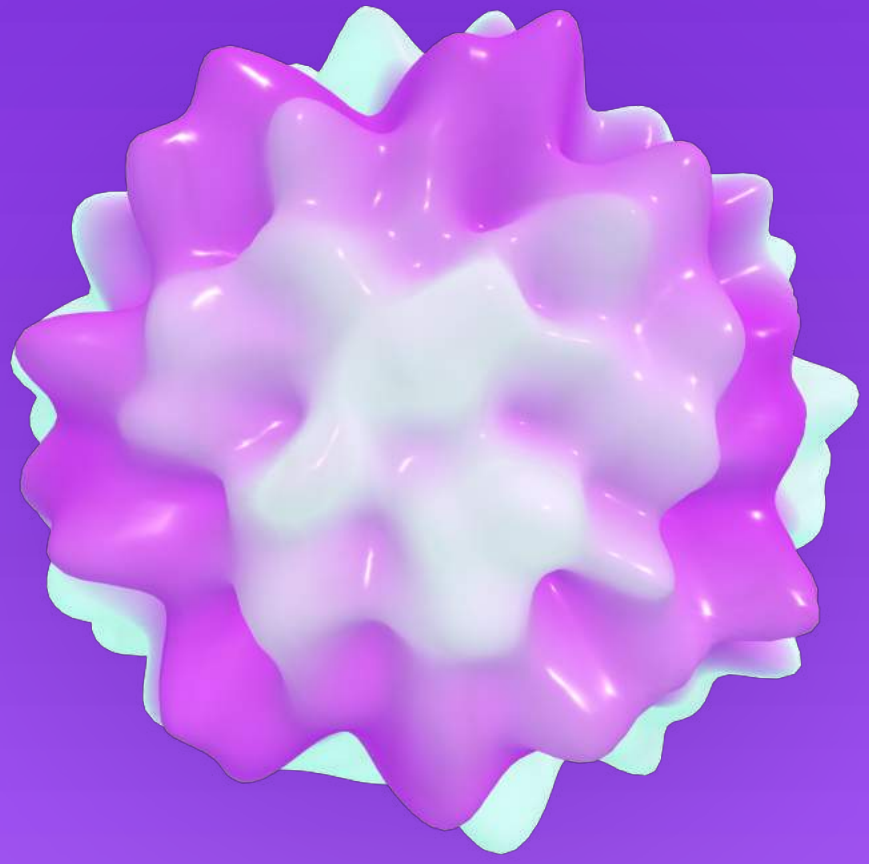
Karol Stefański

✦

DESIGN SYSTEMY PO LUDZKU

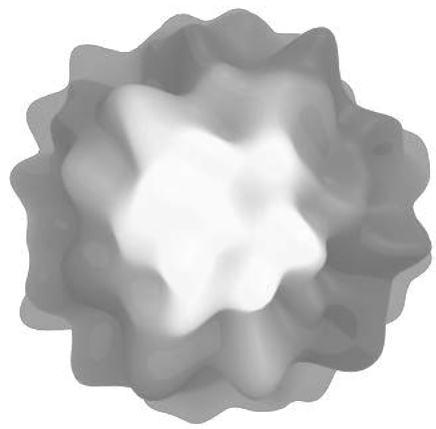
Wprowadzenie dla projektantów i projektantek interfejsów



Kółko Projektowe

JEDEN:

CZYM JEST DESIGN SYSTEM?



WSTĘP

Design system, święty graal projektowania aplikacji. Wybawienie przed chaosem. Pogromca niespójności. Arka przymierza między designerem a developerem.

Ale czym właściwie jest design system?

Cóż, może najpierw zastanówmy się, czym jest... system!

DEFINICJA

Otóż system według PWN-u to nic innego jak zespół wzajemnie sprzężonych elementów, spełniający określoną funkcję i traktowany jako wyodrębniony z otoczenia w określonym celu.

Super sprawa, nie? Sprawdźmy, co jest dalej. Okazuje się, że system to też zespół sposobów działania i wykonywania złożonych czynności, jak również całokształt zasad organizacyjnych, ogół norm i reguł obowiązujący w danej dziedzinie.

Fajnie? Fajnie. Wszystko stało się jasne i ma bardzo dużo sensu, bo tym samym jest właśnie design system. Tylko tutaj sprzężone ze sobą elementy spełniają funkcje projektowe w celu... projektowania!

Podobnie jak i metody działania, które koniec końców mają pomóc nam zbudować interfejs, czy idąc dalej – sposób na komunikację produktu z użytkownikiem.

Są tacy, którzy twierdzą, że design system to grupa *reużywalnych* (chciałbym używać ładniejszego słowa, ale to spełnia swoją rolę tak wspaniale, że muszę to przeżyć) komponentów, które w oparciu o jasne

standardy, reguły, mogą być połączone tak, że pozwolą zbudować dowolną liczbę aplikacji.

Czy jest to stuprocentowo trafne? Pewnie nie, bo artykułów, które próbują zdefiniować design system jest od cholery i ciut więcej. A teraz jeszcze ten ebook! No nic, jakoś to przeżyjemy.

Dla jednych design system będzie biblioteką stylów i komponentów. Dla innych konieczne będzie udokumentowanie zestawu reguł, jak tego właściwie używać, co wolno z tym robić, a czego nie wolno. Dlatego ja powiem krótko: miej to głęboko w poważaniu. Czyli w dupie.

Każdy produkt, zespół, czy organizacja ma inne potrzeby i cele, dlatego każdy system może wyglądać inaczej. Choć często podobnie!

PROBLEM?

W jednym z wspomnianych wcześniej artykułów, które próbują definiować design system, padł zarzut, że ludzie nazywają tak coś, co tak naprawdę jest jedynie biblioteką komponentów, a nie wielką księgą, wręcz biblią budowania i rozwijania produktu.

Co w tym złego? Absolutnie nic. A jak ktoś ma ból dupy o to, że czyjś design system jest niezgodny z czyimiś prywatnymi fanaberiami, to trzymam kciuki, żeby już na zawsze był to jedyny problem w życiu tej osoby. Jak mawia młodzież: essa.

Chodzi o to, że taki system nie będzie nigdy skończony, zatem jest użyteczny odkąd jest zwyczajnie zaczęty. Resztę (czyli na przykład te super ważne reguły gry stanowiące o tym, co, jak, i kiedy możemy stosować) będziemy dorzucać z czasem, kiedy poznamy głębiej potrzeby zespołu.

KOMPONENTY

Interfejsy, przynajmniej te najpopularniejsze, z którymi obcujemy na co dzień na ekranach, składają się najczęściej z powtarzalnych elementów: przycisków, formularzy, suwaków, przełączników, zakładek, i tak dalej.

Kiedy przychodzi ci zaprojektować nową funkcję, podstronę, cokolwiek, co jest rozwinięciem istniejącego już produktu, to nie wymyślasz koła na nowo. Po prostu korzystasz z elementów, które stworzyłeś wcześniej.

W ten sposób ułatwiamy życie:

- Sobie, bo wiemy, co do czego pasuje i nie musimy silić się na kreatywność w wymyślaniu nowych kontrolek.
- Developerowi, bo skorzysta z tego samego kodu, co ostatnim razem i wie, że będzie dobrze.
- Użytkownikom, bo nie muszą się zastanawiać, czy przycisk, który chcą kliknąć zrobi coś innego, niż ten, który kliknęli minutę temu – dwa ekrany wstecz.

Tak, to jest ta słynna „reużywalność”. Obrzydliwe słowo, wiem!

Element interfejsu pojawia się w projekcie już któryś raz, więc staje się *komponentem*. Nie mam na myśli komponentu w pliku Figmy, czy symbolu w Sketchu.

