

Mulige spørgsmål:

Fra Viktoria:

```
1. session_start();  
   if (! isset($_SESSION ['user'])) {  
       header("Location: login");  
   }
```

2. Use of " vs ""

- "" = for tekst og variabler.
- " = kun for tekst. Alt behandles som bogstavelige tegn/bogstaver.

3. Concatination strings

- echo "Hello" . " " . "world";
- echo \$name . " " . \$last_name;

4. Create an array with 3 names

```
a. 
```

5. Loop through the array and show each letter inside a div

6. Highlight the line where the session is removed

(session_destroy)

7. If you make a change to the attribute name (HTML), should you change something in the backend? Yes

- Defined in the variable
- https://www.w3schools.com/html/html_attributes.asp

- c. https://www.w3schools.com/html/html_forms_attributes.asp

Fra andre i klassen:

8. Hvordan finder man længden af et array

- a. Find the length of an array using the `count()`

```
<?php
// Define an array
$array = array("apple", "banana", "orange");

// Get the length of the array
$length = count($array);

// Output the length of the array
echo "The length of the array is: " . $length;
?>
```

- b.
- c. Outputtet kommer til at være: The length of the array is: 3
- d. Da apple, banana og orange giver 3 værdier/values.

9. Forklar noget regex

- a. Det vi har i `validator.js` filen, hvor den validerer en "email" fx.

10. Lav et nyt input field og valider det med php/Validering på client side (validator.js)

```
<input name="user_email" type="text"
data-validate="email"
```

- a.
- b. I `validator.js` filen, sætter man det ind i `data-validate`, som man vil validere:
- `case "str":` = bliver brugt til `user_password`, `user_name`, `user_last_name`.
 - `case "email":` = bliver brugt til `user_email`.
 - `case "match":` = bliver brugt til at matche `user_password` med `user_confirm_password`.

Skal dog også bruge denne linje, for at vide hvad den skal matche med:

`data-match-name="user_password"`

- c. Blev så spurgt "hvordan ville du løse det hvis du var en udvikler i et firma?" Sagde så google, og så bad han mig om at Google det.
- d. Han sagde hvordan ville du google det som en udvikler, så svarede jeg at jeg ville bruge ChatGPT og det havde han det fint med.

11. Logge ind som admin og vise at de kunne Block/unblock - skulle lave en bug så den ikke gjorde det i databasen (fjernede execute linje i api'en/api-toggle-user-blocked.php)

12. Loopede gennem et array

```
// Array
$colors = array(
    "red",
    "green"
);

// Loop array
foreach ($colors as $color) {
    echo $color;
};
```

a.

13. Blev spurgt ind til in_array funktionen

```
$colors = array("red", "green", "blue");

if (in_array("green", $colors)) {
    echo "Found!";
} else {
    echo "Not found!";
}
```

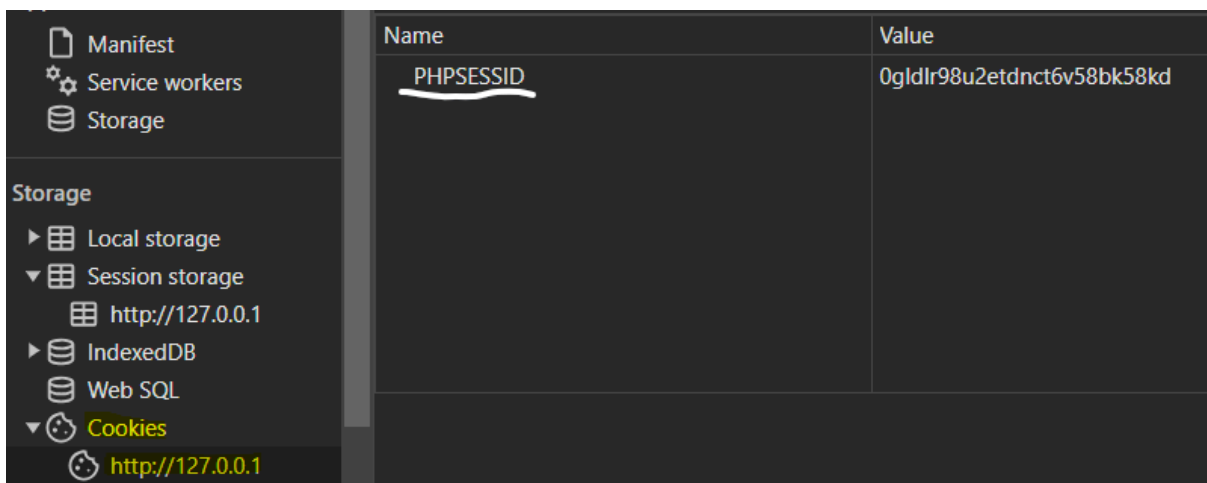
a.

