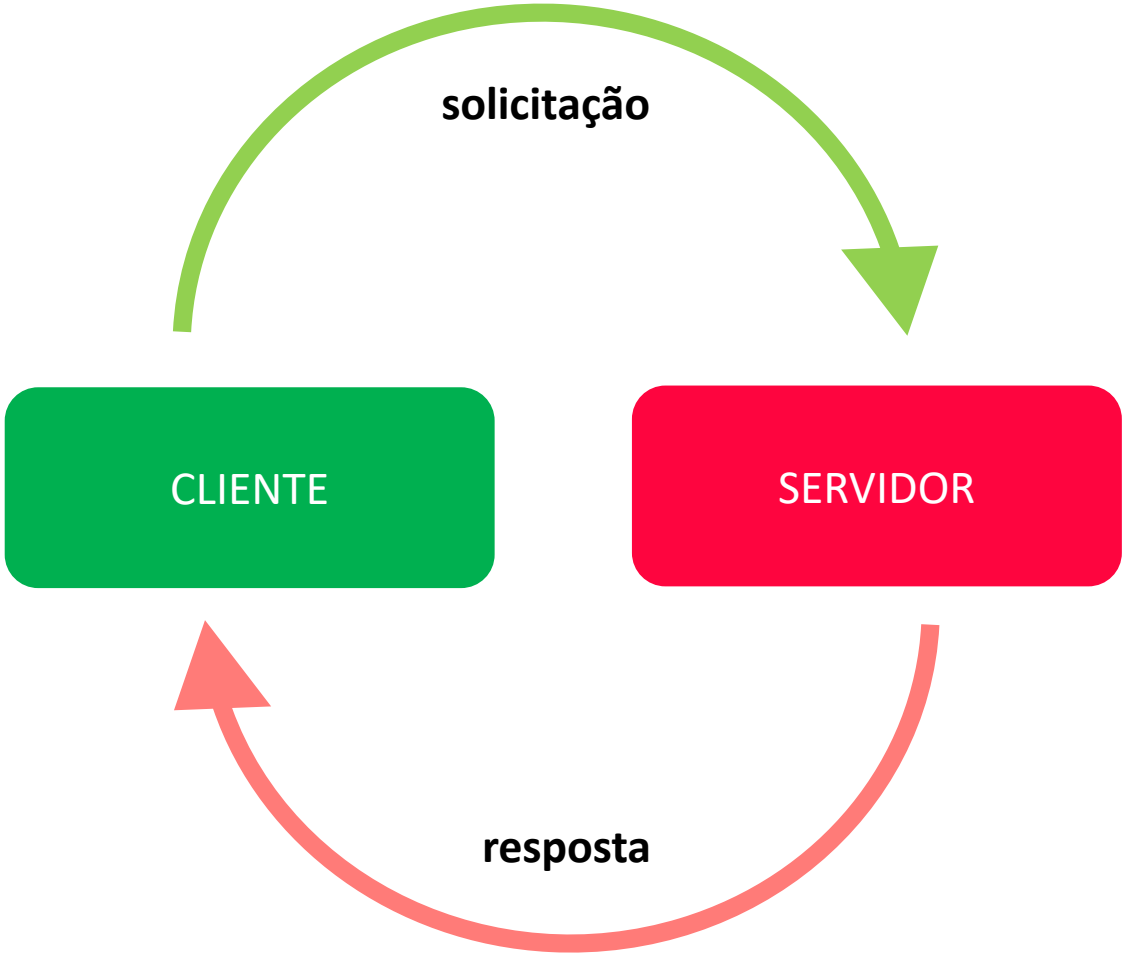
Comunicação Cliente x Servidor

Trabalhando com a resposta do Servidor

Prof. Eduardo Lino
eduardo.lino@pucpr.br

Cliente x Servidor



Cliente x Servidor

CLIENTE

SERVIDOR

Cliente x Servidor

CLIENTE

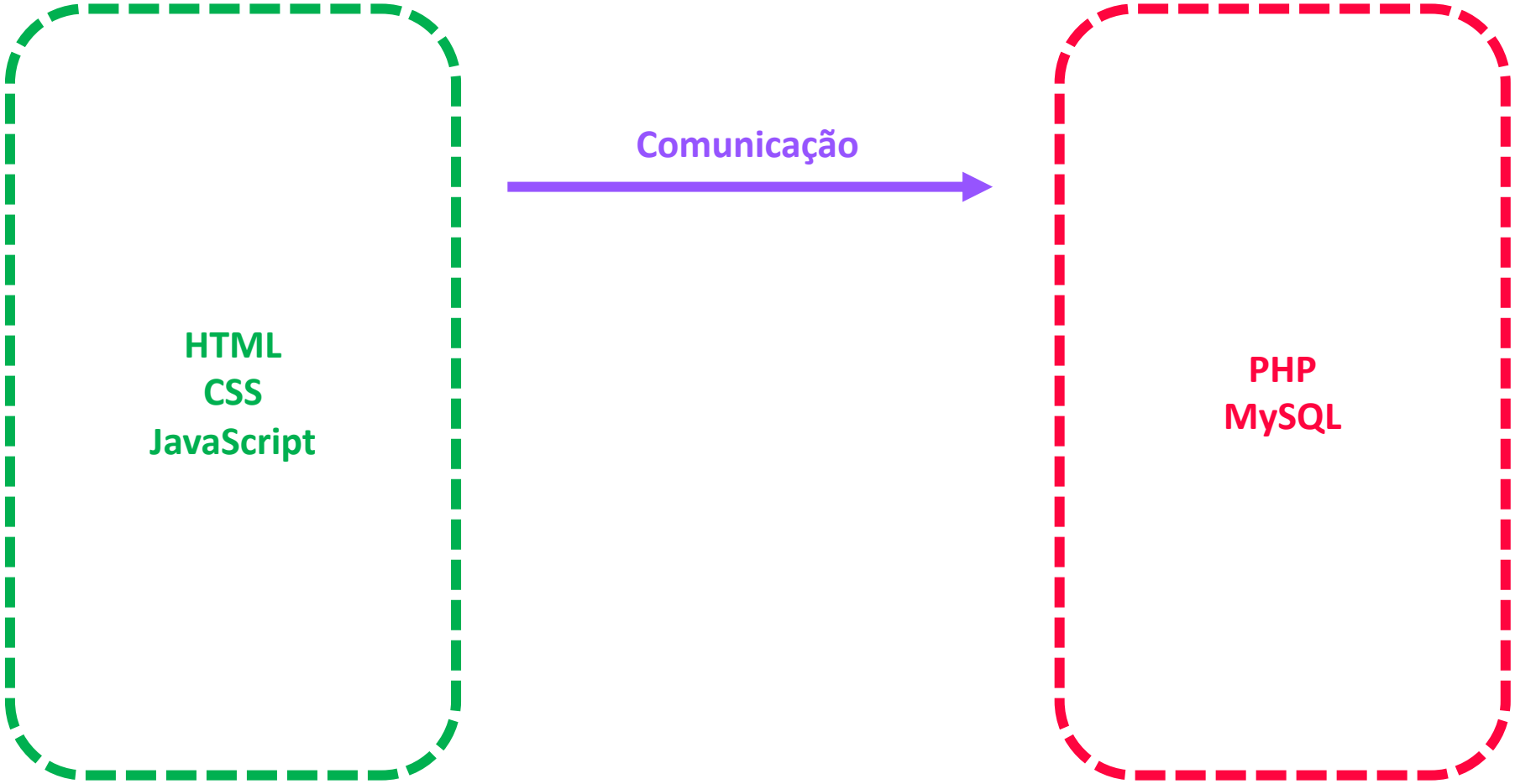
SERVIDOR

Cliente x Servidor

