1. Ce este php ?

PHP este un limbaj de programare realizat în principal pentru realizarea site-urilor web dinamice.
Php: hypertext preprocessor

```
<!DOCTYPE HTML>
<html>
<head>
<title>Exemplu PHP</title>
</head>
<body>
<?php
echo "Primul exemplu de script PHP!";
?>
</body>
</html>
```

2. **echo** - echo afișează toți parametrii primiți ca parametri (echo nu este o funcție). Construcția echo are și o formă scurtă <?=$var ?>. Este folosita pentru mai multe argumente.

3. **print** afișează un șir de caractere primit ca parametru. Valoarea întoarsă este întotdeauna 1. Este folosita pentru un singur argument.

4. Domeniile principale de utilizare ale limbajului PHP:

- server side scripting (scripting la nivel de server) - domeniul de bază al utilizării limbajului PHP
- shell scripting (scripting în linia de comandă)
- aplicațiilor desktop

PHP se poate utiliza cu programarea procedurală sau cu programarea orientată pe obiecte (POO), sau cu o combinație a acestora.

PHP se poate folosi nu doar la afișarea HTML ci și la afișarea de imagini, fișiere PDF. PHP poate genera automat aceste fișiere și le poate salva în sistemul de fișiere.

Una dintre cele mai puternice și semnificative facilități ale PHP este susținerea unui larg domeniu de baze de date.

5. Comentarii

```
/* comentariu multi rand stil C */
// comentariu de un singur rand stil C++
# comentariu de un singur rand stil Unix shell
```

6. Tipuri de date

-Nu este nevoie de declarare
-boolean(true/false)
-integer
-reale : float,double
-string
-notatia heredoc (<<<)
-notatia newdoc (<<<)
-resource
-null
$variabila

7. Variabile(case sensitive)

<nume variabila>=<valoare>;
-resource
-null
Varibilele superglobale sunt disponibile oriunde in script(ex: session,get,post,globals):
$_GET’

8. Constante

pot fi considerate variabile ale căror valori sunt fixate și nu mai pot fi alterate pe parcursul programului

```
const CONSTANT = 'o valoare constanta';
```

9. Expresii în PHP

Expresiile sunt combinații valide de operanzi și operatori. Operanzii pot fi variabile, constante literali sau alte expresii.

10. Operatori

-aritmetici (+, -, *, /, %) sunt cei cunoscuți din limbajele C/C++, Java.
-de asignare ("=" sau "+=")
Exemplu de utilizare operatorilor de asignare "=" , "+=" și concatenare ".="
- **la nivel de bit** (&,|,^, ~, <<, >>)
- **relaționali** (ca noutate se poate menționa aici „===" care are ca rezultat constanta TRUE dacă operanzii au valori identice și în plus sunt de același tip)
- terțiar ?:
Un alt operator condițional este operatorul terțiar "? :". Expresia
-de control al erorilor („@")
Există un singur operator „@" care inhibă erorile ce pot apărea în cadrul expresiei pe care o prefixează. **-logici** (and sau "&&", or sau "||", xor, "!")
-pentru șiruri de caractere
-de concatenare "." asemănător cu operatorul de concatenare din limbajul Perl.

11. Instrucțiunile limbajului PHP

- de atribuire (de forma $variabilă = expresie)
-de test (if și switch)
If: <?php
if (expresie)
instrucţiune
?
-repetitive (while, do, for, foreach)
For:
for (expr1; expr2; expr3) {
instrucțiune
};
For each:
foreach (expresie_array as
$value) instrucțiune
-de control (break, continue, return).

Instrucțiunea else

<?php
$a=5;$b=10;
if ($a > $b)
{ echo "a mai mare decat
b";} else
{ echo "a NU este mai mare decat
b";} ?>

Instrucțiunea elseif

Instrucțiunea elseif este o combinație de if și else. Precum else, extinde o instrucțiune if pentru a executa o altă instrucțiune diferită în cazul în care expresia logică originală if este evaluată la FALSE. Totuşi, spre deosebire de else, ramura alternativă va fi executată dacă şi numai dacă expresia condițională elseif este evaluată la TRUE.

