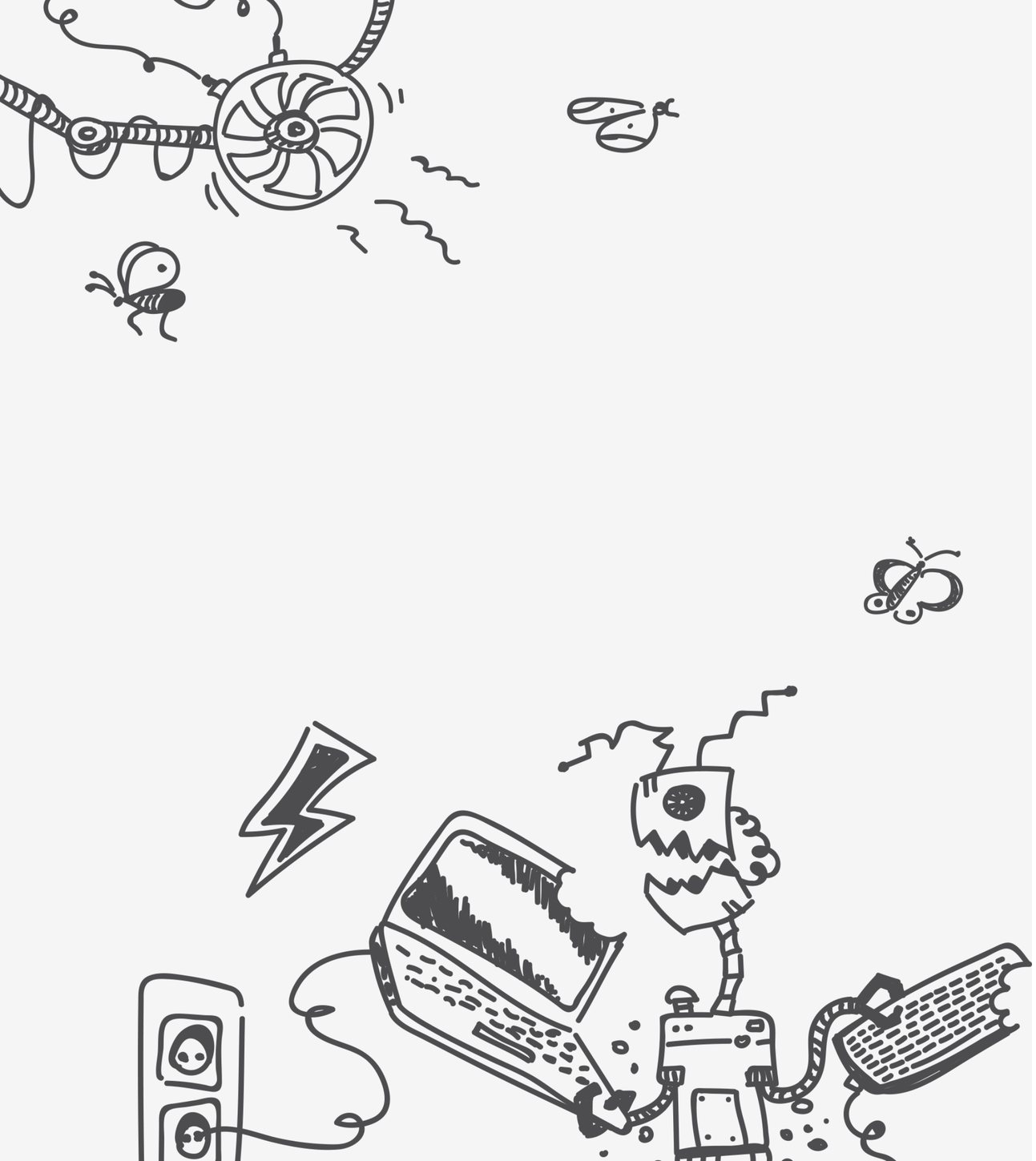


Elastyczne media

Elastyczne media

Czym jest elastyczność?

Media, czyli w głównej mierze obrazy i wideo, powinny również elastycznie dopasować się do szerokości obszaru, w którym są wyświetlane.

Najlepiej definiować szerokość mediów przez zastosowanie jednostek względnych wobec ich rodzica.

```
img, video, canvas {
  max-width: 100%;
}
```

Elastyczne obrazy

Inna metoda

Można dopasować obraz zdefiniowany w **CSS** jako tło. Służy do tego właściwość `background-size`.

Może ona przyjmować wartości zdefiniowanej szerokości `(px, em, %)`, `auto`, `cover`, `contain`, `initial`, `inherit`.

```
img, video, canvas {
  max-width: 100%;
}
.bg-cover {
  width: 100%;
  background-image: url("image.png");
  background-size: cover; /* contain, 100% auto */
  background-repeat: no-repeat;
  background-color: transparent;
}
```

Optymalizacja obrazów

Jak zadbać o optymalizację?

