



# Introdução ao PHP



PHP continua a ser uma ferramenta vital no arsenal de um desenvolvedor web devido à sua flexibilidade, poder e a enorme comunidade de suporte. Quer você esteja criando um simples site pessoal ou uma complexa aplicação empresarial, o PHP oferece as ferramentas e recursos necessários para realizar seu projeto com eficiência e eficácia.



## Primeiro código

```
<?php
// Comentário de uma linha

/*
    Comentário de múltiplas linhas
*/

echo "Olá, Mundo!"; // Exibe "Olá, Mundo!"
```



## Variáveis.

Uma variável na programação é um nome simbólico associado a um valor e cuja principal função é armazenar dados que podem ser utilizados e manipulados ao longo da execução de um programa. Em termos simples, uma variável atua como um recipiente para guardar informações que podem ser alteradas ou referenciadas posteriormente.



## Conceitos Fundamentais de Variáveis

**Nome da Variável:** É a identificação que damos à variável para referenciá-la no código. Geralmente, os nomes de variáveis devem ser descritivos para facilitar a leitura e manutenção do código.

**Tipo de Dado:** Refere-se ao tipo de valor que a variável pode armazenar, como números inteiros, números de ponto flutuante, strings, booleanos, entre outros. Algumas linguagens de programação exigem que o tipo de dado seja explicitamente declarado, enquanto outras determinam o tipo automaticamente.

**Valor:** É a informação ou dado que a variável contém. Este valor pode ser modificado durante a execução do programa.



## Declaração de Variáveis no PHP

No PHP, uma variável é representada por um nome que começa com o símbolo \$, seguido por uma combinação de letras, números e. Elas são usadas para armazenar valores que podem ser alterados e reutilizados ao longo do script.



## Declaração de Variáveis no PHP

```
<?php
$nome = "João"; // String
$idade = 30; // Inteiro
$altura = 1.75; // Float
$isAdult = true; // Booleano
```



## Tipos de Dados Comuns

**Inteiros (Integers):** Números sem ponto decimal.

```
$idade = 30;
```

**Ponto Flutuante (Floats):** Números com ponto decimal.

```
$altura = 1.75;
```

**Strings:** Sequências de caracteres.

```
$nome = "João";
```



## Tipos de Dados Comuns

**Booleanos (Booleans):** Valores verdadeiros ou falsos.

```
$isAdulto = true;
```

**Arrays:** Coleções ordenadas de valores.

```
$cores = array("vermelho", "verde", "azul");
```





## Escopo de Variáveis

**Escopo Global:** Variáveis que podem ser acessadas em qualquer parte do programa.

**Escopo Local:** Variáveis que só podem ser acessadas dentro do bloco de código ou função onde foram declaradas.

**Variáveis Estáticas:** Em algumas linguagens, variáveis dentro de uma função podem manter seu valor entre diferentes chamadas da função.



## Conclusão

**Variáveis são um dos conceitos mais fundamentais na programação, permitindo que dados sejam armazenados, manipulados e reutilizados ao longo da execução de um programa. Compreender como declarar, nomear e utilizar variáveis é essencial para qualquer programador, independentemente da linguagem de programação utilizada.**



## O que é uma Função no PHP

**Uma função no PHP é um bloco de código que pode ser chamado repetidamente em diferentes partes de um programa, permitindo a reutilização do código e a modularização do programa. Funções podem aceitar parâmetros de entrada, executar operações com esses parâmetros e retornar um valor de saída.**



## Declaração de Funções

Para declarar uma função em PHP, utiliza-se a palavra-chave `function`, seguida pelo nome da função, um conjunto de parênteses (que pode conter parâmetros) e um bloco de código entre chaves `{}`.

```
<?php
function nomeDaFuncao() {
    // código da função
}
```



## Parâmetros e Argumentos

Funções podem receber parâmetros, que são variáveis passadas para a função no momento de sua chamada. Esses parâmetros são definidos dentro dos parênteses na declaração da função.

```
<?php
function saudacao($nome) {
    echo "Olá, $nome!";
}
```



## Valores de Retorno

Funções podem retornar valores usando a palavra-chave **return**. Isso permite que o resultado da função seja armazenado em uma variável ou usado diretamente.

```
<?php
function soma($a, $b) {
    return $a + $b;
}

$resultado = soma(5, 3);
echo $resultado; // Saída: 8
```



## E “SE” ou IF

comando `if` em PHP é usado para executar um bloco de código condicionalmente, com base em uma expressão avaliada como verdadeira ou falsa. Ele é um dos componentes fundamentais do controle de fluxo em PHP, permitindo que o código tome decisões dinâmicas durante a execução.



