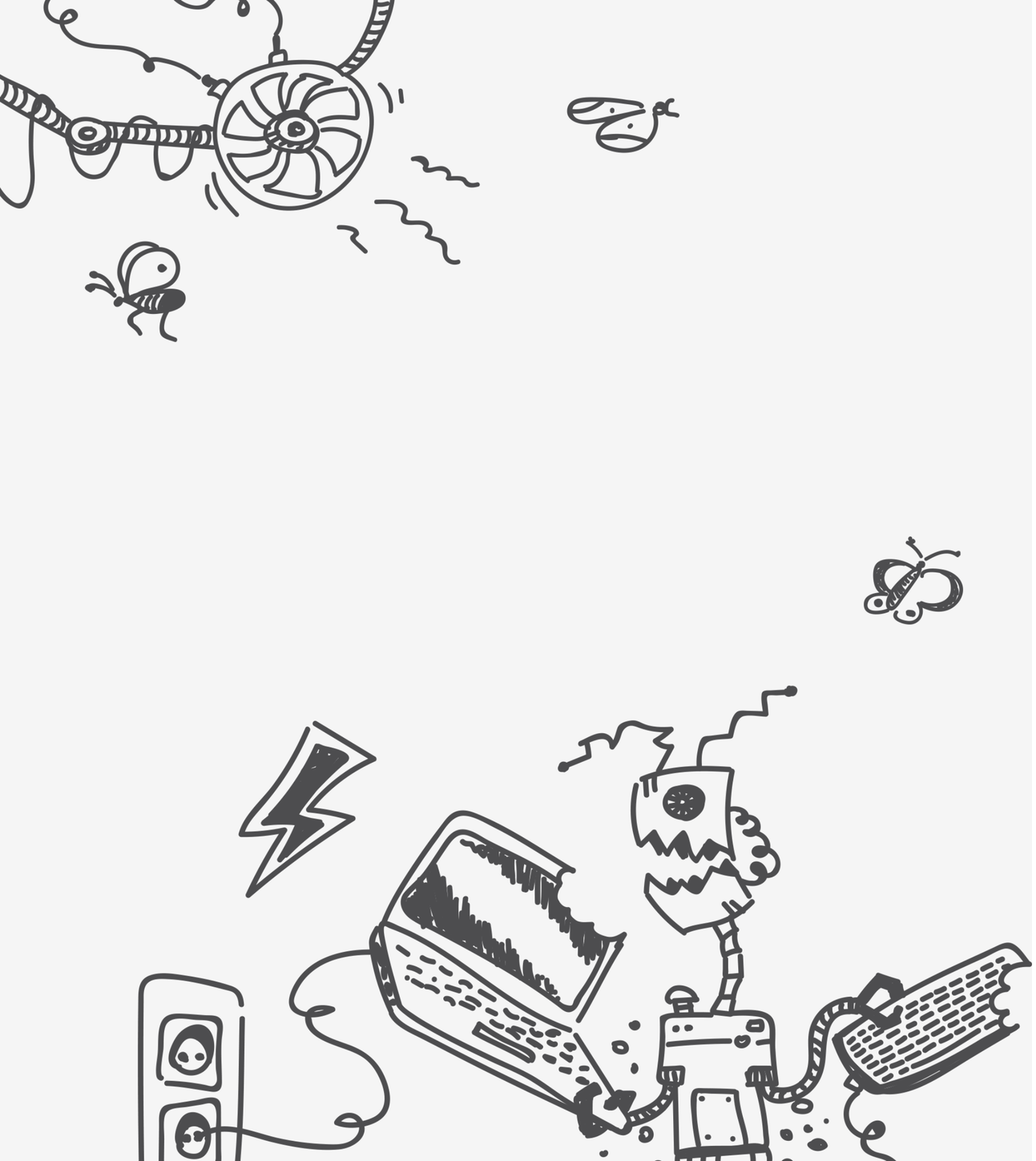


Media queries i viewport

Czym są Media queries?

Co to takiego?

Za pomocą reguły `@media` możemy decydować o tym, która część kodu CSS ma być interpretowana przez przeglądarkę na danym urządzeniu.

