



# Podstawy CSS

Mateusz Kowalski

# Przypomnienie selektorów

Selektory - definiują elementy, do których chcemy przypisać własności CSS

# Id

1. Nadajemy elementowi za pomocą atrybutu Id
2. Określony element może posiadać tylko jeden Id
3. Określone Id może być posiadane tylko przez jeden element



```
<nav id='main-nav'>
  <ul>
    <li>about us</li>
    <li>projects</li>
    <li>where</li>
    <li>contact us</li>
  </ul>
</nav>
```



```
#main-nav {
  width: 300px;
  height: 70px;
  background-color: lightblue;
}
```

# Klasa

1. Nadajemy elementowi za pomocą atrybutu class
2. Określony element może posiadać wiele klas
3. Wiele elementów może posiadać tą samą klasę



```
1 <div class='introduction-text'>
2     Winter came early this year...
3 </div>
```



```
1 .introduction-text {
2     color: purple;
3     border: 1px solid red;
4 }
```

# Znacznik

1. Możemy stylować określone elementy bo nazwie ich znacznika
2. Stylowanie znacznika `<li>` sprawi, że wszystkie znaczniki `<li>` otrzymają te style

```
1 <ul>
2   <li>about us</li>
3   <li>projects</li>
4   <li>where</li>
5   <li>contact us</li>
6 </ul>
```

```
1 li {
2   color: red;
3   border-bottom: 1px solid green;
4 }
```

# CSS w naszym pliku HTML

Sposoby załączania stylów na naszej stronie

# Style inline

1. Nadajemy je elementowi przy pomocy atrybutu style
2. Stosuje się w bardzo specyficznych przypadkach np. Mailing'u



```
1 <div style='background-color: lightblue; border: 2px solid green'>
2     Box with blue background and green border
3 </div>
```

# Znacznik style

1. Umieszczamy go w head naszego dokumentu
2. Pozwala nadawać style selektorom w taki sam sposób jak plik.css
3. Przy dużej ilości styli zmniejsza czytelność dokumentu

```
1 <head>
2   <meta charset="UTF-8">
3   <title>Style</title>
4   <style>
5       .blue-box {
6           width: 50px;
7           height: 50px;
8           background-color: lightblue;
9           border: 2px solid green;
10      }
11  </style>
12 </head>
13 <body>
14   <div class='blue-box'>
15       Box with blue background and green border
16   </div>
17 </body>
```



# Plik CSS

1. Style znajdują się w osobnym pliku z rozszerzeniem .css
2. Plik załączany jest do pliku html przy pomocy znacznika <link> umieszczonego w head
3. Najpopularniejsza i zalecana forma



```
1 <head>
2     <meta charset="UTF-8">
3     <title>Style</title>
4     <link rel="stylesheet" href='main.css'>
5 </head>
6 <body>
7 <div class='blue-box'>
8     Box with blue background and green border
9 </div>
10 </body>
```



```
1 .blue-box {
2     width: 50px;
3     height: 50px;
4     background-color: lightblue;
5     border: 2px solid green
6 }
```

# Kaskadowość CSS

Zbiór zasad jakimi kieruje się język CSS

# Źródło pochodzenia

1. Jeżeli dwa selektory mają taką samą siłę to wygrywa ten, który znajduje się niżej w dokumencie
2. Na poziomie podstawowym musicie wiedzieć, że:  
Style liniowe > znacznik style > plik.css

Codepen: [miejsce w dokumencie](#)

Codepen: [pochodzenie](#)

# Important

1. Po wartości określonego stylu można umieścić flagę !important
2. Flaga !important sprawia, że styl wygra ze wszystkimi innymi w tym dokumencie
3. !important pokonuje również style ze wszystkich innych warstw, niezależnie od ich pochodzenia czy specyficzności

Codepen: Important

# Specyficzność

1. Określa siłę danego selektora w stosunku do innych
2. Na poziomie podstawowym musicie jedynie wiedzieć, że:  
style inline > Id > Klasa > znacznik
3. Poziom specyficzności określa się od 1 (znacznik) do 1000 (styl inline)

Codepen: [specyficzność](#)

# Dziedziczenie

1. Niektóre style nadane elementowi (rodzicowi), będą też nadawane elementom w jego środku (dzieciom)
2. Dziedziczenie wartości można kontrolować przez nadawanie odpowiednich wartości stylom
3. Dziedziczone cechy można nadpisywać w stylach elementów dziecka

