



Dziedziczenie

- Dziedziczenie sprawia, że niektóre style rodzica są przenoszone na dziecko (nie wszystkie).
- Np. gdy ustawimy `font-size`` i `color`` dla `div`` to wszystkie* elementy wewnątrz otrzymają te style. *element `a`` nie przejmie koloru
- Niektóre właściwości nie są dziedziczone np. `height``. Szczegóły możemy znaleźć np. w dokumentacji MDN.



Dziedziczenie

Tabela dla właściwości `width`

<u>Initial value</u>	auto
Applies to	all elements but non-replaced inline elements, table rows, and row groups
<u>Inherited</u>	no
Percentages	refer to the width of the containing block
<u>Computed value</u>	a percentage or auto or the absolute length
Animation type	a length , percentage or calc();



Dziedziczenie

- Kontrolowanie dziedziczenia
 - ``initial`` wartość resetująca daną właściwość do jej początkowej wartości
 - ``inherit`` wartość odwołująca się do wartości rodzica (dziedziczenie)



Klasyfikacja ważności selektorów

- Kolejność ważności
 1. Selektory odwołujące się do identyfikatorów
 2. Selektory odwołujące się do klas
 3. Selektory odwołujące się do elementów
- Podczas liczenia ważności ilość selektorów nie ma znaczenia tj. 1 identyfikator jest ważniejszy od wielu class, a jedna klasa od wielu innych selektorów (np. ``div p``).



Klasyfikacja ważności selektorów

- Przy dziedziczeniu ważność nie ma zastosowania, tj. reguła dziecka jest zawsze ważniejsza.
- Wartość `!important` sprawia, że dana reguła jest ważniejsza od innych mimo np. słabszego selektora. Powinniśmy ją stosować tylko w ostateczności, ponieważ utrudnia utrzymywanie kodu.



2. Popularne pseudoselektory i pseudoelementy



Pseudoselektory (pseudoklasy)

Selektor	Opis
<code>:active</code>	Aktywny (naciśnięty) link
<code>:focus</code>	Element, na którym znajduje się aktywny kursor
<code>:hover</code>	Element, na który najechano myszą
<code>:link</code>	Nieodwiedzone linki
<code>:visited</code>	Odwiedzone linki
<code>:first-child</code>	Pierwszy potomny element danego elementu
<code>:first-of-type</code>	Pierwszy element danego typu w elemencie nadrzędnym



Pseudoselektory (pseudoklasy)

Selektor	Opis
<code>:last-child</code>	Ostatni potomny element danego elementu
<code>:last-of-type</code>	Ostatni element danego typu w elemencie nadrzędnym
<code>:nth-child(n)</code>	N-ty element w elemencie nadrzędnym
<code>:not(selector)</code>	Elementy, które nie są elementami danego typu



Pseudoelementy

Selektor	Opis
<code>::before</code>	Wstawianie jakiejś wartości przed danym elementem
<code>::after</code>	Wstawianie jakiejś wartości za danym elementem
<code>::first-letter</code>	Wybranie pierwszej litery w tekście
<code>::first-line</code>	Wybranie pierwszej linii w tekście



Pseudoelementy

- Pseudo elementy `::before` oraz `::after` potrzebują właściwości `content`, aby były wyświetlane. Może być nawet pusta tj. `content: ""`.
- Więcej: [CSS-tricks.com - pseudo-class-selectors](https://css-tricks.com/pseudo-class-selectors)



info Share
ACADEMY



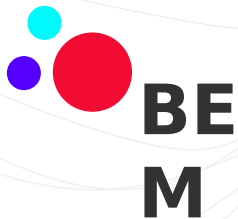
Custom properties

- Custom properties (określane jako zmienne CSS) pozwalają na uproszczenie utrzymywania kodu poprzez zapisanie wartości, a następnie odwoływanie się do niej.
- Przykładowo jeżeli mamy użyte `color: red` 10 razy i później chcemy zmienić na `blue` to musimy to robić 10 razy, przy pomocy zmiennej możemy to zrobić 1 raz.



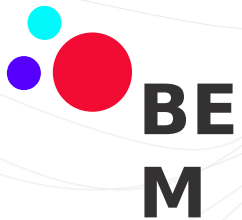
Custom properties

```
div {  
  --font-kolor: ■ #ff99dd;  
}  
  
p {  
  color: var(--font-kolor);  
}  
  
div {  
  color: var(--font-kolor);  
}  
  
:root {  
  --font-color: ■ rgb(80, 120, 180);  
}  
  
p {  
  color: var(--font-color);  
}
```



BEM (Blocks, Elements, Modifiers) - metodyka/podejście do tworzenia kodu CSS w sposób modularny. Oparta jest ona na poniższym podziale elementów.

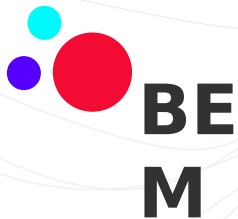
- Bloki - np. menu
- Elementy - części bloku np. linki w menu
- Modyfikatory - warianty elementów np. nieaktywny link



Na podstawie swoich założeń BEM definiuje konwencję nazewnictwa klas.

- .block
- _
element
- --modifier

```
.block {  
  font-size: 1.5rem;  
}  
  
.block__element {  
  width: 20%;  
}  
  
.block__element--modifier {  
  color: ■ red;  
}
```



Zasady BEM można uprościć w dwóch punktach

- Używaj tylko klas - nie należy używać nazw elementów i id
- dla klas bloków i elementów nie należy używać selektorów potomka (jest to dozwolone dla modyfikatorów bloku)

Więcej: [Strona BEM](#)

- Wybranie jednego z gotowych szablonów ze strony: <https://html5up.net/> i modyfikacja wybranych jego cech, nadpisując style własnym arkuszem

