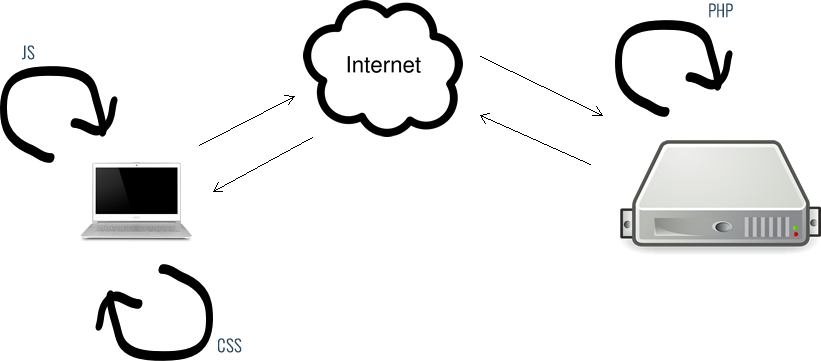
4. PHP

# 4.1. Intro

Hasta ahora, podemos transformar diseños web en código HTML, CSS y JS.

¿Qué pasa cuando queremos iniciar una sesión, guardar información o crear elementos que dependen de una lógica?

PHP es un lenguaje de programación Server-Side orientado al desarrollo web.  
Es server-side porque solamente se ejecuta en el servidor, NO en el cliente.



# **4.2. Algunas funcionalidades simples**

## 4.2.1. Echo

…

<body>

<?php

echo(“Hola”);

?>

</body>

…

## 4.2.2. Redirect

<?php

header(“Location: google.com”);

?>

## 4.2.3. Arrays

<?php

$fruitArray = array(“manzana”,”banana”,”naranja”);   
 $fruitArray = [“manzana”,”banana”,”naranja”];

for($i = 0; $i < count($fruitArray); $i++){

$fruit = $fruitArray[$i];

echo($fruit);

}

foreach($fruitArray as $fruit){

echo($fruit);

}

$teamPointsArray = array(

“fenix”=>21,

“wanderers”=>20,

“danubio”=>18,

“peñarol”=>15,

);

foreach($teamPointsArray as $key=>$value)

echo(“El equipo ”. $key.” tiene ”.$value.” puntos”);

}

?>

# **4.3. Ejercicio: PHP -> HTML**

En el ejercicio anterior, generar la lista de hidden inputs de imágenes, a partir de un array de sources de php.

# **4.4. HTML -> PHP**

Desde el browser del cliente tenemos 2 maneras de mandar información al servidor: GET y POST.

## 4.4.1. GET

GET se utiliza para pasar información insensible, porque aparece en la dirección del browser. También podemos utilizarlo si queremos que se vea en la url del browser. Tiene una limitación de caracteres.

…

<body>

<?php

if(isset($\_GET["user"])

echo("Hola ". $\_GET["user"]);

?>

<form action="/index.php" method="GET">

Ingrese su nombre <input type="text" name="user"/>

<button type="submit">Continuar</button>

</form>

</body>

…

## 4.4.2. POST

POST va a ser el más utilizado en Moon, los datos se pasan sin registro de los mismos en la URL. Sin embargo, se pueden ver en nuestro Browser utilizando las Developer Tools.

…

<body>

<?php

if(isset($\_POST[“user”])

echo(“Hola ”. $\_POST[“user”]);

?>

<form action=”/index.php” method=”POST”>

Ingrese su nombre <input type=”text” name=”user”/>

<button type=”submit”>Continuar</button>

</form>

</body>

…

# **4.3. Ejercicio: HTML-> PHP -> HTML**

Crear un form en “login.php” que pide una contraseña. Si la contraseña ingresada es “MoonPass” redirigirlo a “welcome.php” si no es correcta, mostrar un mensaje de error en la misma página.

# **4.4. Ejercicio: HTML-> JS -> AJAX -> PHP -> AJAX**

Crear un form en “loginAjax.php” que pide una contraseña.

Mandar la contraseña via Ajax, POST, a un nuevo archivo php que se llama checkPassword.php.

Si la contraseña es correcta, redirigir al usuario a welcome.php, si no es correcta mostrar un mensaje en la misma web.

¿Alguna ventaja de utilizar este método en vez del anterior?

var user = "seba";

var contrasena = "sebamoon";

$.ajax({

*type*: 'POST',

url: '/index.php',

data:{'user':*user*, 'contrasena':*contrasena*},

success: function(response){

},

error: function(){

alert('Error obteniendo datos del sitio, verifique su conexión a internet.');

}

});

}

});

# **4.5. Ejercicio: JSON**

JSON simplemente transforma arrays a un formato JSON. Todos los lenguajes saben interpretar información en JSON, por lo que se hace muy fácil compartir información entre lenguajes utilizando JSON.

PHP ( json\_encode(response) ) -------JSON------> JS (#.json())

Hacer el mismo sistema que en el ejercicio anterior, solamente que ahora checkPassword devuelve las siguientes opciones:

$response = [];

$response[‘status’] = ‘ok’;

$response[‘status’] = ‘error’;   
$response[‘errorMessage’] = ‘La contraseña es incorrecta’;

$response[‘status’] = ‘error’;   
$response[‘errorMessage’] = ‘No ingresaste una contraseña’;

echo json\_encode($response);