

# SW-II

# SISTEMAS WEB II

---

Prof. Anderson Vanin

AULA 05 – JSON e PHP

# Trabalhando com JSON em PHP

# JSON (JavaScript Object Notation)

**JSON** (JavaScript Object Notation) é um modelo para **armazenamento e transmissão** de informações no **formato texto**. Apesar de muito simples, tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o parsing dessas informações.

O ambiente PHP, na versão 5.2.0 ou superior, oferece um parser bastante simples e interessante para a manipulação de dados estruturados no formato JSON. Basicamente, você precisa utilizar apenas três funções: "**json\_decode**", "**json\_encode**", e "**json\_last\_error**".

# Decodificando JSON


A função **`json_decode`** recebe como entrada uma string codificada no formato JSON e a converte para uma variável PHP. Mostraremos como utilizar essa função através de alguns exemplos. Para começar, a Listagem 1 mostra um exemplo básico em que os dados de um funcionário (nome, idade e sexo) armazenados em uma string JSON são “importados” para um objeto PHP.

# Decodificando JSON

## Listagem 1. Função "json\_decode" – exemplo inicial

exemplo01.php

```
1  <?php
2
3  //string json contendo os dados de um funcionário
4  $json_str = '{"nome":"Jason Jones", "idade":38, "sexo": "M"}';
5
6  //faz o parsing na string, gerando um objeto PHP
7  $obj = json_decode($json_str);
8
9  //imprime o conteúdo do objeto
10 echo "nome: $obj->nome<br>";
11 echo "idade: $obj->idade<br>";
12 echo "sexo: $obj->sexo<br>";
13
14 ?>
```



nome: Jason Jones  
idade: 38  
sexo: M

# Decodificando JSON

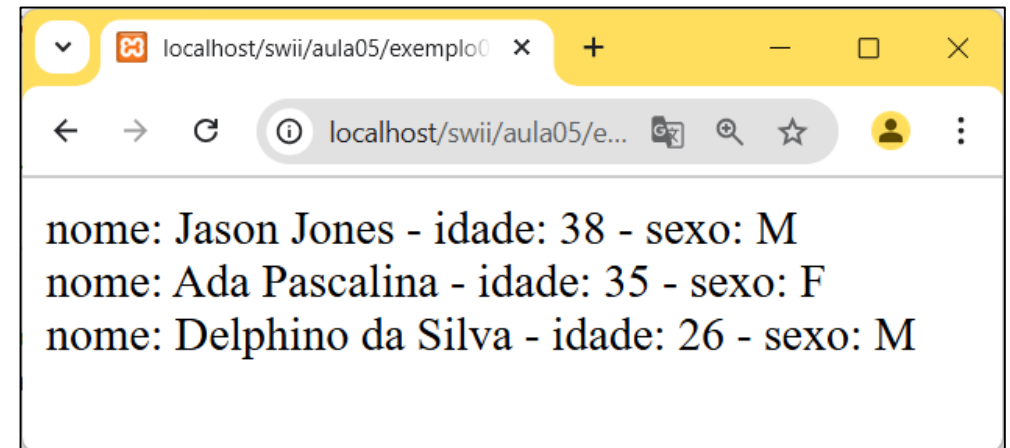
No exemplo apresentado, a string JSON possuía os dados de apenas um empregado. Mas como poderíamos proceder para trabalhar com uma string ou arquivo JSON contendo os dados de múltiplos empregados, como, por exemplo, os dados mostrados na Listagem 2.

# Decodificando JSON

## Listagem 2. Arquivo JSON contendo 3 registros

exemplo02.php

```
1  <?php
2
3      //string json (array contendo 3 elementos)
4      $json_str = '{"empregados": ['.
5          '{"nome":"Jason Jones", "idade":38, "sexo": "M"}',
6          '{"nome":"Ada Pascalina", "idade":35, "sexo": "F"}',
7          '{"nome":"Delphino da Silva", "idade":26, "sexo": "M"}'
8      ']]';
9
10     //faz o parsing da string, criando o array "empregados"
11     $jsonObj = json_decode($json_str);
12     $empregados = $jsonObj->empregados;
13
14     //navega pelos elementos do array, imprimindo cada empregado
15     foreach ( $empregados as $e )
16     {
17         echo "nome: $e->nome - idade: $e->idade - sexo: $e->sexo<br>";
18     }
19
20  ?>
```



# Decodificando JSON

O exemplo da **Listagem 4** mostra o parsing de uma string JSON um pouco mais complexa. Desta vez, o primeiro funcionário possui dois dependentes, enquanto os demais não possuem (ou seja, foi introduzido um “campo” opcional denominado “dependentes”). Também acrescentamos uma nova variável denominada “data” para armazenar data da versão do arquivo de empregados.



