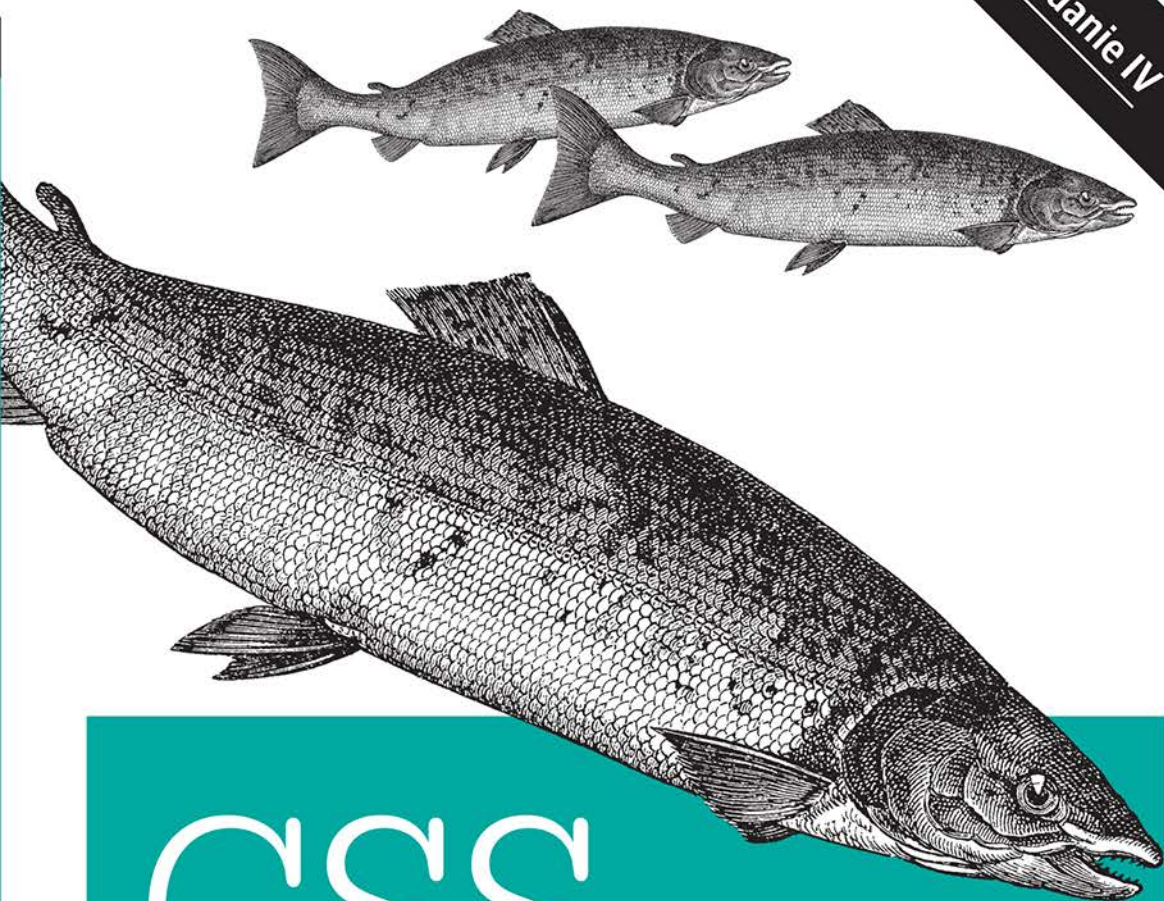


O'REILLY®

Wydanie IV



# CSS

## Kaskadowe arkusze stylów

PRZEWODNIK ENCYKLOPEDYCZNY

Helion

Eric A. Meyer, Estelle Weyl

Tytuł oryginału: CSS: The Definitive Guide: Visual Presentation for the Web, 4th Edition

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-4083-1

© 2019 Helion S.A.

Authorized Polish translation of the English edition of CSS: The Definitive Guide 4e ISBN 9781449393199 © 2018 Eric Meyer and Estelle Weyl

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/kasty4.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/kasty4>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

<b>Przedmowa .....</b>	<b>19</b>
<b>1. CSS i dokumenty .....</b>	<b>27</b>
Krótka historia stylu (w internecie)	27
Elementy	28
Elementy zastępowane oraz niezastępowane	29
Sposoby wyświetlania elementów	29
Łączenie CSS i HTML	33
Znacznik link	33
Element style	38
Dyrektywa @import	38
Odwołania przez HTTP	39
Style wewnętrzne	40
Zawartość arkusza stylów	41
Znaczniki	41
Struktura reguł	41
Prefiksy przeglądarek	42
Obsługa białych znaków	42
Komentarze CSS	44
Zapytania o media	45
Zastosowanie	45
Proste zapytania o media	45
Typy mediów	46
Deskryptory mediów	46
Deskryptory cech mediów i typy wartości	48
Zapytania o właściwości	49
Podsumowanie	52

<b>2. Selektory .....</b>	<b>53</b>
Podstawowe reguły tworzenia stylów	53
Selektory elementów	54
Deklaracje oraz słowa kluczowe	55
Grupowanie	57
Grupowanie selektorów	57
Grupowanie deklaracji	59
Grupowanie wszystkiego	60
Nowe elementy w starych przeglądarkach	61
Selektory klas oraz identyfikatorów	62
Selektory klas	62
Wiele klas	64
Selektory identyfikatorów	66
Wybór pomiędzy klasą a identyfikatorem	66
Selektory atrybutów	67
Proste wybieranie atrybutów	68
Wybieranie według dokładnej wartości atrybutu	69
Wybieranie oparte na częściowej wartości atrybutu	71
Identyfikator braku wrażliwości na wielkość znaków	75
Wykorzystywanie struktury dokumentu	76
Omówienie relacji pomiędzy elementami rodzica i dziecka	76
Selektory kontekstowe	78
Wybieranie elementów dzieci	81
Wybieranie przylegających elementów rodzeństwa	82
Wybieranie następných elementów rodzeństwa	83
Selektory pseudoklas	84
Łączenie pseudoklas	85
Pseudoklasy strukturalne	85
Pseudoklasy dynamiczne	97
Pseudoklasy obsługujące stan interfejsu	102
Pseudoklasa :target	107
Pseudoklasa :lang	108
Pseudoklasa negacji	109
Selektory pseudoelementów	112
Podsumowanie	115
<b>3. Specyficzność i kaskada .....</b>	<b>117</b>
Specyficzność	117
Deklaracje oraz specyficzność	119
Specyficzność selektora uniwersalnego	120
Specyficzność selektorów identyfikatorów oraz atrybutów	120
Specyficzność liniowych stylów wewnętrznych	121
Ważność	121

Dziedziczenie	122
Kaskada	125
Sortowanie według wagi oraz pochodzenia	126
Sortowanie według specyficzności	127
Sortowanie według kolejności	127
Wskazówki prezentacyjne niezwiązane z CSS	130
Podsumowanie	130
<b>4. Wartości oraz jednostki .....</b>	<b>131</b>
Słowa kluczowe, łańcuchy znaków i inne wartości tekstowe	131
Słowa kluczowe	131
Łańcuchy znaków	134
Adresy URL	135
Obrazy	136
Identyfikatory	136
Wartości liczbowe i procentowe	137
Wartości całkowite	137
Wartości liczbowe	137
Wartości procentowe	138
Wartości ułamkowe	138
Odległości	138
Bezwzględne jednostki długości	138
Jednostki rozdzielczości	141
Względne jednostki długości	142
Wartości obliczeniowe	147
Wartości atrybutów	148
Kolory	149
Kolory nazwane	149
Kolory RGB i RGBA	150
Kolory HSL i HSLa	154
Słowa kluczowe dotyczące kolorów	157
Kąty	157
Czas i częstotliwość	158
Położenie	159
Właściwości niestandardowe	160
<b>5. Fonty .....</b>	<b>163</b>
Rodziny fontów	163
Posługiwanie się rodzinami gatunkowymi	164
Określanie rodziny fontów	165

Zastosowanie deklaracji @font-face	168
Wymagane deskryptory	168
Inne deskryptory fontów	173
Łączenie deskryptorów	176
Grubość znaków	178
Jak działają wagi fontów	179
Pogrubianie	182
Wagi lżejsze	184
Deskryptor font-weight	184
Rozmiar tekstu	185
Rozmiary bezwzględne	186
Rozmiary względne	188
Wartości procentowe a rozmiary	188
Rozmiar tekstu a dziedziczenie	189
Używanie jednostek długości	192
Automatyczne korygowanie wielkości znaków	193
Style znaków	195
Deskryptor font-style	197
Rozciąganie znaków	198
Deskryptor font-stretch	200
Kerning znaków	201
Warianty fontów	201
Wartości ze specyfikacji Level 3	203
Cechy fontów	204
Deskryptor font-feature-settings	205
Generowanie odmian znaków	206
Właściwość font	207
Uwzględnianie wysokości wiersza	209
Poprawne wykorzystywanie skrótów	210
Wykorzystywanie fontów systemowych	210
Dobieranie fontów	211
Podsumowanie	213
<b>6. Właściwości tekstu .....</b>	<b>215</b>
Wcięcia oraz wyrównanie w linii	215
Wcięcia tekstu	216
Wyrównywanie tekstu	218
Wyrównywanie ostatniego wiersza	222
Wyrównywanie elementów liniowych w pionie	223
Wysokość wiersza	223
Wyrównanie tekstu w pionie	227

Odstępy pomiędzy słowami oraz literami	232
Odstępy między słowami	233
Odstępy między literami	234
Odstępy a wyrównanie	235
Transformacja tekstu	236
Dekoracja tekstu	238
Dziwne dekoracje	239
Właściwość text-rendering	241
Cień tekstu	242
Obsługa białych znaków	244
Ustalanie wielkości tabulatorów	246
Zawijanie wierszy i dzielenie słów	247
Zawijanie tekstu	252
Tryby pisania	253
Ustawianie trybów pisania	253
Zmiana orientacji tekstu	256
Określanie kierunku	257
Podsumowanie	259
<b>7. Podstawowe formatowanie wizualne .....</b>	<b>261</b>
Podstawowe pojemniki	261
Krótka powtórka	262
Blok zawierający element	263
Zmiana sposobu wyświetlania elementu	264
Zamiana ról	265
Elementy blokowe	267
Formatowanie w poziomie	268
Właściwości poziome	270
Wykorzystywanie wartości auto	271
Więcej niż jedna wartość auto	272
Marginesy ujemne	273
Wartości procentowe	275
Elementy zastępowane	276
Formatowanie w pionie	277
Właściwości pionowe	278
Wartości procentowe w pionie	279
Wysokość automatyczna	280
Składanie marginesów w pionie	281
Marginesy ujemne a składanie	283
Pozycje listy	285
Liniiowe elementy wewnętrzne	287
Układ wierszy	287
Podstawowe pojęcia i koncepcje	289

Formatowanie wewnętrzne	291
Niezastępowane elementy liniowe	292
Budowanie pojemników	292
Wyrównanie w pionie	294
Zarządzanie wysokością wiersza	296
Skalowanie wysokości wiersza	298
Dodawanie właściwości pojemników	299
Zmiana sposobu łamania wierszy	302
Glify a obszar zawartości elementu	303
Zastępowane elementy liniowe	303
Dodawanie właściwości pojemnika	304
Elementy zastępowane a linia bazowa	306
Elementy liniowo-blokowe	308
Wartości flow	310
Deklaracja display: contents	311
Inne wartości właściwości display	312
Wartości wyliczone	312
Podsumowanie	313
<b>8. Dopełnienie, obramowanie, kontury i marginesy .....</b>	<b>315</b>
Podstawowe pojemniki elementów	315
Szerokość oraz wysokość	316
Dopełnienie	318
Replikowanie wartości	320
Dopełnienie jednostronne	321
Dopełnienie a wartości procentowe	323
Dopełnienie a elementy liniowe	325
Dopełnienie elementów zastępowanych	326
Obramowanie	327
Obramowanie ze stylem	328
Szerokości obramowania	332
Kolory obramowania	335
Skrótowe właściwości obramowania	337
Obramowanie globalne	339
Obramowanie i elementy liniowe	340
Zaokrąglanie rogów obramowań	341
Obramowania obrazkowe	348
Kontury	364
Style konturów	364
Szerokość konturu	365
Kolor konturu	366
Różnice dotyczące konturów	367



Marginesy	369
Marginesy i wartości długości	370
Marginesy i wartości procentowe	371
Właściwości marginesów jednostronnych	372
Składanie marginesów	372
Marginesy ujemne	374
Marginesy a elementy liniowe	375
Podsumowanie	377
<b>9. Kolory, tła i gradienty .....</b>	<b>379</b>
Kolory	379
Kolory pierwszego planu	379
Oddziaływanie na obramowanie	381
Oddziaływanie na elementy formularzy	382
Dziedziczenie koloru	383
Tło	384
Kolor tła	384
Przycinanie tła	387
Obrazy w tle	390
Położenie tła	394
Zmiana obszaru pozycjonowania tła	403
Powielanie tła (albo brak powielania)	406
Mocowanie	415
Skalowanie obrazów w tle	419
A teraz... wszystko naraz	426
Wiele teł	429
Gradienty	434
Gradienty liniowe	435
Gradienty kołowe	448
Przetwarzanie obrazów gradientów	459
Gradienty cykliczne	461
Cienie pojemników	465
Podsumowanie	468
<b>10. Elementy pływające i kształty .....</b>	<b>469</b>
Pływanie	469
Elementy pływające	470
Pływanie — szczegóły	472
Rzeczywiste zachowanie	478
Pływanie, zawartość i nakładanie się	483
Właściwość clear	484

Kształty elementów pływających	488
Tworzenie kształtu	488
Kształtowanie na podstawie przezroczystości obrazu	499
Dodawanie marginesu kształtu	500
Podsumowanie	502
<b>11. Pozycjonowanie .....</b>	<b>503</b>
Podstawowe koncepcje	503
Typy pozycjonowania	503
Blok zawierający	504
Właściwości przesunięcia	505
Szerokość oraz wysokość	508
Ustawianie szerokości oraz wysokości	508
Ograniczanie szerokości oraz wysokości	509
Wpływanie oraz przycinanie zawartości	511
Wpływanie	511
Widoczność elementu	513
Pozycjonowanie bezwzględne	514
Bloki zawierające elementy a elementy pozycjonowane bezwzględnie	514
Rozmieszczenie i rozmiar elementów pozycjonowanych bezwzględnie	517
Automatyczne krawędzie	518
Rozmieszczenie oraz rozmiar elementów niezastępowanych	520
Rozmieszczenie oraz rozmiar elementów zastępowanych	524
Rozmieszczenie elementów na osi Z	526
Pozycjonowanie typu fixed	530
Pozycjonowanie względne	531
Pozycjonowanie typu sticky	533
Podsumowanie	537
<b>12. Model Flexible Box .....</b>	<b>539</b>
Podstawy modelu flexbox	539
Prosty przykład	541
Pojemniki flex	545
Właściwość flex-direction	545
Inne kierunki pisania	549
Zawijanie linii flex	551
Definiowanie elastycznego układu treści	553
Właściwość flex-wrap — ciąg dalszy	559
Układanie obiektów flex	560
Pojemnik flex	561
Wyrównywanie treści	561
Przykłady działania właściwości justify-content	567

Wyrównywanie obiektów	568
Wyrównanie do początku, do końca i do środka	573
Wyrównanie do linii bazowej	574
Uwagi dodatkowe	575
Właściwość align-self	576
Wyrównywanie treści	577
Wartości space-between, space-around i space-evenly	581
Obiekty flex	582
Czym są obiekty flex?	582
Cechy obiektów flex	584
Szerokości minimalne	585
Właściwości obiektów flex	587
Właściwość flex	587
Właściwość flex-grow	588
Współczynniki wzrostu a właściwość flex	591
Właściwość flex-shrink	594
Proporcjonalne zwężanie obiektów na podstawie ich szerokości oraz współczynnika kurczenia	598
Różne bazy flex	599
Responsywna zmiana wielkości	601
Właściwość flex-basis	604
Słowo kluczowe content	604
Automatyczna baza flex	606
Wartości domyślne	607
Jednostki długości	608
Baza zerowa	612
Skrótowa właściwość flex	613
Typowe wartości właściwości flex	613
Właściwość order	618
Raz jeszcze o nawigacji zakładkowej	620
<b>13. Układ siatkowy .....</b>	<b>623</b>
Tworzenie pojemnika siatki	623
Podstawowa terminologia związana z siatkami	626
Rozmieszczanie linii siatki	628
Tory siatek o stałej szerokości	629
Elastyczne tory siatek	633
Dopasowywanie zawartości torów	640
Powtarzanie linii siatki	642
Obszary siatki	646

Dołączanie obiektów do siatki	652
Zastosowanie linii kolumn i rzędów	652
Skrótowe właściwości rzędów i kolumn	656
Siatka niejawna	659
Obsługa błędów	661
Zastosowanie obszarów	662
Nakładanie się obiektów siatki	665
Przepływ siatki	666
Automatyczne linie siatki	671
Skrótowa właściwość grid	673
Podsiatki	675
Tworzenie odstępów w siatkach	676
Odstępy (przerwy) między torami	676
Obiekty siatki a model pudełkowy	678
Wyrównywanie i siatki	682
Wyrównywanie i justowanie pojedynczych obiektów	683
Wyrównywanie i justowanie wszystkich obiektów	685
Warstwy i kolejność	687
Podsumowanie	690
<b>14. Układ tabelaryczny .....</b>	<b>691</b>
Formatowanie tabel	691
Tworzenie wyglądu tabeli	691
Wartości wyświetlania tabeli	693
Anonimowe obiekty tabeli	697
Warstwy tabeli	701
Podpisy	702
Obramowanie komórek tabeli	704
Oddzielone obramowanie komórek tabeli	704
Składanie obramowania komórek tabeli	707
Rozmiar tabeli	712
Szerokość	712
Wysokość	718
Wyrównanie	719
Podsumowanie	721
<b>15. Listy oraz zawartość generowana .....</b>	<b>723</b>
Listy	723
Typy list	724
Obrazkowe znaki wypunktowania	726
Pozycja znaku wypunktowania listy	729
Style listy w skrócie	730
Układ listy	731

Zawartość generowana	733
Wstawianie zawartości generowanej	734
Określanie zawartości	736
Liczniki	741
Definiowanie wzorców numerowania	747
Stałe wzorce numerowania	749
Cykliczne wzorce numerowania	751
Symboliczne wzorce numerowania	754
Alfabetyczne wzorce numerowania	757
Liczbowe systemy numerowania	758
Addytywne systemy numerowania	761
Rozszerzanie wzorców numerowania	763
Wymawianie wzorców numeracji	764
Podsumowanie	766
<b>16. Przekształcenia .....</b>	<b>767</b>
Układy współrzędnych	767
Przekształcanie	771
Funkcje przekształceń	774
Więcej właściwości przekształceń	788
Przesuwanie punktu początkowego	788
Wybieranie stylu 3D	791
Zmiana perspektywy	794
Tylne ścianki	797
Podsumowanie	799
<b>17. Przejścia .....</b>	<b>801</b>
Przejścia CSS	801
Właściwości przejść	802
Ograniczanie rodzaju przejść do konkretnych właściwości	806
Ustalanie czasu trwania przejścia	812
Zmiana tempa przejść	814
Opóźnianie przejść	819
Skrótowa właściwość transition	822
Na odwrót: przejście do początku	824
Animowane właściwości i wartości	828
Na czym polega interpolacja wartości właściwości?	829
Wyjścia awaryjne: przejścia to tylko ozdobniki	832
Drukowanie przejść	832

<b>18. Animacje .....</b>	<b>835</b>
Definiowanie klatek kluczowych	836
Konfigurowanie animacji na klatkach kluczowych	837
Nadawanie nazwy animacji	837
Selektory klatek kluczowych	838
Pomijanie wartości from i to	839
Powtarzanie właściwości klatek kluczowych	840
Właściwości dające się animować	841
Nieanimowane właściwości, które nie są ignorowane	842
Zastosowanie skryptów w animacjach @keyframes	842
Animowanie elementów	843
Nazywanie animacji	844
Definiowanie długości animacji	846
Deklarowanie iteracji animacji	847
Ustalanie kierunku animacji	849
Opóźnianie animacji	850
Zdarzenia związane z animacjami	852
Zmiana wewnętrznego tempa animacji	860
Ustawianie stanu odtwarzania animacji	871
Tryby uzupełniania animacji	872
I wszystko razem...	874
Animacje, specyficzność i kolejność	877
Specyficzność i dyrektywa !important	877
Kolejność animacji	878
Iterowanie animacji i reguła display: none	878
Animacja i wątek UI	879
Epilepsja i zaburzenia przedsionkowe	879
Zdarzenia animacji a prefiksy	880
Zdarzenie animationstart	880
Zdarzenie animationend	881
Zdarzenie animationiteration	881
Drukowanie animacji	881
<b>19. Filtry, mieszanie, przycinanie i maskowanie .....</b>	<b>883</b>
Filtry CSS	883
Filtry podstawowe	884
Filtrowanie kolorów	886
Jasność, kontrast i nasycenie	887
Filtry SVG	888
Nakładanie i mieszanie	889
Mieszanie elementów	890
Przyciemnianie, rozjaśnianie, różnica i wykluczanie	891

Mnożenie, ekran i nakładka	892
Ostre i miękkie światło	893
Rozjaśnianie i ściemnianie	894
Barwa, nasycenie, jasność i kolor	895
Mieszanie tła	896
Mieszanie w izolacji	899
Przycinanie i maskowanie	900
Przycinanie	900
Kształty przycinające	902
Pojemniki przycinające	902
Reguły wypełniania kształtów przycinających	905
Maski	906
Definiowanie maski	907
Zmiana trybu działania maski	909
Skalowanie i powtarzanie masek	911
Pozycjonowanie masek	912
Przycinanie i łączenie masek	914
Zbierzmy wszystko w całość...	917
Rodzaje masek	919
Maskowanie w obramowaniach obrazkowych	919
Dopasowywanie i pozycjonowanie obiektu	920
<b>20. Style zależne od medium .....</b>	<b>925</b>
Definiowanie stylów zależnych od medium	925
Podstawowe zapytania o media	925
Złożone zapytania o media	927
Media stronicowe	934
Style wydruków	935
Podsumowanie	948
<b>A Właściwości animowane .....</b>	<b>949</b>
<b>B Zestawienie właściwości .....</b>	<b>957</b>
<b>C Tabela odpowiedników kolorów .....</b>	<b>967</b>
<b>Skorowidz .....</b>	<b>973</b>





Przejścia CSS umożliwiają animowanie właściwości CSS — czyli płynną zmianę od wartości początkowej do nowej wartości docelowej z upływem czasu. Powodują one zmianę jednego stanu elementu na drugi w reakcji na jakąś zmianę — zwykle wynikającą z działania użytkownika, ale także z zaprogramowanych w skrypcie zmian klas, identyfikatorów albo innego rodzaju stanów.