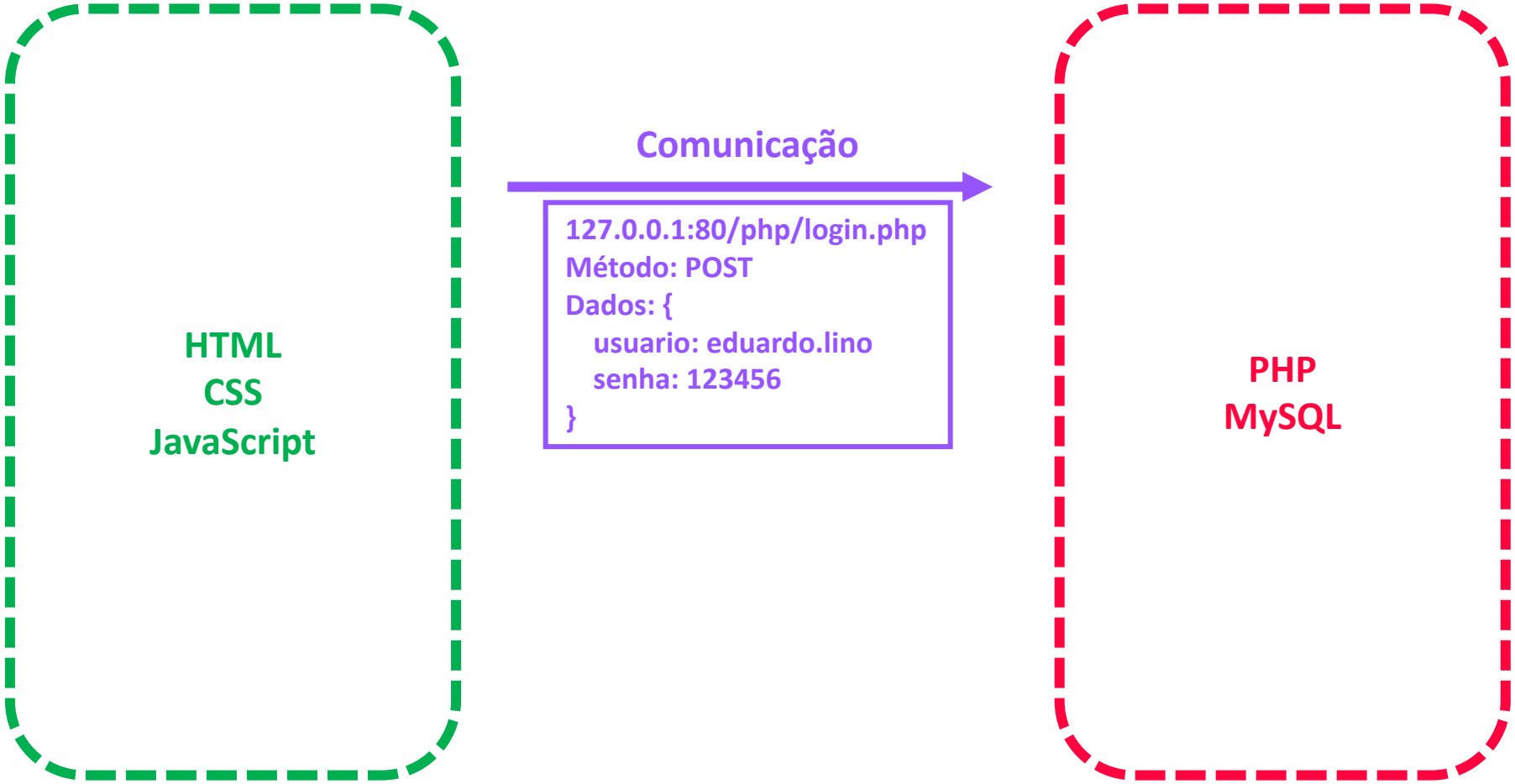
HTML
CSS
JavaScript

PHP
MySQL

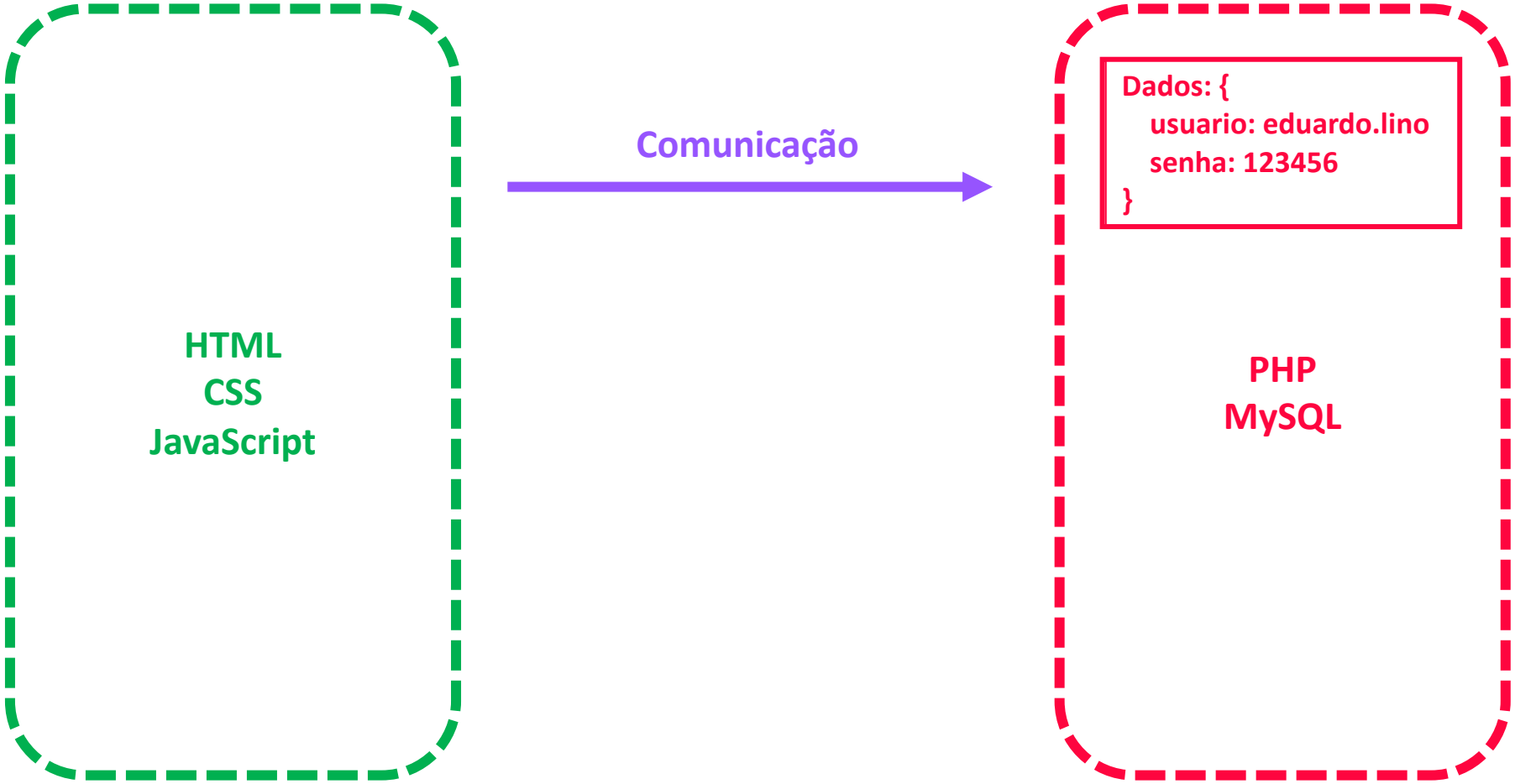
Cliente x Servidor



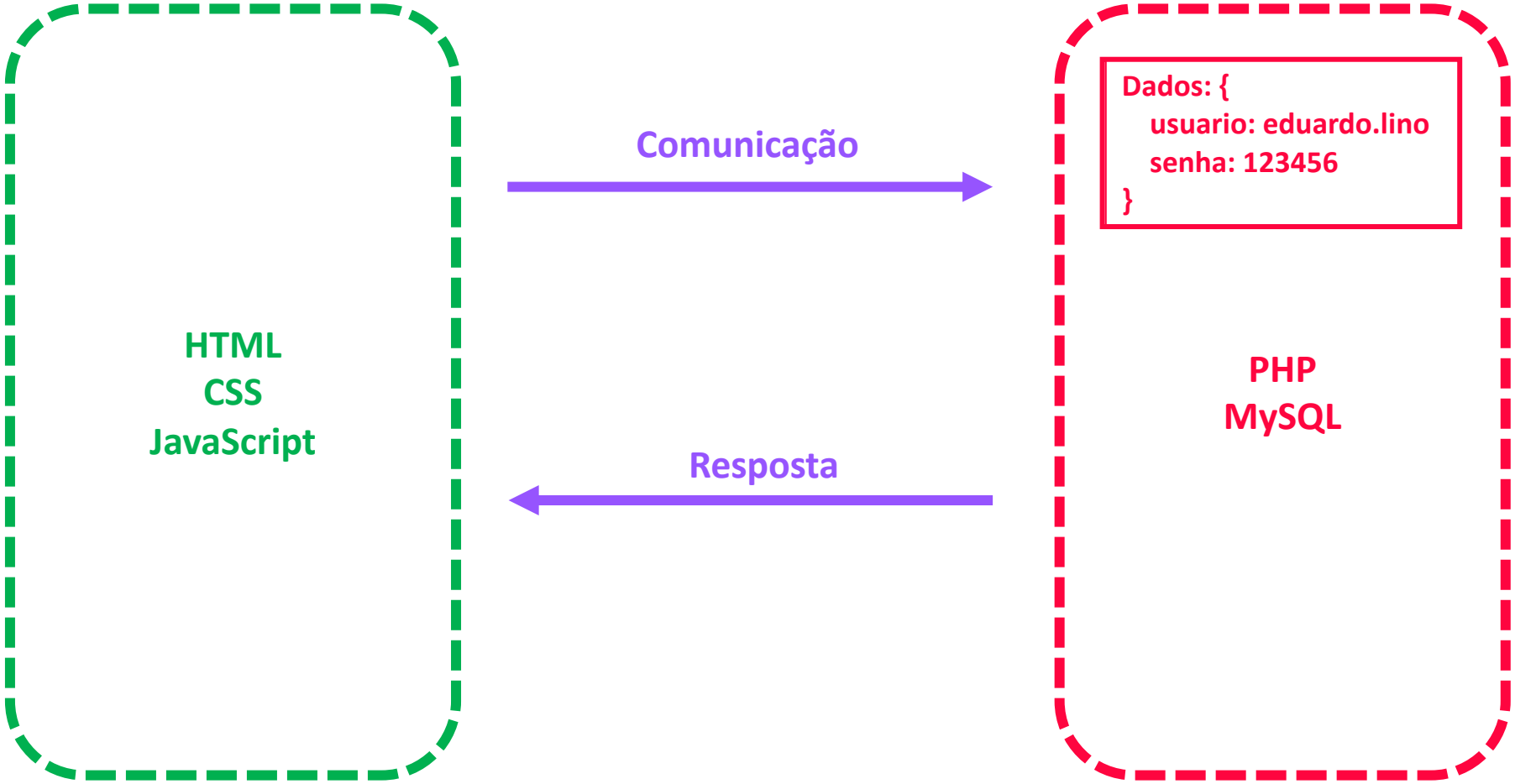
Cliente x Servidor



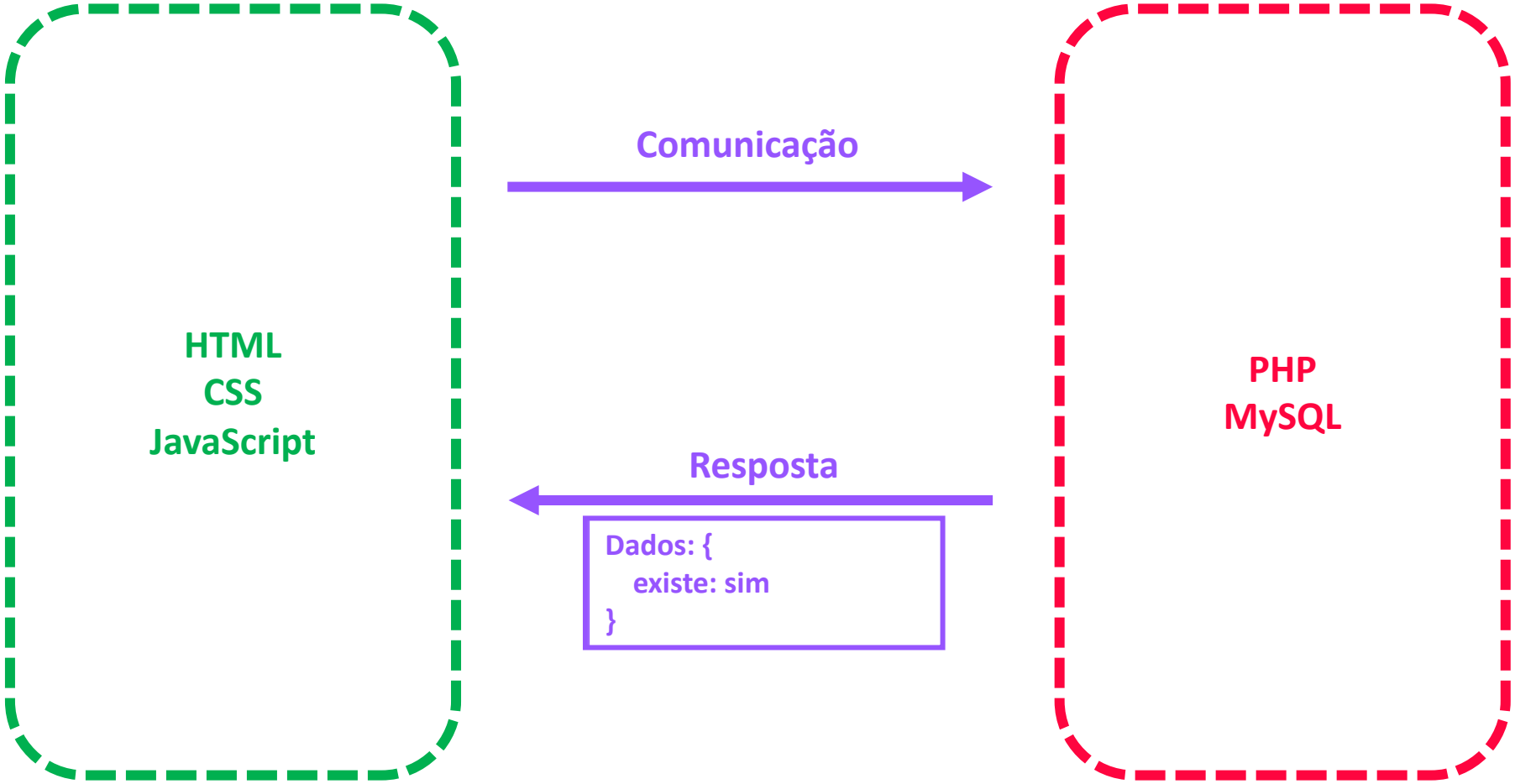
Cliente x Servidor



Cliente x Servidor



Cliente x Servidor



API Fetch


function onClick(){
fetch(url, {
headers: {
"Content-Type": "application/jso
"Accept": "application/json, text-plain, */*"
"X-Requested-With": "XMLHttpRequest",
method: 'post',
credentials: "same-origin",
body: JSON.stringify({
name: 'Tushar',
number: '78987'
JS
{fetch API}

API Fetch

```
fetch("arquivo.php", {
    method: "POST",
    body: dados
});
```

Promisse

- Promisse é uma promessa de execução assíncrona, onde um objeto é usado para processamento assíncrono.

- Representa eventualmente o resultado de uma operação assíncrona.

Promisse

- Promisse é uma promessa de execução assíncrona, onde um objeto é usado para processamento assíncrono.