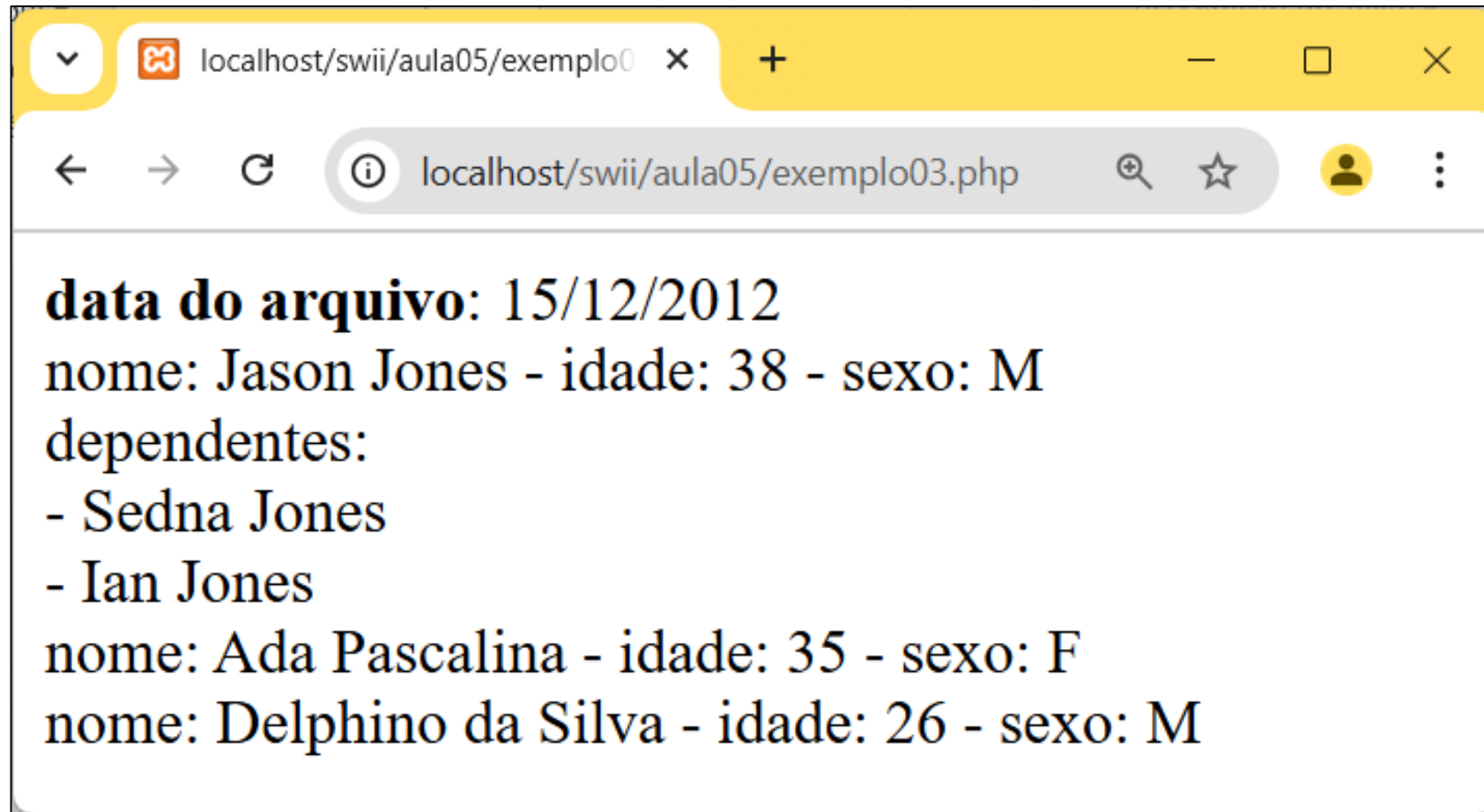
# Decodificando JSON

## Listagem 4. Função "json\_decode" – trabalhando com um objeto complexo

```
exemplo03.php
1  <?php
2      //string json
3      //agora o primeiro empregado possui dependentes e os outros não.
4      //também foi acrescentado um campo denominado "data", contendo a data do arquivo de empregados
5      $json_str = '{"empregados": [
6          '{"nome":"Jason Jones", "idade":38, "sexo": "M", "dependentes": ["Sedna Jones", "Ian Jones"]}',
7          '{"nome":"Ada Pascalina", "idade":35, "sexo": "F"}',
8          '{"nome":"Delphino da Silva", "idade":26, "sexo": "M"}'
9      ],
10     "data": "15/12/2012"}';
11
12     //faz o parsing da string, criando o array "empregados"
13     $jsonObj = json_decode($json_str);
14
15     //cria o array de empregados
16     $empregados = $jsonObj->empregados;
17
18     //imprime a data do arquivo e navega pelos elementos do array, imprimindo cada empregado.
19     //caso o empregado possua dependentes, estes também são exibidos.
20     echo "<b>data do arquivo</b>: $jsonObj->data<br/>";
21     foreach ( $empregados as $e ){
22         echo "nome: $e->nome - idade: $e->idade - sexo: $e->sexo<br/>";
23         if (property_exists($e, "dependentes")) {
24             $deps = $e->dependentes;
25             echo "dependentes: <br/>";
26             foreach ( $deps as $d ) echo "- $d<br/>";
27         }
28     }
29     ?>
```

