JavaScript and PHP

A little about JavaScript

JavaScript – What is it good for?

JavaScript – What is it good for?

JavaScript is a general programming language meant to be embedded into HTML pages to enhance the code and to add dynamics to the normally static content of HTML pages.

JavaScript – What is it good for?

JavaScript is a general programming language meant to be embedded into HTML pages to enhance the code and to add dynamics to the normally static content of HTML pages.

It may be used for everything that HTML is used for and much more and it is meant to execute on the client side.

JavaScript – What is it good for?

JavaScript is a general programming language meant to be embedded into HTML pages to enhance the code and to add dynamics to the normally static content of HTML pages.

It may be used for everything that HTML is used for and much more and it is meant to execute on the client side.

However, CSS manages most of the problems that JavaScript can manage, so today there are not many reasons left to use it.

JavaScript – What is it good for?

JavaScript is a general programming language meant to be embedded into HTML pages to enhance the code and to add dynamics to the normally static content of HTML pages.

It may be used for everything that HTML is used for and much more and it is meant to execute on the client side.

However, CSS manages most of the problems that JavaScript can manage, so today there are not many reasons left to use it.

Some small tasks are still best taken care of by JavaScript.

JavaScript – What is it good for?

JavaScript is a general programming language meant to be embedded into HTML pages to enhance the code and to add dynamics to the normally static content of HTML pages.

It may be used for everything that HTML is used for and much more and it is meant to execute on the client side.

However, CSS manages most of the problems that JavaScript can manage, so today there are not many reasons left to use it.

Some small tasks are still best taken care of by JavaScript.

NOTE! JavaScript is case sensitive

JavaScript – a universal programming language

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.
- It has features to reach all objects and parameters of web pages.

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.
- It has features to reach all objects and parameters of web pages.
- JavaScript has nothing to do with Java. It originated as *LiveScript* and changed name for marketing purposes.

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.
- It has features to reach all objects and parameters of web pages.
- JavaScript has nothing to do with Java. It originated as *LiveScript* and changed name for marketing purposes.
- Most language constructions are as in Java but a weak type system that allows all kinds of values to be bound to a variable:

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.
- It has features to reach all objects and parameters of web pages.
- JavaScript has nothing to do with Java. It originated as *LiveScript* and changed name for marketing purposes.
- Most language constructions are as in Java but a weak type system that allows all kinds of values to be bound to a variable:

  `var a, s = 3;` may be followed by `s = "hello";`

JavaScript – a universal programming language

- You can do all kinds of tasks with JavaScript. The special version that is embedded into browsers have some special features though.
- It has features to reach all objects and parameters of web pages.
- JavaScript has nothing to do with Java. It originated as *LiveScript* and changed name for marketing purposes.
- Most language constructions are as in Java but a weak type system that allows all kinds of values to be bound to a variable:

  `var a, s = 3;` may be followed by `s = "hello";`

  Here, variables `a` and `s` have been declared and an initial value has been assigned to `s`. Then `s` has been given a value of a different type.

Mandatory ”Hello World” JavaScript program

Mandatory ”Hello World” JavaScript program

```
<html>

</html>
```

Mandatory ”Hello World” JavaScript program

```
<html>
   <head>
      <title>Hello World</title>
   </head>




</html>
```

Mandatory ”Hello World” JavaScript program

```
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>




   </body>
</html>
```

Mandatory ”Hello World” JavaScript program

```
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <script type="text/javascript">


      </script>
   </body>
</html>
```

Mandatory ”Hello World” JavaScript program

```
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <script type="text/javascript">
         document.writeln("Hello World");
      </script>
   </body>
</html>
```

JavaScript . . .

JavaScript . . .

- No need to use semicolon (';') unless you put more than one statement on the same line. But as you may end statements with a semicolon, it's best to allways do it.

JavaScript . . .

- No need to use semicolon (';') unless you put more than one statement on the same line. But as you may end statements with a semicolon, it's best to allways do it.
- Strings may be enclosed in quotes ('"') or in apostrophes ('′') allowing for simple inclusion of these into strings.

JavaScript . . .

- No need to use semicolon (';') unless you put more than one statement on the same line. But as you may end statements with a semicolon, it's best to allways do it.
- Strings may be enclosed in quotes ('"') or in apostrophes ('') allowing for simple inclusion of these into strings.

```
var s1 = "it's rather cold today";
var s2 = ' so the "goblins" stay in their caves';
```

JavaScript . . .

- No need to use semicolon (';') unless you put more than one statement on the same line. But as you may end statements with a semicolon, it's best to allways do it.
- Strings may be enclosed in quotes (`'"'`) or in apostrophes (`'''`) allowing for simple inclusion of these into strings.

```
var s1 = "it's rather cold today";
```

```
var s2 = ' so the "goblins" stay in their caves';
```

- The string `var s3 = s1 + s2;` will contain the value

```
it's rather cold today so the "goblins" stay in their caves
```

JavaScript . . .

You may put your script anywhere in the HTML page but the normal is to put declarations in the `<head>...</head>` part and the rest exactly where you want things to happen. To check what i mentioned in the previous slide you may put the following in a file (the name ending with '.html'):

JavaScript . . .

You may put your script anywhere in the HTML page but the normal is to put declarations in the `<head>...</head>` part and the rest exactly where you want things to happen. To check what i mentioned in the previous slide you may put the following in a file (the name ending with '.html'):

```
<html>
   <head>
      <script type="text/javascript">
         var s1 = "it's rather cold today";
         var s2 = ' so the "goblins" stay in their caves';
      </script>
      <title>Testing javascript strings</title>
   </head>
   <body>
      <script type="text/javascript">
         document.write(s1 + s2);
      </script>
   </body>
</html>
```

Find out if JavaScript is turned on (or not available)

If you want to use JavaScript in a page, JavaScript must be enabled in the browser. There are no means to ask the browser if JavaScript is enabled, but you may have a page using JavaScript to redirect the user to the real web page and, in case it doesn't work, issue an error message.

Find out if JavaScript is turned on (or not available)

If you want to use JavaScript in a page, JavaScript must be enabled in the browser. There are no means to ask the browser if JavaScript is enabled, but you may have a page using JavaScript to redirect the user to the real web page and, in case it doesn't work, issue an error message.

```
<html>
   <head>
      <title>JavaScript test</title>
      <script type="text/javascript">
         location='start.html';
      </script>
   </head>
   <body>
      <h4>Sorry, JavaScript is not enabled in your browser!</h4>
      To visit this site, enable JavaScript and then reload this page.
   </body>
</html>
```

Hiding JavaScript from not enabled browsers

```
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <script type="text/javascript">
        <!--
         document.writeln("Hello World");
        // -->
      </script>
   </body>
</html>
```

Checking for JavaScript in enabled browsers

```
<html>
  <head>
    <title>Checking if JavaScript is on</title>
  </head>
  <body>
    <script type="text/javascript">
      <!--
       // your JavaScript code here
      // -->
    </script>
    <noscript>
      Your browser understands JavaScript. However,
      the feature is turned off. Please turn it on
      to enjoy the full capability of this page.
    </noscript>
  </body>
</html>
```

Using JavaScript in modern HTML

As said, most things that JavaScript can do, are better performed by server side languages, so let's focus on things that are more convenient to do on the client side, e.g. using navigation history or perform form input control.

Using JavaScript in modern HTML

As said, most things that JavaScript can do, are better performed by server side languages, so let's focus on things that are more convenient to do on the client side, e.g. using navigation history or perform form input control.

Using JavaScript to navigate

```
<html>
  <head>
    <title>Going back to the previous page</title>
  </head.
  <body>
    The password you typed in is not long enough.
    Please <a href="javascript:history.back()">go
    back</a> to correct it.
  </body>
</html>
```

Using JavaScript to navigate . . .

```
<html>
  <head>
    <title>Going n pages back </title>
  </head>
  <body>
    Click <a href="javascript:history.go(-3)">here</a>
    to go back to the the beginning of the entry form.
  </body>
</html>
```

Using JavaScript to navigate . . .

```
<html>
  <head>
    <title>Going n pages back </title>
  </head>
  <body>
    Click <a href="javascript:history.go(-3)">here</a>
    to go back to the the beginning of the entry form.
  </body>
</html>
```

```
<html>
  <head>
    <title>Moving forward</title>
  </head>
  <body>
    Click <a href="javascript:history.go(1)">here</a>
    to move forward.
  </body>
</html>
```

Using JavaScript to navigate . . .

```
<html>
  <head>
    <title>Navigating to a CSC course</title>
  </head>
  <body>
    Please select one of the courses below
    <br />
    <form>
      <select onChange="location='http://www.nada.kth.se/kurser/kth/' +
                        this.options[this.selectedIndex].value">
        <option>Select a course</option>
        <option value="DD1334/dbtek10/">dbtek10</option>
        <option value="DD1335/gruint10/">gruint10</option>
        <option value="DD2471/moddb10/">moddb10</option>
      </select>
    </form>
  </body>
</html>
```

Using JavaScript for form input control

First we need some basic functions. In the following i omit the

```
<script type="text/javascript">
</script>
```

part as it is obvious that all functions have to be declared inside a script.

The `isEmpty` function

```
function isEmpty(str) {
  if (str == null || str == "")
    return true;
  return false;
}
```

Using JavaScript for form input control . . .

The `trim` function

```
function trim(str) {
  if (str!=null) {
    while (str.length > 0 && str.charAt(str.length - 1) == " ")
      str = str.substring(0, str.length - 1);
    while (str.length > 0 && str.charAt(0) == " ")
      str = str.substring(1, str.length);
  }
  return str;
}
```

Form validating function

```
function validateForm(theForm) {
  if (isEmpty(trim(theForm.username.value))) {
    alert('Please enter a valid name'); return false;
  }
  if (isEmpty(trim(theForm.passwd.value))) {
    alert('Please enter a valid password'); return false;
  }
  return true;
}
```

Using JavaScript for form input control . . .

```
<html>
  <head><title>Form Validation</title>
    <script type="text/javascript">
      // Here is the place for the functions from previous slides
    </script>
  </head>
  <body>
    <form method="post" action="enter.html"
          onSubmit='return validateForm(this)'>
      <table>
        <tr><td>Name:</td>
          <td><input type="text" name="username"></td></tr>
        <tr><td>Password:</td>
          <td><input type="password" name="passwd"></td></tr>
        <tr><td colspan="2" align="right">
          <input type="submit"></td></tr>
      </table>
    </form>
  </body>
</html>
```

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)
- A number of PHP organisations and hosting services around the net

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)
- A number of PHP organisations and hosting services around the net
- A huge catalogue of features developed by volunteers

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)
- A number of PHP organisations and hosting services around the net
- A huge catalogue of features developed by volunteers
- Language in evolution . . . PHP 5 adds exception handling. OOP is also an add-on: Introduced in PHP4, perfected in PHP5

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)
- A number of PHP organisations and hosting services around the net
- A huge catalogue of features developed by volunteers
- Language in evolution . . . PHP 5 adds exception handling. OOP is also an add-on: Introduced in PHP4, perfected in PHP5
  - most OOP concepts and keywords are as in java: class, extends, interface, private, protected, etc

Quick overview of PHP

What is PHP

- PHP: Hypertext Processor (or Personal Home Page?)
- Interpreted language
  - Easy to get something done fast
  - Problems when size and complexity grow
  - Performance penalty (though addressed in the Zend engine)
- A number of PHP organisations and hosting services around the net
- A huge catalogue of features developed by volunteers
- Language in evolution . . . PHP 5 adds exception handling. OOP is also an add-on: Introduced in PHP4, perfected in PHP5
  - most OOP concepts and keywords are as in java: class, extends, interface, private, protected, etc
  - exceptions are as in Java (try/catch, etc)

What can PHP do?

- CGI scripting of course

What can PHP do?

- CGI scripting of course
- Based on escapes within HTML pages
  `<?php ... ?>` or `<% ... %>`

What can PHP do?

- CGI scripting of course
- Based on escapes within HTML pages
  `<?php ... ?>` or `<% ... %>`
- Command line scripting

What can PHP do?

- CGI scripting of course
- Based on escapes within HTML pages
  `<?php ... ?>` or `<% ... %>`
- Command line scripting
- GUI applications (PHP-GTK)
  http://gtk.php.net/

Installation

- Install the PHP interpreter separately. There is good support for this in Linux

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine
- Apache is the typical choice for the web server.

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine
- Apache is the typical choice for the web server.
  - Integrated as a module, so no supplementary processes are created (in CGI, typically there is one process per access, which is very expensive)

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine
- Apache is the typical choice for the web server.
  - Integrated as a module, so no supplementary processes are created (in CGI, typically there is one process per access, which is very expensive)
- Mysql or PostgreSQL is the typical db engine

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine
- Apache is the typical choice for the web server.
  - Integrated as a module, so no supplementary processes are created (in CGI, typically there is one process per access, which is very expensive)
- Mysql or PostgreSQL is the typical db engine
- Most MS Windows users prefer Mysql

Installation

- Install the PHP interpreter separately. There is good support for this in Linux
- In the webserver configuration, associate the php extension with the PHP interpreter
- For serious applications you will need a database engine
- Apache is the typical choice for the web server.
  - Integrated as a module, so no supplementary processes are created (in CGI, typically there is one process per access, which is very expensive)
- Mysql or PostgreSQL is the typical db engine
- Most MS Windows users prefer Mysql
- I'd choose PostgreSQL as it is more complete in terms of functionality

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP

  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP

  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`
- As in JSP, escaping can be interrupted to write some HTML

```
<?php if ($expression) { ?> <b>This is true.</b> <?php } ?>
```

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP

  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`
- As in JSP, escaping can be interrupted to write some HTML

  `<?php if ($expression) { ?> <b>This is true.</b> <?php } ?>`
- You can see the evolution under community pressure, here and in other areas

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP
  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`
- As in JSP, escaping can be interrupted to write some HTML
  `<?php if ($expression) { ?> <b>This is true.</b> <?php } ?>`
- You can see the evolution under community pressure, here and in other areas
- Though it is possible to use
  `<script language="php" > ....</script>`, don't ...

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP
  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`
- As in JSP, escaping can be interrupted to write some HTML
  `<?php if ($expression) { ?> <b>This is true.</b> <?php } ?>`
- You can see the evolution under community pressure, here and in other areas
- Though it is possible to use
  `<script language="php" > ....</script>`, don't ...
  - the `language` attribute is deprecated

PHP in HTML

- The most frequent use of PHP is in scripts embedded in web pages
- Escaping follows the same rules as in ASP and JSP
  `<?php .... ?>`, `<? .... ?>`, `<% .... %>`, `<%=. ... %>`
- As in JSP, escaping can be interrupted to write some HTML
  `<?php if ($expression) { ?> <b>This is true.</b> <?php } ?>`
- You can see the evolution under community pressure, here and in other areas
- Though it is possible to use
  `<script language="php" > ....</script>`, don't ...
  - the `language` attribute is deprecated
  - PHP is a server side language and the `<script>` tag is for client side processing

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring
  - string functions in the function library

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring
  - string functions in the function library
- arrays are mappings between keys and values (Dictionary/Hashtable/Map in java)

```
$arr = array("foo" => "bar", 12 => true)
```

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring
  - string functions in the function library
- arrays are mappings between keys and values (Dictionary/Hashtable/Map in java)
  ```
  $arr = array("foo" => "bar", 12 => true)
  ```
- Automactic type conversions between types (very dangerous...). Explicit type conversions exist too

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring
  - string functions in the function library
- arrays are mappings between keys and values (Dictionary/Hashtable/Map in java)
  `$arr = array("foo" => "bar", 12 => true)`
- Automactic type conversions between types (very dangerous...). Explicit type conversions exist too
- Classes and objects, OOP

Types

- A variable name begins with `$`, no type declaration (type declarations can be required by passing some settings to the interpreter)
- weak type system as in JavaScript
- boolean, integer, float, though internally all numbers are floats.
- string
  - single-quoted, no character escapes
  - double-quoted, with character escapes
  - `$a.$b` appends the string b and at the end of the string a
  - `$a(index1, index2)` gives a substring
  - string functions in the function library
- arrays are mappings between keys and values (Dictionary/Hashtable/Map in java)
  `$arr = array("foo" => "bar", 12 => true)`
- Automactic type conversions between types (very dangerous...). Explicit type conversions exist too
- Classes and objects, OOP
- Resources, a kind of reference

Variables

- See Types

Variables

- See Types
- Assignment by value (not by reference)

Variables

- See Types
- Assignment by value (not by reference)
- Lots of predefined variables, especially related to `HTTP/CGI`

Variables

- See Types
- Assignment by value (not by reference)
- Lots of predefined variables, especially related to `HTTP/CGI`
  - `_SERVER`, `_GET`, `_POST`, `_COOKIE`, `_FILES`, `_REQUEST`, `_SESSION`

Variables

- See Types
- Assignment by value (not by reference)
- Lots of predefined variables, especially related to `HTTP/CGI`
  - `_SERVER`, `_GET`, `_POST`, `_COOKIE`, `_FILES`, `_REQUEST`, `_SESSION`
- External variables, useful for forms

Variables

- See Types
- Assignment by value (not by reference)
- Lots of predefined variables, especially related to `HTTP/CGI`
  - `_SERVER`, `_GET`, `_POST`, `_COOKIE`, `_FILES`, `_REQUEST`, `_SESSION`
- External variables, useful for forms
- Functions as variables

Other procedural stuff

- Operators similar to Java

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`
  - `foreach ()` through arrays, just values, or also keys

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`
  - `foreach ()` through arrays, just values, or also keys
  - `foreach (array_expression as $value)` statement

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`
  - `foreach ()` through arrays, just values, or also keys
  - `foreach (array_expression as $value)` statement
  - `foreach (array_expression as $key => $value)` statement

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`
  - `foreach ()` through arrays, just values, or also keys
  - `foreach (array_expression as $value)` statement
  - `foreach (array_expression as $key => $value)` statement
- Code inclusion with `require()` and `include()`

Other procedural stuff

- Operators similar to Java
- Statements similar to Java, plus:
  - `<? if(...) : ?> ....<? endif; ?>`
  - `foreach ()` through arrays, just values, or also keys
  - `foreach (array_expression as $value)` statement
  - `foreach (array_expression as $key => $value)` statement
- Code inclusion with `require()` and `include()`
- Conditional function definition (similar to C `#ifdef`)

Features

- HTTP authentication, cookies, file uploads

Features

- HTTP authentication, cookies, file uploads
- Remote files (like `java.net.URLConnection`)

Features

- HTTP authentication, cookies, file uploads
- Remote files (like `java.net.URLConnection`)
- Connections (like `java.net.Socket`)

Features

- HTTP authentication, cookies, file uploads
- Remote files (like `java.net.URLConnection`)
- Connections (like `java.net.Socket`)
- Persistent db connections

Features

- HTTP authentication, cookies, file uploads
- Remote files (like `java.net.URLConnection`)
- Connections (like `java.net.Socket`)
- Persistent db connections
  - Normally db connections are defined with engine specific functions as external resources. Each access would open its connection, which is expensive. So applications with extensive use of databases had better use Java or pay a heavy performance penalty.

Functions

- Array functions, calendar fuctions, date functions, character functions, IO functions, printer, file/directory, etc

Functions

- Array functions, calendar fuctions, date functions, character functions, IO functions, printer, file/directory, etc
- Functions for protocols/standards, e.g. FTP, HTTP, URL, LDAP, IRC, Mail, NSAPI, popmail, XML/XSL, Bzip2/Zip/Zlib

Functions

- Array functions, calendar fuctions, date functions, character functions, IO functions, printer, file/directory, etc
- Functions for protocols/standards, e.g. FTP, HTTP, URL, LDAP, IRC, Mail, NSAPI, popmail, XML/XSL, Bzip2/Zip/Zlib
- Functions for various databases (Mysql, Oracle, MS SQL server, mSQL, PostgreSQL, SQLite, dBase), dbx is general

Functions

- Array functions, calendar fuctions, date functions, character functions, IO functions, printer, file/directory, etc
- Functions for protocols/standards, e.g. FTP, HTTP, URL, LDAP, IRC, Mail, NSAPI, popmail, XML/XSL, Bzip2/Zip/Zlib
- Functions for various databases (Mysql, Oracle, MS SQL server, mSQL, PostgreSQL, SQLite, dBase), dbx is general
- Functions for other systems/tools: e.g. Apache, COM, Cyrus, PDF, dBase, DBM, DOM, .NET, Lotus Notes, GNU readline, Hyperware

Conclusions

- Easy to learn from Java or C

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize
- Good language/system for doing something small fast (but then JSP/ASP do most of the same)

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize
- Good language/system for doing something small fast (but then JSP/ASP do most of the same)
- Not a wise choice for a serious/large project due to the lack of type safety, lack of OOP in the libraries, etc.

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize
- Good language/system for doing something small fast (but then JSP/ASP do most of the same)
- Not a wise choice for a serious/large project due to the lack of type safety, lack of OOP in the libraries, etc.
  - Experienced PHP people confirm that larger projects tend to become a big mess due to the freedoms that seemed so good in the begining, which make programmers lazy

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize
- Good language/system for doing something small fast (but then JSP/ASP do most of the same)
- Not a wise choice for a serious/large project due to the lack of type safety, lack of OOP in the libraries, etc.
  - Experienced PHP people confirm that larger projects tend to become a big mess due to the freedoms that seemed so good in the begining, which make programmers lazy
- The array concept is nice, but its name is misleading (array means something very different in the rest of the field of Computer Science)

Conclusions

- Easy to learn from Java or C
- CGI parameters, HTTP sessions, etc are easy to recognize
- Good language/system for doing something small fast (but then JSP/ASP do most of the same)
- Not a wise choice for a serious/large project due to the lack of type safety, lack of OOP in the libraries, etc.
  - Experienced PHP people confirm that larger projects tend to become a big mess due to the freedoms that seemed so good in the begining, which make programmers lazy
- The array concept is nice, but its name is misleading (array means something very different in the rest of the field of Computer Science)
- Still, a very good choice for pragmatists