W normalnej sytuacji każda zmiana wartości właściwości CSS — chwila, gdy nastąpi „zdarzenie zmiany stylów” — zachodzi natychmiastowo. Nowa wartość właściwości zastępuje starą w ciągu kilku milisekund, jakie zajmuje ponowne narysowanie treści, której dotyczy ta zmiana (albo narysowanie i zmiana układu dokumentu, jeśli zajdzie taka konieczność). Większość zmian wartości wydaje się natychmiastowa, bo ich wyświetlenie trwa poniżej 16 milisekund<sup>1</sup>. Nawet jeśli zmiana trwa dłużej, wciąż jest to tylko jeden krok — przejście od jednej wartości do drugiej. Na przykład zmiana koloru tła po wskazaniu danego elementu kursorem myszy jest bezzwłoczna, bez płynnego przejścia między barwami.

## Przejścia CSS

Przejścia (ang. *transitions*) CSS umożliwiają sterowanie sposobem zmiany jednej wartości właściwości na inną w ciągu określonego czasu. Dzięki temu możemy uzyskać płynną zmianę wartości, dającą przyjemny dla oka i (miejmy nadzieję) nieprzeszkadzający w odbiorze treści efekt. Weźmy następujący przykład:

```
button {
  color: magenta;
  transition: color 200ms ease-in 50ms;
}

button:hover {
  color: rebeccapurple;
  transition: color 200ms ease-out 50ms;
}
```

W tym przykładzie, zamiast natychmiastowej zmiany wartości `color` po wskazaniu przycisku kursorem myszy, za pomocą przejść CSS możemy uzyskać płynną zmianę koloru z magenta na rebeccapurple w ciągu 200 milisekund, z trwającym 50 milisekund opóźnieniem przed rozpoczęciem przejścia.

---

<sup>1</sup> Zmiana obrazu w tle, wymagająca zdekodowania tego obrazu i wyświetlenia go, może trwać dłużej niż 16 milisekund. To nie jest jednak kwestia samego przejścia, ale niewielkiej wydajności.

Zmiana koloru, bez względu na to, jak długo albo jak krótko trwa, jest pewną formą przejścia. Jednak dzięki zastosowaniu właściwości CSS o nazwie `transition` zmiana koloru może nastąpić stopniowo, w ciągu pewnego czasu, a jej płynność będzie łatwo dostrzegalna dla oka.

Przejściami CSS możesz się posłużyć już dziś, nawet jeśli wciąż dbasz o użytkowników przeglądarek IE9 albo starszych. Jeśli jakaś przeglądarka nie obsługuje właściwości przejść CSS, to zmiana po prostu nastąpi w jednej chwili, a nie płynnie, co jest zupełnie akceptowalne. Poza tym do raptownej (a nie płynnej) zmiany dojdzie także wtedy, gdy jakaś właściwość albo pewne jej wartości nie dają się animować.



Mamy tu na myśli dowolne właściwości, które dają się animować — czy to za pośrednictwem przejść, czy też animacji (będących tematem następnego rozdziału, „Animacje”). Podsumowanie informacji na ten temat znajdziesz w dodatku A.

Czasami zależy nam na natychmiastowej zmianie jakiejś wartości. Choć w poprzedniej części rozdziału posłużyliśmy się przykładem odwołującym się do kolorów odsyłaczy, to kolory te na ogół zmieniają się w sposób natychmiastowy po wskazaniu odsyłacza kursorem myszy, informując osoby widzące o możliwości interakcji oraz o tym, że wskazany fragment treści jest hiperłączem. Na podobnej zasadzie opcje na liście umożliwiającej automatyczne uzupełnianie wpisywanej treści nie powinny się pojawiać stopniowo: chodzi raczej o to, by były wyświetlane od razu, zamiast pojawiać się wolniej, niż pisze użytkownik. Z punktu widzenia wygody użytkownika natychmiastowa zmiana wartości często jest najlepsza.

Kiedy indziej z kolei może nam zależeć na tym, by zmiana wartości właściwości następowała bardziej stopniowo i zwracała uwagę na to, co się dzieje. Możemy na przykład nadać grze w karty pozory większego realizmu, wprowadzając animację odwracania karty trwającą 200 milisekund, bo w przypadku braku animacji użytkownik może nawet nie zauważyć, co się zmieniło. ▶

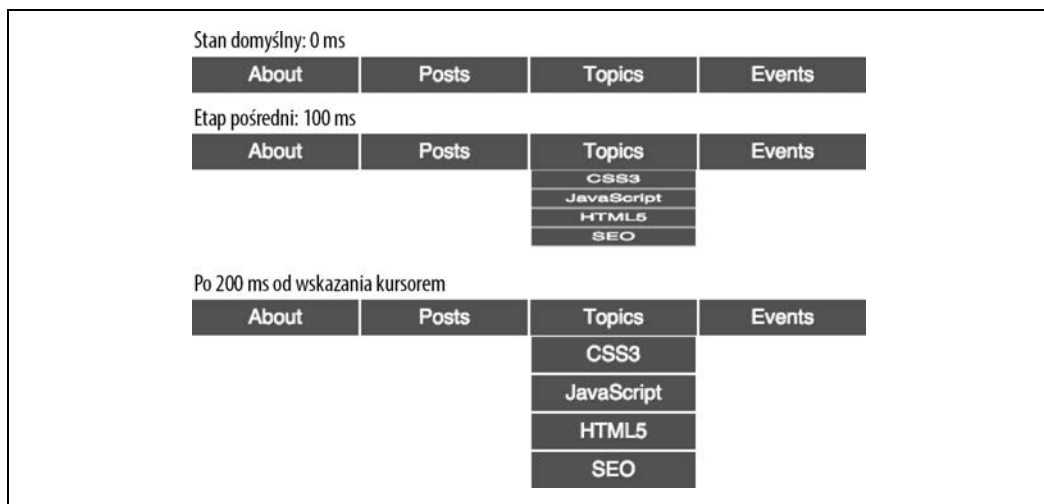


Zwracaj uwagę na symbol odtwarzania ▶, który informuje o możliwości obejrzenia przykładu online. Wszystkie przykłady opisane w tym rozdziale są dostępne pod adresem <https://meyerweb.github.io/csstdg4figs/17-transitions/>.

Weźmy inny przykład: może zależeć Ci na tym, by niektóre rozwijane menu pojawiały się albo rozwijały w ciągu 200 milisekund (a nie natychmiast, co może być irytujące). Dzięki przejściom rozwijane menu mogą się wyświetlać powoli. Na rysunku 17.1 ▶ pokazane są etapy przejścia polegającego na skalowaniu wysokości jednego z podmenu. Jest to jedno z typowych zastosowań przejść CSS, które omówimy w dalszej części tego rozdziału.

## Właściwości przejść

W CSS przejścia tworzy się za pomocą czterech właściwości przejść: `transition-property`, `transition-duration`, `transition-timing-function` oraz `transition-delay`; istnieje też skrótowa właściwość `transition` obejmująca te cztery właściwości składowe.



Rysunek 17.1. Przejście — etap początkowy, pośredni i końcowy

Do utworzenia rozwijanej nawigacji pokazanej na rysunku 17.1 użyliśmy wszystkich czterech właściwości przejść CSS, a oprócz tego także właściwości niezwiązanych z przejściami, definiujących początkowy i końcowy stan przejścia. Przejście przedstawione na rysunku 17.1 można zdefiniować za pomocą poniższego kodu:

```
nav li ul {
  transition-property: transform;
  transition-duration: 200ms;
  transition-timing-function: ease-in;
  transition-delay: 50ms;
  transform: scale(1, 0);
  transform-origin: top center;
}
nav li:hover ul {
  transform: scale(1, 1);
}
```

Zauważ, że choć w naszych przykładach przejść używamy stanu `:hover` jako zdarzenia wyzwalającego zmianę stylów, to przejścia z użyciem właściwości można stosować także w innych sytuacjach. Możesz na przykład dodać albo usunąć klasę bądź w inny sposób zmienić stan elementu — na przykład poprzez zmianę pola formularza z `:invalid` na `:valid` albo z `:checked` na `:not(:checked)`. Możesz też przy użyciu selektorów `:nth-last-of-type` dodać wiersz do tabeli z naprzemiennie pokolorowanymi wierszami („zebra”) albo dołożyć kolejny punkt na końcu jakiejś listy.

W scenariuszu pokazanym na rysunku 17.1 początkowy stan zagnieżdżonych list to `transform: scale(1, 0)` i `transform-origin: top center`. Stan końcowy to `transform: scale(1, 1)` — właściwość `transform-origin` pozostaje bez zmian.



Więcej informacji o właściwościach przekształceń (`transform`) znajdziesz w rozdziale 16.

W tym przykładzie przejście jest zdefiniowane przez właściwość `transform`: po wskazaniu menu kursorem myszy (`:hover`) zagnieżdżona, nienumerowana lista skaluje się do pierwotnego, domyślnego rozmiaru — w ciągu 200 milisekund następuje płynne przejście między starą wartością `transform: scale(1, 0)` a nową wartością `transform: scale(1, 1)`. Przejście rozpoczyna się z 50-milisekundowym opóźnieniem i przyspiesza (odpowiada za to funkcja `ease-in`) — zaczyna się wolno, a potem nabiera tempa.

Przejścia są deklarowane oprócz zwykłych stylów dla elementu. Ilekroć zmienia się właściwość, dla której zdefiniowano przejście, przeglądarka zastosuje to przejście, aby zmiana właściwości odbyła się stopniowo.

Zauważ, że wszystkie właściwości przejść zostały ustalone dla nieaktywnego stanu elementów `ul`. Stan aktywny (po wskazaniu kursorem myszy) został użyty tylko w celu zastosowania przekształcenia, a nie samych parametrów przejścia. I nie bez powodu: dzięki temu bowiem menu nie tylko płynnie otworzy się po wskazaniu kursorem, ale też płynnie zamknie, gdy przestaniemy je wskazywać.

Wyobraźmy sobie, że właściwości przejścia byłyby zamiast tego stosowane za pośrednictwem stanu `:hover`, na przykład tak:

```
nav li ul {
  transform: scale(1, 0);
  transform-origin: top center;
}
nav li:hover ul {
  transition-property: transform;
  transition-duration: 200ms;
  transition-timing-function: ease-in;
  transition-delay: 50ms;
  transform: scale(1, 1);
}
```

Oznaczałoby to, że element, który *nie jest* wskazany kursorem myszy, będzie miał domyślne właściwości przejść — czyli przejścia będą się odbywały natychmiastowo. Menu z poprzedniego przykładu płynnie otwierałoby się po wskazaniu kursorem myszy, ale po zakończeniu stanu `:hover` natychmiast by znikało — bo bez stanu `:hover` właściwości przejść byłyby nieaktywne!

Może się okazać, że właśnie na takim efekcie Ci zależy: płynne wysunięcie elementu, a potem raptowne schowanie go. Jeśli tak, to przypisz przejścia do stanu `:hover`. W przeciwnym razie przypisz je bezpośrednio do elementu, aby miały one zastosowanie zarówno przy rozpoczęciu, jak i przy kończeniu wskazywania elementu kursorem myszy. Zakończenie danego stanu powoduje odwrócenie kierunku przejścia. To domyślne zachowanie możesz zmienić, deklarując różne rodzaje przejść dla stanu początkowego i stanu zmienionego.

Przez „stan początkowy” rozumiemy stan elementu po załadowaniu strony. Może to być stan, który jest dla danego elementu trwały — na przykład określony przez właściwości zdefiniowane za pośrednictwem zwykłego selektora elementu, a nie za pośrednictwem selektora `:hover` dla tego elementu. Może to także być element o edytowalnej treści, który zyskuje stan `:focus`, jak w następującym przykładzie: ▶

```
/* selektor, który zawsze pasuje do danych elementów */
p[contenteditable] {
  background-color: rgba(0, 0, 0, 0);
}
```

```

/* selektor, który pasuje do danych elementów tylko czasami */
p[contenteditable]:focus {
    /* zastępowanie deklaracji */
    background-color: rgba(0, 0, 0, 0.1);
}

```

W tym przykładzie stanem początkowym elementu jest całkowicie przezroczyste tło, które zmienia się tylko wtedy, gdy użytkownik uaktywni (stan `:focus`) ten element. Właśnie to mamy na myśli, mówiąc w tym rozdziale o stanie *początkowym* albo *domyślnym* elementu. Właściwości przejścia podane w selektorze, który pasuje do elementu przez cały czas, będą miały wpływ na ten element przy każdej zmianie jego stanu — na przykład zmianie ze stanu początkowego na docelowy (w poprzednim przykładzie docelowym był stan `:focus`).

Stan początkowy może też mieć charakter przejściowy — na przykład `:checked` dla pola wyboru albo `:valid` dla kontrolki formularza. Może to być nawet klasa, którą da się włączać i wyłączać.

```

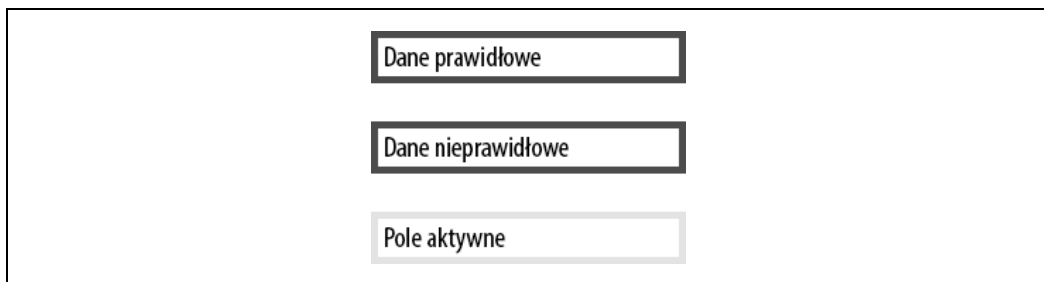
/* selektor, który pasuje do danych elementów tylko czasami */
input:valid {
    border-color: green;
}

/* selektor, który pasuje do danych elementów tylko czasami,
a konkretnie wtedy, gdy poprzedni selektor NIE pasuje */
input:invalid {
    border-color: red;
}

/* selektor, który pasuje do danych elementów tylko czasami,
w zależności od prawidłowości danych */
input:focus {
    /* alternatywna deklaracja */
    border-color: yellow;
}

```

W podanym przykładzie do dowolnego elementu może pasować albo selektor `:valid`, albo `:invalid`, ale nie oba naraz. Selektor `:focus`, zgodnie z tym, co zostało pokazane na rysunku 17.2, wybiera element wtedy, gdy pole wprowadzania danych jest aktywne, bez względu na to, czy pole to jest też jednocześnie wybrane przez selektory `:valid` albo `:invalid`.




Rysunek 17.2. Wygląd pola wprowadzania danych w różnych stanach — dane prawidłowe, dane nieprawidłowe, pole aktywne

W takim przypadku, jeśli odwołujemy się do stanu początkowego, mamy na myśli wartość początkową, którą może być zarówno `:valid`, jak i `:invalid`. Zmieniony stan danego elementu jest przeciwieństwem początkowego stanu `:valid` lub `:invalid`. ▶

Należy pamiętać, że da się zastosować różne wartości przejść dla stanu początkowego i zmienionego, zawsze jednak należy zadbać o wprowadzenie wartości związanej z *wejściem* w dany stan. Weźmy na przykład poniższy kod, w którym przejścia zostały skonfigurowane tak, by menu otwierały się przez 2 sekundy, ale zamykały już po 200 milisekundach.

```
nav li ul {
  transition-property: transform;
  transition-duration: 200ms;
  transition-timing-function: ease-in;
  transition-delay: 50ms;
  transform: scale(1, 0);
  transform-origin: top center;
}
nav li:hover ul {
  transition-property: transform;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
  transform: scale(1, 1);
}
```

Z perspektywy funkcjonalności efekt jest straszny, ale dobrze ilustruje omawianą ideę.  Po wskazaniu menu kursorem myszy proces otwierania się tego menu zajmuje bite 2 sekundy. Zamykanie przebiega znacznie szybciej, bo w 0,2 sekundy. Właściwości przejścia w zmienionym stanie (:hover) dochodzą do głosu wtedy, gdy wskazujemy pozycję listy kursorem myszy. Wtedy zaczyna działać reguła `transition-duration: 2s`, zdefiniowana dla stanu :hover. Po odsunięciu kursora menu wraca do domyślnego, zwiniętego stanu i wykorzystywane są właściwości przejścia zdefiniowane dla stanu początkowego (opisanego regułą `nav li ul`), które powodują trwające 200 milisekund zamknięcie menu.

Przyjrzyjmy się nieco bliżej temu przykładowi, zwłaszcza domyślnym stylom przejść. Kiedy użytkownik przestaje wskazywać kursorem myszy nadrzędny element głównej nawigacji albo element potomny w postaci rozwijanego menu, następuje 50-milisekundowa pauza, poprzedzająca trwające 200 milisekund przejście, które zamyka menu. Akurat ten aspekt funkcjonalności menu jest wygodny, bo daje użytkownikowi szansę (choć krótką) na przesunięcie kursora z powrotem nad menu, zanim znacznie się ono zamykać.