# Decodificando JSON

**Listagem 4.** Função “json\_decode” – trabalhando com um objeto complexo



# Decodificando JSON

**Listagem 4.** Função “json\_decode” – trabalhando com um objeto complexo

Observe que desta vez foi preciso tomar um cuidado adicional. Pelo fato de “dependentes” ser um campo opcional, foi preciso utilizar a função “**property\_exists**” para checar cada empregado, ou seja para verificar se o \$e (empregado corrente) possui uma propriedade denominada “dependentes”. Apenas em caso afirmativo que os dependentes são capturados (jogados para o array “\$deps”) e exibidos na tela.

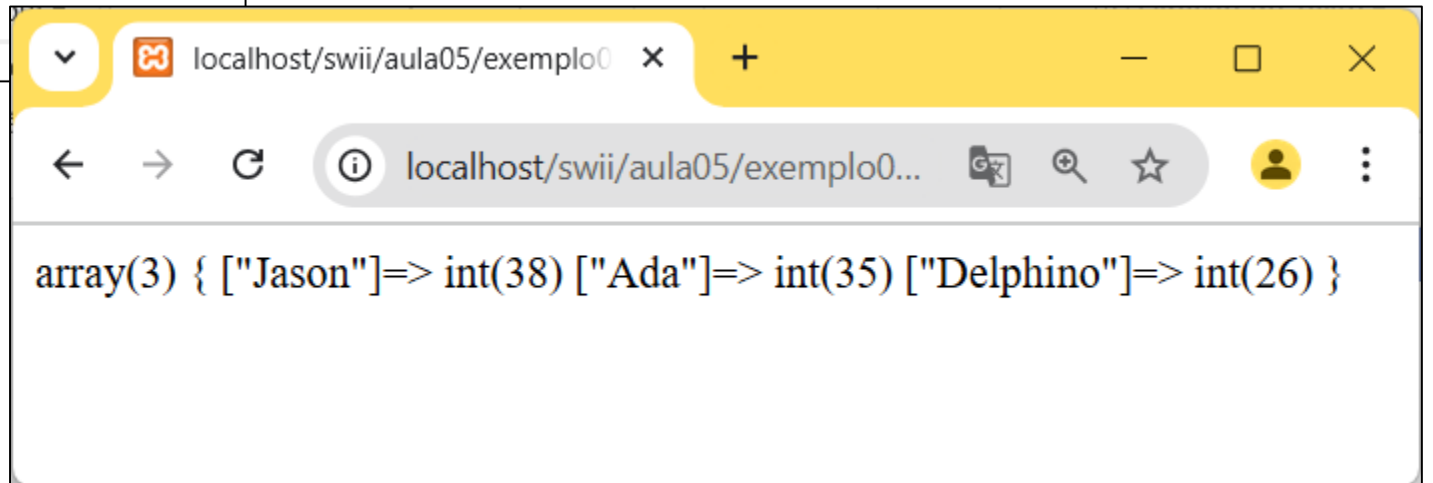
# Decodificando JSON

A função “`json_decode`” possui uma opção bastante interessante que corresponde a gerar um **array associativo** a partir da string JSON. Para fazer isso, basta especificar o valor `true`, como segundo parâmetro na chamada da função, como mostra o exemplo da **Listagem 5**.

# Decodificando JSON

**Listagem 5.** Criando um array associativo com a função "json\_decode"

```
exemplo04.php
1  <?php
2      //cria uma string no formato JSON
3      $json_str = '{"Jason":38,"Ada":35,"Delphino":26}';
4
5      //transforma a string em um array associativo
6      $json_arr = json_decode($json_str, true);
7
8      //exibe o array associativo
9      var_dump($json_arr);
10 ?>
```



