Wprowadzenie do Aplikacji Internetowych (Webowych)

wykład 3 – CSS cd + RWD

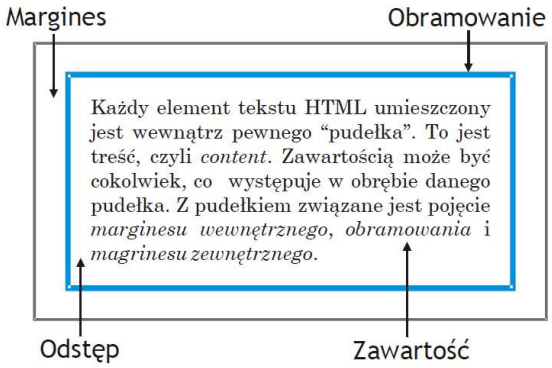
dr inż. Grzegorz Rogus

CSS Kaskadowe Arkusze Styli

1

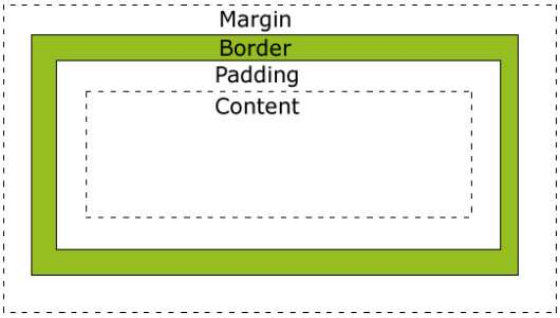
BOX MODEL

Koncepcja modelu pojemnika zakłada, że każdy element dokumentu HTML może być traktowany jako *prostokątny obszar*, którego zawartość otoczona jest *marginesem wewnętrznym* (*odstępem*), *obramowaniem* i *marginesem zewnętrznym*.



2

BOX MODEL – całkowita szerokość



http://www.w3schools.com/css/css_boxmodel.asp

Oznacza to, że rzeczywista wielkość danego obiektu jest sumą wielkości zawartości, paddingu, ramki oraz marginesu.

Szerokość = szerokość elementu + lewy odstęp + prawy odstęp + lewa ramka + prawa ramka + lewy margines + prawy margines

3

BOX MODEL
Przykład wykorzystania modelu pojemnika

```
h1
{
  background-color: skyblue;
}

...

<h1>Model pojemnika &mdash; box model</h1>
```

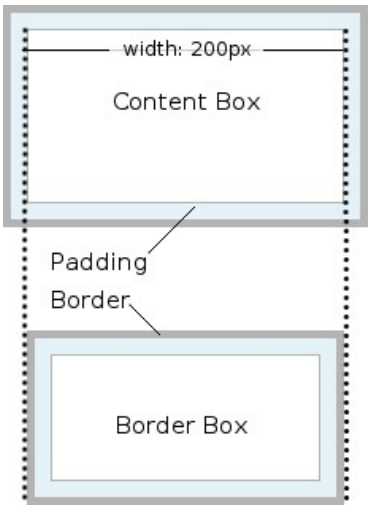
Model pojemnika — box model

```
h1
{
  background-color: skyblue;

  padding-left: 20px;
  padding-top: 15px;
  padding-bottom: 15px;
  padding-right: 20px;
}

...

<h1>Model pojemnika &mdash; box model</h1>
```

Model pojemnika — box model

4

box-sizing – inny model szerokości



Czym jest podana szerokość: ????

Całkowita (pudełka) = 234px;
200+2x5+2x10+2x2

```
p {
   margin: 10px;
   padding: 5px;
   width: 200px;
   border-width: 2px;
}
```

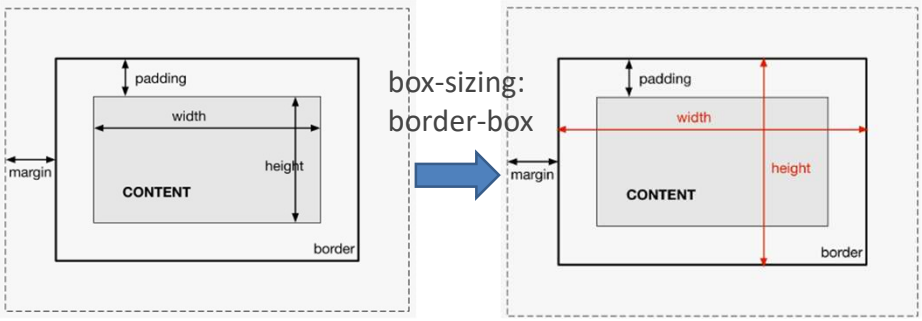
CSS3 wprowadza box-sizing

- content-box (szerokość = zawartość)
- border-box (szerokość = zawartość + obramowanie + odstęp)

Całkowita (pudełka) = 220px;
200+2x10

5

box-sizing – inny model szerokości cd.



width = right border + right padding + content width + left padding + left border

height = top border + top padding + content height + bottom padding + bottom border

width = ~~right border~~ + ~~right padding~~ + content width + ~~left padding~~ + ~~left border~~

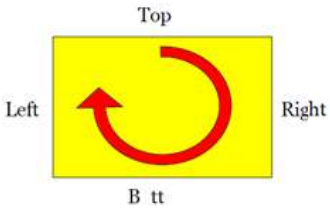
height = ~~top border~~ + ~~top padding~~ + content height + ~~bottom padding~~ + ~~bottom border~~

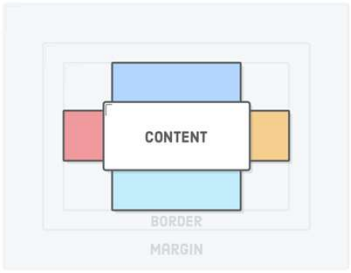
6

ZASADA WSKAZÓWEK ZEGARA

Jak zapamiętać przypisanie wartości do określonych boków pojemnika?

- ***Reguła stopera*** lub ***zegara wskazującego 12-stą***,
- ***Reguła trouble*** – TopRightBottomLeft.



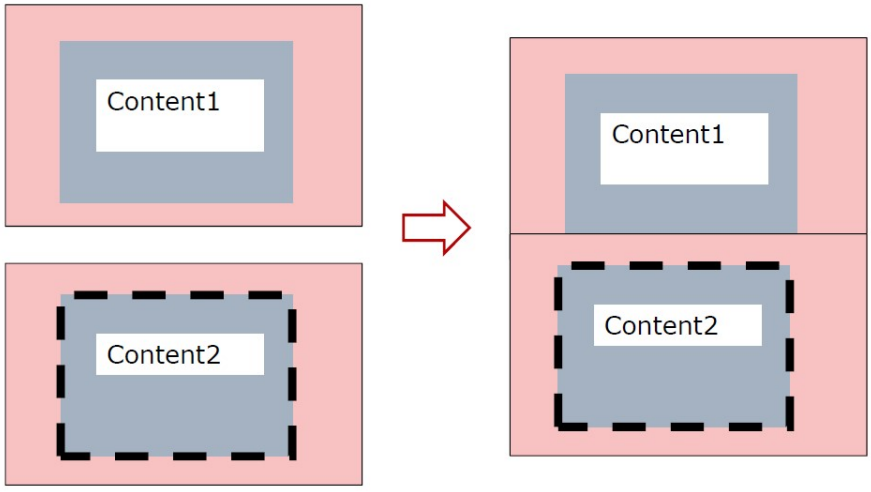


PADDING : TOP RIGHT BOTTOM LEFT

```
border-width: 6px;
border-width: 0px 3px 2px 10px;
border-width: 3px 0px;
```