- b. `in_array()` is a PHP function used to check if a value exists in an array. It returns **true** if the specified value is found in the array, and **false** otherwise.

14. Hvordan man starter en session og hvad der sker når man starter en session

- a. `session_start();`
- b. Once the session is started, you can set session variables using the `$_SESSION` superglobal array.
After starting the session, you can perform various operations such as setting session variables, modifying them, or accessing them later in the script.

15. Cookies - skulle vise cookie, han spurgte om den blev lavet af mig, svarede at det sker når vi starter session



The screenshot shows the Chrome DevTools Storage tab. On the left, the 'Storage' section is expanded, and 'Cookies' is selected. Below it, the domain 'http://127.0.0.1' is listed. The main pane displays a table of cookies. One cookie is visible with the name 'PHPSESSID' (underlined) and the value '0gldlr98u2etdnct6v58bk58kd'.

Name	Value
<u>PHPSESSID</u>	0gldlr98u2etdnct6v58bk58kd

- a. PHPSESSID = is a cookie name commonly used by PHP to store the session ID. When you start a session in PHP using `session_start()`, PHP generates a unique session ID for the user's session. This session ID is then stored in a cookie named "PHPSESSID" on the client's browser.
- b. **How it works:**
 - i. When you call `session_start()` in your PHP script, PHP checks if a session ID cookie named "PHPSESSID" exists in the client's browser.

- ii. If the "PHPSESSID" cookie doesn't exist, PHP generates a new session ID and sends it to the client's browser in the form of a "PHPSESSID" cookie.
- iii. On subsequent requests, the client's browser automatically sends the "PHPSESSID" cookie back to the server, allowing PHP to identify the correct session data associated with the client.
- iv. PHP then uses the session ID to retrieve the corresponding session data stored on the server, allowing you to access and manipulate session variables.

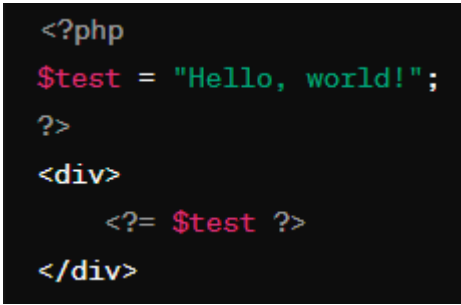
Noter:

1. HTTP statuskoder:

- a. 200 = alt er OK
- b. 400 = client error
- c. 500 = server error

2. echo statement in PHP

- a. `echo "Hello world!";`
- b. `<?= $test ?>` = shortcut syntax.

c. A screenshot of a code editor showing PHP code. The code uses the short tag syntax for both PHP execution and HTML output. It starts with a PHP opening tag, assigns the string 'Hello, world!' to a variable named \$test, then closes the PHP tag. This is followed by an HTML opening tag <div>, which contains a short tag <?= \$test ?> to output the value of \$test, and ends with an HTML closing tag </div>.

```
<?php
$test = "Hello, world!";
?>
<div>
    <?= $test ?>
</div>
```

3. Kode linje for at få en bestemt HTTP statuskode:

- a. `<?php`
`http_response_code(404);`
`?>`

4. REST API always works with JSON. Computers can only send text and can only get text.

5. PHP json_encode() Function:

```
a. <?php
    $age = array("Peter"=>35, "Ben"=>37,
    "Joe"=>43);
    echo json_encode($age);
?>
```

6. PHP Global Variables - Superglobals:

- a. `$_POST` - Create/opret.
 - i. `$_POST` contains an array of variables received via the HTTP POST method.
- b. `$_GET` - Read/læs.
 - i. `$_GET` contains an array of variables received via the HTTP GET method.
- c. `$_SESSION` - A session is a way to store information (in variables) to be used across multiple pages.
Unlike a cookie, the information is not stored on the users computer.

7. Error in the console, på login.php siden:

- a. Uncaught TypeError: Cannot read properties of null (reading 'addEventListener') at app.js:19:13
- b. Denne fejl bliver vist da der ikke er et id der hedder `"#query_results"`, den kode bliver brugt på profil siderne til customer, partner, employee og admin.

8. PHP if Statements:

- a. `if` statement - executes some code if one condition is true.
- b. `if...else` statement - executes some code if a condition is true and another code if that condition is false.

9. PHP preg_match() Function:

`preg_match` — Perform a regular expression match against a string. It returns true if a match is found, and false otherwise.

In regular expressions, the "i" modifier is used to perform a case-insensitive match.