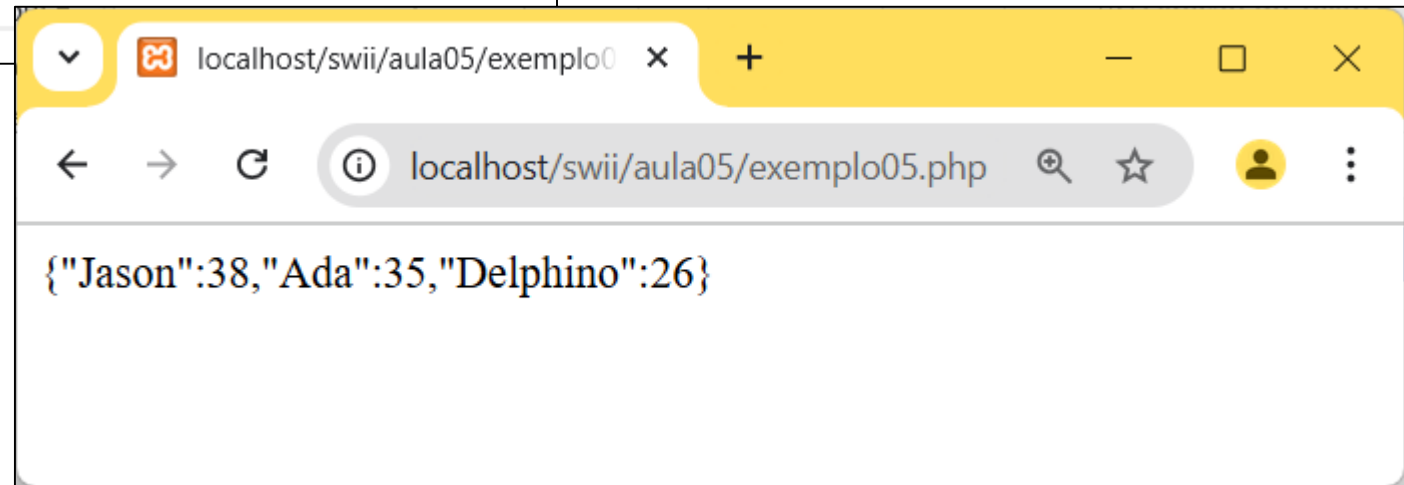
# Codificando JSON

Como o próprio nome indica, esta função faz o caminho inverso: ela converte um objeto PHP para uma string JSON. Um exemplo básico de utilização é apresentado na **Listagem 6**, onde transformamos um array associativo para uma string JSON.

# Codificando JSON

## Listagem 6. Função "json\_encode" – exemplo inicial

```
exemplo05.php
1  <?php
2      //cria o array associativo
3      $idades = array("Jason"=>38, "Ada"=>35, "Delphino"=>26);
4
5      //converte o conteúdo do array associativo para uma string JSON
6      $json_str = json_encode($idades);
7
8      //imprime a string JSON
9      echo "$json_str";
10 ?>
```

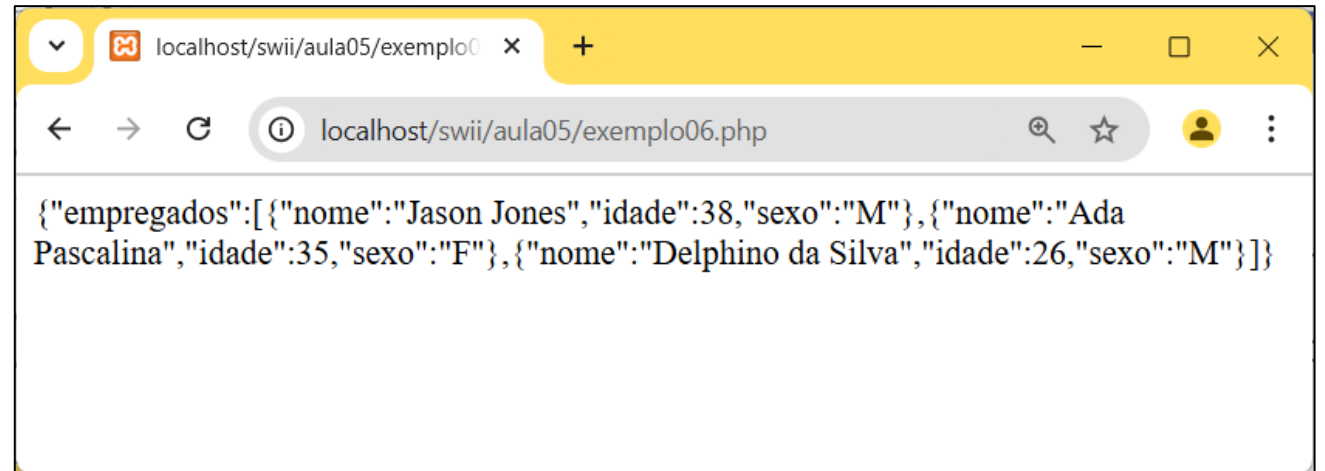


# Codificando JSON

Na **Listagem 7**, apresentamos um exemplo envolvendo um array mais complexo.