7

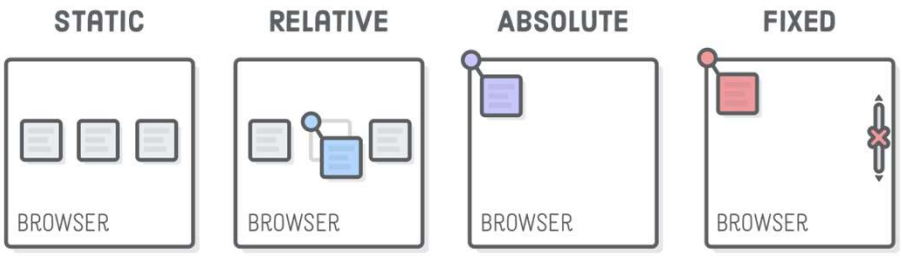
Zapadanie marginesów



- Jeżeli dwa, lub więcej, pojemniki *sąsiadują obok siebie pionowo* to *niezależnie* od ustalonych dla nich marginesów pionowych, odstęp pomiędzy nimi będzie równy *większej z ustalonych wartości.*

8

Pozycjonowanie elementów na stronie



9

Pozycjonowanie elementów na stronie

Pozycjonowanie służy w CSS do ustalania, względem czego układają się elementy. Występują przeważnie z właściwościami *top*, *bottom*, *left* i *right*.

Istnieją 4 rodzaje pozycjonowania:

- **position: static** – elementy układają się w kolejności ich umieszczania, nie nakładają się na siebie;
- **position: fixed** - elementy układają się względem okna przeglądarki, np.:

```
div { position:fixed; top:20px; right:30px; }
```

ustawi element 20px od góry i 30px od prawej strony okna.

10

Pozycjonowanie elementów na stronie

- **position: relative** – element układa się względem elementu poprzedniego, np.:

  ```
  div { position:relative; left:30px; }
  ```

  ustawi element w odległości 30px od elementu poprzedniego.
- **position: absolute** – element układa się względem elementu, w którym jest zamknięty i który nie jest opisany jako position:static.

  ```
  div { position:absolute; left:30px; }
  ```

  ustawi element w odległości 30px od krawędzi elementu nadrzędnego.

11

POSITION I PRZEPŁYW DOKUMENTU

Obiekty **static** i **relative** uczestniczą w tworzeniu przepływu dokumentu.

Obiekty **fixed** i **absolute** są wyrwane z przepływu dokumentu i przez to nie wpływają na wysokość elementów, w których się zawierają.

Aby to obejść bardzo często korzysta się z par obiektów o pozycjonowaniu relative i absolute.

Obiekty „relative" tworzą kontenery dla obiektów „absolute", dzięki czemu można precyzyjnie rozmieścić elementy na stronie, a jednocześnie zachować prawidłowy przepływ.

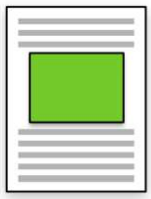
12

PRZEPŁYW DOKUMENTU

Elementy HTML w dokumencie układają się na kilka sposobów:

Normalny przepływ treści (ang. normal flow)

Elementy układają się kolejno jeden pod drugim. Obecność elementu w dokumencie odsuwa inne elementy, tak że żadne na siebie się nie nakładają.



To jest domyślne zachowanie w CSS i określane jest pozycją statyczną:

position: static;

13

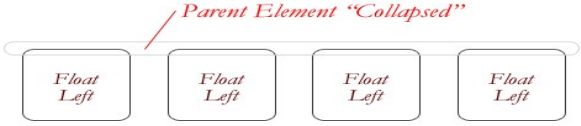
PRZEPŁYW DOKUMENTU

Wyjęcie z przepływu treści

Obiekt przestaje istnieć w swoim oryginalnym miejscu w dokumencie i pozostałe elementy są rozstawiane, jakby tego obiektu nie było.

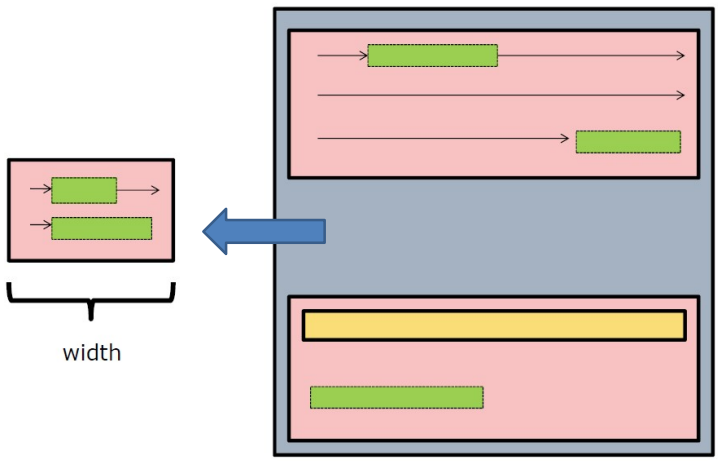
Obiekty wyjęte z biegu dokumentu najczęściej umieszczane są w innym miejscu strony przez **pozycjonowanie absolutne** lub **float**.

Jeżeli wszystkie dzieci danego elementu są wyjęte z biegu dokumentu, to element będzie miał zerową wysokość, ponieważ nie będzie rezerwował miejsca na żaden element w nim.



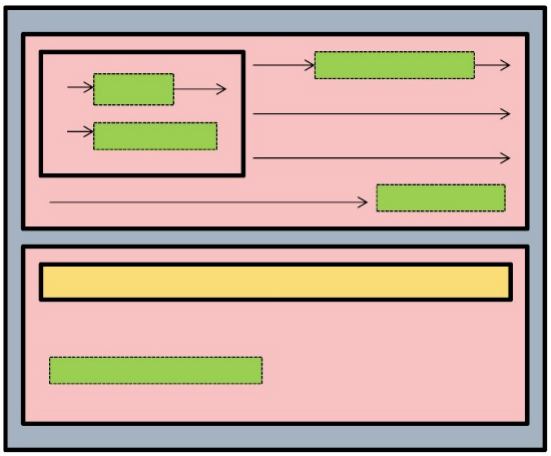
14

Floating – wyjecie z normalnego przepływu



15

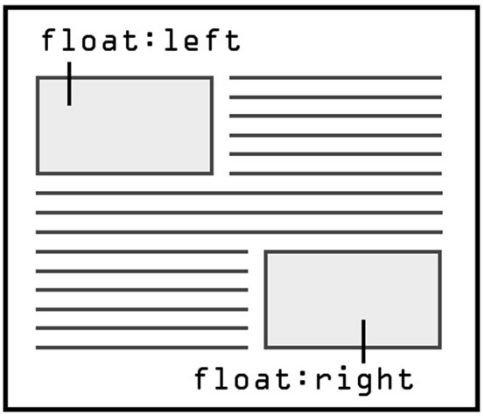
Przepływ dokumentu



16

FLOAT

- **left** - dla umieszczenia elementu z lewej strony,
- **right** - dla umieszczenia elementu z prawej strony,
- **none** - oznacza, że element nie będzie pływający. To jest domyślna wartość dla wszystkich elementów i zazwyczaj nie trzeba jej nadawać.

