



Desenvolupament Web en Entorn Servidor

UD 2 - El llenguatge PHP

2n CFGS Desenvolupament Aplicacions Web





BLOC 1 - Desenvolupament Web en Entorn Servidor

UD1 - Architectures i eines de programació

UD2 - Introducció al llenguatge

UD3 - Programació orientada a objectes

UD4 - Accés a BD MySQL

UD5 - Gestió d'aplicacions



Objectius

Després d'acabar aquesta unitat, l'alumne hauria de ser capaç del:

- Manegar PHP en un entorn propi o en una aplicació HTML **d'una manera fluïda**



UD2 - El llenguatge PHP

1. Introducing PHP

- a. Basic syntax
- b. PHP embedded in HTML & 'echo' Statement
- c. Constants & Variables
- d. Operators
- e. Arrays
- f. Control Structures
 - i. 'if...elseif...else' Statement
 - ii. 'switch' Statement
 - iii. Loop Statements
 - iv. 'try ... catch ... finally' Statement
 - v. error_reporting(), Error_handler() Functions

2. Functions

- a. Procediments i funcions
- b. Els paràmetres i els valors de retorn
- c. Pas per valor o pas per referència



UD2 ... - Introduction

PHP is a widely used general-purpose interpreted language specially designed for web development that can be embedded within HTML code.

It generally runs on a web server, taking PHP code as its **input** and creating HTML pages as its **output**.

It can be deployed on most web servers and on almost all operating systems and platforms at no cost.



UD2 ... - Introduction

A PHP codification example:

```
<html>
  <body>
    <table>
      <?php
        for ($i = 1; $i <= 10; $i++) {
          echo "<tr><td>fila " . $i . "</td></tr>";
        }
      ?>
    </table>
  </body>
</html>
```



UD2 ... - Basic syntax

- A PHP script can be placed anywhere in the document
- A PHP script starts with `<?php` and ends with `?>`:

```
<?php
```

```
// PHP code goes here
```

```
?>
```

- PHP statements end with a semicolon (;)
- In PHP, keywords are not case-sensitive. However; all variable names are case-sensitive !
- The default file extension for PHP files is ".php".



UD2 ... - PHP embedded in HTML

- A PHP file normally contains HTML tags, and some PHP scripting code.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>My first PHP page</h1>

    <?php
      echo "Hello World !";
    ?>

  </body>
</html>
```




UD2 ... - PHP embedded in HTML

Los comentarios de PHP se pueden escribir de varias formas:

// Comentario de una línea

Comentario de una línea

/* Comentario de una o varias líneas */



UD2 ... - PHP embedded in HTML

Tenemos tres posibilidades a la hora de generar contenido en nuestros documentos PHP:

- `echoexpresión;`
- `print(expresión);`
- `<?=expresión ?>`

```
<p><?php echo "Este texto se mostrará en la página web." ?></p>
```

```
<p><?= "Este texto se mostrará en la página web." ?></p>
```

```
<p><?php print("Este texto se mostrará en la página web.") ?></p>
```



UD2 ... - Constants & Variables

Tipos de datos ([php-data-types](#)):

1. Escalares: boolean / integer / float / string ([php-string](#))
2. Compuestos: array / object
3. Especiales: resource (ignorar) / NULL

```
<?php
```

```
    $nl = NULL;
```

```
    echo $nl; //it will not give any output
```

```
?>
```



UD2 ... - Constants & Variables

Variables are the nominal representation of a RAM memory space allocated for a specific purpose. Variables in PHP are represented by a dollar sign (\$) followed by the name of the variable. The variable name is case-sensitive. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

```
<?php
```

```
$4site = 'not valid';           // not valid; starts with a number
$_4site = 'valid';             // valid; starts with an underscore
$var = 'not recommended';      // not recommended; but 'à ' is (Extended) ASCII 228.
$var = 'Bob';                  // not same as $Var
$Var = 'Joe';                  // not same as $var
$myVar = 'Tom';                // not same as $var
echo "$var, $Var";             // outputs "Bob, Joe"
```

```
?>
```



UD2 ... - Constants & Variables

PHP is a Loosely Typed Language, that means that:

- *PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.*

```
<?php
    echo 5 + "5 dias"; // → NO ERROR IS REPORTED
?>
```

- In PHP 7 and later, type declarations were added.

