

Skriptni jezik PHP

Uvod v PHP



- *“PHP je skriptni jezik za strežnike, načrtovan za splet. PHP kodo vgradimo v HTML stran. Izvajana bo ob vsakem obisku te strani. Interpretira jo spletni strežnik in tvori HTML ali kakšen drugačen izpis, ki ga bo obiskovalec lahko videl.*

Kako naj se naučim PHP?



- Začni s preprostimi zgledi, namestitvijo skripta, ki naj izpiše stran HTML
- Razumi, kako PHP podpira interakcijo z brkljalnikom ali drugimi odjemalci
- Razumi, kako PHP podpira integracijo s podatkovnimi bazami (MySQL)
- Razumi, kako PHP podpira integracijo z drugimi aplikacijami . spletnimi storitvami

Alternative PHP



- Practical extraction and Report Language (Perl)
- Active Server Pages (ASP)
- Java server pages (JSP)
- Ruby

Kako deluje spletni strežnik?



- Spletni strežnik je program, ki nepretrgoma teče na računalniku – strežniku, povezanem na internet
- Opazuje zahteve za datoteke, ki jih pošilja odjemalec
- Na spletnem brkljalniku se kot spletna stran pojavijo datoteke v formatu HTML ali podobnem
- Datoteke v drugih formatih (na primer PDF) prikazujejo spletni brkljalniki v skladu s konfiguracijo odjemalčevega računalnika

Vsebina: Statična ali dinamično tvorjena



- Statične HTML datoteke: strežnik jih posreduje brez sprememb
 1. datoteke zahteva odjemalec
 2. Datoteke posreduje strežnik
- Dinamično tvorjene HTML datoteke: intervencija strežnika
 1. Odjemalec zahteva “datoteko”
 2. Strežnik tvori ali spremeni HTML izhod
 3. Strežnik posreduje “datoteko”

Metode tvorbe dinamične vsebine



- CGI – Common Gateway Interface
 - Teče zunanji program (Perl, C, shell script)
 - Programi lahko tečejo le v dogovorjenih direktorijih
 - Izhod usmerja spletni strežnik
- Razširitve spletnega strežnika
 - ASP
 - PHP
 - Drugi : ColdFusion, FrontPage razširitve
 - Programi lahko tečejo na katerikoli lokaciji

ASP (Active Server Pages)



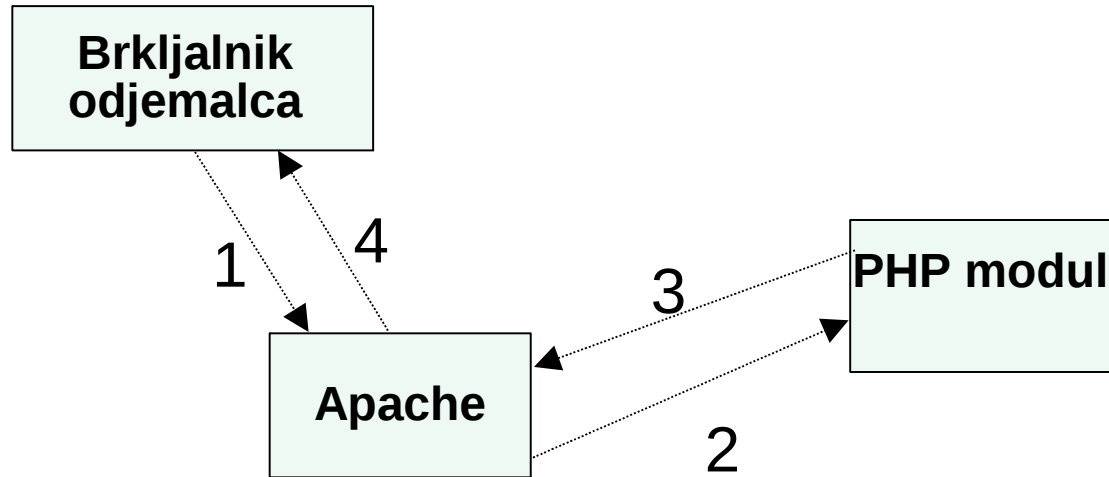
- Omejeni na strežnike MS Windows, na katerih teče IIS
- Visual Basic script
- Enostavno povezovanje z datotekami v MS formatih, kot so Word, Excel, Access
- V nekaterih primerih najlažji način
- Ni zastoj

