

# Server-Side Web Development

---

Databases and Web Applications Laboratory (LBAW)  
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes  
Dept. Informatics Engineering  
FEUP · U.Porto

# Current Status

---

## → Plan:

- 6th week of classes;
- Project: ER feedback; last class before EBD delivery;
- Lecture: server-side web development; review web architectures;
- Labs: finish database specification (EBD);

## → Monitor sessions: Tuesday, at 16h

- Previous sessions: Git and GitFlow; PostgreSQL (setup);
- This week: Postgres indexes, triggers and transactions

# Outline for Today

---

- Server-Side Web Development
  - Overview and main concepts
  - AJAX
  - Review Web Architectures
- Architecture Specification and Prototype (EAP) component
  - A7: Web Resources Specification

Rendering on the Web

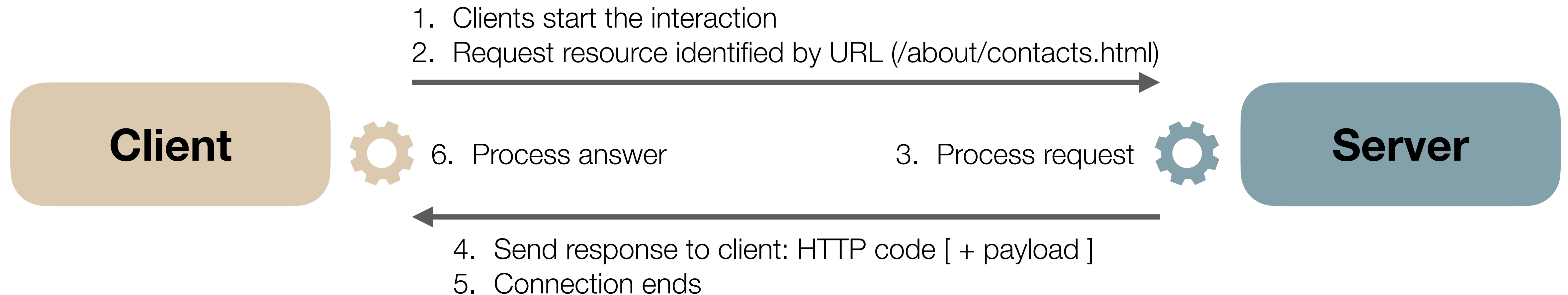
# The World Wide Web

---

- A distributed information system, with
  - Web servers, waiting for client requests to handle
  - Web clients (browsers and others), used to navigate in this 'information space'
- Core technologies
  - URL, defines how to address information resources published on the web
  - HTML, defines how to represent information on the web
  - HTTP, defines how web clients can interact with web servers

# Client-Server Paradigm

---



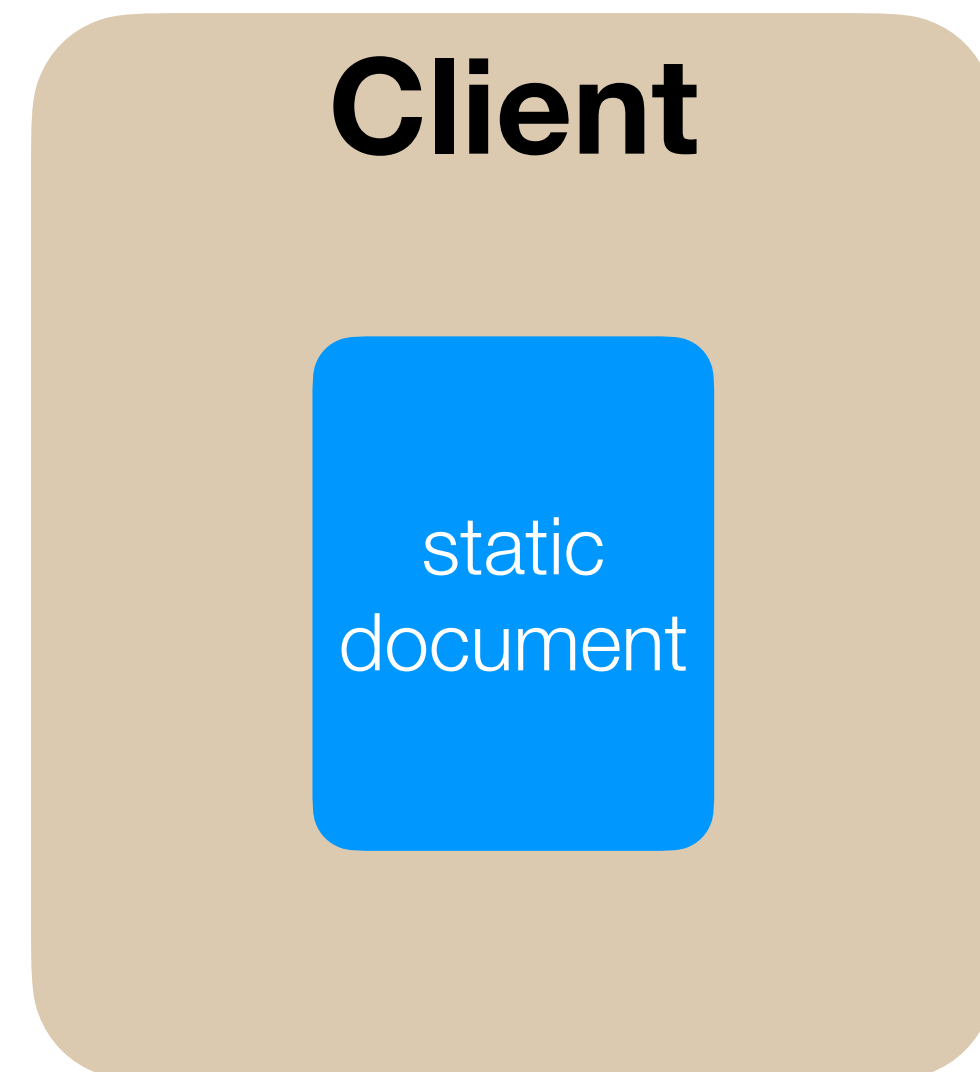
# Static and Dynamic Web Documents


---


- Web documents (or resources) are HTML, CSS, JS, JSON, images, media, i.e. *anything*.
- **Static web documents** have no processing involved
  - Web servers simply locate the resource and send it to clients. No rendering or code execution, documents were previously produced and are served without changes.
  - Use cases: images, CSS, HTML documents without 'live data', rarely updated.
- **Dynamic web documents** are prepared when requested
  - Dynamic web documents don't exist in advance, they are prepared in the moment for the client that issued the request. Can make use of data in databases, external sources, APIs, client information, etc.
  - Use cases: HTML with information read from a database; personalized web page; web application.