Dobrą praktyką jest optymalizacja rozmiaru i jakości obrazów wyświetlanych na stronach.

Pozwala to na szybsze załadowanie strony oraz oszczędności transferu, które są istotne zwłaszcza na łączach sieci komórkowej.

Warto podjąć próby dostosowania jakości do poszczególnych obrazów, gdyż efekt nie zawsze jest jednakowy.

Przydatny plugin do Gulpa – optymalizacja obrazów: https://github.com/sindresorhus/gulp-imagemin

Program do bezstratnej kompresji obrazów: https://imageoptim.com/pl

Strony do optymalizacji:

- https://tinypng.com/
- https://compressor.io/compress
- https://squoosh.app

CSS Sprites

PNG

`Portable Network Graphics` to format zalecany przez `W3C` do stosowania grafiki na stronach internetowych. Charakteryzuje się bezstratną kompresją obrazu.

Przydaje się do wyświetlania obrazów o niewielkim rozmiarze a zwłaszcza tych, które wymagają częściowej przezroczystości.

W porównaniu do innych formatów rozmiar takiego obrazu jest z reguły najmniejszy przy zachowaniu wysokiej jakości.

CSS Sprites

Minimalizowanie zapytania

Aby zminimalizować zapytania wysyłane do serwera (`HTTP Requests`) obrazy można zapisać w jednym pliku i odwoływać się do nich, przesuwając odpowiednio tło.

Flagi widoczne po prawej stronie są jednym plikiem PNG.

Przydatny plugin generujący kod z pozycjami elementów: https://github.com/twolfson/gulp.spritesmith

CSS Sprites

Umieszczanie CSS Sprites

Aby umieścić obrazy (w tym przypadku – flagi) w trzech różnych elementach strony, można użyć następującego kodu CSS.

```
.flags-canada, .flags-mexico, .flags-usa {
  background-image: url("../images/flags.png");
  background-repeat: no-repeat;
}
```

```
.flags-canada {
  height: 128px;
  background-position: -5px -5px;
}
.flags-usa {
  height: 135px;
  background-position: -5px -143px;
}
```

```
.flags-mexico {
  height: 128px;
  background-position: -5px -288px;
}
```

Responsywne obrazy CSS

Jak optymalizować?

Obrazy wstawione za pomocą CSS również można optymalizować dla różnego typu mediów.

Dla małych ekranów serwujemy obraz o niskiej rozdzielczości, a dla dużych – o wysokiej, ale aby ich wielkość nie przekraczała górnej wartości punktu granicznego.

```
.example {
  height: 400px;
  background-repeat: no-repeat;
  background-size: contain;
}

@media (max-width: 499px) {
  .example {
    background-image: url("small.png");
  }
}

@media (min-width: 500px) {
  .example {
    background-image: url("large.png");
  }
}
```

Responsywne obrazy CSS

Ekrany Retina

Podobnie możemy wykorzystać `media query` dla ekranów z większą gęstością pikseli (Retina).

```
.example {
  background-image: url('ico.png');
}

@media only screen and (-webkit-min-device-pixel-ratio: 2),
       only screen and (min-resolution: 2dppx) {
  .example {
    background-image: url('ico@2x.png');
  }
}
```

Responsywne obrazy

Innym wyzwaniem jest serwowanie odpowiedniego pliku obrazu umieszczonego w tagu HTML tak, aby był sformatowany i zoptymalizowany dla danego typu urządzeniu lub ekranu.

Trudność polega na braku łatwej i ustandaryzowanej metody.

Metoda srcset

Co to takiego?

Jest to jedna z proponowanych przez W3C metod serwowania innego źródła dla danego tagu `<img>` w zależności od ekranu.

Warunkiem może być gęstość pikseli lub szerokość, jaką na ekranie będzie miał obraz.

Jest to wciąż robocza wersja HTML, więc w wersji końcowej mogą pojawić się zmiany.

Dlatego też ta metoda nie została wdrożona we wszystkich przeglądarkach.

```
<img width="100"
     height="150"
     src="image100.jpg"
     srcset="image200.jpg 2x" />
```

- Przykład zastosowania atrybutu **srcset**

  http://webkit.org/demos/srcset

http://www.w3.org/html/wg/drafts/html/master/semantics.html#attr-img-srcset

Metoda picture

Metoda ta pozwala na użycie odpowiedniego źródła dla danego `media query`.

```
<picture>
  <source media="(min-width: 45em)" srcset="large.jpg">
  <source media="(min-width: 32em)" srcset="med.jpg">
  <img src="small.jpg" alt="The president giving an award.">
</picture>
```

Polyfill

- Nie wszystkie przeglądarki obsługują `picture`, `srcset` czy też **sizes**.
- Możemy więc podłączyć skrypt, który pozwoli na obsługę tych metod. Jest to tzw. `polyfill`. Jednym z nich jest Picturefill.