Choć każdą z czterech właściwości przejść można zadeklarować osobno, to zapewne zawsze będziesz się posługiwał skrótem. Najpierw przyjrzymy się czterem właściwościom składowym, abyś mógł lepiej zrozumieć działanie każdej z nich.

## Ograniczanie rodzaju przejść do konkretnych właściwości

Właściwość `transition-property` pozwala określić nazwy właściwości CSS, które chcesz poddać działaniu przejść. Umożliwia to ograniczenie rodzaju przejść do konkretnych właściwości, podczas gdy inne będą się zmieniały w sposób natychmiastowy. (I owszem, w dalszej części rozdziału słowo „właściwości” będzie się pojawiało bardzo wiele razy).

## transition-property

<b>Wartości:</b>	none   [ all   <property-name> ]#
<b>Wartość początkowa:</b>	all
<b>Stosuje się do:</b>	Wszystkich elementów oraz pseudoelementów :before i :after
<b>Wartość wyliczona:</b>	Jak określono
<b>Dziedziczona:</b>	Nie
<b>Animowana:</b>	Nie

Właściwość `transition-property` przyjmuje wartości w postaci listy rozdzielonych przecinkami właściwości, słowa kluczowego `none` (jeśli nie chcesz dopuścić do przejść jakichkolwiek właściwości) albo domyślnego słowa `all` (które oznacza: „dopuszczaj wszystkie animowane właściwości”). Słowo `all` może też wystąpić na liście oprócz właściwości rozdzielonych przecinkami.

Jeśli podasz `all` jako jedyne słowo kluczowe — albo pozwolisz, by właściwość `transition-property` domyślnie przyjęła wartość `all` — wszystkie animowane właściwości zostaną poddane równoczesnemu przejściu. Powiedzmy, że chcesz zmienić wygląd jakiegoś pojemnika po wskazaniu go kursorem myszy:

```
div {
  color: #ff0000;
  border: 1px solid #00ff00;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
}
div:hover {
  color: #000000;
  border: 5px dashed #000000;
  border-radius: 50%;
  transform: scale(2) rotate(-10deg);
  opacity: 0.5;
  box-shadow: -3px -3px rgba(255, 0, 0, 0.5);
  width: 100px;
  padding: 20px;
}
```

Po wskazaniu kursorem elementu `div` wszystkie właściwości, które mają inne wartości w stanie początkowym niż w stanie `:hover` (zmienionym), ulegną zmianie i przyjmą wartości zgodne ze stanem `:hover`. Właściwość `transition-property` służy do określania, które z tych właściwości będą płynnie animowane przez pewien czas (zamiast raptownie się zmienić). Po wskazaniu elementu kursorem myszy wszystkie właściwości zmieniają się z wartości domyślnych na wartości `:hover`, ale tylko animowane właściwości zadeklarowane we właściwości `transition-property` zmieniają się w sposób płynny, w ciągu całego czasu trwania przejścia. Niedające się animować właściwości, takie jak `border-style`, zmieniają swoją wartość w sposób natychmiastowy.

Jeśli jedyną (albo ostatnią na liście wartości rozdzielonych przecinkami) wartością właściwości `transition-property` jest `all`, to wszystkie animowane właściwości dokonają równoczesnego przejścia. Aby uzyskać inny efekt, należy podać rozdzieloną przecinkami listę właściwości, które mają podlegać wpływowi przejść.

Jeśli więc chcemy dopuścić do efektu przejścia dla wszystkich właściwości, to poniższe deklaracje będą niemal równoważne:

```
div {
  color: #ff0000;
  border: 1px solid #00ff00;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
  transition-property: color, border, border-radius, transform, opacity,
  ↪box-shadow, width, padding;
  transition-duration: 1s;
}

div {
  color: #ff0000;
  border: 1px solid #00ff00;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
  transition-property: all;
  transition-duration: 1s;
}
```

Obie powyższe deklaracje `transition-property` zezwalają na animowane przejścia we wszystkich wymienionych właściwościach, ale dla pierwszej deklaracji oznacza to tylko osiem właściwości. Należy pamiętać, że reguły dla tych właściwości mogą też występować w innych blokach reguł. Innymi słowy, akurat w tym przypadku wszystkie osiem animowanych właściwości zostało wymienionych w tym samym bloku, w którym znajduje się deklaracja `transition-property`, ale nie musi tak być.

Deklaracja `transition-property: all` w drugim bloku reguł gwarantuje, że *wszystkie* animowane właściwości zmieniają swoją wartość w przypadku zajścia zdarzenia, które wywoła zmianę stylu — bez względu na to, w którym bloku reguł CSS wystąpi właściwość o zmienionej wartości — i zmiana ta będzie trwała dokładnie sekundę. Przejście będzie miało zastosowanie do wszystkich animowanych właściwości wszystkich elementów pasujących do selektora, a nie tylko tych właściwości, które zostały zadeklarowane w tym samym bloku stylów, w którym użyto słowa kluczowego `all`.

W omawianym przypadku pierwsza wersja deklaracji ogranicza rodzaj przejść do ośmiu wymienionych właściwości, ale daje nieco większą kontrolę nad tym, jak te przejścia będą w ich przypadku wyglądały. Zadeklarowanie osobnych właściwości pozwala na określenie różnych prędkości, opóźnień i (lub) czasów trwania przejścia dla każdej z nich, jeśli także właściwości poszczególnych przejść zdefiniujemy osobno:



```

div {
  color: #ff0000;
  border: 1px solid #0f0;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
}

.foo {
  color: #00ff00;
  transition-property: color, border, border-radius, transform, opacity,
    box-shadow, width, padding;
  transition-duration: 1s;
}
<div class="foo">Hello</div>

```

Jeśli chcesz zdefiniować przejścia dla każdej właściwości osobno, wymień je wszystkie i rozdziel poszczególne właściwości przecinkiem. Jeżeli chcesz zastosować animację niemal wszystkich właściwości (powiedzmy, z kilkoma wyjątkami), która nastąpi jednocześnie, z takim samym opóźnieniem i w tym samym tempie, to możesz użyć kombinacji słowa kluczowego `all` oraz nazw tych właściwości, które mają być animowane w innym czasie czy z inną prędkością. Pamiętaj, aby wówczas podać `all` jako pierwszą wartość:

```

div {
  color: #f00;
  border: 1px solid #00ff00;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
  transition-property: all, border-radius, opacity;
  transition-duration: 1s, 2s, 3s;
}

```

Słowo kluczowe `all` występujące na liście właściwości rozdzielonych przecinkami obejmuje wszystkie właściwości podane w powyższym przykładzie, w tym także odziedziczone właściwości CSS oraz wszystkie właściwości zdefiniowane w dowolnym innym bloku reguł CSS, który pasuje do danego elementu bądź jest przez ten element dziedziczony.

W poprzednim przykładzie wszystkie właściwości, którym nadane zostaną nowe wartości, zostaną poddane przejściu o takim samym czasie trwania, opóźnieniu i tempie — z wyjątkiem właściwości `border-radius` oraz `opacity`, które w sposób jawny wymieniliśmy osobno. Ponieważ umieściliśmy je na rozdzielonej przecinkami liście wartości, po wartości `all`, możemy albo poddać je przejściu jednocześnie, z tym samym opóźnieniem i tempem, albo podać dla nich dwóch inne wartości czasu, opóźnień i tempa. W tym konkretnym przypadku wszystkie właściwości będą trwały 1 sekundę, oprócz właściwości `border-radius` i `opacity`, których przejście będzie trwało, odpowiednio, 2 i 3 sekundy. (Właściwość `transition-duration` została omówiona w dalszej części tego rozdziału).



Pamiętaj o wymienieniu słowa kluczowego `all` na początku listy wartości rozdzielonej przecinkami, bo właściwości zadeklarowane przed słowem `all` będą podlegały tym samym regułom co `all`. W rezultacie parametry przejść tych właściwości, które zamierzałeś podać osobno, zostaną nadpisane parametrami dotyczącymi zbioru właściwości opisanego słowem kluczowym `all`.

## Wyłączanie przejść poprzez ograniczanie puli właściwości

Choć przejścia inne niż natychmiastowe są domyślnie wyłączone, to jeśli zdefiniujesz jakieś przejście CSS i będziesz chciał je wyłączyć w jakiejś konkretnej sytuacji, możesz w tym celu użyć deklaracji `transition-property: none`. Dzięki temu zyskasz gwarancję wyłączenia przejść dla wszystkich właściwości.

Słowo kluczowe `none` może być użyte tylko jako pojedyncza wartość tej właściwości — nie może być jedną z rozdzielonych przecinkami wartości na liście. Jeśli chcesz wyłączyć przejścia dla pewnego ograniczonego zbioru właściwości, musisz postąpić niejako na odwrót i wymienić wszystkie właściwości, których przejścia nadal chcesz zostawić włączone. Właściwości `transition-property` nie da się użyć do wykluczania właściwości; służy ona raczej do uwzględniania ich.



Inna metoda polega na ustawieniu dla danej właściwości opóźnienia i czasu trwania przejścia na `0s`. Dzięki temu zmiany wartości właściwości będą zachodzić natychmiastowo, tak jakby nie było dla nich zdefiniowanego przejścia.

## Zdarzenia związane z przejściami

W modelu DOM na końcu każdego przejścia — w obu kierunkach i dla każdej właściwości, która podlega przejściu w ciągu dowolnej ilości czasu *albo* po dowolnym opóźnieniu — wyzwalane jest zdarzenie `transitionend`. Dzieje się tak bez względu na to, czy dana właściwość została zadeklarowana osobno, czy stanowi część deklaracji związanej ze słowem kluczowym `all`. W przypadku niektórych na pozór pojedynczych deklaracji właściwości może zajść kilka zdarzeń `transitionend`, bo każda animowana składowa w ramach właściwości skrótowej wywoła własne zdarzenie tego typu. Rozważmy następujący przykład:

```
div {
  color: #f00;
  border: 1px solid #00ff00;
  border-radius: 0;
  transform: scale(1) rotate(0deg);
  opacity: 1;
  box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
  width: 50px;
  padding: 100px;
  transition-property: all, border-radius, opacity;
  transition-duration: 1s, 2s, 3s;
}
```

Po zakończeniu przejść dojdzie do co najmniej ośmiu zdarzeń `transitionend`. Na przykład przejście samej tylko właściwości `border-radius` generuje cztery zdarzenia `transitionend`, po jednym dla:

- border-bottom-left-radius,
- border-bottom-right-radius,
- border-top-right-radius,
- border-top-left-radius.

Właściwość padding też jest wszak skrótem dla czterech pełnych właściwości:


- padding-top,
- padding-right,
- padding-bottom,
- padding-left.

Skrótowa właściwość border generuje osiem zdarzeń transitionend: czterokrotnie dla czterech właściwości odpowiadających skróтови border-width i czterokrotnie dla właściwości reprezentowanych przez border-color:

- border-left-width,
- border-right-width,
- border-top-width,
- border-bottom-width,
- border-top-color,
- border-left-color,
- border-right-color,
- border-bottom-color.

Zdarzenia transitionend nie zajmą jednak dla właściwości border-style, bo nie jest ona właściwością animowaną.

Skąd wiemy, że właściwość border-style nie jest animowana? Możemy tak założyć, bo nie ma sensownego punktu pośredniego między jej dwiema wartościami, takimi jak solid i dashed. Możemy to sprawdzić na liście animowanych właściwości w dodatku A albo w specyfikacji każdej z nich z osobna.

W naszym scenariuszu z wymienionymi ośmioma konkretnymi właściwościami zajdzie 21 zdarzeń transitionend, bo wśród tych ośmiu są właściwości skrótowe, które mają różne wartości w stanach przed przejściem i po nim. W przypadku słowa kluczowego all będziemy mieli do czynienia z co najmniej 21 zdarzeniami transitionend: po jednym dla każdej ze składowych 8 właściwości, które mają zdefiniowane stany „przed i po” oraz potencjalnie inne, wynikające z dziedziczenia albo deklaracji właściwości w innych blokach stylów mających wpływ na animowany element. 

Zdarzeń transitionend można nasłuchiwać w następujący sposób:

```
document.querySelector('div').addEventListener('transitionend',
  function (e) {
    console.log(e.propertyName);
  });
```

Zdarzenie `transitionend` zawiera trzy atrybuty, które to zdarzenie opisują:

1. `propertyName` — jest to nazwa właściwości CSS, której przejście właśnie się zakończyło.
2. `pseudoElement` — jest to poprzedzony dwoma dwukropkami pseudoelement, którego dotyczyło przejście. Może to być pusty łańcuch znaków, jeśli przejście dotyczyło zwykłego węzła DOM.
3. `elapsedTime` — jest to ilość czasu, jaką zajęło przejście (w sekundach). Zwykle jest to czas podany jako wartość właściwości `transition-duration`.

Zdarzenie `transitionend` zachodzi tylko wtedy, gdy zmiana wartości właściwości w ramach przejścia zakończy się powodzeniem. Zdarzenie `transitionend` nie zachodzi, jeśli przejście zostało przerwane — na przykład przez kolejną zmianę tej samej właściwości tego samego elementu.

Przywrócenie wartości początkowej właściwości powoduje zajście kolejnego zdarzenia `transitionend`. To drugie zdarzenie zachodzi zawsze, jeśli tylko przejście się rozpocznie — nawet jeśli nie zostanie ono dokończone w pierwotnym kierunku.

## Ustalanie czasu trwania przejścia

Właściwość `transition-duration` przyjmuje wartość w postaci rozdzielonej przecinkami listy wartości określających interwały czasowe, w sekundach (s) albo w milisekundach (ms). Wartości te opisują czas trwania przejścia z jednego stanu do drugiego.

<b>transition-duration</b>	
<b>Wartości:</b>	<code>&lt;time&gt;#</code>
<b>Wartość początkowa:</b>	0s
<b>Stosuje się do:</b>	Wszystkich elementów oraz pseudoelementów <code>:before</code> i <code>:after</code>
<b>Wartość wyliczona:</b>	Jak określono
<b>Dziedziczona:</b>	Nie
<b>Animowana:</b>	Nie

Jeśli mamy do czynienia z przejściem w tę i z powrotem między dwoma stanami, a w deklaracji został podany czas trwania dotyczący tylko jednego z tych stanów, to ów czas będzie rzutował tylko na przejście *do* tego stanu. Rozważmy:

```
input:invalid {
  transition-duration: 1s;
  background-color: red;
}

input:valid {
  transition-duration: 0.2s;
  background-color: green;
}
```

Jeśli dla właściwości `transition-duration` zadeklarowane zostaną różne wartości, to czas trwania danego przejścia będzie odpowiadał wartości `transition-duration` zadeklarowanej w bloku reguł opisującym docelowe właściwości tego przejścia. W poprzednim przykładzie zmiana koloru tła pola

wprowadzania danych na czerwone, jeśli w polu tym zostanie wprowadzona nieprawidłowa wartość, potrwa 1 sekundę. Przejście do zielonego tła po wprowadzeniu prawidłowej wartości będzie zaś trwało tylko 200 milisekund. ▶

Właściwość `transition-duration` przyjmuje wartości dodatnie w sekundach (s) albo w milisekundach (ms). Jednostka czasu — ms albo s — jest wymagana w specyfikacji nawet w przypadku czasu ustawionego na 0s. Domyślnie zmiana właściwości z jednej wartości na drugą zachodzi błyskawicznie, bez widocznej animacji — właśnie dlatego domyślna wartość czasu trwania przejścia wynosi 0s.

Jeśli pominięto deklarację właściwości `transition-duration`, to (chyba że właściwość `transition-delay` ma wartość dodatnią) efekt jest taki, jakby właściwość `transition-property` też nie została zadeklarowana — nie zachodzi wtedy także zdarzenie `transitionend`. Dopóki łączny czas przejścia jest większy niż zero sekund (czas zerowy może wynikać z jawnej deklaracji 0s albo z pominięcia właściwości `transition-duration`, co skutkuje nadaniem jej domyślnej wartości 0s), przejście nastąpi, a po jego zakończeniu zajdzie zdarzenie `transitionend`.

Ujemne wartości właściwości `transition-duration` są nieprawidłowe, a jeśli wartość taka zostanie podana, unieważni całą deklarację właściwości.

Korzystając z tej samej co wcześniej, bardzo długiej formy deklaracji `transition-property`, możemy określić wspólny czas trwania dla wszystkich właściwości bądź osobne czasy trwania dla każdej z nich. Możemy też zdecydować, by na przykład co druga z podanych na liście właściwości była animowana przez taki sam czas. Jeśli chcemy zadeklarować pojedynczy czas trwania dla wszystkich właściwości biorących udział w przejściu, wystarczy, że podamy jedną wartość `transition-duration`:

```
div {
  color: #ff0000;
  ...
  transition-property: color, border, border-radius, transform, opacity,
  ↪box-shadow, width, padding;
  transition-duration: 200ms;
}
```

Dla właściwości `transition-duration` możemy też podać tyle samo rozdzielonych przecinkami wartości czasu, ile właściwości CSS zostało wymienionych w charakterze wartości właściwości `transition-property`. Jeśli chcemy, aby przejście dla każdej z zadeklarowanych właściwości zajmowało różny czas, musimy dla każdej z osobna podać wartość czasu na liście:

```
div {
  color: #ff0000;
  ...
  transition-property: color, border, border-radius, transform, opacity,
  ↪box-shadow, width, padding;
  transition-duration: 200ms, 180ms, 160ms, 140ms, 120ms, 100ms, 1s, 2s;
}
```

Jeśli liczba zadeklarowanych właściwości jest inna niż liczba zadeklarowanych czasów trwania, przeglądarka radzi sobie z tą różnicą, kierując się ściśle określonymi zasadami. Otóż jeżeli czasów trwania jest więcej niż właściwości, zbędne czasy trwania są ignorowane. Jeśli z kolei właściwości jest więcej niż czasów, wartości czasów są powtarzane od początku. W poniższym przykładzie przejścia właściwości `color`, `border-radius`, `opacity` i `width` będą trwały po 100 milisekund, zaś przejścia właściwości `border`, `transform`, `box-shadow` i `padding` będą trwały po 200 milisekund.

```
div {
  ...
  transition-property: color, border, border-radius, transform, opacity,
  ↳box-shadow, width, padding;
  transition-duration: 100ms, 200ms;
}
```

Jeśli zadeklarujemy dwa czasy trwania oddzielone przecinkiem, każda nieparzysta właściwość będzie animowana zgodnie z pierwszą podaną wartością czasu, a każda parzysta zgodnie z drugą.

Należy pamiętać o wygodzie obsługi. Jeśli przejście jest zbyt wolne, strona internetowa będzie sprawiała wrażenie ociężałej albo reagującej z opóźnieniem i skupi uwagę użytkownika na czymś, co powinno być jedynie subtelnym efektem. Jeśli z kolei przejście jest za szybkie, może być trudne do dostrzeżenia. Choć da się podać dowolną dodatnią wartość czasu trwania przejść, to Twoim celem powinno być raczej uatrakcyjnienie odbioru strony, a nie poirytowanie użytkownika. Efekty powinny trwać wystarczająco długo, by dało się je zauważyć, ale nie aż tak długo, by przyciągały uwagę. Na ogół najlepsze rezultaty dają czasy rzędu 100 – 200 milisekund, które przekładają się na przejścia zauważalne, ale nie dekoncentrujące.

W przypadku naszego rozwijanego menu też zależy nam na wygodzie użytkownika, czas przejścia obu animowanych właściwości ustaliliśmy więc na 200 milisekund.

```
nav li ul {
  transition-property: transform, opacity;
  transition-duration: 200ms;
  ...
}
```

## Zmiana tempa przejść

Chciałbyś, aby przejście zaczynało się wolno i przyspieszało, zaczynało się szybko i zwalniało, odbywało się w jednostajnym tempie, polegało na zmianach skokowych, a może nawet ulegało „sprężystemu” wytłumianiu? Nie ma problemu — do sterowania tempem przejścia służy właściwość `transition-timing-function`.

