Web Programming Course

Resources

Link

- all online courses ⇒ https://docs.google.com/document/d/1Q4NSd4gqRloFseLCLN0JI_nznNW-dTX_Iu5YkP3J3NY/edit
- assignments ⇒ https://www.cs.ubbcluj.ro/~forest/impulse/#
- Forest's website ⇒ https://www.cs.ubbcluj.ro/~forest/wp/
- Forest's youtube ⇒ https://www.youtube.com/@adriansterca2719

Lecture 1 - HTML (HyperText Markup Language)

publishing language of WorldWideWeb

Elements of HTML = tags
<tag-name attribute1="val1" … attributeN="valN" event ="function">
text-cotentn
</tag-name>
or self-closing tag: <tag-name attributes event />

Comments: <!-- →

Example:
<!DOCTYPE html>
<html>
<head>
meta-data like description, title, etc.
</head>
<body>
actual content
</body>
</html>

Tags

MetadataTags

<title> …</title>

<base href="url" target="_blank | _parent | _self | _top | framename ">
this tag specifies a default URL and a default target for all the links of a page
<link> - the relationship between a document and an external source;
attributes: href, rel, rev, type, target
<link rel="stylesheet" type="text/css" href="./css/style.css">

<meta> - information about the html doc, is not displayed

<meta name="description" content="…">
<meta name="author" cotent="…">
<meta name="keyword" content="a,b,c,…">

<style> - to define CSS inside html

SectionTags

<body> - contains all the text, links, tables, img etc.

<head> - for metatags

<div> - section in HTML, groups together elements

<frameset> - set of frames; mutually exclusive w body

```
<frameset cols="pixels | x % | numOfCols" rows="analog">
    <frame src="frame_a.html">
    ....
</frameset>
```

<frame> - a frame (window) within a frameset

<iframe> - an inline frame that contains another doc withing the current doc

<h1><h2>...<h6> - headings

<p> - paragraph

<b> bold

<i> italic

<strong> strong text

<u> underline

<s> strikethrough

<del> deleted text

<sub> and <sup> subscript and superscript text

<pre> - preformatted text

<small> and <big>

Grouping Tags

```
<dl>
    <dt> Name 1 </dt>
    <dd> Name 1 dd </dd>
    <dt> Name 2 </dt>
    <dd> Name 2 dd </dd>
    <dt> Name 3 </dt>
    <dd> Name 3 dd </dd>
</d>
===⇒
Name 1
          Name 1 dd
Name 2
          Name 2 dd
Name3
          Name 3 dd
```

Lists

ordered list:

```
<ol>
     <li> elem 1</li>
     <li> elem2 </li>
</ol>
```

unoredered list:
<ul>

drop-down list:

```
<select>
  <option value="ferrari">Ferrari</option>
```

  .....
</select>

Image Tag

<img src=”…” height=”…” width=”…%” border align>

Anchor Tag

links the current doc to another document or section of a document (w doc id)

<a href=”…”> link </a>

Table Tag

<table> - table haed
<th> - table header>
<tr> - table row
<td> - table data

Script Tag

for inserting action scripting

<script src=”main.js” type=”text/javascript”>

Other Tags

<br/> - moves to next line
<hr/> - draws a horizontal line

<svg> - to create some sort of drawing type shit

<audio> - for audio

<video width height>
<source src= “” type=”video/mp4 or /ogg or whatever>
</video>

<figure>
<img src=””>
<figcaption> what is in the image </figcaption>
</figure>

Structural Tags

<main>

<section>

<article>

<header>

<footer>

<nav>

<aside>

HTTP (HyperText Transfer Protocol)

