

# Zaawansowany HTML i CSS Dzień 3

v3.3

# Plan

- Stylowanie formularzy
- Tworzenie układu strony
- CSS Reset
- Dobre praktyki
- Koncepcja nazewnictwa klas

# Stylowanie formularzy

# placeholder

**Placeholder** to tekst, który widzisz w linii formularza, zanim zaczniesz uzupełniać to pole. Można go ostylować za pomocą pseudoklas. Przeglądarki wymagają korzystania z prefiksów. Uwaga! W Firefox każdy **placeholder** ma **opacity** ustawione na **0.4**!

```
<input type="text" name="name"
        placeholder="wpisz swoje imię"/>
```

```
::-webkit-input-placeholder {
    color: red;
    /* Chrome, Opera, Safari */
}
::-moz-placeholder {
    color: red;
    /* Firefox */
}
::-ms-input-placeholder {
    color: red;
    /* IE 10+ */
}
```

# focus

Przeglądarki domyślnie mają ustawione obramowanie wokół pól formularzy, gdy kursor myszy znajduje się wewnątrz. Możemy jednak nadpisać tę własność.

Lepiej nie nadpisywać wartości **focus** ze względu na użytkowników, którzy używają urządzeń dotykowych lub przełączają się pomiędzy formularzami za pomocą tabulatora.

Imię

```
input:focus {  
  outline: 0;  
}
```

Imię

# submit

Możemy stylować poszczególne elementy i nadpisywać domyślne wartości.

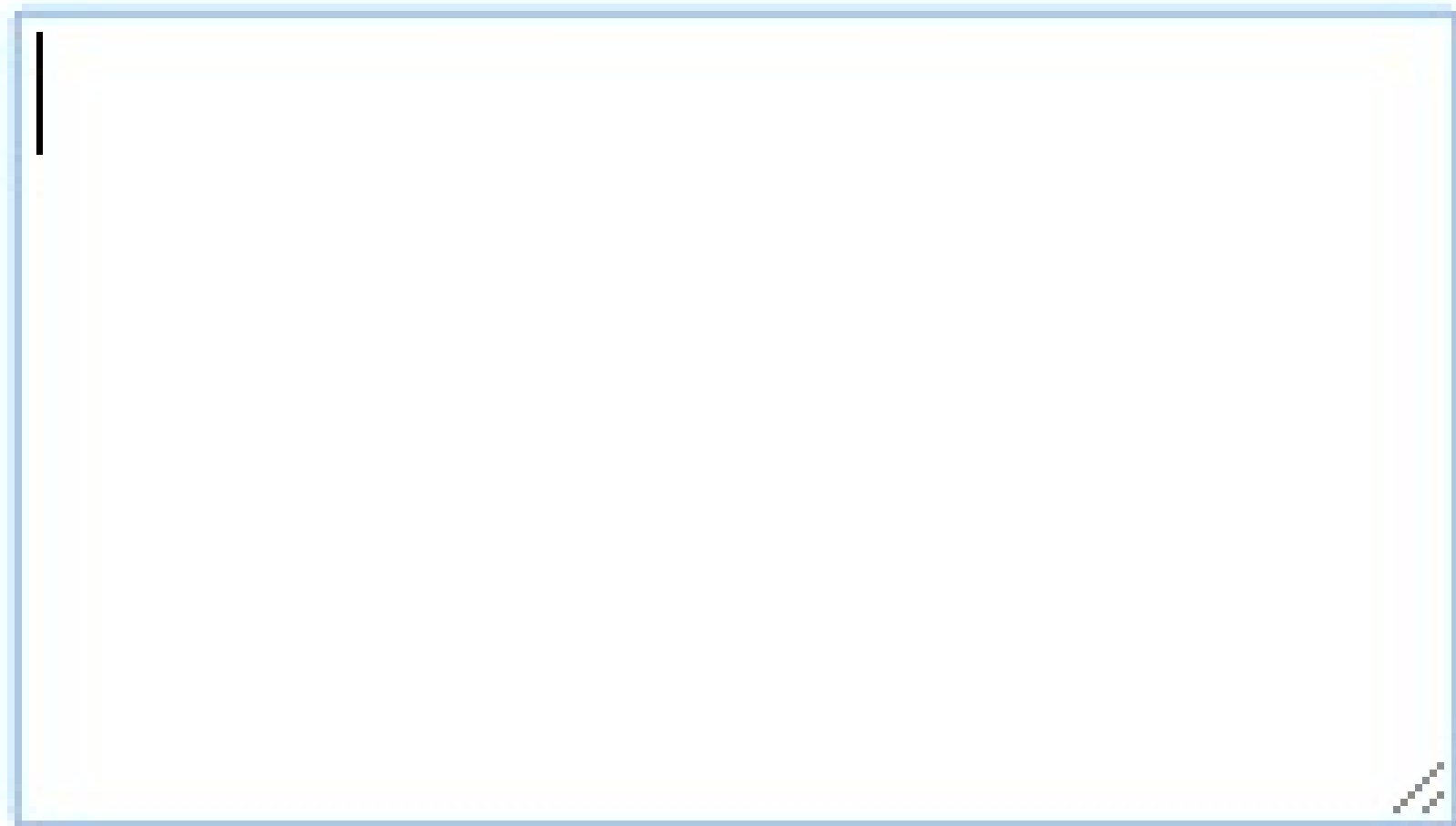


```
<input type="submit" value="Wyślij">
```

```
input[type="submit"] {  
  background-color: blue;  
  border: none;  
  color: white;  
  border-radius: 20%;  
}
```

# textarea

Jeżeli nie chcemy, aby użytkownik mógł powiększać pole **textarea**, możemy je zablokować za pomocą CSS.



```
textarea {  
  resize: none;  
}
```

# Przykłady formularzy CSS

- <http://codepen.io/miroot/pen/qwlgC>
- <http://cssdeck.com/labs/login-form-3>
- <http://red-team-design.com/slick-login-form-with-html5-css3>
- <http://codepen.io/petertoth/pen/BtGkp>
- <http://codepen.io/kman/pen/DFAzG>
- <http://cssdeck.com/labs/apple-dev-login>
- <http://designerfuel.tumblr.com/post/15555140593/login-form-psd-live>