```
<?php declare(strict_types=1); // disables automatic type conversion
    echo 5 + "5"; // → ERROR IS REPORTED
?>
```



UD2 ... - Constants & Variables

Constants can be understood as variables without the possibility of being able to change their value during the execution of a program.

```
<?php
    DEFINE('PI','3.141592');
    $radius = 10;
    $area = 2 * PI * pow($radius, 2);

    echo $area;

?>
```



UD2 ... - Constants & Variables

Superglobals: super global variables are built-in variables that are always available in all scopes.

- ◉ **\$ _GET**: Stores the values of variables that are transmitted via the HTTP protocol using the GET method. It is commonly used in forms.
- ◉ **\$ _POST**: Stores the values of variables that are transmitted via the HTTP protocol using the POST method. Its use is generally more appropriate than the GET method for security issues.
- ◉ **\$ _REQUEST**: is an associative array containing the data of \$ _GET, \$ _POST and \$ _COOKIE.
- ◉ **\$ _SESSION**:
- ◉ **\$ _ENV**: Contains environment variables.
- ◉ **\$ _COOKIE**



UD2 ... - Constants & Variables

```
</html>

<body>
  <div id="page-wrap">
    <h1>Php - Water service</h1>

    <form action="" method="post" id="quiz-form">
      <input type="number" name="units" id="units" placeholder="Please enter no. of Units" />
      <input type="submit" name="unit-submit" id="unit-submit" value="Submit" />
      <p>
        <input readonly="readonly" name="result" value="<?php echo $result_str; ?>" /> <b>Euros</b>
      </p>
    </form>
  </div>
</body>
</html>
```




UD2 ... - Operators

Operators are symbols that indicate how operands should be manipulated. The operators together with the operands form a regular expression, which is a formula that defines the calculation of a value.

OPERATOR	TYPE
=	Assignment
+ - * / %	Arithmetic operators
== === != !== < > >= <=	Comparison operators
AND OR ! && !	Logical operators
.	String operators



UD2 ... - Arrays

Els **arrays** ofereixen una forma ideal per guardar, manipular i recuperar conjunts de dades. En el cas de PHP, no cal que aquestes siguin homogènies.

Hi ha tres tipus diferents d'arrays.

- **Array indexats:** el seu índex és numèric. Els seus valors es guarden i son accedits de forma seqüencial i per ordre.
- **Array Associatiu:** es poden entendre com a parelles clau-valor semblants als Hashmap de Java on la clau és un string. Els valors guardats estan associats a un determinat valor clau (únic).
- Fins i tot és possible construir un array amb l'índex numèric i associatiu a la vegada



UD2 ... - Arrays

Els arrays indexats: A diferència d'altres llenguatges PHP no es necessita assignar una mida a l'hora de la seva creació. De fet no fa falta ni declarar-lo abans del seu us (PHP és un llenguatge molt poc tipificat).

EXEMPLE:

```
<?php
```

```
// Primera manera
```

```
$numeros = array( 1, 2, 3, 4, 5 );
```

```
// Segona manera
```

```
$numeros = [1,2,3,4];
```

```
// Tercera manera
```

```
$numeros[0] = 1; $numeros[1] = 2; $numeros[2] = 2; ...
```

```
// Quarta manera
```

```
$numeros = array();
```

```
$numeros[] = 1; $numeros[] = 2; $numeros[] = 3;
```

```
?>
```

```
// Cinquena manera
```

```
$numeros = [ 0 => 1,
```

```
1 => 2,
```

```
2 => 3 ];
```

```
// ... fins i tot
```

```
$numeros = array(1, "dos", 3, "quatre");
```

```
// Accés a l'array per índex
```

```
foreach( $numeros as $n ) {
```

```
    echo $n . "<br/>";
```

```
}
```



UD2 ... - Arrays

Els arrays associatius: fan servir un string per indexar cada valor, fent així una relació més intuïtiva.

EXAMPLE:

```
<?php
// Primera manera
$notes = array( "Pep"=>6, "Maria"=>"nou", "Joan"=>4 );
// Segona manera
$notes["Pep"] = 6; $notes["Maria"] = "nou"; $numeros["Joan"] = 4;
// Accés a l'array associatiu
foreach( $notes as $clau => $valor ) {
    echo "La nota de $clau es $valor <br/>";
}
// En arrays associatius, la funció 'isset()' esdevé necessària per saber si la 'clau' de l'array existeix o no
if (isset($notes["Pep"]) ) { // es troba la nota de "Pep" en l'array
    ...
}
```

??



UD2 ... - Arrays

Els arrays amb índex mixt:

EXEMPLE:

```
<?php
    $fruites = [];
    $fruites[0] = "apples";
    $fruites[1] = "bananas";
    $fruites["fruita"] = "cherries";

?>
```



UD2 ... - Arrays

Atenció a això ... en PHP tot és possible:

EXAMPLE:

```
<?php
```

```
$operacions = [ "sumar" => function($a, $b) { return $a + $b; },  
                "restar" => function($a, $b) { return $a - $b; },  
                "multiplicar" => function($a, $b) { return $a * $b; },  
                "dividir" => function($a, $b) { return $b != 0 ? $a / $b : 0; } ];  
  
echo "Suma: " . $operacions["sumar"](10, 2) . "<br>";  
  
echo "Resta: " . $operacions["restar"](10, 2) . "<br>";  
  
echo "Multiplicació: " . $operacions["multiplicar"](10, 2) . "<br>";  
  
echo "Divisió: " . $operacions["dividir"](10, 2) . "<br>";
```

```
?>
```



UD2 ... - Arrays

Les funcions relacionades amb arrays:

https://www.w3schools.com/php/php_arrays_functions.asp

https://www.w3schools.com/php/php_ref_array.asp



UD2 ... - Control Structures

Control Structures are statements that allow you to control how the code flows in your script based on certain factors.

A PHP script is built from statements. A statement can be an assignment, a function call or a control structure such as a conditional (`if ...`, `switch ...`), a loop (`while ...`, `for ...`, `foreach ...`) or a complex structured block (`try ... catch ... finally ...`).

→ Control structures uses brackets () to enclose conditions and curly braces { } to enclose a set of statements.



UD2 ... - Control Structures

'if' Statement

```
<?php
```

```
    $a = 10;
```

```
    $b = $a * 2;
```

```
    if ($a > $b) {
```

```
        echo "a is bigger than b";
```

```
    } elseif ($a == $b) {
```

```
        echo "a is equal to b";
```

```
    } else {
```

```
        echo "a is smaller than b";
```

```
    }
```

```
?>
```



UD2 ... - Control Structures

'switch' statement

```
<?php
$beer = 'san miguel';
switch($beer) {
    case 'alhambra':
        echo 'Sure ?';
        break;
    case 'carlsberg':
    case 'heineken':
        echo 'Good choice';
        break;
    default:
        echo 'Please make a new selection ...';
        break;
}
```

??>

It is important to understand how the 'switch' statement is executed in order to avoid mistakes.

The `switch` statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a `case` statement is found whose expression evaluates to a value that matches the value of the `switch` expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the `switch` block, or the first time it sees a `break` statement. If you don't write a `break` statement at the end of a case's statement list, PHP will go on executing the statements of the following case, including `default` option



UD2 ... - Control Structures

'while' statement

```
<?php
    $i = 1;

    while ($i <= 10) {
        echo "The number is: $i <br>";
        $i = $i + 1;
    }
?>
```



UD2 ... - Control Structures

'for' statement

```
<?php
    for ($i = 1; $i <= 10; $i++) {
        echo "The number is: $i <br>";
    }
?>
```



UD2 ... - Control Structures

'foreach' statement

```
<?php
```

```
    $arr = array(1, 2, 3, 4); // indexed array
    foreach ($arr as $value) {
        echo $value;
    }
```

```
?>
```

```
<?php
```

```
    $age = array("Peter"=>35, "Ben"=>37, "Joe"=>43); // associative array

    echo json_encode($age);
```

```
?>
```



UD2 ... - Control Structures

'try ... catch ... finally' statement

```
<?php
    try {
        statements_of_try;
        if ( ... ) {
            // throws an exception manually
            throw new Exception ('Error_Message');
        }
    } catch (Error | Exception $e) {
        Statements_of_catch;
        statements_of_catch;
    } finally {
        statements_of_finally;
    }
}
```

?>



UD2 ... - Types of exception

Throwables Exceptions

An **Error** is an unexpected program result, which can not be handled by the program itself. That can be solved by modifying the code manually.

EXAMPLE:

```
$var2 = 0;
```

CODE1

```
$var = $var1 / $var2;
```

CODE2

```
if ($var2 == 0) {  
    throw new Exception();  
}
```

An **Exception** also is an unexpected result of a program but Exception can be handled by the program itself by throwing another exception.

EXAMPLE:

```
$conn = mysqli_connect(HOST, USERNAME, PASSWD, DBNAME);
```



UD2 ... - An example

'try ... catch' statement: you should always check whether a statement fires an error or exception or none of them ...

```
<?php
try {
    $file = fopen("/tmp/examp
    while(! feof($file)) {
        $line = fgets($file);
        echo $line . "<br>";
    }
    fclose($file);
} catch(Error $e) {
    echo "An error has been c
}

?>
```

```
<?php
try {
    if (file_exists("/tmp/example.txt")) {
        $file = fopen("/tmp/example.txt", "r");

        while(! feof($file)) {
            $line = fgets($file);
            echo $line . "<br>";
        }
        fclose($file);
    } else {
        throw new Exception ( "File does not exists");
    }
} catch(Error $e) {
    echo "An error has been caught: " . $e-> getMessage();
}

?>
```




UD2 ... - Custom exceptions - myException

'try ... catch' statement

```
<?php
```

```
class myException extends Exception {}
$config_file_path = "/var/.../config.php";

try {
    if (!file_exists($config_file_path)) {
        throw new myException("Configuration file not found.");
    }
}
catch(myException $e) {
    echo "Defined Exception: " . $e->getMessage();
}
```

```
?>
```



UD2 ... - Defining the error level

'error_reporting()' function

PHP has many levels of errors. Using this function sets that level for the duration (runtime) of your script. If the optional level is not set, *error_reporting()* will just return [the current error reporting level](#). The '*error_reporting*' variable in PHP.ini sets the default value.

```
<?php
// Now, only 'WARNINGS' are reported
error_reporting(E_WARNING);

try {
    $var = 10 / 0;
} catch(Exception $e) {
    // No Error caught
    echo "A 'catch(Exception)' is not necessary";
}
```

?>

Other '*error_reporting()*' examples:

```
// Report all PHP errors
error_reporting(E_ALL);

// Report simple running errors
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// Report all errors except E_NOTICE
error_reporting(E_ALL & ~E_NOTICE);
```



UD2 ... - a Global Exception Handler

'set_error_handler()' function

This **global exception handler** function sets a user-defined error handler function in order to catch exception fired by '*trigger_error()*' function instead exceptions fired by '*throw*'.

```
<?php
```

```
function customError ($errno, $errstr) { echo "Error general: [$errno] $errstr"; }
```

```
set_error_handler("customError");
```

```
try {
```

```
    trigger_error(' ... '); // instead of 'throw'
```

```
} catch(Exception $e) {
```

```
    ...
```

```
}
```

```
?>
```



UD1: ... - Functions

Fins ara els nostres programes han estat molt senzills. Tot el codi es podia executar de cop de manera seqüencialment.

En realitat, els programes poden resultar més complexos i en aquests casos és bona idea desenvolupar cada una de les accions de l'algorisme resultant en un conjunt d'instruccions que condueixen a un fi comú.

- En PHP, una **funció** és un conjunt d'instruccions definides sota un nom descriptiu i que realitza una determinada tasca. Les funcions poden ser invocades des d'altres funcions.



UD1: ... - Functions

EXAMPLE:

```
<?php
    function Exemple() {
        echo "Hello world!!";
    }

    Exemple();
?>
```



UD1: ... – Procediments i funcions

Una **function** pot actuar en forma de ...

1. **PROCEDIMENT**, en aquest cas, el procediment executa un conjunt d'instruccions, però no retorna cap valor, per tant no s'empra la paraula clau **'return'**.

