

Podstawy PHP

Dzień III

v.1.2

Plan zajęć

- Zmienne superglobalne
- Przekazywanie informacji między stronami
- Dołączanie zewnętrznych plików z kodem
- Instalowanie bibliotek za pomocą Composer



Zmienne superglobalne

Zmienne superglobalne

- PHP udostępnia zestaw zmiennych predefiniowanych, które są dostępne z każdego miejsca w skrypcie.
- Zajmiemy się zmiennymi z tablicy **\$_SERVER**, o reszcie powiemy w dalszej części kursu przy omawianiu formularzy i przekazywaniu informacji pomiędzy stronami.

Wszystkie zmienne superglobalne znajdują się w tablicach asocjacyjnych:

- **\$_SERVER** – informacje o środowisku i serwerze na jakim wykonywany jest skrypt,
- **\$_GET** – zmienne **HTTP GET**,
- **\$_POST** – zmienne **HTTP POST**,
- **\$_FILES** – zmienne **HTTP FILES** odpowiedzialne za przesyłanie plików,
- **\$_COOKIE** – zmienne odpowiedzialne za pliki ciasteczek,
- **\$_SESSION** – zmienne sesyjne,
- **\$_REQUEST** – połączenie tablic **\$_GET**, **\$_POST** i **\$_COOKIE**
- **\$_ENV** – zmienne środowiskowe.

Zmienne superglobalne

Wybrane przydatne zmienne:

- **\$_SERVER['HTTP_REFERER']** – adres URL strony, z której użytkownik trafił na aktualnie wyświetlaną stronę.
- **\$_SERVER['REMOTE_ADDR']** – adres IP użytkownika.
- **\$_SERVER['REQUEST_URI']** – część aktualnego adresu URL (bez protokołu i domeny).
- **\$_SERVER['HTTP_USER_AGENT']** – informacje o systemie operacyjnym i przeglądarce użytkownika.

Zmienne superglobalne dotyczące środowiska i serwera, na którym wykonywany jest skrypt, można wyświetlić za pomocą polecenia:

```
var_dump($_SERVER);
```

Zmienne superglobalne

Najważniejsza informacja znajdująca się w tablicy **\$_SERVER** to informacją jaką metodą użytkownik wszedł na naszą stronę. Informacja o tym znajduje się w **\$_SERVER['REQUEST_METHOD']**

Na razie powinny was interesować dwie wartości które mogą się znajdować w tej zmiennej:

- **POST** – na stronę weszliśmy przekazując dane typu POST,
- **GET** – na stronę weszliśmy przekazując dane typu GET.


Zwracane dane są typu **string** (napisy)

```
if($_SERVER['REQUEST_METHOD'] === 'POST'){  
    ...  
}
```

Weszliśmy na stronę przesyłając dane POST (formularz).

```
if($_SERVER['REQUEST_METHOD'] === 'GET'){  
    ...  
}
```

Weszliśmy na stronę przesyłając dane GET (formularz lub dane w linku).



**Przekazywanie
informacji
między stronami**

Przekazywanie informacji między stronami

- Skrypty, które do tej pory pisaliśmy, nie potrafiły komunikować się między sobą.
- W PHP dostępnych jest kilka mechanizmów przekazywania informacji między skryptami:
 - dołączanie parametrów do hiperłączy (metoda **GET**),
 - formularze (metoda **GET** i **POST**),
 - sesje (tablica **\$_SESSION**),
 - ciasteczka (tablica **\$_COOKIE**).

Przesyłanie danych – walidacja

Podczas przesyłania danych pomiędzy stronami zawsze będziemy musieli:

- Dowiedzieć się jaką metodą wchodzimy na stronę. Służy do tego zmienna superglobalna **\$_SERVER['REQUEST_METHOD']** (na razie będzie tam napis POST albo GET).
- Czy przesłaliśmy sobie odpowiednie dane. Do tego służy funkcja **isset(\$variableToCheck)**. Funkcja ta przyjmuje jako argument zmienną i zwraca wartość logiczną **TRUE** w przypadku kiedy zmienna jest nastawiona (istnieje i nie jest nastawiona na **NULL**)
- Usunąć z przesłanych danych spacje znajdujące się na początku i na końcu. Uczynimy to dzięki funkcji **trim(\$string)**. Funkcja ta przyjmuje napis i zwraca ten sam napis ale z usuniętymi spacjami z początku i końca.
- Sprawdzić długość przekazanych napisów. To możemy uczynić dzięki funkcji **strlen(\$string)**. Funkcja ta przyjmuje napis i zwraca jego długość.