I `$str` står der "World" og i `$pattern` står der "world". På denne måde kan der findes et match, da den ene har et stort forbogstav og den anden har et lille forbogstav.

```
a. <?php
    $str = "Hello World";
    $pattern = "/world/i";
    echo preg_match($pattern, $str);
?>
```

```
// Input string
$string = "Hello, world!";

// Regular expression pattern
$pattern = "/world/";

// Perform the match
if (preg_match($pattern, $string)) {
    echo "Match found!";
} else {
    echo "No match found.";
}
```

10. PHP Regular Expressions

- A regular expression is a sequence of characters that forms a search pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

11. htaccess

- RewriteEngine On

12. Concatenation of two strings in PHP

- There are two string operators. The first is the concatenation operator (`.`), which returns the concatenation of its right

and left arguments. The second is the concatenating assignment operator (`.=`), which appends the argument on the right side to the argument on the left side.

```
<?php
// Define the first string
$string1 = "Hello";

// Define the second string
$string2 = "World";

// Concatenate the two strings with a space in between
$combinedString = $string1 . " " . $string2;

// Output the concatenated string
echo $combinedString;
?>
```

13. PHP | Magic Constants

- a. `__DIR__` = This magic constant returns the directory of the executed file.
- b. `require_once __DIR__ . "/../_ .php";`

14. PHP `require_once` Keyword

- a. The `require_once` keyword is used to embed PHP code from another file. If the file is not found, a fatal error is thrown and the program stops. If the file was already included previously, this statement will not include it again.
- b. `require_once __DIR__ . "/../_ .php";`

15. PHP `include_once` Keyword

- a. The `include_once` keyword is used to embed PHP code from another file. If the file is not found, a warning is shown and the program continues to run. If the file was already included previously, this statement will not include it again.

- b. `include_once`
`__DIR__.'/_form_search_orders.php';`

16. Try/catch

```
try {  
    // Code that may throw an exception  
} catch (Exception $e) {  
    // Code to handle the exception  
}
```

- a.
- b. `try` - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown".
- c. `throw` - This is how you trigger an exception. Each "throw" must have at least one "catch".
- d. `catch` - A "catch" block retrieves an exception and creates an object containing the exception information.

17. PHP Exception Handling

- a. `Exception` is the type of exception being caught. It can be a specific exception class (e.g., `PDOException`, `InvalidArgumentException`, etc.) or the generic `Exception` class to catch any type of exception.
- b. `$e` is the variable name that holds the exception object. You can use this variable to access information about the exception, such as its message, code, file, line number, and stack trace.
- c. For example, you can access the exception message using `$e->getMessage()`, the exception code using `$e->getCode()`, and so on.
- d. Using `$e` allows you to handle the exception gracefully, such as logging an error, displaying a user-friendly message, or taking corrective action in your code.
- e. `Exception()` = The constructor of the Exception object.
 - i. Vi bruker `throw new`, når vi vil bruke en `Exception`.

- ii. `throw new Exception('ups...', 500);`
- f. `getCode()` = Returns the exception code.
- g. `getMessage()` = Returns a string describing why the exception was thrown.
- h. `getLine()` = Returns the line number of the line of code which threw the exception.

18. Null Coalescing Operator (??)

- a. `$variable = $value1 ?? $value2;`
- b. Here, if `$value1` is not null, `$variable` will be assigned the value of `$value1`. If `$value1` is null, `$variable` will be assigned the value of `$value2`.

19. Logical Operators

Example	Name	Result
<code>\$a and \$b</code>	And	true if both <code>\$a</code> and <code>\$b</code> are true .
<code>\$a or \$b</code>	Or	true if either <code>\$a</code> or <code>\$b</code> is true .
<code>\$a xor \$b</code>	Xor	true if either <code>\$a</code> or <code>\$b</code> is true , but not both.
<code>! \$a</code>	Not	true if <code>\$a</code> is not true .
<code>\$a && \$b</code>	And	true if both <code>\$a</code> and <code>\$b</code> are true .
<code>\$a \$b</code>	Or	true if either <code>\$a</code> or <code>\$b</code> is true .

20. PHP Constants

- a. Constants are like variables, except that once they are defined they cannot be changed or undefined.

- b. A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- c. A valid constant name starts with a letter or underscore (no \$ sign before the constant name).
- d. **Note:** Unlike variables, constants are automatically global across the entire script.
- e. `define('USER_NAME_MIN', 2);`
`define('USER_NAME_MAX', 20);`
`USER_NAME_MIN` is being defined as a constant with the value `2`. This means that `USER_NAME_MIN` will always have the value `2` throughout the execution of the script, and it cannot be modified elsewhere in the script.

21. Query string

- a. A query string is a part of a URL that appears after the question mark (?). It is used to pass data between a web server and a web browser. The query string consists of one or more `key-value` pairs, where each pair is separated by an ampersand (&), and the `key` and `value` are separated by an equals sign (=).

`one.php?name=maria&school=kea`

22. PHP foreach Loop

```
// Array
$colors = array(
    "red",
    "green"
);

// Loop array
foreach ($colors as $color) {
    echo $color;
};
```

- a.
- b. Viser "red" og "green" på siden.