Codepen: dziedziczenie



# Jednostki i kolory

# Jednostki

**Absolutne:** niezależne od otoczenia

1. Px (pixel) - jest zawsze takiej samej wielkości

**Relatywne:** zależne od otoczenia

1. Em - zależne od rozmiaru czcionki w danej elemencie
2. Rem - zależny od rozmiaru czcionki dokumentu (określana w stylach znacznika body)
3. Procenty (%) - zależne od rozmiaru rodzica lub szerokości ekranu
4. Vw - zależne od szerokości ekranu (viewport width)
5. Vh - zależne od wysokości ekranu (viewport height)

Codepen: [jednostki](#)



# Kolory

## Rodzaje kolorów:

1. Nazwa - wywoływanie koloru po nazwie słownej np. Yellow, lightyellow
2. RGB - kolor podawany jest jako mix kolorów: czerwonego, niebieskiego i zielonego. Każdy kolor podawany jest osobno, w skali od 0 do 255.
3. RGBa - RGB z czwartą wartością: opacity - określa przezroczystość kolorów
4. HSL - podaje się wartości: barwy, nasycenia, jasności
5. HSLa - HSL z czwartą wartością: opacity
6. HEX - każdy kolor posiada swój sześciocyfrowy kod, którym jest wywoływany

Codepen: [kolory](#)

# Stylowanie tekstu

# Font-family

1. Określa listę nazw rodzin czcionek, którymi ostyleowany zostanie tekst w określonym elemencie strony
2. Czcionki ustawione są pod względem ważności
3. Jeśli żadna z czcionek nie jest obsługiwana przez przeglądarkę lub się nie ładuje, to wykorzystana zostanie bazowa czcionka przeglądarki
4. Nazwy czcionek zawierające przerwy powinny być umieszczone w cudzysłowach



```
1 .page-container {  
2   font-family: Courier, "Lucida Console", monospace  
3 }
```

Codepen: [text-styling](#)

# Font-size

1. Określa wielkość tekstu w naszym elemencie
2. Jest dziedziczony z samego root dokumentu (bazowy font-size przeglądarek to 16px)
3. Najczęściej definiowany przy użyciu px, rem, em



```
1 .page-container {  
2   font-family: Courier, "Lucida Console", monospace;  
3   font-size: 24px  
4 }
```

Codepen: [text-styling](#)

# Font-style

1. Służy do pochylenia tekstu
2. Możliwe wartości to: normal, italic, oblique
3. Wartość normal wymusza powrót czcionki do wersji bez pochylenia
4. Wartość italic włącza pochyloną wersję załączonej czcionki lub symuluje pochylenie ręcznie
5. Wartość oblique wymusza symulowanie pochylenia na czcionce

```
1 .page-container {  
2   font-size: 27px;  
3   font-style: italic;  
4 }
```

Codepen: [text-styling](#)

# Font-weight

1. Definiuje grubość liter w tekście
2. Możliwe wartości słowne to: light, normal, bold, extrabold, black
3. Możliwe wartości liczbowe: 100, 200, 300, 400, 500, 600, 700, 800, 900
4. Wartości liczbowe zawsze muszą być pełnymi setkami



```
1 .page-container {  
2   font-family: Courier, "Lucida Console", monospace;  
3   font-size: 24px;  
4   font-weight: 700;  
5 }
```

Codepen: [text-styling](#)

# Text-transform

1. Przekształca określony font w jedną z wartości
2. Możliwe wartości:
  1. None: blokuje zmienianie się tekstu
  2. Uppercase: zmienia wszystkie litery na wielkie
  3. Lowercase: przeciwieństwo uppercase
  4. Capitalize: wszystkie pierwsze litery słów są wielkie
  5. Full-width: wszystkie znaki mają tą samą szerokość



```
1 .page-container {  
2   font-size: 27px;  
3   text-transform: uppercase; /* none, lowercase, capitalize, full-width */  
4 }
```



# Text decoration

1. Dekoruje font w jedną z wartości
2. Możliwe wartości:
  1. None: blokuje wszelkie dekoratory tekstu
  2. Underline: tworzy linię pod tekstem np. Linki na różnych stronach
  3. Overline: tworzy linię nad tekstem
  4. Line-through: tworzy linię na tekście/przekreśla go



```
1 .page-container {  
2   font-size: 27px;  
3   text-decoration: underline; /* none, overline, line-through */  
4 }
```