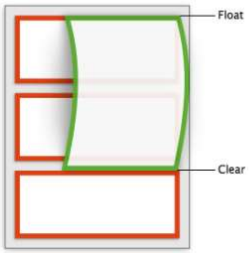


17

ZAPOBIEGANIE OPŁYWANIU (CLEAR)

Przerywanie przez clear

Ponieważ obiekty z float są wyjęte z normalnego biegu dokumentu, mogą rozciągać się nad kilkoma innymi — np. ilustracja może rozciągać się z boku kilku akapitów.



Aby uniknąć tego efektu należy jakiemuś elementowi za elementem z float nadać właściwość clear — powoduje ona ponowne „złączenie" float z biegiem dokumentu.

18

Overflow

Za pomocą właściwości overflow możemy ustalić jak ma zachować się element HTML w momencie gdy jego zawartość nie będzie mieściła się w jego rozmiarach.

- visible - domyślne
- hidden - zakrywanie nie mieszczącej się zawartości w zadeklarowanych ramach
- scroll - przewijanie
- auto - w razie konieczności pojawią się przewijanie

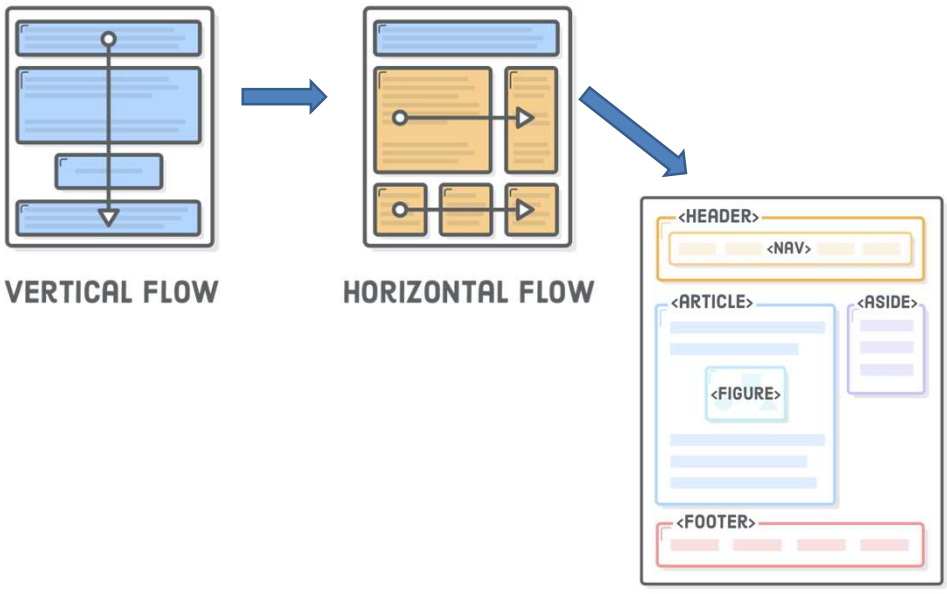
Nadwiar tekstu w tym bloku zostanie przycięty.

Cały tekst jest widoczny, bez przycinania i konieczności pojawienia się przewijania.

Pełny tekst będzie można

19

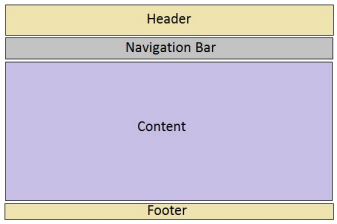
Tworzenie Layout (Układ strony)