Mam na myśli coś, co zaczyna istnieć w naszym zasobniku na stałe. Coś, po co możemy sięgnąć i wiemy, że nie powinno nas zawieść.

Nagle takich komponentów powstaje kilka, część z nich łączy się w grupy, na przykład przycisk główny i przycisk drugorzędny. Wrzucasz je do jednego pudełka. Pudełko podpisujesz "przyciski".

Właśnie zacząłeś tworzyć swoją bibliotekę komponentów, brawo!

Pudełek zaczyna przybywać, masz osobne na formularze i ich pola, jeszcze inne na ikony, kolejne na checkboksy i radio – biblioteka rośnie, można powiedzieć, że wręcz puchnie w oczach!

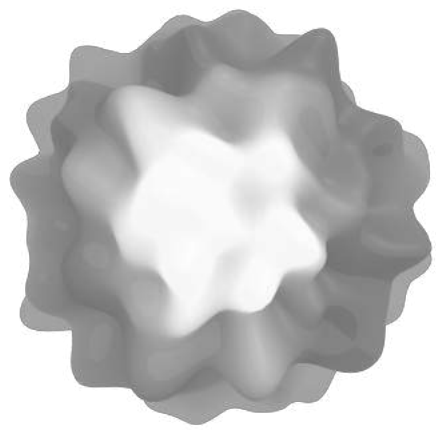
Do pudełek zaczynasz przyklejać karteczki: to wolno robić, tego nie wolno robić, a jak zrobisz to, to urwę ci łeb przy samym dupsku. Komponenty stają się elegancko posegregowane i opisane.

Mam nadzieję, że widzisz dokąd zmierzam. Tak powstaje design system. Nie jest to fizyka kwantowa, a jedynie skrupulatne podejście do organizacji – inwestycja czasu w to, by żyło się lepiej.

Nie tylko projektantom!

DWA:

DESIGN SYSTEMY W PRAKTYCE



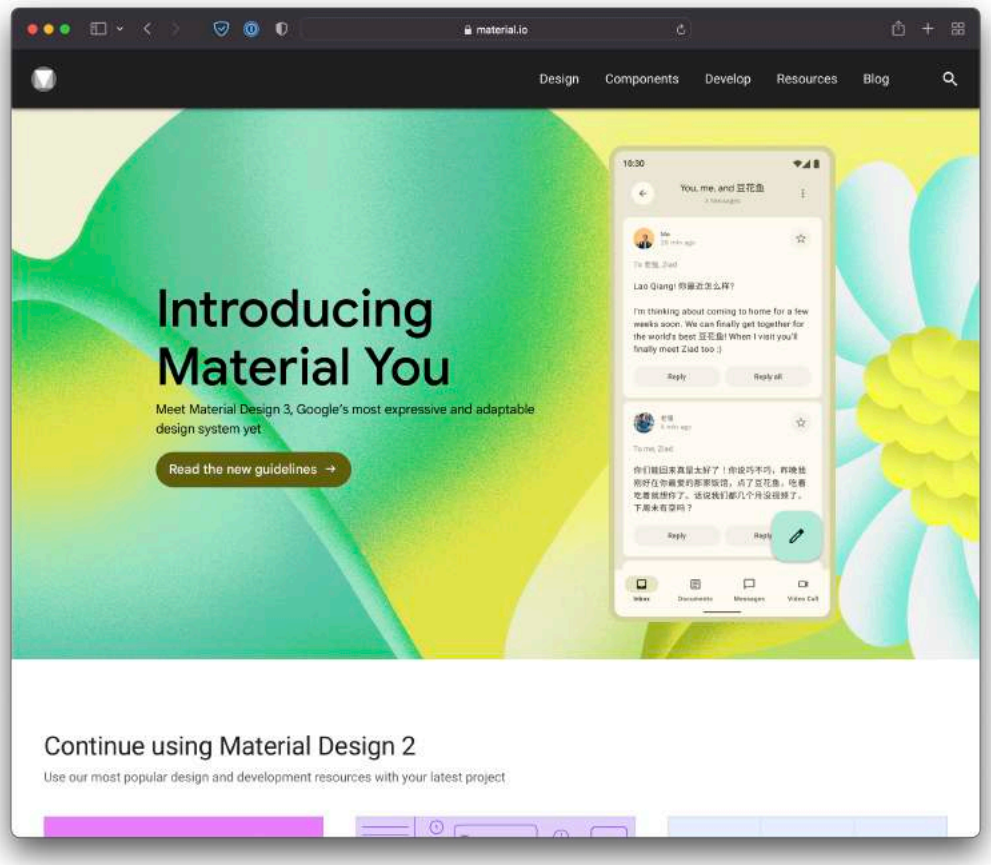
A WIĘC TAK

Powiedzieliśmy sobie, czym jest design system i dlaczego bóldupiących o niepoprawność względem własnych wyobrażeń frustratów należy spychać na trzeci plan.

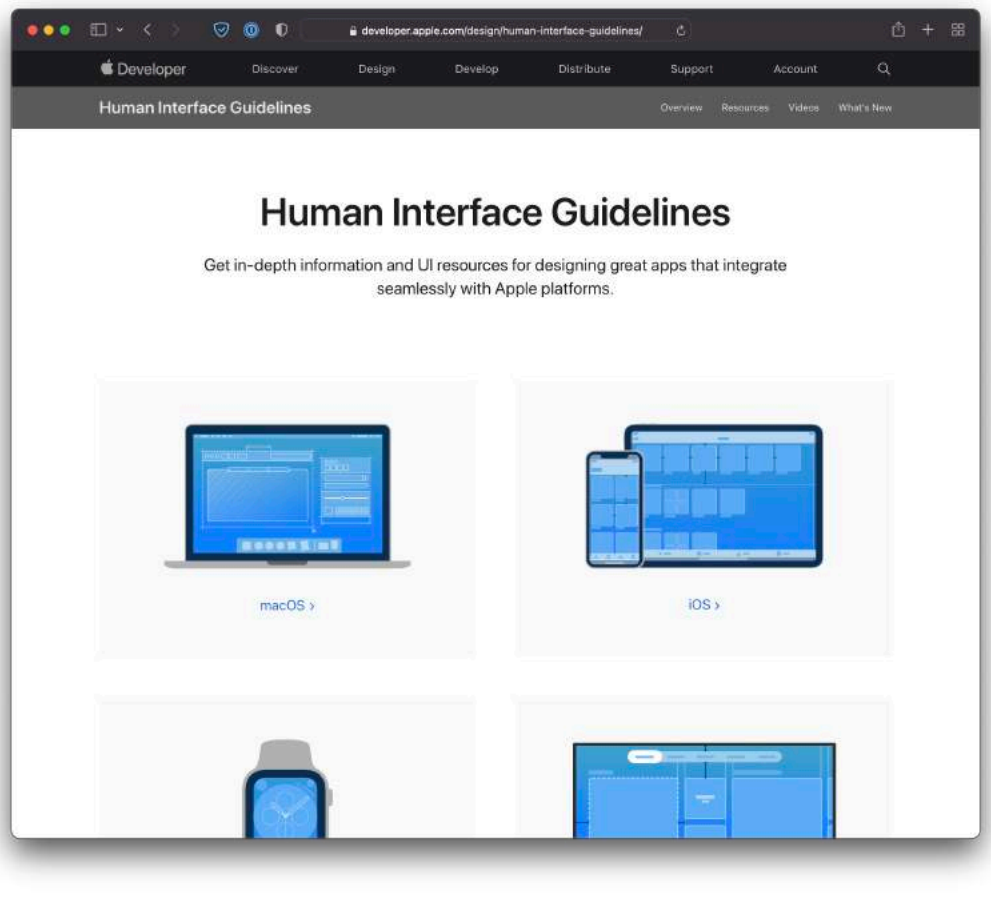
Spójrzmy zatem na istniejące już systemy, zastanówmy się chwilę nad tym, jak na nie patrzeć, i co możemy z tego patrzenia wyciągnąć.