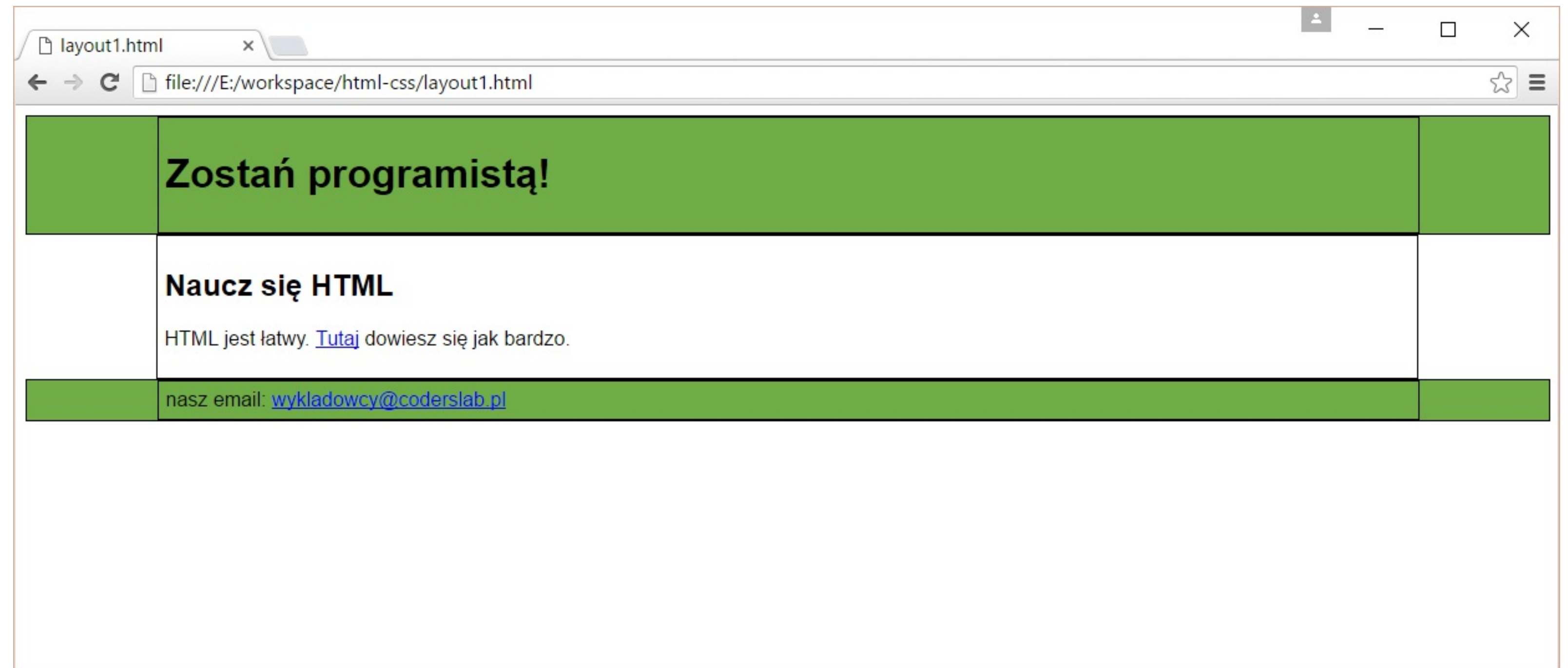
Czas na zadania

# Tworzenie układu strony

# Co chcemy osiągnąć?

Chcemy osiągnąć następujący efekt / stworzyć elementy:

- strona na środku ekranu,
- nagłówek,
- treść,
- stopka.



# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">

  </head>
  <body>
  </body>
</html>
```

# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">

  </head>
  <body>
  </body>
</html>
```

→ Deklaracja typu - HTML,

# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">
  </head>
  <body>
  </body>
</html>
```

- Deklaracja typu - HTML,
- nagłówek HTML,

# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">
  </head>
  <body>
  </body>
</html>
```

- Deklaracja typu - HTML,
- nagłówek HTML,
- tytuł strony,

# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">
  </head>
  <body>
  </body>
</html>
```

- Deklaracja typu - HTML,
- nagłówek HTML,
- tytuł strony,
- kodowanie czcionek,



# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">
  </head>
  <body>
  </body>
</html>
```

- Deklaracja typu - HTML,
- nagłówek HTML,
- tytuł strony,
- kodowanie czcionek,
- deklaracja pliku CSS,

# Budujemy szkielet strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Zostań programistą
    </title>
    <meta charset="UTF-8">
    <link rel="stylesheet"
          type="text/css"
          href="style.css">

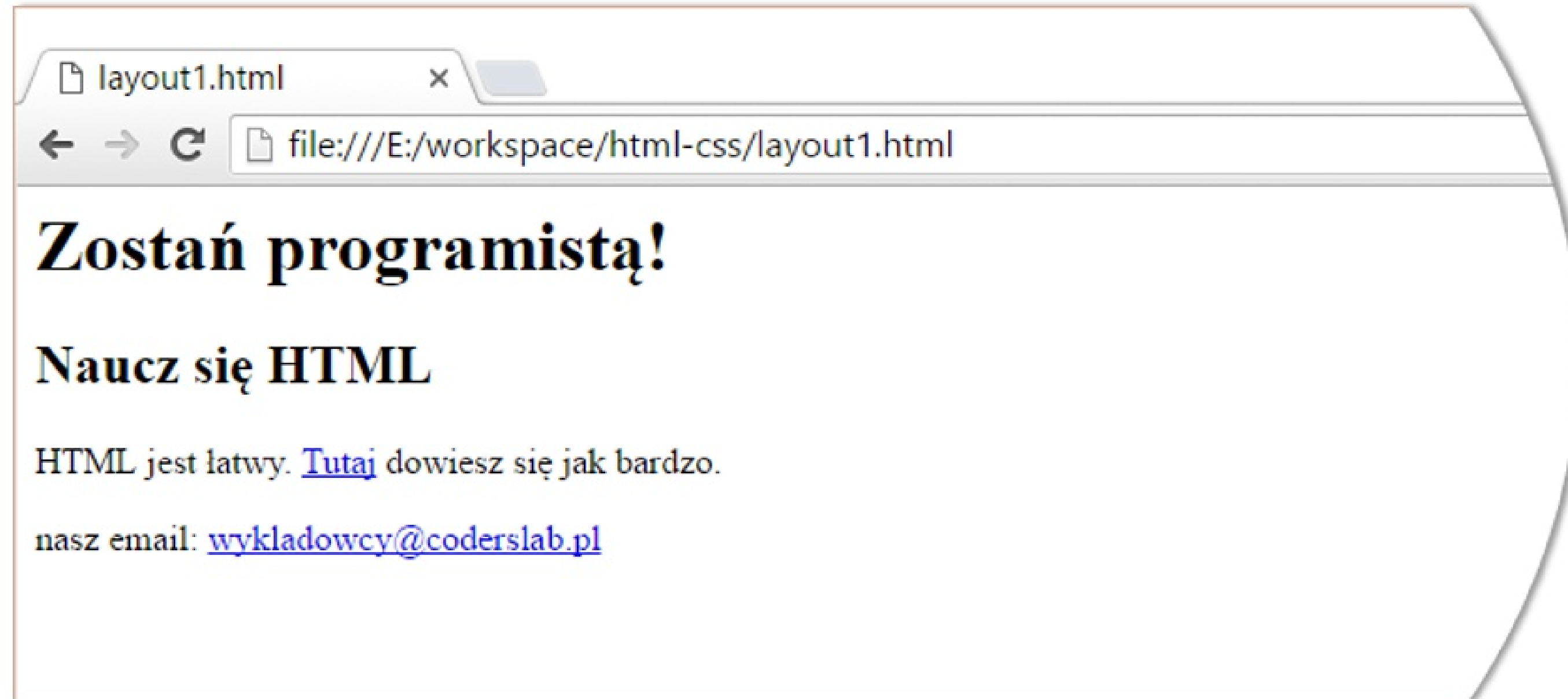
  </head>
  <body>
  </body>
</html>
```

- Deklaracja typu - HTML,
- nagłówek HTML,
- tytuł strony,
- kodowanie czcionek,
- deklaracja pliku CSS,
- treść strony.

# Tworzymy zawartość strony

Tworzymy semantyczną zawartość naszej strony:

- nagłówek strony ( **header**),
- artykuł o HTML-u ( **section**),
- stopka z adresem email ( **footer**).



# Tworzymy zawartość strony