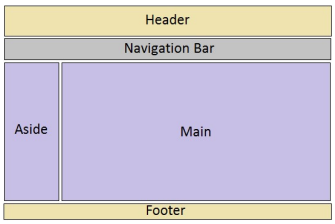
20

Najpopularniejsze szablony

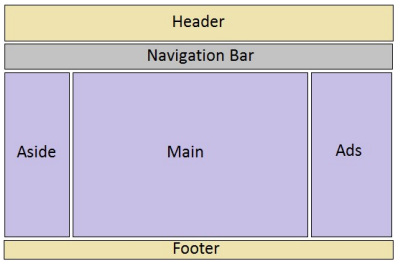
- One-column layout



- Two-column layout



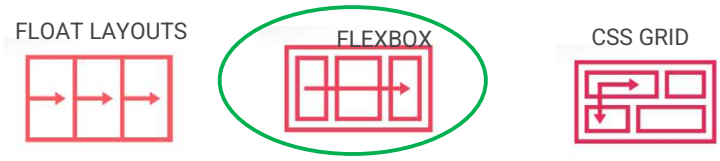
- Three-column layout

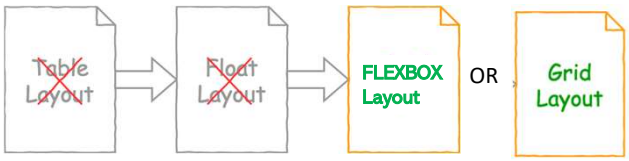


21

Tworzenie layout

- Floats
- Inline-block
- ~~display: table~~
- Absolute & Relative positioning





22

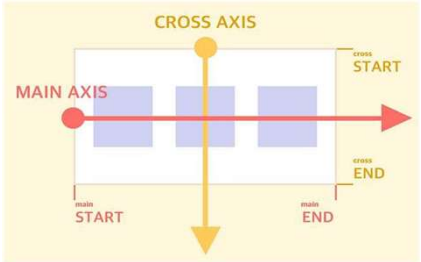
Flexbox - zastosowanie

23

FLEXBOX – najważniejsze fakty

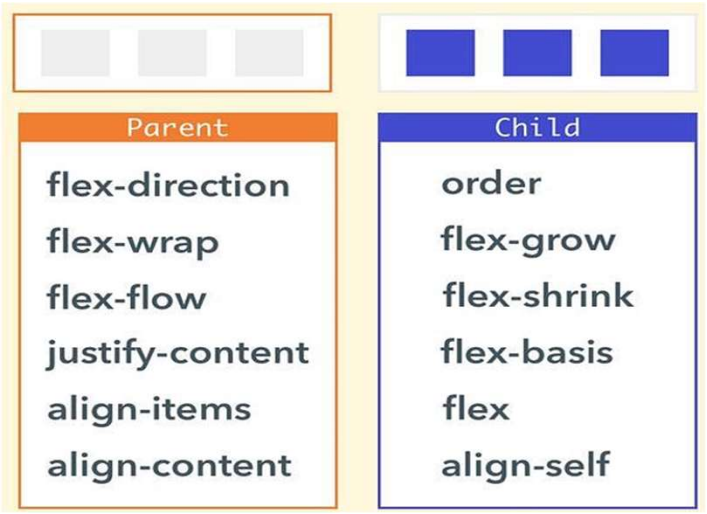
id=flex-container

Umieść w kontenerze: **`display: flex;`**



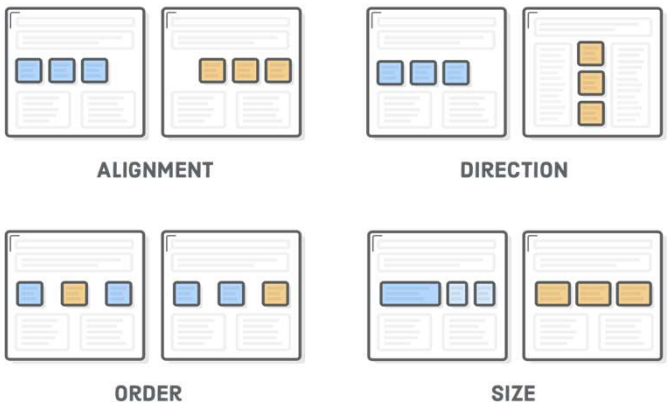
24

Flexbox



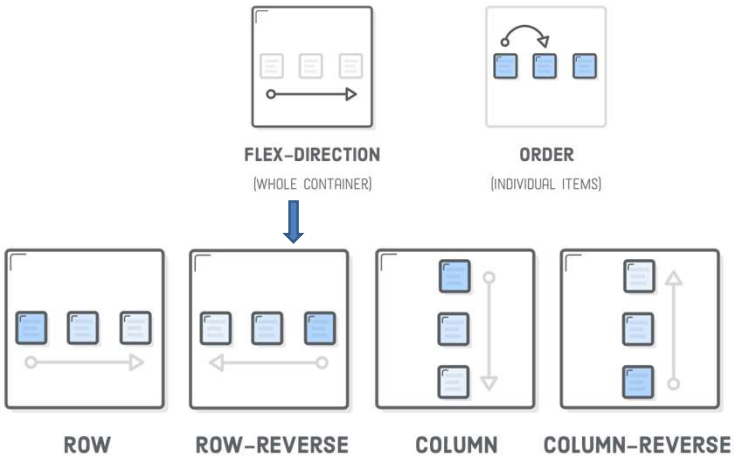
25

FLEXBOX – co kontrolujemy?



26

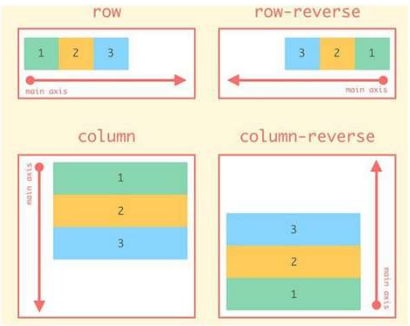
flex container order



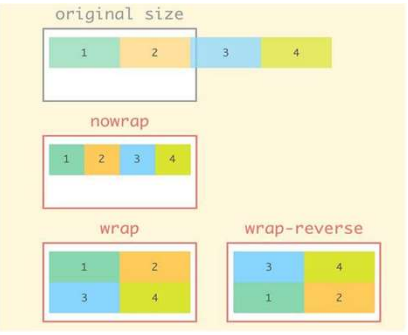
27

FLEXBOX – właściwości

flex-direction



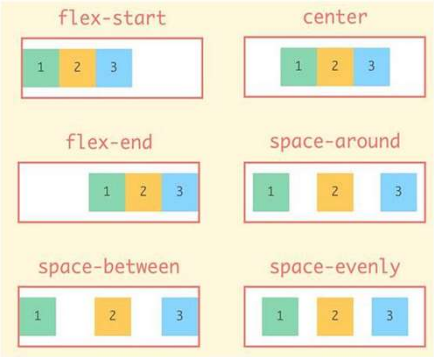
flex-wrap

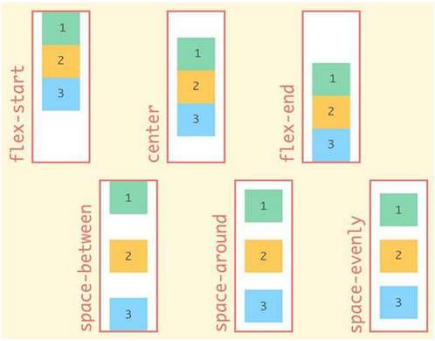


28

FLEXBOX – właściwości

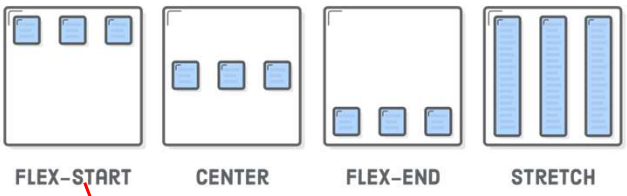
justify-content



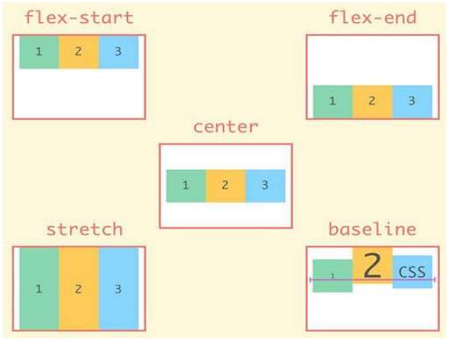


29

aligning (vertical) a flex item



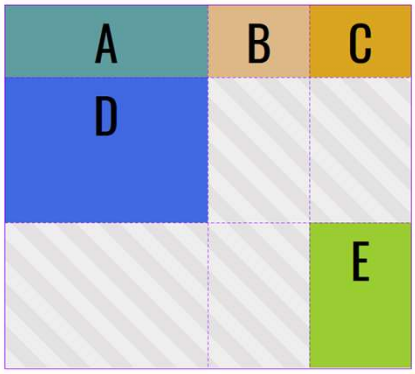

```
#flex-container {
display: flex;
align-items: flex-start;
}
```



30

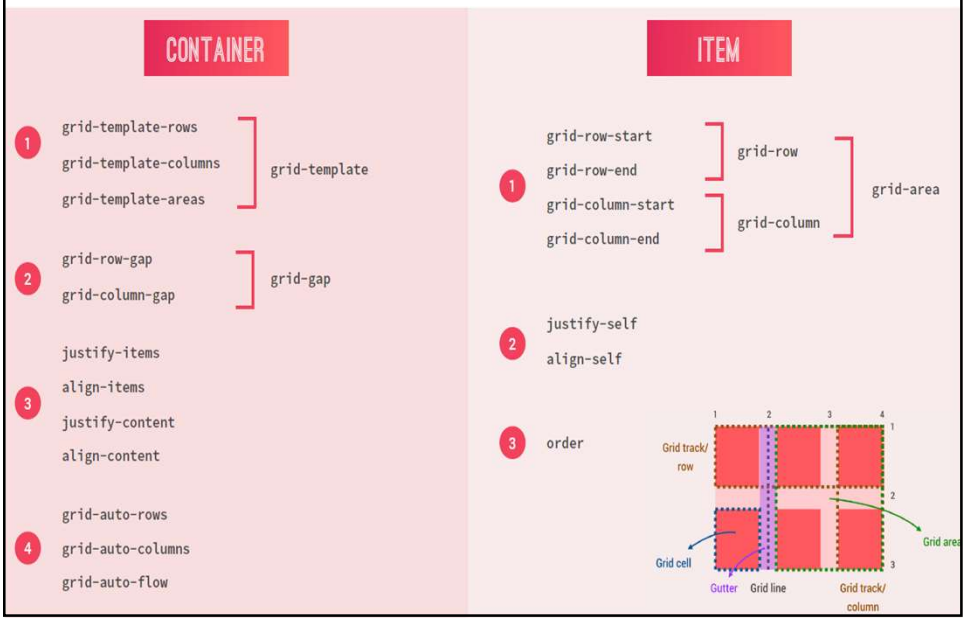
O co chodzi z CSS Grid

Idea, która stoi za CSS Grid, to oprzeć layout o strukturę siatki. Siatka składa się z kolumn i wierszy (jest dwuwymiarowa). Najmniejszą funkcjonalną częścią siatki jest komórka.