HTML Forms

```
<form attribute="value">
    text...(label tag)
    input....(input tag)
</form>
```

those are used for getting user input

<form> attributes

action = URL: where the form-data is submitted to

accept= MIME type - what types of files can be submittet through upload

enctype

method = get / post (get→form data is sent to the web sv in the header of the http request
post→ in the body)

name = string →name of the form

<input> tag

type =

- text
- password
- button
- reset
- submit
- radio(can choose one out of any)
- checkbox
- file
- image
- hidden

accept = MIME_type for file type

checked for checkbox/radio

alt = text for type image

disabled

maxlength for type text/password

name = text

readonly for text/password

size = number: width of the input elem

src=url for type image as a submit button

value = text → value of the input elem

<textarea> tag

a multi-line text input control, unlimited no characters, fixed-width

cols = numbers: visible number of columns in the text area

rows = number: visible no rows

disabled

readonly

name

<label> tag

defines a label for the input elem

<button> tag

defines a push button, that can contain text or images

disabled

name = text

type = button | reset | submit

value = text

<select> and <option>

```
<select>
    <option value="ford"> Ford</option>
    <option value ="ferrari">Ferrari</option>
<select>
```

attributes of select:
disabled
multiple = multiple (allows multiple selections)
name
size = number : num of visible options

attributes of option:
disabled
selected = selected (this is selected by default)
value = text

Other tags

<legend> - defines a caption for a fieldset element

```
<fieldset>
   <legend> caption </legend>
    <input type="..."><br>
    <input type="...">
</fieldset>
```

<optgroup> ⇒ GROUPS together related options in a select list

```
<optgroup label="Fruits">
     <option value="">...</option>
     <option value="">...</option>
</optgroup>
<optgroup label="Sports">
     <option value="">...</option>
     <option value="">...</option>
</optgroup>
```

Sets of HTML characters

ASCII

ISO-8859-1

Math, greek and other symbols

URL

- url identifies a resource in the www
- subset of URIs = Uniform Resource Identifiers

GENERAL FORM OF URL

resource_type:://domain:port/filepathname?querystring#anchor

resource_type: the scheme name(protocol) which defines the namespace, syntax and remaining part of the URL

domain: registered domain name or IP address of location (case-insensitive)

port: port number, optional

fielpathname: path to the resource/file on the server

querystring: data submitted to the server through forms

anchor: specific location inside that doc

URI

foo://username:pasword@example.com:8042/over/there/index.dtb:type=animal?name=ferret#nose

foo → scheme

authority:

- userinfo = username and password
- hostname = example.com
- port = 8042
- pathfile = /over/there/index.dtb
- filename = index
- file extension = .dtb
- parameter: type=animal
- query: name=ferret
- nosse - fragment

Web Communication

When you want to access a webpage, you send an HTTP request over the Internet, that's received by a server and then you get an HTTP reply over the internet

HTTP - HyperText Transfer Protocool

- together with HTML forms the base of the WWW
- a request-response protocol
- stateless (does not maintain a state of a session)
- asynchronous (parts of the HTML are loaded asynchronous on the webpage as soon as they are available)
- runs on top of TCP, standard port = 80

HTTP Request

Request-Method SP Request-URL SP HTTP-Version <cr> <lf>
(generic header | request header | entity-header <cr> <lf>)
<cr> <lf>
[message body]

Request Method

- get requests information identified by the request url
- post - request that serer accepts the entity enclosed in the Request
- options - requests information about communication options
- delete - request that the server delete the resource identified by Request-URL
- trace
- connect - used by proxies in SSL connections
- HEAD - identical to get, but server doesnt need to return a message body in response

Request Header

it can have the following fields

- Accept : MIME types of resources accepted by the browser
- Accept-Charset
- Accept-Language
- Accept-Encoding
- Authorization: user-agent wishes to authenticate itself w a sv
- Host: the host Request-URL points to
- Referer: the URL of document refering this URL
- User-Agent: Firefox, Safari etc.

HTTP Response

HTTP-Version SP Status-Code SP Reason-Phrase <cr> <lf>
(generic header | request header | entity-header <cr> <lf>)
<cr> <lf>
[message body]

Response Header:

- Age: amount of time since the response was generated by the server
- Location: redirect the client to a location other than RequestURL for completion of the request
- Server: info about software used by the sv to handle the request
- Retry-After: indicate to client how long the service is expected to be unavailable
- Accept-Ranges: server indicates its acceptance of range requests

CSS and CSS3

CSS is used to defines how to display an html document

Syntax

selector {
property: value;
property: value;