KRÓLEWSKA PARA

Nie będzie ulegało żadnej wątpliwości, że najpopularniejszym design systemem jest Material Design od Wielkiego G. Mówię o Google, świntuchy. Nie chcę mu jednak poświęcać za dużo uwagi, ale fakty pozostaną faktami.



Razem z jabłkowym Human Interface Guidelines tworzą razem tę królewską parę, która wzajemnie od siebie czerpie i z każdym rokiem przecina się tu i ówdzie. Mimo to, że firmy ze sobą konkurują!

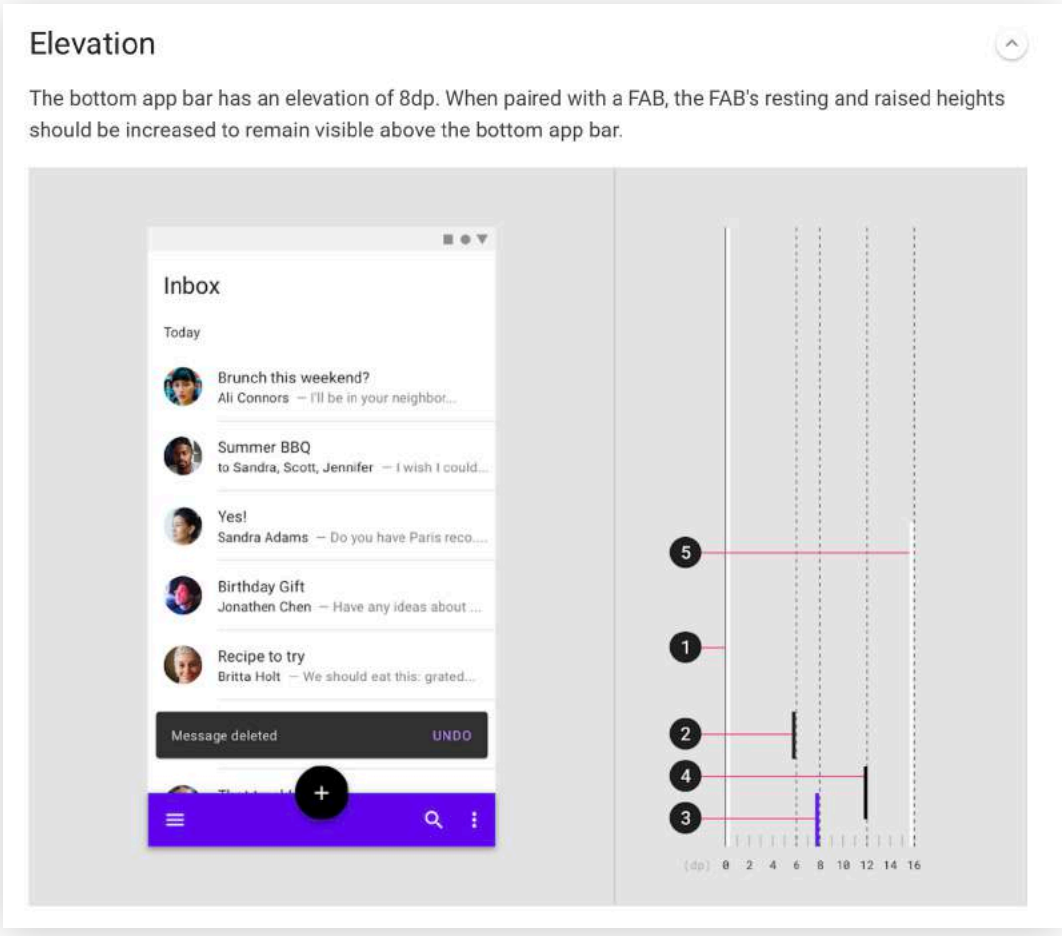


Systemy te stawiają jednak akcenty w zupełnie różnych miejscach. Material jest o wiele bardziej dokładny, jeśli chodzi o specyfikację interfejsu. Mam wrażenie, że jest bliższy developerom na poziomie spójności pomiędzy projektem a wdrożeniem.

Apple daje bardziej… teoretyczne wytyczne. Nie ma tu aż tylu doprecyzowanych odstępów pomiędzy elementami, ich bardzo skonkretyzowanych rozmiarów, i tak dalej. Wynika to również stąd, że mnóstwo z tych rzeczy jest realizowana na poziomie natywnych elementów interfejsu, które są niejako układane przez developerów, gdy budują oprogramowanie za pomocą oficjalnych narzędzi.

Material mówi do nas zdecydowanie bardziej wizualnym językiem, nie boi się pokazywać, jak coś powinno wyglądać. Często jest mocno precyzyjny i techniczny. Oczywiście – pierwsza część, czyli guidelines, to w dużej

mierze teoria, ale w połączeniu z pozostałymi częściami układanki, tworzy bardzo konkretne narzędzie do budowania sensownych, cyfrowych doświadczeń.



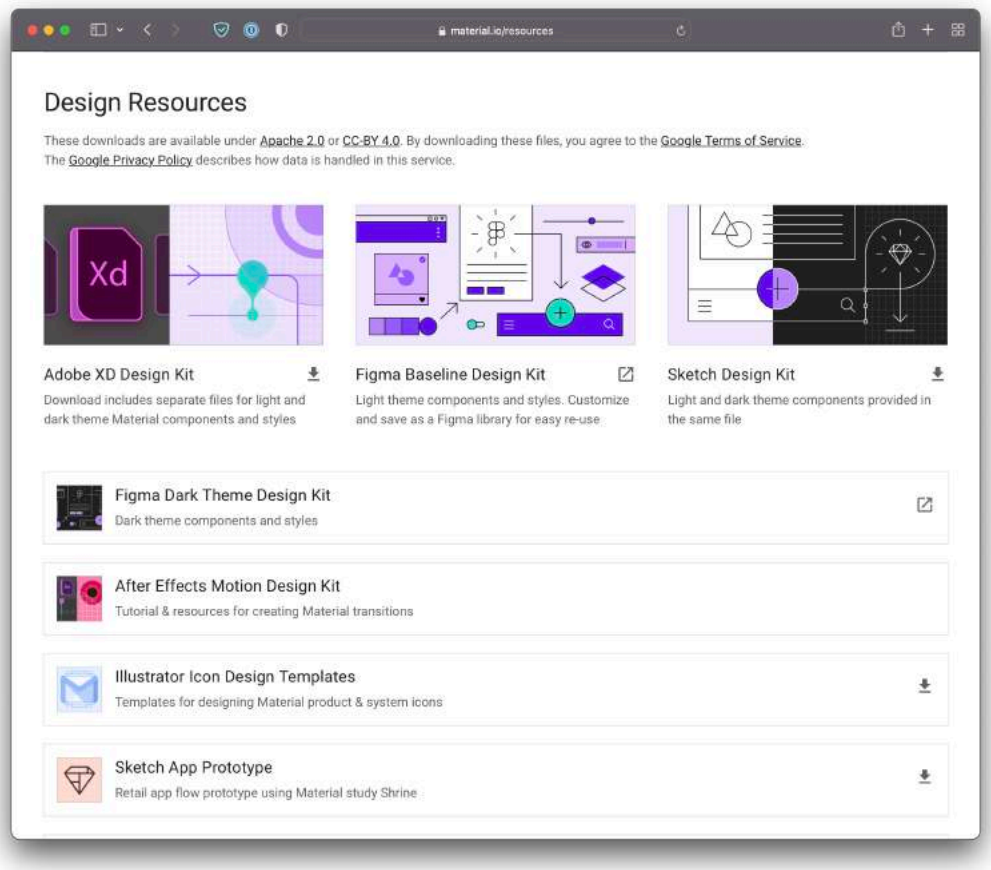
Google i Apple jednak przede wszystkim dają designerom i developerom platformy do tworzenia własnych aplikacji. Jasne, zespoły wewnętrzne również z nich korzystają, bo przecież mamy natywne apki zintegrowane z Androidem czy iOS-em. Kalendarz, Gmail, notatki!