## Listagem 7. Convertendo um array com 3 empregados para uma String JSON

```
exemplo06.php
1  <?php
2  //cria um array contendo 3 empregados
3  $empregados = array('empregados' => array(
4      array(
5          'nome' => 'Jason Jones',
6          'idade' => 38,
7          'sexo' => 'M'
8      ),
9      array(
10         'nome' => 'Ada Pascalina',
11         'idade' => 35,
12         'sexo' => 'F'
13     ),
14     array(
15         'nome' => 'Delphino da Silva',
16         'idade' => 26,
17         'sexo' => 'M'
18     )
19 ));
20 //converte o conteúdo do array para uma string JSON
21 $json_str = json_encode($empregados);
22
23 //imprime a string JSON
24 echo "$json_str";
25 ?>
```





# Codificando JSON

*OBS: A Listagem 7 apresentou a conversão de uma estrutura complexa representada em um array. Porém, é importante esclarecer que também é possível converter objetos (instâncias de classes PHP) com o uso da função "json\_encode".*

# Tratamento de Erros

Para realizar o tratamento de erros o PHP disponibiliza a função ***json\_last\_error***. Esta função simplesmente retorna uma das seguintes constantes pré-definidas para indicar o erro ocorrido.

- **0 = JSON\_ERROR\_NONE**: nenhum erro ocorreu;
- **1 = JSON\_ERROR\_DEPTH**: a profundidade máxima de aninhamento de uma string JSON foi excedida (esse valor máximo é 512);
- **2 = JSON\_ERROR\_STATE\_MISMATCH**: erro de underflow ou outro tipo de estado inválido;
- **3 = JSON\_ERROR\_CTRL\_CHAR**: foi encontrado um caractere de controle no corpo da string JSON;
- **4 = JSON\_ERROR\_SYNTAX**: erro de sintaxe;
- **5 = JSON\_ERROR\_UTF8**: erro na codificação UTF-8 da string JSON;

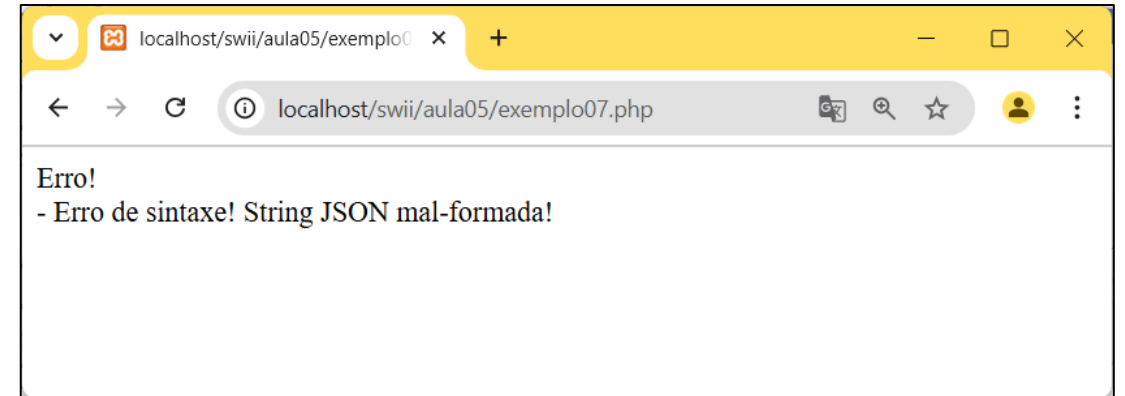
# Tratamento de Erros

Um exemplo de código para tratamento de erros é mostrado na **Listagem 8**. Neste exemplo, **deixamos propositalmente sem aspas o valor da variável sexo (M)**. Com isto, um erro de sintaxe (JSON\_ERROR\_SYNTAX) será acusado pela função "json\_last\_error".