Przesyłanie za pomocą GET

Najprostszym sposobem przesyłania danych jest przesłanie za pomocą **GET**.

Jest to metoda w której wartości naszych zmiennych są ukryte w adresie url.

Na końcu adresu dodajemy znak ?. Za nim wpisujemy zmienne które chcemy przekazać w formacie:

<nazwa zmiennej>=<wartość zmiennej>

Kolejne zmienne łączymy ze sobą znakiem &

Link który przesyła do strony o nazwie strona2.php

``

Znak rozdzielający przesyłane zmienne

Znak który rozpoczyna informacje o przesyłanych zmiennych

Przesyłamy zmienną foo o wartości 32

Przesyłamy zmienną bar o wartości Some_text

Generowanie linków z danymi GET

Generowanie linków

Generowanie linków które trzymają w sobie dane GET jest bardzo proste. Wystarczy że w atrybut **href** naszego tagu **<a>** dodamy wartości które przechowujemy w jakiejś zmiennej.

Możemy zrobić to za pomocą np. pętli i tablicy:

showNamesLink.php:

```
<?php
```

```
$namesArray = array('Jan', 'Adam', 'Piotr');
```

```
foreach($namesArray as $name) {
```

```
    echo("<a href='getNames.php?userName=$name'>Link do strony $name</a>");
```

```
}
```

```
?>
```

Pętla która przejdzie nam po wszystkich elementach tablicy **\$namesArray**

Wyświetlamy na ekranie link który poprowadzi nas do innej strony przesyłając jej dane **GET**

Odbieranie parametrów GET

Odbieranie danych GET

Wszystkie dane które przesyłamy metodą GET znajdują się w zmiennej superglobalnej `$_GET`.
Wartość zmiennej znajduje się pod takim samym kluczem jak nazwa która znajduje się w adresie.

Żeby poprawnie odebrać dane przesłane metodą **GET** należy zawsze:

- Sprawdzić jaką metodą użytkownik wszedł na stronę,
- Sprawdzić czy zmienne o podanej nazwie zostały przesłane,
- Odczytać odpowiednie komórki z zmiennej superglobalnej **`$_GET`**

Odbieranie parametrów GET

Odbieranie danych GET

getNames.php

<?php

```
if($_SERVER['REQUEST_METHOD'] === 'GET'){
```

```
    if(isset($_GET['name']) === true){
```

```
        echo('Witaj na stronie 2. Metodą Get przesłałeś imię:' . $_GET['name']);
```

```
    }
```

```
    else{
```

```
        echo('Witaj na stronie 2. Metodą Get nie zostały przekazane dane pod kluczem name');
```

```
    }
```

```
}
```

?>

Sprawdzamy czy weszliśmy na stronę metodą GET

Sprawdzamy czy zostały przekazane odpowiednie informacje

Używamy przesłanych informacji

Czas na zadania

- Przeróbcie ćwiczenia z części A z katalogu „Przesyłanie danych”.

Formularze

- Formularze dają większe możliwości niż **GET** w przekazywaniu informacji między stronami.
 - Element **HTML** odpowiedzialny za tworzenie formularza (**<form>**) oraz elementy tworzące poszczególne jego pola (**<input>**, **<select>**, **<textarea>** itd.) już poznaliśmy na kursie.
 - Formularze mogą przekazywać dane metodą **GET** lub **POST**.
- Metoda **GET** przekazuje wartości pól formularza za pomocą adresu **URL** – są one widoczne w pasku adresu przeglądarki (jest to mniej bezpieczna opcja).
 - Metoda **POST** do przekazywania parametrów wykorzystuje nagłówek zapytania, jej parametry nie są widoczne w adresie **URL**. Jest to bezpieczniejsza i częściej używana opcja do przekazywania danych wysłanych formularzem.

Formularze - tworzenie

- Teraz zajmiemy się odbieraniem elementów i sprawdzaniem wartości, które użytkownik wpisał do formularza.
- Adres skryptu, do którego mają być wysłane dane z formularza, definiujemy w atrybucie **action**.
- Metodę wysłania parametrów definiujemy w atrybucie **method**.

```
<form action="form.php" method="POST">  
...  
</form>
```

Do jakiej strony
prześle nas formularz

Jaką metodą zostaną
przesłane dane (mamy do
wyboru **POST** albo **GET**)

Formularze – pola

Następnie musimy wpisać do formularza pola w które użytkownik będzie mógł wpisywać dane.

