



Let's CSS like a pro!

Mikołaj Dyczkowski

Developer Tools

Do czego ich używamy?

- badanie zawartości strony
- szybkie podglądanie zmian, które chcemy wprowadzić
- debugowanie (konsola, JS)
- podgląd RWD (jak strona wygląda na różnych ekranach)

Developer Tools - jak otworzyć?

⌘ + ⇧ + I lub Ctrl + Alt + I otwiera DevTools w Chrome/Firefox.

Kliknięcie prawym przyciskiem myszy na stronie i “Zbadaj”/”Skontroluj”

Do not open Safari/Edge DevTools, please :(

Większość developerów używa Chrome Developer Tools, niektórzy Firefoxa.

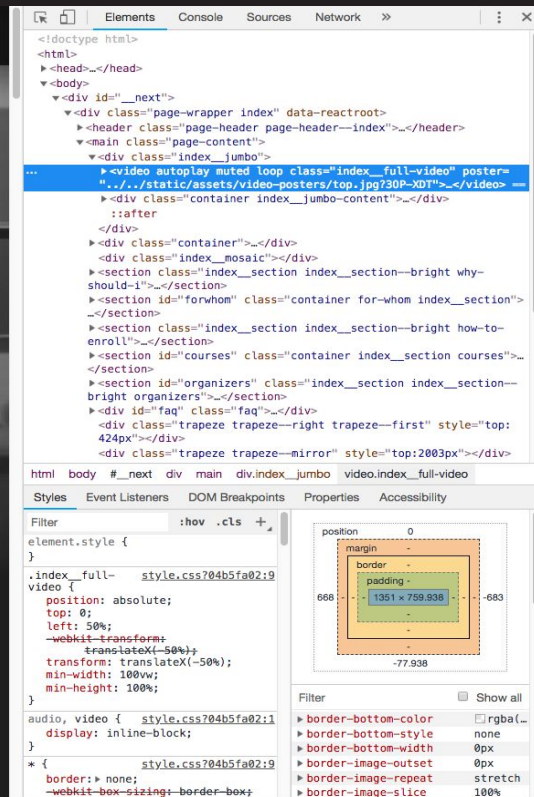
Czasami warto użyć obu - Firefox na przykład wyświetla prowadnice, ułatwiające np. sprawdzenie wyrównania elementu.

Developer Tools - “Elements”

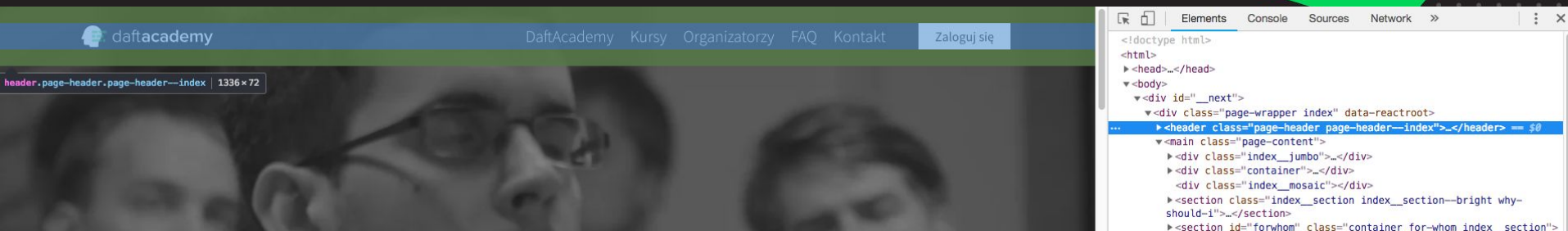


DaftAcademy?