12. Tablouri

În PHP există trei tipuri de tablouri:
- tablouri indexate (tablouri cu indice numeric)
- tablouri asociative
- tablouri multidimensionale

13. Funcții definite de utilizator

O funcție poate fi definită prin utilizarea următoarei
sintaxe: function functionName() {
code to be executed;
}

Orice cod PHP valid poate fi inclus în interiorul unei funcții, chiar și alte funcții și definiții de obiecte (class).
Variabilele statice isi pastreaza valoarea de la un apel la altul al fucntiei($$ numele variabilei este pastrat intr-o alta variabila)

Valorile returnate de funcții

Valorile sunt returnate prin utilizarea instrucțiunii opționale return. Orice tip de date poate fi returnat, incluzând liste și obiecte. Această instrucțiune determină funcția să-și termine execuția sa imediat și să cedeze controlul înapoi liniei de program de unde ea a fost apelată.
Exemplu - utilizarea instrucțiunii return

```
<?php
function square($num)

{
return $num * $num;
}
echo square(4); // Afisează '16'.
?>
```

Funcții pentru determinarea tipului variabilelor

Există funcții care determină tipul variabilelor.
Exemplu: verificăm daca variabila $a este de tip întreg

```
<?php
$a=19;
if (is_integer($a)) {
echo '$a este intreg';
} else {
echo '$a nu este intreg';
}
?>
```

14. Variabile predefinite (superglobals)

Variabilele PHP superglobale (Superglobals) sunt variabile disponibile oriunde în script:
$GLOBALS = conține referințe către toate variabilele care sunt disponibile în scop global scriptului - pot fi accesate toate variabilele globale care sunt accesibile script-ului PHP curent
$_SERVER = variabile furnizate scriptului de către serverul web - conține o serie de variabile ale căror valori sunt setate de server-ul web; majoritatea valorilor variabilelor din acest vector depind de mediul de execuție al script-ului curent.
$_GET și $_POST conțin variabile primite de script prin intermediul unor transferuri care folosesc metodele HTTP get, respectiv post. De exemplu, prin intermediul acestor vectori, pot fi accesate valorile câmpurilor dintr-un formular care a fost completat și transmis folosind una dintre cele două metode.
$_GET variabile furnizate scriptului via HTTP GET (provin dintr-un formular în care method="GET")
$_POST variabile furnizate scriptului via HTTP POST (provin dintr-un formular în care method="POST") $_COOKIE conține valorile variabilelor care cuprind informații referitoare la cookie-urile păstrate pe calculatorul utilizatorului ce accesează pagina Web (variabile furnizate scriptului via HTTP cookies). $_FILES conține variabile primite de script prin intermediul încărcărilor de fișiere prin metoda post. $_ENV conține variabile disponibile prin intermediul mediului în care este executat.
$_REQUEST conține variabile disponibile prin intermediul oricărui tip de mecanism cu ajutorul căruia utilizatorul poate introduce date.
$_SESSION conține variabile care corespund sesiunii curente a script-ului.

15. **O clasa** este o colecție de variabile si funcții care operează asupra variabilelor respective.Sintaxa folosită pentru declararea unei clase in PHP este:
    Class ‚nume' {lista de proprietati}

Pentru a crea o instanță a unei clase, se utilizează cuvântul cheie new.

16. Propeietati

- **proprietățile non-statice** pot fi accesate prin utilizarea operatorului -> $this -> proprietate (în cazul în care proprietate este numele proprietății).
-Proprietățile statice sunt accesate folosind operatorul ::
`nume clasa`::$proprietate

17. Constante de clasă

Constantele diferă de variabile normale, nu se utilizează simbolul $ pentru declarare și utilizare. Vizibilitatea implicită ale constantelor de clasă este publică. Valoarea trebuie să fie o expresie constantă, nu o variabilă, nici o proprietate și nici un apel de funcție.

18. Constructori

Sintaxa unui constructor:
void __construct ([mixed $args = "" [, $...]])
PHP permite programatorilor să definească constructori pentru clase. Clasele care au definit un constructor vor apela această metodă la fiecare obiect nou creat – este folosit pentru inițializarea obiectului înainte de a fi folosit.