Najważniejszy z poziomu back endu jest atrybut **name** który musi się znaleźć w każdym z inputów.

Określa on pod jakim kluczem będzie znajdowała się wartość z danego inputa po przesłaniu danych.

```
<label>  
  Imię:  
  <input type="text" name="userName">  
</label>
```

```
<label>  
  Nazwisko:  
  <input type="text" name="userSurname">  
</label>
```

Ważne jest żeby nigdy nie zapomnieć o atrybucie **name**

Formularze – pole submit

Na końcu formularze musi znajdować się specjalny **input** o typie **submit**. Dzięki niemu użytkownik będzie mógł wysłać przygotowany formularz.

```
<label>  
  <input type="submit">Wyślij</input>  
</label>
```

Odbieranie parametrów POST

Odbieranie danych POST

Wszystkie dane które przesyłamy metodą POST znajdują się w zmiennej superglobalnej `$_POST`. Wartość zmiennej znajduje się pod takim samym kluczem jak atrybut **name** danego pola formularza.

Jeżeli dwa pola będą miały taką samą wartość atrybutu **name**, to będziemy w stanie odebrać dane tylko z jednego z nich.

Żeby poprawnie odebrać dane przesłane metodą **POST** należy zawsze:

- Sprawdzić jaką metodą użytkownik wszedł na stronę,
- Sprawdzić czy zmienne o podanej nazwie zostały przesłane,
- Odczytać odpowiednie komórki z zmiennej superglobalnej **\$_POST**, usunąć im białe znaki z początku i końca i ewentualnie sprawdzić ich długość

Odbieranie parametrów POST

Odbieranie danych POST

getNames.php

<?php

```
if($_SERVER['REQUEST_METHOD'] === 'POST'){  
    $name = "";  
    $surname = "";  
  
    if(isset($_POST['name']) === true && strlen(trim($_POST['name'])) > 5){  
        $name = trim($_POST['name']);  
    }  
    else{  
        echo( 'Brak przekazanych danych albo złe dane (imię musi mieć co najmniej 5 znaków)');  
    }  
}
```

?>

Sprawdzamy czy weszliśmy na stronę metodą POST

Sprawdzamy czy zostały przekazane odpowiednie informacje

Używamy przesłanych informacji

Formularze – pola typu select

Jeżeli chcemy stworzyć na stronie a potem obsłużyć pole typu **select** powinniśmy dodać atrybut **name** tylko do tagu **<select>**.

Następnie atrybut **value** dodajemy do wszystkich tagów **option** które są zagnieżdżone w naszym tagu **select**.

```
<label>
  <select name="gender">
    <option value="">Select...</option>
    <option value="M">Male</option>
    <option value="F">Female</option>
  </select>
</label>
```

Wartości pojawią się w
superglobalnej **\$_POST**
pod podanym tutaj
kluczem

Te wartości pojawią
się w superglobalnej
\$_POST

Formularze – pola typu select

Jeżeli chcemy dać możliwość wybrania kilku opcji w naszym selekcie to musimy do nazwy klucza którą wpisujemy w tagu **name** dodać **[]**.

Dzięki temu PHP będzie wiedziało że pod tym kluczem będzie znajdować się tablica z wszystkimi wybranymi wartościami.

Wartość znajdująca się pod kluczem zawsze będzie tablicą, nawet jeżeli zostanie zaznaczona tylko jedna odpowiedź.

```
<label>
  <select name="colors[]" multiple>
    <option value="">Select...</option>
    <option value="red">Czerwony</option>
    <option value="blue">Niebieski</option>
    <option value="green">Zielony</option>
    <option value="yellow">żółty</option>
  </select>
</label>
```

Dodatkowe nawiasy na końcu zmiennej oznaczają że dane będą w formie tablicy

Te wartości pojawią się w superglobalnej **\$_POST** jako wartości w tablicy

Formularze – pola typu checkbox

Podobnie musimy poradzić jeżeli chcemy w formularzu mieć kilka pól typu **checkbox** które będą ze sobą logicznie powiązane.

Należy wszystkim takim polom nadać taką samą nazwę (nie zapominając o nawiasach na końcu).

Dzięki temu wszystkie zaznaczone opcje będą w tablicy pod podanym kluczem.