PHP (PHP Hypertext Preprocessor)



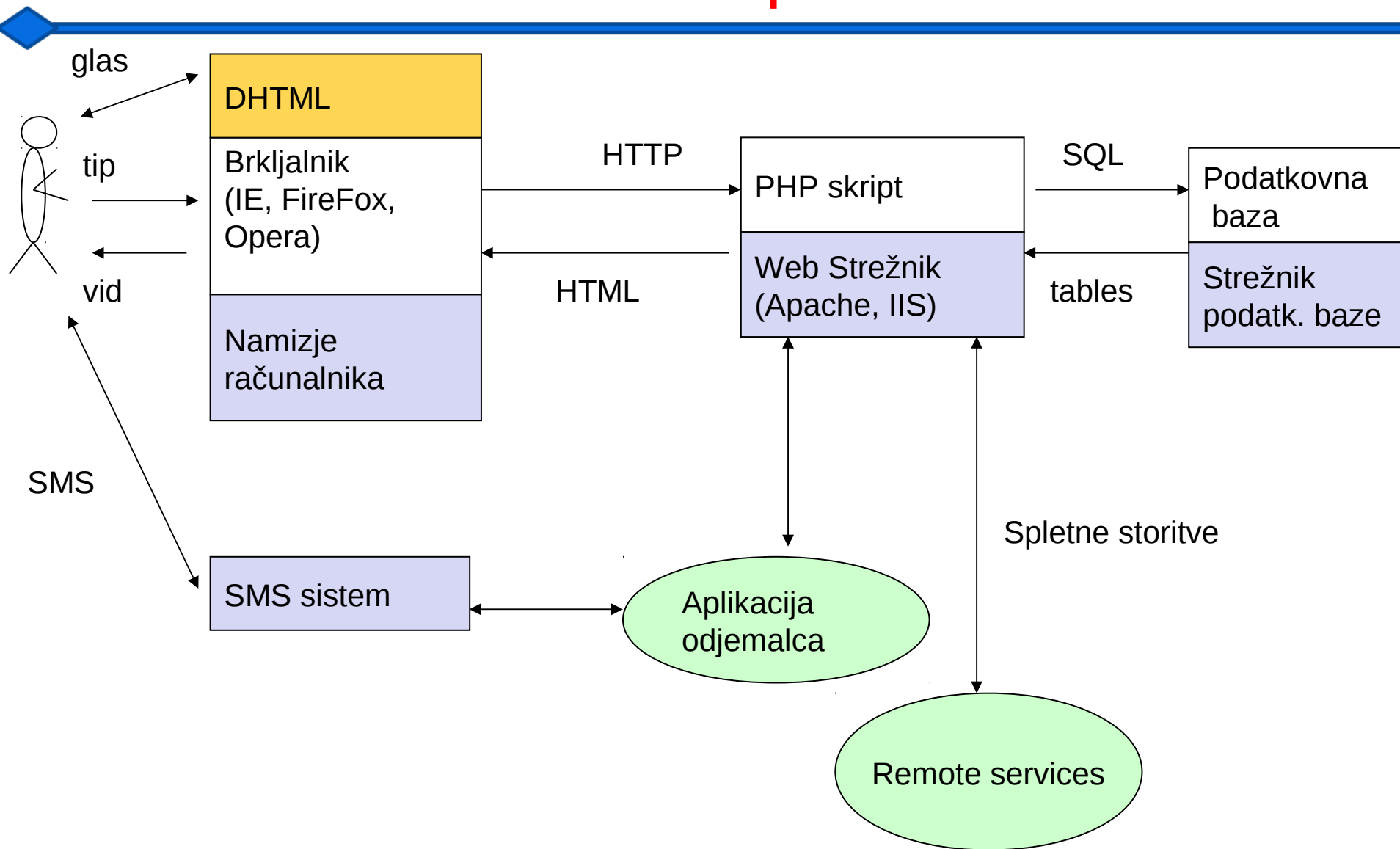
- Odprtokodna tehnologija
- Enostavnost uporabe (sintaksa podobna C in Perl)
- Stabilnost in hitrost
- Različne platforme
- Podpora številnim podatkovnim bazam
- Veliko skupnih vgrajenih knjižnic
- Vnaprejšnja namestitvev pri Linuxu