<b>transition-timing-function</b>	
<b>Wartości:</b>	<code>&lt;timing-function&gt;#</code>
<b>Wartość początkowa:</b>	Ease
<b>Stosuje się do:</b>	Wszystkich elementów oraz pseudoelementów <code>:before</code> i <code>:after</code>
<b>Wartość wyliczona:</b>	Jak określono
<b>Dziedziczona:</b>	Nie
<b>Animowana:</b>	Nie

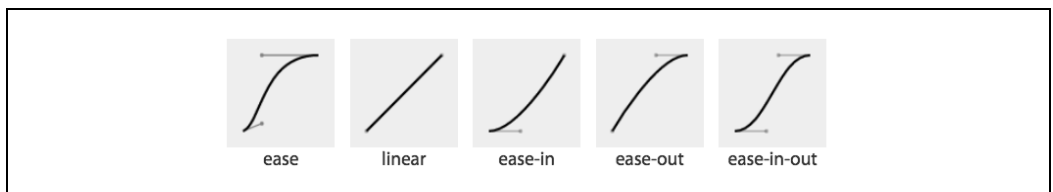
Właściwość `transition-timing-function` przyjmuje następujące wartości: `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out`, `step-start`, `step-end`, `steps(n, start)` — gdzie `n` jest liczbą kroków — `steps(n, end)` oraz `cubic-bezier(x1, y1, x2, y2)`. (Wartości te są też przyjmowane przez właściwość `animation-timing-function` i zostały bardziej szczegółowo omówione w rozdziale 18.).

Słowa kluczowe inne niż `step` są funkcjami płynnej zmiany prędkości, służącymi jako wygodne zamienniki konkretnych funkcji Béziera drugiego stopnia, których wykresy mają postać łagodnych krzywych. Specyfikacja obejmuje definicje pięciu takich funkcji, zebranych w tabeli 17.1.

Tabela 17.1. Słowa kluczowe odpowiadające zmianom prędkości opisanym krzywymi Béziera

Funkcja czasu	Opis	Odpowiednik w postaci cubic-bezier
<code>cubic-bezier()</code>	Określa krzywą Béziera drugiego stopnia.	<code>cubic-bezier(x1, y1, x2, y2)</code>
<code>ease</code>	Przejęcie zaczyna się wolno, następnie przyspiesza, potem spowalnia i kończy się bardzo wolno.	<code>cubic-bezier(0.25, 0.1, 0.25, 1)</code>
<code>linear</code>	Przejęcie przez cały czas odbywa się w tym samym tempie.	<code>cubic-bezier(0, 0, 1, 1)</code>
<code>ease-in</code>	Przejęcie zaczyna się wolno, potem przyspiesza.	<code>cubic-bezier(0.42, 0, 1, 1)</code>
<code>ease-out</code>	Przejęcie zaczyna się szybko, potem zwalnia.	<code>cubic-bezier(0, 0, 0.58, 1)</code>
<code>ease-in-out</code>	Podobnie jak <code>ease</code> — przejęcie jest szybsze w połowie, zaczyna się wolno, ale kończy się nie aż tak wolno jak <code>ease</code> .	<code>cubic-bezier(0.42, 0, 0.58, 1)</code>

Krzywe Béziera, także te, które odpowiadają za pięć predefiniowanych funkcji regulacji tempa zebranych w tabeli 17.1 i przedstawionych na wykresach na rysunku 17.3, przyjmują cztery parametry liczbowe. Na przykład wartość `linear` odpowiada wyrażeniu `cubic-bezier(0, 0, 1, 1)`. Pierwszy i trzeci parametr funkcji Béziera muszą się zawierać w przedziale od 0 do +1.

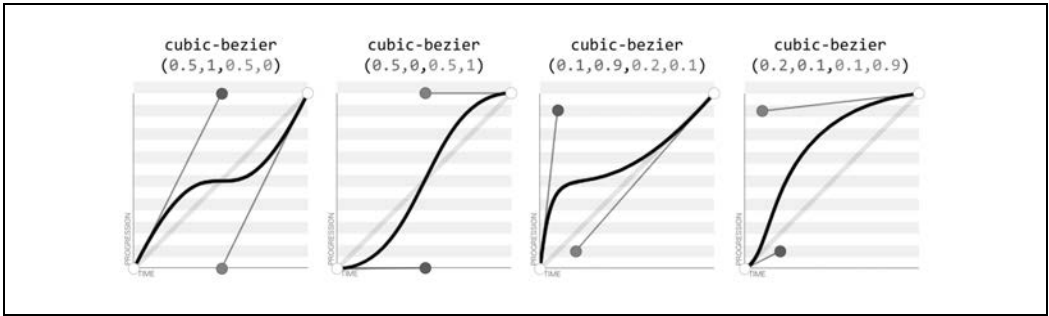


Rysunek 17.3. Krzywe odpowiadające predefiniowanym funkcjom Béziera

Cztery parametry funkcji `cubic-bezier()` definiują współrzędne  $x$  oraz  $y$  dwóch *uchwyty* na diagramie funkcji. Uchwyty te są powiązane z końcami wykresu krzywej, który rozciąga się od lewego dolnego do prawego górnego rogu diagramu. Krzywe są konstruowane na podstawie tych dwóch rogów oraz współrzędnych uchwytów z wykorzystaniem funkcji Béziera.

Aby się przekonać, na czym to polega, przyjrzyj się krzywym oraz odpowiadającym im wartościom, pokazanym na rysunku 17.4.

Rozważmy pierwszy przykład. Dwie pierwsze wartości, odpowiadające parametrom  $x1$  i  $y1$ , to 0.5 oraz 1. Położenie pierwszego uchwytu wyznaczamy, idąc najpierw do połowy diagramu ( $x1 = 0,5$ ), a potem na samą górę ( $y1 = 1$ ). Na tej samej zasadzie współrzędne 0.5, 0 dla parametrów  $x2, y2$  wyznaczają położenie drugiego uchwytu — na dole diagramu, pośrodku. Pokazana krzywa wynika z położenia obu uchwytów.

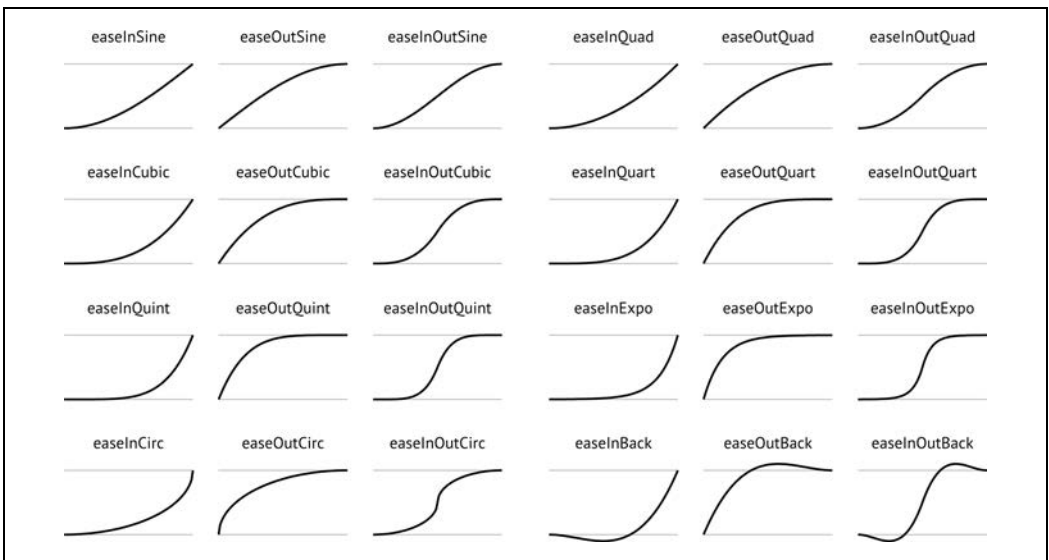


Rysunek 17.4. Cztery krzywe Béziera oraz wartości funkcji  $\text{cubic-bezier}()$  (na podstawie <http://cubic-bezier.com>)

Na drugim przykładzie współrzędne uchwytów zostały zamienione miejscami, co poskutkowało zmianą przebiegu krzywej. Na tej samej zasadzie zmienione zostały przykłady trzeci i czwarty, które są niejako swoimi przeciwieństwami. Zwróć uwagę na zmianę w kształcie otrzymanej krzywej, wynikającą z zamiany położenia uchwytów.

Predefiniowane słowa kluczowe mają dość ograniczone możliwości. Jeśli chciałbyś uzyskać efekty lepiej odzwierciedlające prawa animacji, to zamiast predefiniowanych słów być może powinieneś użyć funkcji Béziera drugiego stopnia opisanych czterema parametrami w postaci liczb rzeczywistych. Jeśli masz algebrę w małym palcu albo ogromne doświadczenie w obsłudze programów takich jak FreeHand czy Illustrator, to być może bez trudu umiesz sobie wyobrazić potrzebne krzywe; w przeciwnym razie skorzystaj z narzędzi online umożliwiających eksperymentowanie z różnymi wartościami (na przykład <http://cubic-bezier.com>). Pozwalają one na porównywanie różnych krzywych — zarówno tych odpowiadających predefiniowanym słowom kluczowym, jak i samodzielnie zdefiniowanych funkcji Béziera.

Na rysunku 17.5 zostały pokazane różne inne przebiegi funkcji Béziera, dostępne na stronie <http://easings.net>. Umożliwiają one uzyskanie bardziej realistycznych i estetycznych animacji.



Rysunek 17.5. Przydatne funkcje Béziera opracowane przez innych autorów (ze strony <http://easings.net>)



Choć autorzy wspomnianej strony nazwali opracowane przez siebie rodzaje animacji, ich nazwy nie wchodzą w skład specyfikacji CSS. Należy je zapisać następująco:

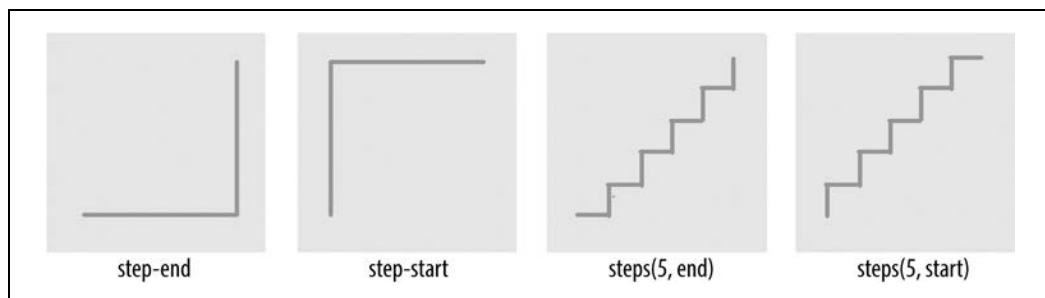
Nieoficjalna nazwa	Parametry funkcji Béziera drugiego stopnia
easeInSine	cubic-bezier(0.47, 0, 0.745, 0.715)
easeOutSine	cubic-bezier(0.39, 0.575, 0.565, 1)
easeInOutSine	cubic-bezier(0.445, 0.05, 0.55, 0.95)
easeInQuad	cubic-bezier(0.55, 0.085, 0.68, 0.53)
easeOutQuad	cubic-bezier(0.25, 0.46, 0.45, 0.94)
easeInOutQuad	cubic-bezier(0.455, 0.03, 0.515, 0.955)
easeInCubic	cubic-bezier(0.55, 0.055, 0.675, 0.19)
easeOutCubic	cubic-bezier(0.215, 0.61, 0.355, 1)
easeInOutCubic	cubic-bezier(0.645, 0.045, 0.355, 1)
easeInQuart	cubic-bezier(0.895, 0.03, 0.685, 0.22)
easeOutQuart	cubic-bezier(0.165, 0.84, 0.44, 1)
easeInOutQuart	cubic-bezier(0.77, 0, 0.175, 1)
easeInQuint	cubic-bezier(0.755, 0.05, 0.855, 0.06)
easeOutQuint	cubic-bezier(0.23, 1, 0.32, 1)
easeInOutQuint	cubic-bezier(0.86, 0, 0.07, 1)
easeInExpo	cubic-bezier(0.95, 0.05, 0.795, 0.035)
easeOutExpo	cubic-bezier(0.19, 1, 0.22, 1)
easeInOutExpo	cubic-bezier(1, 0, 0, 1)
easeInCirc	cubic-bezier(0.6, 0.04, 0.98, 0.335)
easeOutCirc	cubic-bezier(0.075, 0.82, 0.165, 1)
easeInOutCirc	cubic-bezier(0.785, 0.135, 0.15, 0.86)
easeInBack	cubic-bezier(0.6, -0.28, 0.735, 0.045)
easeOutBack	cubic-bezier(0.175, 0.885, 0.32, 1.275)
easeInOutBack	cubic-bezier(0.68, -0.55, 0.265, 1.55)

## Tempo krokowe

Dostępne są też funkcje umożliwiające zdefiniowanie tempa krokowego, a także dwa predefiniowane rodzaje takiego przejścia:

Funkcja tempa	Definicja
step-start	Przez cały czas przejścia wyświetlany jest ostatni krok animacji. Równoważne steps(1, start).
step-end	Przez cały czas przejścia wyświetlany jest ostatni krok animacji. Równoważne steps(1, end).
steps(n, start)	Wyświetlanych jest <i>n</i> pojedynczych kroków, przy czym pierwszy krok pojawia się w chwili odpowiadającej <i>n</i> /100 procent całkowitego czasu przejścia.
steps(n, end)	Wyświetlanych jest <i>n</i> pojedynczych kroków, przy czym wartości początkowe tych kroków są widoczne przez <i>n</i> /100 procent całkowitego czasu przejścia.

Jak wynika z rysunku 17.6, funkcje krokowe przedstawiają przejście od wartości początkowej do wartości końcowej w sekwencji kroków, a nie na podstawie płynnej krzywej.



Rysunek 17.6. Funkcje opisujące tempo krokowe

Funkcje opisujące tempo krokowe umożliwiają podzielenie przejścia na równe kroki. Funkcje te definiują liczbę i kierunek kroków. Są dwie opcje kierunku: start i end. W przypadku opcji start pierwszy krok odpowiada samemu początkowi animacji. W przypadku opcji end ostatni krok odpowiada samemu końcowi animacji. Na przykład wyrażenie `steps(5, end)` spowodowałoby wyświetlenie następujących równych kroków przejścia: 0%, 20%, 40%, 60% i 80%. Z kolei wyrażenie `steps(5, start)` spowodowałoby wyświetlenie następujących równych kroków: 20%, 40%, 60%, 80% i 100%.

Funkcja `step-start` jest odpowiednikiem wyrażenia `steps(1, start)`. W przypadku jej zastosowania przejście właściwości polega na wyświetleniu *końcowej* wartości tego przejścia, od początku do samego końca jego trwania. Funkcja `step-end` jest odpowiednikiem wyrażenia `steps(1, end)` i powoduje wyświetlenie *początkowej* wartości przejścia, od początku do samego końca jego trwania.



Tempo krokowe, ze szczególnym uwzględnieniem znaczenia wartości start i end, zostało omówione w rozdziale 18.

Wróćmy do wspomnianej wcześniej bardzo długiej deklaracji właściwości `transition-property`. Za jej pomocą możemy zadeklarować jedną funkcję tempa dla wszystkich właściwości albo zdefiniować różne funkcje tempa dla każdej właściwości z osobna i tak dalej. W poniższym przypadku wszystkie właściwości ze zdefiniowanymi przejściami będą trwały tyle samo i przebiegną w tym samym tempie:

```
div {
  transition-property: color, border-width, border-color, border-radius, transform,
  ↪opacity, box-shadow, width, padding;
  transition-duration: 200ms;
  transition-timing-function: ease-in;
}
```

Na własną prośbę możemy też utrudnić użytkownikowi korzystanie ze strony, animując przejścia różnych właściwości w zupełnie innym tempie. (Jak w kolejnym przykładzie).

Pamiętaj, że właściwość `transition-timing-function` nie zmienia czasu trwania przejścia — od tego mamy właściwość `transition-duration`. Właściwość ta zmienia tylko tempo przejścia w ramach określonego czasu. Rozważmy następujący przykład:

```

div {
  ...
  transition-property: color, border-width, border-color, border-radius, transform,
  ↳opacity, box-shadow, width, padding;
  transition-duration: 200ms;
  transition-timing-function: ease, ease-in, ease-out, ease-in-out, linear,
  ↳step-end, step-start, steps(5, start), steps(3, end);
}

```

Nawet jeśli podamy dziewięć różnych funkcji tempa dla dziewięciu różnych właściwości, to dopóki przejścia będą miały ten sam czas trwania i to samo opóźnienie, wszystkie właściwości rozpoczną i zakończą przejście jednocześnie. Funkcje tempa decydują tylko o przebiegu przejścia w czasie jego trwania, ale nie mają wpływu na czas, w jakim to przejście się zakończy. (Jak już wspomnieliśmy, przejścia takie jak pokazane przed chwilą byłyby bardzo uciążliwe pod względem użytkowym. Wystrzegaj się tego rodzaju metod).

Najlepszym sposobem obeznania się z różnymi funkcjami tempa jest poeksperymentowanie z nimi i przekonanie się, które najlepiej nadają się do uzyskania potrzebnego Ci w danej sytuacji efektu. Podczas testowania nadaj właściwości `transition-duration` stosunkowo dużą wartość, aby lepiej uwidocznić różnice między funkcjami. 🎵 Przy szybszych animacjach możesz nie być w stanie dostrzec różnic między niektórymi funkcjami tempa; pamiętaj tylko, by skrócić czas trwania przejść przed umieszczeniem projektu w internecie!

## Opóźnianie przejść

Właściwość `transition-delay` umożliwia wprowadzenie opóźnienia między zmianą inicjującą przejście zdefiniowane dla danego elementu a rzeczywistym rozpoczęciem tego przejścia.

<b>transition-delay</b>	
<b>Wartości:</b>	<code>&lt;time&gt;#</code>
<b>Wartość początkowa:</b>	0s
<b>Stosuje się do:</b>	Wszystkich elementów oraz pseudoelementów <code>:before</code> i <code>:after</code>
<b>Wartość wyliczona:</b>	Jak określono
<b>Dziedziczona:</b>	Nie
<b>Animowana:</b>	Nie

Wartość 0s właściwości `transition-delay` (wartość domyślna) oznacza, że przejście rozpocznie się natychmiast — zacznie się ono w chwili, gdy stan elementu ulegnie zmianie. Takie podejście znamy na przykład z natychmiastowej zmiany typu `a:hover`.

W przypadku wartości innej niż 0s parametr `<time>` właściwości `transition-delay` określa czas opóźnienia od chwili zmiany danej właściwości — przez czas opóźnienia żadne zmiany określone przez właściwości `transition` albo `transition-property` nie będą stosowane — aż do chwili, w której rozpocznie się animacja zadeklarowana we właściwościach `transition` albo `transition-property` (i będzie trwała aż do osiągnięcia wartości docelowych).

Co ciekawe, w przypadku tej właściwości można stosować ujemne wartości czasu. Efekty możliwe do uzyskania za pomocą ujemnych wartości `transition-delay` zostały opisane w dalszej części tego rozdziału, w podpunkcie „Ujemne wartości opóźnień”.

Wróćmy do przykładu z deklaracją `transition-property` obejmującą 8 (albo 21) właściwości. Aby wszystkie właściwości rozpoczęły przejścia od razu, bez opóźnienia, możemy pominąć deklarację właściwości `transition-delay` albo uwzględnić ją i nadać jej wartość `0s`. Równie dobrze można też rozpocząć animację połowy przejść od razu, a reszty — 200 milisekund później, jak w poniższym przykładzie:

```
div {
  transition-property: color, border, border-radius, transform, opacity,
    box-shadow, width, padding;
  transition-duration: 200ms;
  transition-timing-function: linear;
  transition-delay: 0s, 200ms;
}
```

Zastosowanie reguły `transition-delay: 0s, 200ms` dla sekwencji kilku właściwości sprawia, że (w tym przypadku) `color`, `border-radius`, `opacity` i `width` rozpoczną przejście od razu. Dla reszty właściwości animacja rozpocznie się w chwili, gdy parzyste właściwości zakończą przejście, bo wartość `transition-delay` dla tej reszty jest równa wartości czasu `transition-duration` dla wszystkich animowanych właściwości.