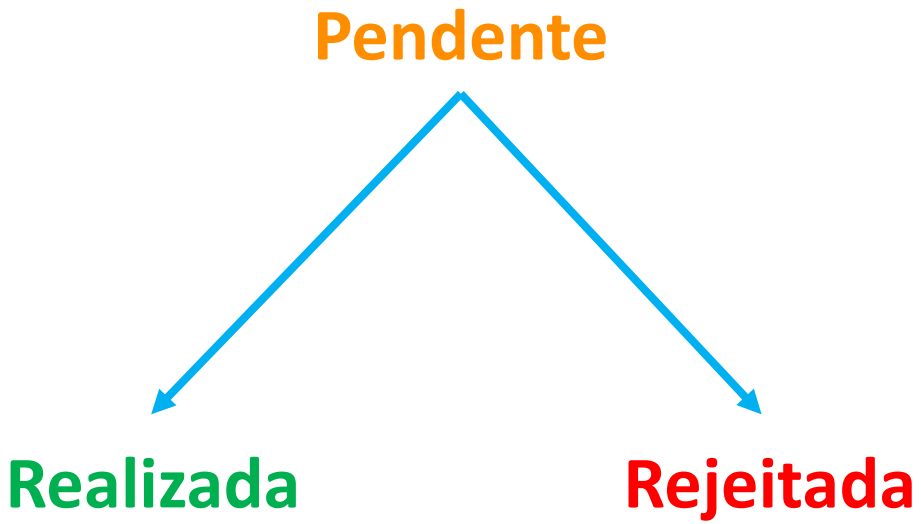
- Representa **eventualmente** o **resultado** de uma operação assíncrona.

Promisse

- Uma Promisse é um objeto que representa e gerencia o ciclo de vida de um resultado futuro!

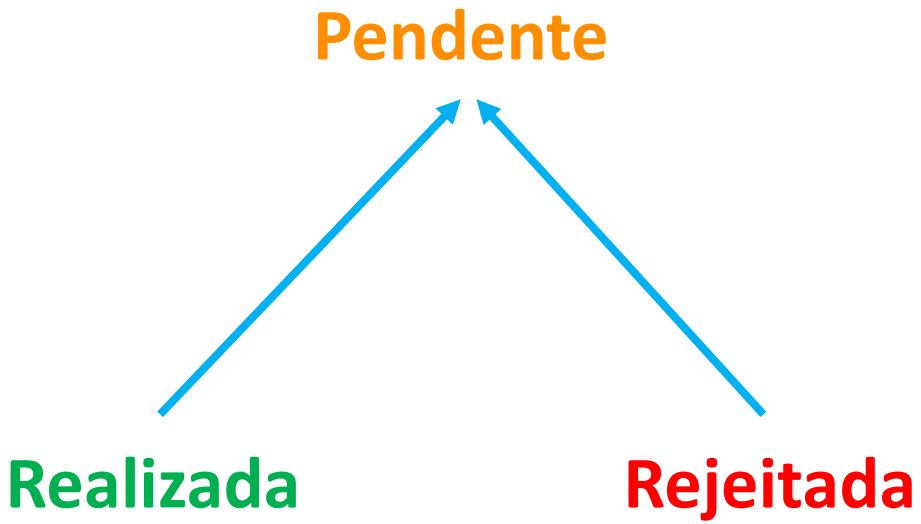
Promisse

- Status que pode retornar uma Promisse:



Promisse

- Status de retorno de uma Promisse.



Promisse

- Uma Promisse pode ser realizada ou rejeitada.

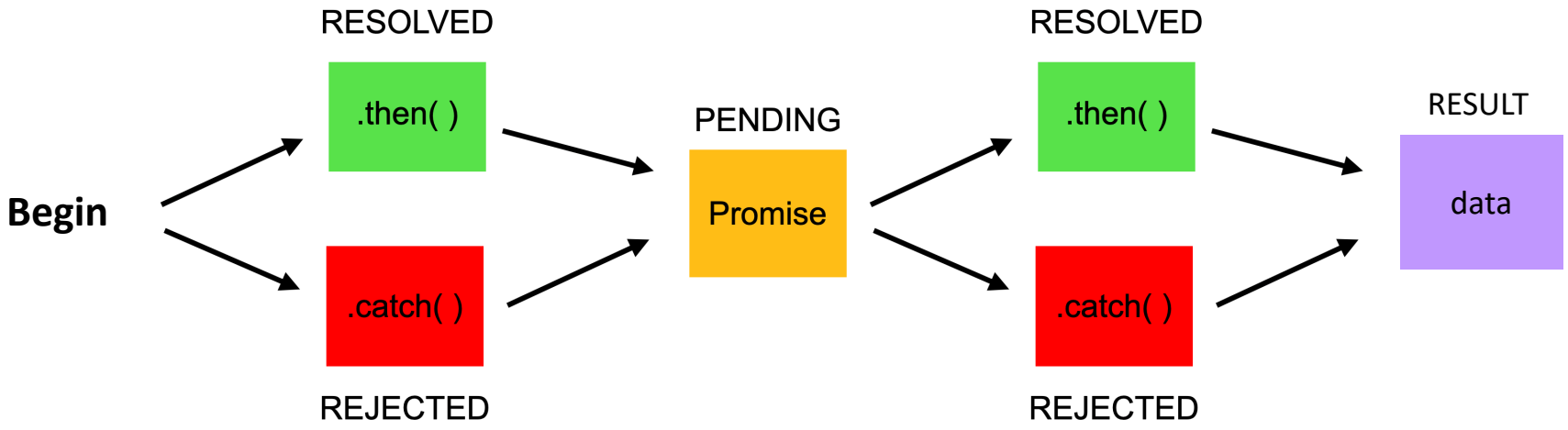
```
// New Promises start in "Pending" state
new Promise(function (resolve, reject) {

  // Transition to "Rejected" state
  reject(new Error('A meaningful error'))

  // Transition to "Fulfilled" state
  resolve({ my: 'data' })

})
```

Promisse

- Todo processamento inicial de uma Promisse, retorna uma nova Promisse, ou seja, uma Promisse é encadeada.