Dzięki temu możemy określić, jak treść – zależnie od urządzenia – będzie tam prezentowana.

W specyfikacji CSS2 przy użyciu reguły `@media` można było wybrać rodzaj medium:

- `screen`
- `print`

Specyfikacja CSS3 pozwoliła na sprecyzowanie kryteriów reguły `@media` przez zwiększenie zakresu rodzaju medium i dodanie właściwości takich jak szerokość.

Rodzaje mediów

Rodzaje media

- `all` – dla wszystkich urządzeń,
- `screen` – dla kolorowych ekranów komputerowych lub mobilnych,
- `tv` – dla telewizorów,
- `print` – dla wydruku,
- `projection` – dla prezentacji/projektorów,
- `speech` – dla syntezatorów mowy,
- `braille` – dla dotykowych urządzeń obsługujących system Braille'a,
- `embossed` – dla drukarek obsługujących system Braille'a,
- `handheld` – dla prostych urządzeń komórkowych z prostym wyświetlaczem,
- `tty` – dla mediów ze stałą liczba znaków w jednym wierszu.

Czym jest viewport?

Obszar widoczny

`Viewport` to widoczny obszar ekranu, na jakim użytkownik wyświetla stronę.

Na jego podstawie powinny być sprawdzane warunki szerokości i wysokości zadeklarowane w `media queries`.

Z reguły tylko część strony mieści się w `widocznym obszarze`, pozostałą można zobaczyć dzięki przewinięciu jej zawartości.

Co jeśli nie ustawimy viewportu?

Jeśli obszar widoczny nie jest skonfigurowany, urządzenia mobilne renderują stronę według rozdzielczości zastępczej, która z reguły waha się między `800px` a `1024px`.

W wyniku tego strona zostanie przeskalowana według tej rozdzielczości tak, aby zapełniła ekran urządzenia. Zmusza to użytkownika do powiększenia strony, aby z niej skorzystać.

Metatag viewport - ustawienie

Wielkość strony

Aby strona mogła być responsywna i jej wielkość odpowiednio skalowalna do obszaru widocznego urządzenia,należy zdefiniować `viewport` za pomocą `metatagu`.

Taki tag powinien znaleźć się w sekcji `head` każdej strony.

```
<meta name="viewport">
```

Składnia

```
<meta name="viewport" content=" ">
```

Składnia - przykład

```
<meta name="viewport" content="initial-scale=1">
```

Metatag viewport - ustawienie

Wielkość strony

Aby strona mogła być responsywna i jej wielkość odpowiednio skalowalna do obszaru widocznego urządzenia,należy zdefiniować `viewport` za pomocą `metatagu`.

Taki tag powinien znaleźć się w sekcji `head` każdej strony.

```
<meta name="viewport">
```

Składnia

```
<meta name="viewport" content=" ">
```

Tu wpisujemy wartości.

Składnia - przykład

```
<meta name="viewport" content="initial-scale=1">
```

Metatag viewport

Wartości

Metatag `viewport` może przyjąć następujące wartości, które wpisujemy w atrybucie `content`:

- `width i height` – pozwala zdefiniować szerokość i wysokość obszaru widocznego,
- `initial-scale` określa domyślne powiększenie (zoom) strony,
- `minimum-scale, maximum-scale` – określa wielkość minimalnej (0) i maksymalnej (10) wielkości strony

