📌 Explicación detallada del artículo "Trabajar con JSON en PHP"

JSON (**JavaScript Object Notation**) es un formato ligero para el intercambio de datos, muy usado en aplicaciones web y APIs. PHP proporciona funciones integradas para **codificar (convertir a JSON)** y **decodificar (convertir desde JSON)** de manera eficiente.

1️⃣ Convertir datos en PHP a JSON (`json_encode`)

PHP tiene la función `json_encode()` para convertir arrays y objetos en una cadena JSON.

Ejemplo 1: Convertir un array asociativo a JSON

```php
$usuario = [
    "nombre" => "Juan",
    "edad" => 30,
    "email" => "juan@example.com"
];

$json = json_encode($usuario);
echo $json;
```

- **Salida JSON:**

```
{"nombre":"Juan","edad":30,"email":"juan@example.com"}
```

📌 `json_encode()` **convierte el array PHP en una cadena JSON que puede ser enviada o almacenada.**

2️⃣ Convertir JSON a PHP (`json_decode`)

Para interpretar un JSON en PHP, usamos `json_decode()`.

Ejemplo 2: Convertir JSON a un array en PHP

```php
$json = '{"nombre":"Juan","edad":30,"email":"juan@example.com"}';

$array = json_decode($json, true); // `true` convierte a array asociativo
print_r($array);
```

- **Salida en PHP (`print_r` muestra un array):**

```
Array
(
```

```
    [nombre] => Juan
    [edad] => 30
    [email] => juan@example.com
)
```

📌 **Si `json_decode()` se usa sin `true`, devuelve un objeto en lugar de un array.**

Ejemplo 3: Convertir JSON a un objeto PHP

```php
$datos = json_decode($json);
echo $datos->nombre; // Accede como objeto
```

◇ **Salida:**

```
Juan
```

---

3 Opciones adicionales en `json_encode()`

`json_encode()` acepta opciones para modificar la salida.

Ejemplo 4: Formatear JSON con `JSON_PRETTY_PRINT`

```php
$usuario = [
    "nombre" => "Juan",
    "edad" => 30,
    "email" => "juan@example.com"
];

$json = json_encode($usuario, JSON_PRETTY_PRINT);
echo $json;
```

◇ **Salida JSON con formato legible:**

```
{
    "nombre": "Juan",
    "edad": 30,
    "email": "juan@example.com"
}
```

📌 **Se usa `JSON_PRETTY_PRINT` para una mejor visualización, útil en debugging.**

4️⃣ Errores en JSON (`json_last_error`)

Si el JSON tiene errores, `json_decode()` puede fallar. Para verificar errores, usamos `json_last_error()`.

Ejemplo 5: Detectar errores en JSON

```
$json_mal = '{"nombre": "Juan", "edad": 30, "email": "juan@example.com"'; // Falta el cierre }

$data = json_decode($json_mal);

if (json_last_error() !== JSON_ERROR_NONE) {
    echo "Error en el JSON: " . json_last_error_msg();
}
```

- **Salida:**

```
Error en el JSON: Syntax error
```

📌 **Siempre verifica errores después de `json_decode()`.**

5️⃣ Trabajar con archivos JSON (`file_get_contents` y `file_put_contents`)

Podemos leer y escribir archivos JSON en PHP.

Ejemplo 6: Leer un archivo JSON

```
$json = file_get_contents('datos.json'); // Lee el contenido del archivo
$data = json_decode($json, true); // Convierte JSON a array
print_r($data);
```

- **Esto carga los datos de `datos.json` en un array PHP.**

Ejemplo 7: Guardar datos en un archivo JSON

```
$usuario = [
    "nombre" => "Ana",
    "edad" => 25,
    "email" => "ana@example.com"
];

$json = json_encode($usuario, JSON_PRETTY_PRINT);
file_put_contents('datos.json', $json); // Guarda el JSON en un archivo
```

📌 **Esto guarda los datos en `datos.json`, creando o sobrescribiendo el archivo.**

6️⃣ Enviar y recibir JSON en APIs (Ejemplo con `Content-Type`)

Ejemplo 8: Devolver JSON en una API con PHP

```php
header('Content-Type: application/json'); // Indica que la respuesta es JSON

$usuarios = [
    ["nombre" => "Carlos", "edad" => 28],
    ["nombre" => "Laura", "edad" => 35]
];

echo json_encode($usuarios);
```

◇ **Salida JSON en el navegador o en una API:**

```
[
    {"nombre": "Carlos", "edad": 28},
    {"nombre": "Laura", "edad": 35}
]
```

📌 **Cuando PHP actúa como API, usa `header('Content-Type: application/json');` para indicar que la respuesta es JSON.**

☑️ Resumen

✔ `json_encode()` → Convierte arrays y objetos PHP a JSON.
✔ `json_decode()` → Convierte JSON a array u objeto PHP.
✔ **Opciones como `JSON_PRETTY_PRINT` hacen JSON más legible.**
✔ `json_last_error()` permite detectar errores en JSON.
✔ `file_get_contents()` **y** `file_put_contents()` permiten leer y escribir archivos JSON.
✔ **Las APIs en PHP devuelven JSON usando `header('Content-Type: application/json');`.**

🚀 **¡Ahora entiendes cómo trabajar con JSON en PHP de manera eficiente!** 🔥