Jakacyjna - cykl praktycznych kursów technicznych
specjalistów z grupy Daftcode przy współpracy

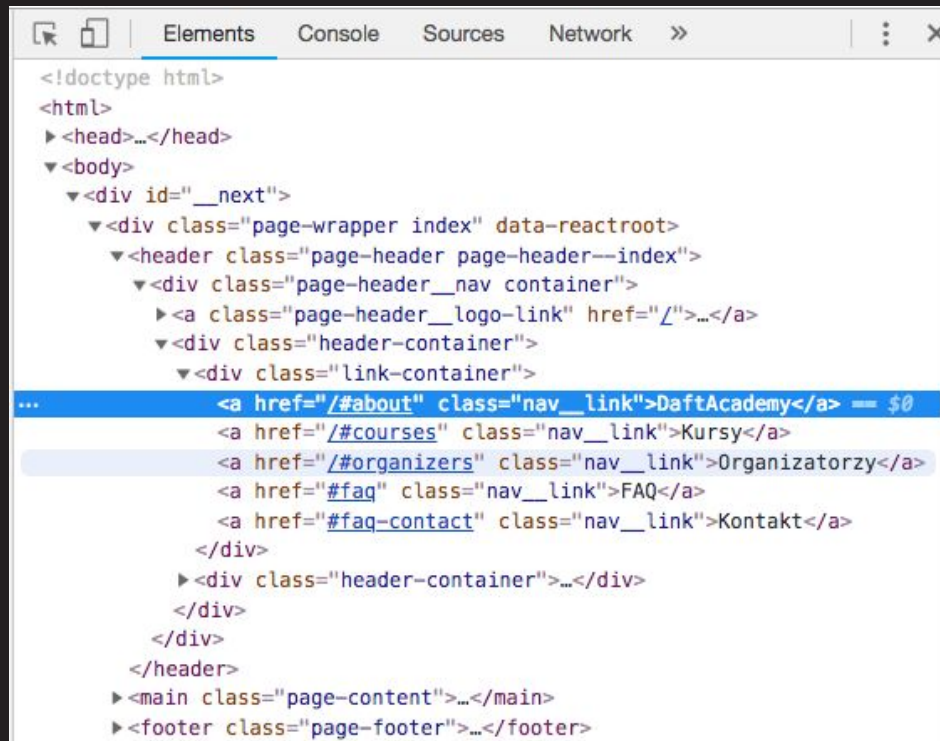


Developer Tools - “Elements”



Mamy podświetlenie aktualnie zaznaczonego elementu i podane niektóre jego atrybuty (wymiary, klasy, rodzaj elementu). Na zielono oznaczone są paddingi, na niebiesko content.

Developer Tools - “Elements”

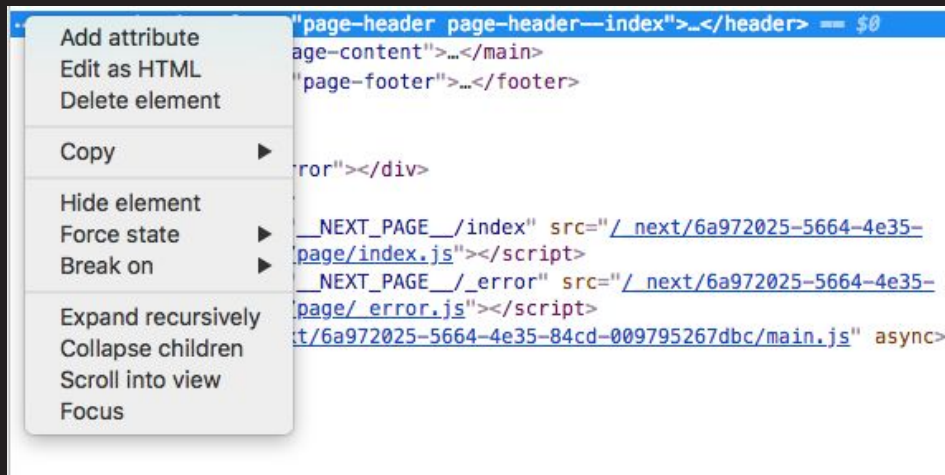


```
<!doctype html>
<html>
  ><head>...</head>
  ><body>
    ><div id="__next">
      ><div class="page-wrapper index" data-reactroot">
        ><header class="page-header page-header--index">
          ><div class="page-header__nav container">
            ><a class="page-header__logo-link" href="/">...</a>
            ><div class="header-container">
              ><div class="link-container">
                ><a href="/#about" class="nav__link">DaftAcademy</a>
                ><a href="/#courses" class="nav__link">Kursy</a>
                ><a href="/#organizers" class="nav__link">Organizatorzy</a>
                ><a href="/#faq" class="nav__link">FAQ</a>
                ><a href="/#faq-contact" class="nav__link">Kontakt</a>
              </div>
            ><div class="header-container">...</div>
          </div>
        </div>
      ><div class="header-container">...</div>
    </div>
  ><div class="page-content">...</div>
  ><div class="page-footer">...</div>
</body>
</html>
```