19. Destructori

Sintaxa unui destructor:
void __destruct (void)
PHP5 introduce conceptul de destructor al unui obiect similar cu cel regăsit în alte limbaje de programare orientate pe obiecte (C++). Metoda de distrugere va fi apelată imediat ce nu mai sunt referințe către un anumit obiect sau oricând în timpul secvenței de deconectare.

20. Vizibilitatea

Vizibilitatea unei proprietăți sau a unei metode poate fi definită prefixând declarația cu unul din cuvintele cheie: **public, protected sau private**.

21. Moștenirea

Când se extinde o clasă, subclasa moștenește toate metodele publice și protejate din clasa părinte. Cu excepția cazului unei clase suprascrie aceste metode, ele își vor păstra funcționalitatea lor originală.

22. Operatorul de rezoluție de scop (: :)

Domeniul de aplicare al operatorului de rezoluție de scop permite accesul la proprietăți sau metode statice, constante și proprietăți sau metode supraîncărcate ale unei clase. Când se face referire la aceste elemente din afara definiției clasei, se folosește numele clasei.

23. Metode si proprietati statice

Proprietățile si metodele statice nu pot fi accesate prin intermediul obiectului folosind operatorul ->

24. Clase abstracte si clase anonime

Clasele definite ca abstracte nu pot fi instanțiate și orice clasă care conține cel puțin o metodă abstractă trebuie să fie și ea abstractă. Metodele abstracte declară doar semnătura unei metode și nu pot defini o implementare concretă.
Clasele anonime sunt utile atunci când sunt e necesar ca obiecte simple să fie create

25. Interfețe

Interfețele permit specificarea metodelor pe care o clasă trebuie să le implementeze, fără a fi nevoie să se definească aceste metode.
Interfețe sunt definite în același mod ca și o clasă, dar folosind cuvântul cheie interface și fără a defini conținutul metodelor.

26. Overloading (supraîncărcarea)

Overloading în PHP oferă mijloace de a crea dinamic proprietăți și metode. Aceste entități dinamice sunt procesate prin metode magice.

Metodele tip overloading sunt invocate atunci când se interacționează cu proprietăți sau metode care nu au fost declarate sau nu sunt vizibile în scopul curent.
Toate metodele tip overloading trebuie definite ca publice.

27. Iterarea proprietăților unui obiect

PHP 5 oferă o modalitate pentru iterarea proprietăților printr-o listă de elemente, de exemplu cu o structură foreach. În mod implicit, toate proprietățile vizibile sunt folosite pentru repetare.

28. Compararea obiectelor

Atunci când se utilizează operatorul comparație (==), variabilele obiect sunt comparate într-un mod simplu, și anume: două instanțe obiect sunt egale dacă au aceleași atribute și valori (valori sunt comparate cu ==), și sunt instanțe din aceeași clasă.

29. Final

PHP 5 introduce cuvântul cheie final, care împiedică subclasele să suprascrie metodelor prin prefixarea definiției utilizând cuvântul cheie final.

30. PHP Data Objects (PDO)

PHP Data Objects (PDO) este o extensie PHP ce definește o interfață ușoară, consistentă pentru accesarea bazelor de date în PHP.
Script-urile care folosesc interfața PDO pentru conectare la baza de date **efectuează în general următoarele operații**:

- conectare la serverul bazei de date, prin crearea unei instanțe din clasa PDO, obținând un obiect pentru lucru cu acea bază de date.
- aplicare funcții specifice PDO pentru efectuarea interogărilor la baza de date.
- reținerea si prelucrarea datelor returnate.
- deconectarea de la server.

31. Conexiuni și gestionarea conexiunilor

Exemplu - conectarea la MySQL

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user,
$pass); ?>
```

Exemplu - conexiuni persistente

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass,
array(PDO::ATTR_PERSISTENT => true
));
?>
```

Pentru a iniția o tranzacție se utilizează metoda PDO :: beginTransaction (). În cazul în care driver-ul bazei de date nu acceptă tranzacții, va fi generată o excepție PDOException. Pentru a finaliza o tranzacție se utilizează PDO:: commit () sau PDO :: rollback () pentru a anula o tranzacție.