Promisse

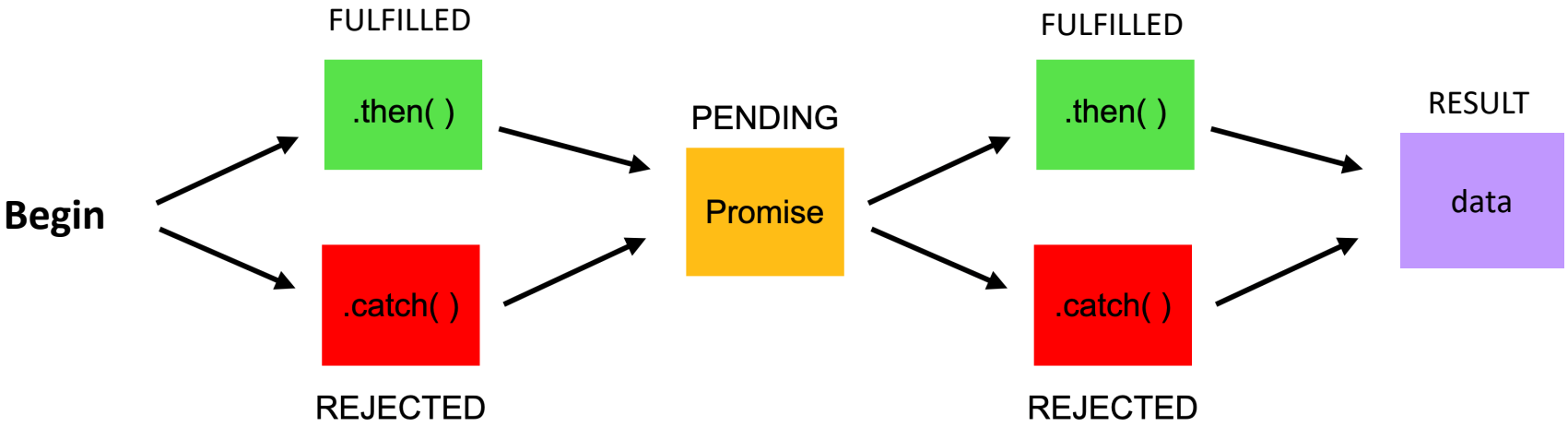
- Todo processamento inicial de uma Promisse, retorna uma nova Promisse, ou seja, uma Promisse é encadeada.



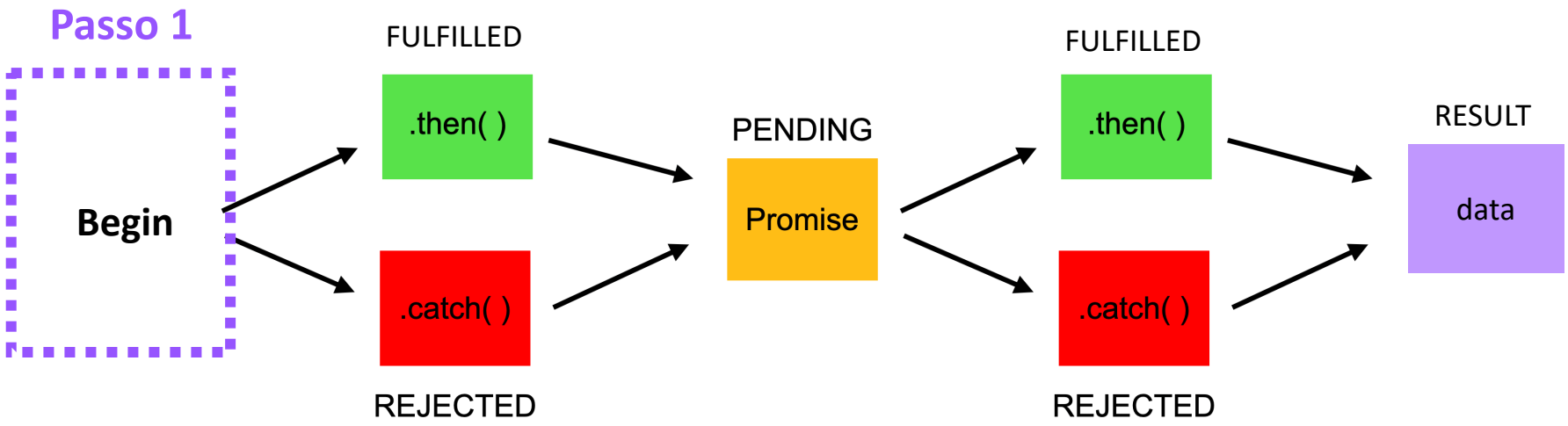
Async e Await

Async () => { Await }

Promisse

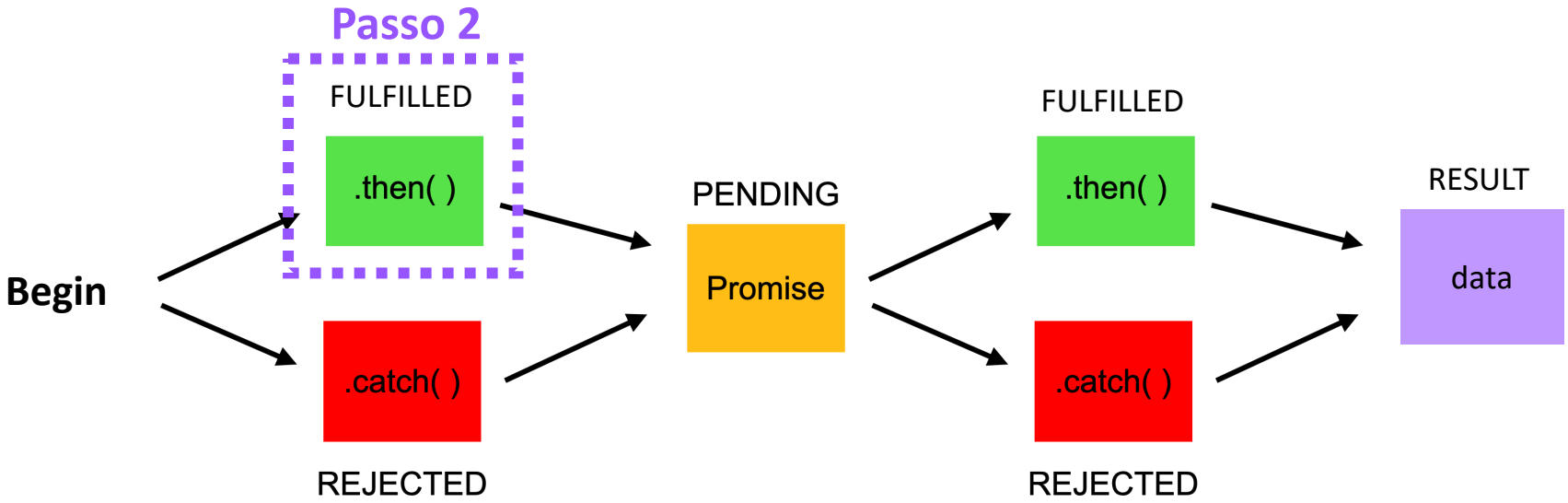

FULFILLED
.then( )
Begin
.catch( )
REJECTED
PENDING
Promise
FULFILLED
.then( )
.catch( )
REJECTED
RESULT
data