Widzimy całe drzewko HTML, możemy je rozwijać i podglądać dzieci lub zawartość danego elementu.

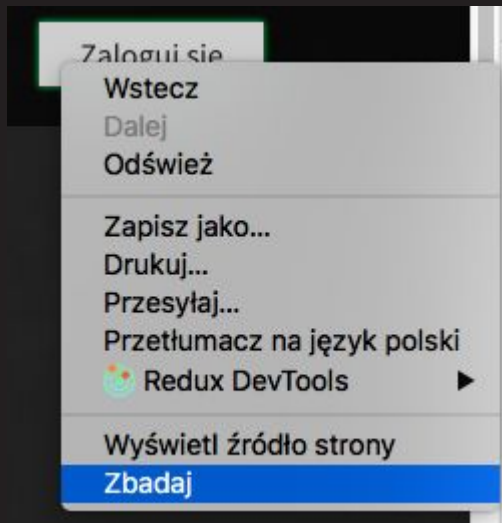
Kliknięcie w link otworzy nam również stronę, do której prowadzi widoczny tag `<a>`.

Developer Tools - “Elements”



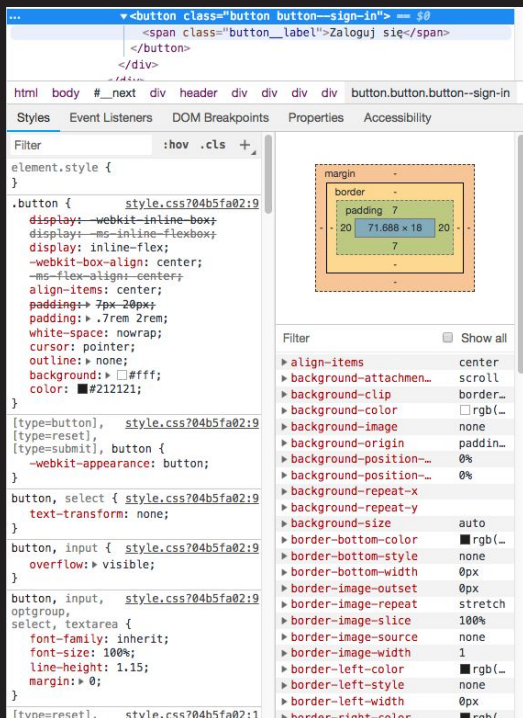
Kliknięcie prawym przyciskiem myszy, bądź w trzy kropki widoczne z lewej strony danego elementu otwiera menu kontekstowe. Możemy z niego dodać atrybut, edytować jako HTML (czyli tak jak w naszym edytorze), usunąć element itp.

Developer Tools - badanie elementu

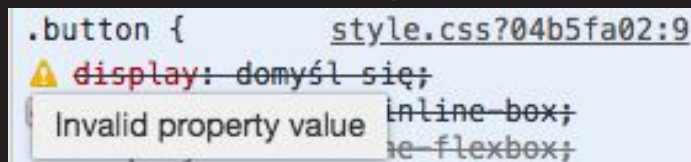


Po kliknięciu zbadaj na elemencie zakładka “Elements” podświetli nam wybrany element. W DevToolsach zobaczymy również sekcję z jego stylami.