31

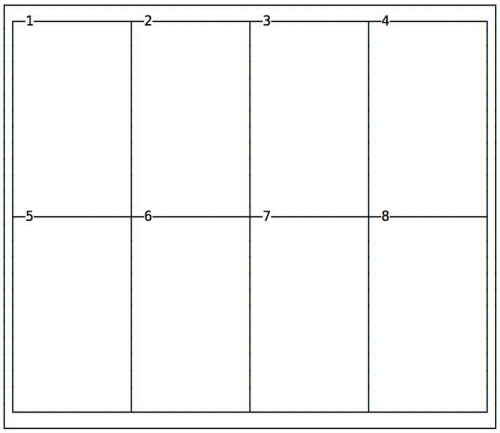
Grid – koncepcja i właściwości



32

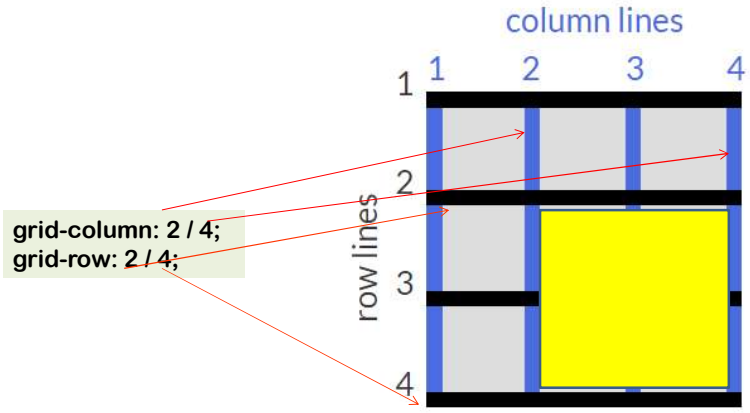
CSS grid

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 300px 300px;
}
```



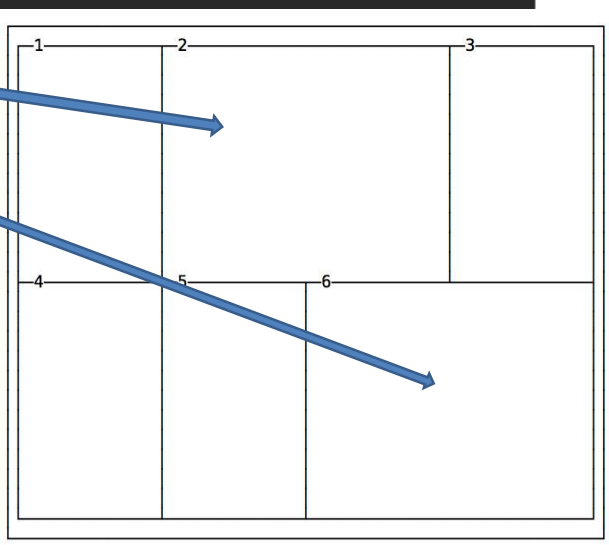
33

Łączenie komórek



34

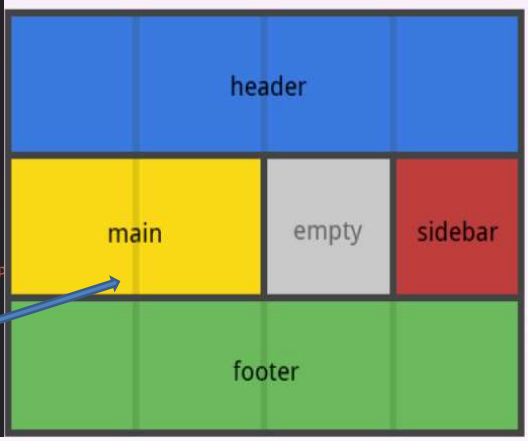
```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 300px 300px;
}
.item1 {
  grid-column-start: 2;
  grid-column-end: 4;
}
.item6 {
  grid-column-start: 3;
  grid-column-end: 5;
}
```



35

GRID – budowa układu strony

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.container {
  grid-template-columns: 50px 50px 50px 50p
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



36

CSS3 – co nowego

- **CSS3 Borders - Cienie i Zaokrąglenia**
- **CSS3 Text Effects**
- **CSS3 Backgrounds**
- **CSS3 Fonts**
- **CSS3 2D Transforms**
- **CSS3 3D Transforms**
- **CSS3 Transitions**
- **CSS3 Animations**
- **CSS3 Układ elastyczny**
- **CSS3 Zaawansowane filtry**

37

CSS3 Transformacje 2D i 3D

Transformacje 2D

translate()
rotate()
scale()
skew()
matrix()

38

Multiple Transforms dla jednego elementu

```
<style type="text/css">

  #submenu {
    background-color: #eee;
    -webkit-transition: all 1s ease-in-out;
    -moz-transition: all 1s ease-in-out;
    -o-transition: all 1s ease-in-out;
    -ms-transition: all 1s ease-in-out;
    transition: all 1s ease-in-out;
  }
  #submenu:hover {
    background-color: #fc3;
    -webkit-transform: rotate(360deg) scale(2);
    -moz-transform: rotate(360deg) scale(2);
    -o-transform: rotate(360deg) scale(2);
    -ms-transform: rotate(360deg) scale(2);
    transform: rotate(360deg) scale(2);
  }