Przykład:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Metatag viewport

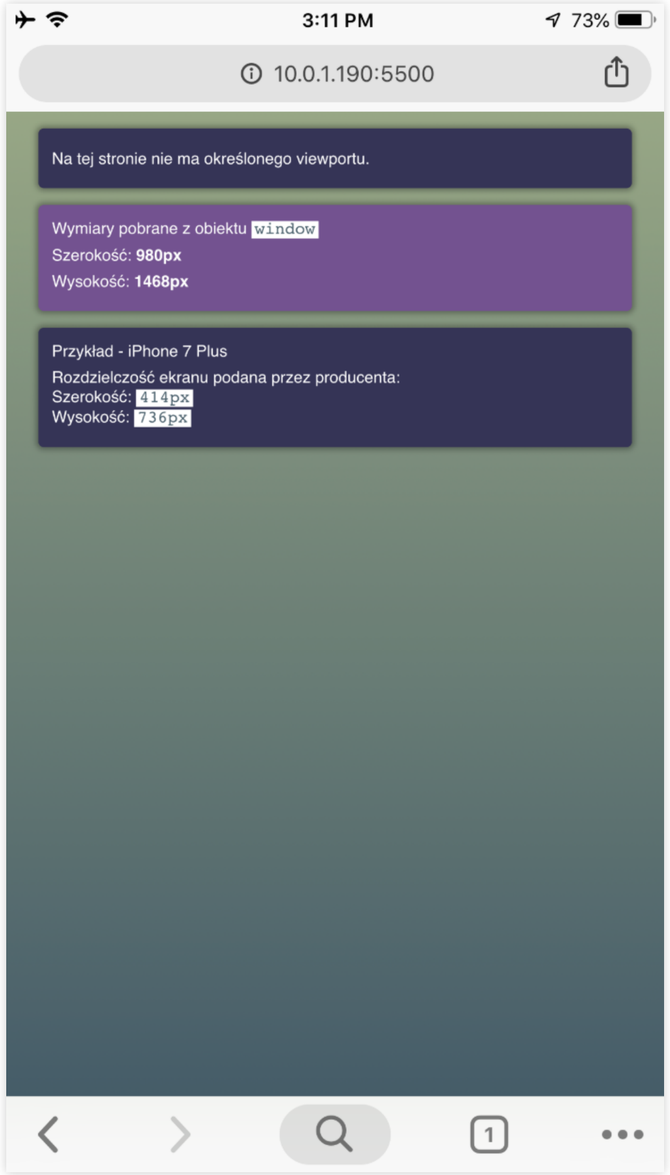
Brak metatagu viewport

Na stronie obok nie ma określonego viewportu. Strona otwarta była na urządzeniu iPhone 7 Plus o rozdzielczości ekranu (podanej przez producenta):

- Szerokość: `414px`
- Wysokość: `736px`

Wymiary urządzenia odczytane z obiektu `window` w JavaScript wskazuje na następujące wymiary:

- Szerokość: `980px`
- Wysokość: `1468px`



Metatag viewport

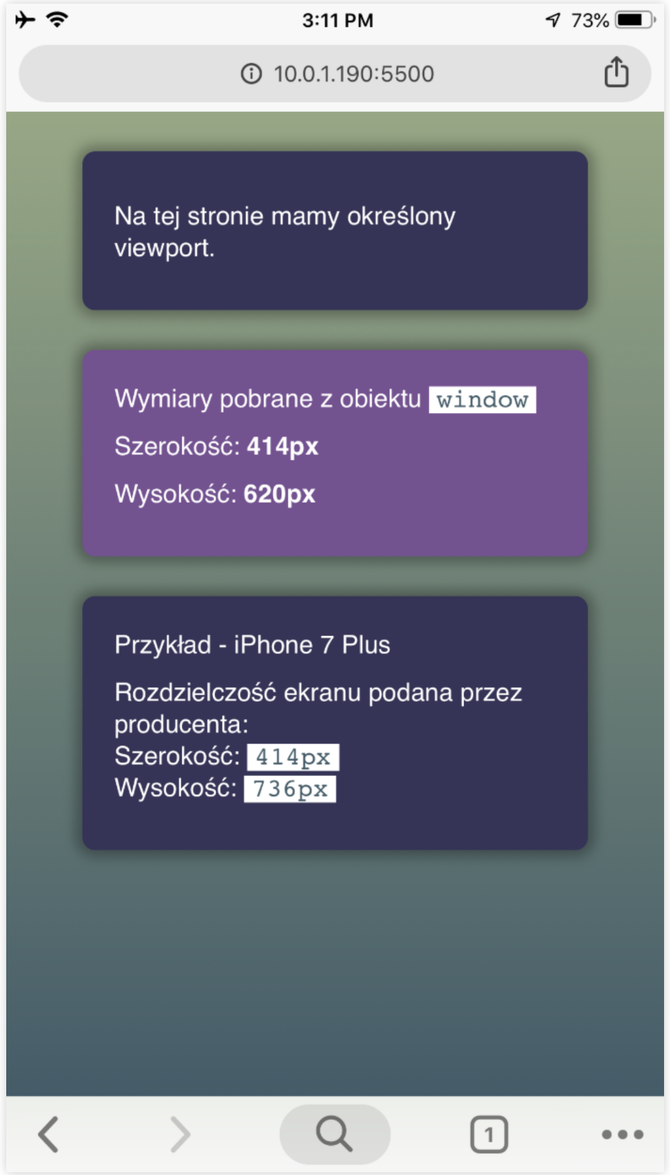
Efekt użycia metatagu viewport

Na stronie obok został określony metatag viewportu.

```
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
```

Wymiary urządzenia odczytane z obiektu `window` są już podobne do wymiarów podanych przez producenta (mamy mniejszą wartość wysokości ze względu na górną i dolną belkę którą zajmuje przeglądarka):

- Szerokość: `414px`
- Wysokość: `620px`



Budowanie media queries

Kilka słów o @media

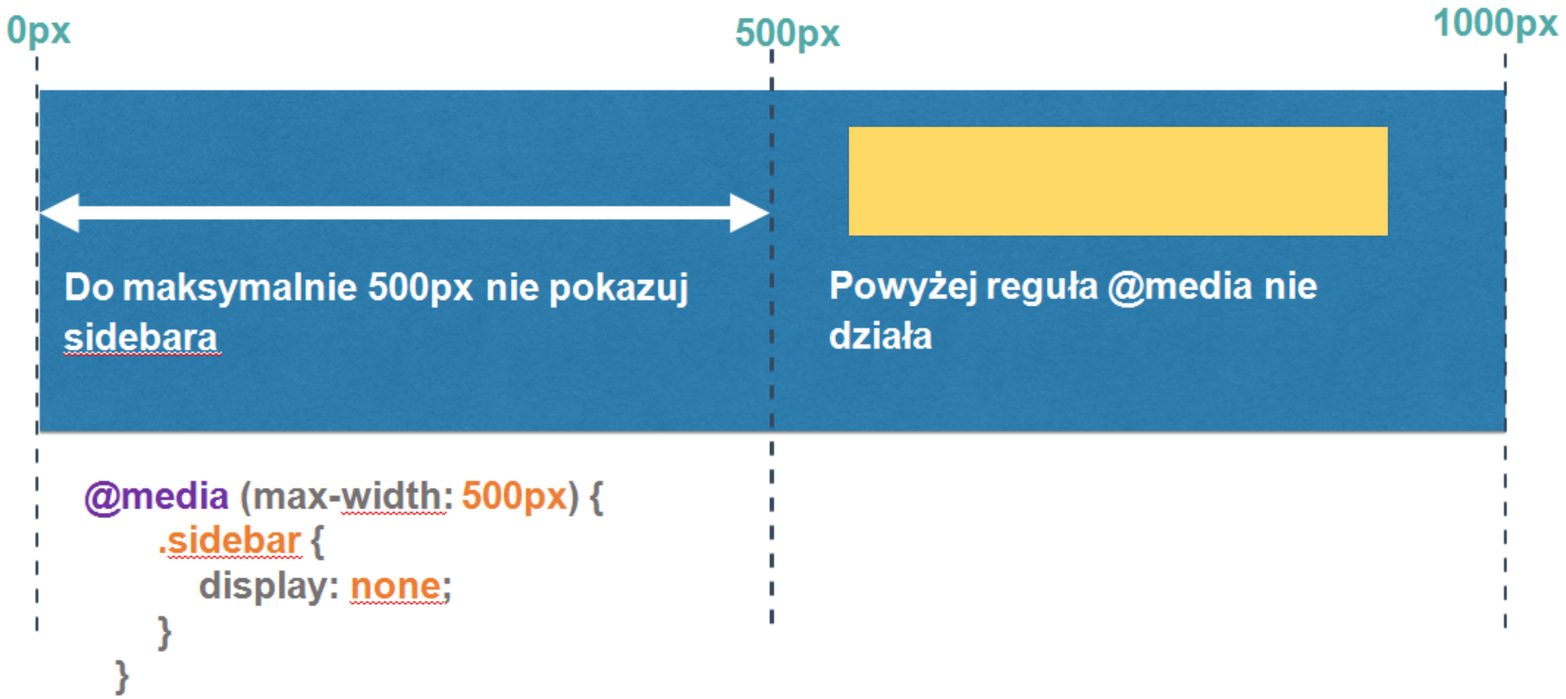
- Po wstawieniu odpowiedniego tagu `<meta>`, optymalną metodą jest wstawienie reguł `@media` w jednym pliku CSS.
- Efektywna reguła `@media` powinna składać się z określenia rodzaju medium i jego właściwości.
- Najbardziej przydatnymi właściwościami reguł `@media` są szerokość, orientacja i gęstość pikseli.