Developer Tools - badanie elementu

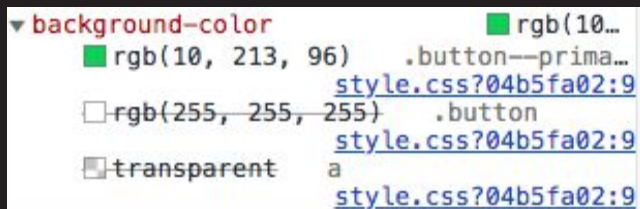


Sekcja ta pokazuje style zastosowane (bądź nie - skreślone style są nadpisane w innym miejscu bądź błędnie napisane - w takim przypadku przeglądarka poinformuje nas o tym wyświetlając dodatkowo ostrzeżenie.



Z prawej strony widzimy wizualne przedstawienie modelu pudełkowego elementu - wymiary jego zawartości, oraz paddingi, marginesy i bordery. Poniżej znajdują się ostatecznie wyliczone style.

Developer Tools - badanie elementu



Po rozwinięciu zobaczymy ostatecznie wyliczoną wartość danej reguły. Po prawej widzimy że nasz element o klasie button otrzymał cursor: pointer. Reguła ta nadpisała wartość default, domyślną dla tej przeglądarki (user agent stylesheet). Warto przypomnieć, że różne przeglądarki mogą posiadać różne domyślne style. Zawsze warto testować swoje strony na kilku przeglądarkach (choć nie przesadzajmy z Internet Explorerem 6...).

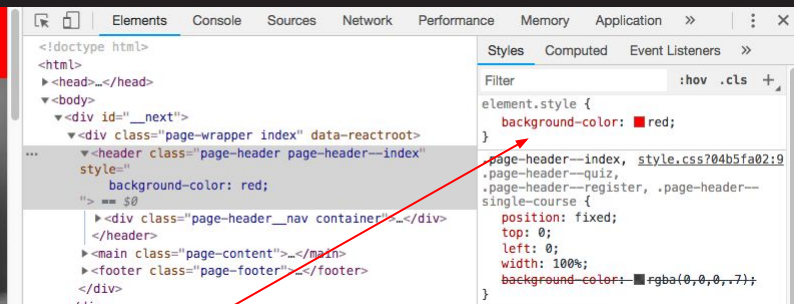
Developer Tools - dopisywanie stylu

W DevToolsach możemy dopisać własne style do już istniejących - lub stworzyć nową klasę. Te zmiany znikną po odświeżeniu strony w przeglądarce, nadają się więc głównie do sprawdzania “co się stanie jeśli” i debugowania np. źle wyświetlającego się elementu.



DaftAcademy Kursy Organizatorzy FAQ Kontakt

Zaloguj się



Dopisany styl

Developer Tools - stany i efekty

The image shows a web browser window with a dark theme. The top navigation bar includes links for 'DaftAcademy', 'Kursy', 'Organizatory', 'FAQ', 'Kontakt', and a 'Zaloguj się' button. The main content area features a large green button labeled 'Kursy' and two course cards: '4 RUBY ON RAILS BEGINNERS' and '4 UX / UI BEGINNERS'. The 'Kursy' button and the first course card are highlighted in green. The browser's developer tools are open on the right side, showing the 'Styles' panel. The 'Filter' dropdown is set to ':hov', and the 'Force element state' section has ':hover' checked. The 'element.style' section is expanded, showing a list of styles for the selected element, including 'display: flex', 'flex-direction: column', and 'margin-bottom: 4.4rem;'. The 'Courses' section of the HTML DOM is visible in the background.

DaftAcademy Kursy Organizatory FAQ Kontakt Zaloguj się

Kursy

4 RUBY ON RAILS BEGINNERS

4 UX / UI BEGINNERS

Styles Computed Event Listeners

Filter :hov .cls

Force element state

☐ :active ☒ :hover ☐ :focus ☐ :focus-within

element.style {

.courses_item, style.css?04b5fa02:9

.courses_item:activ

e, .courses_item:focus,

.courses_item:hover {

text-decoration: none;

}

.courses_item { style.css?04b5fa02:9

display: flex;

display: flex;

display: flex;

-webkit-box-orient: vertical;

-webkit-box-direction: normal;

-ms-flex-direction: column;

flex-direction: column;

cursor: pointer;

margin-bottom: 44px;

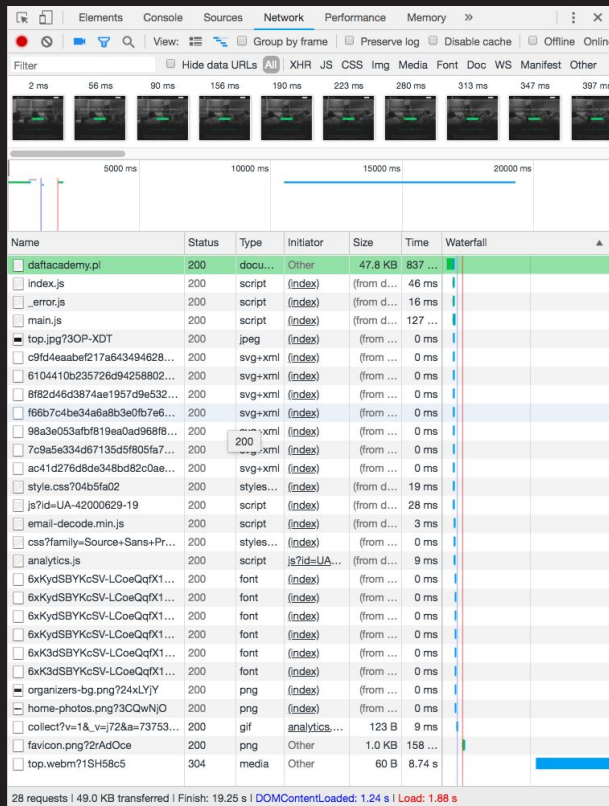
margin-bottom: 4.4rem;

}

* { style.css?04b5fa02:9

Klikając w :hov możemy “udawać” że element jest na przykład kliknięty/zhoverowany.

Developer Tools - “Network”



W zakładce “Network” możemy zobaczyć wszystkie requests, które wysłała strona - to znaczy wszystkie pliki, które musiała załadować (np. HTML, CSS, ale też JS, czy fonty). Przy każdym z nich widoczny jest czas, który upłynął od wysłania requestu do jego zakończenia, rozmiar pobranego pliku, jego typ, status (czy request był udany - status 200, bądź nieudany - np. znane wszystkim 404).

Display

Na pierwszym wykładzie usłyszeliście, że elementy HTML dzielą się (w większości) na inline i block. Tą właściwość można zmienić w CSS używając reguły display: <wartość>.



```
1 div {  
2   display: inline;  
3 }
```

Takie ostylowanie znacznika div sprawi, że wszystkie divy na stronie będą wyświetlane jak elementy inline. Może być to wyjątkowo błędogenne i prowadzić do nieprzewidzianych konsekwencji, dlatego najlepszą metodą jest stylowanie tylko po klasach.

Display - wartości

- block
- inline
- inline-block (element zachowuje się jak inline'owy, możemy mu jednak nadawać właściwości takie jak height, width, margin-top, margin-bottom)
- flex
- ... wiele innych, znacznie rzadziej używanych.

Further reading: [display](#)

Flexbox

Element, który posiada `display: flex` nazywamy *flex container*. Jego dzieci to *flex items*.

Głównym założeniem flexboxa jest pozwolenie kontenerowi na zmienianie wymiarów swoich dzieci w zależności od dostępnego miejsca - to jest zwiększanie ich, jeśli jest go dużo i zmniejszanie jeśli jest mało. Jest to szczególnie przydatne, gdyż dostosowuje rozmiary contentu w zależności od, na przykład, rozmiaru ekranu na którym otwieramy stronę (telefon, tablet, desktop).

Flex-direction

Reguła flex-direction ustala kierunek układania dzieci flex containera. Domyślną wartością jest row, w którym elementy układają się od lewej do prawej. Wartość column spowoduje układanie dzieci od góry do dołu. Wartości row-reverse i column-reverse stosują odwróconą kolejność.