Wszystkie pola mają takie same wartości w atrybucie **name**

<label>

Wybierz dodatki:

<input type="checkbox" name="topings[]" value="cheese">Dodatkowy ser

<input type="checkbox" name="topings[]" value="ham">Dodatkowa szynka

<input type="checkbox" name="topings[]" value="pineapple">Ananas

<input type="checkbox" name="topings[]" value="tomato">Pomidor

<input type="checkbox" name="topings[]" value="salami">Salami

</label>

Czas na zadania

- Przeróbcie ćwiczenia z części B z katalogu „Przesyłanie danych”.

Sesje

- Umożliwiają przekazywanie danych między stronami w łatwy sposób.
 - Zmienne są przechowywane po stronie serwera, u klienta trzymane jest tylko ID sesji.
 - ID jest zapisane w ciasteczku lub przekazywane przez URL.
 - Podobnie jak w przypadku ciasteczek, aby używać zmiennych sesyjnych przed rozpoczęciem sesji, do przeglądarki nie mogą być wysłane żadne inne dane.
- Po rozpoczęciu sesji funkcją **session_start()** PHP sprawdza, czy przypisano już ID sesji.
 - Jeśli tak, to odczytuje zmienne zarejestrowane w tej sesji.
 - Jeśli nie, generowany jest nowy, unikalny identyfikator sesji.

Sesje – zapis i odczyt zmiennych

- Zmienne sesyjne są przechowywane w tablicy superglobalnej **\$_SESSION**.
- Zmienne sesyjne rejestrujemy przez odwołanie się do tablicy **\$_SESSION** tak jak do zwykłej tablicy.
- Możemy usunąć zmienną sesyjną za pomocą funkcji **unset()**.
- Sesja kończy się w kilku przypadkach: zamknięciu przeglądarki, upływie maksymalnego czasu określanego przez serwer, albo braku komunikacji klient-serwer przez dłuższy czas.

`<?php`

`session_start();`

```
if(!isset($_SESSION['licznik'])) {  
    $_SESSION['licznik'] = 1;  
} else {  
    $_SESSION['licznik']++;  
}
```

`echo('liczba odświeżeń: ' . $_SESSION['licznik']);`
`?>`

Jeżeli korzystamy z sesji to zawsze musimy ją na początku uruchomić

Z sesji korzystamy jak z normalnej tablicy

Czas na zadania

- Przeróbcie ćwiczenia z części C z katalogu „Przesyłanie danych”.

Ciasteczka

Ciasteczka są to pliki przechowujące niewielkie ilości danych na komputerze oglądającego stronę.

Ciasteczka ustawione przez daną stronę dostępne są tylko dla niej i przez określony czas.

Umożliwiają przechowywanie informacji nawet po zamknięciu przeglądarki.

Ciasteczka przekazywane są za pomocą nagłówków HTTP. W związku z tym musimy zagwarantować że zostaną one wysłane zanim na stronie pojawi się jakakolwiek treść:

- Przed zapisaniem ciasteczka nie może być żadnego wywołania funkcji **echo** i pochodnych.
- Element otwierający tryb PHP musi być pierwszym znakiem w pliku – nie może być żadnych białych znaków (spacji, enterów).

Timestamp

Timestamp jest specjalnym sposobem zapisu czasu w informatyce.

Jest on zapisany jako ilość sekund od 1 stycznia 1970 roku.

Jest bardzo często używany do obliczania różnic w czasie (np. ile czasu dokładnie upłynie pomiędzy dwoma wydarzeniami).

W PHP mamy dwie funkcje zwracające **timestamp**:

- **time()** – zwraca ilość sekund od 1 stycznia 1970 roku.
- **microtime()** – zwraca ilość milisekund od 1 stycznia 1970 roku. Nie działa na wszystkich systemach.

Ciasteczka – tworzenie

Funkcja **setcookie()** służy do tworzenia i zapisywania ciasteczek. Jako parametry przyjmuje:

- | | |
|--|---|
| <ul style="list-style-type: none">1. nazwę ciasteczka,2. jego wartość,3. czas wygaśnięcia (w formacie timestamp),4. ścieżkę, z której dostępne będzie ciasteczko (domyślnie ścieżka do skryptu, w którym ciasteczko zostało utworzone),5. domenę, z której dostępne będzie ciasteczko (domyślnie domena, pod którą znajduje się skrypt, w którym ciasteczko zostało utworzone), | <ul style="list-style-type: none">6. sposób przesyłania – czy ciasteczko ma być przesyłane szyfrowanym połączeniem HTTPS,7. tylko HTTP – określa, czy ciasteczko ma być dostępne tylko przez protokół HTTP, <p>Wymagany jest tylko pierwszy parametr (nazwa).</p> |
|--|---|

Jako że ciasteczka nie są trzymane na serwerze, tylko na komputerze klienta, pamiętajcie że będą one widoczne dopiero po odświeżeniu strony!