Przykład

```
@media (max-width: 600px) {
  .sidebar {
    display: none;
  }
}
```

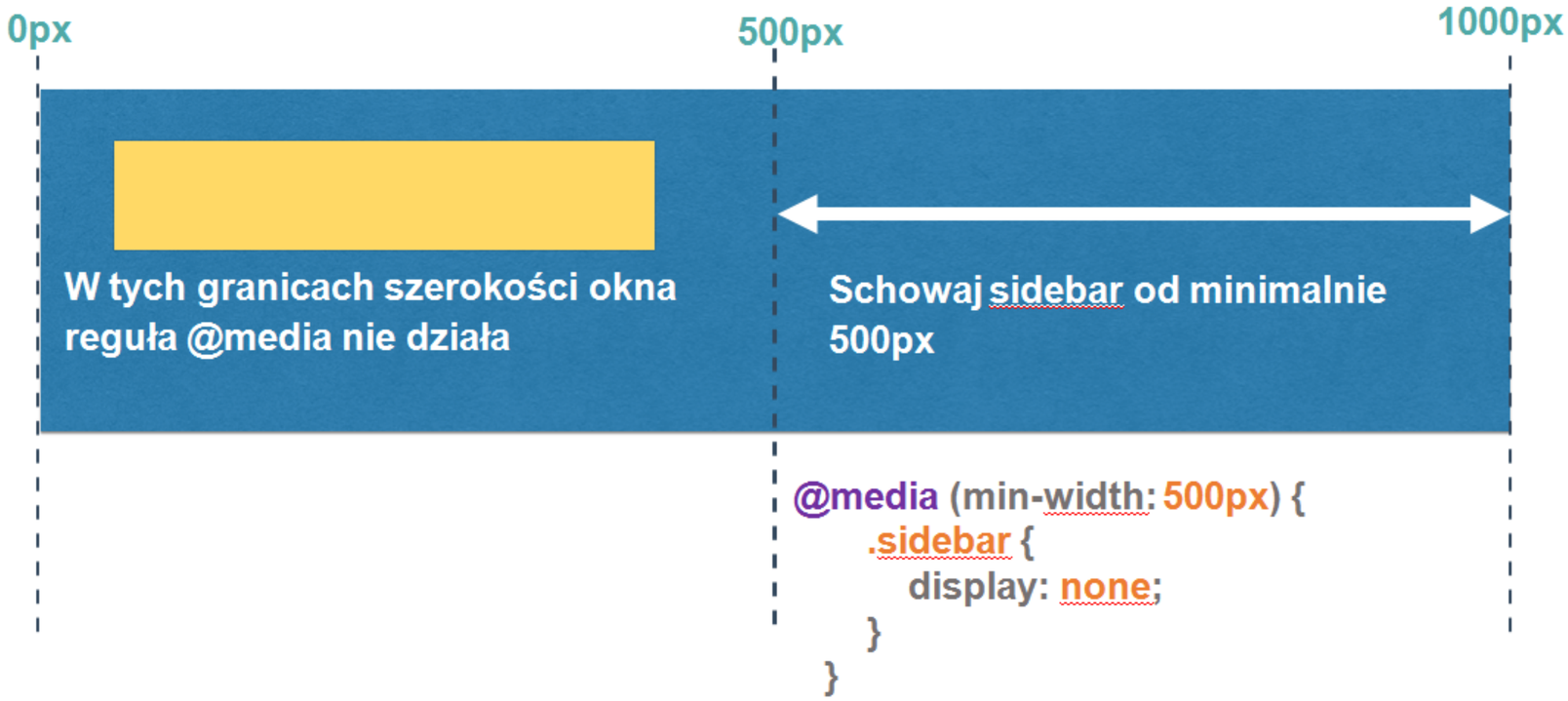
Szerokość - max-width

Jak czytamy wartości max-width w media?