```
<body>
  <header>
    <h1>Zostań programistą!</h1>
  </header>
  <section>
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy. <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </section>
  <footer>
    <div class="container">nasz email: ...</div>
  </footer>
</body>
```

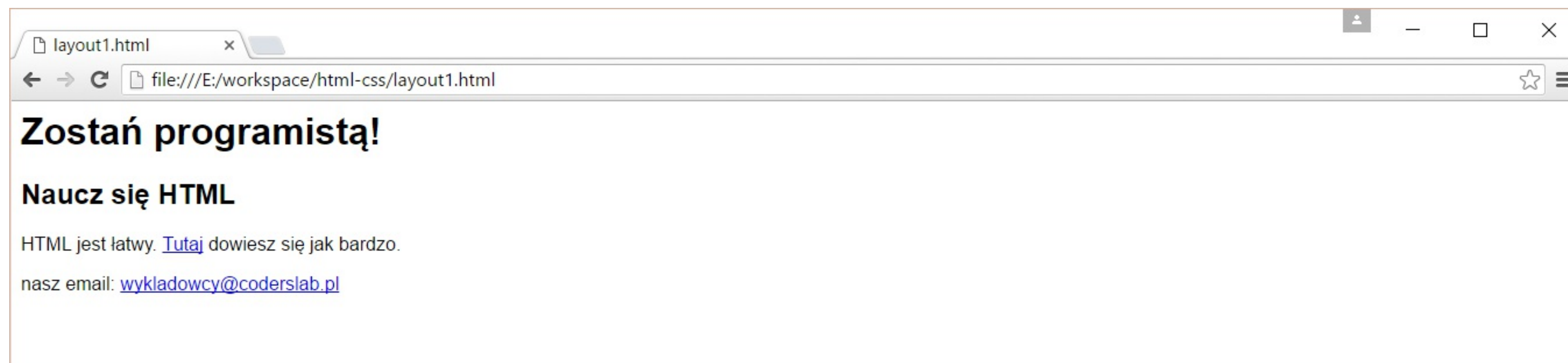
# Czcionki!

Czcionka, której używamy na naszej stronie, to **Arial**.

Musimy pamiętać o tym, żeby dać przeglądarce jakąś alternatywę, w przypadku, gdy **Arial** będzie niedostępny.

A jeśli przeglądarka nie znajdzie niczego - wtedy musimy użyć tzw. *fallbacku*.

```
body {  
  font-family: Arial, Helvetica, sans-serif;  
}
```



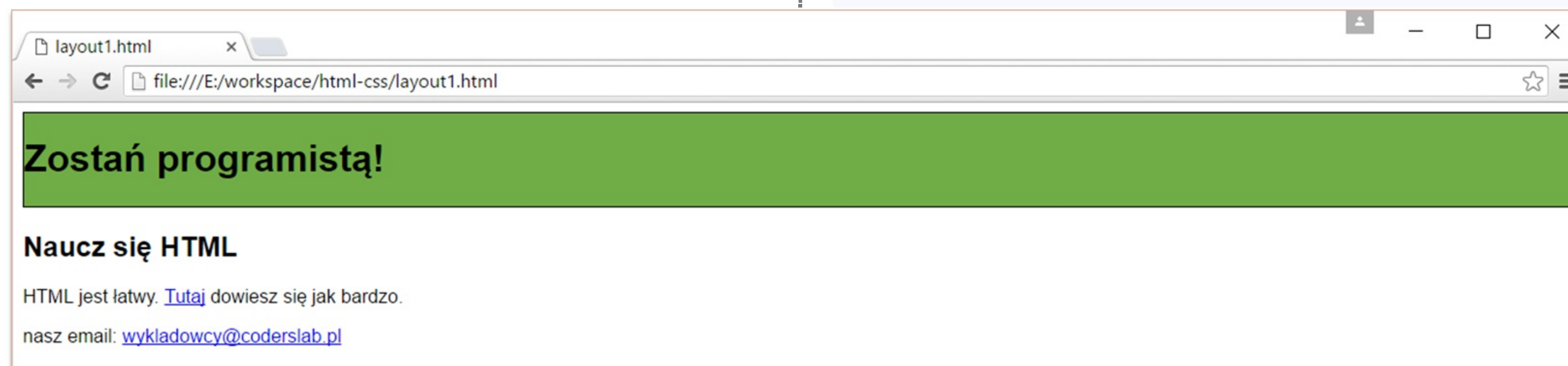
# Ustawiamy nagłówek na środku

W tym celu musimy ustawić szerokość nagłówka na 100%.

Pamiętajmy o obramowaniach każdej części. Będziemy dzięki temu wiedzieć, gdzie dany element się zaczyna, a gdzie kończy.

```
<header>
  <h1>Zostań programistą!</h1>
</header>
```

```
header {
  width: 100%;
  border: 1px solid #000;
  background-color: #70AD47;
}
```





# Ustawiamy nagłówek na środku

Hm... Nie do końca o to chodziło.

Naszym celem jest nagłówek w ramce umieszczony na środku strony.

W tym celu najlepiej obudować wewnątrz elementu **header** elementem **div**, któremu nadamy odpowiednią szerokość i przesuniemy na środek strony.

```
<header>
  <div class="container">
    <h1>Zostań programistą!</h1>
  </div>
</header>
```

```
.container {
  margin: 0 auto;
  width: 1000px;
  border: 1px solid black;
}
```

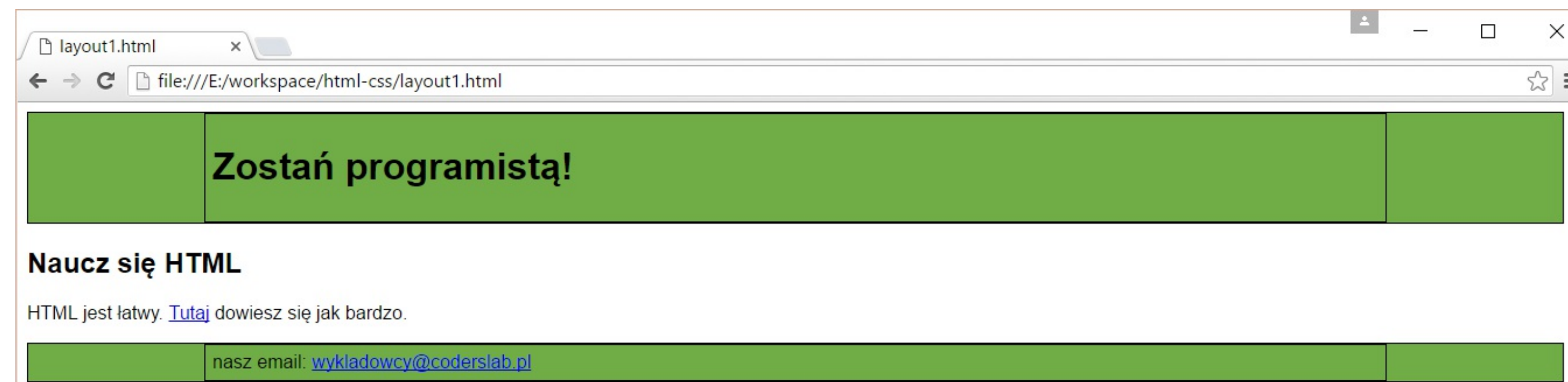


# Formatowanie stopki

Aby odpowiednio sformatować stopkę, wystarczy zrobić z nią to samo, co z nagłówkiem i zdefiniować odpowiednie style.

```
<footer>
  <div class="container">
    nasz email: ...
  </div>
</footer>
```

```
header, footer {
  width: 100%;
  border: 1px solid #000;
  background-color: #70AD47;
}
```





# Zawartość strony

Bardzo podobnie obejdziemy się z sekcją, w której przechowujemy zawartość strony.

W tym celu zmodyfikujemy HTML, tak by lepiej oddawał semantykę naszego dokumentu: opakujemy artykuł w element **article**.

```
<section>
  <article>
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</section>
```



# Zawartość strony

Następnie wyśrodkujemy element article tak, jak zrobiliśmy to poprzednio z nagłówkiem i stopką.

**Voilà!**

```
<section>
  <article class="container">
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</section>
```

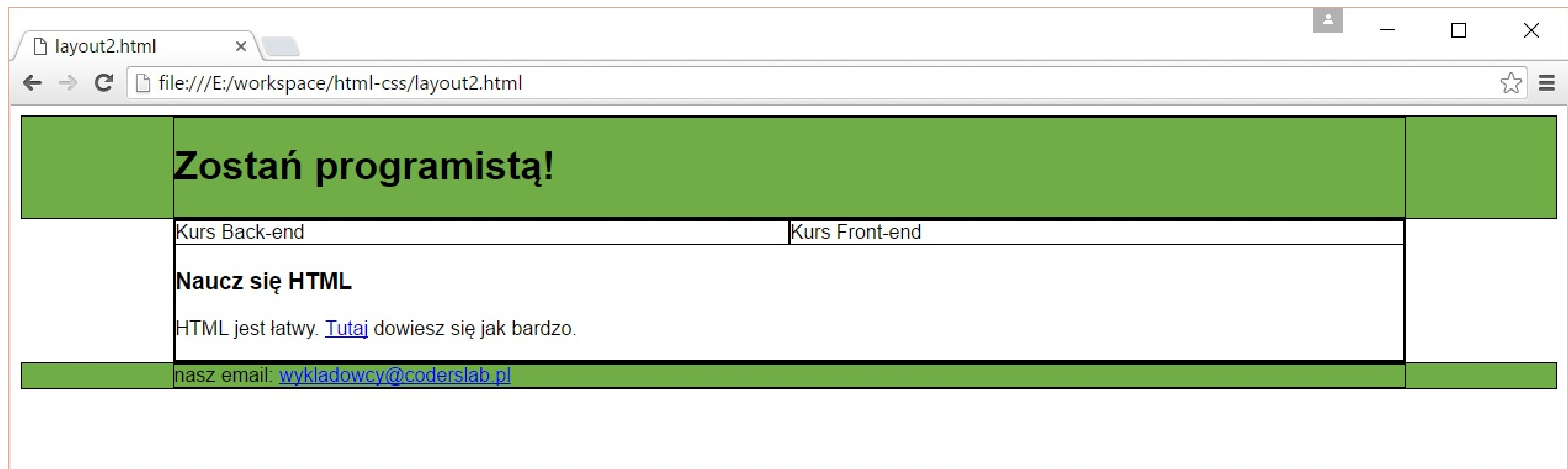


**Strona z wieloma  
artykułami**

# Co chcemy osiągnąć?

Chcemy dodać kilka artykułów na stronie, tak jak na ilustracji.

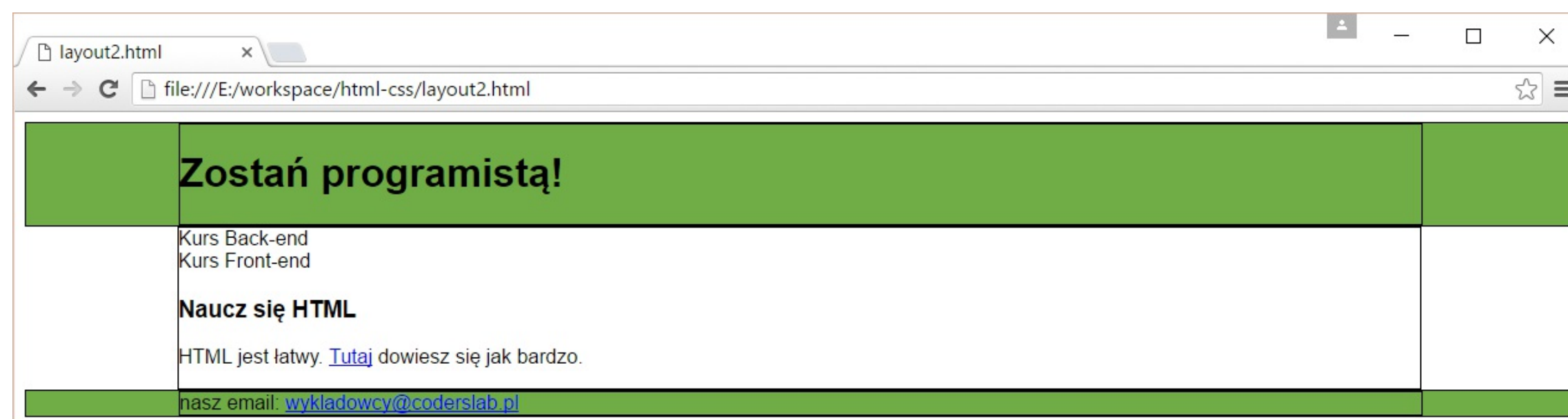
Można to osiągnąć na kilka sposobów: np. używając CSS-owej dyrektywy **float**, albo **display**.



# Modyfikujemy zawartość strony

Modyfikujemy semantyczną zawartość naszej strony:

- usuwamy klasę **.container** z elementu **article**,
- dodajemy klasę **.container** do elementu **section** (kto wie dlaczego?),
- dodajemy kolejne artykuły do strony.



```
<section class="container">
  <article>Kurs Back-end</article>
  <article>Kurs Front-end</article>
  <article>
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</section>
```

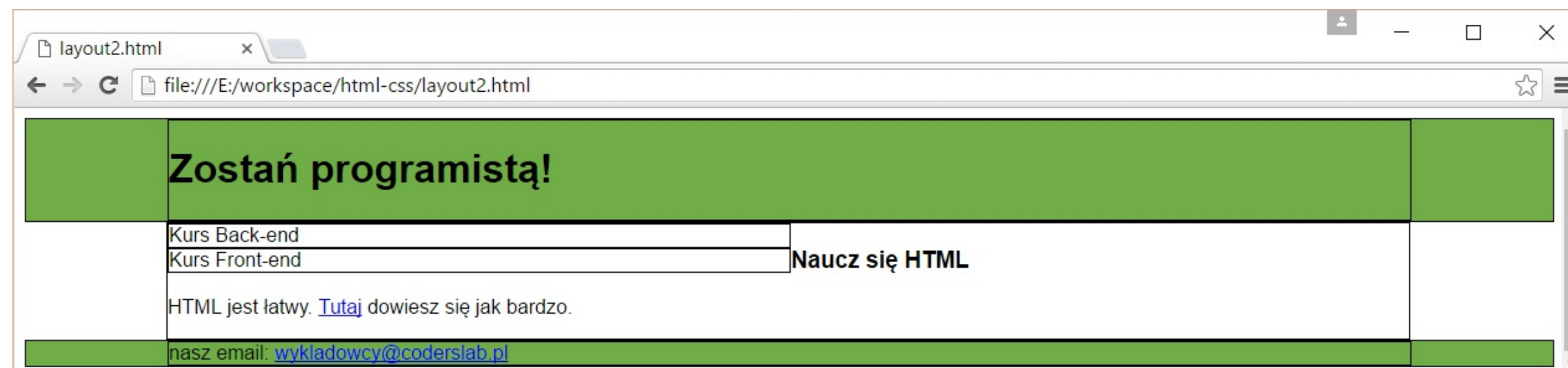
# Górne artykuły, styl float

Każdy z górnych artykułów powinien:

- mieć szerokość 50% całego kontenera,
- opływać inne elementy z lewej strony.

Zdefiniujemy dla niego klasę **.small-box**, w której zamieścimy te dyrektywy.

Pamiętajmy o **border**!



```
<article class="small-box">
    Kurs Back-end
</article>
<article class="small-box">
    Kurs Front-end
</article>
```

```
.small-box {
    float: left;
    width: 50%;
    border: 1px solid black;
}
```



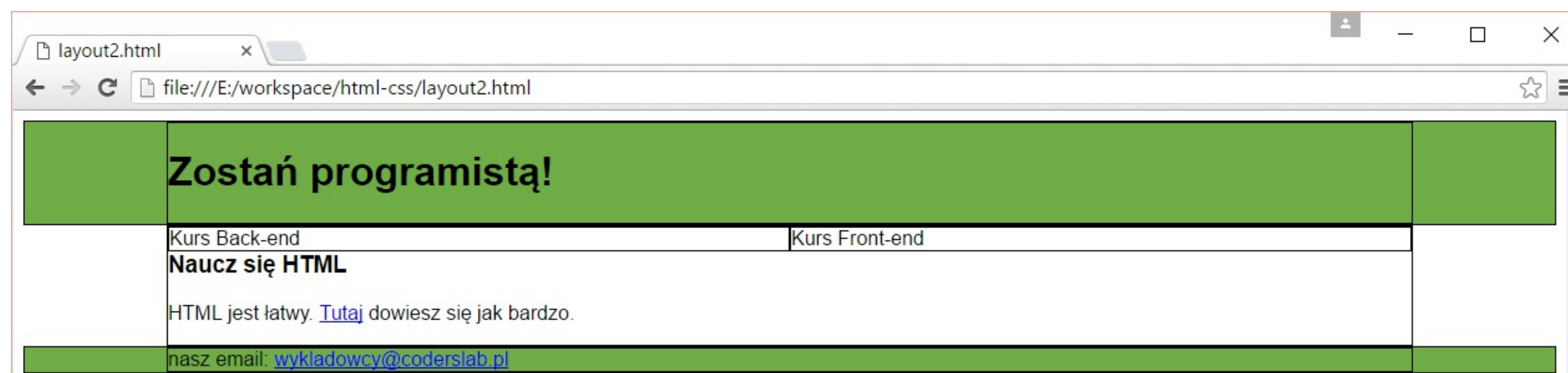
# Ustawiamy górne artykuły

Ups! Nie wygląda to dobrze...

Zapomnieliśmy o „rozpychaniu” pudełka elementu przez ramkę ( **border** )!

Aby to naprawić, najlepiej zdefiniować wszystkim elementom właściwość **box-sizing** i ustawić jej wartość **border-box**.

```
* {  
  box-sizing: border-box;  
}  
.small-box {  
  float: left;  
  width: 50%;  
  border: 1px solid black;  
}
```



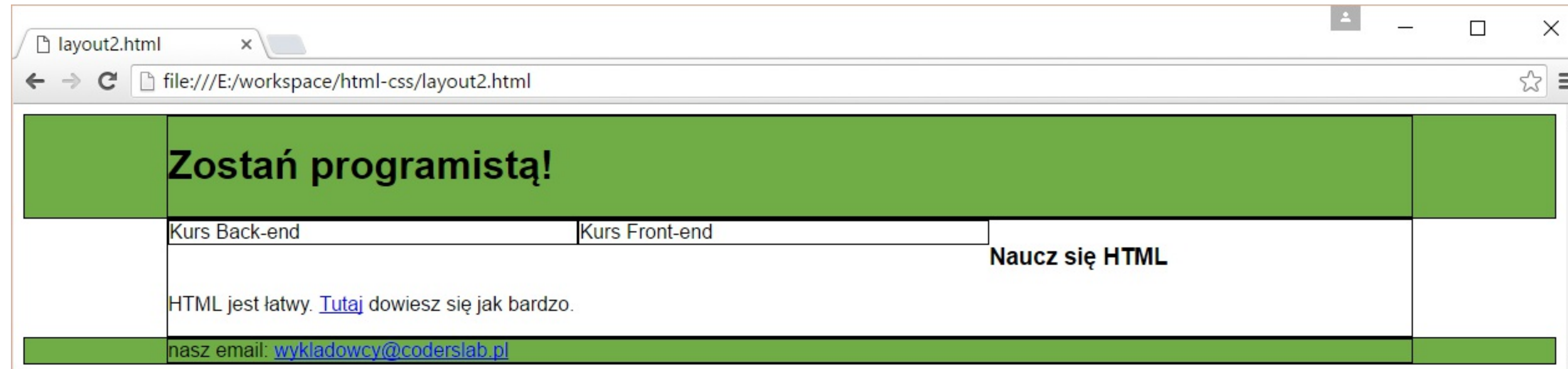
# Dolny artykuł

Czy dolny artykuł jest poprawnie sformatowany?

**Nie.** Przekonamy się o tym wówczas, gdy zmodyfikujemy klasę **.small-box** tak, by zajmowała nie 50%, a np. 33% powierzchni.

Musimy jawnie określić, że dolny artykuł ma się wyświetlać pod górnymi.