## Estrutura Básica do **if**

A estrutura básica de um comando **if** é a seguinte:

```
<?php
if (condição) {
    // bloco de código a ser executado se a condição for verdadeira
}
```





## Exemplo Básico

```
<?php
$idade = 20;

if ($idade >= 18) {
    echo "Você é maior de idade.";
}
```



## Comando **if-else**

O comando **if** pode ser expandido com **else** para especificar um bloco de código alternativo que será executado se a condição for falsa.

```
<?php
$idade = 16;

if ($idade >= 18) {
    echo "Você é maior de idade.";
} else {
    echo "Você é menor de idade.";
}
```

## Comando `if-elseif-else`

Para verificar múltiplas condições, você pode usar `elseif` (ou `else if`).

```
<?php
$nota = 75;

if ($nota >= 90) {
    echo "A nota é A.";
} elseif ($nota >= 80) {
    echo "A nota é B.";
} elseif ($nota >= 70) {
    echo "A nota é C.";
} else {
    echo "A nota é D ou inferior.";
}
```



## Comando `if-elseif-else`

Para verificar múltiplas condições, você pode usar `elseif` (ou `else if`).

```
<?php
$nota = 75;

if ($nota >= 90) {
    echo "A nota é A.";
} elseif ($nota >= 80) {
    echo "A nota é B.";
} elseif ($nota >= 70) {
    echo "A nota é C.";
} else {
    echo "A nota é D ou inferior.";
}
```



## Uso de Operadores Lógicos

Você pode usar operadores lógicos para combinar múltiplas condições em uma única expressão `if`.

### Operador **AND**

```
<?php
$idade = 25;
$renda = 3000;

if ($idade >= 18 && $renda >= 2000) {
    echo "Você atende aos critérios.";
}
```



## Operador OR

```
<?php
$idade = 16;
$renda = 3000;

if ($idade >= 18 || $renda >= 2000) {
    echo "Você atende a pelo menos um dos critérios.";
}
```



## Operador NOT

```
<?php
$idade = 16;

if (!($idade >= 18)) {
    echo "Você é menor de idade.";
}
```



## Operador Ternário

O operador ternário é uma forma compacta de escrever uma declaração `if-else`.

```
<?php
$idade = 20;
$mensagem = ($idade >= 18) ? "Você é maior de idade." : "Você é menor de idade.";
echo $mensagem;
```





## XOR (xor)

**Descrição:** Retorna verdadeiro se uma, e apenas uma, das expressões for verdadeira.

```
<?php
$a = true;
$b = false;

if ($a xor $b) {
    echo "Uma, e apenas uma, das expressões é verdadeira.";
}
```



## Tipos de Operadores Lógicos

1. **AND** (&& e **and**)
2. **OR** (|| e **or**)
3. **NOT** (!)
4. **XOR** (**xor**)



## Protocolo HTTP



O protocolo HTTP (Hypertext Transfer Protocol) é o principal protocolo de comunicação da web. Ele é responsável pela transmissão de informações entre clientes (como navegadores web) e servidores.



## O modelo OSI

**OSI (Open Systems Interconnection) é uma estrutura de referência para a comunicação de sistemas de computação. Ele define sete camadas diferentes que descrevem como dados devem ser transmitidos entre dois pontos em uma rede. Cada camada tem uma função específica e se comunica com as camadas adjacentes. Aqui está uma breve descrição de cada uma das sete camadas:**



**7 Camada de Aplicação (Application Layer):** Fornece serviços de rede diretamente às aplicações dos usuários. É a camada mais próxima do usuário final. Protocolos como HTTP, FTP, SMTP e DNS operam nesta camada.



HTTP Client



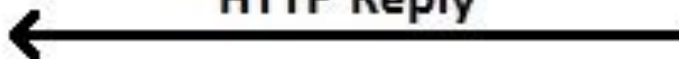
HTTP Server



HTTP Request



HTTP Reply





**1 Camada Física (Physical Layer):** Esta é a camada mais baixa e lida com a transmissão física de dados brutos através de um meio de comunicação, como cabos de rede. Ela especifica os parâmetros elétricos, ópticos e mecânicos do sistema, incluindo conectores, tensões, sincronização, etc.