32. Conexiunile persistente

nu sunt închise la sfârșitul script-ului, ci sunt memorate în cache și reutilizate atunci când un alt script cere o conexiune, folosind aceleași informații de conectare

33. Tranzacții și auto-commit

Tranzacțiile oferă 4 caracteristici majore (ACID):
- atomicitate,
- coerență,
- izolare
- durabilitate.

34. Prepared statements

Acestea pot fi privite ca șabloane compilate pentru comenzi SQL pe care o aplicație le rulează și care pot fi personalizate utilizând parametri variabili. Prepared statements oferă două avantaje majore:

- interogarea are nevoie doar să fie analizată (sau pregătită), o singură dată, dar poate fi executată de mai multe ori cu aceeași parametri sau parametri diferiți.
- parametrii pentru prepared statements nu trebuie încadrați în apostrofuri; driver-ul se ocupă în mod automat de acest lucru.

35. Utilizarea obiectelor mari (LOB)

Obiectele mari pot fi de natură textuală sau binară (mai mari de 4KB).

36. Erori

PDO oferă o gamă de 3 strategii diferite pentru gestiunea erorilor:
- PDO::ERRMODE_SILENT : Acesta este modul implicit
- PDO::ERRMODE_WARNING În plus față de setarea codului de eroare, PDO generează un mesaj E_WARNING(utila in timpul depanarii,testarii)
- PDO::ERRMODE_EXCEPTION În plus față de setarea codului de eroare, PDO va genera o excepție PDOException și vor fi setate proprietățile excepției pentru a reflecta codul de eroare și informațiile despre eroare.

37. Clasa PDO

Clasa PDO reprezintă o conexiune între PHP și un server de baze de date.
Exemple :
Metoda PDO :: beginTransaction
PDO::commit
PDO::rollBack
PDO::errorCode
PDO :: errorInfo
PDO :: getAttribute

38. Clasa PDOStatement

reprezintă un prepared statement (o declarație pregătită) și, după ce instrucțiunea este executată, un result set (set de rezultate) asociat.

39. Utilizarea sesiunilor

Sesiunile reprezintă un mod simplu de a stoca date pentru utilizatori individuali folosind unui id unic de sesiune. Acest lucru poate fi folosit pentru a păstra informații între cereri diferite.
Exemple:
-session_start();
-session_id();
-session_destroy();
-session_status();

40. Headere HTTP

Headerele HTTP sunt elemente prin care browser-ul și serverul web comunică în fundal pentru a afișa o pagină web în bune condiții.
Există două tipuri de headere HTTP: cele emise de browser (headere de cerere – request headers) și cele emise de server (headere de raspuns – response headers)

41. Request headers

De fiecare dată când un utilizator accesează o pagina web, browser-ul trimite către server cantități mici de date, sub formă de request headers (sau antete de cerere). Aceste antete cuprind detalii despre pagina care a fost solicitată, modul de transfer a ei, precum și informații despre capabilitățile browser-ului.

42. Response headers

În răspuns la request headers primite de la browsere, serverele web trimit înapoi două tipuri de informație:
- headere de răspuns (response headers)
- conținutul efectiv al paginii solicitate (conținut ce poate fi construit/modificat prin intermediul PHP)

43. Spații de nume în PHP (namespaces)

Spațiile de nume reprezintă un mod de încapsulare a unor elemente.

44. Definirea spațiilor de nume

Un spațiu de nume afectează: clase (inclusiv clase abstracte și trăsături - traits), interfețe, funcții și constante.Spațiile de nume sunt declarate folosind cuvântul cheie namespace.
Se pot declara mai multe spații de nume în același fișier. Există două sintaxe permise.
Un nume de clasă poate fi menționat în trei moduri:
- nume de clasă necalificat sau un nume de clasă neprefixat
- nume calificat sau un nume de clasă prefixat
- numele complet calificat sau un nume de prefixat cu operatorul prefix la nivel global

45. Cookie-uri

Cookie-rile reprezintă porțiuni de informații (stocate sub formă de fișiere de mici dimensiuni) ce se află pe calculatorul utilizatorului și care sunt create și folosite de către browser în comunicarea cu serverul web.