Podobnie jak ma to miejsce w przypadku właściwości `transition-duration` i `transition-timing-function`, jeśli liczba rozdzielonych przecinkami wartości `transition-delay` przewyższa liczbę rozdzielonych przecinkami wartości `transition-property`, nadmiarowe wartości opóźnień są ignorowane. Jeśli liczba rozdzielonych przecinkami wartości `transition-property` przewyższa liczbę rozdzielonych przecinkami wartości `transition-delay`, po wyczerpaniu dostępnych wartości opóźnień ich lista jest przeglądana od początku.

Możliwe jest zadeklarowanie dziewięciu różnych wartości `transition-delay` w taki sposób, by każda właściwość zaczęła przejście dopiero w chwili, gdy poprzednia je ukończy:

```
div {
  ...
  transition-property: color, border-width, border-color, border-radius, transform,
    ↪opacity, box-shadow, width, padding;
  transition-duration: 200ms;
  transition-timing-function: linear;
  transition-delay: 0s, 0.2s, 0.4s, 0.6s, 0.8s, 1s, 1.2s, 1.4s, 1.6s;
}
```

W tym przykładzie za pomocą właściwości `transition-duration` zadeklarowaliśmy, że każde przejście ma trwać 200 milisekund. Następnie przy użyciu właściwości `transition-delay` podaliśmy listę rozdzielonych przecinkami wartości — każda kolejna jest większa od poprzedniej o 200 milisekund (0,2 sekundy), czyli tyle samo, ile trwa przejście poszczególnych właściwości. W ten sposób kolejna właściwość rozpocznie przejścia w chwili, gdy poprzednia je ukończy.

Możemy też wyliczyć czas trwania i opóźnienia tak, by wszystkie animowane właściwości zakończyły przejście równocześnie:

```
div {
  ...
```

```

transition-property: color, border-width, border-color, border-radius, transform,
↳opacity, box-shadow, width, padding;
transition-duration: 1.8s, 1.6s, 1.4s, 1.2s, 1s, 0.8s, 0.6s, 0.4s, 0.2s;
transition-timing-function: linear;
transition-delay: 0s, 0.2s, 0.4s, 0.6s, 0.8s, 1s, 1.2s, 1.4s, 1.6s;
}

```

W podanym przykładzie wszystkie właściwości kończą przejścia po upływie 1,8 sekundy, ale każde z przejść ma inny czas trwania i inne opóźnienie. Dla każdego jednak suma wartości właściwości `transition-duration` oraz `transition-delay` wynosi 1,8 sekundy.

Zasadniczo należy dbać o to, by wszystkie przejścia zaczynały się jednocześnie. Możesz to osiągnąć, podając pojedynczą wartość dla właściwości `transition-delay` — wartość ta zostanie przypisana wszystkim animowanym właściwościom. W rozwijanym menu z rysunku 17.1 zastosowaliśmy opóźnienie o wartości 50 milisekund. To opóźnienie nie jest na tyle duże, by użytkownik zwrócił na nie uwagę, i nie spowoduje subiektywnego spowolnienia aplikacji. Takie trwające 50 milisekund opóźnienie może za to pomóc w uniknięciu niepożądanego, nagłego otwierania się menu, gdy podczas przesuwania kursora z jednej strony serwisu WWW (lub aplikacji) na drugą użytkownik przypadkiem wskaże którąś z pozycji menu.

## Ujemne wartości opóźnień

Ujemna wartość właściwości `transition-delay`, mniejsza od wartości `transition-duration`, spowoduje natychmiastowe rozpoczęcie przejścia, ale nie od początku, lecz od pewnego momentu w trakcie tego przejścia. Na przykład: ▶

```

div {
  transform: translateX(0);
  transition-property: transform;
  transition-duration: 200ms;
  transition-delay: -150ms;
  transition-timing-function: linear;
}
div:hover {
  transform: translateX(200px);
}

```

Biorąc pod uwagę fakt, że wartość `transition-delay` wynosi `-150ms`, a całe przejście trwa 200ms, w tej sytuacji przejście rozpocznie się w trzech czwartych całego czasu jego trwania i będzie trwało 50 milisekund. W tym scenariuszu, przy założeniu liniowej funkcji tempa, od razu po wskazaniu elementu kursorem myszy element ten „przeskoczy” o 150px wzdłuż osi x, a pozostałą część drogi od 150. piksela do 200. piksela pokona już płynnie w czasie 50 milisekund.

Jeśli wartość bezwzględna ujemnej wartości `transition-delay` jest równa lub większa od wartości `transition-duration`, to zmiana wartości właściwości nastąpi w sposób natychmiastowy — tak jakby w ogóle nie zadeklarowano reguły `transition`. W tej sytuacji zdarzenie `transitionend` *nie zachodzi*.

Domyślnie przy przejściu powrotnym od stanu `:hover` do stanu początkowego ma zastosowanie ta sama wartość właściwości `transition-delay`. W poprzednim scenariuszu, ponieważ wartość `transition-delay` dla stanu `:hover` nie została zmodyfikowana, w drodze powrotnej dojdzie najpierw do przeskoku o 75% całości ruchu wstecz (czyli licząc w kierunku pierwotnego przejścia, do pozycji znajdującej się w 25% drogi), a potem do płynnego przejścia do punktu początkowego. Innymi słowy,

po odsunięciu kursora myszy znad elementu przeskoczy on wzdłuż osi x do położenia odległego o 50 pikseli od położenia początkowego, a potem w ciągu 50 milisekund płynnie powróci do punktu wyjścia (czyli znajdzie się w miejscu o zerowym przesunięciu względem osi x).

## Skrótowa właściwość transition

Skrótowa właściwość transition łączy cztery omówione dotąd właściwości: transition-property, transition-duration, transition-timing-function oraz transition-delay.

<b>transition</b>	
<b>Wartości:</b>	<single-transition>#
<b>Wartość początkowa:</b>	all 0s ease 0s
<b>Stosuje się do:</b>	Wszystkich elementów oraz pseudoelementów :before i :after
<b>Wartość wyliczona:</b>	Jak określono
<b>Dziedziczona:</b>	Nie
<b>Animowana:</b>	Nie
<b>&lt;single-transition&gt; = [ [ none   &lt;transition-property&gt; ]    &lt;time&gt;    &lt;transition-timing-function&gt;    &lt;time&gt; ]#</b>	

Właściwość transition przyjmuje wartość none lub dowolną liczbę rozdzielonych przecinkami zbiorów wartości opisujących *pojedyncze przejścia*. Pojedyncze przejście składa się z nazwy właściwości, która ma zostać mu poddana (bądź słowa kluczowego all w celu zastosowania przejścia dla wszystkich właściwości), czasu trwania przejścia, funkcji tempa oraz wartości opóźnienia.

Jeśli w skrótowej właściwości transition zadeklarujemy jeden zbiór parametrów przejścia i pominiemy w nim właściwość do animowania, to zamiast nazwy przyjęta zostanie domyślna wartość all. Jeśli pominięta zostanie wartość składowej transition-timing-function, przyjmie ona domyślną wartość ease. Jeśli podasz tylko jedną wartość czasu, będzie ona potraktowana jako czas trwania, a opóźnienie zostanie wyzerowane — tak jakby wartość właściwości transition-delay została ustawiona na 0s.

W ramach każdego zbioru parametrów przejścia istotna jest kolejność czasu trwania i czasu opóźnienia: pierwsza wartość, którą da się zinterpretować jako czas, będzie potraktowana jako czas trwania. Jeśli przed przecinkiem albo przed końcem deklaracji wystąpi jeszcze jedna wartość czasu, zostanie ona potraktowana jako opóźnienie.

Oto trzy równoważne sposoby na uzyskanie tego samego efektu przejścia:

```
nav li ul {
  transition: transform 200ms ease-in 50ms,
             opacity 200ms ease-in 50ms;
}

nav li ul {
  transition: all 200ms ease-in 50ms;
}

nav li ul {
  transition: 200ms ease-in 50ms;
}
```

W pierwszym przykładzie mamy skrót obejmujący dwie właściwości. Ponieważ przejście dotyczy wszystkich podanych właściwości, które zmieniają się w stanie `:hover`, moglibyśmy użyć słowa kluczowego `all`, tak jak w drugim przykładzie. A ponieważ `all` jest wartością domyślną, równie dobrze możemy zapisać całość w skrótce obejmującym tylko czas trwania, funkcję tempa oraz opóźnienie. Gdybyśmy użyli funkcji tempa `ease` zamiast `ease-in`, moglibyśmy także i ją pominąć w deklaracji, bo funkcja `ease` jest wybierana domyślnie.

Musimy jednak podać czas trwania, bo w przeciwnym razie przejście będzie niewidoczne. Innymi słowy, jedyna składowa właściwość `transition`, jaką możemy uznać za absolutnie niezbędną, to `transition-duration`.

Jeżeli zależałoby nam tylko na tym, by zmiana polegająca na otwarciu zamkniętego menu była po prostu opóźniona o pewien czas, ale przebiegała bez stopniowego przejścia, to i tak musielibyśmy podać wartość czasu trwania równą `0s`. Pamiętaj, że pierwsza wartość dająca się zinterpretować jako czas będzie potraktowana jako czas trwania, druga zaś jako opóźnienie:

```
nav li ul {
  transition: 0s 200ms; ...
```



Ta animacja będzie opóźniona o 200 milisekund, a potem na ekranie nagle, bez płynnego przejścia, pojawi się otwarte, w pełni nieprzezroczyste menu. Z perspektywy funkcjonalności jest to fatalne rozwiązanie. Choć jeśli zmienisz selektor z `nav li ul` na `*`, to może to być niezły pomysł na primaaprilisowy żart...

Jeśli mamy do czynienia z rozdzieloną przecinkami listą przejść (a nie z pojedynczą deklaracją), w której występuje słowo kluczowe `none`, cała deklaracja przejścia zostanie uznana za nieprawidłową i zignorowana.

```
div {
  transition-property: color, border-width, border-color, border-radius, transform,
  ↪opacity, box-shadow, width, padding;
  transition-duration: 200ms, 180ms, 160ms, 140ms, 120ms, 100ms, 1s, 2s, 3s;
  transition-timing-function: ease, ease-in, ease-out, ease-in-out, linear, step-end,
  ↪step-start, steps(5, start), steps(3, end);
  transition-delay: 0s, 0.2s, 0.4s, 0.6s, 0.8s, 1s, 1.2s, 1.4s, 1.6s;
}

div {
  transition:
    color 200ms,
    border-width 180ms ease-in 200ms,
    border-color 160ms ease-out 400ms,
    border-radius 140ms ease-in-out 600ms,
    transform 120ms linear 800ms,
    opacity 100ms step-end 1s,
    box-shadow 1s step-start 1.2s,
    width 2s steps(5, start) 1.4s,
    padding 3s steps(3, end) 1.6s;
}
```

Dwa poprzednie bloki reguł CSS są funkcjonalnymi odpowiednikami: możesz zadeklarować rozdzielone przecinkami wartości dla czterech osobnych składowych właściwości przejść bądź sporządzić jedną, rozdzieloną przecinkami listę wielu kompletnych przejść i przypisać ją skrótowej właściwości

transition. Nie da się jednak połączyć obu tych podejść: deklaracja `transition: transform, opacity 200ms ease-in 50ms` spowoduje płynną (w tempie `ease-in`) zmianę przezroczystości w ciągu 200 milisekund po trwającym 50 milisekund opóźnieniu, ale zmiana wynikająca z właściwości `transform` będzie natychmiastowa i nie wygeneruje zdarzenia `transitionend`.

## Na odwrót: przejście do początku

W poprzednich przykładach deklarowaliśmy pojedyncze przejścia. Wszystkie przejścia zaczynały się od stanu domyślnego i były inicjowane przez wskazanie kursorem myszy (`:hover`). W przypadku takich deklaracji po odsunięciu kursora myszy znad elementu właściwości wracają do stanu początkowego z użyciem tego samego przejścia, odwrotnej funkcji tempa i z takim samym opóźnieniem.

W przypadku globalnych deklaracji przejść zarówno stan domyślny, jak i stan `:hover` bazują na tej samej regule `transition` — selektor pasuje do obydwu tych stanów. Możemy jednak uniknąć powielania całego przejścia (albo niektórych jego właściwości) w drodze powrotnej, podając inne wartości właściwości przejścia dla stanu globalnego i inne dla stanu `:hover`.

W przypadku deklaracji przejść dla różnych stanów każda z nich odnosi się do przejścia *do* danego stanu:

```
a {
  background: yellow;
  transition: 200ms background-color linear 0s;
}
a:hover {
  background-color: orange;
  /* opóźnienie przy przejściu DO stanu :hover */
  transition-delay: 50ms;
}
```

W tym przypadku, gdy użytkownik wskaże łącze kursorem myszy, kolor tła zacznie się zmieniać na pomarańczowy po opóźnieniu wynoszącym 50 milisekund. Kiedy użytkownik odsunie kursor znad łącza, kolor tła zacznie się zmieniać natychmiast. Przejście zajmuje 200 milisekund w każdą stronę, a zmiana następuje w sposób liniowy. Ponieważ opóźnienie trwające 50 milisekund zostało podane tylko dla stanu `:hover` („pomarańczowego”), opóźnienie to zachodzi tylko przed zmianą koloru tła na pomarańczowy. ▶

W naszym przykładzie z rozwijanym menu w stanie `:hover` menu pojawia się i wydłuża w ciągu 200 milisekund w tempie `ease-in` po opóźnieniu wynoszącym 50 milisekund. Przejście zostało zdefiniowane za pomocą właściwości `transition` dla stanu domyślnego (element nie jest wskazany kursorem myszy). Gdy użytkownik odsunie kursor, wartości właściwości wracają do stanu pierwotnego w ciągu 200 milisekund, w tempie `ease-out`, po upływie trwającego 50 milisekund opóźnienia. Ów odwrotny efekt bazuje na wartości właściwości `transition` dla stanu, w którym element nie jest wskazany kursorem myszy. Jest to zachowanie domyślne, ale możemy je zmienić. Wprawdzie to domyślne zachowanie zapewnia największą wygodę obsługi, więc raczej nie należy go modyfikować, warto jednak wiedzieć, że da się to zrobić.

Jeśli zależałoby nam na tym, by zamykanie menu przebiegało skokowo i wolno (*nie* powinno nam na tym zależeć, bo będzie to uciążliwe dla użytkownika; na potrzeby tego przykładu powiedzmy jednak, że potrzebujemy takiego efektu), to możemy zadeklarować dwa różne przejścia:



```

nav ul ul {
  transform: scale(1, 0);
  opacity: 0;
  ...
  transition: all 4s steps(8, start) 1s;
}
nav li:hover ul {
  transform: scale(1, 1);
  opacity: 1;
  transition: all 200ms linear 50ms;
}

```

Przypominamy, że przejścia następują *do* stanu: kiedy dochodzi do zmiany stylu, w grę wchodzi nowe wartości właściwości biorących udział w przejściu, a nie stare. W podanym przykładzie zdecydowaliśmy się na płynną, liniową animację w stanie `:hover`. Przejście, które ma wtedy zastosowanie, jest przejściem do stanu docelowego. W przykładzie tym, kiedy użytkownik wskaże kursorem myszy element rodzica rozwijanego menu, czyli element `li`, nastąpi stopniowe, ale szybkie otwarcie tego menu — potrwa ono 200 milisekund, z opóźnieniem wynoszącym 50 milisekund. Kiedy użytkownik odsunie kursor myszy znad rozwijanego menu albo znad elementu rodzica (`li`), po opóźnieniu wynoszącym 1 sekundę nastąpi czterosekundowe przejście wstecz, podzielone na osiem kroków.

Jeśli mamy tylko jedno przejście, umieszczamy je w globalnym stanie, *od* którego wychodzimy (źródłowym), bo chcemy, aby przejście to nastąpiło *do* jakiegoś stanu — na przykład związanego ze wskazaniem elementu kursorem myszy albo ze zmianą klasy. Ponieważ chcemy, aby przejście nastąpiło w przypadku dowolnej zmiany, zwykle umieszczamy deklarację tego przejścia w początkowym, domyślnym (najmniej sprecyzowanym) bloku reguł. Jeśli zależy Ci na pełniejszej kontroli nad sytuacją i chciałbyś uzyskać różne efekty, w zależności od kierunku przejścia, pamiętaj o umieszczeniu deklaracji przejść we wszystkich potrzebnych klasach i stanach interfejsu.



Wystrzegaj się definiowania przejść jednocześnie na elementach przodków i elementach potomnych. Deklarowanie przejść mających się zacząć wkrótce po zajściu jakiejś zmiany, która spowoduje innego rodzaju przejście na węzłach przodków albo węzłach potomnych, może mieć nieoczekiwane rezultaty. Jeżeli przejście na elementach potomnych zakończy się przed dokończeniem przejścia na elementach przodków, element potomny ponownie odziedziczy (wciąż będąc w trakcie przejścia) wartość właściwości rodzica. I niekoniecznie musi to być spodziewany efekt.

### Zmiana kierunku przerwanych przejść

Jeśli przejście zostanie przerwane przed jego ukończeniem (na przykład w wyniku odsunięcia kursora myszy znad przykładowego rozwijanego menu, zanim skończy się ono otwierać), właściwości są zerowane do wartości, które miały one przed rozpoczęciem przejścia, i następuje przejście powrotne do tych wartości. Ponieważ powtórzenie tego samego czasu trwania i funkcji tempa w odniesieniu do powrotnego, częściowego przejścia może prowadzić do dziwacznych rezultatów i być niewygodne dla użytkownika, specyfikacja przejść CSS przewiduje skrócenie przerwanych animacji powrotnych.

W naszym przykładowym menu mamy zadeklarowaną właściwość `transition-delay` o wartości 50s dla stanu domyślnego i żadnych właściwości przejść dla stanu `:hover`. To oznacza, że przeglądarka odczeka 50 milisekund przed rozpoczęciem odwrotnego (zamykającego) przejścia.

Kiedy przejście biegnące do przodu zakończy się osiągnięciem wartości końcowych i wyzwolone zostanie zdarzenie `transitionend`, wszystkie przeglądarki powielą właściwość `transition-delay` dla stanu odwrotnego.

Jak widać w tabeli 17.2, jeśli przejście się nie skończy — na przykład użytkownik odsunie kursor myszy znad nawigacji, zanim przejście to zostanie dokończzone — wszystkie przeglądarki oprócz Microsoft Edge powtórzą opóźnienie w przeciwnym kierunku. Niektóre przeglądarki powielą też właściwość `transition-duration`, ale w Edge i w Firefoxie zostały zaimplementowane zgodnie ze specyfikacją współczynniki skracające animację powrotną.

Tabela 17.2. Obsługa niedokończonych przejść w różnych przeglądarkach

Przeglądarka	Opóźnienie przejścia powrotnego	Czas przejścia	Czas łączny
Chrome 37	Tak	200 ms	0,200 s
Chrome 42	Tak	200 ms	0,250 s
Safari 8	Tak	200 ms	0,200 s
Firefox 41	Tak	38 ms	0,038 s
Opera 32	Tak	200 ms	0,250 s
Edge 12	Nie	38 ms	0,038 s

Przypuśćmy, że użytkownik odsunie kursor znad menu po 75 milisekundach od chwili rozpoczęcia przejścia. To oznacza, że rozwijane menu zacznie się zamykać, jeszcze zanim będzie w pełni otwarte i nieprzezroczyste. Przeglądarka powinna zastosować trwające 50 milisekund opóźnienie przed zamknięciem menu, na tej samej zasadzie, na jakiej odczekała 50 milisekund przed jego otwarciem.


Akurat to zachowanie można uznać za poprawne, bo daje ono ułamek sekundy opóźnienia przed zamknięciem, pozwalający uniknąć skokowej animacji menu, jeśli użytkownik odsunie kursor tylko przypadkiem. Jak widać w tabeli 17.2, postępują tak wszystkie przeglądarki oprócz Microsoft Edge.