# Tratamento de Erros

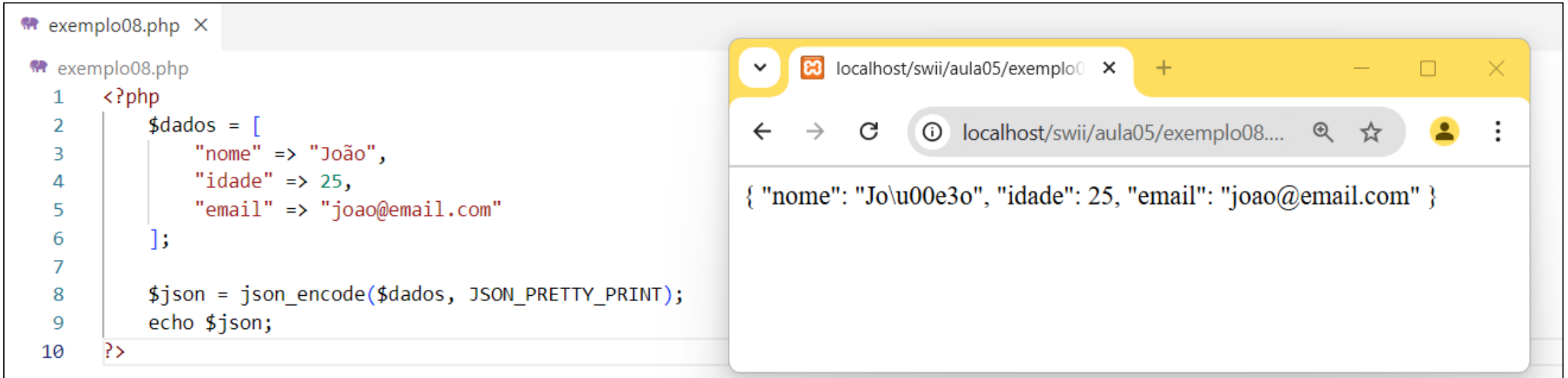
## Listagem 8. Função "json\_last\_error"

```
1 <?php
2 //string json contendo os dados de um funcionário.
3 //esta string não está bem-formada, pois o valor M não está entre aspas
4 $json_str = '{"nome":"Jason Jones", "idade":38, "sexo": M}';
5
6 //faz o parsing na string, gerando um objeto PHP
7 $obj = json_decode($json_str);
8
9 //testa se houve erro no parsing! Vai acusar erro de string mal-formada (JSON_ERROR_SYNTAX)
10 if (json_last_error() == 0) {
11     echo '- Nao houve erro! O parsing foi perfeito';
12 }
13 else {
14     echo 'Erro!<br>';
15     switch (json_last_error()) {
16         case JSON_ERROR_DEPTH:
17             echo ' - profundidade maxima excedida';
18             break;
19         case JSON_ERROR_STATE_MISMATCH:
20             echo ' - state mismatch';
21             break;
22         case JSON_ERROR_CTRL_CHAR:
23             echo ' - Caracter de controle encontrado';
24             break;
25         case JSON_ERROR_SYNTAX:
26             echo ' - Erro de sintaxe! String JSON mal-formada!';
27             break;
28         case JSON_ERROR_UTF8:
29             echo ' - Erro na codificação UTF-8';
30             break;
31         default:
32             echo ' ⚠ Erro desconhecido';
33             break;
34     }
35 }
36 ?>
```



# Outros exemplos

# Convertendo um Array para JSON



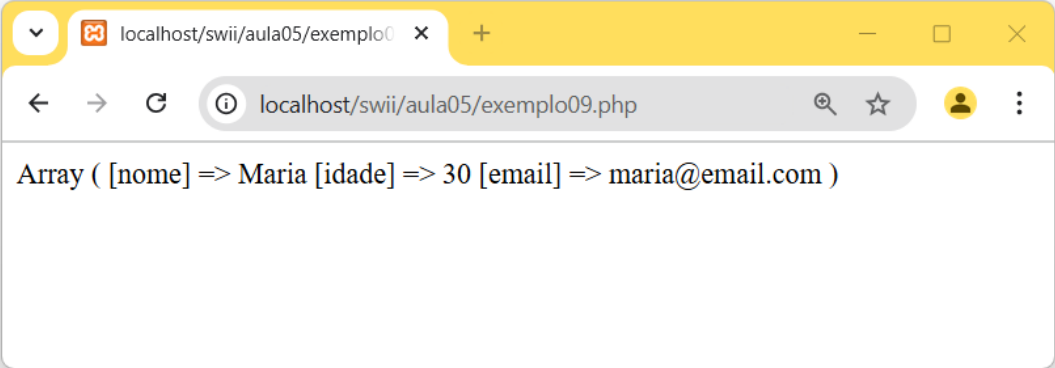
The image shows a PHP script in a code editor and its output in a web browser. The PHP script, named `exemplo08.php`, defines an array `$dados` with three elements: `"nome" => "João"`, `"idade" => 25`, and `"email" => "joao@email.com"`. It then uses `json_encode` with the `JSON_PRETTY_PRINT` flag to format the array as JSON and echoes the result.

```
1 <?php
2 $dados = [
3     "nome" => "João",
4     "idade" => 25,
5     "email" => "joao@email.com"
6 ];
7
8 $json = json_encode($dados, JSON_PRETTY_PRINT);
9 echo $json;
10 ?>
```

The browser window shows the output of the script at the URL `localhost/swii/aula05/exemplo08...`. The output is a JSON object: `{ "nome": "Jo\u00e3o", "idade": 25, "email": "joao@email.com" }`.