46. Cuvântul cheie namespace și constanta __NAMESPACE__

PHP suportă două moduri de accesare a elementelor din cadrul spațiului de nume actual, constanta magică __NAMESPACE__ și cuvântul cheie namespace.
Valoarea __NAMESPACE__ este un șir de caractere care conține numele spațiului de nume curent. În codul global, fără spațiu de nume, aceasta conține un șir gol.

47. Traits

Traits este un mecanism de reutilizare a codului în limbajul PHP care implementează moștenirea simplă. Poate defini proprieteati,metode statice,metode abstracte

48. Precedența

Un membru moștenit dintr-o clasă de bază este înlocuit de un membru inserat printr-un Trait.

49. Traits-uri multiple

pot fi inserate într-o clasă prin enumerarea lor în declarația use, separate prin virgula.

50. Rezolvarea conflictului

În cazul în care două Traits introduc o metodă cu același nume, se produce o eroare fatală în cazul în care conflictul nu este rezolvată în mod explicit. Pentru a rezolva conflictele de nume în Traits utilizate în aceeași clasă, trebuie utilizat operatorul insteadof pentru a alege exact una dintre metodele aflate în conflict.

51. Schimbarea vizibilității metodei

Cu ajutorul sintaxei as, se poate ajusta vizibilitatea metodei din clasa expozantă

52. Gestiunea erorilor în PHP

se trimite browser-ului un mesaj de eroare cu numele fișierului, numărul liniei și un mesaj care descrie eroarea.

53. Gestiunea erorilor folisind functia die()

Codul este mai eficient deoarece folosește un mecanism simplu de gestiune a erorilor prin care se oprește scriptul după eroare.

54. Setarea gestionarului de tratare a erorilor

PHP are un gestionar implicit de tratare a erorilor. PHP oferă posibilitatea de schimbare a gestionarului implicit de tratare a erorilor și de a fi aplicat doar pentru unele erorișI în acest fel script-ul poate manipula diferite erori în moduri diferite.

55. Declanșarea unei erori

Declanșarea unei erori se poate face în PHP utilizând funcția trigger_error ()

56. Logarea erorilor

În mod implicit, PHP trimite un jurnal de erori la sistemul de logare al serverului sau într-un fișier, în funcție de modul în care este configurată proprietatea error_log în fișierul php.ini. Prin utilizarea funcției error_log () se pot trimite jurnalele de eroare într-un fișier specificat sau la o destinație la distanță.

57. Utilizarea de bază a excepțiilor

Atunci când o excepție este aruncată, codul care urmează nu va mai fi executat, iar PHP-ul va încerca să găsească un bloc catch. În cazul în care o excepție nu este prinsă se generează o eroare fatală cu un mesaj Uncaught Exception.

58. Try, throw and catch

Try –O funcție care poate genera o excepție trebuie să fie într-un bloc try. Dacă excepția nu se declanșează, codul va continua în mod normal. Dacă excepția se declanșează, o excepție este generată.
Throw - Acesta este modul în care declanșează o excepție. Fiecare throw trebuie să aibă cel puțin un block catch
Catch - Un bloc care preia o excepție și creează un obiect care conține informația despre excepție

59. Excepții multiple

Este posibil ca un script să utilizeze mai multe excepții pentru a verifica mai multe condiții. Se pot utiliza mai multe blocuri if..else, un switch sau se pot imbrica mai multe excepții

60. Rearuncarea excepțiilor

Uneori când se aruncă o excepție este posibil să se arunce o excepție pentru a doua oară într-un bloc de tip catch.

61. Setarea unui gestionar pentru excepții de nivel superior

Funcția set_exception_handler () stabilește o funcție definită de utilizator care să se ocupe de toate excepțiile neînregistrate.

62. Fisiere in PHP

Fișiere în PHP PHP oferă facilități avansate pentru lucrul cu fișiere.
Astfel se pot crea, modifica, manipula și șterge fișiere. Fișierele sunt colecții de informații stocate pe un dispozitiv de stocare (hard-disk, CD, etc).
În funcție de formatul datelor conținute fișierele se împart în două categorii: fișiere text(conțin informația stocată sub forma unui șir de caractere (eventual, pe mai multe linii).) și fișiere binare(stochează informația brut, fără prelucrări exact așa cum apare ea în memorie).