Kako PHP tvori HTML/JS spletne strani



- 1: Odjemalec pošlje iz brkljalnika HTTP zahtevek (s spremenljivkami POST/GET)
- 2: Apache spozna, zahtevo PHP skripta in pošlje to zahtevo modulu PHP
- 3: PHP interpreter izvede PHP skript, zbere izhod skripta in ga vrne
- 4: Apache odgovori odjemalcu tako, da izhod PHP skripta uporabi kot HTML izhod

Imamo lahko tudi večplastno arhitekturo

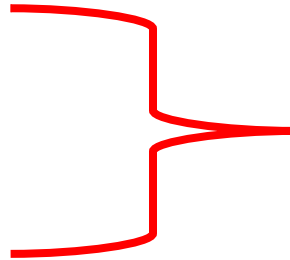


PHP je podoben C

- Prost format- presledke ignoriramo
- Stavke zaključujemo s podpičjem;
- Stavke združujemo z { ... }
- Komentarje začenjamo z // ali uporabimo /* */
- prirejanje vrednosti '=': \$a=6
- Relacijski operatorji so , < , > == (dva znaka =)
- Krmilni stavki oblike if (cond) {..} else { }, while (cond) { .. } , for(sstartcond; increment; endcond) { }
- Polja dostopamo z [] : \$x[4] je peti element polja \$x –indeksi začenjajo z 0
- Asociativna polja (hash polje v Perl, slovar v Javi) dostopamo na enak način \$y[“fred”]
- Funkcije kličemo po imenu, kateremu slede med oklepajema argumenti v fiksnem vrstnem redu : substr(“fred”,0,2)
- Razlikovanje velikih in malih črk- \$fred je druga spremenljivka kot \$FRED

Pozdravljen svet! (na spletu)

```
<html>
<head>
  <title> Pozdrav svetu! PHP skript</title>
</head>
<body>
  <?
  echo "Hello World!";
  ?>
</html>
```



PHP tag, omogoča vstavek s PHP kodo. To bo tolmačil modul PHP in kodo nadomestil z izhodom te kode

Še en kratek primer

- `<html>`
`<head>Primer z izpisom casa</head>`
`<body>`
Cas je sedaj
`<?php` *tu "skočimo v PHP"*
`echo date();`
`?>` *tu "skočimo spet ven"*
`<hr>`
`</body>`
`</html>`

O funkcijah

Uporabnik definira funkcijo na naslednji način:

```
function function_name([seznam_parametrov]opt)
```

```
{.....koda implementacije.....}
```

Seznam parametrov je zaporedje spremenljivk, ločenih z “,”

- Ne smemo prekriti imena že obstoječe funkcije;
- Imena funkcij niso občutljiva na velike in male črke;
- Vsakemu parametru moramo prirediti privzeto (default) vrednost;
- argumente lahko posredujemo po vrednosti ali referenci
- Uporabljamo lahko spremenljivo število parametrov

.