```
function nomProcediment (): void {  
    ... // conjunt d'instruccions del procediment  
}
```

2. **FUNCIO** pròpiament dita, en aquest cas, retorna un valor mitjançant la paraula clau **'return'**

```
function nomFuncio (): tipus {  
    ... // conjunt d'instruccions de la funció  
    return ...  
}
```



UD1: ... – Els paràmetres i els valors de retorn

Un paràmetre és ...

una variable declarada en la signatura d'una funció. que s'utilitza com a variable local per tal de rebre valors quan es crida aquesta. Els paràmetres es defineixen dins els parèntesis de la declaració i s'utilitzen per especificar quin tipus de dades s'espera rebre quan es crida la funció. Aquests **permeten que les funcions siguin flexibles i reutilitzables** ja que aquestes dades poden ser emprades dins la funció per realitzar diferents operacions.

EXEMPLE:

```
function suma($y, $z) { // ← SIGNATURA: nom + relació de paràmetres
    ...                // '$y' i '$z' són 2 paràmetres de 'suma'
}
```



UD1: ... - Els paràmetres i els valors de retorn

Tot i que PHP és un llenguatge no tipat (amb referència a les variables), les versions actuals de l'interpret fan possible especificar el tipus associat a cada un dels paràmetres d'una funció, així com el valor de retorn, **podent especificar un tipus primitiu (anomenat també escalable): 'int', 'float', 'string' o 'bool'.**

EXEMPLE1:

```
function suma(int $y, int $z): int {  
    ... // Es dispara l'excepció 'TypeError' en cas de no coincidir el tipus  
    return ...  
}
```

EXEMPLE2:

```
function exemple2(int $y, int $z): void { ... }
```




UD1: ... - Els paràmetres i els valors de retorn

PHP, a més, permet el tipat mixte tant de paràmetres com del retorn de la funció:

EXAMPLE:

```
function suma(int|string $y, int|string $z): int|string {  
    if (is_int($y) && is_int($z)) {  
        return $y + $z;  
    } else {  
        return $y . $z;  
    }  
}
```

NO RECOMANAT



UD1: ... - Els paràmetres i els valors de retorn

PHP també permet assignar el valor per defecte d'un paràmetre.

EXAMPLE:

```
function suma(int $y=0, int $z=0): int {  
    return ...  
}
```

En aquest cas, les cridades a la funció poden ser:

EXAMPLE:

```
suma(); // emprant els valors per defecte dels 2 paràmetres  
suma(10); // emprant el valor per defecte del segon paràmetre  
suma(z:10); // emprant el valor per defecte del primer paràmetre  
suma(10,20); // sense emprar els valors per defecte
```



UD1: ... - Els paràmetres i els valors de retorn

Atenció, en PHP no es pot definir un paràmetre opcional abans d'un paràmetre obligatori, per tant el següent exemple no seria correcte

EXAMPLE:

```
function suma(int $y, int $z=0): int {  
    return ...  
}
```

En aquest cas:

- o bé caldria donar un valor per defecte a `$y`

```
function suma(int $y=0, int $z=0): int { ... }
```

- o bé canviar l'ordre dels paràmetres

```
function suma(int $z=0, int $y): int { ... }
```



UD1: ... - Els paràmetres i els valors de retorn

A més, PHP permet especificar si el valor del paràmetre pugui ser nullable. Aquesta definició també afecta al valor de retorn. El caràcter clau és '?'.
?

EXEMPLE:

```
function suma(?int $y=null, ?int $z=null): ?int {  
    if ($y === null || $z === null) {  
        return null;  
    } else {  
        return $y + $z;  
    }  
}
```



UD1: ... - Els paràmetres i els valors de retorn

Finalment, PHP permet especificar un nombre indeterminat de paràmetres. El caràcter clau és '...'.

EXEMPLE:

```
function suma(... $numeros) { // '$numeros' es comporta com a un 'array'
    $suma = 0;
    foreach ($numeros as $n) {
        $suma += $n;
    }
    return $suma;
}
```



UD1: ... – Pas de paràmetres per valor o per referència

En general, en programació els paràmetres es poden passar com a ...

- **Pas per valor:** se pasa al mètode el valor de l'argument. Els canvis realitzats en el paràmetre dins el mètode no afecten el valor de la variable original fora del mètode.
- **Pas per referència:** se pasa al mètode l'adreça de memòria com argument del paràmetre en comptes del seu valor. En aquest cas, els canvis realitzats en l'objecte referenciat en el mètode afecten a l'objecte original fora del mètode.

Veiem-ho tot plegat amb més detalls ...



UD1: ... – Pas de paràmetres per valor o per referència

En PHP, el pas de paràmetres per defecte s'entén com a pas '*per valor*'. PHP també admet el pas de paràmetres '*per referència*'. El caràcter clau en aquest cas és '&'.
&

EXEMPLE pas per valor:

```
<?php
function incrementa($param1) {
    $param1 += 1;
    echo "$param1\n";    // $param1=6
}

$var1 = 5;
incrementa($var1);
echo "$var1\n";    // $var1=5
?>
```



UD1: ... – Pas de paràmetres per valor o per referència

EXEMPLE pas per referència:

```
<?php
    function incrementa(&$param1) {
        $param1 += 1;
        echo "$param1\n";    // $param1=6
    }

    $var1 = 5;
    incrementa($var1);
    echo "$var1\n";    // $var1=6
?>
```




Recursos de la UD

- <https://aitor-medrano.github.io/dwes2122/index.html>
- <https://www.apuntesinformaticafp.com/modulos/dwes.html>

Additional resources

- PHP Tutorial
 - <https://www.php.net/manual/es/>
 - <https://www.w3schools.com/php/default.asp>
 - <https://desarrolloweb.com/manuales/manual-php.html>
 - https://www.youtube.com/watch?v=_rEj-RE8cLs&list=PLZ2ovOgdI-kUSqWuyoGJMZL6xldXw6hIg
 - <https://www.youtube.com/watch?v=VGJjCub9vtM&list=PLZ2ovOgdI-kUSqWuyoGJMZL6xldXw6hIg&index=26>
- Laravel, curso tutorial
 - <https://www.youtube.com/watch?v=A-BL8lr7puE&list=PLZ2ovOgdI-kWWS9aq8mfUDkJRfYib-SvF>



Recursos de la UD

Additional resources

- Exceptions: <https://www.php.net/manual/en/language.exceptions.php>
- Error hierarchy <https://www.php.net/manual/en/language.errors.php7.php>
 - Error: <https://www.php.net/manual/en/class.error.php>
 - Exception: <https://www.php.net/manual/en/class.exception.php>
- Predefined constants: <https://www.php.net/manual/en/errorfunc.constants.php>
- set_error_handler: <https://www.php.net/manual/es/function.set-error-handler.php>
- CRUD: <https://www.tutorialrepublic.com/php-tutorial/php-mysql-crud-application.php>
- EXAMPLES: https://www.w3bai.com/es/php/php_examples.html#gsc.tab=0



Recursos de la UD

Additional resources

- mysqli: <https://www.php.net/manual/en/class.mysqli.php>
- PDO: <https://www.php.net/manual/en/class.pdo.php>

PDO vs mysqli comparison (Object-oriented way)

- <https://websitebeaver.com/php-pdo-vs-mysqli>



Bibliografia

- **Implantació d'Aplicacions Web**
CFGS – Administració de sistemes informàtics i xarxes
Institut d'Ensenyaments a Distància de les Illes Balears
 - <https://iedib.net/avirtual/course/view.php?id=263>
Usuari: '*visitant*'; Passwd: '*iedib*'
Revisar els Lliuraments 3, 4 i 5



Autor i Llicència:

Autor: Bartomeu Vives

Sou Lliures de:

Compartir — copiar i redistribuir el material en qualsevol mitjà i format

Adaptar — remesclar, transformar i crear a partir del material

El llicenciador no pot revocar aquestes llibertats, sempre que seguiu els termes de la llicència.

Amb els termes següents:

Reconeixement — Heu de reconèixer l'autoría de manera apropiada, proporcionar un enllaç a la llicència i indicar si heu fet algun canvi. Podeu fer-ho de qualsevol manera raonable, però no d'una manera que suggereixi que el llicenciador us dóna suport o patrocina l'ús que en feu.

NoComercial — No podeu utilitzar el material per a finalitats comercials.

CompartirIgual — Si remescleu, transformeu o creeu a partir del material, heu de difondre les vostres creacions amb la mateixa llicència que l'obra original.

No hi ha cap restricció addicional — No podeu aplicar termes legals ni mesures tecnològiques que restringeixin legalment a altres de fer qualsevol cosa que la llicència permet

<https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>