....
}

Selectors

- a tag name: p, a, body
- a group of tag names: h1,h2,h3,h4,h5,h6{} or header, fooder{} etc.
- a class name: .myClass{}
- an id: #myId{}

Pseudo-classes

- special kind of selectors that select multiple tags and the are diferent tha the selectors presented above
- a:visited
- a:hover
- a:active
- p:first-child
- li:nth_child

Pseudo-elements

:first-letter

:first-line

:before

:after

Adding style sheets to a document:

- specific the link in the head, to an external sheet:
  <link rel="stylesheet" src="...">
- specify the style inline:
  <p style=" color:red;"> ...</p>
- add a style tag inside the <head>
  <style>
  p{ margin: 2rem;}
  h1{color:red;}
  </style>

Background properties

- background - for all properties in one
- background-attachment - wheter a background image is fixed or scrolls
- background-color
- background-image
- background-position - starting position of a background image
- background-repeat - how many times the image will repeat

The Box Model

- the box model is as following:
  an element has a content and then a padding, then comes the border and then the margin

Margin

it is completely transparet, has a width on each side, and sets elements apart from each other

PROPERTIES

- margin-bottom / -left /-right /-top : width

Padding

same properties as margin; this one sets a 'margin' between the content and the border, in a way

Border

- border: set all the border properties in one declaration
- border-bottom: sets all bottom border properties (there is top, left and right too)
- border-color
- border-style
- border-width
- border-radius: rounded corners
- outline: sets all outline properties in one declaration
- outline-color
- outline-style
- outline-width

Dimension properties

- height, max-height, max-width, min-height, min-width and width
- they can be specified as a % of the parent element, as viewheights or viewwidths (how much of the entire browser view (100vh = all height), in rem, px etc. (same for padding and margin)

Text and font properties

Font

- color
- direction
- letter-spacing: spacing between every2 characters
- line-height
- text-aling: how to align the text within it's box
  center
  left
  right etc.
- text-decoration
- text-indent: indentation of the first line in a text-block
- text-shadow
- text-transform: controls the capitalization of text
- vertical-align : aling vertically

- white-space: how white-space inside an elem is handled
- word-spacing: increases the spaces between words (or decreases)

Font

- font: sets all
- font-size
- font-style
- font-family
- font-weight: 500,600,700=bold etc.
- font-variant

List and table properties

List

- list-style
- list-style-image
- list-style-type: type of the list-item marker
- list-style-position: where to place the marker

Table

- border-collapse
- border-spacing
- caption-side
- empty-cells
- table-layout

Positioning

- bottom
- clear
- clip
- cursor - cursor type to be displayed
- float - right / left; → elements can be pushhed left or right, other elemennts can wrap around them;
- left
- right
- top
- z-index: z-index of -1 is under all other standard elements, while z-index 4 would be above;
- position
- overflow
- display

Types of positioning

- static - default

- fixed - will not move even if the window is scrolled;
  position:fixed;
  top:20px;
  left:10px;
  ⇒ so it's fixed away 20px from the top and 10 from the left
- relative: relative to it's normal position in his parent element
- absolute: it is relative to the first parent element that has a position other thhen static; if none, then to <html>
- sticky - it sticks

Display

display: inline; - takes as much width as necesarry

display:block; - takes the whole width or however much is specified;

display:inline-block; it is an inline elemenet for which you can specify the height and width

display: flex; → flex container

display:grid; → grid contaiiner

display:none; → it doesnt show

CSS3

Selectors

- nth-child(n)
- nth-of-type(n) - the n-th sibling of the element specified
- first-of-type
- last-of-type
- E +F element immediately after E
- E > F ; F, child of E
- * - everything
- E - F ⇒ the element preceding E
- etc.

