obramowania itd.).
- Poza stosowaniem tego typu systemów, niekiedy należy stosować dodatkowo niezależnie od nich "zwykłe" komentarze, które są

przeznaczone dla programistów rozwijających dane oprogramowanie. Często zamierzoną intencją twórców oprogramowania jest też ukrywanie szczegółów implementacyjnych, zatem nie chcemy ich ujawniać i dokumentować w dokumentacji publicznie dostępnej (i jednocześnie zachęcać zewnętrznych programistów, aby wykorzystywali wiedzę o tych szczegółach we własnym kodzie; jest to spowodowane faktem, że te szczegóły implementacyjne mogą się zmienić i zewnętrzny programista nie powinien polegać na nich w swoim kodzie, ponadto w złożonych systemach może to doprowadzić do utraty spójności danych i innych nieoczekiwanych błędów).

- Niekoniecznie należy komentować dosłownie każdą linię kodu; istnieje w branży "miara jakości" kodu źródłowego (comment/code ratio), która zakłada, że odpowiedni stosunek komentarzy do kodu gwarantuje jego wysoką jakość. Ścisłe trzymanie się jakiegoś ustalonego arbitralnie wskaźnika jest jednak raczej bezcelowe (zdarzały się przypadki, w których programiści pisali rozwlekłe komentarze tylko w celu utrzymania odpowiedniej wielkości tego wskaźnika! zamiast realnej potrzeby użytkowej). Dokumentować należy raczej wykonywanie złożonych, trudnych do zrozumienia, przyswojenia, względnie zapamiętania jak dokładnie działają fragmentów kodu. Komentarze powinny być utrzymywane z taką samą dyscypliną jak kod źródłowy (najgorsze co może się pojawić, to nieaktualne komentarze, które stają się wtedy zamiast źródłem informacji, źródłem dezinformacji) https://stackoverflow.blog/2021/12/23/best-practices-for-writing-code-comments/

6. Linki do źródeł oraz potwierdzenie praw autorskich do użytku treści wykorzystanych w projekcie takich jak: grafiki, szablony, fragmenty kodu.
7. Raport SEO - opis wykonanych czynności, zwiększających pozycjonowanie strony w przeglądarkach:
    a. https://www.seobility.net/en/seocheck/
    b. Light-House

Przykłady

Co do zasady strony wykonane na wzór poniższego przykładu (podgląd mobilny, gdzie widać, że nie ma żadnej próby adaptacji treści):

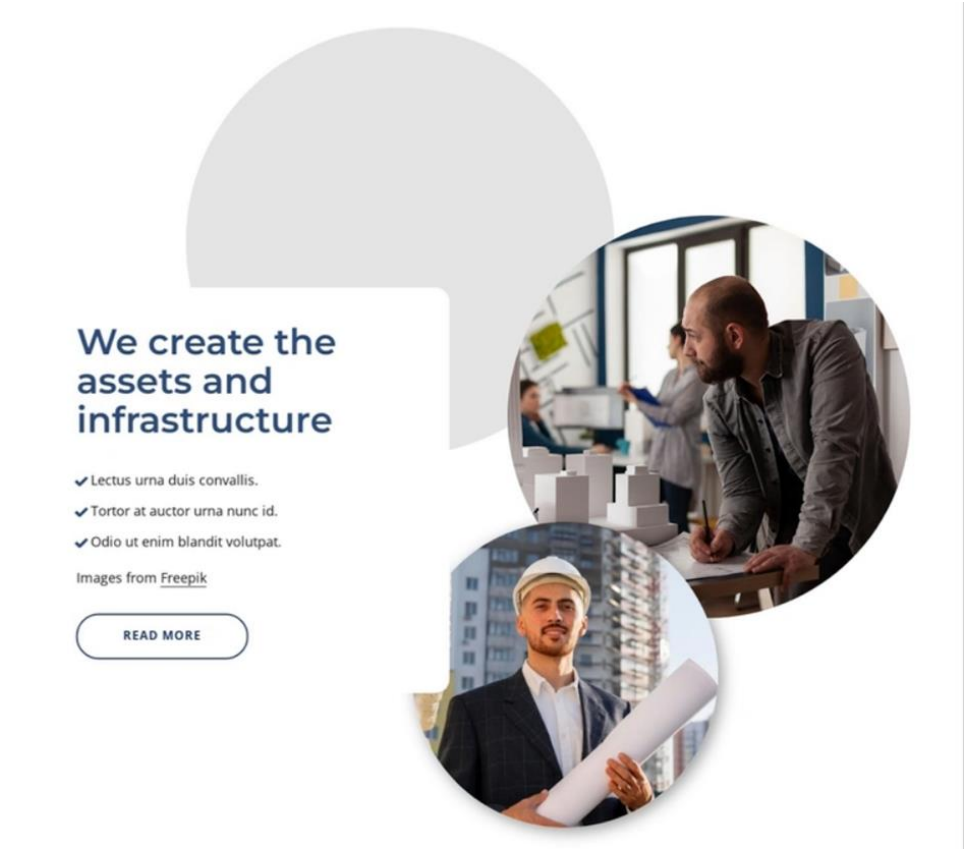
Nie mogą być zaliczone na ocenę pozytywną!

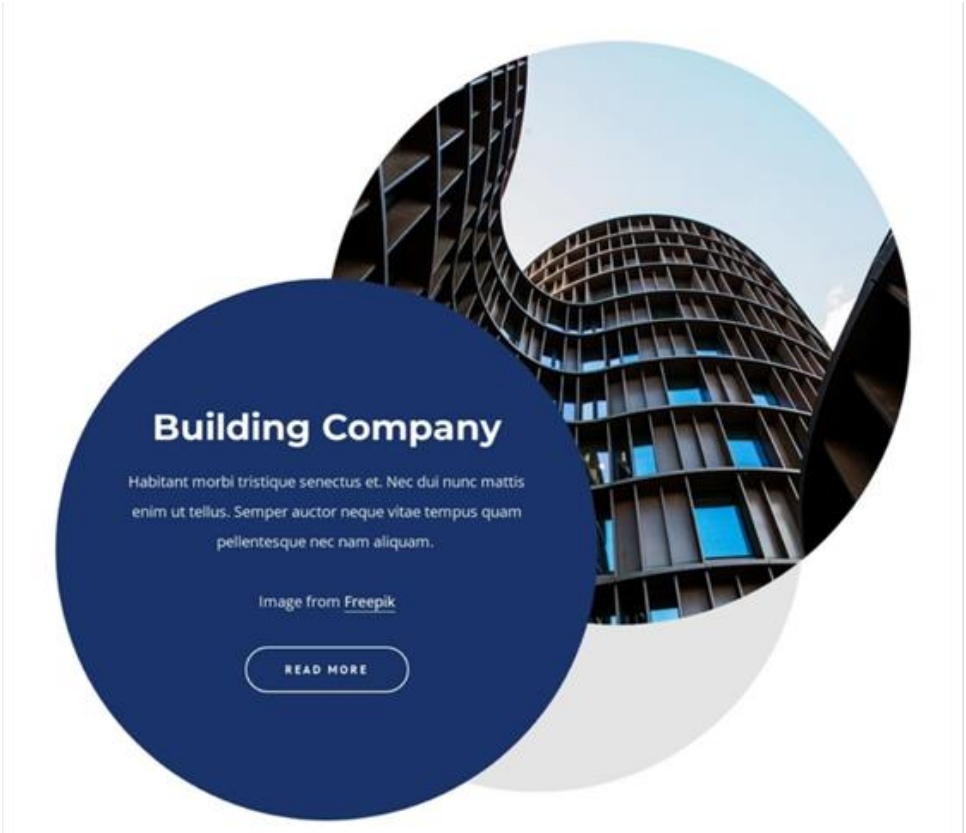
Jeżeli w projekcie wykorzystano inspirację jakimś ogólnodostępnym przykładem, należy podać link w źródłach.

Oczywiście próba oddania jako własnego ogólnodostępnego przykładu skutkuje oceną niedostateczną.

Przykłady:

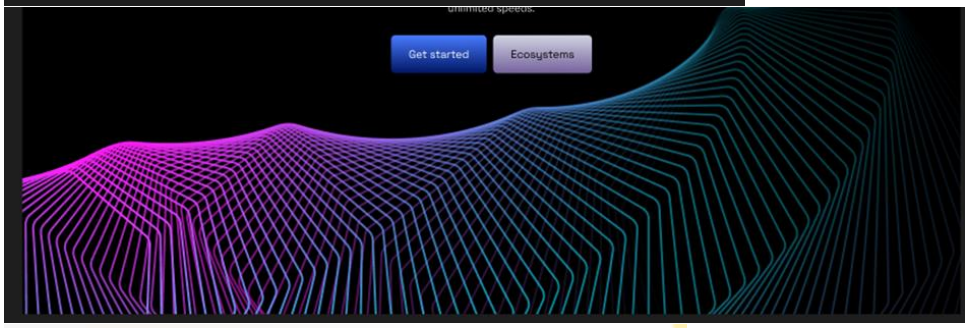
1. Złożone układy treści:





2. Elementy ozdobne:


Cap


Get started
Ecosystems


CHỦ ĐỀ ĐƯỢC QUAN TÂM