63. Folosirea fișierelor în PHP

În orice limbaj de programare precum și în limbajul PHP când se lucrează cu fișiere trebuie efectuate următoarele operații:
- deschiderea fișierului
- citirea din fișier/scrierea în fișier
- închiderea fișierului

64. Funcția fopen

deschide fișierul sau URL specificat
fopen () leagă o resursă specificată prin numele de fișier la un flux

65. Protocoale și wrappere suportate

PHP vine cu multe wrappere încorporate pentru diverse protocoale de tip URL pentru a fi utilizate cu funcțiile sistemului de fișiere, cum ar fi fopen (), copy (), file_exists () și filesize ().

66. Funcția fread string fread

-permite citirea a unui număr specificat de octeți din fișier.

67. Funcția fwrite

-scrie șirul de caractere în fluxul indicat de pointerul de fișier handle.

68. Instrucțiuni pentru fișiere text

Pentru simplificarea codului de citire/scriere a unui fișier în situații generale, PHP oferă câteva funcții foarte convenabile: - file_get_contents () = citește întregul fișier într-un șir de caractere
-file_put_contents() = scrie un șir de caractere într-un fișier.
- file()=citește întregul fișier într-o matrice

69. Manipularea fișierelor în PHP

-creare
-mutare,
-copiere,
-modificare atribute, etc
// copiere fisier : copy('sursa.txt', 'destinatie.txt');
// stergere fisier : unlink('fisier.txt');
// redenumire sau mutare : rename('vechi.txt', 'nou.txt');

70. Directoare în PHP

Manipularea directoarelor (folderelor) folosind PHP se face la fel de ușor ca în cazul fișierelor. Majoritatea funcțiilor folosite pentru fișiere se pot aplica și la directoare (de exemplu copy, rename, is_file, etc)

71. Upload de fișiere

Încărcarea (copierea) unui fișier de pe calculatorul personal pe server (upload de fișiere) se face prin intermediul formularelor. Pentru aceasta facilitate se folosește un formular puțin diferit: are un atribut special (enc-type) și metoda POST.

72. Upload de fișiere multiple

Când este nevoie să se încarce mai multe fișiere în același timp, formularul va conține mai multe elemente de tip INPUT FILE, denumite sub forma unui vector (array)

73. Filtrarea datelor

se face prin validare și purificare. Acest lucru este util mai ales atunci când sursa de date conține date necunoscute.(tipuri: de validare, de purificare)
-filter_var: — filtrează o variabilă folosind filtrul indicat.

74. JSON JSON (JavaScript Object Notation)

este un format de interschimbare a datelor. Este ușor de citit de oameni.
Este ușor de analizat și generat de către calculator.
este un format de text, care este complet independent de limbaj

75. Funcții JSON

json_decode — convertește un șir reprezentat în format JSON într-o variabilă
json_encode — întoarce o reprezentare JSON pentru valoarea dată
json_last_error_msg — întoarce mesajul de eroare a ultimului apel
json_encode() sau json_decode() json_last_error — întoarce ultima eroare care a apărut

76. XML DOM Parsers

Analizorul XML DOM face posibilă procesarea documentelor XML în PHP. Analizorul XML DOM este un analizor bazat pe arbori.

77. XML Expat Parser

Procesorul Expat este inclus în PHP de la versiunea 3. Analiza XML este bazată pe evenimente, fiecare tip de nod al arborelui asociat documentului XML declanșând un anumit eveniment care va trebui tratat de o funcție definită de programator.

78. MVC (Model View Controller)

este un design pattern software ce permite separarea datelor (model) de interfața utilizator (view).

79. XML DOM

Cu ajutorul funcțiilor DOM PHP se poate citi conținut XML existent și se pot crea documente noi XML. În exemplul următor se creează un document XML și se salvează într-un fișier.
Clase utile :

- Clasa DOMDocument

- Clasa DOMElement

- Clasa DOMNode

- Clasa DOMNodeList

- Clasa DOMAttr

80. AJAX (Asynchronous JavaScript and XML)

AJAX este o tehnică de programare pentru crearea de aplicații web interactive, Intenția este să facă paginile web să devină mai rapide prin schimbul în fundal al unor cantități mici de date cu serverul, astfel încât să nu fie nevoie ca pagina să fie reîncărcată la fiecare acțiune a utilizatorului.