</style>
```

39

CSS3 Transitions

Transitions (Przemiana)

Definiuje przemianę (przejście) obiektu

Użycie: **transition: własność czas-trwania rozkład-w-czasie opóźnienie**.

Algorytm użycia transition w CSS:

1. Zdefiniować dla obiektu styl zwykły
2. Zdeklarować końcowy stan dla obiektu np., dla hover
3. Do stanu zwykłego dodać funkcję transition przy użyciu własnosci : transition-property, transition-duration,transition-timing-function, and transition-delay.

lista własności jakie mogą by mogą być przemieniane

■ background-color and background-position ■ border-color, border-spacing, and border-width ■ bottom, top, left, and right ■ clip ■ color ■ crop ■ font-size and font-weight ■ height and width ■ letter-spacing ■ line-height ■ margin ■ max-height, max-width, min-height, and min-width ■ opacity ■ outline-color, outline-offset, and outline-width ■ padding ■ text-indent ■ text-shadow ■ vertical-align ■ visibility ■ word-spacing ■ z-index

Dla leniwych : http://www.css3-generator.de/transform.html

40

Responsive
Web
Design

41

Składowe nowoczesnej aplikacji webowej

czyli jak pisać dobry kod HTML I CSS ... i budować dobre strony internetowe

PROJEKTOWANIE RESPONSYWNE

- elastyczny układ (layouts)
- media queries
- responsywne zdjęcia
- właściwe jednostki
- desktop-first a mobile-first

JAKOŚĆ KODU

Kod skalowany i łatwy do zarządzania

- czystość formy
- łatwy do zrozumienia
- łatwość rozbudowy
- wielokrotnego użycia
- organizacja struktury plików
- nazewnictwo klas i struktur
- struktura HTML

WYDAJNOŚĆ

- mniej zapytań HTTP
- mniej kodu
- kompresja kodu
- użycie preprocessora CSS
- mniej obrazów
- kompresja obrazów

42

Problem - wygląd strony na różnych nośnikach

43

Rozwiązania

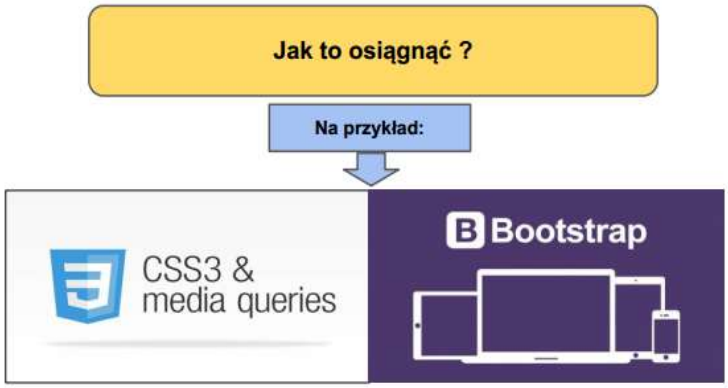
- Adaptive Design – serwer zwraca spreparowaną stronę na podstawie informacji o urządzeniu klienckim. Jeden adres URL.

- Separate Mobile Site (.m)- oddzielny adres URL dla stron mobilnych
- Responsive Web Design (RWD)

44

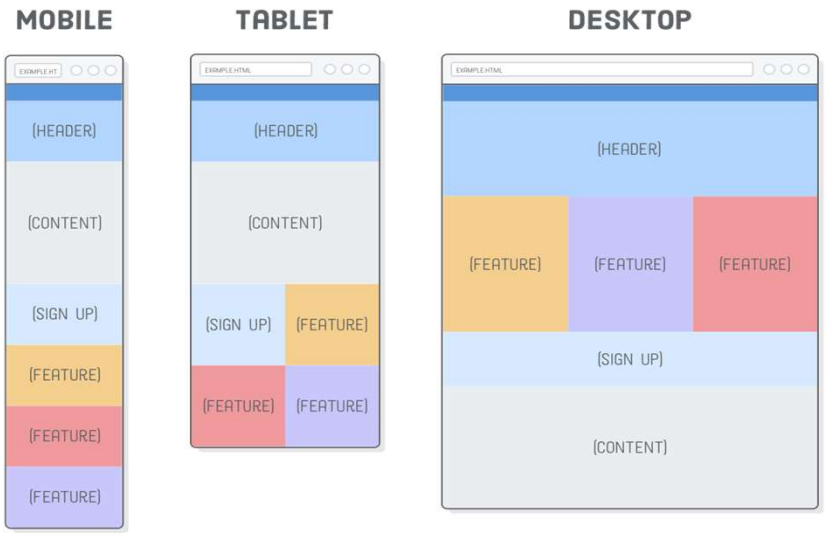
Nowoczesne aplikacje internetowe - FrontEnd

RWD Responsive Web Design ***[Reaktywne Projektowanie Stron]*** **- jest to nowoczesne podejście do projektowania stron internetowych, w którym programista zapewnia, że niezależnie od tego na jakie urządzenie strona zostanie podana wyświetli się ona prawidłowo. W tym kontekście programista zapewnia dostosowanie się wyglądu strony, rozmiaru oraz układu strony do rozmiaru okna przeglądarki w którym będzie wyświetlana.**



45

Idea RWD



46

Responsive Design

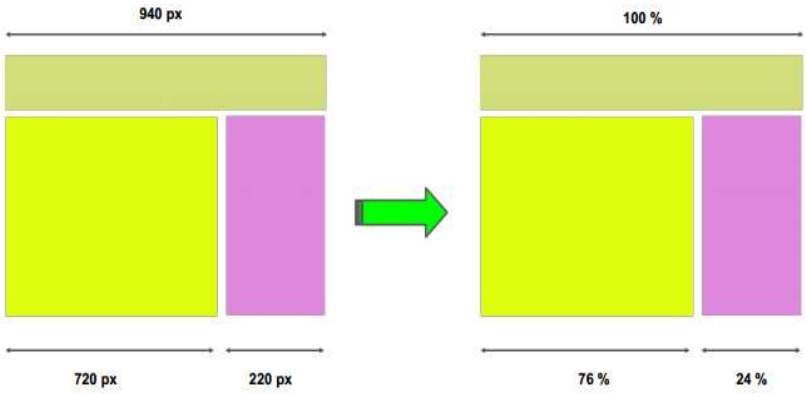
- Definiowany przez 3 najważniejsze elementy
  - Flexible grid-based layout – układ siatkowy
  - style warunkowe oparte o tzw. Media queries (CSS3)
  - Zdjęcia o elastycznym rozmiarze (Images resize)

- www.alistapart.com/articles/responsive-web-design/

47

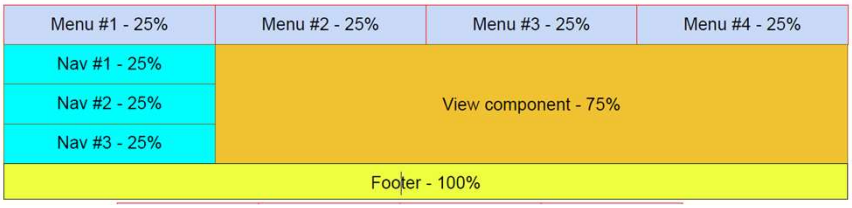
1. Flexible grid-based layout

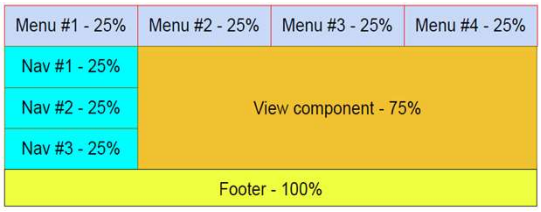
Chcąc zaprojektować "elastyczną" stronę dopasowującą się do całego ekranu powinniśmy zrezygnować z jednostek bezwzględnych wyrażonych np. w *px* i zastąpić je jednostkami względnymi w procentach %.
Pozwoli to uzyskać dopasowany układ do różnych szerokości ekranów.
Atrybuty *max-width* powinny być używane często zwłaszcza w przypadku elementów, które mogą nam "rozpychać layout" tj. obrazów, osadzonych odtwarzaczy video etc.



48

Układ elastyczny - przykład



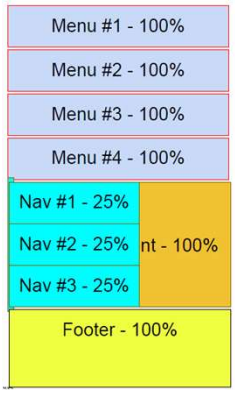


49

Ale czasami sam układ elestyczny nie wystarczy

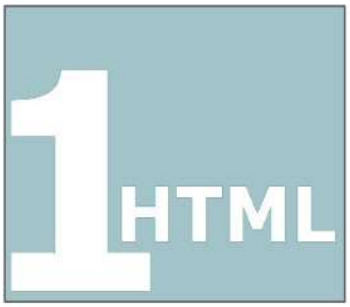
Reguły CSS:

```
@media only screen and (min-width:768px) {
/* layout do tabletów i desktopów */ }

@media only screen and (max-width: 767px)
{  /* smartfony */ }

@media only screen and (max-width:767px)
and (orientation: portrait) {
/* smartfony orientcja portrait */ }
```



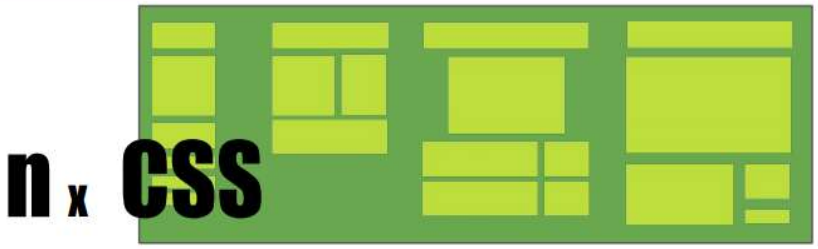
50

2. Style warunkowe



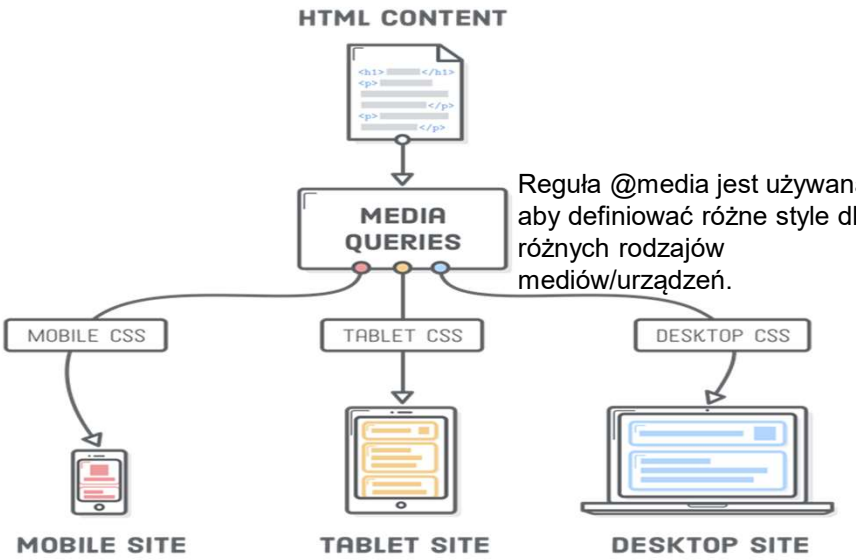
ZASADA:

W ogólności zasada jaka powinna przyświecać tworzeniu stron zgodnie z metodologią RWD jest utworzenie jednego pliku HTML, a dla formatowania jego wyglądu wiele "layoutów" w CSS które będą zawierać odpowiednie reguły wyświetlania strony w zależności od rozdzielczości ekranu.



51

RWD



Reguła @media jest używana, aby definiować różne style dla różnych rodzajów mediów/urządzeń.

52

Zapytania mediów - media queries

- Technika CSS wprowadzona w CSS3.
- Używa reguły @media żeby dołączyć jakiś blok właściwości CSS jeśli zdefiniowany warunek jest prawdziwy.

Przykłady:

- jeśli okno przeglądarki jest mniejsze niż 500 pikseli to kolor tła zmieni się na jasnoniebieski

```
@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
```

- kiedy ekran (okno przeglądarki) stanie się mniejsze niż 768 pikseli, każda kolumna będzie miała szerokość 100%:

```
@media only screen and (max-width: 768px) {
    [class*="col-"] { /* dla mobile phones: */
        width: 100%;
    }
}
```

53

Media Query – informacje

Zapytania mediów sprawdzają możliwości urządzenia, mogą być użyte do sprawdzenia wielu rzeczy, np.:

- szerokości i wysokości okna roboczego,
- szerokości i wysokości ekranu urządzenia
- orientacji (czy tablet/telefon jest poziomo czy pionowo?)
- rozdzielczości i wielu innych.

Zapytanie mediów standardowo składa się z:

- **typu** lub **grupy** medium,
- słowa kluczowego **and**
- **cechy** medium umieszczonej w nawiasie.

Operatory logiczne

- AND
- przecinek
- NOT

54

REGUŁY MEDIA QUERIES

@media warunek1, warunek2, (...), warunek-n {
/* kod css dla urządzeń spełniających podane warunki */
}
@media screen {
* { font-family: sans-serif }
}
@media screen and (min-width: 400px) and (max-width: 700px)
{ ... }
@media **not** screen and (color) {
/* kod css dla urządzeń nie posiadających kolorowego ekranu */
}

55

Inne przykłady

```
@media (pointer: coarse) and (min-width: 600px) {
...
}
@media (min-width: 900px), (orientation: landscape) {
...
}
@media not (orientation: landscape) {
...
}
```

56

Typy mediów

- all - dla wszystkich typów urzadzen
- print - uzywane dla drukarek (dla podglądu wydruku
- screen - dla ekranów komputera, tabletu, smartfonów itp.
- speech - dla czytników ekranu, które czytają strone
- aural - **nieaktualne**, dla syntezatorów mowy i dźwięku
- braille - **nieaktualne**, dla urządzeń przeznaczonych dla niewidomych
- embossed - **nieaktualne**, dla drukarek brailla
- handheld - **nieaktualne**, dla bezprzewodowych urządzeń ręcznych
- projection - **nieaktualne**, dla prezentacji projektorowych
- tty - **nieaktualne**, dla dalekopisów, terminali z ograniczonymi możliwościami wyświetlania
- tv - **nieaktualne**, dla telewizora

57

MEDIA QUERIES

Reguły media queries bazują na cechach nośnika, takich jak:

- Szerokość: width, min-width, max-width
- Wysokość: height, min-height, max-height
- Szerokość urządzenia: device-width, min-device-width, max-device-width
- Wysokość urządzenia: device-height, min-device-height, max-device-height
- Orientacja: orientation
- Proporcje obrazu: aspect-ratio, min-aspect-ratio, max-aspect-ratio
- Proporcje obrazu urządzenia: device-aspect-ratio, min-device-aspect-ratio, max-device-aspect-ratio
- Głębia koloru: color, min-color, max-color
- Paleta kolorów: color-index, min-color-index, max-color-index
- Urządzenia monochromatyczne: monochrome, min-monochrome, max-monochrome
- Rozdzielczość: resolution, min-resolution, max-resolution
- Technika wyświetlania: scan
- Siatka: grid

58

Sposoby implementacji media queries?

Trzy sposoby implementacji:

1. Użycie importu styli zewnętrznych w pliku CSS - @import.