# Text-shadow

1. Tworzy cień za tekstem
2. Przyjmuje cztery wartości liczbowe:
  1. Pionowe przesunięcie cienia
  2. Poziome przesunięcie cienia
  3. Rozproszenie się cienia
  4. Kolor cienia (przyjmuje czarny jako bazowy)

```
1 .page-container {  
2   font-size: 27px;  
3   text-shadow: 10px 7px 4px blue;  
4 }
```

Codepen: [text-styling](#)

# Text-align

1. Decyduje o układaniu się tekstu wewnątrz elementu
2. Przyjmuje wartości:
  1. Left
  2. Right
  3. Center
  4. Justify (układa tekst tak by wszystkie linijki były tej samej długości - odradzany!)



```
1 .page-container {  
2   font-size: 27px;  
3   text-align: center /* left, right, justify */  
4 }
```

Codepen: [text-styling](#)

# Line-height


1. Definiuje wysokość każdej linii tekstu
2. Przyjmuje wartości:
  1. Jednostki relatywne
  2. Jednostki absolutne
  3. Wartości bezjednostkowe np. 1 czy 1,5
3. Wartość bezjednostkowa jest mnożnikiem font-size danego elementu np 1.2 X 18px

```
1 .page-container {  
2   font-size: 27px;  
3   line-height: 1.3 /* 1.3 * 27px */  
4 }
```

Codepen: [text-styling](#)

# Letter-spacing / Word-spacing

1. Definiują odstępy pomiędzy literami/słowami w tekście
2. Przyjmują wartości:
  1. Jednostki absolutne
  2. Jednostki relatywne



```
1 .page-container {  
2   font-size: 27px;  
3   letter-spacing: 3px;  
4   word-spacing: 1rem;  
5 }
```

Codepen: [text-styling](#)



# Tło elementów

Kolor | Gradient | Obrazek | Kombinacje

# Kolor

1. Aplikujemy za pomocą własności **background-color**
2. Domyślnie, background-color ma ustawiony wartość *transparent*
3. Przyjmuje wszystkie dostępne rodzaje kolorów (nazwa, RGB, RGBa, HEX, HSL)



```
1 .tank-article {  
2   background-color: rgba(10, 34, 200, 0.5)  
3 }
```

# Gradient Liniowy

1. Ustawiamy za pomocą własności **background**
2. Jako wartość podajemy funkcje CSS o nazwie **linear-gradient()**
3. Przyjmuje ona minimum dwie wartości:
  1. Kierunek (to left, to right, to top, to bottom lub wartość w deg) - nie wymagany!
  2. Pierwszy kolor -wymagany
  3. Drugi kolor - wymagany
4. Możemy podać więcej niż dwa kolory
5. Podanie wartości procentowej przy kolorze określi miejsce rozpoczęcia gradientu

Lektura dodatkowa: [gradient-kołowy](#)

Codepen: [gradient-liniowy](#)

# Background-image

1. Ustawiamy za pomocą własności background-image
2. Jako wartość podajemy funkcję url()
3. Funkcja url() przyjmuje wartości:
  1. Ścieżka do zdjęcia umieszczonego w folderze naszego projektu
  2. Ścieżka do zewnętrznego źródła
4. Należy ustawić własność background-color na wypadek gdyby zdjęcie się nie załadowało
5. **Wartości wspierające** background-image:
  1. Background-repeat
  2. Background-size
  3. Background-position

Codepen: [background-image](#)



# Background-repeat

1. Definiujemy za pomocą własności background-image
2. Pozwala kontrolować zachowanie się obrazka gdy jest mniejszy niż jego element
3. Przyjmuje wartości:
  1. Repeat
  2. Repeat-x
  3. Repeat-y
  4. No-repeat
  5. Inherit

Codepen: [background-repeat](#)

# Background-size

1. Definiujemy za pomocą własności background-size
2. Określa wielkość obrazka, który dodaliśmy
3. Przyjmuje wartości:
  1. Contain (skaluje obrazek do maksymalnych wymiarów, z zachowaniem ratio)
  2. Cover (skaluje element do maksymalnych rozmiarów i przycina go by zakrywał cały element)
  3. Wartości liczbowe (szerokość i wysokość)

Codepen: [background-size](#)