**2 Camada de Enlace de Dados (Data Link Layer):** Responsável pela transferência de dados entre dois nós diretamente conectados. Garante que os dados sejam transferidos de forma livre de erros. Esta camada inclui protocolos como Ethernet e PPP (Point-to-Point Protocol).



**3 Camada de Rede (Network Layer):** Controla a operação da sub-rede e a movimentação de pacotes entre dispositivos. Ela define o endereçamento lógico e o roteamento. Protocolos como IP (Internet Protocol) operam nesta camada.

**4 Camada de Transporte (Transport Layer):** Garante a transferência confiável de dados entre sistemas finais, controla a segmentação dos dados e a recomposição dos segmentos. Protocolos como TCP (Transmission Control Protocol) e UDP (User Datagram Protocol) são encontrados aqui.





**5 Camada de Sessão (Session Layer):** Estabelece, gerencia e encerra sessões entre duas máquinas. Isso inclui o controle do diálogo e a sincronização.

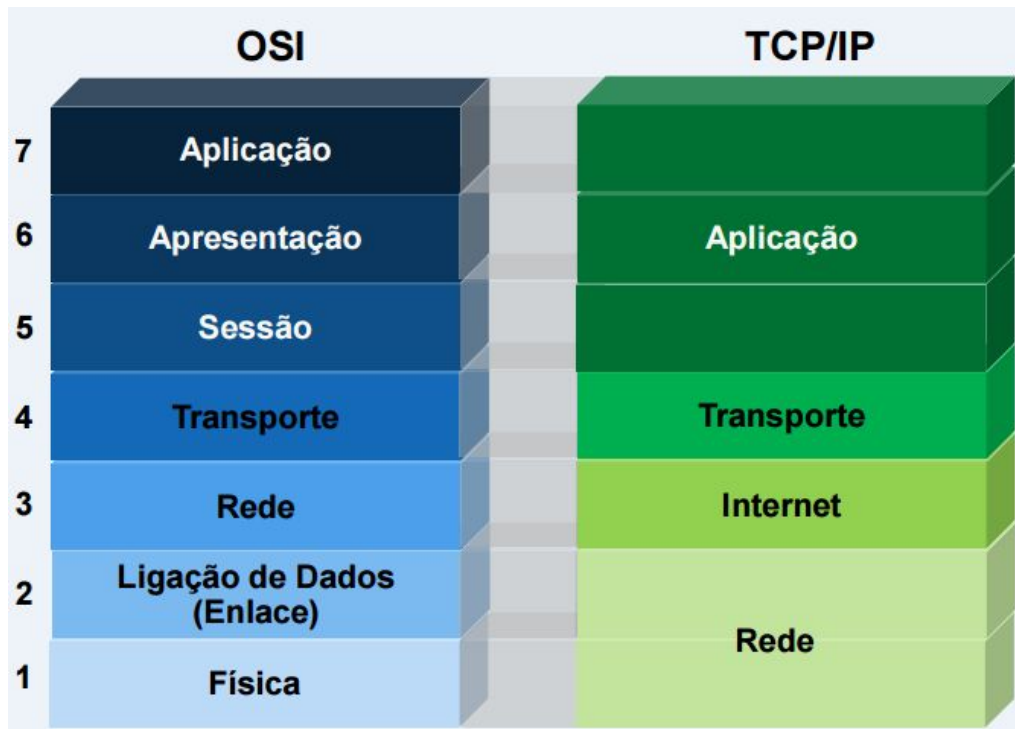
**6 Camada de Apresentação (Presentation Layer):** Atua como tradutora de dados, convertendo dados entre o formato usado pela rede e o formato necessário para a aplicação. Inclui a criptografia e a compressão de dados.





## OSI vs TCP/IP

O modelo TCP/IP (Transmission Control Protocol/Internet Protocol) é uma estrutura de comunicação de rede usada na Internet e em redes similares. Ele foi desenvolvido pela DARPA (Defense Advanced Research Projects Agency) dos EUA durante os anos 1970 e 1980. Ao contrário do modelo OSI de sete camadas, o modelo TCP/IP tem quatro camadas principais, cada uma das quais corresponde aproximadamente a um ou mais níveis do modelo OSI. Estas camadas são: a Camada de Acesso à Rede, a Camada de Internet, a Camada de Transporte e a Camada de Aplicação.