Choć daliśmy przeglądarce tylko 75 milisekund na częściowe otwarcie rozwijanego menu przed jego zamknięciem, przywrócenie stanu początkowego w niektórych przeglądarkach potrwa 200 milisekund — pełną wartość właściwości `transition-duration`. Inne przeglądarki, w tym Firefox i Edge, mają zaimplementowany zgodny ze specyfikacją CSS mechanizm skracania przejścia zwrotnego i korekty wartości początkowej tego przejścia. Jeśli mechanizm ten działa prawidłowo, czas potrzebny do ukończenia częściowego przejścia powrotnego będzie podobny do początkowej wartości czasu, ale niekoniecznie taki sam.

W przypadku krokowej funkcji tempa przeglądarki Firefox i Edge zastosują taki czas przejścia powrotnego, jaki wynika z zaokrąglenia czasu potrzebnego do wykonania tylu kroków, ile zostało wyświetlonych przed przerwaniem animacji. Przypuśćmy, że przejście miało trwać 10 sekund i składać się z 10 kroków, a przerwa (i związana z nią zmiana wartości właściwości na początkowe) nastąpiła po 3,25 sekundy — w efekcie przejście zakończyło się w jednej czwartej etapu między trzecim a czwartym krokiem (3 kroki ukończone, co odpowiada 30% całego przejścia). W tej sytuacji powrót do pierwotnych wartości zajmie 3 sekundy. W poniższym, ilustrującym tę sytuację przykładzie szerokość elementu `div` wzrosłaby do 130 pikseli, zanim (po odsunięciu kursora myszy) nastąpiłby powrót do szerokości 100 pikseli:

```
div {
  width: 100px;
  transition: width 10s steps(10, start);
}
div:hover {
  width: 200px;
}
```

O ile jednak czas powrotnego przejścia zostanie zaokrąglony do czasu, jaki upłynął do osiągnięcia ostatniego zakończonego kroku, to przejście w tym kierunku zostanie podzielone na *początkowo* zadeklarowaną liczbę kroków, a nie liczbę kroków zakończonych. W naszym przykładzie z przerwaniem przejścia po 3,25 sekundy przejście powrotne będzie trwało 3 sekundy, ale obejmie wszystkie 10 kroków. Te powrotne kroki będą krótsze — każdy z nich będzie trwał 300 milisekund i każdy spowoduje zwężenie elementu o 3 piksele, a nie o 10.

Jeśli animowalibyśmy „duszka” przy użyciu przejścia właściwości `background-position` , to takie rozwiązanie wyglądałoby fatalnie. Być może specyfikacja oraz jej implementacje ulegną zmianie, aby przejście w przeciwnym kierunku składało się z tej samej liczby kroków co niepełne przejście w przód.

W niektórych przeglądarkach w opisanej sytuacji przejście powrotne trwałoby 10 sekund — przez ten czas nastąpiłoby cofnięcie 3 wykonanych kroków, ale podzielone na 10 kroków. To oznacza, że przykładowy element zwężałby się w 3-pikselowych krokach, ale co sekundę. Chodzi tu o przeglądarki, w których nie został zaimplementowany mechanizm skracania przejścia powrotnego — przejście to potrwa wówczas pełne 10 sekund zamiast 3 sekund, a powrót z 30% początkowej zmiany zostanie podzielony na 10 kroków. Bez względu na to, czy początkowe przejście zostało ukończone, czy nie, w przeglądarkach tych zostanie zastosowany cały czas trwania przejścia początkowego, ewentualnie pomniejszony o wartość bezwzględnej ujemnej wartości `transition-delay`, jeśli została ona zadeklarowana. Stanie się tak bez względu na rodzaj funkcji tempa. W przypadku tempa krokowego w opisanym przed chwilą przykładzie przejście powrotne zajmie więc 10 sekund. W przykładzie z rozwijanym menu przejście powrotne potrwa 200 milisekund, bez względu na to, czy menu zostało w pełni rozwinięte, czy nie.

W przeglądarkach, w których mechanizmy korygowania czasu przejścia powrotnego funkcjonują prawidłowo, jeśli wybrana została liniowa funkcja tempa, czas trwania przejścia będzie taki sam w obu kierunkach. Jeśli wybrana została krokowa funkcja tempa, czas przejścia powrotnego będzie równy czasowi, jaki zajęło osiągnięcie ostatniego ukończonego kroku. W przypadku wszystkich pozostałych funkcji `cubic-bezier` czas trwania będzie proporcjonalny do postępu początkowego przejścia przed jego przerwaniem (<https://drafts.csswg.org/css-transitions/transition-reversing-demo>). Ujemne wartości właściwości `transition-delay` również są proporcjonalnie skracane. Dodatkowo wartości opóźnień pozostają bez zmian w obu kierunkach.

W przypadku niedokończonego przejścia w żadnej przeglądarce nie zostanie wyzwolone zdarzenie `transitionend` dla stanu `:hover`, ale we wszystkich zdarzenie `transitionend` zostanie wyzwolone dla stanu powrotnego — w chwili, gdy menu skończy się związać. Rzeczywisty czas trwania przejścia powrotnego (`elapsedTime`) będzie uzależniony od tego, czy przeglądarka zamknie menu w czasie pełnych 200 milisekund, czy raczej potrwa to tyle, ile zajęło częściowe otwarcie tego menu.

Aby zmodyfikować te wartości, zadeklaruj właściwości przejścia zarówno dla stanu początkowego, jak i końcowego (na przykład dla stanów wskazania i niewskazania elementu kursorem myszy). Choć nie będzie to miało wpływu na skrócenie przejścia powrotnego, da nieco większą kontrolę nad jego przebiegiem.

## Animowane właściwości i wartości

Zanim zajmiesz się tworzeniem przejść i animacji, powinieneś zwrócić uwagę na fakt, że nie wszystkie właściwości da się animować. Przejścia (i animacje) można tworzyć tylko w przypadku dających się animować właściwości CSS... tylko które to są właściwości?



Listę tych właściwości zamieściliśmy w dodatku A. CSS jednak nieustannie się rozwija i lista animowanych właściwości ([https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_animated\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties)) zapewne się rozszerzy.

Jednym ze sposobów na domyslenie się, które właściwości można animować, jest sprawdzenie, które z nich przyjmują wartości dające się interpolować. *Interpolacja* to proces wyliczania punktów danych wypadających pomiędzy znanymi punktami danych. Najważniejszym warunkiem decydującym o zdolności danej właściwości do animacji jest możliwość interpolacji *wartości wyliczonej*. Jeśli wartością wyliczoną właściwości są słowa kluczowe, to nie dadzą się one interpolować; chyba że słowa te mogą być zamienione na jakiegoś rodzaju wartości numeryczne. Podstawowa zasada jest taka, że jeżeli da się wyznaczyć wartość pośrednią między dwiema wartościami jakiejś właściwości, to właściwość ta zapewne daje się animować.

Na przykład wartości właściwości `display`, takie jak `block` i `inline-block`, nie są numeryczne i nie można określić między nimi wartości pośredniej — a to oznacza, że nie da się ich animować. Wartości właściwości `transform`, takie jak `rotate(10deg)` oraz `rotate(20deg)`, mają wartość pośrednią — tu w postaci `rotate(15deg)` — a zatem da się je animować.

Właściwość `border` jest skrótem właściwości `border-style`, `border-width` i `border-color` (które z kolei same są skrótami właściwości decydujących o obramowaniach czterech boków elementu). Choć nie da się wyznaczyć wartości pośredniej między dwiema dowolnymi wartościami `border-style`, to wartości długości `border-width` są numeryczne, da się je więc animować. Słowa kluczowe `medium`, `thick` i `thin` mają odpowiedniki liczbowe i da się je interpolować: wyliczona wartość właściwości `border-width` jest w istocie wartością długości odpowiadającą tym słowom.

Kolory w przypadku właściwości `border-color` są wartościami liczbowymi (nawet kolory nazwane mają odpowiedniki w postaci liczb szesnastkowych), a zatem kolory także da się animować. Jeśli zadeklarujesz przejście od `border: red solid 3px` do `border: blue dashed 10px`, szerokość obramowania i jego kolory będą animowane zgodnie z podaną prędkością, ale właściwość `border-style` przeskoczy z wartości `solid` na `dashed` od razu po rozpoczęciu przejścia (po ewentualnym opóźnieniu).

Jak już wspomnieliśmy (zob. też dodatek A), wartości numeryczne na ogół dają się animować. Słowa kluczowe, których nie da się przełożyć na wartości numeryczne, zwykle nie dają się animować. Funkcje CSS, które jako parametry przyjmują wartości liczbowe, zazwyczaj także mogą być animowane. Jedynym wyjątkiem od tej reguły jest `visibility` — choć nie ma punktu pośredniego między

wartościami `visible` i `hidden`, to wartości właściwości `visibility` dają się interpolować pomiędzy „widocznością” a „niewidocznością”. W przypadku właściwości `visibility` albo wartość początkowa, albo końcowa musi być jednak równa `visible`, bo w przeciwnym razie interpolacja nie będzie możliwa. W przypadku przejścia z wartości `visible` na `hidden` zmiana dokona się na końcu animacji. W przypadku przejścia z `hidden` na `visible` zmiana nastąpi na początku animacji.

Wartość `auto` na ogół należy traktować jako nieanimowaną i powinno się jej unikać w przypadku animacji oraz przejść. Zgodnie ze specyfikacją nie jest to wartość animowana, ale niektóre przeglądarki interpretują bieżącą wartość numeryczną `auto` (taką jak `height: auto`) jako `0px`. Wartości `auto` nie da się animować w przypadku właściwości takich jak `height`, `width`, `top`, `bottom`, `left`, `right` i `margin`.

Często da się jednak zastosować alternatywną właściwość albo wartość. Na przykład zamiast stworzyć przejście z `height: 0` na `height: auto` zrób przejście z `max-height: 0` na `max-height: 100vh`, co na ogół powinno dać oczekiwany rezultat. Wartość `auto` da się animować w przypadku właściwości `min-height` i `min-width`, bo `min-height: auto` ma wyliczoną wartość `0`.

## Na czym polega interpolacja wartości właściwości?

Interpolacja jest możliwa wtedy, gdy da się wyliczyć wartości pośrednie między dwiema lub większą liczbą znanych wartości. Wartości dające się interpolować można stosować w przejściach i animacjach.

Liczby rzeczywiste są interpolowane jako wartości zmiennoprzecinkowe. Liczby całkowite są po interpolacji zaokrąglane do pełnych wartości (wzrost albo spadek następuje w postaci wartości całkowitych).

W CSS jednostki długości i wartości procentowe są przeliczane na liczby rzeczywiste. W przypadku przejść albo animacji z użyciem funkcji `calc()` bądź przejść z jakiejś wartości długości na wartość procentową (lub na odwrót) wartości te zostaną przekształcone na funkcję `calc()` i interpolowane jako liczby rzeczywiste.

Kolory — zarówno w postaci HSLA, RGB, jak i kolorów nazwanych (jak `aliceblue`) — są na potrzeby przejść przeliczane na odpowiedniki RGBA i interpolowane w przestrzeni kolorów RGBA.

W przypadku animowania grubości znaków, jeśli użyjesz słów kluczowych takich jak `bold`, słowa te zostaną przekształcone na wartości numeryczne i animowane w krokach będących wielokrotnościami wartości `100`. Być może zmieni się to w przyszłości, jeśli grubości znaków będzie się dało wyrażać w dowolnych liczbach całkowitych — w takim przypadku będą one interpolowane jako wartości całkowite, a nie wielokrotności `100`.

Jeśli zastosujesz animowaną właściwość składającą się z kilku komponentów, to każdy z nich będzie interpolowany zgodnie ze specyfiką tego komponentu. Na przykład właściwość `text-shadow` może się składać z czterech komponentów: koloru oraz wartości `x`, `y` i `blur`. Kolor jest interpolowany jako właściwość `color`, składniki `x`, `y` oraz `blur` są interpolowane jako wartości długości. Cienie typu `box-shadow` mają dwie dodatkowe, opcjonalne właściwości: akcent `inset` (lub jego brak) oraz `spread`. Właściwość `spread`, jako wartość długości, jest interpolowana właśnie jako długość. Słowo kluczowe `inset` nie ma odpowiednika liczbowego: możesz dokonać przejścia między jednym cieniem z właściwością `inset` a innym albo między jednym wieloskładnikowym cieniem a drugim, ale zasadniczo nie ma sposobu na uzyskanie płynnego przejścia między cieniem `inset` a zwykłym.

Podobnie jak ma to miejsce w przypadku wartości wieloskładnikowych, przejścia z użyciem gradientów są możliwe tylko w przypadku gradientów tego samego typu (liniowych lub kołowych), o równej liczbie znaczników koloru. Kolory każdego znacznika są interpolowane jako wartości koloru, a położenie każdego znacznika jest interpolowane jako długość albo wartość procentowa.

## Interpolowanie powtarzających się właściwości

W przypadku list zawierających różnego rodzaju właściwości każda pozycja na liście jest interpolowana w sposób właściwy dla jej rodzaju — jeśli tylko na listach jest ta sama liczba pozycji (lub dające się cyklicznie powtórzyć pozycje), a każdą parę wartości można interpolować:

```
.img {
  background-image:
    url(1.gif), url(2.gif), url(3.gif), url(4.gif),
    url(5.gif), url(6.gif), url(7.gif), url(8.gif),
    url(9.gif), url(10.gif), url(11.gif), url(12.gif);
  background-size: 10px 10px, 20px 20px, 30px 30px, 40px 40px;
  transition: background-size 1s ease-in 0s;
}
.img:hover {
  background-size: 25px 25px, 50px 50px, 75px 75px, 100px 100px;
}
```

Weźmy przykładowe przejście dotyczące czterech rozmiarów obrazu w tle (`background-size`). Każda para wartości na obu listach została podana w pikselach, można więc na przykład dokonać płynnego przejścia od trzeciej wartości `background-size` sprzed przekształcenia do trzeciej wartości `background-size` z listy po przekształceniu. W podanym przykładzie szerokość i wysokość obrazów numer 1, 6 i 10 po wskazaniu tych obrazów kursorem myszy zostanie zmieniona z 10px na 25px. Na tej samej zasadzie obrazy numer 3, 7 i 11 zmienią rozmiar z 30px na 75px i tak dalej.

To oznacza, że wartości właściwości `background-size` zostaną powtórzone trzykrotnie, tak jakbyśmy napisali kod CSS w następujący sposób:

```
.img {
  ...
  background-size: 10px 10px, 20px 20px, 30px 30px, 40px 40px,
                  10px 10px, 20px 20px, 30px 30px, 40px 40px,
                  10px 10px, 20px 20px, 30px 30px, 40px 40px;
  ...
}
.img:hover {
  background-size: 25px 25px, 50px 50px, 75px 75px, 100px 100px,
                  25px 25px, 50px 50px, 75px 75px, 100px 100px,
                  25px 25px, 50px 50px, 75px 75px, 100px 100px;
}
```

Jeśli liczba rozdzielonych przecinkami wartości jest mniejsza niż liczba obrazów w tle, to lista wartości jest powtarzana od początku, aż wartości wystarczy — nawet jeśli lista dla stanu `:hover` nie jest zgodna z listą dla stanu początkowego:

```
.img:hover {
  background-size: 33px 33px, 66px 66px, 99px 99px;
}
```

Jeśli zażyczylibyśmy sobie przejścia od czterech deklaracji `background-size` dla stanu początkowego do trzech deklaracji `background-size` dla stanu `:hover` (wszystkie wartości w pikselach) i nadal mieli

byśmy 12 obrazów w tle do przetworzenia, to wartości dla stanu `:hover` i dla stanu początkowego zostałyby powtórzone (odpowiednio, trzykrotnie i czterokrotnie) aż do uzyskania 12 potrzebnych wartości. Byłoby to odpowiednikiem poniższej deklaracji:

```
.img {
  ...
  background-size: 10px 10px, 20px 20px, 30px 30px,
                  40px 40px, 10px 10px, 20px 20px,
                  30px 30px, 40px 40px, 10px 10px,
                  20px 20px, 30px 30px, 40px 40px;
  ...
}
.img:hover {
  background-size: 33px 33px, 66px 66px, 99px 99px,
                  33px 33px, 66px 66px, 99px 99px,
                  33px 33px, 66px 66px, 99px 99px,
                  33px 33px, 66px 66px, 99px 99px;
}
```

Jeśli dla jakiejś pary wartości nie da się określić wartości pośrednich — na przykład jeżeli właściwość `background-size` zmienia się z `contain` (w stanie domyślnym) na `cover` (w stanie `:hover`) — to zgodnie ze specyfikacją list z takimi wartościami nie da się interpolować. Niektóre przeglądarki ignorują jednak tę konkretną parę wartości na potrzeby przejścia i wciąż animują pozostałe wartości — te, które da się interpolować.

Niektóre właściwości da się animować, jeśli przeglądarka może uzupełnić brakujące, niezadeklarowane jawnie wartości. Na przykład w przypadku cieni przeglądarka zastosuje regułę `box-shadow: transparent 0 0 0` albo `box-shadow: inset transparent 0 0 0`, zastępując dowolne wartości, które nie zostaną w sposób jawny zadeklarowane dla stanu przed przejściem albo po nim. Przykłady tego rodzaju znajdziesz na stronie z materiałami pomocniczymi do tej książki (<https://meyerweb.github.io/csstdg4figs/>).

Tylko dające się interpolować wartości wyzwalają zdarzenia `transitionend`.

Jak już wspomnieliśmy, właściwość `visibility` jest animowana inaczej od pozostałych: jeśli animujemy lub tworzymy przejście do stanu `visible` (albo od niego), to proces ten jest traktowany jako pojedynczy krok. Element jest zawsze widoczny w trakcie przejścia albo animacji, jeśli rezultat funkcji tempa mieści się w zakresie między 0 a 1. W przypadku przejścia z `hidden` na `visible` do przełączenia wartości dojdzie na początku tego przejścia. W przypadku przejścia z `visible` na `hidden` do przełączenia wartości dojdzie na końcu tego przejścia. Należy pamiętać, że procesem tym da się kierować za pomocą funkcji tempa.

Jeśli przez przypadek zastosujesz w przejściu właściwość, której nie da się animować, nie przejmuj się. Cała deklaracja nie zostanie zignorowana. Przeglądarka po prostu nie dokona przejścia właściwości, której nie da się animować. Warto przy tym podkreślić, że niedające się animować (albo nieistniejące) właściwości CSS nie są wtedy zupełnie pomijane. Przeglądarka uwzględnia nierozpoznane albo nieanimowane właściwości, aby zachować ich miejsce na liście i dzięki temu zadbać o to, by kolejne rozdzielone przecinkami parametry przejść nie zostały powiązane z nieodpowiednimi właściwościami<sup>2</sup>.

---

<sup>2</sup> Może się to zmienić. CSS Working Group rozważa umożliwienie animowania wszystkich właściwości. W razie braku możliwości określenia punktu pośredniego między wartościami sprzed przejścia i po nim miałyby to polegać na jednorazowym przeskoku między tymi wartościami w połowie przebiegu funkcji tempa.



Przejścia mogą dotyczyć tylko tych właściwości, które w danej chwili nie są animowane z poziomu CSS. Jeśli dany element jest animowany, to związane z nim przejścia mogą nastąpić, ale tylko wtedy, gdy nie dotyczą one właściwości sterowanych w danej chwili przez animację. O animacjach CSS przeczytasz w rozdziale 18.

## Wyjścia awaryjne: przejścia to tylko ozdobniki

Obsługa przejść w przeglądarkach stoi na znakomitym poziomie. Wszystkie przeglądarki, w tym Safari, Chrome, Opera, Firefox, Edge i Internet Explorer (od wersji IE10), obsługują przejścia CSS.

Przejścia są ozdobnikami interfejsu użytkownika, ewentualny brak pełnej obsługi przejść nie powinien Cię więc zniechęcać do ich stosowania. Nawet jeśli jakaś przeglądarka nie obsługuje przejść CSS, to zmiany, które chcesz wprowadzić za pomocą przejścia, i tak zajdą — po prostu „przejście” ze stanu początkowego do końcowego nastąpi raptownie, w chwili przeliczenia parametrów stylu.