23. PHP Functions

- a. https://www.w3schools.com/php/php_functions.asp

24. Friendly URL

- a. <https://www.danielmorell.com/guides/htaccess-seo/user-friendly-urls/make-your-urls-seo-friendly>
- b. This URL is **NOT** SEO-friendly:
 - i. <https://cdn03.example.com/8cb42/index.php?35872=8zh3n9vadbxcgac9c&id=851368#top>
- c. This URL is SEO-friendly:
 - i. <https://example.com/blog/introduction-to-seo-friendly-urls>

25. PHP require

- a. `require` is identical to `include` except upon failure it will also produce a fatal **E_COMPILE_ERROR** level error. In other words, it will halt the script.

26. PHP require_once

- a. The `require_once` expression is identical to `require` except PHP will check if the file has already been included, and if so, not include (`require`) it again.

27. PHP include

- a. `include` only emits a warning (**E_WARNING**) which allows the script to continue.

28. PHP include_once

- a. The `include_once` expression is identical to `include` except PHP will check if the file has already been included, and if so, not `include` it again.

29. PHP Sessions

- a. PHP sessions provide a way to persist data across multiple page requests for a single user session.
- b. A session is started with the `session_start()` function.

- c. Session variables are set with the PHP global variable: `$_SESSION`.

```
<?php
// Start the session
session_start();

// Check if the user is logged in
if(isset($_SESSION['user_id'])) {
    // User is logged in, display a welcome message
    echo "Welcome back, " . $_SESSION['username'] . "!";
} else {
    // User is not logged in, display a login form
    ?>
    <form action="login.php" method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </form>
    <?php
}

// Example of setting session variables
$_SESSION['user_id'] = 123;
$_SESSION['username'] = "john_doe";

// Example of destroying a session
// session_destroy();
?>
```

d.

30. PHP ternary

- a. `(Condition) ? (Statement1) : (Statement2);`
- b. **Condition:** It is the expression to be evaluated which returns a boolean value.
- c. **Statement 1:** it is the statement to be executed if the condition results in a **true** state.

- d. **Statement 2:** It is the statement to be executed if the condition results in a **false** state.
- e. app.js line 151:

```
user_is_blocked = user_is_blocked === "1" ? "0" : "1";
```

Condition: 1, hvis en user er blocked er det 0

Statement 1: 0, hvilket er **true**, da 0 = user_is_blocked

Statement 2: 1, hvilket er **false**, da 1 = user is not blocked

- f. _.php line 188:

```
function _is_user_customer(){  
    return (! isset($_SESSION['user']) || $_SESSION['user']['user_role_id'] != 1) ? false : true;  
}
```

Condition: Hvis useren i **SESSION** ikke er der (**! isset**) eller (**||**), hvis useren i **SESSION** ikke har **user_role_id = 1**, bliver useren returneret (**return**).

Note: Hvis en user er customer, har de role id 1.

Statement 1: false, da det er **true** at useren ikke er en customer.

Statement 2: true, da det er **false** og useren er en customer.

The function returns **true** if the user is logged in and has a role ID of 1, and **false** otherwise.

31. PHP cookies

- a. A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too.

32. PHP var_dump

- a. Debugging and displaying information about variables, including their type and value. Can also be used for arrays.

33. PHP print_r

- a. Display the contents of an array. Prints human-readable information about a variable.

34. PHP filter_var() Function

- a. The `filter_var()` function filters a variable with the specified filter. It's commonly used for data validation and sanitization.
- b. Master file/_.php line 86:

```
$_POST['user_email'] = trim($_POST['user_email']);  
if( ! filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL) ){  
    throw new Exception($error, 400);  
}
```

Checks if the user_email is a valid email address using `filter_var()` with the `FILTER_VALIDATE_EMAIL` filter.

- c. PHP Predefined Filter Constants =
`FILTER_VALIDATE_EMAIL`

Check if the variable \$email is a valid email address:

```
<?php  
$email = "john.doe@example.com";  
  
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {  
    echo("$email is a valid email address");  
} else {  
    echo("$email is not a valid email address");  
}  
?>
```

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

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

35. PHP strlen

- a. The `strlen()` function returns the length of a string.
- b. `echo strlen("Hello");` = resultatet bliver 5, da der er 5 bogstaver.

36. PHP trim

- a. The `trim()` function removes whitespace and other predefined characters from both sides of a string.

- b. Bliver fx brugt til `user_email`, hvis nu en user kommer til at lave et mellemrum.

37. **PHP rtrim() Function**

- a. The `rtrim()` function removes whitespace or other predefined characters from the right side of a string.

38. **Adjacent elements**

- a. <https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentHTML>