****Sesc Senac****  
****Aluno: Guilherme de Moura Rodrigues****



Superglobais:

**$\_SESSION**

Essa superglobal guarda dados, temporários do usuario enquanto ele navega no site, e no site esses dados ficam salvos no servidor e cada usuario tem sua própria sessão, e a sessão dura até o usuario sair do site ou fechar o navegador

**Exemplo**

<?php

session\_start();

$\_SESSION["nome"] = $\_POST['nome'] ?? 'esperando';

echo "Bem vindo ". $\_SESSION["nome"] ;

?>

<form *action*="<?=$\_SERVER['PHP\_SELF'];?>" *method*="post">

<label *for*="nome">Digite seu nome:</label>

<input *type*="text" *id*="nome" *name*="nome" *required*>

<input *type*="submit" *value*="Enviar">

</form>

</body>

</html>

**$\_POST**

Essa superglobal pega dados enviados, é um método de pegar informações digitadas pelo usuário em um formulário por exemplo, no caso ela pode acessar os dados que foram enviados, mas só funciona quando o servidor está ligado, as informações não são guardadas sozinhas no caso se desligar e ligar as informações somem só se você as guardar no banco de dados

**Exemplo**

<form action="<?=$\_SERVER['PHP\_SELF']?>" method="post">

<label for="id">informe seu id</label>

<input type="text" name="id" id="id">

<?php

$id = $\_POST['id'] ?? 'nada';

echo "<br>Seu id=$id esta registrado";

?>

**$\_COOKIE**

Essa superglobal ela serve para pegar cokies que estão armazenados no navegador, e ler eles no exemplo abaixo criem um cookie e usei a superglobal para lê-lo

**Exemplo**

<?php

setcookie("nome", "Fulaninho", time()+3600);

echo "Seu nome é " . $\_COOKIE["nome"];

?>

Armazenamento no navegador:

**LocalStorage**

É tipo um mini banco de dados ele guarda informações no local como diz o nome, no navegador que ficam salvas até se você fechar o navegador

<label *for*="nome">Digite seu nome:</label>

<input *type*="text" *id*="nome" *placeholder*="Seu nome">

<button *onclick*="salvarNome()">Salvar</button>

<h3 *id*="boasVindas"></h3>

<script>

function salvarNome() {

const nome = document.getElementById("nome").value;

localStorage.setItem("usuario", nome);

mostrarMensagem();

}

function mostrarMensagem() {

const nomeSalvo = localStorage.getItem("usuario");

if (nomeSalvo) {

document.getElementById("boasVindas").textContent = "Bem-vindo, " + nomeSalvo + "!";

}

}

mostrarMensagem();

</script>

Funções importantes:

**isset()**

Essa função pode ver se uma variável está definida e não é nula. Se não existe ou é nula, se ela estiver definida e não for nula. ela retorna o valor true, se não existir ou estiver nula, ela retorna false

**Exemplo**

<?php

$x = "Oie";

$y = null;

$z;

var\_dump(isset($x));

echo "<br>";

var\_dump(isset($y));

echo "<br>";

var\_dump(isset($z));

?>

**empty()**

Essa função pode ver se uma variável está vazia ou não, se a variável for considerada vazia ele retorna true no caso considerada vazia no caso seria zero, um array vazio, null ou nada. Se ela não for considerada vazia e ter qualquer coisa sem ser as coisas que falei antes ela retornara false

**Exemplo**

<?php

$x = "Oie";

$y = null;

$z;

$w = "0";

$q = array();

var\_dump(empty($x));

echo "<br>";

var\_dump(empty($y));

echo "<br>";

var\_dump(empty($z));

echo "<br>";

var\_dump(empty($q));

echo "<br>";

var\_dump(empty($w));

?>

**filter\_input()**

Essa função como o nome diz é um filtro, essa função pode filtrar inputs por exemplo informações que vem de um formulário, nessa função existe vários filtros cada um com uma função eles são:

FILTER\_SANITIZE\_STRING = tira tags html e códigos perigosos, mas não é a função melhor para evitar ataques de hackers, ele só limpa o básico

FILTER\_SANITIZE\_NUMBER\_INT = tira letras e símbolos de número, no caso só aceita números inteiros

FILTER\_VALIDATE\_EMAIL = ele só aceita um valor que é considerado valido de E-mail por exemplo: [gui@gmail.com](mailto:gui@gmail.com)

FILTER\_VALIDATE\_URL = garante que a url seja valida

**Exemplo**

<form action="<?=$\_SERVER['PHP\_SELF']?>" method="post">

<label for="ma">Digite seu Email</label>

<input type="text" name="ma" id="ma">

<input type="submit" value="Enviar">

</form>

<?php

$email = filter\_input(INPUT\_POST,"ma", FILTER\_VALIDATE\_EMAIL);

echo " o e-mail agora esta salvo $email";

?>

**filter\_var()**

Essa função filtra as variáveis, e bem parecido com o anterior, você coloca a função filter\_var() e depois você escolhe um filtro cada filtro tem uma funcionalidade são eles, uma diferença entre filter\_input() e filter\_var e que filter\_var() pode retornar o valor false. Os filtros são:

FILTER\_VALIDATE\_INT = válidas números inteiros, se o número for uma String “8” ele vira int e se não for um número “oi sou o número oito” ele retorna false

FILTER\_VALIDATE\_EMAIL = a mesma coisa do anterior só que esse só valida e-mails

FILTER\_SANITIZE\_STRING = ela tirava os caracteres especiais, mas foi descontinuada

FILTER\_SANITIZE\_URL = tira caracteres que são proibidos em URLs.

FILTER\_SANITIZE\_NUMBER\_INT = tira tudo menos números e os sinais de + e -;

<?php

$numero = "8";

$naonumero = "oi sou o numero oito";

$string = "eu vou aparecer4vougg+fb4!!!!=grggggr8";

$email = "[jean@gmail.com](mailto:jean@gmail.com)";

$talvezemail = "com certeza sou um email";

var\_dump(filter\_var($naonumero, FILTER\_VALIDATE\_INT));

echo"<br>";

var\_dump(filter\_var($numero, FILTER\_VALIDATE\_INT));

echo"<br>";

var\_dump(filter\_var($email, FILTER\_VALIDATE\_EMAIL));

echo"<br>";

var\_dump(filter\_var($talvezemail, FILTER\_VALIDATE\_EMAIL));

echo"<br>";

var\_dump(filter\_var($string, FILTER\_SANITIZE\_NUMBER\_INT));

?>

**htmlspecialchars()**

Essa função serve como proteção transformando caracteres especiais como < , “, >,& etc. em entidades como se fosse nada especial como se fosse algo escrito mesmo, isso ajuda a impedir que um hacker coloque scripts ou códigos em um formulário por exemplo

**Exemplo**

<?php

$duvidoso = "<h1><script> alert('hackei você muahhahahahahahahahahahahahahhahahahahahahaahahahahhahahahaahhahaahahhahahahahahaahhahahahahahahahahahahahahahahahhahahhhhhhhahaahahahahhahaahhahahaha')</script></h1>";

$seguro = htmlspecialchars($duvidoso);

echo "$seguro";

?>

**array()**

Essa função ela cria uma variável que tem vários valores dentro e como se fosse uma prateleira de carinhos reta pode armazenar varios valores, os valores dentro de um array não precisam ser do mesmo tipo então pode ter uma string junto com um int e um bool até mesmo uma array que armazena várias variáveis, esses valores podem ser pegos por meio de chaves siga o exemplo abaixo

**Exemplo**

<?php

$exem = "sou um exemplo:)";

$exemploarray = array("teste",1 ,false,"oiii", $exem);

echo" pegando as chaves<br>

$exemploarray[0] <br>

$exemploarray[1] <br>

$exemploarray[2] <br>

$exemploarray[3] <br>

$exemploarray[4]";

?>

**Foreach**

Essa função é tipo um controle de arrays ela serve para iterar elementos arrays e objetos, ela pode mostrar todos os valores dentro de uma array de uma vez só como exemplo, isso deixa mais facil olhar todos os valores dentro da array

**Exemplo**

<?php

$exem = "sou um exemplo:)";

$exemploarray = array("teste",1 ,false,"oiii", $exem);

foreach($exemploarray as $chave => $valor){

echo "O valor com a chave ". $chave . " é " . $valor ."<br>";

}

?>

**array\_push()**

Essa função pode adicionar valores no final de uma array já criada é como se você comprasse mais carinhos para a prateleira depois dela já feita, e colocasse eles no final da fila

**Exemplo**

<?php

$exem = "sou um exemplo:)";

$exemploarray = array("teste",1 ,false,"oiii", $exem);

var\_dump($exemploarray);

array\_push($exemploarray, "oi cheguei", true, 15);

echo"<p>";

var\_dump($exemploarray);

?>

**in\_array()**

Essa função é como se fosse um verificador ele verifica se algum valor existe em um array se esse valor existir ele retorna true, se não existir retorna false.

**Exemplo**

<?php

$souumarray = array("falso","verdade");

var\_dump(in\_array("verdade", $souumarray));

echo"<br>";

var\_dump(in\_array("verdadeiro", $souumarray));

?>

**explode()**

Essa função ela serve como um criador de array, ele pode gerar array com string, e pode dividir um valor com & da string tipo oi&bomdia o explode vai criar uma array que contém oi e bomdia separado.

**Exemplo**

<?php

$exem = "sou&um&exemplo:)";

$explodido = explode("&",$exem);

var\_dump($explodido);

?>

**Implode()**

Essa função é tecnicamente o contrário do explode(), ela pode transformar um array em um string e juntar todas os valores e pode colocar algum valor para intercalar os valores

**Exemplo**

<?php

$exem = "jean";

$exemploarray = array("oii" ,"professor",$exem);

$euimplode = implode(" ",$exemploarray);

echo $euimplode;

?>

**strlen()**

Essa função conta quantos bytes tem em uma variável normalmente é o número de caracteres tem alguns caracteres como emojis que valem mais bytes

**Exemplo**

<?php

$exemplo = "money,money,money";

$bytes = strlen($exemplo);

echo $bytes;

?>

**str\_replace()**

Essa função e como se fosse um trocador de palavras em arrays e strings, ele procura um valor e troca ele

**Exemplo**

<?php

$exemplo = "Oiiii euvousumir Guanabara";

$trocado = str\_replace("euvousumir", "Gustavo", $exemplo);

echo $trocado

?>

**substr()**

Essa função é como se fosse uma vizinha fofoqueira, ele pode pegar informações de um string, a partir de um ponto que você define e ele se guia por bytes, que no caso seria até onde ela ia ouvir dessa fofoca

**Exemplo**

<?php

$fofoca = "o cachorro da vizinha esta com raiva";

$Cleide = substr($fofoca, 0, 10);

echo $Cleide;

$Neide = substr($fofoca, 10, 26);

echo $Neide;

?>

**strpos()**

Essa função serve para ver se um valor existe em uma string, e se existir fala em que posição de caractere ela se encontra e se não existir retorna false

**Exemplo**

<?php

$exemplo = "sou um exemplo e só";

$eustrpos = strpos($exemplo,"só");

var\_dump($eustrpos);

$eustrpos2 = strpos($exemplo, "nada");

var\_dump($eustrpos2);

?>

Importação de arquivos:

**Include**

Essa função pode trazer outro arquivo para o arquivo atual, e como dar ctrl c ctrl v na página