Ikony wektorowe

Deklaracja @font-face

Popularnym rozwiązaniem dostosowania ikon do różnych ekranów jest użycie ich w postaci czcionki i deklaracji `@font-face`.

Dzięki temu ikony są zdefiniowane za pomocą wektorów i mogą być skalowane do różnych rozmiarów bez strat na jakości.

Ważną częścią podczas tworzenia projektu jest odpowiednie radzenie sobie z czcionkami np. svg użytymi w projekcie. Możemy użyć wtyczki do Gulpa https://github.com/jvanaert/gulp-webfont lub Icomoon https://icomoon.io.

Gotowe i darmowe czcionki ikon

- FontAwesome,
- Fontello (generator),
- Genericons,
- Entypo.

Ikon wektorowych używa się ich przez użycie klas, np:

```
<i class="fa fa-globe"></i>
```

Skalowanie zagnieżdżonego wideo

Największą trudnością przy skalowaniu zagnieżdżonego wideo (np. w tagu `iframe`) jest zachowanie jego proporcji.

W przeciwieństwie do obrazków, wysokość elementu `iframe` musi być zdefiniowana i nie jest skalowana proporcjonalnie do jego szerokości.

Jest jednak na to sposób.

Skalowanie zagnieżdżonego wideo

Kod HTML

```
<figure>
  <iframe src="https://www.youtube.com/embed/4Fqg43ozz7A">
  </iframe>
</figure>
```

Kod CSS

```
figure {
  height: 0;
  padding-bottom: 56.25%; /* 16:9 */
  position: relative;
  width: 100%;
}
```

```
iframe {
  height: 100%;
  left: 0;
  position: absolute;
  top: 0;
  width: 100%;
}
```

Typografia

Punkt graniczny

Zakłada się, że przyszłość mobile opiera się głównie na typografii.

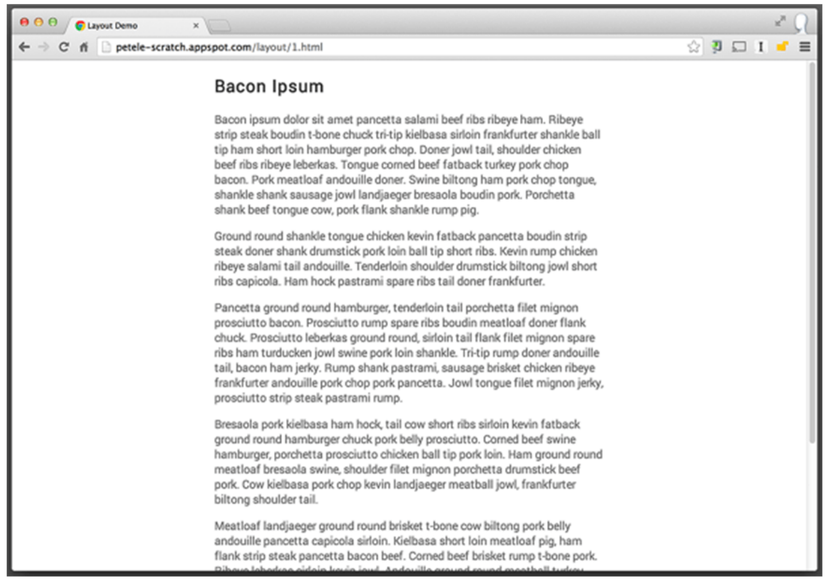
Mało dostępnego miejsca wymusza redukcję liczby informacji widocznych na ekranie.

Czytelność tekstu staje się zatem kluczowa.

Według klasycznych zasad gwarantujących czytelność tekstu, idealna szpalta powinna zawierać 70–80 znaków w wierszu (około 8–10 wyrazów).

Za każdym razem, gdy wiersz w bloku tekstu przekroczy 10 wyrazów, należy rozważyć utworzenie punktu granicznego.

Responsywna typografia



Responsywna typografia

- Responsywna typografia powinna opierać się na jednostkach względnych.
- Taki tekst łatwiej skalować i dostosowywać do preferencji użytkownika.

```
html { font-size: 62.5%; }
h1 { font-size: 10vw; }
h2 { font-size: 2.2rem; /* 22px */ }
h3 { font-size: 1.8rem; /* 18px */ }
```

```
@media (min-width: 480px) {
  h2 { font-size: 5rem; /* 50px */ }
  h3 { font-size: 3rem; /* 30px */ }
}
```

```
@media (min-width: 1024px) {
  h2 { font-size: 8rem; /* 80px */ }
  h3 { font-size: 5rem; /* 50px */ }
}
```