Knjižnica funkcij

- Osnovne naloge
 - Rokovanje z nizi
 - Matematične funkcije
 - Regularni izrazi
 - Rokovanje z datumom in časom
 - Branje in pisanje datotek
- Bolj posebne funkcije za:
 - Interakcijo s podatkovnimi bazami–
 - MySQL, Oracle, Postgres, Sybase, MSSQL ..
 - Enkripcijo
 - Prevajanje besedil
 - Tvorbo slik
 - XML

Še en malo bolj zapleten primer

```
<?php
include "utilities.php";
?>
<html>
<head>Test page</head>
<body>
<?php
if ($utils->isFriendly()) {
    echo "Cas je sedaj" . date();
} else {
    echo "Ne povem ti trenutnega casa";
}
?>
<hr>
</body>
</html>
```

Spremenljivke (1)

- Za uporabo ali prireditve neke spremenljivke mora biti pred njenim imenom znak \$
- Prireditveni operator je '='
- Tipa spremenljivk ni treba deklarirati
- Trenutno shranjena vrednost implicitno tvori kasto tipa spremenljivke.
- Spremenljivko lahko uporabimo, preden ji kaj priredimo

```
$A = 1;
```

```
$B = "2";
```

```
$C = ($A + $B); // celostevilčna vsota
```

```
$D = $A . $B; // verizenje nizov
```

```
echo $C; // prints 3
```

```
echo $D; // prints 12
```

Spremenljivke (2)

- Funkcija **isset** preverja, ali je spremenljivki že prirejena neka vrednost

```
$A = 1;
```

```
if (isset($A))
```

```
    print "A is set"
```

```
if (!isset($B))
```

```
    print "B is NOT set";
```

- Uporaba \$\$

```
$help = "hiddenVar";
```

```
$$help = "hidden Value";
```

```
echo $$help; // prints hidden Value
```

```
$$help = 10;
```

```
$help = $$help * $$help;
```

```
echo $help; // print 100
```

Nizi (1)

- *Niz je zaporedje znakov*

```
$stringTest = "this is a sequence of chars";
```

```
echo $stringTest[0]; output: t
```

```
echo $stringTest; output: this is a sequence of chars
```

- *Nize med enojnimi navednicami prikazujemo "take-kot so"*

```
$age = 37;
```

```
$stringTest = 'I am $age years old'; // output: I am $age years old
```

```
$stringTest = "I am $age years old"; // output: I am 37 years old
```

- *Veriženje (Concatenation)*

```
$conc = "is "."a "."composed "."string";
```

```
echo $conc; // output: is a composed string
```

```
$newConc = 'Also $conc '.$conc;
```

```
echo $newConc; // output: Also $conc is a composed string
```

Nizi (2)

- Funkcija **explode**

```
$sequence = "A,B,C,D,E,F,G";
```

```
$elements = explode(",", $sequence);
```

```
// Now elements is an array with all substrings between "," char
```

```
echo $elements[0]; // output: A;
```

```
echo $elements[1]; // output: B;
```

```
echo $elements[2]; // output: C;
```

```
echo $elements[3]; // output: D;
```

```
echo $elements[4]; // output: E;
```

```
echo $elements[5]; // output: F;
```

```
echo $elements[6]; // output: G;
```

Funkcija explode razbije niz na manje podnize

Rokovanje z nizi

- Konstante nizov (literali) so vključeni v dvojne ali enojne narekovaje
- Med "" spremenljivke nadomestimo z njihovimi vrednostmi (interpolacija spremenljivk): "Moje ime je \$name, če imam prav"
- Med enojnimi narekovaji do interpolacije ne pride
- Nize povezujemo (verižimo) z operatorjem pika: "key"."board" == "keyboard"
- Imamo več standardnih funkcij, na pr: strlen(), substr() itd
- Vrednosti lahko pretvorimo v nize in obratno. Pretvorba števil v niz je v kontekstu niza implicitna.

.