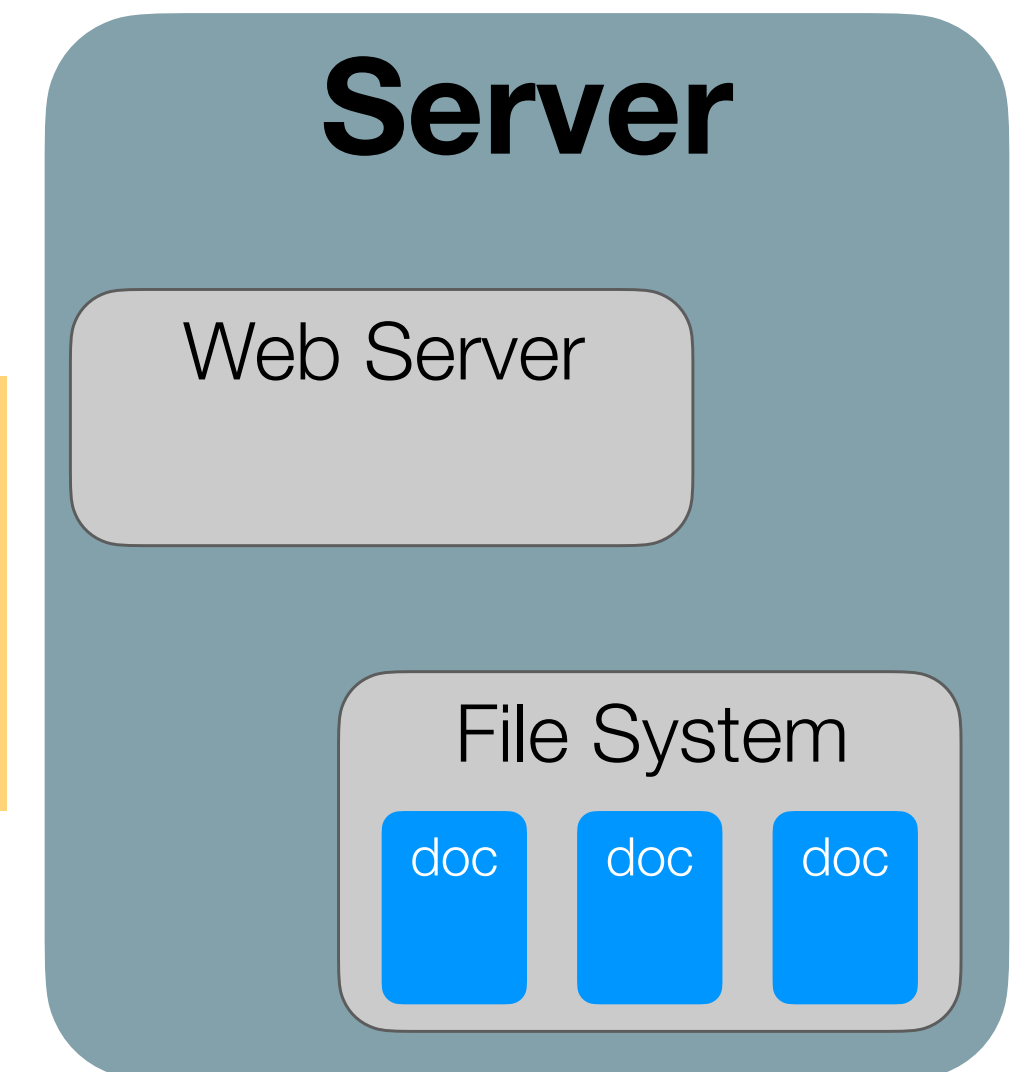
# Static Web Documents

1. Client starts the interaction
2. Request resource identified by URL (/about/contacts.html)



- 
6. Process answer
  7. Present document

3. Process request
  - 3.1 Parse URL
  - 3.2 Locate document
  - 3.3 Send document to client
- 



4. Send response to client: HTTP code [ + payload ]
5. Connection ends



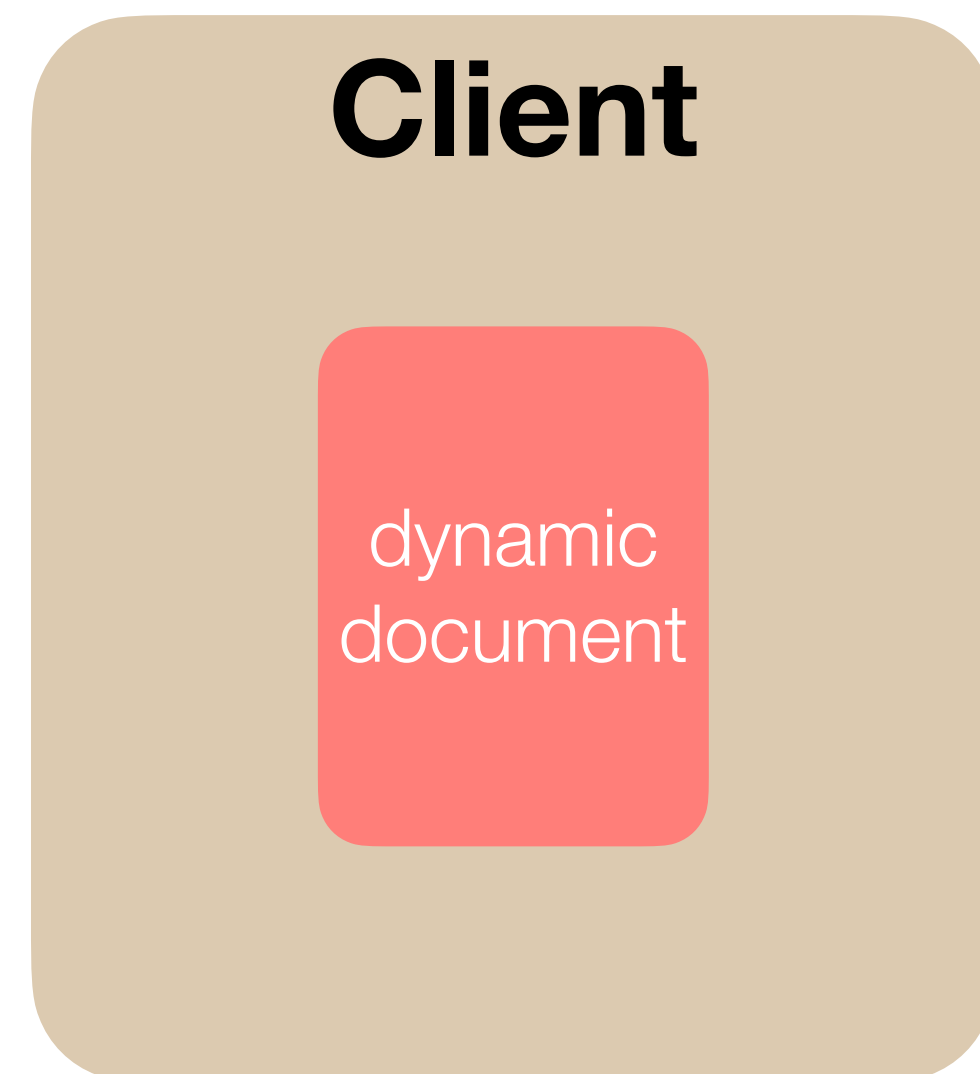
# Dynamic Web Documents


---


- Dynamic web documents are prepared on the fly in response to a specific client request and can be prepared on the server or on the client.
- Server-side rendering (SSR)
  - The document is prepared on the server and the result is sent to the client.
  - Code execution is on the server.
- Client-side rendering (CSR)
  - The server send to the client the necessary elements to build the document (JS code in particular)
  - The client builds the document using JavaScript.
- A mix of these techniques can be implemented, e.g. AJAX.

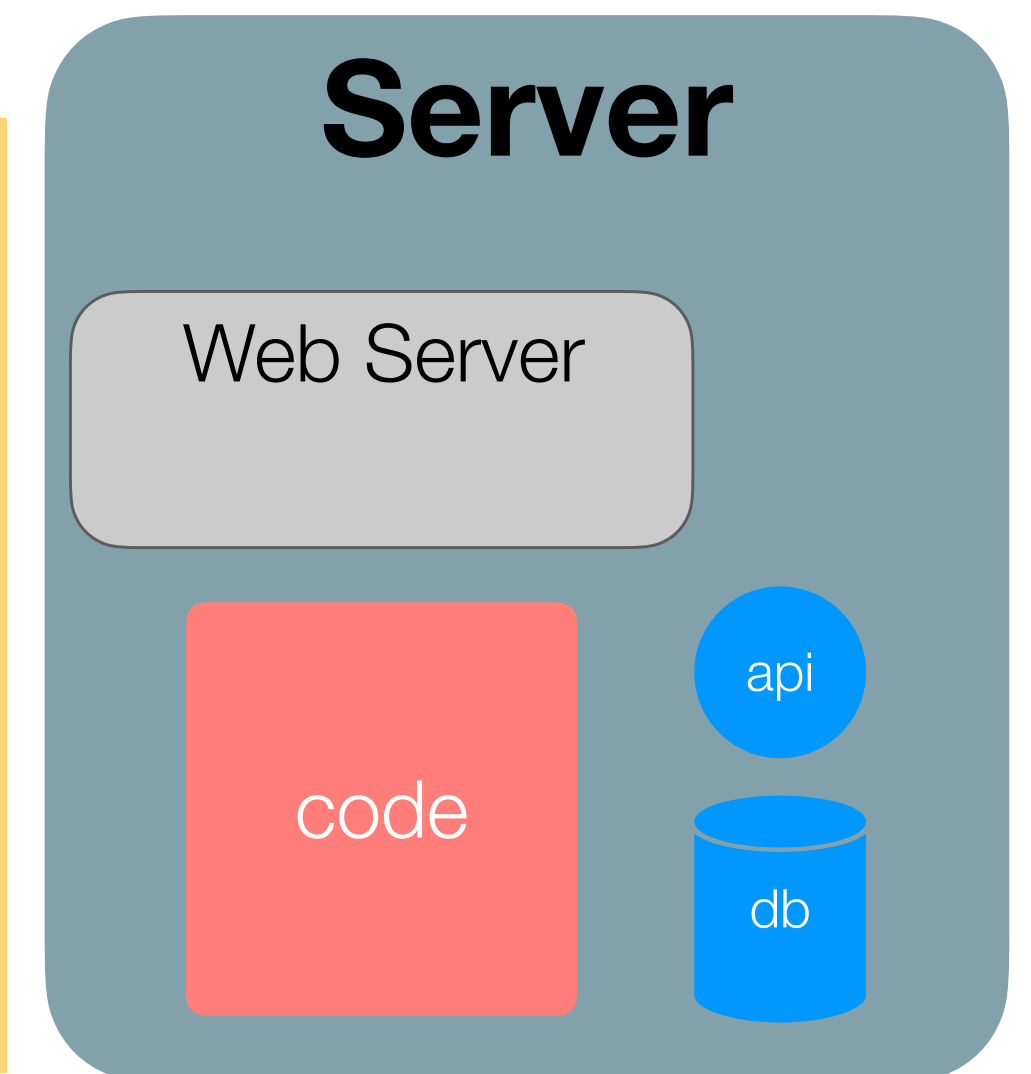
# Server-Side Dynamic Web Documents

1. Client starts the interaction
2. Request resource identified by URL



- 
6. Process answer
  7. Present document

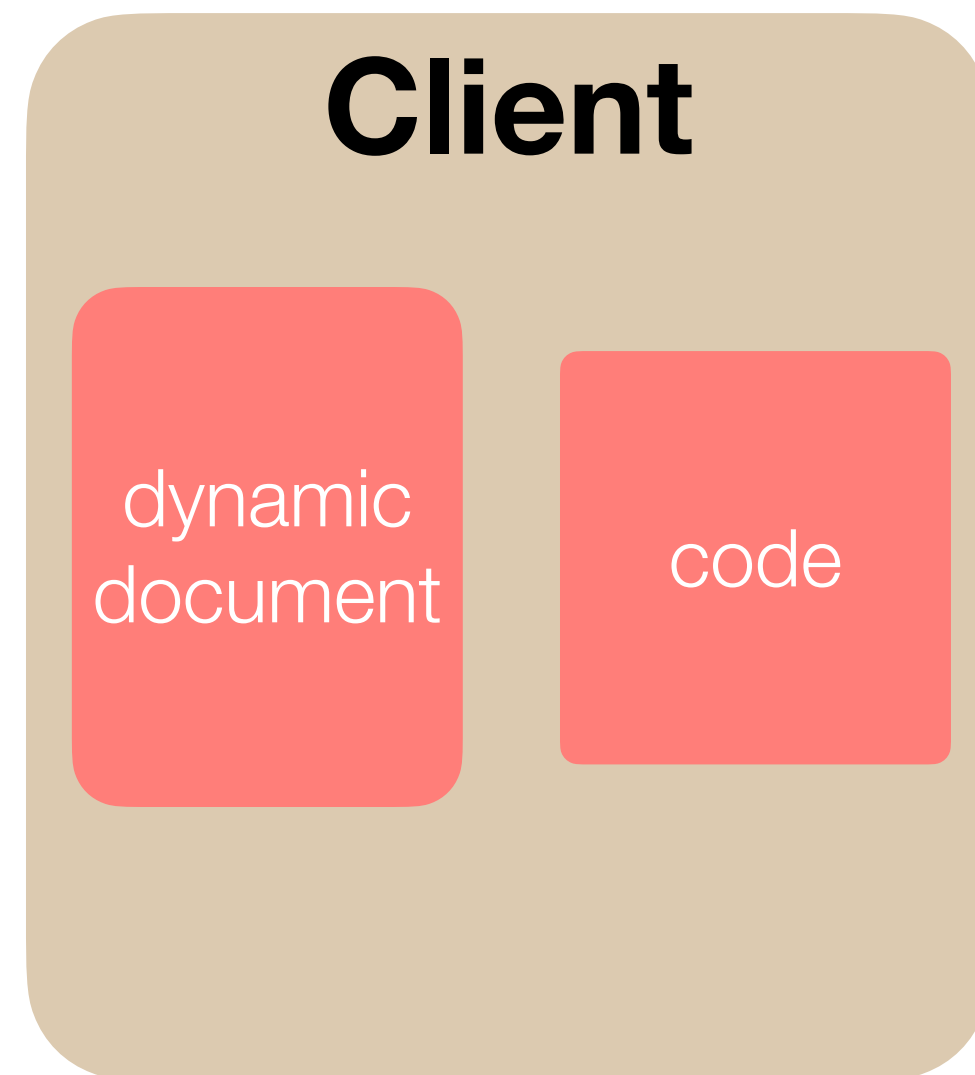
- 
3. Process request
    - 3.1 Parse URL
    - 3.2 Execute code
      - 3.2.1 [ Connect to database ]
      - 3.2.2 [ Connect to API ]
      - 3.2.3 [ etc ]
    - 3.3 Send code output (the dynamic document) to the client



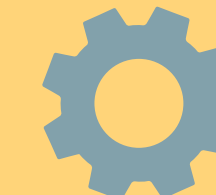
4. Send response to client: HTTP code [ + payload ]
5. Connection ends

# Client-Side Dynamic Web Documents

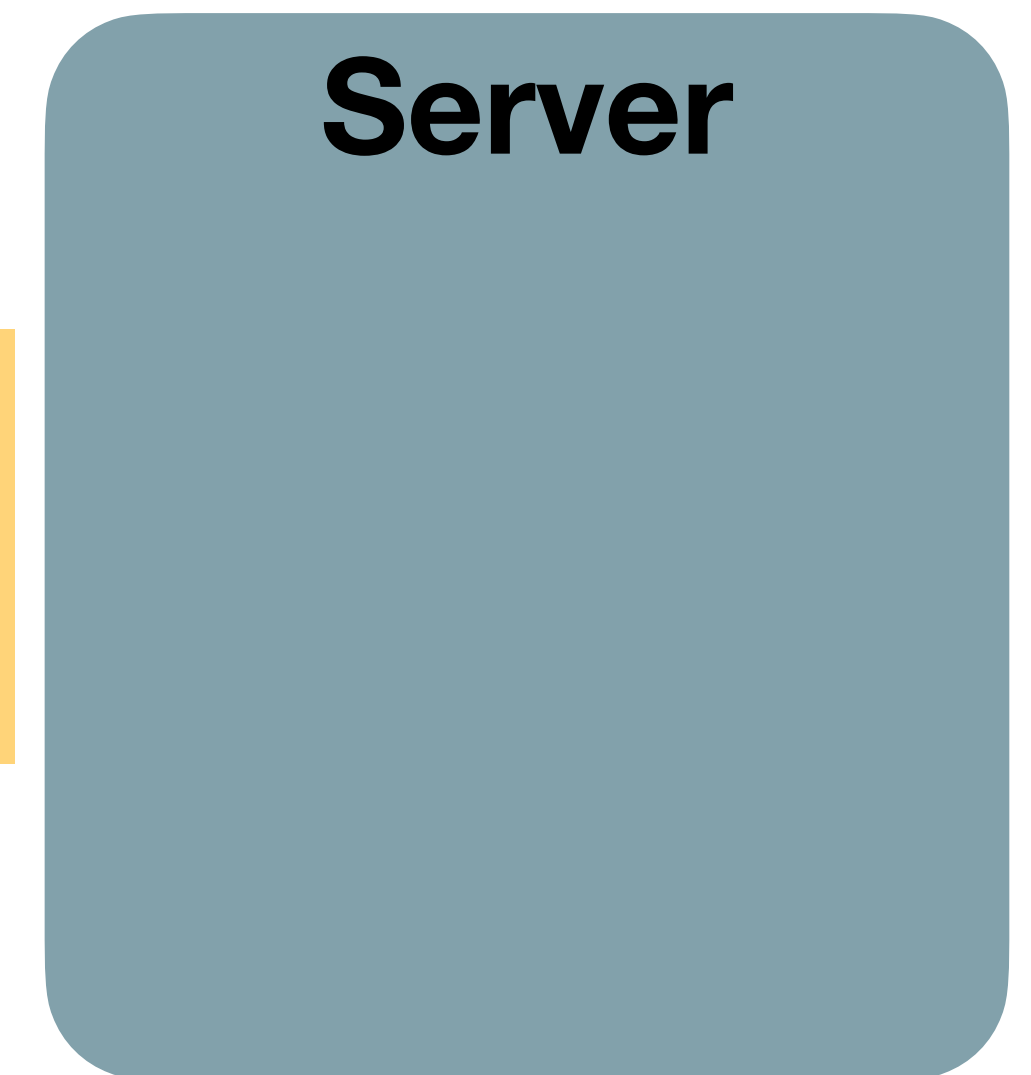
1. Client starts the interaction
2. Request resource identified by URL



6. Process answer
  - 6.1 Execute JavaScript code
    - 6.1.1 [ Request additional resources ]
    - 6.1.2 [ Connect to local or remote APIs ]
    - 6.1.3 [ etc ]
  - 6.2 Present dynamic document



3. Process request
  - 3.1 Send document



4. Send response to client: HTTP code [ + payload ]
5. Connection ends

# Rendering on the Web

---

- Deciding where to render web resources is an architectural decision involving multiple trade-offs.
- No solution (as always!) is best for all cases.
- Static web documents
  - Fast, but rigid, i.e. no updates, no personalization, hard to maintain in scale.
- Dynamic web documents
  - Server-side: complex documents with live information, but require full round trips to the server for user interaction.
  - Client-side: complex interactive document (feel like a local app), but heavy on the client and not really hypertext.
- In practice, multiple solutions coexist for each particular use case, i.e. multiple architectural options in different parts of web systems, but also multiple architectural options on individual web documents.

# Server-Side Web Development

# Server-Side Web Development

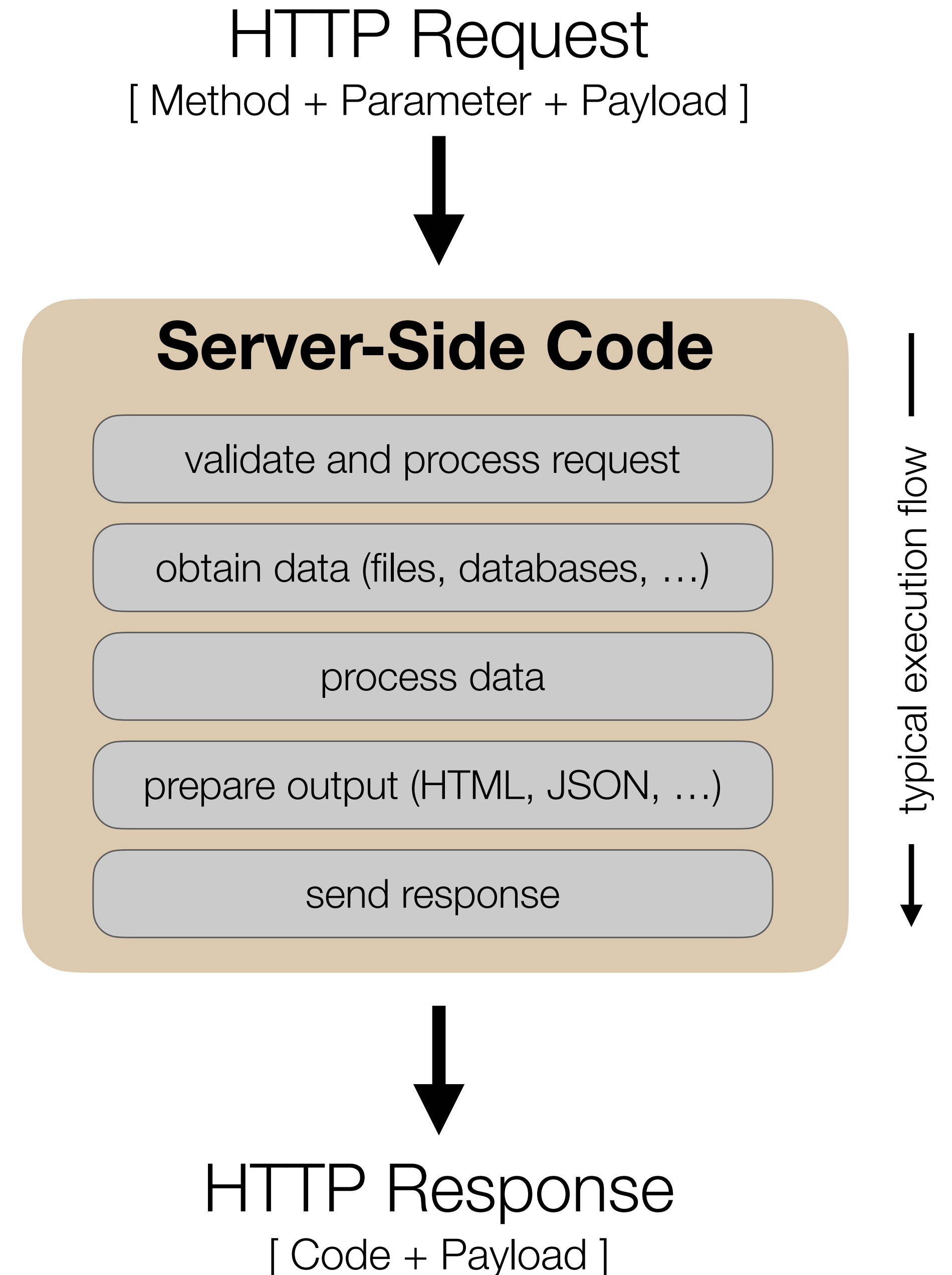
---

- In LBAW we adopt a server-side approach.
- With client-side code in specific use cases to improve user experience and support immediate changes without the need of a new full-page request (using AJAX).
- In LBAW, to simplify code, improve modularity, and organize development, we define two types of web resources (endpoints or simply pages).
  - View web resources, only access data for presentation — output is HTML.
  - Action web resources, change data and then redirect to a view web resource.

# Server-Side Web Development

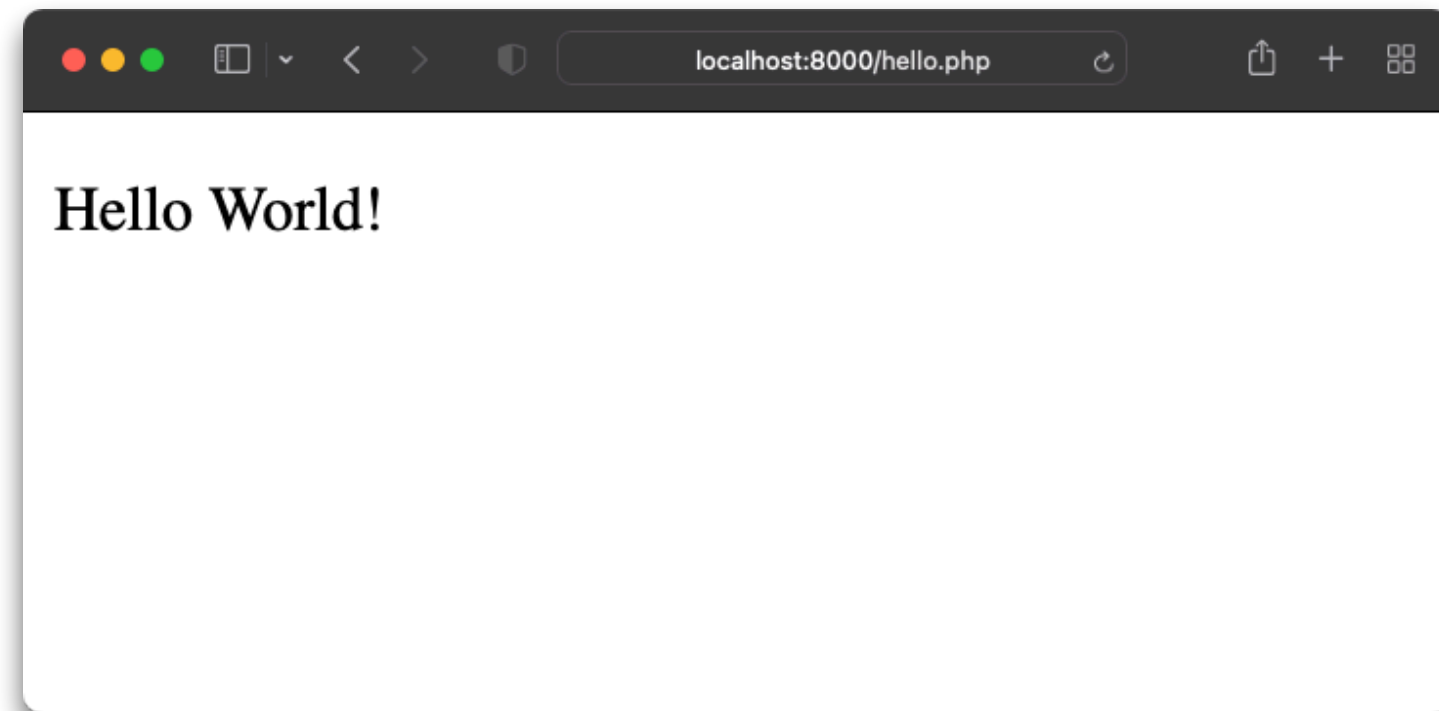
---

- A web-based software system based on the server publishes a set of endpoints.
  - /
  - /about.html
  - /students/view\_student.php
  - /search?q=flowers
- Each endpoint, also called web resource, accepts HTTP requests and outputs HTTP responses.
  - HTTP requests includes a method and, optionally, parameters and a payload.
  - HTTP responses include a code and, optionally, a payload.

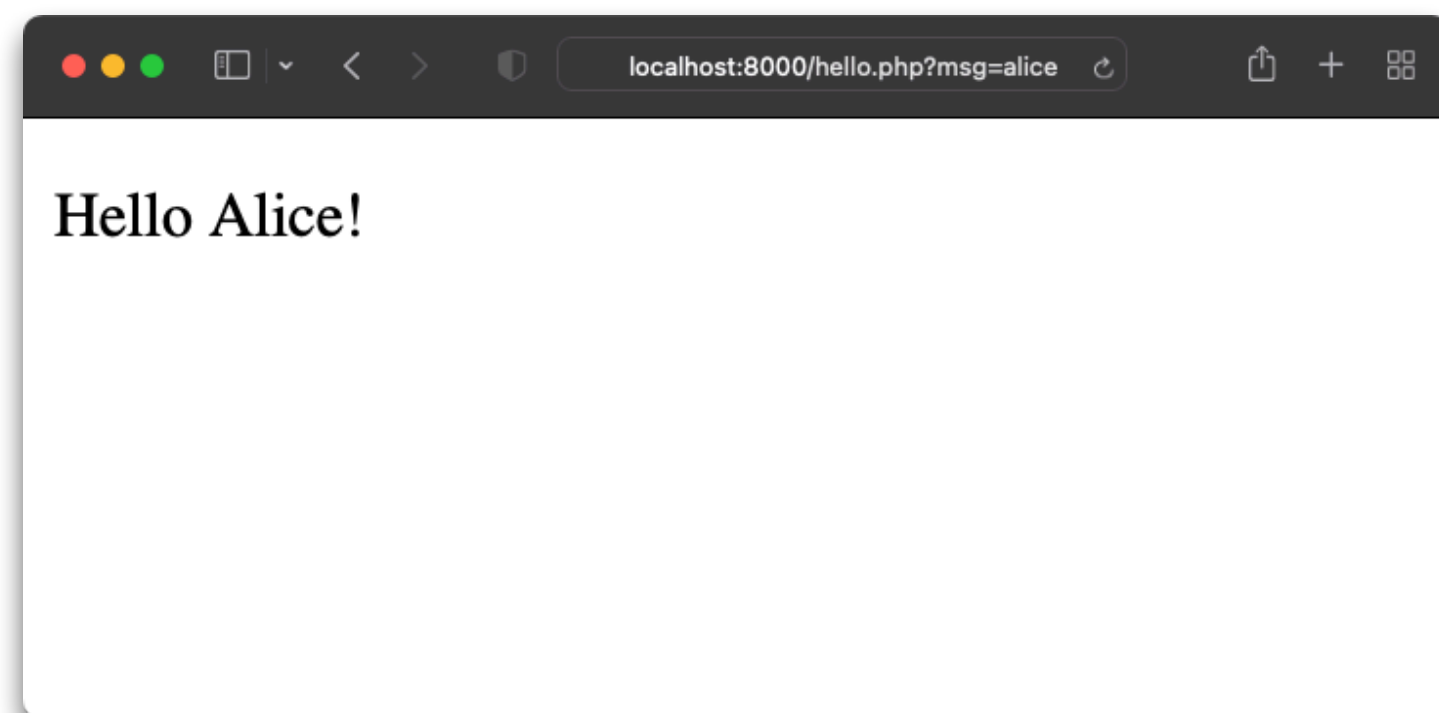


# Example in PHP

/hello.php



/hello.php?msg=alice



```
<?php
// hello.php file

// Check if parameters were sent.
if (isset($_GET["msg"])) {
    $msg = $_GET["msg"];
} else {
    $msg = "world";
}

// Capitalize message.
$msg = ucfirst($msg);

// Output HTML.
echo "<!DOCTYPE html>\n";

// Example of multiline printing.
echo <<<HTML_HEAD
<html>
<body>
HTML_HEAD;

// Print the paragraph with the message.
echo "\t<p>Hello $msg!</p>\n";
?>

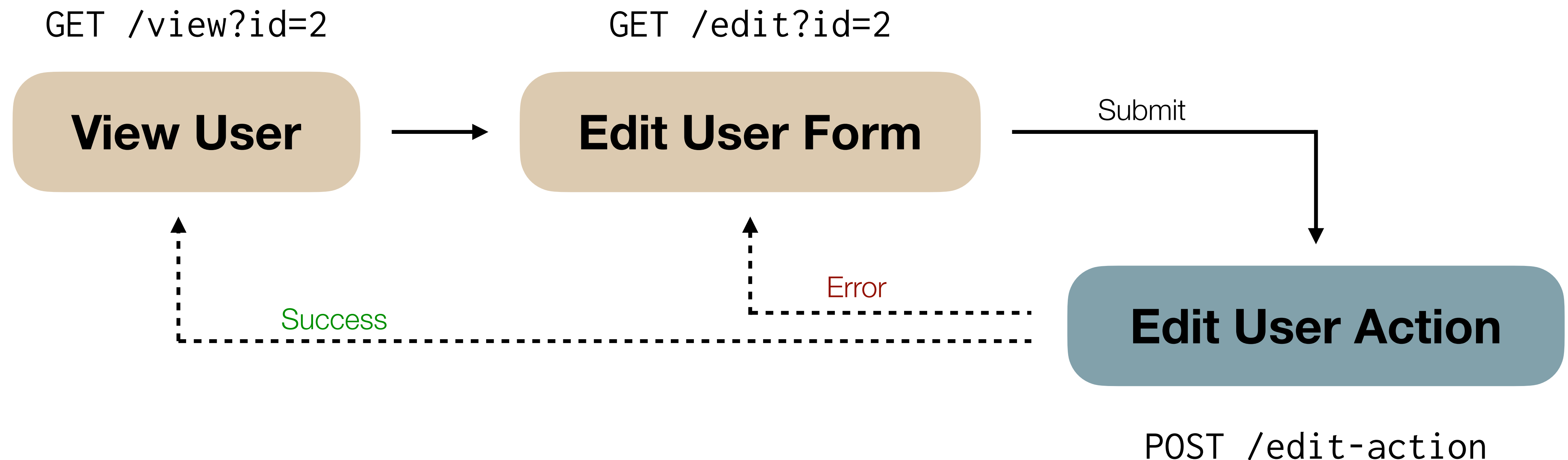
<!-- This is outside the PHP block. Just plain HTML. -->
</body>
</html>
```