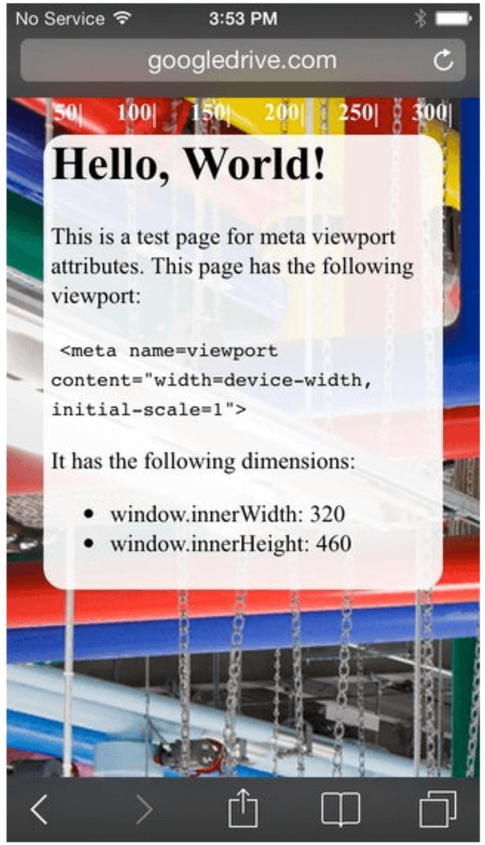
Szerokość - min-width

Jak czytamy wartości min-width w media?

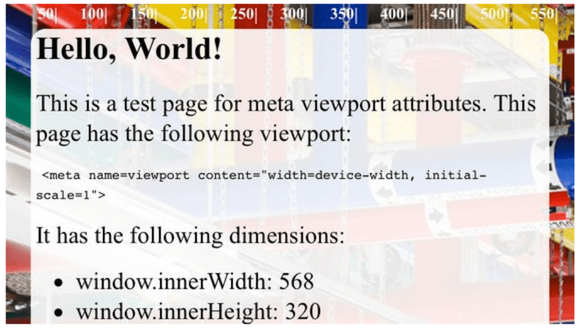


Orientacja

Orientacja pionowa



Orientacja pozioma



Gęstość pikseli a CSS

Piksel fizyczny

Jeden piksel CSS może być reprezentowany przez wiele pikseli fizycznych urządzenia. Taką relację określa się przez wskaźnik `dppx` (dots per px unit).

Jeśli przyjmiemy wartość `1dppx`, to jeden piksel fizyczny odpowiada jednemu pikselowi CSS.

W takim przypadku obrazy o standardowej gęstości pikseli (1 do 1) często wyglądają fatalnie na urządzeniach z większą gęstością pikseli.

Dlatego dla wyświetlaczy z wyższą gęstością pikseli serwuje się obrazki o wyższej definicji.

Np. jeśli obraz ma wymiary CSS `200x200px`, to dla wyświetlacza z rozdzielczością `2dppx` powinien być serwowany obraz o rozdzielczości `400x400` pikseli, w czym pomogą nam `media queries`.

Łączenie warunków media

Media query czyli wyrażenie logiczne `if`

`Media queries` wykorzystują następujące operatory do łączenia warunków:

- **and** (x and y),
- **przecinek** (x or y),
- **not** (np. not x and y = not (x and y) ≠ (not x) and y.

Operator and

Jeśli chcemy by `@media` spełniało kilka warunków **jednocześnie** używamy `and`.

```
@media (min-width: 700px) and
(orientation: landscape) {
}
@media tv and (min-width: 700px) and
(orientation: landscape) {
}
```

Operator AND

Operator and

Jeśli chcemy by `media query` spełniało kilka warunków jednocześnie używamy `and`. Oba warunki muszą być spełnione.

```
@media (min-width: 700px) and (orientation: landscape) {
  .sidebar {
    display: none;
  }
}
```

Operator OR

Operator przecinek

Przecinek łączy warunki tak jak operator logiczny `or`.