CodePen: [flex-direction](#)

Flex-wrap

Domyślnie, wszystkie flex itemy będą próbować zmieścić się w jednej linii (lub kolumnie). Reguła flex-wrap pozwala zmienić to zachowanie i zawijać niemieszczące się elementy do następnej linii(kolumny). Wartość wrap-reverse zawija je w odwrotnej kolejności - to jest niemieszczące się elementy zostają w pierwszym wierszu (kolumnie), a reszta przechodzi do następnej.



```
1 .flex-container {  
2   display: flex;  
3   flex-wrap: nowrap | wrap | wrap-reverse;  
4 }
```

CodePen: [flex-wrap](#)

Flex-grow/shrink

Flex grow pozwala elementowi rosnać proporcjonalnie do innych elementów. Analogiczna reguła flex-shrink określa z kolei “kurczenie się” elementu.

```
1 .flex-item-1 {  
2   flex-grow: 1;  
3 }  
4  
5 .flex-item-2 {  
6   flex-grow: 3;  
7 }
```

Na tym przykładzie flex-item-1 zajmie $\frac{1}{4}$ dostępnego miejsca, flex-item-2 zaś - $\frac{3}{4}$ (jeśli będzie to możliwe). Dostępne miejsce oznacza miejsce pozostałe w kontenerze z display: flex po zmieszczeniu obecnego contentu.

CodePen: [flex-grow](#)

Pozycjonowanie flex itemów

Dzieci kontenera flexowego ustawiane są zgodnie z regułami justify-content, align-content i align-items.

- justify-content definiuje rozłożenie elementów wzdłuż głównej osi kontenera (poziomej, gdy flex-direction ma wartość *row*, pionowej, gdy *column*). Reguła ta dysponuje miejscem pozostałym, gdy wszystkie elementy mają ustalone wymiary, bądź osiągnęły już swoją maksymalną wielkość.
- align-items definiuje rozłożenie elementów wzdłuż osi prostopadłej do głównej osi (tj. odwrotnej niż justify content).
- align-content układa rzędy (lub kolumny) w kontenerze, jeżeli pozostało w nim wolne miejsce.

Justify-content

Przyjmuje wartości:

- flex-start (do lewej gdy flex-direction jest row, na górze gdy column)
- flex-end (prawa - row, dół - column)
- center
- space-around
- space-evenly
- space-between

CodePen: [justify-content](#)

Align-items

Przyjmuje wartości:

- flex-start (do lewej gdy flex-direction jest column, na górze gdy row)
- flex-end (prawa - column, dół - row)
- center
- baseline (wyrównuje flex itemy tak, aby ich linia tekstu była na tym samym poziomie)
- stretch (rozciąga elementy na całą wysokość (szerokość) kontenera)

CodePen: [align-items](#)

Align-content

Przyjmuje wartości:

- flex-start (kolumny po lewej gdy flex-direction jest column, rzędy na górze gdy row)
- flex-end (prawa - column, dół - row)
- space-between
- space-around
- stretch (rozciąga elementy na całą wysokość (szerokość) kontenera)

CodePen: [align-content](#)

Align-self

Reguła używana na dziecku flexowego kontenera, pozwala nadpisać jego domyślne (bądź nałożone regułą align-items) ustawienie względem osi prostopadłej do głównej. Dostępne wartości są takie same jak w przypadku align-items + wartość auto.

CodePen: [align-self](#)

Order

Domyślnie, dzieci kontenera flexowego ustawiane są w kolejności zgodnie ze swoją kolejnością w drzewie HTML. Kolejność tą można zmienić ustawiając flex itemowi regułę order. Reguła ta przyjmuje wartości całkowitoliczbowe. Domyślną wartością jest 0. Element z mniejszą wartością order będzie ustawiany przed elementem z większą wartością. Elementy z taką samą wartością ustawiane będą zgodnie z kolejnością w dokumencie.

CodePen: [order](#)

Further reading

- [Kompendium wiedzy o flexboksie w przystępny sposób](#)
- [Flexbox Froggy - gra ucząca Flexboxa! Bardzo polecam :\)](#)
- [Materiały na temat flexboxa na MDN](#)
- [Wstęp do flexboxa z animowanymi przykładami \(znacznie łatwiejszymi niż te na moim CodePenie :\)\)](#)
- [Google DevTools For Beginners](#)



Dziękuję!

Pytania?