Gradient colors and graphics transforms

- linear-gradient(to top/left/right/bottom, start-color: rgba(30,100,255,0.85), to-color: #f4f4f4);
- radial-gradient = elliptical radient defined by it's center;
  radial-gradient(center-position shape size, color1, color2, color3…)
  center-position = center(default) or 2 points
  shape = circle / ellipse
  size = radius of gradient given as length or percentage or closest-side, farthest-side, closest-corner, farthest-corner
- conic-gradient(from angle [at position], color degree, color degree, …);

Transform

- scale(X,Y) or scaleX(x) or scaleY(y) - scales the dimension on the X axis or on the Y axis
- rotate(angle)
- translate(x,y) moves elements along X andd Y axis (translateX() and translateY())
- skew(X-angle, Y-angle)

Transitions and animations

- transition-property: what properties can be modified
- transition-duration: how long it lasts
- transition-delay: the delay at which it starts
- transition-timing-function: ease/ease in/ease out/ ease in-out
- transition === all in one

- @Keyframes defines the frames of an animation
  @keyframes myAnimation{
  0% { /* the css at 0% animation */}
  30% {...}
  100%{...}
  }
  .animated{
  animation: myAnimation 3s ease-in;
  }
- animation-name: what keyframe animation
- animation-duration: how long it takes
- animation-delay
- animation-timing-function
- animation-iteration-counter
- animation-play-state
- animation for all

Borders, shadows, backgrounds and sprites

- border-radius for each corner and length/percentage
- background-position
- background: url('myPhoto.png');

SHADOWS

- text-shadow: 2px 2px 4px #ff00dd;
  h-shadow, v-shadow and blur color
- box-shadow: 4px 6px 6px -2px #aaff32;
  h-shadow v-shadow blur-spread color inset;

FLEXBOX

- An item that has display: flex; is a flex container and his children elements are flex-items
- a flexbox aligns items on a row or on a column, specified by flex-direction;
- justify-content: align of flex items on MAIN axis
  center, start, end, space-around, space-between, space-evenly
- align-items: alignment on CROSS axis
  stretch, basline, center, flex-start, flex-end
- gap: gap between elements

Multiple columns

- column-count: no of columns an element is divided
- column-fill - how to fill the columns: Balance | Auto
- column-gap: space between columns
- column-span: span of a column
- column-width: width of a column
- column-rule -style/-color/-width
- columns: shorthand for all

Responsive Web Design

@media only screen and (max/min-width: 600px){

/* styling for certain elemenst that needs to change for smaller/wider screens*

}

CSS neat typography: web fonts and neat icons

- you can utilise google web fonts, or font-awesome icons by importing them:
  <link rel=”stylesheet” href=”https://fonts.googleapis.com/icon?family=Mateiral+Icons”>
- there are many other methods, no reason to list them here

Javascript and Document Object Model (DOM)

- https://www.youtube.com/watch?v=QJjY1srfRWM

Main Properties

- used in the browser
- exec on the client (node.js → on the server side)
- used to add functionality to HTML

Intro

- documnet.getElementById(”myId”) → gets the element with id = myId from that html doc
- document.getElementById(”myId”).innerHTML = ‘Hello World!’;
  → displays HelloWorld! inside the element with that id
- change the image
  document.getElementById(”myImage”).src= ‘…’;
- change an attribute:
  document.getElementById(”id”).style.fontSize = “32px”;
  document.getElementById(”id”).style.display =”none”;
- Insert JavaScript in HTML: (either in body or in head)
  <script>
  document.getElem…
  </script>
  ————-
  <script src=”../myFolder/javascripts/myJavascript.js”></script>
- Function:
  function myFunction(){
  document.getElemById().style.display=”none”;

}
→ and it can then be attributed to a button onClick or smth

- // for comments
- /* ...
  ...
  multiple line comments
  ...
  */

Js Display Possibilties

- innerHTML → changes the content of the html (for a div, p, h1-6 etc.)
- document.write(...) → will write in the HTML
  IF document.write(..) is used after HTML is loaded ⇒ all existing HTML is deleted!!!!!
  - only use for testing
- (window.) alert(...) → creates an alert window with a message (a pop up)
- console.log() → displays smth inside the console, helps w debug
- window.print() → literally prints the whole page

Operators

- =, +, -, *, %, /, ++, --, **(exponent)
- +=, -= etc.
- == (equal)
- ===(equal and equal type)
- !=(not equal)
- ! == (not equal value / type)
- >, <, >=, <=
- LOGICAL: &&, ||, !
- typeof → returns the type
- instanceof→ ret true if an object is an instance of an object type
- Binary Operators:
  &, |, ~(not), ^(xor), >>, << (left/right shift)