Zależy im jednak na tym, by produkty cyfrowe tworzone przez innych ludzi, a dostępne w ich sklepach były jednak jak najbliższe doświadczeniu, które oferuje system. Tak, żeby były jego przedłużeniem, a nie zupełnie osobnymi bytami lewitującymi w cyberprzestrzeni.

Podobnie, choć na mniejszą skalę, działa system Polaris od Shopify. To ziomki od sklepów internetowych. Shopify też jest platformą i dostarczają solidny zestaw narzędzi dla developerów, którzy chcą poszerzyć ją o nowe funkcje, które nie będą wyglądały całkiem z dupy, kiedy podłączy się je do tego, co użytkownicy dostają z oficjalnego źródła.

Apple i Google dostarczają projektantom pliki, dzięki którym możemy sklejać sobie swoje interfejsy prawie (z dużym naciskiem na prawie) tak, jak układalibyśmy sobie klocki Lego.

Nie wiem, dlaczego Apple upiera się, by nie wspierać Figmy, ale na szczęście mamy wolontariuszy, którzy dzielą się dobrem w społeczności. Często za darmo, co jest absolutnie cudowne i musimy być za to wdzięczni.



Jaki jest wniosek? Taki, że Material i Human Interface Guidelines warto poznać. Nie, nie trzeba tego kuć na pamięć, bo to strata pojemności mózgu, ale na pewno wypadałoby ogarnąć, co oferują, i w razie potrzeby wiedzieć, gdzie szukać. Nikt nie każe ci tego czytać od deski do deski, po prostu... interesuj się!

To właśnie oni najczęściej wyznaczają trendy i przecierają szlaki. Również to oni mogą sobie pozwolić na najwięcej teoretyzowania, rzucania pomysłami i budowania UX-owej świadomości. Nie są to jednak jedyne design systemy na świecie.

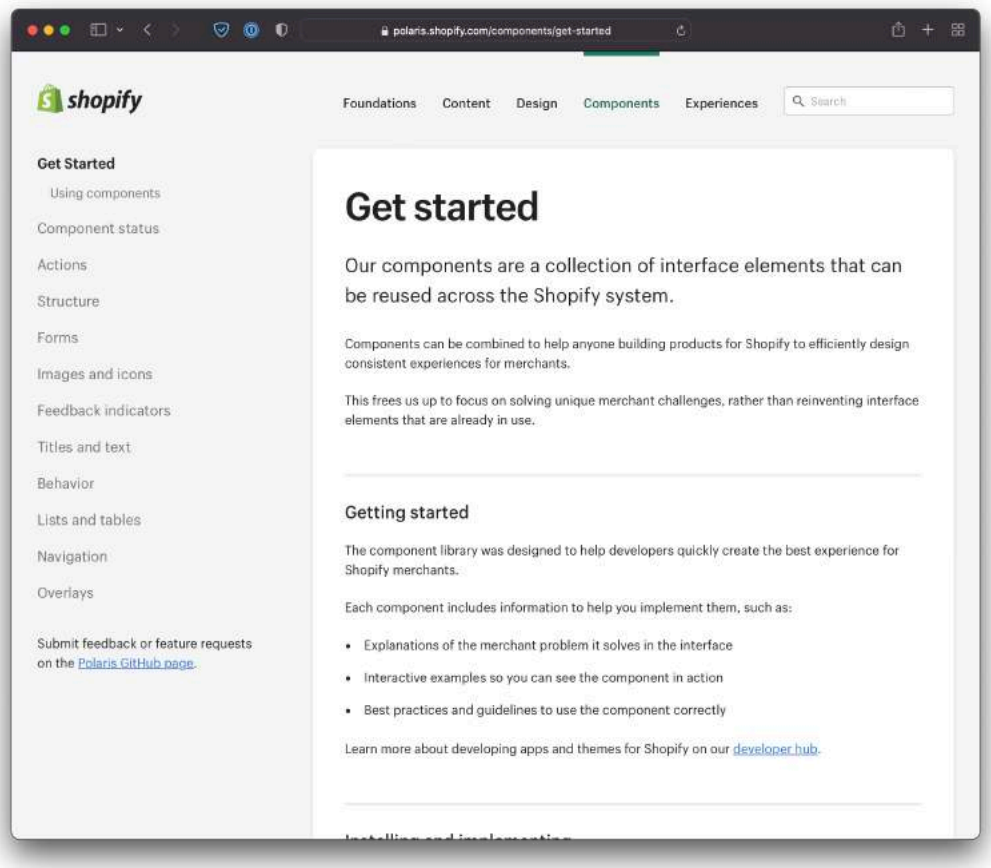
NIE TYLKO GIGANCI

Shopify

Wróćmy jednak do Shopify i ich słynnego design systemu, który nazwę wziął od wody mineralnej sprzedawanej w Biedronce.

Żartuję.

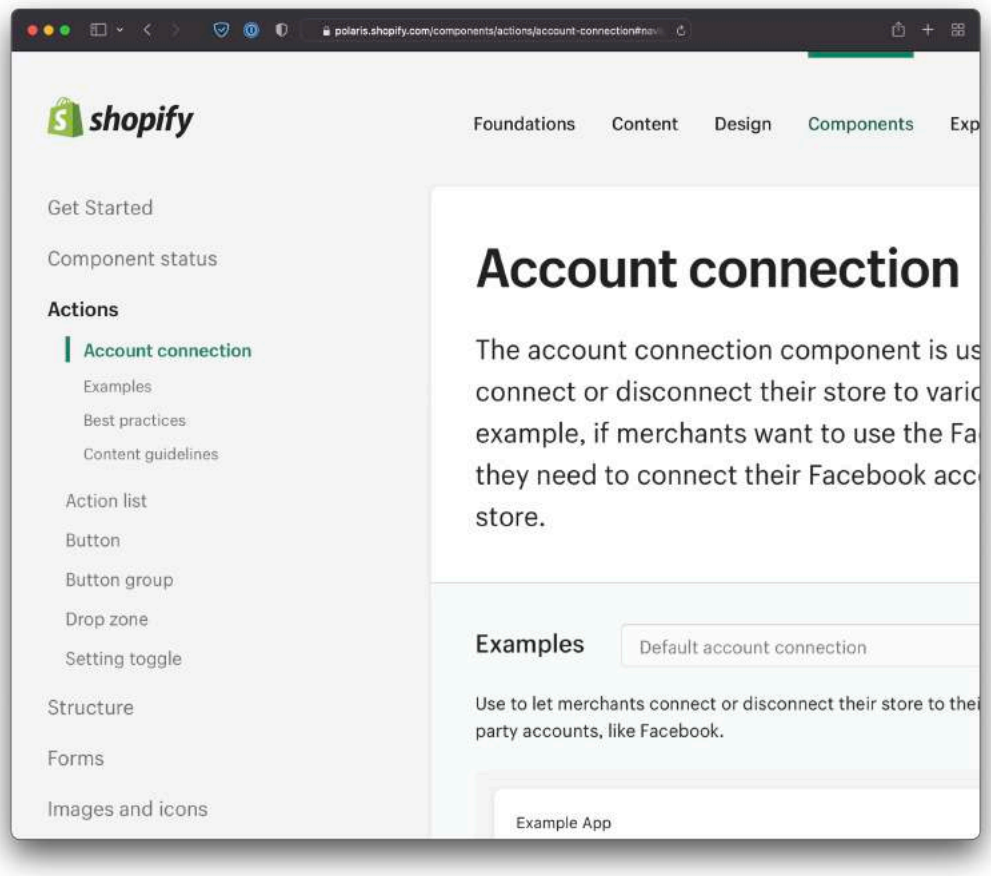
Shopify stawia komponenty interfejsu dopiero na czwartej pozycji w spisie treści. Najpierw skupiamy się na fundamentach i treści. Te pierwsze pozwalają nam zapoznać się z wartościami firmy, a także kluczowymi kwestiami, takimi jak dostępność czy internacjonalizacja.



Szczególnie warto przyjrzeć się tej drugiej, bo produkt kierowany jest na rynki całego świata. Poza tak oczywistymi tematami jak różne języki, które mogą wpływać na długości tekstów w elementach interfejsu, brane

są również pod uwagę różnice kulturowe. Choćby ikonografia, zależnie od części świata, może być różna.

Spójrzmy na komponenty. Polaris w bardzo ciekawy sposób grupuje je z podziałem na akcje (gdzie mamy m.in. przyciski i przełączniki), strukturę (z kartami, stosami i paskami), formularze (gdzie przypada prawie wszystko, co w HTML-u potrafi przyjąć jakieś dane), i tak dalej.



Zwracam na to uwagę, bo tego rodzaju kategoryzacja jest czymś, o czym warto myśleć tworząc system nawet na poziomie pliku w Figmie, Sketchu, czy czegokolwiek używacie do projektowania.

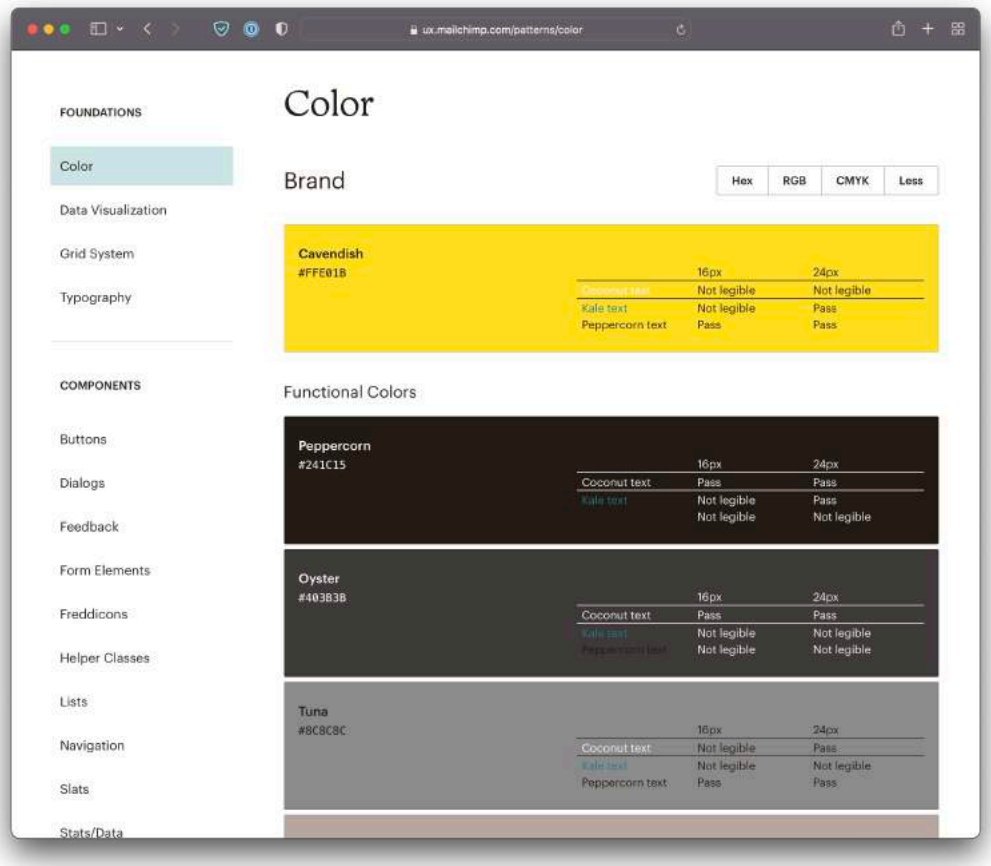
Część komponentów ma trzy warianty zależnie od platformy: web, Android i iOS. Podejście to podyktowane jest najprawdopodobniej próbą sprawienia, by mobilne odpowiedniki były jak najbliżej natywnego doświadczenia.

Zauważ, że karty w wersji na Androida mają akcje reprezentowane przez ikony, a na iOS mają etykiety (labelki) tekstowe. Taka ciekawostka.

Mailchimp

A można coś mniejszego? Pewnie że tak! Mailchimp to wielki produkt (serio, próbowałem go używać i poległem z racji przekomplikowania), a jednocześnie mały system!

Do tej pory nie wspominałem w ogóle o kolorach, a to przecież jeden z kluczowych elementów w projektowaniu wizualnym, nie? Mogą one być częścią design systemu, najczęściej przecież marka przecina się z produktem wielokrotnie!



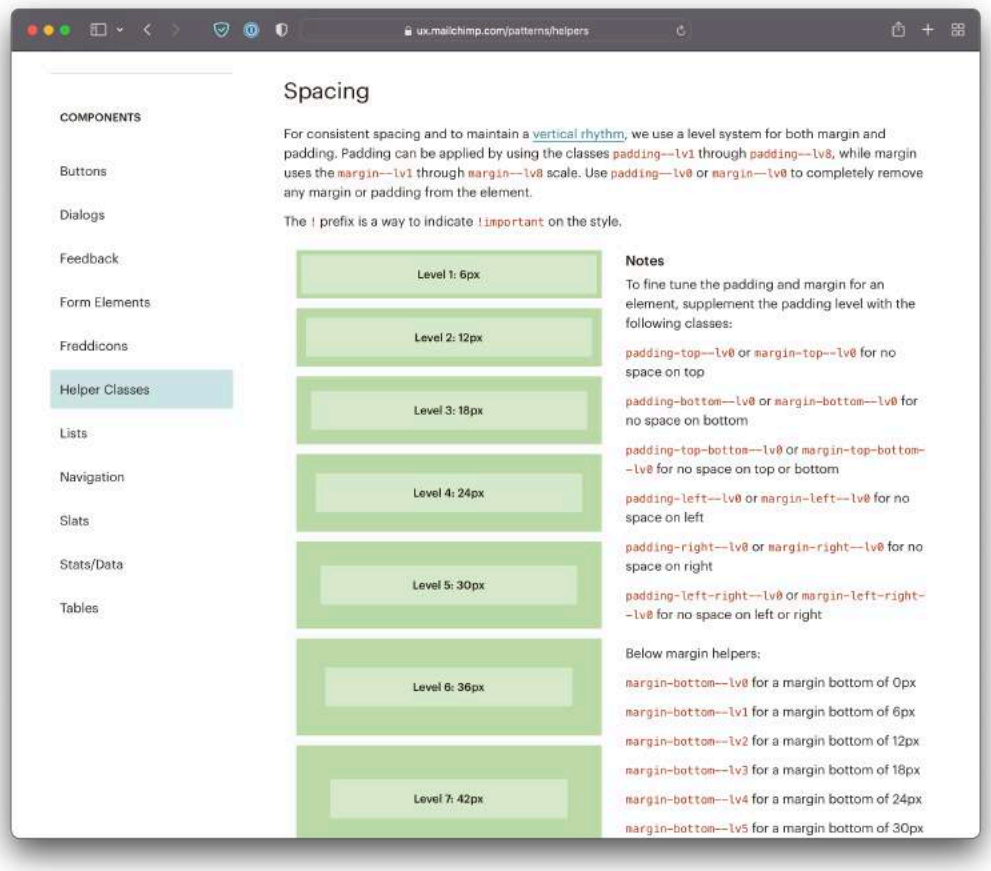
Tutaj mamy to rozwiązane pięknie, bo przesmacznie nazwane kolory (kokosowy biały i jarmużowy zielony to moje ulubione) poza tym, że ładnie się prezentują, to pełnią fantastyczną funkcje wskazującą nam, czy kombinacja tła z tekstem o danej barwie będzie spełniała normy dostępności, czyli czy kontrast będzie odpowiednio duży.

Nie mamy tu kolosalnej liczby elementów interfejsu, bo w porównaniu do takiego Material Design jest to raczej bieda, ale za to rewelacyjnie

dopieszczona bieda. Bieda, która ostatecznie okazuje się być funkcjonalną żyłą złota. Albo żyłą Bitcoinów, żeby być bardziej na czasie.

Tak czy inaczej, małpi system jest mocno zintegrowany z kodem, co sprawia, że o wiele łatwiej ograniczyć odstępstwa tego, co dowożą developerzy od zaprojektowanych przez designerów ekranów.

Co ciekawe, w design systemie zostały zdefiniowane tzw. klasy pomocnicze, czyli CSS-owe klasy, które zawierają w sobie jakiś zestaw stylów. Na takiej zasadzie został zbudowany popularny w ostatnim czasie framework Tailwind CSS. Świadczy to o tym, jak otwartym i szerokim pojęciem jest design system. Także wiesz, nie spinaj się.



No ale żeby dokończyć myśl, to w prostych słowach chodzi o to, że jeśli chcemy, żeby jakiś element miał ze wszystkich stron padding o rozmiarze np. 24 pikseli, to w kodzie stosujemy jedną klasę: *padding--lv4*, co jest bardzo spoko technicznie, chociaż szczerze mówiąc system poziomów paddingów nie jest w mojej opinii najbardziej intuicyjnym rozwiązaniem świata, ale hej – to nie ja w tym robię, tylko Mailchimpowcy.

Jeśli dla nich to sprawnie działa, to ja się bardzo cieszę, winszuję, i przyklaskuję z uśmiechem!

PODSUMOWUJĄC

Design systemy są super, warto zaznajamiać się z tym konceptem nawet, jeśli nie tworzymy własnego. Ba! Nawet, jeśli obecnie z żadnym nie pracujemy. Dlaczego?

Dlatego, że otwierają nam głowę na to, jak hierarchizować i strukturyzować elementy, które projektujemy w interfejsach na co dzień. Nawet jeśli robimy zwykłą stronkę internetową.

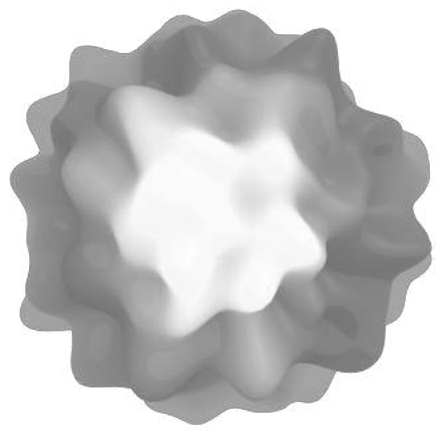
Jeśli czujecie głód systemów, to jest w internecie wiele miejsc, gdzie możecie je sobie w spokoju konsumować. Niewątpliwym plusem jest to, że jesteśmy w trendzie otwartego designu i nawet korporacje pokazują, jak wyglądają ich kredki i mazaki.

Linki

- Design Systems Repo (https://designsystemsrepo.com)
- Repository.Design (https://repository.design/)
- Design Systems (https://www.designsystems.com/)
- Awesome Design Systems (https://github.com/alexpate/awesome-design-systems)

TRZY:

CZY WARTO TWORZYĆ DESIGN SYSTEM?



NAPRAWDĘ MUSZĘ?

Czy design system to temat tak zajebisty, że nie da się bez niego projektować interfejsów?

Da się, oczywiście że tak. Jeśli budujesz jedynie szybki mock, prototyp, proof of concept, a nawet MVP, to budowanie własnego systemu nie ma wiele sensu, prawdopodobnie przepalisz tylko czas. Powiem więcej: być może więcej korzyści przyniesie wam zbudowanie produktu w oparciu o istniejący system bądź framework.

Pogadaj z developerami i zobaczcie, jak stoicie z terminami, jaki jest scope projektu, a może okaże się, że koledzy i koleżanki niedawno używali któregoś rozwiązania i mają ten patent dobrze ograny.

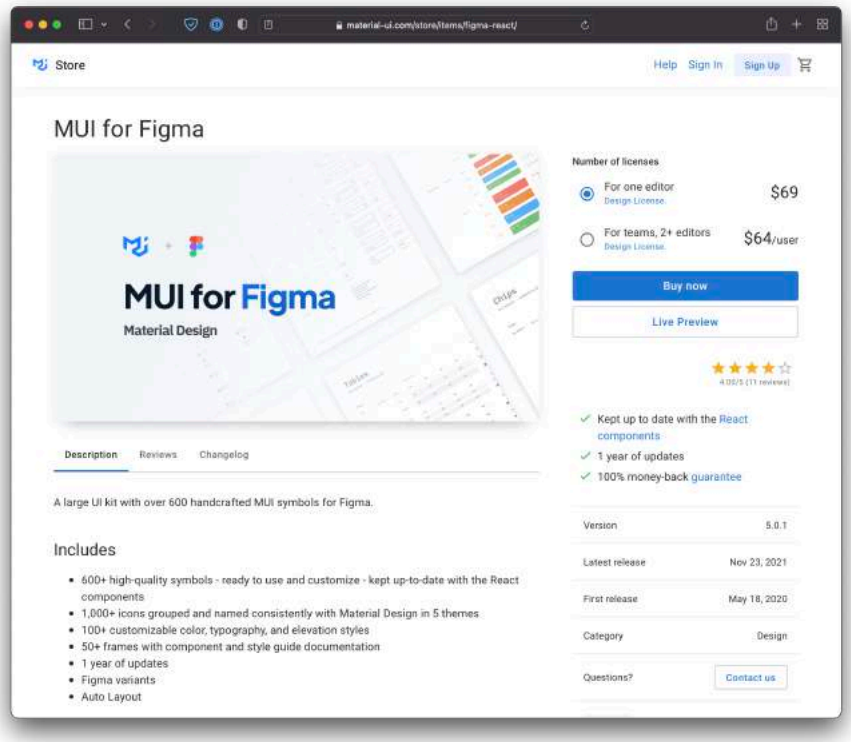
Jakiś czas temu popełniłem błąd. Oczywiście nie był to tylko mój błąd, no ale nieważne. Robiliśmy aplikację w React Native, która miała też swoją część napisaną jako aplikacja webową. Nie mogę ci tego pokazać, więc uruchom wyobraźnię.

Zamiast wykorzystać coś, co zostało już w dużej mierze zaprojektowane i zakodowane, zacząłem tworzyć ekrany praktycznie od zera.

Podstawową platformą był iOS. Gdzieś w trakcie developmentu okazało się, że większość UI była implementowana za pomocą ekstremalnie zmodyfikowanych komponentów z materialowego frameworka, część faktycznie była pisana od zera, a część była zlepkiem innych pakietów NPM, które jakoś tam pasowały.

Wszystko to sprawiło, że development zajmował kilka razy dłużej niż musiał, efekt wcale nie był zadowalający, bo pojawiało się pełno niespójności, glitchy i bugów, a kod i tak stał się na tyle pogmatwany, że naprawianie tego było naprawdę uciążliwe. A zegar tykał.

Co można było zrobić lepiej? Mogliśmy ugadać się, że pierwszą wersję aplikacji tworzymy całkowicie w oparciu o np. materialowy framework. Wystarczyłoby kupić dostęp do pliku Figmy, zrobić widoki korzystając z tej biblioteki, a developerzy nie musieliby robić absolutnie nic więcej, niż używać klas z dokumentacji



W kolejnych iteracjach moglibyśmy powoli customizować standardowe komponenty zmieniając ich formę na bardziej indywidualną, stylem dopasowaną do marki i charakteru projektu.

Jeśli jednak podobnie jak ja chcesz robić wszystko od zera, to spoko – korzystanie z design systemu, a co dopiero robienie go, nie jest konieczne, ale perspektywiczne myślenie o tym, że może przyjść na to pora – już tak. O wiele bardziej, zdecydowanie tak!

GRANULARNOŚĆ

To dobry moment, żeby wrócić do *reużywalności*, a co za tym idzie, *granularności*, komponentów. Nie, nie ma to nic wspólnego z granulatem dla zajęcy, królików, ani innych niewiniątek.

Zaczynamy myśleć o interfejsie jak o czymś rozbitym na naprawdę małe części. Na atomy. Atomy tworzące molekuły. Molekuły tworzące organizmy i tak dalej. Nie jest szalenie istotne to, które rzeczy będą czym, ale chodzi o zrozumienie zasady.

Po tym ebooku zachęcam cię do przeczytania artykułu Brada Frosta (https://bradfrost.com/blog/post/atomic-web-design/), który napisał nawet książkę o Atomic Design.

Dzięki temu możemy rozbić poszczególne elementy interfejsu na mniejsze, co sprawi, że będziemy mogli układać je w różnych konfiguracjach, a zmiany w mniejszych częściach będą automatycznie rzutowały na większe całości bez potrzeby ingerencji w nie.

Jeśli myślę o polu formularza jak o labelce połączonej z polem tekstowym, to w momencie, w którym zmienię rozmiar i grubość fonta labelek, nie będę musiał przejść przez cały projekt i edytować każdego formularza osobno.

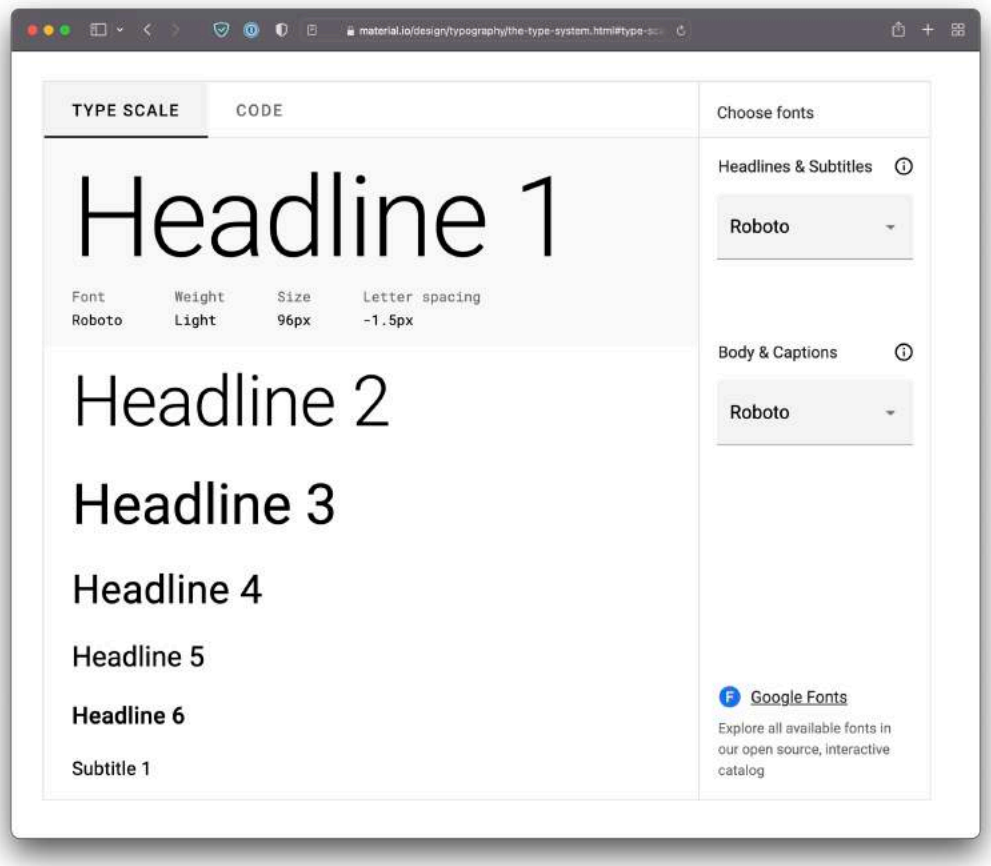
Atomami zatem mogą być kolory, typografia, odstępy pomiędzy elementami – te rzeczy, które w design systemach często stanowią sekcję całkowicie fundamentalną.

Tak też w naszym projekcie możemy założyć na przykład kolor podstawowy, kolor akcentu, kolor głównego tekstu, itd.

Więc teraz tworząc przycisk z call to action, to nie będzie "zielony przycisk", tylko "przycisk koloru podstawowego". Jest to ważne, bo w momencie, gdy twoją firmę kupi inna i uderzy cię niespodziewany rebranding, nie musisz zmieniać wszystkich przycisków z zielonego na czerwony, tylko zmieniasz jedną zmienną.

Kolor podstawowy to od teraz czerwony i po problemie, samo się robi. Taka sytuacja miała miejsce, kiedy BZWBK zamienił się w Santander.

Idziemy dalej. Tworząc system warto myśleć o typografii trochę szerzej. Rozsądne podejście proponuje Material, który zakłada 13 stylów, w tym 6 nagłówków, które powinny pokryć wszelkie typograficzne potrzeby.



Kiedy jesteś na początku danego projektu, to pewnie zorientujesz się, że sporo stylów działa lepiej lub gorzej dopiero w trakcie ich stosowania. Dlatego ważne jest, by łatwo można było je edytować, a właśnie temu służą style i zmienne. Zmienne, które od jakiegoś czasu istnieją również w CSS, dzięki czemu ładnie można je wdrożyć w kodzie.