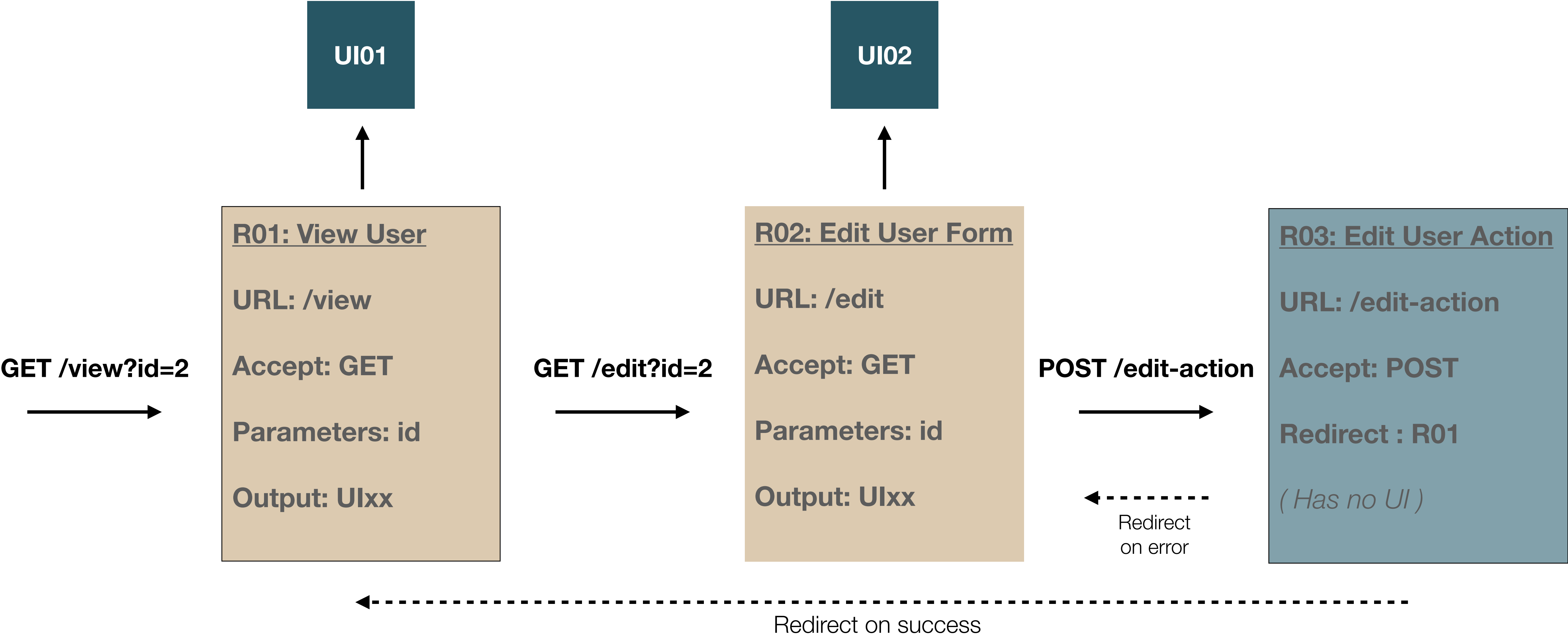
# View and Action Web Resources

---

- View pages read and present data. No changes to data.
- Action pages alter data and redirect to a view page. No presentation of data.