**Exemplo**

**Informa.php**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Vou aparecer na outra pagina oiiiii</h1>

</body>

</html>

**Index.php**

<?php

include 'informa.php';

?>

**require**

Essa função tem tecnicamente a mesma função do include so que se essa função nao achar o arquivo ela dá fatal error

**Exemplo**

**Informa.php**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Vou aparecer na outra pagina oiiiii</h1>

</body>

</html>

**Index.php**

<?php

require 'informa.php';

?>

**include\_once**

Essa função tem a mesma função do include, mas com uma diferença o arquivo puxado so aparece uma vez, se eu escrevesse várias vezes o include eu teria o mesmo arquivo várias vezes, mas se eu fizer a mesma coisa com o include\_once ele so vai aparecer uma vez

**Exemplo**

**Informa.php**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Vou aparecer na outra pagina oiiiii</h1>

</body>

</html>

**Index.php**

<?php

include\_once 'informa.php';

include\_once 'informa.php';

?>

**require\_once**

Essa função é literalmente a mesma coisa da include\_once, só que com require, então essa função puxa um arquivo só uma vez e se ele não existir da fatal error  
**Exemplo**

**Informa.php**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Vou aparecer na outra pagina oiiiii</h1>

</body>

</html>

**Index.php**

<?php

require\_once 'informa.php';

require\_once 'informa.php';

?>

Parte extra (vale como reforço!):

***1-O que são variáveis superglobais? Cite outras além das já listadas.***

***R:*** As superglobais são variáveis automáticas do php que podem ser usadas em qualquer código, alguns exemplos são $\_SERVER que eu usei em alguns exemplos para fazer um formulário de só uma página essa é uma das funções dessa variável por exemplo ela contém todos os dados do servidor, $\_REQUEST que funciona em formulários por exemplo o $\_REQUEST é tipo uma fusão do get, post e cookie, $\_GET pega informações pela URL basicamente a mesma função do $\_POST

***2- Qual a diferença entre isset() e empty()?***

***R:*** isset manda a resposta true se existir alguma coisa na variável que nao seja null ou nada, o empty é o ao contrário se tiver alguma coisa na variável ele retorna false e retorna true só se for considerada vazia

***3-Para que serve htmlspecialchars()?***

***R:*** serve para proteger o site de scripts maliciosos exemplo se um hacker colocar um script no formulário o htmlspecialchars transforma esses caracteres em texto, evitando que o script seja executado

***4-Explique a diferença entre explode() e implode() com exemplo.***

***R:*** explode serve para quebrar um string em partes usando um separador que você escolhe, no caso dividir um valor seja de um string como exemplo o%i posso colocar para explodir o % e se for uma string vira um array com os valores o e i, já o implode pode transformar um array em uma string juntando todos os valores em um texto

***5-O que acontece se usar require para incluir um arquivo que não existe? E se usar include?***

***R:*** require faz a mesma coisa que include a diferença e se não existir arquivo o require da fatal error e para tudo, já o include só da erro, mas não fatal então o código continua funcionando

***6-O que é LocalStorage e quando ele é mais útil que cookies ou sessões?***

***R:*** LocalStorage guarda informações no navegador e elas ficam salvas mesmo se fechar o navegador. É melhor que cookies quando você quer guardar dados grandes ou que não precisam ir pro servidor sempre. É melhor que sessão quando quer que os dados fiquem por mais tempo, porque sessão some quando fecha o navegador.

***7-Em um sistema de login, quando você usaria $\_SESSION e quando usaria $\_COOKIE?***

***R:*** Usaria $\_SESSION pra guardar os dados só enquanto a pessoa estiver no site, tipo o nome do usuário. Quando sair ou fechar o navegador, some tudo. Já o $\_COOKIE é bom pra lembrar do login por mais tempo, tipo “lembrar de mim”, porque ele fica salvo no navegador mesmo depois de fechar.