```
.small-box {  
  float: left;  
  width: 33%;  
  border: 1px solid black;  
}
```





# Dolny artykuł

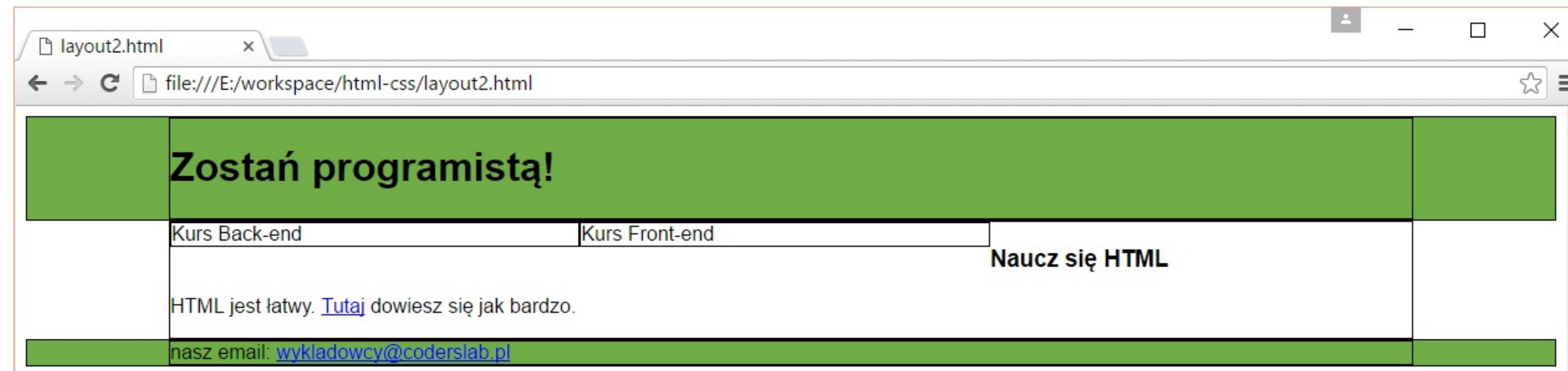
Czy dolny artykuł jest poprawnie sformatowany?

**Nie.** Przekonamy się o tym wówczas, gdy zmodyfikujemy klasę **.small-box** tak, by zajmowała nie 50%, a np. 33% powierzchni.

Musimy jawnie określić, że dolny artykuł ma się wyświetlać pod górnymi.

```
.small-box {  
  float: left;  
  width: 33%;  
  border: 1px solid black;  
}
```

To tylko na próbę! Po sprawdzeniu przywróćcie wartość 50%!



# Dolny artykuł

W tym celu musimy znowu semantycznie zmodyfikować HTML:

Opakujemy każdy rząd artykułów w klasę **.row**.

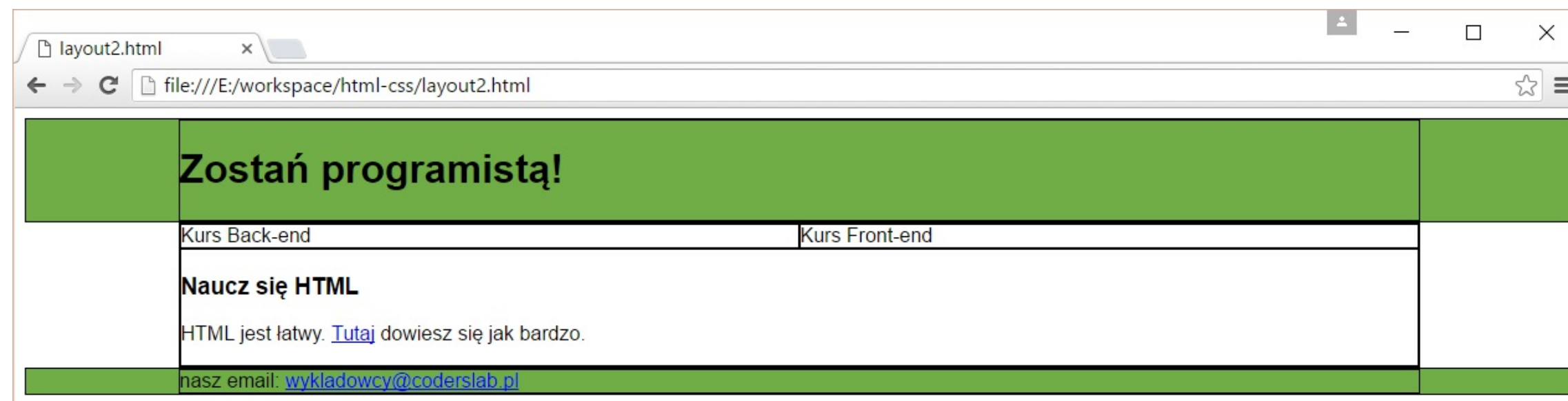
Dolnemu artykułowi nadajemy klasę **.large-box**.

```
<div class="row">
  <article>Kurs Back-end</article>
  <article>Kurs Front-end</article>
</div>
<div class="row">
  <article class="large-box">
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</div>
```

# Dolny artykuł

Sztuczka polega na tym, że koniec pudełka klasy **.row** musi zerować styl **float** (użyj do tego właściwości **clear**).

Musi to być zrobione zaraz po zakończeniu **.row**. Najlepiej użyć do tego pseudoelementu **:after**.



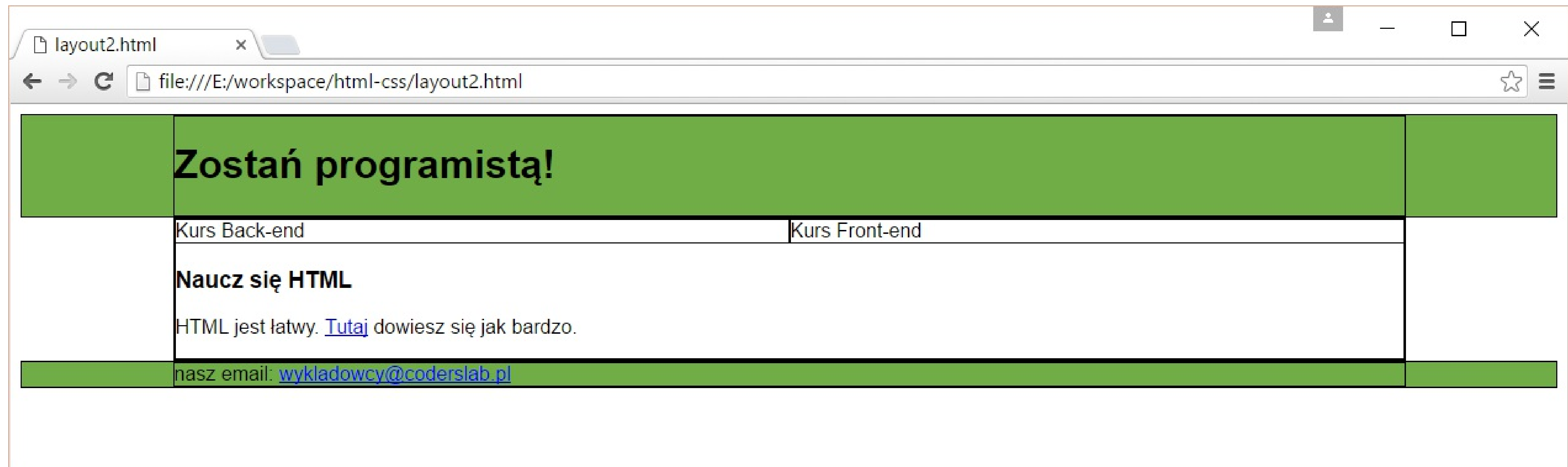
Należy też wyczyścić zawartość pseudoelementu oraz ustawić wyświetlanie na blokowe.