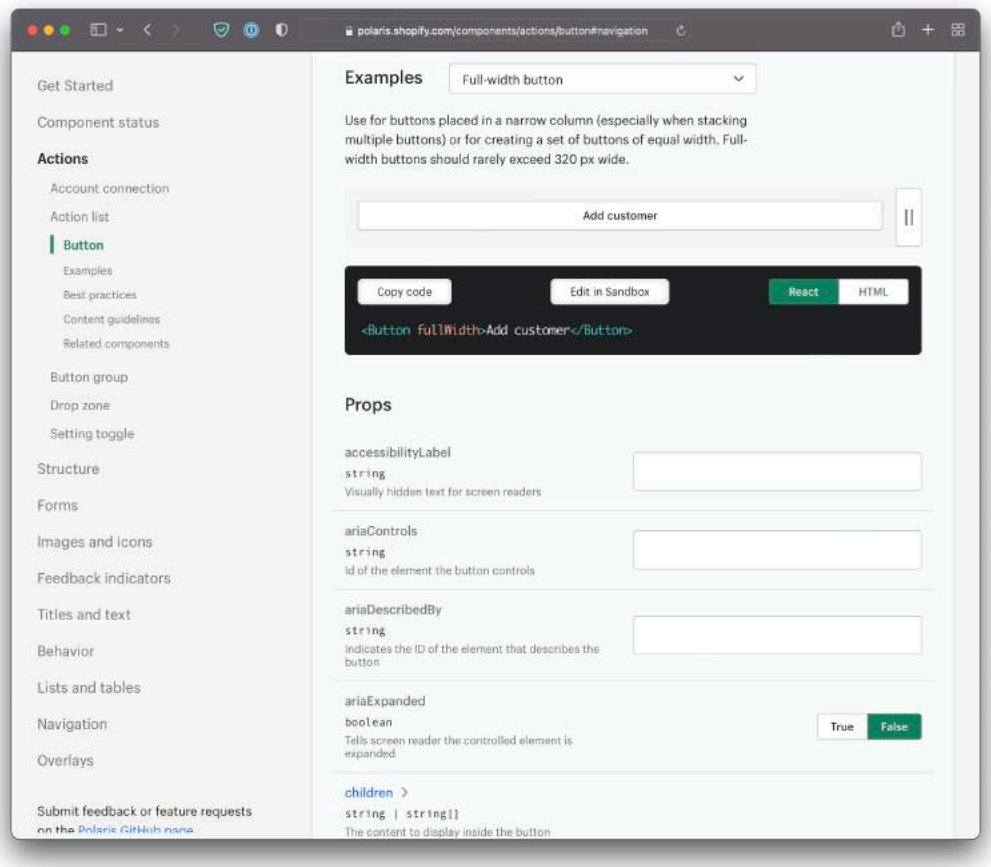
Elementy na ekranie potrzebują oddechu, który nadaje im rytm. Jeśli okaże się, że karta pod nagłówkiem lepiej wygląda odsunięta o 40, a nie 32 piksele, no to zmiana każdego widoku będzie o wiele bardziej bóldupogenna, niż edycja jednej zmiennej. Hmm, może faktycznie poziomy paddingów z Mailchimpa mają sens?

Podsumowując temat zmiennych – myślmy o nich w kontekście funkcji, a nie formy. Kolory i fonty mogą się zmieniać cały czas, ale ich zastosowanie już dużo rzadziej.

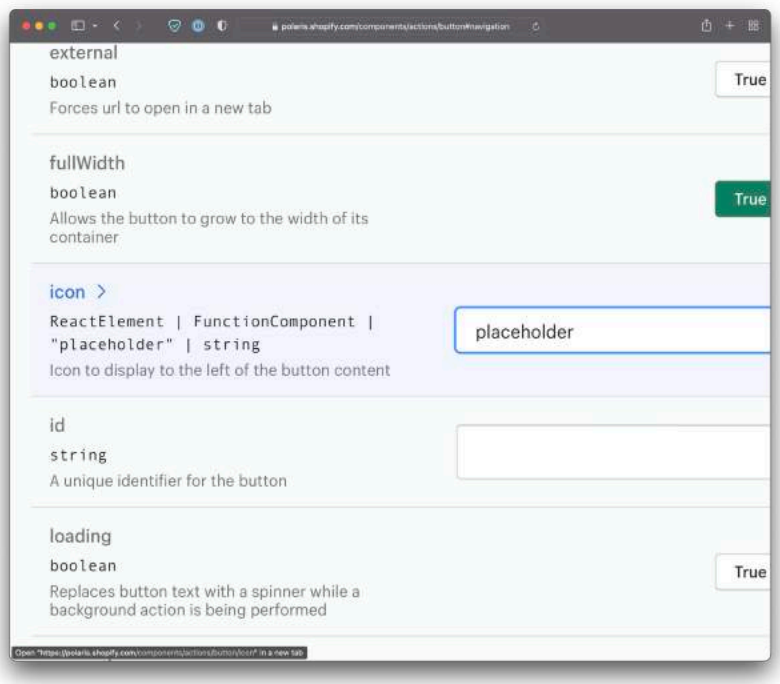
PROPSY, ZIOMECZKU!

Idąc dalej w atomiczne (czy może jednak atomowe?) podejście, trzeba powiedzieć o atrybutach elementów interfejsu. Nowoczesne rozwiązania, jak na przykład Angular i React, pozwalają manipulować jednym komponentem na naprawdę szeroką skalę za pomocą np. tzw. propsów.

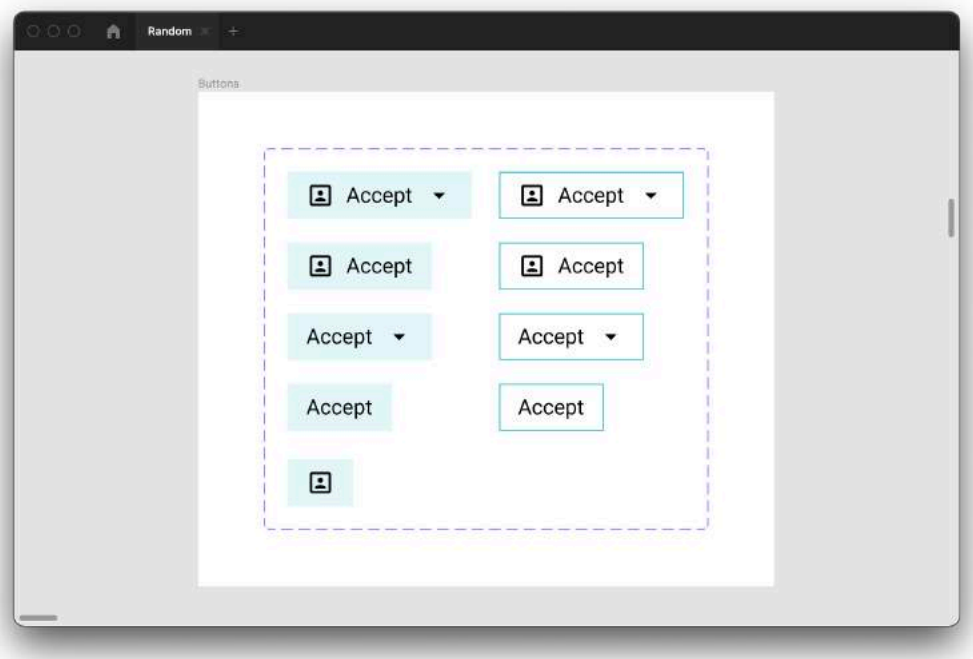
Spójrzmy jeszcze raz na Polaris. Nie, nie masz teraz patrzeć na wodę!



Przycisk w tym systemie to tak naprawdę jeden komponent, który ma tak dużo różnych opcji, że pozwala z siebie zrobić przynajmniej kilka całkowicie różnych przycisków. Jeden przełącznik dodaje nam do przycisku ikonę.



Projektując system, myślmy o komponentach w podobny sposób. W Figmie spokojnie mogę zrobić sobie jeden przycisk, który będzie miał ikony po obu stronach tekstu – a jednocześnie będzie przyciskiem bez ikon, jeśli nie będę ich potrzebował.



NA KONIEC

W tym wszystkim jednak turbo ważne jest porozumienie.

Jeśli ty będziesz sobie projektować to wszystko dla zabawy i radości z fajnie ułożonego projektu, a developerzy zignorują to podejście i będą robić to całkiem po swojemu, albo modyfikować jakieś tam inne paczki, byle tylko wyglądały, no to cała ta impreza nie ma za bardzo sensu.

Grunt to się dogadać, bo inaczej wszystko można o kant dupy rozbić.

Nie bez powodu zacząłem od tego, że trzeba usiąść z developerami i pokminić nad tym, jak powinna wyglądać współpraca i wdrożenie.

Nie bez powodu też kończę tym samym.

DZIĘKI ZA POŚWIĘCONY CZAS!

- Pamiętaj o subskrypcji kanału na YouTube
- Obserwuj mnie na Instagramie
- Złapmy się na LinkedIn
- Pogadajmy na Discordzie