Promisse: Passo 1



Passo 1: Inicie a Promisse

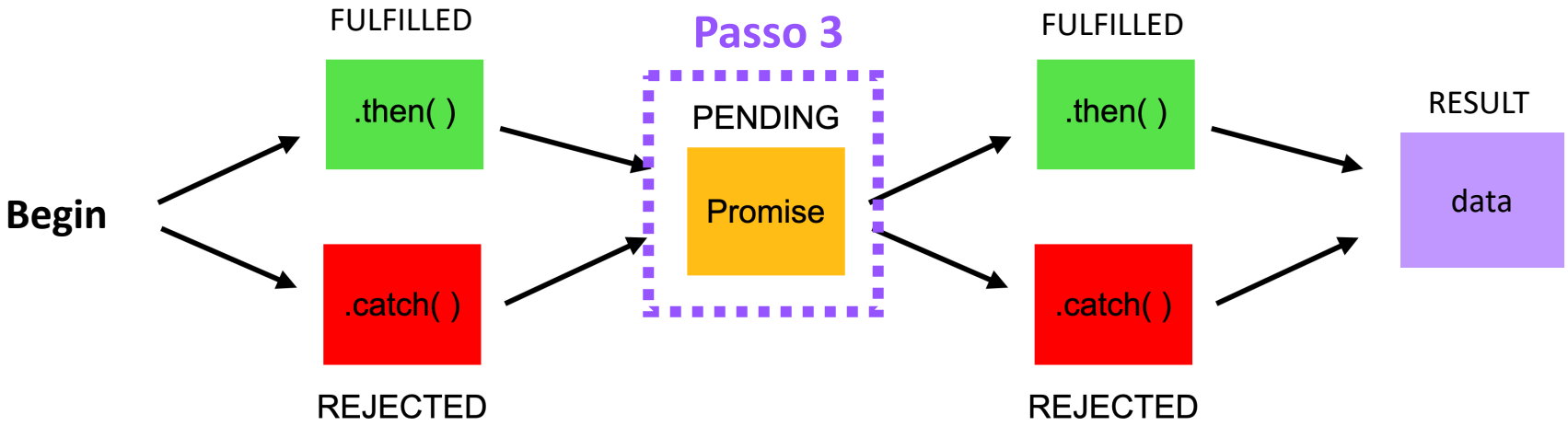
Promisse: Passo 2



Passo 1: Inicie a Promisse

Passo 2: Execute a Promisse, se tudo estiver correto, deverá retornar “Fulfilled”

Promisse: Passo 3

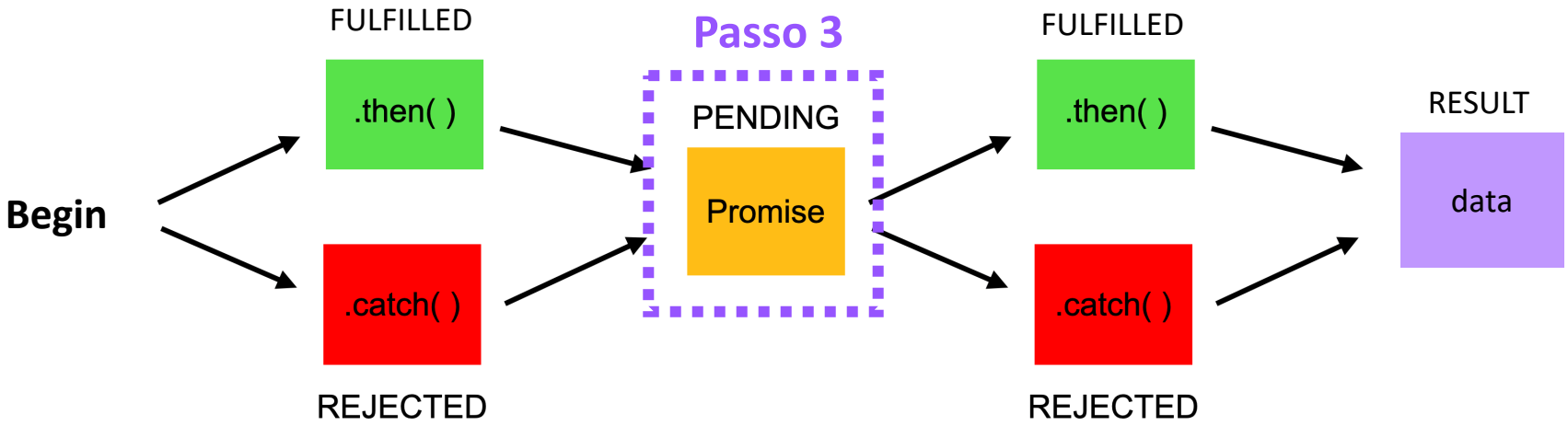


Passo 1: Inicie a Promisse
Passo 2: Execute a Promisse, se tudo estiver correto, deverá retornar “Fulfilled”
Passo 3: Se retorno for “Fulfilled”, retornará uma nova Promisse para ser executada