# Convertendo JSON para Array (ou Objeto)

```
exemplo09.php
1  <?php
2      $json = '{"nome": "Maria", "idade": 30, "email": "maria@email.com"}';
3
4      $dados = json_decode($json, true); // O segundo parâmetro "true" converte para array
5      print_r($dados);
6  ?>
```



Se quiser um objeto em vez de um array, remova o true:

```
$dados = json_decode($json);
echo $dados->nome; // Maria
```

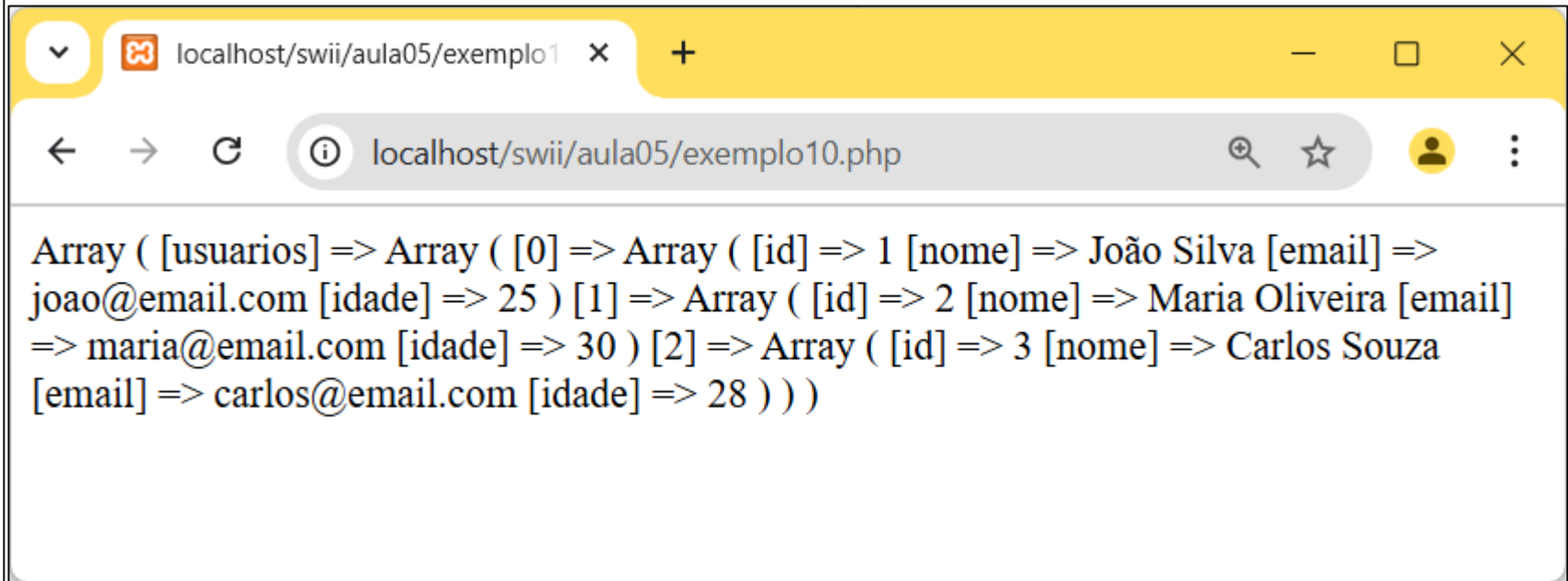
# Ler um JSON de um arquivo

exemplo10.php

```
1 <?php
2     $conteudo = file_get_contents("dados.json");
3     $dados = json_decode($conteudo, true);
4
5     print_r($dados);
6 ?>
```

dados.json > ...