# Background-position

1. Definiujemy za pomocą własności background-position
2. Określa początkowe położenie obrazka
3. Przyjmuje wartości:
  1. Słowne (top, left, right, bottom) - które można ze sobą łączyć
  2. Wartości liczbowe (szerokość i wysokość)

Codepen: [background-position](#)

# Stylowanie elementów

List-style-type

# List-style-type

1. Definiuje jakiego rodzaju punktatorów będzie używać lista
2. Najpopularniejsze przyjmowane wartości:
  1. None
  2. Decimal (1. )
  3. Lower-Latin (a. )
  4. Circle
3. Lista wszystkich możliwych wartości dla **list-style-type**

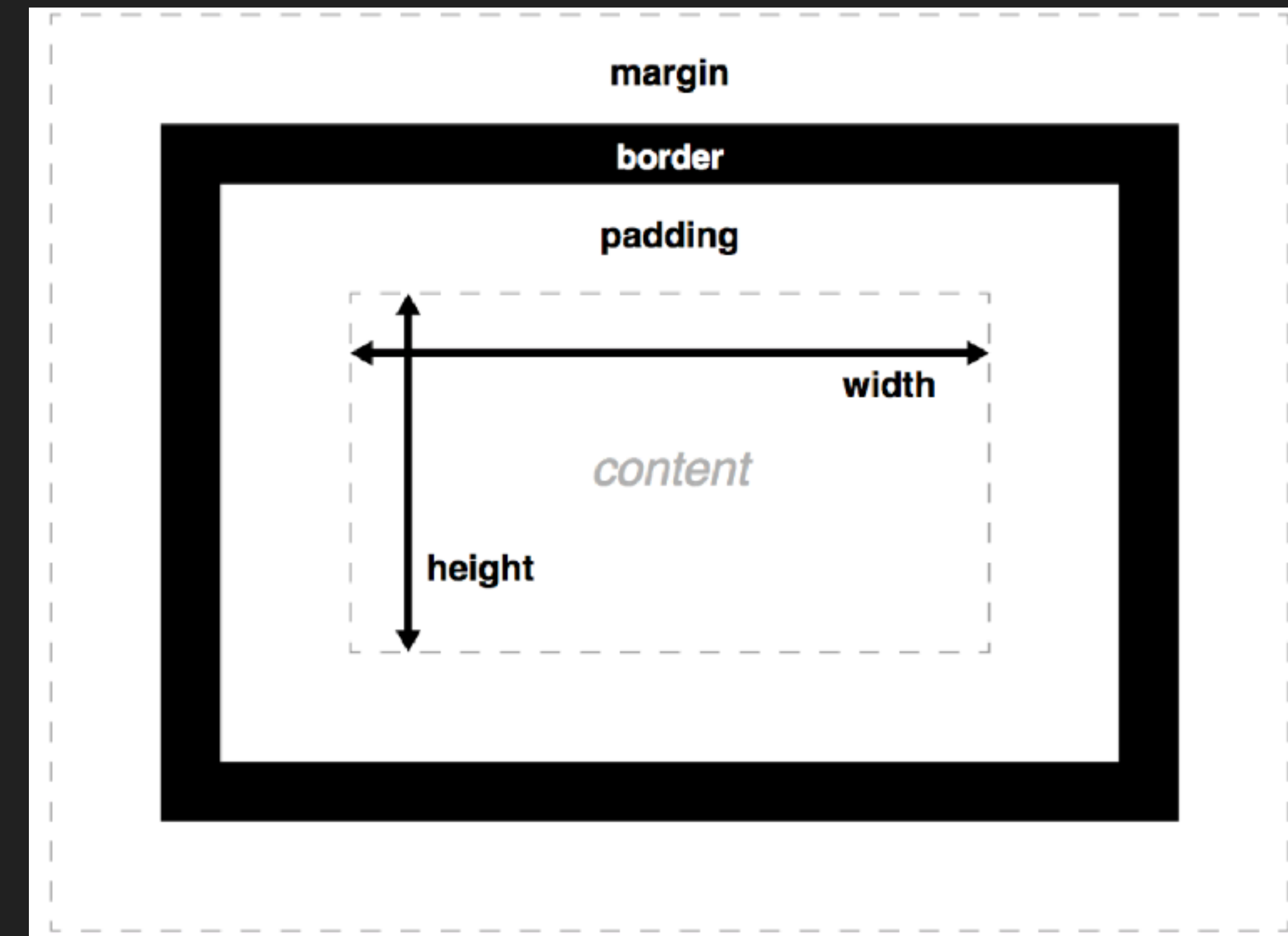
```
1 .linear-gradient-box {  
2   color: blue;  
3   list-style-type: none;  
4 }
```