Types and liberals

- numbers: integer(base 2 8 10 16) and real
- boolean: trull/false
- null - abscence of a value
- undefined - val of a variable that hasnt gotten a value yet
- NaN -Not a Number
- String
- vectors: ['a',,, 'bbb', `ccc'] → 5 elementss
- Objects: list of 0 or more pair: property < = > value
  - Example:

    - dog = { name: dog, type: animal, characteristics: getProps(”dog”), age:4}

Automatic type conversion:

when applying operators, javascript automatically convert parameters to the same type, depending on the context:

a = “string”+2;	⇒ “string2”
b = 2+”3”	⇒ “23”
c =2+true	⇒ 3
d = “string”+false	⇒ “stringfalse”
[]==0	⇒ true
10-”1”	⇒ 9

Statements

- const firstName = “Marian”; - constant
- let x = 2, y; - variable (dont use var)
- y = 3*x + 5 - 2;
- let fullName = firstName + “ “ + “Pop”;
- if
- for
- switch
- while
- try{..} catch(exception) {...}

Strings

charAt(index): return character from index
concat(str): concatenate “str” to this string
includes(str): searches “str” in this string
startsWith(str), endsWith(str): check if this string starts/ends with
indexOf(char): index of char in this string
match(regex): check if regular expression matches this string
replace(what, replaceWith): replace in this string
search(str): search string
slice(beginIndex, endIndex): extract subsection of this string
split(separator): return an array of strings by splitting this string

Collections

- let myArray = new Array(2,3,4) || Array(”a”,”b”, “denis”) || [1,2,”castron”]
- myArray[25]=10; || ⇒ myArray.length = 26 (not 4, if there are only 4 elements)

Array Methods

nconcat() joins two arrays and returns a new array

njoin(delimiter = ',') joins all elements of an array into a string

npush() adds one or more elements to the end of an array and returns the resulting length of the array.

npop() removes the last element from an array and returns that element

nshift() removes the first element from an array and returns that element

nslice(startIndex, uptoIndex) extracts a section of an array and returns a new array.

nsplice(index, countToRemove, addElement1, addElement2, ...) removes elements from an array and (optionally) replaces them. It returns the items which were removed from the array

nreverse() transposes the elements of an array, in place: the first array element becomes the last and the last becomes the first. It returns a reference to the array

nsort() sorts the elements of an array in place, and returns a reference to the array

nindexOf(searchElement[, fromIndex]) searches the array for searchElement and returns the index of the first match

nforEach(callback[, thisObject]) executes callback on every array item and returns undefined.

nmap(callback[, thisObject]) returns a new array of the return value from executing callback on every array item

eval

- eval("2+3") ⇒ 5
- eval("let x=5; console.log(x)") ⇒ creates x , assigns 5 and console logs it

Functions

function *name_fct*(*parameters*, *arguments*) { *... statements ...* }

- no return ⇒ returns undefined
- function myF (x,y=0, ...restArgs){ // y is a default parameter
  ...}
  myF(1,2,3,4,5,6) ⇒ restArgs = [3,4,5,6]
- you can assign functions to a variable
  let myFunction = function square(x) {return x*x;}
  console.log(myFunction, [1,2,3,4]) ⇒ 1 4 9 16
- create with:
  new Function('par1', 'par2', return 'par1'+'par2');

Arrow functions

(param1, param2, ...) ⇒ {statements}

Classes and Objects

Create Objects

- objectName = {property1:value1, property2:value2,..., propertyN:valueN}
- function Thing(x, y, z) { this.prop1=x; this.prop2=y; this.prop3=z; this.method1=print;}
  ob = new Thing(a, b, c);
- var person = new Object(); person.name="Forest"; person.age=25;
- objects are deleted using "delete objectName"
- properties are accessible by obj.property or obj[index_property] or obj["property"]
- new properties can be added to object on run-time: obj.newProp=val
- EXAMPLE:
  function sayHi () {
  console.log("This is student "+this.firstName+"   "
  +this.lastName);
  }
  function Student (firstName, lastName, year) {
  this.firstName = firstName;

```
this.lastName = lastName;
this.year = year;
this.sayHi = sayHi;
}
var stud = new Student(“Gulin”, “Tudor”, 2);
stud.sayHi();
```