Użytkownicy mogą nie zobaczyć atrakcyjnego (a czasami irytującego...) efektu, ale nie odbierze im to możliwości zapoznania się z treścią strony.

Ponieważ przejścia są zasadniczo ozdobnikami, nie ma potrzeby sztucznego implementowania ich w „zabytkowych” wersjach przeglądarki IE. Choć za pomocą JavaScriptu da się zaimplementować przejścia w IE9 lub starszych wersjach Internet Explorera, a w przypadku Androida 4.3 (i starszych) używać prefiksów właściwości przejść, potrzeba stosowania tego rodzaju obejść jest znikoma.

## Drukowanie przejść

Przy drukowaniu strony albo aplikacji internetowej wykorzystywany jest arkusz stylów przystosowany do druku. Jeśli zastosowałeś style obsługujące jedynie atrybut `screen`, to CSS nie będzie miał żadnego wpływu na wygląd wydruku.

Często atrybuty mediów są pomijane, co stanowi odpowiednik domyślnej deklaracji `media="all"`. W zależności od przeglądarki próba wydrukowania elementu, dla którego zastosowano przejście, spowoduje albo zignorowanie interpolowanych wartości, albo wydruk elementu zgodnie z bieżącym stanem wartości jego właściwości.

Oczywiście na wydruku nie da się zobaczyć przejścia elementu, ale w niektórych przeglądarkach, takich jak Chrome, jeśli dany element jest w trakcie przejścia z jednego stanu do drugiego, to na wydruku zostanie pokazany bieżący stan tego elementu w chwili wywołania funkcji `print` (jeśli tylko dana właściwość jest drukowalna). Na przykład jeśli zmienia się kolor tła, to na wydruku nie będzie widać ani koloru sprzed przejścia, ani po przejściu, bo kolory tła zasadniczo nie są drukowane w ogóle. Jeśli jednak to kolor tekstu uległ zmianie z jednej wartości na inną, to na kolorowej drukarce albo w pliku PDF zobaczymy barwę zgodną z bieżącą wartością właściwości `color`.

W innych przeglądarkach, takich jak Firefox, o tym, czy na wydruku pojawi się wartość sprzed rozpoczęcia przejścia, czy wartość po zakończeniu przejścia, decyduje sposób inicjalizacji przejścia. Jeśli jest ono inicjowane przez stan `:hover`, to na wydruku pojawi się wartość nieuwzględniająca tego stanu, bo podczas korzystania z okna dialogowego wydruku kursor myszy nie będzie już raczej



wskazywał aktywnego elementu. Jeśli przejście polega na dodaniu klasy, to na wydruku pojawi się wartość po przejściu, nawet jeśli nie zostało ono ukończone. W takich sytuacjach wydruk następuje z pominięciem właściwości przejścia.

Ponieważ możemy zastosować osobny arkusz stylów lub reguł @media dla druku, przeglądarki przeliczają każdy arkusz osobno. W przypadku reguł przeznaczonych do druku style się nie zmieniają, nie ma więc mowy o przejściach. W takich przypadkach drukowanie działa tak, jakby wartości zmieniły się w sposób nagły, a nie stopniowo, z upływem czasu.



## A

adresy URL, 135  
alternatywne arkusze stylów, 36  
animacja @keyframes, 842  
animacje, 828, 835, 954  
  definiowanie długości, 846  
  deklarowanie iteracji, 847  
  drukowanie, 881  
  funkcje zmiany tempa, 864  
  iterowanie, 878  
  klatki kluczowe, 836  
  kolejność, 878  
  konfigurowanie, 837  
  łączenie, 853  
  nadawanie nazwy, 837  
  nazwy, 844  
  opóźnianie, 850  
  opóźnienie iteracji, 857  
  powtarzanie właściwości klatek kluczowych, 840  
  selektory klatek kluczowych, 838  
  skrypty, 842  
  tryby uzupełniania, 872  
  ustalenie kierunku, 849  
  ustawianie stanu odtwarzania, 871  
  wartości from i to, 839  
  wątek UI, 879  
  wydajność, 855  
  zdarzenia, 852, 880  
  zmiana tempa, 860  
animowanie  
  elementów, 843  
  funkcji tempa, 868  
  obiektów tabeli, 697  
apostrofy, 167  
arkusz stylów  
  białe znaki, 42  
  komentarze, 44

  prefiksy przeglądark, 42  
  reguły, 41  
  znaczniki, 41  
atrybuty znacznika link, 35  
automatyczna baza flex, 606  
automatyczne  
  krawędzie, 518  
  linie siatki, 671  
  rozmieszczanie obrazów, 668  
  skalowanie kolumn, 672  
  wypełnianie torów, 644

## B

barwa, 895  
baza flex, 599, 604  
  automatyczna, 606  
  zerowa, 612  
białe znaki, 42, 244  
blok  
  blockquote, 79  
  zawierający element, 263, 504  
budowanie pojemników, 292

## C

cechy fontów, 204  
cienie, 885  
  pojemników, 465  
  tekstu, 242  
CSS, Cascading Style Sheets, 27, 767  
cudzysłowy, 167  
czas, 158  
  trwania przejścia, 812  
częstotliwość, 158

## D

### definiowanie

- długości animacji, 846
- klatek kluczowych, 836
- maski, 907
- rozmiaru strony, 936

### deklaracja, 55, 119

- @font-face, 168, 172
- display:contents, 311

### dekoracja tekstu, 238, 950

### deskryptor, 168

- @counter-style, 748
- additive-symbols, 761
- aspect-ratio, 931
- color, 931
- color-index, 931
- device-aspect-ratio, 931
- device-height, 930
- device-width, 930
- fallback, 756
- font-family, 168
- font-feature-settings, 204, 205
- font-stretch, 200
- font-style, 197
- font-variant, 204
- font-weight, 184
- grid, 932
- max-aspect-ratio, 931
- max-color, 931
- max-color-index, 931
- max-device-aspect-ratio, 931
- max-device-height, 930
- max-device-width, 930
- max-resolution, 931
- max-width, 930
- min-aspect-ratio, 931
- min-color, 931
- min-color-index, 931
- min-device-aspect-ratio, 931
- min-device-height, 930
- min-device-width, 930
- min-monochrome, 931
- min-resolution, 931
- min-width, 930
- monochrome, 931
- negative, 759
- orientation, 932
- pad, 759

### prefix, 752

- range, 755
- resolution, 931
- scan, 932
- speak-as, 765
- src, 168
- suffix, 752
- symbols, 749
- system, 749
- unicode-range, 174
- width, 930

### deskryptory

- cech mediów, 48
- fontów, 173
- mediów, 46
- właściwości mediów, 930

### dobieranie fontów, 211

### dołączanie obiektów do siatki, 652

### dopasowywanie

- fragmentu łańcucha, 73, 75
- jednego słowa, 72
- obiektu, 920
- zawartości torów, 640

### dopełnienie, padding, 315, 318

- elementy liniowe, 325
- elementy zastępowane, 326, 327
- jednostronne, 321
- wartości procentowe, 323

### druk, 935

### drukowanie

- animacji, 881
- przejsć, 832

### drzewo dokumentu, 77, 82

### dynamiczna stylizacja, 101

### dyrektywa

- !important, 877
- @import, 38

### dziedziczenie, 117, 122, 189, 225

- koloru, 383

### dzielenie słów, 247

## E

### efekt

- negatywu, 386
- transformacji typu flat, 792

### ekran, 892, 935

### elastyczne tory siatek, 633

## F

- elastyczność
  - metoda numeryczna, 617
  - paska nawigacji, 569
  - wartość
    - auto, 615
    - initial, 614
    - none, 615
- elastyczny układ treści, 553
- element
  - nav, 265
  - span, 191
  - style, 38
- elementy, 28
  - blokowe, 30, 267
  - formularzy, 382
  - główne, 262
  - interfejsu, 103
  - liniowe, 30, 292, 325, 340, 375
    - zastępowane, 303
  - liniowo-blokowe, 308
  - niezastępowane, 29, 262
    - rozmiar, 520
    - rozmieszczenie, 520
  - szerokość, 316
  - wysokość, 316
  - plywające, 469, 470, 480
    - elipsy, 492
    - kształty typu inset, 491
    - margines kształtu, 500
    - ograniczenia, 478
    - okręgi, 492
    - przycinanie, 495
    - ustawianie, 476, 479
    - wielokąty, 497
    - zapobieganie nakładaniu, 474, 475
  - rodzica i dziecka, 76
  - tabeli, 691
  - wewnętrzne, 287
    - tabeli, 691
  - zastępowane, 29, 262, 276, 306, 326
    - rozmiar, 524
    - rozmieszczenie, 524
- elipsy, 492
  - o zerowej szerokości, 457
- filtrowanie kolorów, 886
- filtry, 883
  - podstawowe, 884
  - SVG, 888
- flexbox, 539
  - linie flex, 554
  - oś główna, 554
  - układanie obiektów, 560
- fonty, 163, 952
  - bezseryfowe, 164
  - cechy, 204
  - dobieranie, 211
  - nieproporcjonalne, 164
  - niestandardowe, 173
  - systemowe, 210
  - seryfowe, 163
  - wagi, 179
  - warianty, 201
- formatowanie
  - tabel, 691
  - w pionie, 277
  - w poziomie, 268
  - wewnętrzne, 291
  - wizualne, 261
- formaty fontów, 171
- fragmenty, 107
- funkcja
  - filter, 884
  - matrix(), 785
  - matrix3d(), 786
  - perspective(), 783, 785
  - repeat(), 644
  - rotate3d(), 780, 782
  - rotateZ(), 779
  - scale3d(), 778
  - scaleY(), 773
  - skew(), 783
  - translate3d(), 778
- funkcje
  - macierzowe, 785
  - obracania, 778
  - przechylania, 782
  - przekształceń, 774
  - skalowania, 777
  - translacji, 775
  - zmiany tempa, 864

## G

generowanie  
  cudzysłowu, 739  
  ikon, 735  
  odmian znaków, 206  
  zawartości blokowej, 735  
  zawartości tekstowej, 734  
glify, 303  
gradienty, 434  
  cykliczne, 461  
  kolory, 436  
  kołowe, 448  
  pozycjonowanie, 451  
  promień, 453  
  rozmiar, 450, 464  
liniowe, 435, 443  
przetwarzanie obrazów, 459  
przycięcie, 438  
uśrednianie kolorów, 464  
grubość znaków, 178  
grupowanie  
  deklaracji, 59  
  selektorów, 57  
  wszystkiego, 60

## H

hiperłącza, 98  
HTTP, 39

## I

identyfikator, 66, 136  
  fragmentu, 107  
indeks  
  dolny, 228  
  górnym, 228  
interfejs użytkownika, 953  
interlinia, 289  
interpolacja wartości właściwości, 829  
iteracja animacji, 847, 878

## J

jasność, 887, 895  
jednostka  
  cał, 138  
  centymetr, 138

  ch, 144  
  ćwiartki milimetra, 139  
  em, 142  
  ex, 142  
  milimetr, 139  
  pica, 139  
  pixel, 139  
  punkt, 139  
  rem, 143  
  vh, 146  
  vmax, 146  
  vmin, 146  
  vw, 145  
jednostki  
  długości, 138, 192, 608  
  bezwzględne, 138  
  względne, 142  
  kątowe, 158  
  procentowe, 609  
  rozdzielczości, 141  
  ułamkowe, 633  
justowanie  
  pojedynczych obiektów, 683  
  wszystkich obiektów, 685

## K

kafelki, 414  
kanały alfa, 885  
kapitaliki, 202  
kaskada, 125  
kaskadowe arkusze stylów, CSS, 27  
kąty, 157  
kerning znaków, 201  
kierunek  
  blokowy, 215  
  liniowy, 215  
kierunki pisania, 549  
klasy, 66  
klatki kluczowe, 836  
  powtarzanie właściwości, 840  
kolejność animacji, 878  
kolor, 149, 379, 895, 949  
  domyślny, 380  
  efekty specjalne, 385  
  elementów formularza, 383  
  dziedziczenie, 383  
  gradientów, 436  
  HSL, 154

- HSLa, 154
- konturu, 366
- nazwane, 149
- obramowania, 335, 382
- pierwszego planu, 379
- pozycjonowanie znaczników, 437
- RGB, 150
- RGB szesnastkowe, 153
- RGBa, 152
- RGBa szesnastkowe, 154
- tabela odpowiedników, 967
- tła, 384
- ustawianie wskaźników, 441
- kolumny, 696, 949
- komentarze CSS, 44
- konfigurowanie animacji, 837
- kontener, 545
- kontrast, 887
- kontury, outline, 315, 364
  - kolor, 366
  - style, 364
  - szerokość, 365
- korygowanie wielkości znaków, 193
- krawędzie odniesienia, 401
- krzywa Béziera, 861
- kształty, 954
  - dodawanie marginesu, 500
  - elementów pływających, 488, 497
  - przycinające, 902
  - typu inset, 491
- kursywa, 207

## L

- liczniki, 741, 953
  - inkrementowanie, 742
  - wykorzystywanie, 743
  - zagnieżdżone, 747
  - zakres, 746
  - zerowanie, 742
- linia bazowa, 306
- linie
  - automatyczne siatki, 671
  - flex, 554
  - gradientu, 443
  - kolumn i rzędów, 652
  - siatki, 626–630, 642, 650
- liniowe elementy wewnętrzne, 287
- liniowy pojemnik wewnętrzny, 290

- listy, 285, 723, 953
  - pozycja znaku wypunktowania, 729, 733
  - przekształceń, 772
  - style, 724, 730
  - układ, 731

## Ł

- łamanie
  - stron, 940, 944
  - wierszy, 302
- łańcuchy znaków, 134
- łączenie
  - animacji, 853
  - CSS i HTML, 33
  - deskryptorów, 176
  - masek, 914
  - pseudoklas, 85

## M

- marginesy, margin, 273, 315, 369
  - elementy liniowe, 375
  - jednostronne, 372
  - kształtu, 500
  - składanie, 372
  - strony, 939
  - ujemne, 273, 283, 374, 481
  - w pionie, 281
  - wartości
    - długości, 370
    - procentowe, 371
- maski, 906
  - łączenie, 914
  - powtarzanie, 911
  - pozycjonowanie, 912
  - przycinanie, 914
  - rodzaje, 919
  - skalowanie, 911
  - w obramowaniach obrazkowych, 919
  - zmiana trybu działania, 909
- maskowanie, 900
- media, 45
  - stronicowe, 934
- metoda nonzero, 498
- mieszanie, 889, 954
  - elementów, 890
  - tła, 896
  - w izolacji, 899

miękkie światło, 893  
mnożenie, 892  
mocowanie tła, 415, 416  
model  
flexbox, 539  
liniowy, 307  
pudełkowy, 316, 678, 951

## N

nadawanie elastyczności  
metoda numeryczna, 617  
wartość auto, 615  
wartość initial, 614  
nagłówek HTTP, 39  
nakładanie, 889, 954  
nakładka, 892  
nasylenie, 887, 895  
nawigacja zakładkowa, 542, 620  
nazwa animacji, 837, 844  
numerowanie, 747  
additive-symbols, 761  
alphabetic, 757  
cyclic, 751  
extends, 763  
fixed, 749  
numeric, 758  
symbolic, 754

## O

obiekty  
anonimowe tabeli, 697  
dopasowywanie, 920  
flex, 545, 554, 560, 582  
cechy, 584  
nadawanie rozmiaru, 610  
proporcjonalne zwężanie, 585  
responsywna zmiana wielkości, 601  
szerokości minimalne, 585  
wartości domyślne, 607  
właściwości, 587  
wyrównanie do linii bazowej, 574  
wyrównanie do początku, 573  
justowanie, 683, 685  
pozycjonowanie, 920  
siatki, 626, 665, 678  
wyrównywanie, 568, 683, 685

obramowanie, border, 315, 327, 950, 951  
całkowity brak, 334  
elementy liniowe, 340  
globalne, 339  
kolory, 335, 381  
komórek tabeli, 704  
odstępny, 705  
składanie, 707  
obrazkowe, 348, 350  
szerokość, 357  
właściwości, 361  
płynne przejścia w narożnikach, 346  
przesuwanie na zewnątrz, 357  
przezroczyste, 336  
składanie układu, 708  
szerokość, 332  
typu inset, 329  
typu outset, 330  
właściwości, 337  
ze stylem, 328  
zaokrąglanie rogów, 341  
obrazy, 136, 953  
dopasowywanie, 426  
gradientów, 459  
odstępny, 410  
powielanie, 360, 413  
powtarzanie, 409, 425  
pozycjonowanie, 409, 426  
przycinanie, 414, 426  
rozmieszczanie, 361  
skalowanie, 410, 413  
w tle, 390  
zasłanianie, 426  
zastosowanie, 390  
zmiana szerokości, 353  
obróć, 158, 770, 778–781  
obszar zawartości elementu, 289, 303  
obszary siatki, 646, 662  
odległości, 138  
odstępny, 235  
między literami, 234  
między słowami, 233  
odsyłacze, 99  
ograniczanie  
szerokości, 509  
zakresu znaków, 174  
okręgi, 492



opóźnianie  
animacji, 850  
przejść, 819  
ostre światło, 893

## P

perspektywa grupowa, 794  
piksel, 140  
pisanki, 164  
pływanie, 469, 472, 483  
podsiatki, 675  
pogrubianie, 182  
pojemnik, 261, 292  
cienie, 465  
dodawanie właściwości, 299, 304  
elementu, 315  
blokowego, 263  
liniowego, 263  
em, 289, 292  
flex, 545, 561, 582  
flexbox, 950  
grupy kolumn, 692  
grupy wierszy, 692  
kolumny, 692  
komórki, 693  
kształtu, 498  
liniowo-blokowy, 263  
przycinający, 902  
siatki, 623  
wewnętrzny, 293  
wiersza, 290, 692  
z odjętym dopełnieniem, 269  
położenie obrazu, 159, 397  
równoważne metody, 398  
wartości długości, 399  
wartości ujemne, 399  
powielanie  
obrazu, 360, 425  
tła, 406  
pozycje listy, 285  
pozycjonowanie, 503, 952  
bezwzględne, 514, 585  
rozmiar elementów, 517  
rozmieszczenie elementów, 517  
gradientów kołowych, 451  
masek, 912  
obiektu, 920

tła, 403  
typu fixed, 530  
typu sticky, 533  
względne, 531  
znaku wypunktowania, 733  
prefiksy przeglądarek, 42  
projektowanie responsywne, 933  
promień gradientu, 453  
przechylenie, 782  
przejścia, 801, 954  
czas trwania, 812  
do początku, 824  
drukowanie, 832  
jako ozdobniki, 832  
ograniczanie, 806  
ograniczanie puli właściwości, 810  
opóźnianie, 819  
tempo krokowe, 817  
tonalne, 435  
ujemne wartości opóźnień, 821  
właściwości, 802  
zdarzenia, 810  
zmiana kierunku, 825  
zmiana tempa, 814  
przekształcanie, 767  
elementy div, 772  
funkcje, 774  
modyfikowanie, 773  
typu flat, 792  
właściwości, 788  
zastępowanie, 773  
przekształcenie  
obrót, 770, 778  
przechylenie, 782  
przesunięcie, 505, 775, 788  
skalowanie, 777  
przepływ siatki, 666  
przesunięcie, 505, 775  
obrazu, 401  
punktu początkowego, 788  
przetwarzanie obrazów gradientów, 459  
przezroczystość obrazu, 499  
przyciemnianie, 891  
przycinanie, 900  
gradientu, 438  
maski, 914  
tła, 387  
zawartości, 511

pseudoelement  
::after, 115  
::before, 114  
::first-letter, 112, 624  
::first-line, 114, 624

pseudoklasa  
:active, 100  
:checked, 102  
:default, 102, 104  
:disabled, 102  
:empty, 86  
:enabled, 102  
:first-child, 90  
:first-of-type, 92  
:focus, 100  
:hover, 100  
:indeterminate, 102  
:in-range, 102, 106  
:invalid, 102, 105  
:lang, 108  
:last-child, 91  
:link, 98  
:nth-child(), 93  
:only-child, 87  
:only-of-type, 89, 93  
:optional, 102, 105  
:out-of-range, 102  
:read-only, 102, 107  
:read-write, 102, 107  
:required, 102, 105  
:root, 86  
:target, 107  
:valid, 102, 105  
:visited, 98  
not(), 109  
nth-child(), 93

pseudoklasy, 84  
dotyczące opcji domyślnej, 104  
dynamiczne, 97  
działań użytkownika, 100  
hiperłączy, 98  
opcjonalności, 105  
poprawności, 105  
stanu interfejsu, 102  
strukturalne, 85  
zakresu, 106  
zmienności, 107  
pudełka elementu, 261

punktory, 723  
ciągi znaków, 726  
obrazkowe, 726  
w formie gradientów, 729

## R

reguła, 41  
@keyframes, 836  
display: none, 878  
reguły  
tworzenia stylów, 53  
wstawiania obiektów, 698  
wypełniania kształtów przycinających, 905  
replikowanie wartości, 320  
responsywna zmiana wielkości, 601  
rodzaje masek, 919  
rodziny fontów, 163, 165  
rogi obramowań, 341  
rozciąganie znaków, 198  
rozjaśnianie, 891, 894  
rozmiar, 188  
elementów niezastępowanych, 520  
elementów pozycjonowanych bezwzględnie, 517  
elementów zastępowanych, 524  
gradientów kołowych, 450  
obiektu flex, 607, 610  
strony, 936  
tabeli, 712  
tekstu, 185  
bezwzględny, 186  
względny, 188  
rozmieszczanie  
elementów, 517  
na osi Z, 526  
niezastępowanych, 520  
pozycjonowanych bezwzględnie, 517  
zastępowanych, 524  
linii siatki, 628  
obrazów, 361  
znaczników koloru, 439  
różnica, 891