```
@import url(style600.css) screen and (min-width: 600px);
```

2. użycie Media Queries bezpośrednio w arkuszu styli

```
@media screen and (min-width: 600px){ #section{ width: 550px; } }
```

3. Użycie query jako atrybutu w stylu zewnętrznym

```
<link rel="stylesheet" type="text/css" media="screen and
(max-device-width: 600px)" href="style600.css" />
```

59

REGUŁA MEDIA QUERIES

Meta-tag *viewport* zapewnia poprawne dostosowanie szerokości oraz brak powiększenia. Jeżeli zapomnisz go umieścić, strona będzie prezentować się źle.

```
<head>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
</head>
```

- width=device-width – ustawia szerokość strony adekwatną do szerokości wyświetlacza urządzenia
- initial-scale=1 – ustawia poziom powiększenia podczas pierwszego uruchomienia strony.

Wyłączenie zoomowania dla urządzeń mobilnych: zasada **mobile-first**

```
<meta name = 'viewport' content='width=device-width, initial-scale=1,
maximum-scale = 1'>
```

60

Desktop-First
vs
Mobile-First

61

Desktop-First

```
.example-section {
display: flex;
width: 60%;
}
.example-section__text {
font-size: 32px;
}
@media (max-width: 768px) {
.example-section {
flex-direction: column;
width: 80%;
```

62

Mobile-First

```
.example-section {
display: flex;
flex-direction: column;
width: 80%;
}
.example-section__text {
font-size: 24px;
}
@media (min-width: 768px) {
.example-section {
flex-direction: row;
```

63

Jak ustalić breakpointy?

- Podział popularnych urządzeń na kilka grup
- BP zgodne z 'wymaganiami' designu

64

Typowe ustawienia

Label	Layout Width
Smartphones	480px and below
Portrait Tables	480px to 768px
Landscape Tablets	768px to 940px
Default	940px and up
Large Screens	1210px and up

65

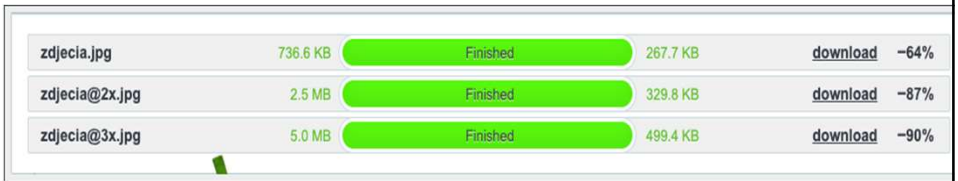
PUNKTY PRZEŁAMANIA (RWD BREAKPOINTS)

```
/*========== Mobile First Method ==========*/
/* Custom, iPhone Retina */
@media only screen and (min-width : 320px) { ... }

/* Extra Small Devices, Phones */
@media only screen and (min-width : 480px) { ... }

/* Small Devices, Tablets */
@media only screen and (min-width : 768px) { ... }

/* Medium Devices, Desktops */
@media only screen and (min-width : 992px) { ... }

/* Large Devices, Wide Screens */
@media only screen and (min-width : 1200px) { ... }
```

66

Optymalizacja zdjęć/grafik

- Kompresja plików

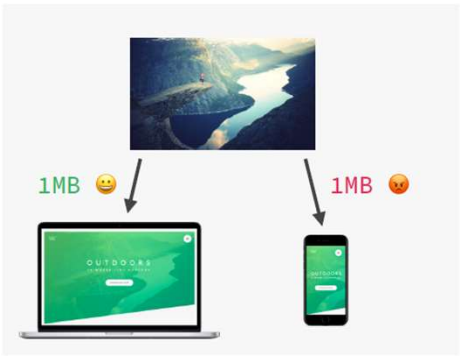


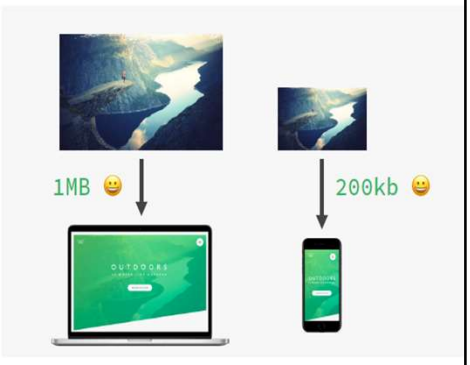
- Różne pliki dla różnych ekranów
- Skalowanie zdjęć i filmów video w celu dopasowana do rozmiaru obszaru na ekranie

```
img { width: 100%; height: auto; }
video { width: 100%; height: auto; }
```

67

Różne pliki dla różnych ekranów





https://responsivebreakpoints.com/

68

Zalecenia

- Używaj obrazów, które najlepiej pasują do cech wyświetlacza. Weź pod uwagę rozmiar ekranu, rozdzielczość urządzenia i układ strony.
- Gdy chcesz określić różne obrazy wyświetlane w zależności od cech urządzenia (tzn. dostosować grafikę), użyj elementu **picture**.
- Użyj atrybutu **srcset** i deskryptora x w elemencie **img**, by przy wyborze obrazu podpowiedzieć przeglądarce, której rozdzielczości najlepiej użyć.

69

Grafika w zależności od gęstości pikseli

```
<img
srcset="img/foto.jpg, // użyj jeśli 1px w CSS wyświetlany
przez 1 fizyczny piksel
       img/foto@2x.jpg 2x, // użyj jeśli 1px w CSS wyświetlany
przez 2 fizyczne piksele
       img/foto@3x.jpg 3x" // użyj jeśli 1px w CSS wyświetlany
przez 3 fizyczne piksele
src="img/foto@3x.jpg"
alt="Opis zdjecia"
>
```

70

Grafika w zależności od rozmiaru ekranu

```
<img
srcset="img/foto.jpg 480w,img/foto2.jpg 700w"
sizes="(max-width: 600px) 580px,
(max-width: 1200px) 48vh, 800px"
src="img/foto3.jpg"
alt="Opis zdjęcia"
>
```

sizes są podobne do zapytań o media, opisujące, ile miejsca zajmuje obraz w rzutni.

- jeśli rzutnia jest większa niż 1200px, obraz ma dokładnie 800px
- jeśli rzutnia ma wielkość między 1200px a 600px, obraz zajmuje 48% rzutni
- jeśli rzutnia jest mniejsza niż 640px, obraz zajmuje 580 px

srcset po prostu mówi przeglądarce, jakie obrazy mamy dostępne i jakie są ich rozmiary.

- img/foto.jpg ma szerokość 480 pikseli,
- img/foto2.jpg ma szerokość 700 pikseli,

71

Grafika w zależności od 'wizji artystycznej'

```
<picture>
<source media="(max-width: 799px)"
srcset="img/foto.jpg">
<source media="(min-width: 800px)"
srcset="img/foto2.jpg">
<img src="img/foto3.jpg" alt="opis zdjecia">
</picture>
```

72