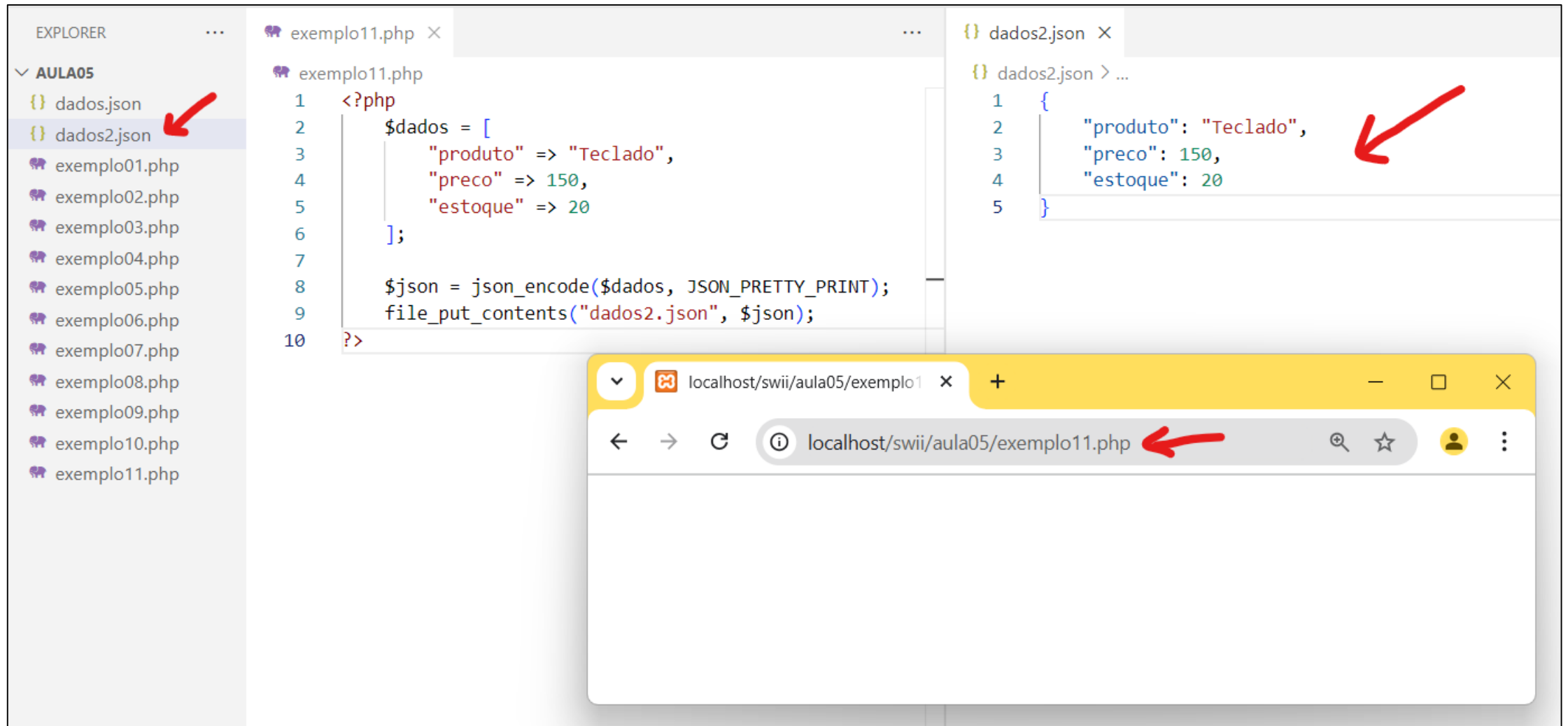
```
1 {
2     "usuarios": [
3         {
4             "id": 1,
5             "nome": "João Silva",
6             "email": "joao@email.com",
7             "idade": 25
8         },
9         {
10            "id": 2,
11            "nome": "Maria Oliveira",
12            "email": "maria@email.com",
13            "idade": 30
14        },
15        {
16            "id": 3,
17            "nome": "Carlos Souza",
18            "email": "carlos@email.com",
19            "idade": 28
20        }
21    ]
22 }
```



```
Array ( [usuarios] => Array ( [0] => Array ( [id] => 1 [nome] => João Silva [email] => joao@email.com [idade] => 25 ) [1] => Array ( [id] => 2 [nome] => Maria Oliveira [email] => maria@email.com [idade] => 30 ) [2] => Array ( [id] => 3 [nome] => Carlos Souza [email] => carlos@email.com [idade] => 28 ) ) )
```



# Salvar um Array como JSON em um Arquivo



# Exercícios

## 1 - Criar um JSON a partir de um array e salvar em um arquivo

Crie um *array* com informações de 3 produtos (**nome, preço e quantidade**). Converta o *array* para JSON e salve no arquivo **produtos.json**.

### Passos:

- a) Criar um *array* associativo com os produtos.
- b) Converter para JSON com *json\_encode()*.
- c) Salvar o JSON em um arquivo com *file\_put\_contents()*.

# Exercícios

## 2 - Ler um arquivo JSON e exibir os dados na tela

Crie um arquivo **usuarios.json** contendo uma lista de usuários com os atributos *id*, *nome* e *email*.

Depois, faça um script PHP que:

- a) Leia o arquivo JSON usando *file\_get\_contents()*.
- b) Converta o JSON em um *array* PHP com *json\_decode()*.
- c) Exiba os nomes e emails de todos os usuários.

# Exercícios

## 3 - Adicionar um novo item a um arquivo JSON existente

Usando o arquivo **produtos.json** do exercício 1, faça um script que:

- a) Leia o JSON do arquivo e o converta para um *array*.
- b) Adicione um novo produto ao *array*.
- c) Converta o *array* atualizado para JSON e salve de volta no arquivo.

# Exercícios

## 4 - Buscar um item dentro do JSON

Usando **usuarios.json**, crie um script que:

- a) Pergunte ao usuário um email (via `$_GET` ou variável fixa).
- b) Busque no array de usuários um que tenha esse email.
- c) Exiba os dados do usuário encontrado ou uma mensagem de erro.

# Exercícios

## 5 - Remover um item de um arquivo JSON

Usando **produtos.json**, faça um script que:

- a) Leia os produtos do JSON.
- b) Remova um produto específico (exemplo: pelo nome).
- c) Salve o novo JSON atualizado no arquivo.