- There is a new syntax (but it’s just syntactic sugar, it does the same shit)

  ```
  class Student {
      constructor(firstName, lastName) {
          this.firstName = firstName;
          this.lastName = lastName;
          this.grades = [ ];
          }
        get studyYear() { return this.year; }              // getter
        get specialization()  { return this.spec; }        // getter
        set studyYear(year) { this.year = year; }          // setter
        set specialization(spec)  { this.spec = spec; }    // setter
        // method:
        addGrade(course, grade) { this.grades[course] = grade; }
        // static method:
        static sayHi(text) {  console.log("This is a student.", text);
      }
  var stud = new Student(“Gulin”, “Tudor”);
  ```

  - stud.studyYear;
  - stud.addGrade(”OS”, 10);

Inheritance

- class Person {

  ```
          constructor(firstName, lastName) {
          this.firstName = firstName;
          this.lastName = lastName;
          }
          sayHi(text) {  console.log("This is " + this.firstName + " " + this.lastName + ". " + text);  }
    }
  class Student extends Person {            // inheritance
          constructor(firstName, lastName) {
          super(firstName, lastName);    // calling constructor from base class
          this.grades = [ ];
          }
          sayHi(text) {

  super.sayHi();         // calling method from base class
          console.log("I'm also a student");
          }
  }
  ```

Template liberals

var name = “forest”;
var str = ‘
`this is ${name}`’;   // variable replacement
var str1 = ‘ this is a
multiline

string'
`;`

`var str2 = `'do the sum ${1+2+3}`; // computing arithmetic   expression

THE SPREAD OPERATOR (...)

var a = [1, 2, 3];
var b = [...a, 4, 5, 6]; // b = [1,2,3,4,5,6]
var c = [...a] ; // array copy
var obj = { prop1: 1; prop2: "2"; }
var objcopy = { ...obj }; // object cloning
var str = "hello";
var helloarray = [...str]; // helloarray = ['h','e','l','l','o']
// calling a function with an array parameter:
const f = (arg1, arg2) => {}
const a = [1, 2]
f(...a)

Destructing

a = [1, 2, 3, 4, 5, 6];
[first, ,third]=a; // first=1 and third=3

Strict mode

- introduces some restrictions to the js engine → better
- how to apply it: use strict; at the start of your script /js file

Exporting

- you have to export variables, function etc.
- export { symbol1, symbol2, myFunctionName } - at the end of your .js file
- you can 'export' multiple times, only one 'export default'

Importing

import {symbol1, function1 } from '.../.../.../myFile.js'

or

import * from ...

- in html you just <script type="module" src="main.js"></script>
- you can import as a module:
  import * as Module1 from ...
  - you use them like this:
    Module1.myFunction();

EVENTS

- Javascript is an event based language
- EVENTS:
  - click
  - key pressed

    - element loosing focus
    - etc.
- Event handlers are associated to a tag:
    1. <TAG eventHandler="Javascript code">
    2. <script type="text/javascript">
       function evHandle(x) { ... }
       </script>
       <TAG eventHandler="evHandle(this)">
    3. <script type="text/javascript">
       obj.eventHandler="Javascript code";
       </script>

Pop-Ups

- alert("...text...") : displays text and the Ok button
- confirm("... text...") : displays text and returns true if the Ok button is clicked and false if the Cancel button is clicked
- prompt("text", "default value"): the user can enter an input value and then click Ok (return the value) or Cancel (return null)

DOM (Document Object Model)

DOM Browser Objects

Window object
Navigator object
Screen object
History object
Location object

PHP, Ajax and JSON

What is PHP

- Hypertext Preprocessor
- server-side programming language
- free, open-source, runs on Apache and IIS

PHP code in HTML files

1. <?php ... code ... ?>
2. <script language="php">
   ...code...
   </script>
3. <? ...code... ?>
   <?= expression =?>
4. <% ... code ... %> = ASP-style tags