Ciasteczka – tworzenie

Jako że ciasteczka nie są trzymane na serwerze, tylko na komputerze klienta, pamiętajcie że będą one widoczne dopiero po odświeżeniu strony!

Ciasteczka – czas wygaśnięcia

- Czas, po jakim ciasteczko zostanie usunięte, musi być podany w formacie **timestamp**.
- Jeśli czas wygaśnięcia będzie wcześniejszy niż aktualny czas, ciasteczko zostanie usunięte.
- Jeśli czas wygaśnięcia będzie równy 0 (wartość domyślna), to ciasteczko będzie ważne tylko do zamknięcia przeglądarki.
- Do otrzymania dokładnego czasu w którym nasz skrypt został wywołany używamy funkcji **time()**

```
<?php  
setcookie('jezyk', 'en');
```

Ciasteczko wygaśnie za godzinę (3600 sekund)

```
setcookie('jezyk', 'en', time() + 3600);
```

Ciasteczko wygaśnie za dzień (24 godziny)

```
setcookie('jezyk', 'en', time() + 3600 * 24);  
?>
```


Ciasteczka – odczyt

- PHP automatycznie odczytuje ciasteczka i zamienia je na zmienne.
- Są one przechowywane w superglobalnej tablicy asocjacyjnej **\$_COOKIE**.
- Kluczami w tablicy **\$_COOKIE** są nazwy ciasteczek.

```
<?php
if( !isset($_COOKIE['jezyk']) ) {
    setcookie('jezyk', 'en', time() + 3600 * 24);
    echo('ciasteczko utworzone');
```

Jeżeli nie ma takiego ciasteczka to je utwórz

```
    }
    else {
        var_dump($_COOKIE['jezyk']);
    }
?>
```

Jeżeli istnieje takie ciasteczko to wyświetl jego wartość

Ciasteczka – usuwanie

Aby usunąć ciasteczko należy użyć funkcji **setcookie()**, podając czas wygaśnięcia wcześniejszy niż aktualny.

```
<?php
```

```
if(isset($_COOKIE['jezyk'])) {  
    setcookie('jezyk', 'en', time() - 3600);  
    echo('ciasteczko język usunięte<br>');
```

```
}  
?>
```

Jeżeli istnieje ciasteczko to je
usuń (nastaw czas życia na
wartość która już minęła)

Ciasteczka – przechowywanie tablic

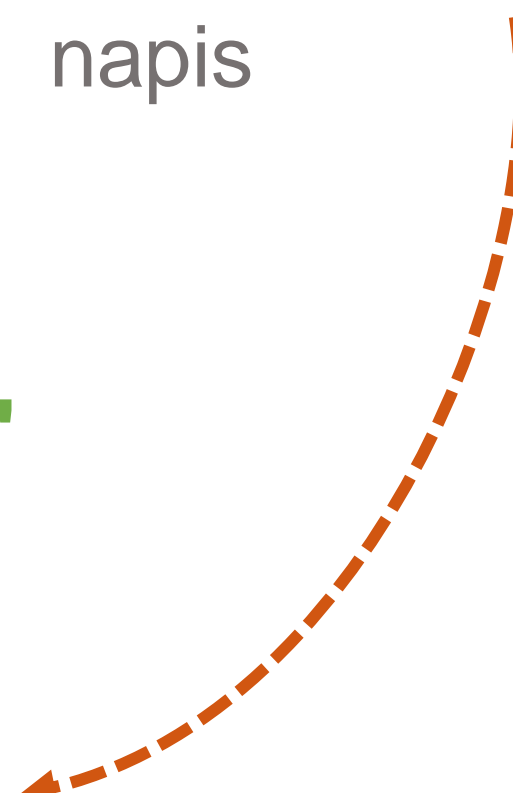
- Jeżeli chcemy w ciasteczku przechowywać tablicę, należy zamienić ją na ciąg znaków.
- Służy do tego funkcja **serialize()**.
- Odwrotnego procesu dokonuje funkcja **unserialize()**.

```
<?php
    $klient = array(
        'imię'    => 'Łukasz',
        'nazwisko' => 'Pokrzywa',
        'miasto'   => 'Warszawa'
    );

    if(!isset($_COOKIE['klient'])) {
        setcookie('klient', serialize($klient), time() + 3600 * 24);
        echo('ciasteczko utworzone');
    }

    if(isset($_COOKIE['klient'])) {
        var_dump($_COOKIE['klient']);
        var_dump(unserialize($_COOKIE['klient']));
    }
?>
```