```
.row:after {  
  clear: both;  
  display: block;  
  content: ' ';  
}  
.large-box {  
  border: 1px solid black;  
}
```

**Strona z wieloma  
artykułami:  
alternatywa**

# Co chcemy osiągnąć?

Podobny efekt można uzyskać prościej, używając stylu **display**, który jest łatwiejszy do opanowania, ale ma pewne ograniczenia.



# Przywracamy stary HTML

Aby sformatować stronę CSS-em **display**, nie potrzebujemy tak skomplikowanej struktury. Usuńmy **div**-y o klasie **.row**.

```
<div class="row">
  <article>Kurs Back-end</article>
  <article>Kurs Front-end</article>
</div>
<div class="row">
  <article class="large-box">
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</div>
```

# Przywracamy stary HTML

Aby sformatować stronę CSS-em **display**, nie potrzebujemy tak skomplikowanej struktury. Usuńmy **div**-y o klasie **.row**.

```
<div class="row">
  <article>Kurs Back-end</article>
  <article>Kurs Front-end</article>
</div>
<div class="row">
  <article class="large-box">
    <h2>Naucz się HTML</h2>
    <p>
      HTML jest łatwy.
      <a href="#">Tutaj</a>
      dowiesz się, jak bardzo.
    </p>
  </article>
</div>
```

Te elementy usuwamy z HTML-a.



# Górne artykuły, styl display

Żeby ustawić górne artykuły w jednym rzędzie, należy ustawić właściwość **display** na **inline-block**.

```
.small-box {  
  width: 496px;  
  display: inline-block;  
  border: 1px solid black;  
}
```

Spowoduje to ustawianie kolejnych elementów, posiadających ten styl, obok siebie. W momencie, gdy kolejny element przekroczy granicę kontenera, linia zostanie złamana i element spadnie do kolejnej.



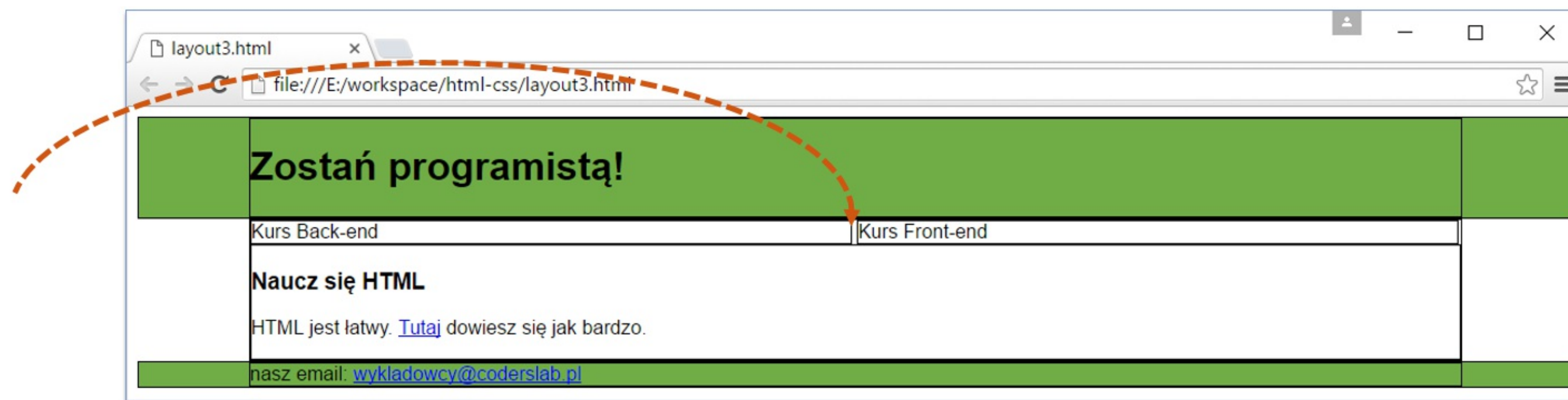


# Górne artykuły, styl display

Zwróćcie uwagę na przerwy pomiędzy górnymi artykułami. Między elementami wyświetlanymi jako **inline-block**, zawsze dodawana jest kilkupikselowa przerwa, dlatego że renderowane się białe znaki pomiędzy elementami inline-blockowymi. Można ją usunąć, używając różnych technik, niemniej jest to czasochłonne i zwyczajnie nieopłacalne.

Jeśli chcecie, aby elementy dokładnie do siebie przylegały, użyjcie metody, wykorzystującej dyrektywę **float**.

Między elementami wyświetlanymi jako inline-block, zawsze dodawana jest kilkupikselowa przerwa.



Czas na zadania

# CSS Reset

# CSS Reset

CSS Reset to kod resetujący wszystkie podstawowe style elementów.

Style ustawione wewnątrz tego pliku resetują domyślne ustawienia niektórych elementów. Przykładem jest usunięcie marginesu dla elementu **body**.

Niestety CSS Reset zeruje również **outline** w pseudoklasie **:focus**, co jest niezgodne z odpowiednim podejściem accessibility i uniemożliwia osobom korzystającym z czytników stron na prawidłowe użytkowanie. Dlatego korzystając z pliku resetującego trzeba doprojektować **:focus** dla butonów, inputów, itp.

# CSS Reset

Najprostszy reset CSS możemy zrobić sami zerując margines wewnętrzny i zewnętrzny dla każdego elementu:

```
* {  
  padding: 0;  
  margin: 0;  
}
```

Przykład kodu reset CSS:

- <http://github.com/murtaugh/HTML5-Reset/blob/master/assets/css/reset.css>
- Kod znajdziesz też w dołączonym pliku **reset.css**.



# Dobre praktyki

# Wcięcia

Przy tworzeniu kodu HTML i CSS pamiętajmy o odpowiednich wcięciach. Kod z wcięciami lepiej się czyta i poprawia.

```
<html lang="pl-PL">
  <head>
    <title>To jest tytuł</title>
    <meta charset="UTF-8">
    <meta name="description"
      content="opis">
  </head>
</html>
```