Variables in PHP

- variables are not bound to a specific type (loosely-typed language)
- a varName is prededed by “$”
- Example:
  $text = “marian”;
  $no = 4;
  $b = TRUE;
  $no1=5.6l;
  $vect=array(1,2,3,4,5);
- $x1 = &$x; → $x1 is an allias for $x

Global variables

$a=2, $b=5;

function myFunction(){

global $a, $b; → use global variables, that arent defined in your function

$c = $a+$b; → local scope, just in myFunction

$c = $GLOBALS[‘a’] + $GLOBALS[‘b’]; → $GLOBALS is an array for all global variables

}

Superglobal Variables

superglobal variables are available in all scopes throughout the script; no need to be declared global in a local function; were introduced in PHP 4

$GLOBALS – contains references to all variables defined in the global scope of the script
$_SERVER - array containing information such as headers, paths, and script locations; built by the web server
$_GET - array of variables passed to the current script via the URL parameters
$_POST - array of variables passed to the current script via the HTTP POST method
$_FILES - array of items uploaded to the current script via the HTTP POST method
$_COOKIE - array of variables passed to the current script via HTTP Cookies
$_SESSION - array containing session variables available to the current script
$_REQUEST - array that by default contains the contents of $_GET, $_POST and $_COOKIE
$_ENV - array of variables passed to the current script via the environment method

- to access a global variable u have to say “global $myGlobalVariable”
  and then you can use it:
  echo $myGlobalVariable
- to access a superglobal variable u can do it directtly:
  echo $mySuperGlobalVariable

$GLOBALS

- if you have
  $a = ‘Destroy’
  function myFunc(){
  $a = ‘FlowerPower’
  echo $GLOBALS[’a’] → Destroy
  echo $a → FlowerPower
  }

$_SERVER

keys:
‘PHP_SELF’ – the filename currently executed

‘SERVER_ADDR’ – the IP address of the server
‘SERVER_PROTOCOL’ – name and version of the protocol via which the page is requested; HTTP/1.1
‘REQUEST_METHOD’ – the request method
‘QUERY_STRING’ – the query string
‘DOCUMENT_ROOT’ – the document root under which the current script is executed
‘REMOTE_ADDR’ – the client IP address
‘REMOTE_PORT’ – the client port
‘HTTP_ACCEPT’ – the HTTP accept field of the HTTP protocol
etc.

$_GET

- an html example
  <form action="welcome.php" method="get">Name: <input type="text" name="fname" />Age: <input type="text" name="age" /><input type="submit" />
  </form>
- after submit, the URL is:

  http://www.w3schools.com/welcome.php?fname=Peter&age=37
- the ‘welcome.php’ file:
  Welcome <?php echo $_GET["fname"]; ?>.<br />
  You are <?php echo $_GET["age"]; ?> years old!

$_POST

- an html example
  <form action="welcome.php" method=“post">Name: <input type="text" name="fname" />Age: <input type="text" name="age" /><input type="submit" />
  </form>
- after submit, the URL is:

  http://www.w3schools.com/welcome.php
- the ‘welcome.php’ file:
  Welcome <?php echo $_POST["fname"]; ?>.<br />
  You are <?php echo $_POST["age"]; ?> years old!

FUNCTIONS

- function functionName ($para1, $para2, ...){
  ...code...
  }
- Example:
  <?php
  function add($x,$y) {$total=$x+$y;return $total;
  }
  echo "1 + 16 = " . add(1,16);
  ?>

CLASSES

Create a class + a class object

class SimpleClass
{   // property declaration   public $var = 'a default value';   // method declaration   public function displayVar() {   echo

```
>var;    }
}
```

```
$instance = new SimpleClass();
```

Extend a class

```
class ExtendClass extends SimpleClass
{    // Redefine the parent method    function displayVar() {        echo "Extending class\n";        parent::displayVar();    }
}
```

```
$extended = new ExtendClass();
$extended→displayVar();
```

Random

```
function Thing(x, y, z) { this.prop1=x;  this.prop2=y;  this.prop3=z;  this.method1=print;}
ob = new Thing(a, b, c);
```