# Corresponding Web Resources Specification



Ajax

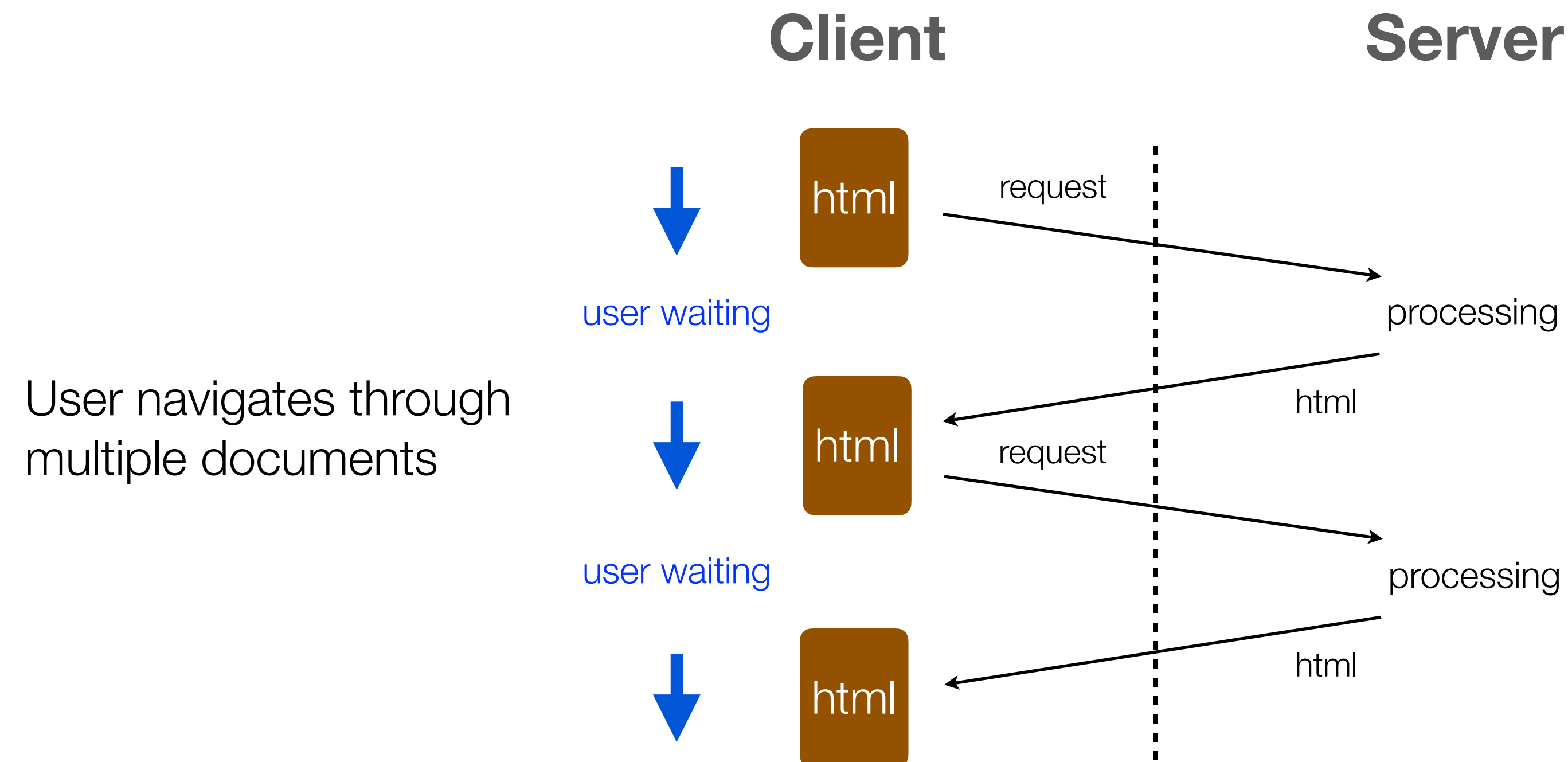
# Ajax

---

- Set of web development techniques to implement asynchronous web interactions.
- In standard synchronous web interaction, the user navigates across independent documents. There is a one-to-one mapping between application views and web pages.
- With AJAX, user interactions can be implemented within the same document. A n-to-one mapping between application views and web pages is possible.
- Basic use case: like action in a post does not result in leaving the current document.
- Using AJAX techniques, web documents can make requests to the server.

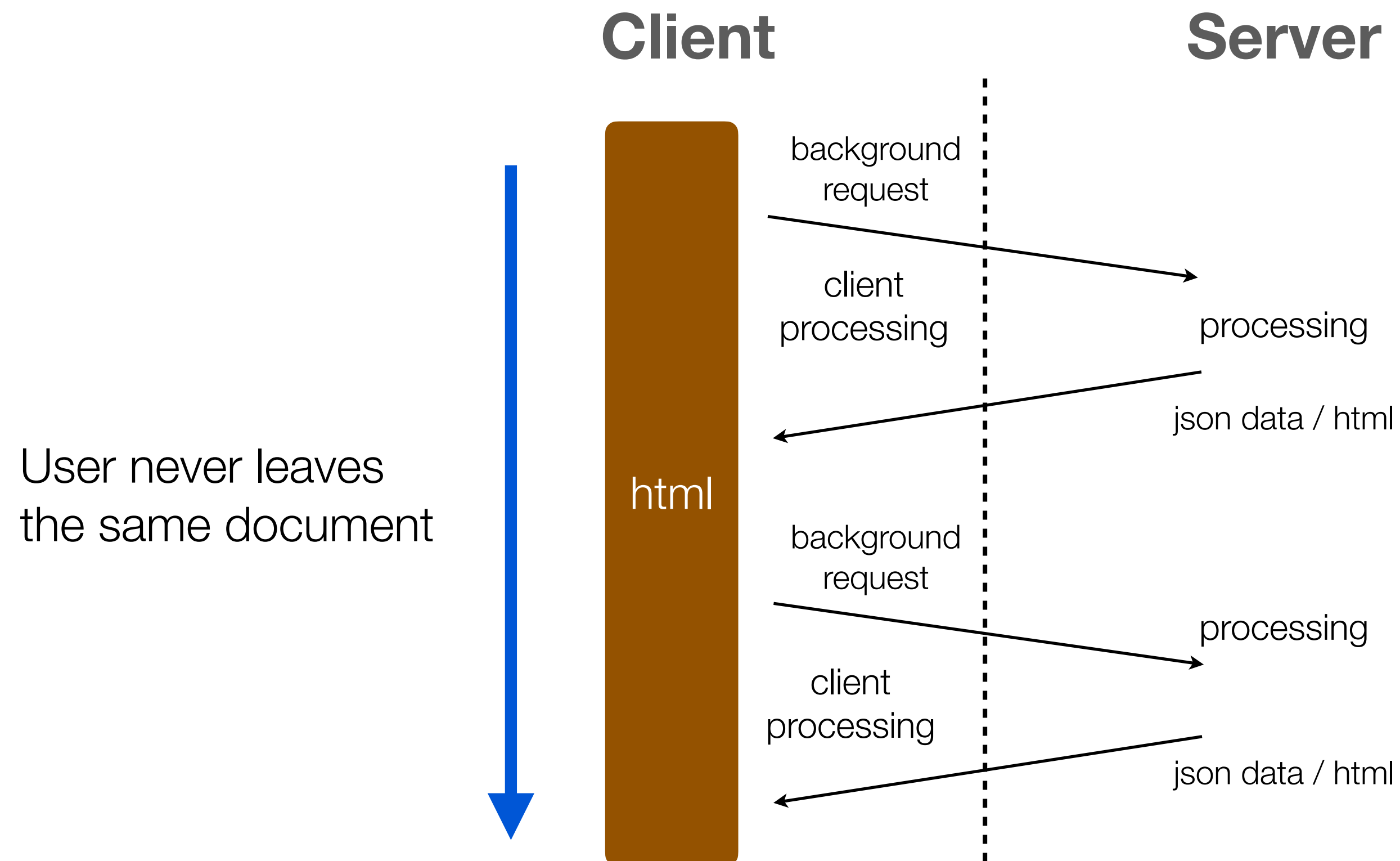
# Synchronous User Interaction

- In the classic web interaction model, a click results in the request of a new document: round trip to the server, the server prepares a new document, the client needs to render the document from zero.



# Asynchronous User Interaction

→ In asynchronous user interaction, the document makes requests to the server.



# Server-Side Technologies

---

- In server-side web development there is a large and diverse set of options
- Web servers, e.g. Apache, IIS, nginx, cloud, ...
- Web frameworks, e.g. Laravel, Django, Ruby on Rails, Spring, ...
- Storage solutions, e.g. relational, document, key-value, ...
- Programming languages, e.g. Python, PHP, Go, Ruby, ...
  - Routing libraries
  - Templating libraries
  - ...
- Next: a lot of common and recurring server-side use cases let to the development of web frameworks.

# Summary

---

- In LBAW we adopt a server-side architecture using Laravel, a PHP framework.
- AJAX is used to improve user experience.
- In the Architecture Specification and Vertical Prototype component (EAP)
  - Define the architecture of the application by specifying the web resources;
  - Implement a vertical prototype to validate technology stack.



# References

---

- PHP: The Right Way  
<https://phptherightway.com/>
- MDN Web Docs, Server-side website programming  
<https://developer.mozilla.org/docs/Learn/Server-side>
- MDN Web Docs, Ajax  
<https://developer.mozilla.org/docs/Web/Guide/AJAX>
- Google Developer, Rendering on the Web (2019)  
<https://developers.google.com/web/updates/2019/02/rendering-on-the-web>