Oznacza to, że jeśli którykolwiek z warunków zostanie spełniony, to dana część kodu CSS będzie interpretowana.

```
@media (min-width: 700px), screen and (orientation: landscape) {
  .sidebar {
    display: none;
  }
}
```

Operator NOT

Operator not

Jeśli chcemy, by `media query` nie było przeznaczone dla mediów spełniających dane warunki, używamy operatora `not`.

Operator ten neguje całe wyrażenie i nie może być zastosowany do poszczególnych warunków łączonych przez operator `and` lub **przecinek**.

```
@media not print {
  .sidebar {
    display: none;
  }
}
```

Operator ONLY

Kiedy używamy operatora only?

Używamy tego operatora, jeśli chcemy, by kod CSS nie był interpretowany przez starsze przeglądarki.

Dlaczego?

Operator `only` występuje tutaj jako typ media, który nie istnieje. Starsze przeglądarki nie będą interpretowały zatem dalszej części warunku a tym samym kodu **CSS**.

```
@media only screen and (min-width: 600px) {
}
```

Właściwości mediów

- `min-width/min-height i max-width/max-height` – określenie zakresu wartości szerokości/wysokości obszaru widocznego.
- `min-device-width/min-device-height i max-device-width/max-device-height` – określenie zakresu wartości szerokości/wysokości urządzenia.

```
@media screen and (min-width: 500px)
and (max-height: 800px) { ... }
```

```
@media screen and (min-device-height: 480px)
and (max-device-width: 640px) { ... }
```

Właściwości mediów

- `orientation` – określenie orientacji pionowej lub poziomej urządzenia,
- `resolution` – określenie wartości gęstości pikseli dla danego urządzenia.

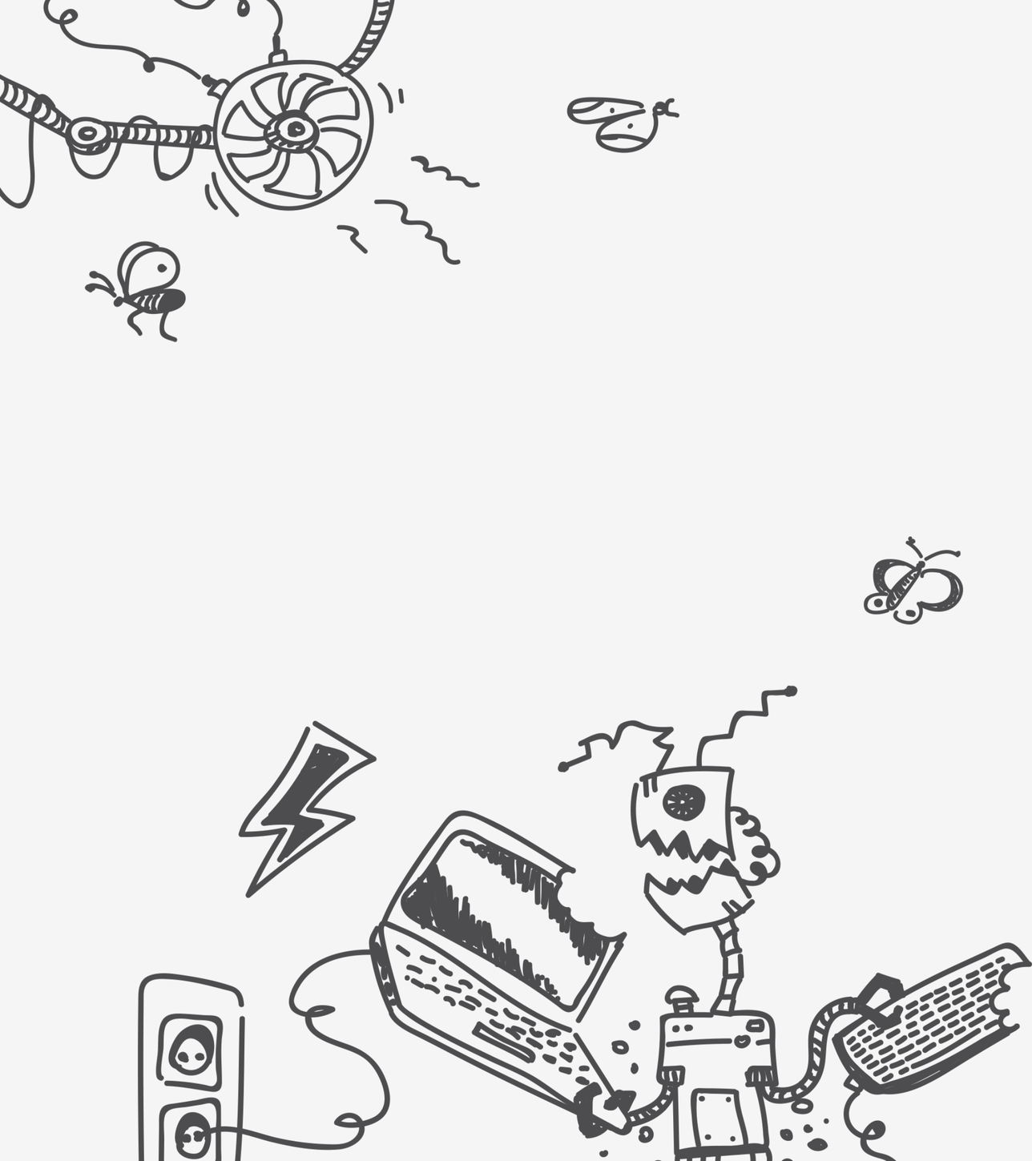
```
@media all and (orientation: portrait) { ... }
```

```
@media print and (min-resolution: 300dpi) { ... }
@media screen and (min-resolution: 2dppx) { ... }
```