```
.my_class {
  font-size: 28px;
  color: #309;
}
```

# Nagłówki

Dobłą praktyką jest używanie nagłówków według hierarchii ( **<h1>** - **<h6>**) zgodnie z tematyką strony. Czyli im ważniejsza treść tym wyższy nagłówek.

Pamiętajmy o tym aby nie robić skoków w nagłówkach tzn. jeżeli mamy na stronie 3 różne nagłówki to powinny to być **<h1>**, **<h2>** i **<h3>**, a nie **<h1>**, **<h3>**, **<h6>**.

Ponadto na stronie powinien znajdować się tylko jeden **h1** co ma znaczenie dla SEO i Accessibility.

Według specyfikacji HTML5 każdy element podziału strony czyli: **<body>**, **<section>**, **<article>**, **<nav>** i **<aside>** powinien mieć nagłówek.

Dobrym narzędziem do sprawdzenia nazw sekcji w dokumencie jest rozszerzenie do Chrome: [Outliner](#).

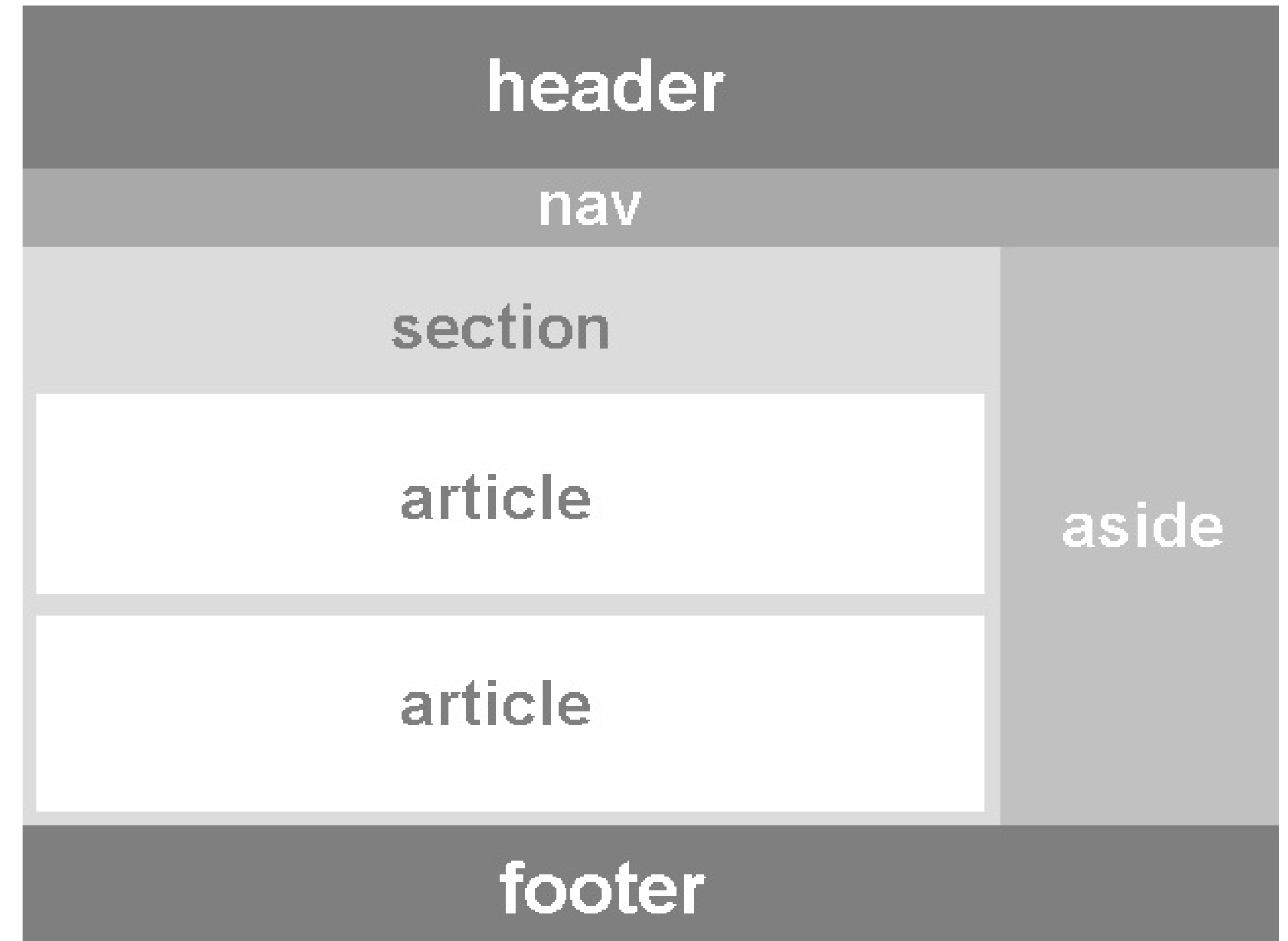


# Za dużo divów

Całą stronę można stworzyć za pomocą tagów **div**.

Czytelność takiej strony jednak znacząco spada, zarówno dla robotów wyszukiwarek, jak i dla osób pracujących nad kodem.

Z pomocą przychodzą nam nowe elementy semantyczne HTML5. Korzystajmy z nich. Zamiast zagnieżdżania w sobie wielu divów o różnych identyfikatorach i klasach.



# Odpowiednie nazwy identyfikatorów i klas

Jeśli nadajesz klasy lub id danym elementom, to twórz zrozumiałe nazwy.

**Źle**

```
<div class="red">Error!</div>
```

**Dobrze**

```
<div class="alert">Error!</div>
```

# Oddzielaj HTML od CSS

Najlepszą praktyką podczas tworzenia stron jest umieszczanie stylów w osobnych plikach.

Staraj się nie mieszać kodu CSS między znacznikami HTML.



# Używaj określonych klas, kiedy to konieczne

Zamiast ustawiania za długich zagnieżdżeń lepiej użyć klasy.

**Źle**

```
div.panel section article footer span {  
    /* CSS */  
}
```

**Dobrze**

```
.info {  
    /* CSS */  
}
```

# Używaj skróconych atrybutów i wartości

Skrócone wersje niektórych atrybutów są łatwiejsze w użytkowaniu.

```
.button {  
  margin-top: 20px;  
  margin-right: 10px;  
  margin-bottom: 2px;  
  margin-left: 2px;  
}
```

```
.button {  
  margin: 20px 10px 2px 2px;  
}
```

# Unikaj wpisywania jednostek przy wartości zero

Nie jest ważne, czy stosujesz piksele, procenty czy inne jednostki.

Jeżeli wartość atrybutu wynosi zero, nie dodawaj niepotrzebnie **px**, **em** itp.

```
.button {  
  margin: 0;  
}
```

# Organizacja kodu

Organizuj kod CSS od góry do dołu. Tak aby był spójny z kolejnością w kodzie HTML

```
section {  
  margin: 0;  
}  
article {  
  margin: 0;  
}  
footer {  
  margin: 0;  
}
```

# **Koncepcje nazewnictwa klas (metodologie)**



# Metodologia BEM

W metodologii BEM mamy trzy typy nazewnictwa klas:

- blok zawierający określony element strony np. komentarz użytkownika ( **.block** ),
- element wewnątrz danego bloku np. avatar lub nazwa użytkownika ( **.block\_\_element** ),
- klasę – modyfikator np. najnowszy komentarz ( **.block--modifier** ).

→ Oficjalna strona metodologii

→ <http://en.bem.info>

```
<div class="comment">
  <span class="comment__user">
    Piotr
  </span>
  <p class="comment__content">
    Komentarz
  </p>
</div>
<div class="comment comment--latest" >
  <span class="comment__user">
    Piotr
  </span>
  <p class="comment__content">
    Komentarz
  </p>
</div>
```

# Inne koncepcje nazewnictwa klas

## → OOCSS

<http://github.com/stubbornella/oocss/wiki>

## → SMACSS

<http://smacss.com>

## → BEMIT

<http://csswizardry.com/2015/08/bemit-taking-the-bem-naming-convention-a-step-further>



CSS nie jest językiem tak łatwym, jak powinien być, co jest mylące zwłaszcza dla początkujących.

J. ZELDMAN