## S

selektor uniwersalny, 59  
selektory, 53  
attributów, 67  
elementów, 56

- identyfikatorów, 66
- klas, 62
- klatek kluczowych, 838
- kontekstowe, 78, 79
- pseudoelementów, 112
- pseudoklas, 84
- siatka, 623
  - automatyczne
    - linie, 671
    - rozmieszczanie obrazów, 668
    - wypełnianie torów, 644
  - dołączanie obiektów, 652
  - dopasowywanie zawartości torów, 640
  - komórka, 627
  - linie, 630
  - linie kolumn i rzędów, 652
  - metody wyrównywania, 680
  - nakładanie się obiektów, 665
  - nazwy linii, 650
  - nazwy obszarów, 651
  - niejawna, 659
  - obiekty, 626
  - obręb pojemnika, 632
  - obsługa błędów, 661
  - obszar, 627, 646
  - odstęp między torami, 676
  - powtarzanie linii, 642
  - pozycjonowanie bezwzględne, 681, 682
  - przepływ, 666
  - rozmieszczanie linii, 628
  - skalowanie torów, 639
  - tory elastyczne, 633
  - tory o stałej szerokości, 629
  - tworzenie, 631
  - tworzenie odstępów, 676
  - wartości
    - automatyczne, 682
    - justowania, 683
    - wyrównywania, 683
  - wizualizacja rodzajów przepływu, 669
  - wyrównywanie w pionie, 688
  - zastosowanie obszarów, 662
- siatki
  - liniowe, 624
  - zwykłe, 624
- skalowanie, 777
  - maski, 911
  - obrazów, 419
  - torów siatki, 639
  - wysokości wiersza, 298
- składanie, 283
  - marginiesów, 372, 625
  - marginiesów w pionie, 281
  - obramowania, 709
- skrótowa właściwość
  - flex, 613
  - grid, 673
  - mask, 917
  - transition, 822
- skrótowe właściwości rzędów i kolumn, 656
- skrót, 210
- słowa kluczowe globalne, 131
- słowo kluczowe, 55, 131
  - all, 133
  - content, 604
  - currentColor, 157
  - inherit, 132
  - initial, 132
  - medium, 191
  - transparent, 157
  - unset, 133
- sortowanie według
  - kolejności, 127
  - specyficzności, 127
  - wagi oraz pochodzenia, 126
- specyficzność, 117
  - atrybutów, 120
  - liniowych stylów wewnętrznych, 121
  - selektora uniwersalnego, 120
  - selektorów identyfikatorów, 120
- stała szerokość znaków, 191
- strony, 953
  - definiowanie rozmiaru, 936
  - dopełnienia, 939
  - łamania, 940, 944
  - marginiesy, 939
  - pozycjonowanie elementów, 947
  - typy, 939
- struktura
  - dokumentu, 76
  - drzewa dokumentu, 77
  - reguł, 41
- styl 3D, 791
- style
  - jednostronne, 331
  - konturów, 364

style  
list, 724, 730  
obramowania, 329  
responsywne, 933  
wewnętrzne, 40  
wydruków, 935  
zależne od medium, 925  
znaków, 195  
stylizacja  
pierwszego wiersza, 113  
pierwszej litery, 112  
szerokość, 508  
elementu, 316  
konturu, 365  
minimalna, 585  
obramowania, 332  
obramowania obrazkowego, 357  
obrazu, 353  
tabeli, 712

## Ś

ściemnianie, 894  
światło  
miękkie, 893  
ostre, 893

## T

tabele, 691, 952  
anonimowe obiekty, 697  
caption-side, 703  
elementy, 691  
elementy wewnętrzne, 691  
formatowanie, 691  
kolumny, 696  
obramowanie komórek, 704  
pierwszeństwo wierszy, 696  
podpisy, 702  
składanie obramowania komórek, 707  
składanie układu obramowania, 708  
szerokość, 712  
tworzenie, 692  
warstwy, 701  
wartości wyświetlania, 693  
wstawianie obiektów, 698  
wygląd, 691  
wyrównanie zawartości komórek, 719, 720  
wysokość, 718

tabulatory, 246  
tekst, 949  
anonimowy, 289  
cień, 242  
dekoracja, 238  
dopasowywanie do rodzica, 221  
określanie kierunku, 257  
transformacja, 236  
tryby pisania, 253  
wcięcia, 216  
właściwości, 215  
wyjustowany, 221  
wyrównanie w pionie, 227  
wyrównywanie, 218  
zawijanie, 252  
zmiana orientacji, 256  
tempo krokowe, 817  
tło, 384, 950  
dopasowywanie, 423  
kolejność warstw, 431  
kolor, 384  
mocowanie, 415  
położenie, 394  
powielanie, 406  
prycinanie, 387  
skalowanie obrazów, 419  
uzupełnianie brakujących wartości, 432  
wiele obrazów, 429  
wyrównanie, 418  
zasłanianie, 423  
zmiana obszaru pozycjonowania, 403  
tor siatki, 629  
automatyczne wypełnianie, 644  
elastyczny, 633  
dopasowywanie zawartości, 640  
uwzględniające treść, 637  
transformacja, *Patrz* przekształcenie  
transformacja tekstu, 236  
translacja, 775, 776  
tryb  
mieszania, 892  
pisania, 253  
tworzenie  
kształtu, 488  
pojemnika siatki, 623  
siatki, 631  
stylów, 53  
tabel, 692  
tylne ścianki, 797

typy  
mediów, 46  
pozycjonowania, 503  
wartości, 48

## U

układ  
elastyczny, 553  
flexbox, 602  
listy, 731  
normalny, 262  
o automatycznej szerokości, 714  
o stałej szerokości, 712  
siatkowy, 623  
tabelaryczny, 691  
wierszy, 287  
współrzędnych, 767  
kartezjański, 768  
sferyczny, 769  
układanie obiektów flex, 560  
ustawianie  
szerokości, 272  
tła, 393

## W

wagi  
fontów, 179  
lżejsze, 184  
warianty fontów, 201  
warstwy, 687  
tabeli, 701  
wartości  
atrybutów, 148  
całkowite, 137  
długości, 370, 398  
flow, 310  
liczbowe, 137  
obliczeniowe, 147  
procentowe, 138, 188, 231, 275, 323, 396  
procentowe w pionie, 279  
ułamkowe, 138  
wartość  
auto, 272  
typu <ratio>, 932  
typu <resolution>, 932  
ważne deklaracje, 121  
ważność, 121

wątek UI, 879  
wcięcia tekstu, 216  
wdowy i sieroty, 943  
widoczność elementu, 513  
wielkość znaków, 75, 193  
wielokąty, 497  
wiersz, 696  
konstruowanie, 224  
łamanie, 302  
skalowanie wysokości, 298  
wysokość, 223  
zarządzanie wysokością, 296  
zawijanie, 247  
właściwości, 49  
animowane, 949  
dające się animować, 841  
klatek kluczowych, 840  
marginesów jednostronnych, 372  
nieanimowane, 842  
niestandardowe, 160  
obiektów flex, 587  
obramowania, 337  
obramowania obrazkowego, 361  
pionowe, 278  
pojemników, 299, 304  
poziome, 270  
przejsć, 802, 954  
przekształceń, 788  
przesunięcia, 505  
pseudoelementów, 114  
rzędów i kolumn, 656  
tekstu, 215  
właściwość, 957–966  
align-content, 577  
align-items, 569, 685  
align-self, 576, 683  
animation, 875  
animation-delay, 850  
animation-direction, 849  
animation-duration, 846  
animation-fill-mode, 872  
animation-iteration-count, 847  
animation-name, 844  
animation-play-state, 872  
animation-timing-function, 860, 868  
backface-visibility, 797  
background, 427  
background-attachment, 415  
background-blend-mode, 897

## właściwość

- background-clip, 388
- background-color, 384
- background-image, 391
- background-origin, 403
- background-position, 394
- background-repeat, 406
- background-size, 419
- border, 339
- border-bottom, 337
- border-bottom-color, 336
- border-bottom-left-radius, 347
- border-bottom-right-radius, 347
- border-bottom-style, 331
- border-bottom-width, 332
- border-collapse, 704
- border-color, 335
- border-image, 361
- border-image-outset, 358
- border-image-repeat, 359
- border-image-slice, 349
- border-image-source, 348
- border-image-width, 354, 357
- border-left, 337
- border-left-color, 336
- border-left-style, 331
- border-left-width, 332
- border-radius, 341
- border-right, 337
- border-right-color, 336
- border-right-style, 331
- border-right-width, 332
- border-spacing, 705
- border-style, 328
- border-top, 337
- border-top-color, 336
- border-top-left-radius, 347
- border-top-right-radius, 347
- border-top-style, 331
- border-top-width, 332
- border-width, 332
- bottom, 505
- box-shadow, 465
- box-sizing, 267
- clear, 484
- clip-path, 901
- clip-rule, 906
- color, 380
- content, 737
- counter-increment, 742
- counter-reset, 742
- direction, 258
- display, 30, 264, 311, 694, 878
- empty-cells, 707
- filter, 883
- flex, 587, 613
- flex-basis, 604
- flex-direction, 545
- flex-flow, 553
- flex-grow, 588
- flex-shrink, 594
- flex-wrap, 552, 559
- float, 469, 585
- font-family, 164
- font-kerning, 201
- font-size, 186
- font-size-adjust, 193
- font-stretch, 199
- font-style, 195
- font-synthesis, 206
- font-variant, 202
- font-weight, 178
- grid, 673
- grid-area, 662
- grid-auto-columns, 671
- grid-auto-flow, 666
- grid-auto-rows, 671
- grid-column, 657
- grid-column-end, 652
- grid-column-gap, 676
- grid-column-start, 652
- grid-gap, 678
- grid-row, 657
- grid-row-end, 652
- grid-row-gap, 676
- grid-row-start, 652
- grid-template-areas, 646
- grid-template-columns, 628
- grid-template-rows, 628
- height, 317, 508
- hyphens, 248
- isolation, 899
- justify-content, 561, 567
- justify-items, 685
- justify-self, 683
- left, 505
- letter-spacing, 234
- line-break, 251

- line-height, 223, 225
- list-style, 730
- list-style-image, 727
- list-style-position, 729
- list-style-type, 724
- margin, 369
  - margin-bottom, 372
  - margin-left, 372
  - margin-right, 372
  - margin-top, 372
- mask, 917
  - mask-clip, 915
  - mask-composite, 915
  - mask-image, 907
  - mask-mode, 909
  - mask-origin, 914
  - mask-position, 913
  - mask-repeat, 912
  - mask-size, 911
  - mask-type, 919
- max-height, 510
- max-width, 510
- min-height, 509
- min-width, 509
- mix-blend-mode, 890
- object-fit, 920
- object-position, 922
- order, 618
- orphans, 943
- outline, 367
  - outline-color, 366
  - outline-style, 364
  - outline-width, 365
- overflow, 512
  - overflow-wrap, 252
- padding, 318
  - padding-bottom, 322
  - padding-left, 322
  - padding-right, 322
  - padding-top, 322
- page, 940
  - page-break-after, 941
  - page-break-before, 941
  - page-break-inside, 943
- perspective, 794
  - perspective-origin, 796
- position, 503
- quotes, 739
- right, 505
- shape-image-threshold, 499
- shape-margin, 500
- shape-outside, 488
- size, 937
- table-layout, 712
- tab-size, 247
- text-align, 219
  - text-align-last, 222
- text-indent, 216
- text-orientation, 257
- text-rendering, 241
- text-shadow, 243
- text-transform, 236, 238
- top, 505
- transform, 771
  - transform-origin, 788
- text-rendering, 241
- transform-style, 791
- transition, 822
  - transition-delay, 819
  - transition-duration, 812
  - transition-property, 807
  - transition-timing-function, 814
- unicode-bidi, 258
- vertical-align, 227, 625, 719
- visibility, 513
- white-space, 244, 246
- widows, 943
- width, 317, 508
- word-break, 249
- word-spacing, 233
- word-wrap, 252
- writing-mode, 253
- z-index, 526, 689
- wskazniki koloru, 441
- współczynnik
  - kurczenia, 598
  - wzrostu, 591
- wstawianie adresów URL, 738
- wybieranie
  - atrybutów, 68
  - co n-tego dziecka, 93
  - co n-tego elementu danego typu, 96
  - elementów dzieci, 81
  - elementów rodzeństwa, 82, 83
  - elementu głównego, 86
  - pierwszego i ostatniego dziecka, 90
  - pierwszego i ostatniego elementu, 92
  - pustych elementów, 86

- wybijanie
    - unikatowych dzieci, 87
    - według wartości atrybutu
      - częściowej, 71
      - dokładnej, 69
  - wydajność animacji, 855
  - wykluczanie, 891
  - wykorzystywanie wartości auto, 271
  - wypełnianie
    - obszarów obramowania, 355
    - wielokątów, 498
  - wypływanie, 511
  - wyrównanie, 235
    - center, 564
    - do dołu, 229
    - do góry, 230
    - do linii bazowej, 228, 574
    - do początku, 573
    - do środka, 231
    - flex-end, 564
    - flex-start, 563
    - linii flex, 581
    - space-around, 566
    - space-between, 565
    - space-evenly, 567
  - wyrównywanie
    - elementów liniowych, 223
    - obiektów flex, 568
    - ostatniego wiersza, 222
    - pojedynczych obiektów, 683
    - siatki w pionie, 688
    - tekstu, 218
    - tekstu w pionie, 227
    - treści, 561, 577
    - w linii, 215
    - w pionie, 294
    - według wartości odległości, 232
    - wszystkich obiektów, 685
    - zawartości komórek, 719, 720
  - wysokość, 508
    - automatyczna, 280
    - elementu, 316
    - tabeli, 718
    - wiersza, 209, 223, 225, 296, 298
  - wyświetlanie
    - elementów, 29, 264
    - tabeli, 693
  - wzorce numerowania
    - addytywne, 761
    - alfabetyczne, 757
    - cykliczne, 751
    - definiowanie, 747
    - liczbowe, 758
    - rozszerzanie, 763
    - stałe, 749
    - symboliczne, 754
    - wymawianie, 764
- ## Z
- zagnieżdżone liczniki, 747
  - zamiana numeracji na mowę, 764
  - zaokrąglanie rogów, 341, 346
  - zapytania
    - o media, 45
    - podstawowe, 925
    - złożone, 927
    - o właściwości, 49
  - zasłanianie tła, 423, 424
  - zastosowanie obszarów, 662
  - zawartość generowana, 733, 953
    - określanie zawartości, 736
    - wstawianie, 734
    - wstawianie wartości atrybutów, 738
  - zawijanie
    - linii flex, 551
    - tekstu, 252
    - wierszy, 247
  - zaznaczanie, 103
  - zdarzenia
    - związane z animacjami, 852, 880
    - związane z przejściami, 810
  - zdarzenie
    - animationend, 881
    - animationiteration, 881
    - animationstart, 880
  - zmiana
    - kierunku przejść, 825
    - obszaru pozycjonowania tła, 403
    - orientacji tekstu, 256
    - perspektywy, 794
    - szerokości obrazu, 353
    - tempa animacji, 860
    - tempa przejść, 814
    - trybu działania maski, 909
  - znacznik link, 33
  - znaczniki kolorów, 437, 453



## znak

&gt;, 81

cudzysłowu, 739

dwukropka, 85

kropki, 63

krzyżyka, 66

ukośnika, 56

wypunktowania, 723, 728

## znaki

automatyczne korygowanie wielkości, 193

generowanie, 206

kerning, 201

rozciąganie, 198

style, 195

żywa pagina, 946



# PROGRAM PARTNERSKI

— GRUPY HELION —

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion**

## CSS. Poznaj i stosuj najnowsze specyfikacje!

CSS służy do opisywania sposobu prezentowania treści internetowych wyświetlanych na ekranach, na potrzeby druku czy syntezy mowy. Jest obsługiwany przez wszystkie przeglądarki i urządzenia, takie jak smartfony, komputery, gry wideo, telewizory, zegarki, kioski multimedialne czy konsole samochodowe. CSS pozwala zwiększyć wygodę użytkownika, przyspieszyć proces projektowania, uniknąć potencjalnych błędów, a także urozmaicić i ożywić aplikacje. Język ten wciąż się rozwija i od wielu lat stanowi niezbędny element warsztatu każdego profesjonalnego projektanta stron internetowych.

Ta książka to kompleksowy i zaktualizowany przewodnik po implementacji CSS. Zawiera obszerną analizę najnowszych specyfikacji CSS. Przedstawiono tu szereg istotnych zagadnień i wyrafinowanych technik stylizowania stron oraz poprawiania dostępności treści – wykorzystanie tych sposobów pomaga zaoszczędzić czas i wysiłek. Ta publikacja powstała z myślą o profesjonalnych projektantach stron, niemniej jednak bardzo przyda się osobom, które zaczynają naukę CSS i chcą od razu wykorzystywać jego najlepsze cechy. Znalazł się tu szczegółowy opis wszystkich funkcji CSS powszechnie obsługiwanych przez przeglądarki, w tym funkcji, które w czasie pisania tej książki były przygotowywane do wprowadzenia.

### Wybrane zagadnienia:

- Selektory, specyficzność i kaskada
- Właściwości tekstu, dopełnienia, marginesy, tła i gradienty
- Układy, w tym *flexbox* i *grid*
- Przekształcenia 2D i 3D, przejścia i animacje
- Filtry: mieszanie, przycinanie i maskowanie

**Eric A. Meyer** – to niekwestionowany autorytet w dziedzinie HTML, CSS i standardów internetowych. Założył firmę Complex Spiral Consulting. Do najważniejszych inicjatyw, w które się zaangażował, należą ruch na rzecz promowania mikroformatów oraz seria konferencji An Event Apart.

**Estelle Weyl** – jest promotorką otwartych standardów internetowych i inżynierem społeczności. Od 1999 roku zajmuje się projektowaniem stron internetowych, zwracając uwagę na ich zgodność ze standardami i dostępność. Często zabiera głos na konferencjach na całym świecie.

**Helion**  
  
 helion.pl  
 HELION SA  
 ul. Kościuszki 1c  
 44-100 Gliwice  
 tel.: 32 230 98 63  
 helion@helion.pl

Sprawdź nasze szkolenia!

SZKOLENIA  
  
 AKADEMIA IT & BUSINESS  
 WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI  
 Sięgnij po więcej! ▶



ISBN 978-83-283-4083-1



9 788328 340831