Jednostki relatywne, responsywność i viewport

- **RWD** zakłada użycie jednostek relatywnych.
- Ze względu na dużą fragmentację rozmiarów ekranów, dobranie jednostek sztywnych (`px`) - tak by strona wyświetlała się prawidłowo na wszystkich ekranach – jest bardzo trudne. Dlatego lepszym rozwiązaniem jest używanie jednostek relatywnych takich jak:
  - `procenty`,
  - `emy`,
  - `remy`.
- CSS3 pozwala na używanie jednostek relatywnych w odniesieniu do wielkości `viewport`:
  - `vh` (viewport height),
  - `vw` (viewport width),
  - `vmin i vmax.`

Jedna jednostka viewport to 1% jego szerokości lub wysokości.



Media queries w Sass

Media query ze zmienną

Przykład

```
$tablet: "(min-width: 768px) and (max-width: 1023px)";
$desktop: "(min-width: 1024px)";
p {
  font-size: 16px;
  @media #{$tablet} {
    font-size: 18px;
  }
  @media #{$desktop} {
    font-size: 20px;
  }
}
```

Media query jako mixin

Składnia SCSS

```scss
$tablet-width: 768px;
@mixin tablet {
  @media (min-width: #{$tablet-width}) {
   @content;
  }
}
.main-section {
  margin: 0 auto;
  @include tablet {
   padding: 6px 12px;
   width: 900px;
  }
}
```

Składnia CSS

```css
.main-section {
  margin: 0 auto;
}
@media (min-width:768px) {
  .main-section {
    padding: 6px 12px;
    width: 900px;
  }
}
```

Media query jako mixin

Składnia SCSS

```scss
$tablet-width: 768px;
@mixin tablet {
  @media (min-width: #{$tablet-width}) {
   @content;
  }
}
.main-section {
  margin: 0 auto;
  @include tablet {
   padding: 6px 12px;
   width: 900px;
  }
}
```

Składnia CSS

```css
.main-section {
  margin: 0 auto;
}
@media (min-width:768px) {
  .main-section {
    padding: 6px 12px;
    width: 900px;
  }
}
```

Wywołujemy `mixin tablet`.
Przekazujemy do niego tzw. `content`.