# Box Model

Każdy element na stronie to pudełko

# Podstawowe wartości box-model

1. Każdy element na naszej stronie posiada pięć podstawowych wartości:
  1. Width
  2. Height
  3. Padding
  4. Border
  5. Margin



Codepen: [basic-box-model](#)

# Width | Height

1. Obie wartości określają rozmiary content-box, czyli przestrzeni, w której element przechowuje text oraz inne elementy (dzieci)
2. Obie własności przyjmują jednostki absolutne i relatywne
3. Ważny przypadek: height: 100% nie da nam wysokości całego ekranu!

Codepen: [basic-box-model](#)



# Padding

1. Jest wewnętrznym marginesem elementu
2. Znajduje się pomiędzy content-box'em a border
3. Każdej stronie content-box można nadać indywidualną wartość padding używając:
  1. Padding-left
  2. Padding-right
  3. Padding-top
  4. Padding-bottom

Codepen: [basic-box-model](#)

# Border

1. Wyznacza granicę elementu
2. Znajduje się pomiędzy padding a margin
3. Przyjmuje trzy wartości:
  1. Grubość
  2. Rodzaj border (solid, dashed, thick-double, ridge) - z solid jako wyjściowy
  3. Kolor
4. Podobnie jak w przypadku padding, każdej stronie można nadać indywidualne wartości:
  1. Border-left
  2. Border-right
  3. Border-top
  4. Border-bottom

Codepen: [basic-box-model](#)

# Margin

1. Wyznacza margines odległości pomiędzy elementem, a pozostałymi
2. Znajduje się za border
3. Jak w padding i border, każdej stronie można nadawać indywidualne wartości:
  1. Margin-left
  2. Margin-right
  3. Margin-top
  4. Margin-bottom

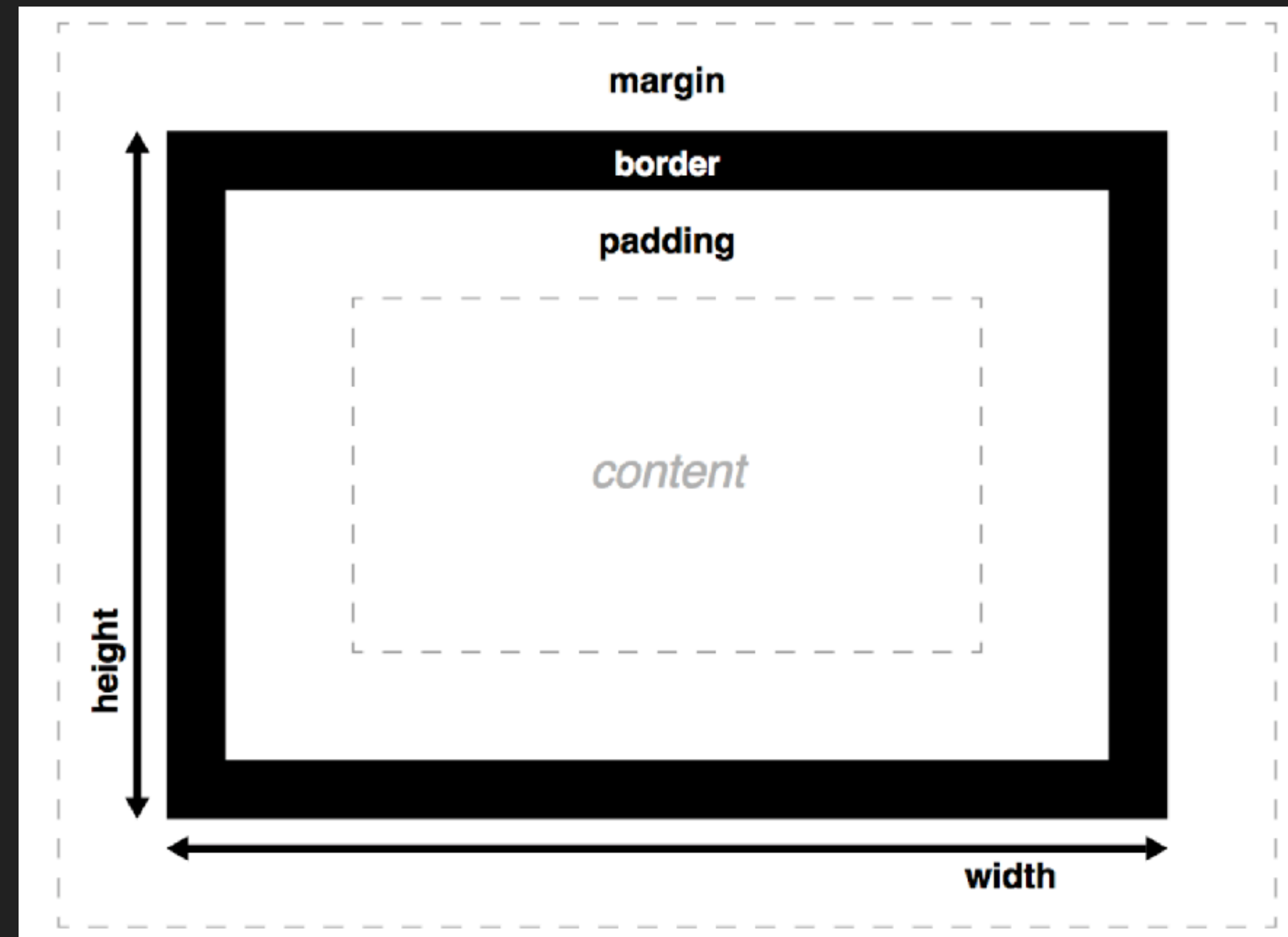
Codepen: [basic-box-model](#)