Promisse: Passo 1, 2 e 3

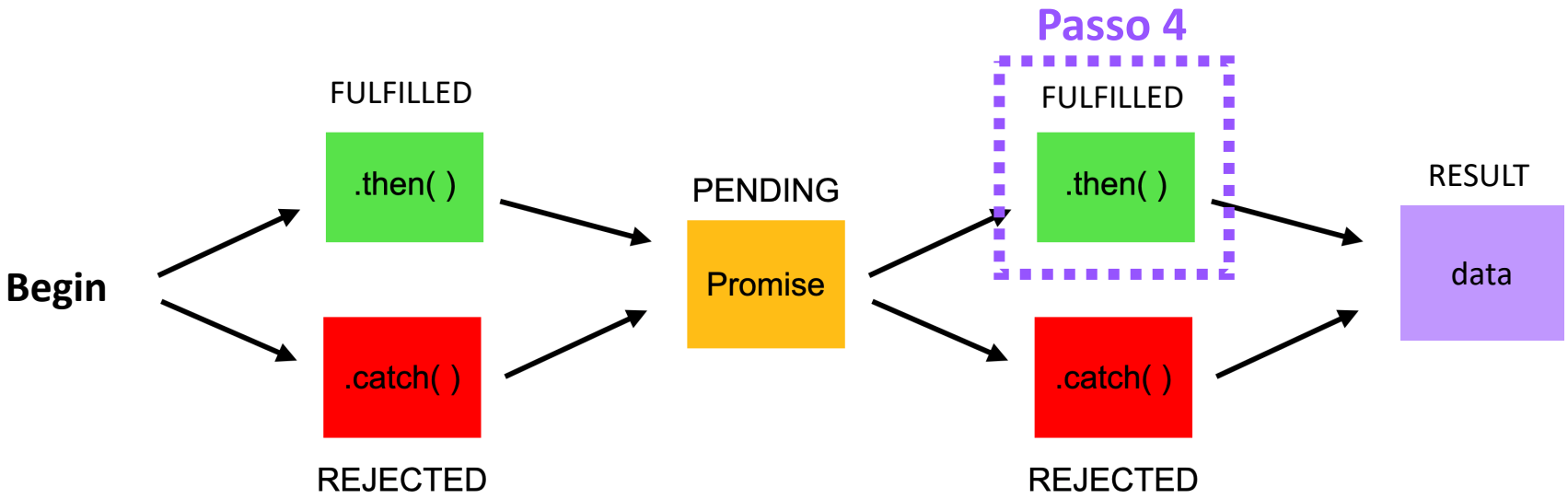
```
function funcao()
{

    var promisse = fetch("php/arquivo.php", {
        method: "POST",
        body: dados
    });

    console.log(promisse); // pendente!

}
```

Promisse: Passo 3



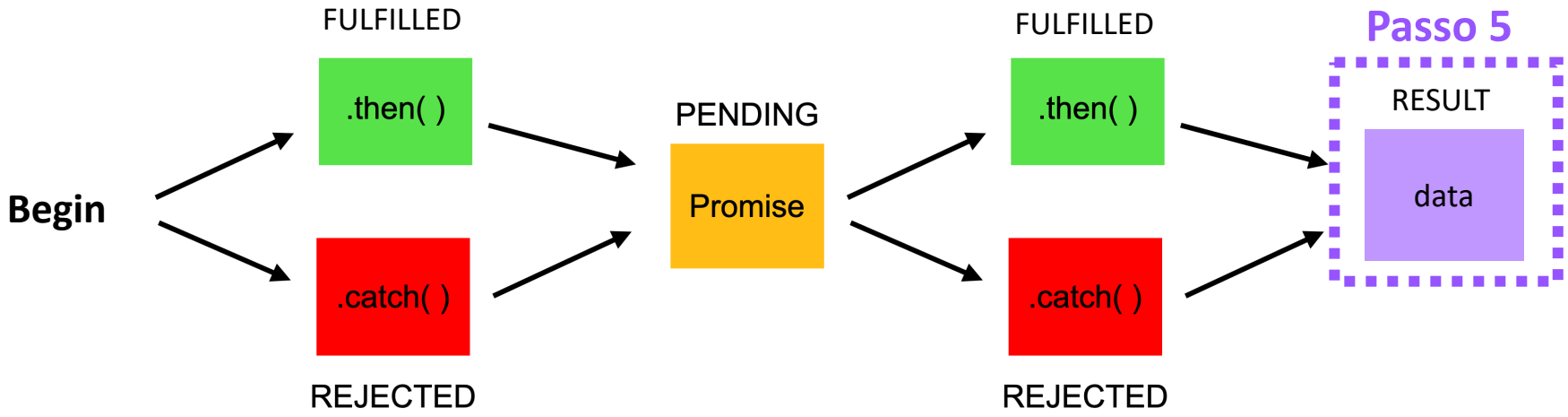
Passo 1: Inicie a Promisse
Passo 2: Execute a Promisse, se tudo estiver correto, deverá retornar “Fulfilled”
Passo 3: Se retorno for “Fulfilled”, retornará uma nova Promisse para ser executada

Promisse: Passo 4



Passo 1: Inicie a Promisse
Passo 2: Execute a Promisse, se tudo estiver correto, deverá retornar “Fulfilled”
Passo 3: Se retorno for “Fulfilled”, retornará uma nova Promisse para ser executada
Passo 4: Execute a Promisse, se tudo estiver correto, retornará os dados da requisição

Promisse: Passo 5



Passo 1: Inicie a Promisse
Passo 2: Execute a Promisse, se tudo estiver correto, deverá retornar “Fulfilled”
Passo 3: Se retorno for “Fulfilled”, retornará uma nova Promisse para ser executada
Passo 4: Execute a Promisse, se tudo estiver correto, retornará os dados da requisição
Passo 5: Utilize os dados de retorno!

Promisse: Passo 4 e 5

```
async function funcao()
{
    var form = document.getElementById('formCadastro');
    var dados = new FormData(form);

    var promisse = await fetch("php/arquivo.php", {
        method: "POST",
        body: dados
    });

    console.log(promisse); // Nova Promisse!

    var resultado = promisse.json();

    console.log(resultado); // Pendente!

}
```

Promisse: Passo 4 e 5

```
async function funcao()
{
    var form = document.getElementById('formCadastro');
    var dados = new FormData(form);

    var promisse = await fetch("php/arquivo.php", {
        method: "POST",
        body: dados
    });

    console.log(promisse); // Nova Promisse!

    var resultado = await promisse.json();

    console.log(resultado); // Resultado!
}
```

Exercício em Sala 1

- Faça o download do projeto "Exercício Semana 4" e salve a pasta do projeto chamada "projeto" no diretório do servidor para execução.

Exercício em Sala 2

- Desenvolva uma nova aplicação web, onde deverá haver um campo de entrada no HTML requisitando o país de origem do usuário.
- No servidor, verifique o país de origem e retorne a moeda do país.
- Imprima na página, o retorno que veio do servidor.