Polja (1)

Združevanje skupine spremenljivk. Vsak element je pomnjen v polju s prirejenim ključem (indeksom)

```
$books = array( 0=>"php manual",1=>"perl manual",2=>"C manual");
```

```
$books = array( 0=>"php manual","perl manual","C manual");
```

```
$books = array ("php manual","perl manual","C manual");
```

```
echo $books[2]; //output: C manual
```

Polja v PHP so asociativna

```
$books = array( "php manual"=>1,"perl manual"=>1,"C manual"=>1); // HASH
```

```
echo $books["perl manual"]; //output: 1
```

```
$books["lisp manual"] = 1; // Add a new element
```

Polja (2)

- Delo s polji

```
$books = array( "php manual", "perl manual", "C manual");
```

- Zanke

```
for ($i=0; $i < count($books); $i++)
```

```
    print ($i+1)."-st book of my library: $books[$i]";
```

- each

```
$books = array( "php manual"=>1, "perl manual"=>2, "C manual"=>3);
```

```
while ($item = each( $books )) // Retrieve items one by one
```

```
    print $item["value"]."-st book of my library: ".$item["key"];
```

```
// each retrieve an array of two elements with key and value of current element
```

- each in list

```
while ( list($value,$key) = each( $books ))
```

```
    print "$value-st book of my library: $key";
```

```
// list collect the two element retrieved by each and store them in two different // variables
```


Večdimenzijska polja

```
$books = array( array("title"=>"php manual","editor"=>"X","author"=>"A"),  
                array("title"=>"perl manual","editor"=>"Y","author"=>"B"),  
                array("title"=>"C manual","editor"=>"Z",author=>"C"));
```

Zanka

```
for ($i=0; $i < count($books); $i++ )  
    print "$i-st book, title: ".$books[$i]["title"]." author: ".$books[$i]["author"].  
        " editor: ".$books[$i]["editor"];
```

```
// Add ".\n" for text new page or "<BR>" for HTML new page;
```

Uporaba list in each

```
for ($i=0; $i < count($books); $i++)  
{  
    print "$i-st book is: ";  
    while ( list($key,$value) = each( $books[$i] ))  
        print "$key: $value ";  
    print "<BR>"; // or "\n"  
}
```

Študijski primer: majhna podatkovna baza(1)

Potrebujemo eno ali več spletnih strani za upravljanje z našo knjižnico, vendar:

– *“Nimamo časa in znanja o načrtovanju podatkovnih baz”*

ali

– *“Ne vemo, kako namestiti prosto dostopno podatkovno bazo”*

in

– *“Podatkovna baza, ki jo želimo narediti, je majhna” (največ nekaj tisoč elementov)*

Študijski primer: majhna podatkovna baza(2)

Imejmo datoteko *library.txt* v direktoriju */usr/local/myDatabaseDirectory/*

```
php manual X A 330
perl manual Y B 540
C manual Z C 480
```

(polja so ločena s tabulatorji (<tab>). Na koncu vsake vrstice je new line)

```
<? // script to show all book in my library
```

```
$books = file("/usr/local/myDatabaseDirectory/library.txt"); // retrieve library "database"
```

```
for ($i=0; $i<count($books), $i++ )
```

```
    $books_array[$i] = explode( "\t", $books[$i]); // Extract elements from line
```

```
...
```

```
for ($i=0; $i<count($books_array), $i++ )
```

```
    print "$i-st book, title: ".$books_array[$i]["title"]." author: ".$books_array[$i]["author"].
```

```
        " editor: ".$books_array[$i]["editor"]." <BR>";
```

Študijski primer: ponovna uporaba kode (1)

Ponovno uporabo kode omogoča uporaba funkcij:

```
<? // config.php, is a good idea to use configuration files
$tableFiles = array ( "books"=>"/usr/local/myDatabaseDirectory/books.txt");
$bookTableFields = array ("title","author","editor","pages");
// future development of the library project (add new tables)
$tableFiles = array ( "users"=>"/usr/local/myDatabaseDirectory/users.txt");
$userTableFields = array ("code","firstName","lastName","age","institute");
?>
```