# Zachowanie Box-model

1. Background-color / Background-image rociągają się aż do border
2. Border ignoruje wartości zawarte w procentach
3. Końcowa wielkość pudełka to wysokość/szerokość + padding + border
4. Jeśli box jest większy od dostępnej szerokości ekranu, to pojawią nam się scroll-bar'y
5. Jeśli element w środku box'a będzie od niego większy, to ,wypłynie' on z niego
6. Obliczanie końcowej wielkości pudełka można kontrolować przez box-sizing

# Box-sizing

1. Nadawane przez własność box-sizing
2. Definiuje jak na końcu powinna zostać wyliczona wielkość pudełka
3. Możliwe wartości:
  1. **Content-box**: oblicza końcowy rozmiar jako:  $\text{width/height} + \text{padding} + \text{border}$
  2. **Border-box** - width/height to końcowe wymiary
4. W przypadku border-box, margin i padding będą zabierały miejsce wewnątrz elementu



Codepen: [box-sizing](#)

# Pozycjonowanie elementów

# Position

1. Nadajemy za pomocą własności position
2. Określa, w jaki sposób element jest położony na stronie
3. Przyjmuje cztery możliwe wartości:
  1. Static - wartość bazowa
  2. Relative
  3. Absolute
  4. Sticky
4. Do określenia położenia elementów wykorzystuje się własności:
  1. Top
  2. Bottom
  3. Left
  4. Right

# Relative

1. Element zajmuje miejsce wśród innych elementów
2. Można kontrolować jego położenie przy pomocy wartości top, bottom, left, right
3. Jest niezbędny do oczekiwanego działania wartości absolute - o tym w kolejnym slajdzie

Codepen: [position-relative](#)



# Fixed

1. Element NIE zajmuje miejsce wśród innych elementów
2. Można kontrolować jego położenie przy pomocy wartości top, bottom, left, right
3. Początkowa pozycja elementu ustalana jest w stosunku do początkowego bloku strony
4. Element z wartością sticky będzie zawsze widoczny na ekranie, niezależnie od przewijania strony (scroll'owania)

Codepen: [position-fixed](#)

# Sticky

1. Element zajmuje miejsce wśród innych elementów
2. Można kontrolować jego położenie przy pomocy wartości top, bottom, left, right
3. Wyjaśnienie zachowania elementu w przykładzie

Codepen: [position-sticky](#)

# Słowa końcowe

Budując powyższą prezentację posiłkowałem się materiałami ze strony z dokumentacją dla developerów - [developer.mozilla.org](https://developer.mozilla.org)

**Odradzam** korzystania ze strony - [w3schools.com](https://w3schools.com)

Wszystkie przykłady CodePen znajdziecie [tutaj](#)



Dziękuję za udział!