Serializujemy dane –
zamieniamy w specjalny
napis



Odczytując dane musimy je
zamienić w tablicę



Czas na zadania

- Przeróbcie ćwiczenia z części D z katalogu „Przesyłanie danych”.

Dołączanie zewnętrznych plików z kodem

Dołączanie zewnętrznych plików z kodem

- Dobrą praktyką w każdym języku programowania jest zachowywanie modułowej struktury programu.
 - W uproszczonym przypadku taką modułowość możemy osiągnąć dzięki rozbiciu kodu programu na kilka plików.
 - W jednym pliku możemy trzymać np. funkcje, które przydadzą się nam w programie.
 - Plik taki dołączamy następnie do każdego pliku, w którym chcemy użyć funkcji – nie musimy definiować wtedy tych samych funkcji w każdym pliku z osobna.
- Dzięki takiemu rozwiązaniu unikniemy poprawiania definicji funkcji w każdym pliku osobno.
 - Wystarczy zmiana w jednym pliku, który dołączamy do wszystkich skryptów korzystających z funkcji.

include(), require()

W PHP istnieją dwie funkcje pozwalające dołączyć do kodu zawartość innego pliku także zawierającego kod PHP: **include()** i **require()**. Obie funkcje przyjmują jeden parametr – nazwę pliku, który ma być dołączony do kodu, może to być sama nazwa lub ścieżka do pliku.

Funkcja **require()** dołącza plik do kodu jeszcze przed parsowaniem, w każdym miejscu, gdzie znajduje się instrukcja **require**.

Z tego powodu nie nadaje się do dołączania plików, których nazwa pobierana jest ze zmiennej.

Jeśli użyliśmy instrukcji **require()**, ale okazało się, że dołączany plik nie istnieje, wtedy cały skrypt, do którego dołączamy plik, nie będzie wykonany.

Funkcja **include()** dołącza plik do kodu w trakcie działania naszego programu.

Z tego powodu można dynamicznie dołączać pliki (których nazwa jest trzymana w zmiennej).

Jeśli użyliśmy instrukcji **include()**, ale okazało się, że dołączany plik nie istnieje, wtedy skrypt będzie wykonywany dalej, ale wyświetli się nam ostrzeżenie..

require()

funkcje.php

```
<?php
function ObliczKosztDostawy($masa) {
    if($masa <= 5)
        return 8;
    else
        return $masa * 1.8;
}
?>
```

index.php

```
<?php
require('funkcje.php');

$masaProduktu = 16;
echo("koszt dostawy produktu o masie
    $masaProduktu kg wynosi ");
echo(ObliczKosztDostawy($masaProduktu) .
    " zł");
?>
```


include()

funkcje.php

```
<?php
function ObliczKosztDostawy($masa) {
    if($masa <= 5)
        return 8;
    else
        return $masa * 1.8;
}
?>
```

index.php

```
<?php
$fileName = 'funkcje.php';
include($fileName);

$masaProduktu = 16;
echo("Koszt dostawy produktu o masie
    $masaProduktu kg wynosi ");
echo(ObliczKosztDostawy($masaProduktu) .
    " zł ");
?>
```

include(), require() include once(), require once()

Obie funkcje dołączające kod występują też w dwóch wersjach. Z końcówką **_once** albo bez niej.

Funkcje z końcówką **_once** dołączają plik tylko raz. Jeżeli w naszym kodzie będzie próba dołączenia tego pliku jeszcze raz to nie zostanie on dołączony.

Przydatne jest to w chwili kiedy chcemy użyć pliku w którym mamy jakieś funkcje albo klasy (redefinicja funkcji albo klasy to błąd).

Funkcje bez końcówki **_once** dołączają plik za każdym razem kiedy zostaną wywołane.

Przydatne jest to w chwili w której mamy wyodrębniony sposób wyświetlania czegoś w osobnym pliku.

KONIEC