## Métodos HTTP

Os métodos HTTP (HyperText Transfer Protocol) são um conjunto de comandos que definem as ações que podem ser realizadas sobre os recursos em um servidor web. Eles são utilizados para enviar e receber dados entre clientes (como navegadores web) e servidores. Aqui estão os principais métodos HTTP e suas funcionalidades:



## Métodos HTTP

### GET

- **Função:** Recuperar dados de um servidor.
- **Uso Comum:** Obter páginas web, imagens, vídeos, e outros tipos de dados.
- **Comportamento:** Não altera o estado do recurso no servidor; é um método idempotente (chamadas repetidas produzem o mesmo efeito).



## Métodos HTTP

### POST

- **Função:** Enviar dados ao servidor para criar ou atualizar um recurso.
- **Uso Comum:** Submeter formulários, fazer uploads de arquivos.
- **Comportamento:** Pode alterar o estado do servidor ou criar novos recursos; não é idempotente (chamadas repetidas podem produzir efeitos diferentes).



## Métodos HTTP

### PUT

- **Função:** Atualizar um recurso existente ou criar um novo recurso com os dados fornecidos.
- **Uso Comum:** Atualizar registros em um banco de dados.
- **Comportamento:** Idempotente; chamadas repetidas com o mesmo dado produzem o mesmo efeito.



## Métodos HTTP

### DELETE

- **Função:** Remover um recurso do servidor.
- **Uso Comum:** Excluir registros em um banco de dados.
- **Comportamento:** Idempotente; a exclusão repetida de um recurso terá o mesmo efeito.





## Métodos HTTP

### PATCH

- **Função:** Aplicar modificações parciais a um recurso.
- **Uso Comum:** Atualizar parcialmente registros em um banco de dados.
- **Comportamento:** Não necessariamente idempotente.



## Métodos HTTP

- Metodos HTTP suportados pelo PHP
- Todos
- Metodos HTTP suportador pelo JavaScript





## HTTP status code

- Os códigos de resposta HTTP indicam o resultado de uma requisição HTTP ao servidor. Eles são agrupados em cinco classes, cada uma representando um tipo diferente de resposta. Aqui estão os códigos de resposta HTTP mais comuns e seus significados:



## HTTP status code

### Códigos de Resposta 1xx (Informativos)

- **100 Continue:** O servidor recebeu a parte inicial da solicitação e o cliente deve continuar.
- **101 Switching Protocols:** O servidor está mudando os protocolos conforme solicitado pelo cliente.
- **102 Processing:** O servidor recebeu a solicitação e está processando-a, mas ainda não tem uma resposta final.
-



## HTTP status code

### Códigos de Resposta 1xx (Informativos)

- **100 Continue:** O servidor recebeu a parte inicial da solicitação e o cliente deve continuar.
- **101 Switching Protocols:** O servidor está mudando os protocolos conforme solicitado pelo cliente.
- **102 Processing:** O servidor recebeu a solicitação e está processando-a, mas ainda não tem uma resposta final.
-



# HTTP status code

## Códigos de Resposta 2xx (Sucesso)

- **200 OK:** A solicitação foi bem-sucedida. O significado da resposta depende do método HTTP usado.
- **201 Created:** A solicitação foi bem-sucedida e um novo recurso foi criado como resultado.
- **202 Accepted:** A solicitação foi aceita para processamento, mas o processamento ainda não foi concluído.
- **203 Non-Authoritative Information:** A solicitação foi bem-sucedida, mas a resposta pode ter sido modificada por um servidor intermediário.
- **204 No Content:** A solicitação foi bem-sucedida, mas não há conteúdo para enviar na resposta.
- **205 Reset Content:** A solicitação foi bem-sucedida, e o agente do usuário deve redefinir a visualização que causou a solicitação.
- **206 Partial Content:** O servidor está entregando apenas parte do recurso devido a um cabeçalho de intervalo enviado pelo cliente.



# HTTP status code

## Códigos de Resposta 2xx (Sucesso)

- **200 OK:** A solicitação foi bem-sucedida. O significado da resposta depende do método HTTP usado.
- **201 Created:** A solicitação foi bem-sucedida e um novo recurso foi criado como resultado.
- **202 Accepted:** A solicitação foi aceita para processamento, mas o processamento ainda não foi concluído.
- **203 Non-Authoritative Information:** A solicitação foi bem-sucedida, mas a resposta pode ter sido modificada por um servidor intermediário.
- **204 No Content:** A solicitação foi bem-sucedida, mas não há conteúdo para enviar na resposta.
- **205 Reset Content:** A solicitação foi bem-sucedida, e o agente do usuário deve redefinir a visualização que causou a solicitação.
- **206 Partial Content:** O servidor está entregando apenas parte do recurso devido a um cabeçalho de intervalo enviado pelo cliente.



## HTTP status code

### Códigos de Resposta 3xx (Redirecionamento)

- **300 Multiple Choices:** Há várias opções para o recurso solicitado.
- **301 Moved Permanently:** O recurso solicitado foi movido permanentemente para uma nova URL.
- **302 Found:** O recurso solicitado foi movido temporariamente para uma nova URL.





## HTTP status code

### Códigos de Resposta 3xx (Redirecionamento)


- **300 Multiple Choices:** Há várias opções para o recurso solicitado.
- **301 Moved Permanently:** O recurso solicitado foi movido permanentemente para uma nova URL.
- **302 Found:** O recurso solicitado foi movido temporariamente para uma nova URL.



## HTTP status code

### Códigos de Resposta 4xx (Erro do Cliente)

- **400 Bad Request:** A solicitação não pôde ser entendida ou foi malformada.
- **401 Unauthorized:** A solicitação requer autenticação do cliente.
- **402 Payment Required:** Reservado para uso futuro. Originalmente criado para sistemas de pagamento digital.
- **403 Forbidden:** O servidor entendeu a solicitação, mas se recusa a autorizá-la.
- **404 Not Found:** O recurso solicitado não foi encontrado no servidor.
- **405 Method Not Allowed:** O método HTTP usado não é permitido para o recurso solicitado.

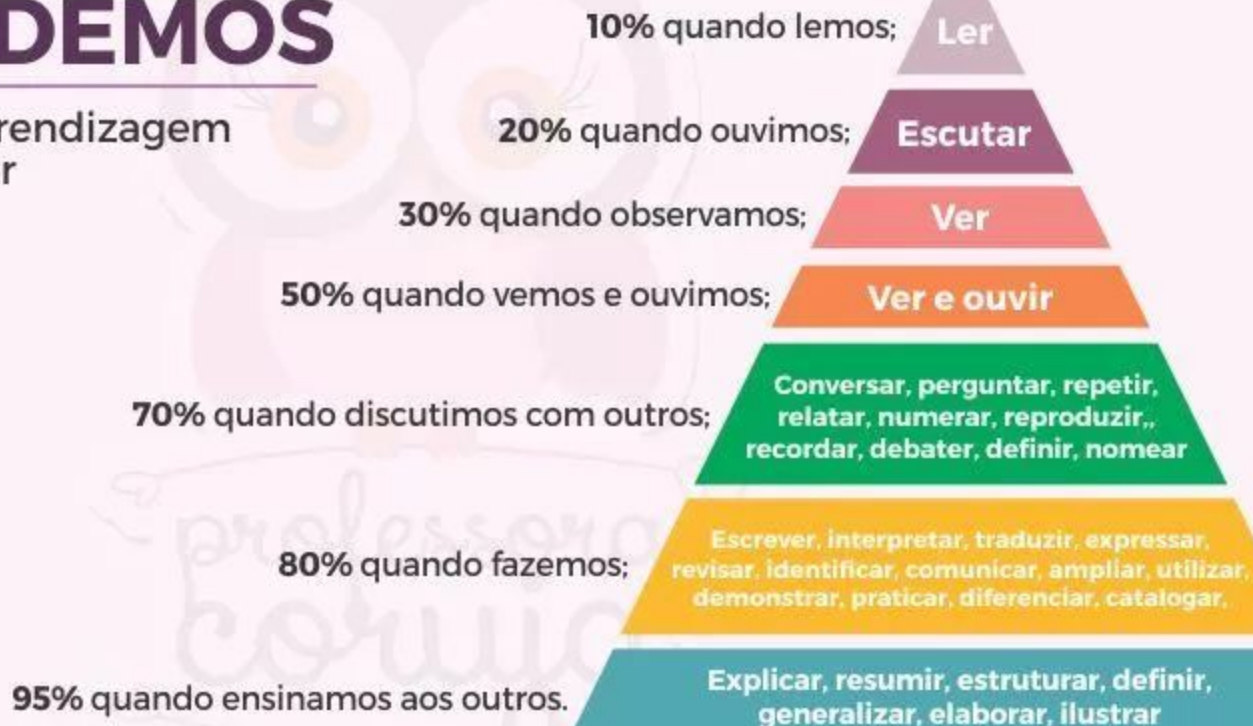
- 
- O psiquiatra americano William Glasser (1925-2013) aplicou sua *teoria da escolha* para a educação. De acordo com esta teoria, o professor é um *guia* para o aluno e não um *chefe* .
  - Glasser explica que não se deve trabalhar apenas com memorização, porque a maioria dos alunos simplesmente esquecem os conceitos após a aula. Em vez disso, o psiquiatra sugere que os alunos aprendem efetivamente com você, *fazendo* .
  - Além disso, Glasser também explica o grau de aprendizagem de acordo com a técnica utilizada.
  - Esta é a pirâmide de aprendizagem:

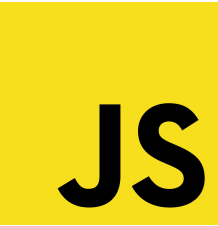


# COMO APRENDEMOS

A pirâmide de aprendizagem de William Glasser

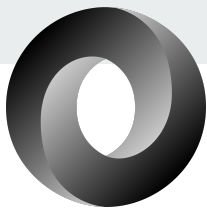
Aprendemos...





## Descrição

- Arrays JavaScript começam com índice zero: o primeiro elemento de um array está na posição 0 e o último elemento está na posição equivalente ao valor da propriedade [length](#)



## JSON

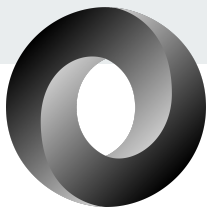
- JSON (JavaScript Object Notation) é um formato leve de troca de dados, fácil de ler e escrever tanto para humanos quanto para máquinas. Ele é amplamente utilizado em aplicações web para enviar e receber dados entre um cliente e um servidor. Aqui está uma explicação detalhada sobre o que é JSON:





## Exemplo de um objeto JSON

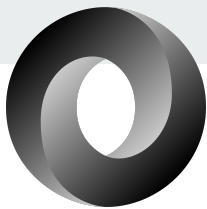
```
const clientes = [  
  {  
    nome: 'Chuck Norris',  
    cpf: '000.000.000-00',  
    idade: 30,  
    peso: 197.35,  
    ativo: true  
  },  
  {  
    nome: 'Bruce Willys',  
    cpf: '000.000.000-00',  
    idade: 30,  
    peso: 197.35,  
    ativo: true  
  }  
];  
console.log(clientes);
```



## Exemplo de um Array PHP

```
$cliente = [  
    [  
        'nome' => 'Chuck Norris',  
        'cpf' => '000.000.000-00',  
        'idade' => 30,  
        'peso' => 197.35,  
        'ativo' => true  
    ],  
    [  
        'nome' => 'Bruce Willys',  
        'cpf' => '000.000.000-00',  
        'idade' => 30,  
        'peso' => 197.35,  
        'ativo' => true  
    ]  
];
```

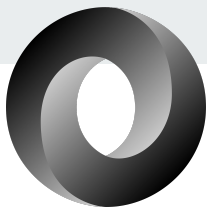




## Exemplo de JSON de retorno

```
{  
  "cep": "01001-000",  
  "logradouro": "Praça da Sé",  
  "complemento": "lado ímpar",  
  "unidade": "",  
  "bairro": "Sé",  
  "localidade": "São Paulo",  
  "uf": "SP",  
  "estado": "São Paulo",  
  "regiao": "Sudeste",  
  "ibge": "3550308",  
  "gia": "1004",  
  "ddd": "11",  
  "siafi": "7107"  
}
```





## Exemplo de JSON de retorno

```
1  <?php
2  #Define o cabeçalho para JSON, será o tipo de resposta
3  header('Content-Type: application/json');
4
5  #Cria um array associativo para retornar como JSON
6  $data = [
7      'status' => 'success',
8      'msg' => 'Dados retornados com sucesso',
9      'id' => 1
10 ];
11 #Converte o array em JSON e envia a resposta
12 echo json_encode($data);
```