81. Obiectul XMLHttpRequest

Scopul obiectului XMLHttpRequest este de a permite JavaScript să formuleze cereri HTTP și să le trimită la server, dând astfel posibilitatea comunicării cu serverul și afișarea datelor primite fără a fi necesară reîncărcarea paginii

82. Proprietati si metode

Proprietăți
- onreadystatechange - Folosit ca un "event handler", determină care eveniment va fi apelat la schimbarea stării "readyState".
- readyState - un număr intre 0 și 4 care reprezintă starea cererii
- responseText - returnează răspunsul primit de la server, în format șir text (string).
- responseXML - returnează răspunsul primit de la server în format XML.
- status - codul de stare HTTP al răspunsului de la server, în format numeric (200 pt. răspuns corect, 404 pt. "Negăsit", 500 pt. o eroare de server, etc.).
- statusText - codul de stare HTTP al răspunsului de la server, în format text ("OK", "Not found", "Internal Server Error", etc.).

Metode
- abort() - Anulează cererea curenta
- getAllResponseHeaders() - Returnează sub forma de șir toate Header-ele HTTP primite ca răspuns
- getResponseHeader(x) - Returnează valoarea Header-ului 'x' specificat
- open(method, URL, flag) - Creează cererea care va fi trimisă.
- send(content) - Trimite cererea curentă la server.
- setRequestHeader('eticheta', 'valoare') - Permite stabilirea de Headere care să fie transmise la server împreună cu cererea creata cu "open()"

83. Utilizare Ajax cu metoda HTTP POST și PHP

Utilizare Ajax cu metoda HTTP POST și PHP Un alt mod de trimitere a unor date dintr-o pagina web la server este prin metoda POST. Cu Ajax, cererea pentru trimiterea datelor cu POST se face tot prin metoda "open()" a obiectului "XMLHttpRequest", sintaxa acesteia fiind open("POST", url, flag), unde "POST" este metoda de transfer, "url" este adresa fișierului PHP la care va fi transmisă cererea, iar "flag" este o valoare booleană true sau false.

84. Utilizare Ajax cu date preluate din formulare

În general, datele adăugate în casetele dintr-un formular sunt trimise prin POST la un script de pe server specificat în atributul "action" al etichetei. Cu Ajax se pot trimite datele preluate din formulare atât prin metoda GET cât și prin metoda POST.

85. AJAX cu XML

În unele cazuri, răspunsul primit de Ajax (datele transmise de la scriptul de pe server) poate fi conținutul unui document XML. Această metodă de lucru fiind frecvent utilizată în aplicațiile API de transfer a datelor de la un server la altul (de la un server extern la site-ul care a solicitat transferul)

86. AJAX cu JSON

JSON (JavaScript Object Notation) este un format simplu de structurare a datelor în format text, folosit pentru schimbul de informații și poate fi folosit ca o alternativa mai ușoară la XML.

87. Servicii web

Un serviciu web este un serviciu pus la dispoziție utilizatorilor pe Internet.

88. WSDL (Web Services Description Language)

Web Services Description Language sau Limbajul de Descriere a Serviciilor Web este un limbaj bazat pe XML ce oferă un model de descriere a serviciilor web

89. Clasa SoapServer

oferă un server pentru protocoalele SOAP 1.1 și SOAP 1.2. Poate fi utilizat cu sau fără o descriere a serviciului WSDL.

90. Clasa SoapClient

oferă un client pentru servere SOAP 1.1, SOAP 1.2. Poate fi utilizat în modul WSDL sau non-WSDL.

91. REST

este acronimul de la Representational State Transfer și reprezintă un model arhitectural pentru crearea serviciilor web. REST descrie o arhitectură orientată pe resurse, aceasta fiind prezentată în detaliu prima dată în teza de doctorat a lui Roy Fielding.

92. CSS

înseamnă Cascading Style Sheets (foi de stil în cascadă). Stilurile definesc cum vor fi afișate elementele HTM