Študijski primer: ponovna uporaba kode (2)

```
<? // script to show all book in my library
$books = file("/usr/local/myDatabaseDirectory/library.txt");
// retrieve library "database"
for ($i=0; $i<count($books), $i++ )
    $books_array[$i] = explode( "\t", $books[$i]); // Extract elements from line
...
for ($i=0; $i<count($books_array), $i++ )
    print "$i-st book, title: ".$books_array[$i]["title"]." author: ".$books_array[$i]["author"].
        " editor: ".$books_array[$i]["editor"]." <BR>";
```

Objektno usmerjen PHP



- Enkapsulacija
- Polimorfizem
- Dedovanje
- Večkratno dedovanje: ni podprto

Enkapsulacija



<?

```
class dayOfWeek {
```

```
    var $day,$month,$year;
```

```
    function dayOfWeek($day,$month,$year) {
```

```
        $this->day = $day;
```

```
        $this->month = $month;
```

```
        $this->year = $year;
```

```
    }
```

```
    function calculate(){
```

```
        if ($this->month==1){
```

```
            $monthTmp=13;
```

```
            $yearTmp = $this->year - 1;
```

```
        }
```

```
        if ($this->month == 2){
```

```
            $monthTmp = 14;
```

```
            $yearTmp = $this->year - 1;
```

```
        }
```

```
        $val4 = (($month+1)*3)/5;
```

```
        $val5 = $year/4;
```

```
        $val6 = $year/100;
```

```
        $val7 = $year/400;
```

```
        $val8 = $day+($month*2)+$val4+$val3+$val5-$val6+$val7+2;
```

```
        $val9 = $val8/7;
```

```
        $val0 = $val8-($val9*7);
```

```
        return $val0;
```

```
    }
```

```
}
```

```
// Main
```

```
$instance =
```

```
    new dayOfWeek($_GET["day"],$_GET["week"],$_GET[" month"]);
```

```
print "You born on ".$instance->calculate()."\n";
```

```
?>
```

Dedovanje

Omogoča tvorbo hierarhije razredov

```
Class reuseMe {  
  
    function  
        reuseMe(){...}  
  
    function  
        doTask1(){...}  
  
    function  
        doTask2(){...}  
  
    function  
        doTask3(){...}  
}
```

```
Class extends reuseMe {  
  
    function example(){  
        ... // local  
        initializations  
        // call super  
        constructor  
        reuseMe::reuseMe();  
    }  
  
    function doTask4(){...}  
  
    function doTask5(){...}  
  
    function doTask6(){...}  
}
```


Polimorfizem

Članska funkcija lahko prekrije implementacijo nadrazreda. Tako lahko v podrazredih spremenimo implementacijo skupnih vmesnikov.

```
class reuseMe {
```

```
    function reuseMe(){...}
```

```
    function doTask1(){...}
```

```
    function doTask2(){...}
```

```
    function doTask3(){...}
```

```
}
```

```
Class extends reuseMe {
```

```
    function example(){  
        ... // local initializations  
        // call super constructor  
        reuseMe::reuseMe();  
    }
```

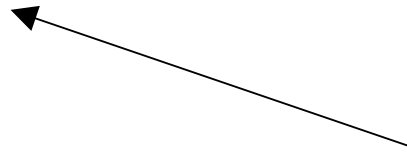
```
    function doTask4(){...}
```

```
    function doTask5(){...}
```

```
    function doTask6(){...}
```

```
    function doTask3(){...}
```

```
}
```



Povzetek

- PHP je proceduralen jezik
 - V primerjavi z Javascript, ki je dogodkovno voden
- Sintaksa je podobna jeziku C { } ;
- na voljo je obširna knjižnica funkcij
- Zelo dobra integracija s spletnim strežnikom
 - Skript je vgrajen v HTML
 - Lahek dostop do podatkovnih obrazcev in izpis HTML strani
- Ni povsem objektno usmerjen
 - Java je povsem objektno usmerjena – vse funkcije morajo pripadati kakšnemu razredu
 - V PHP so razredi preprost dodatek