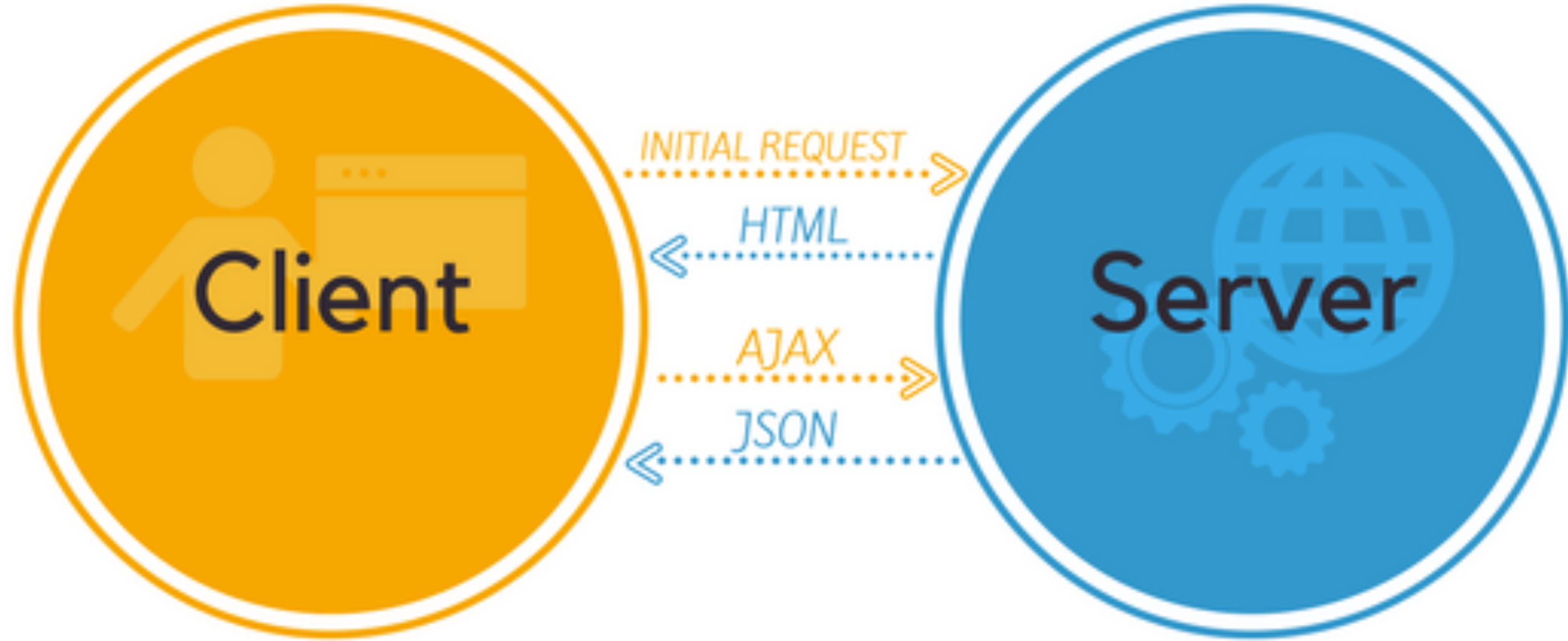


## Svelte Introduction



A review of the  
fundamental features of the  
Svelte framework.





JS

Javascript

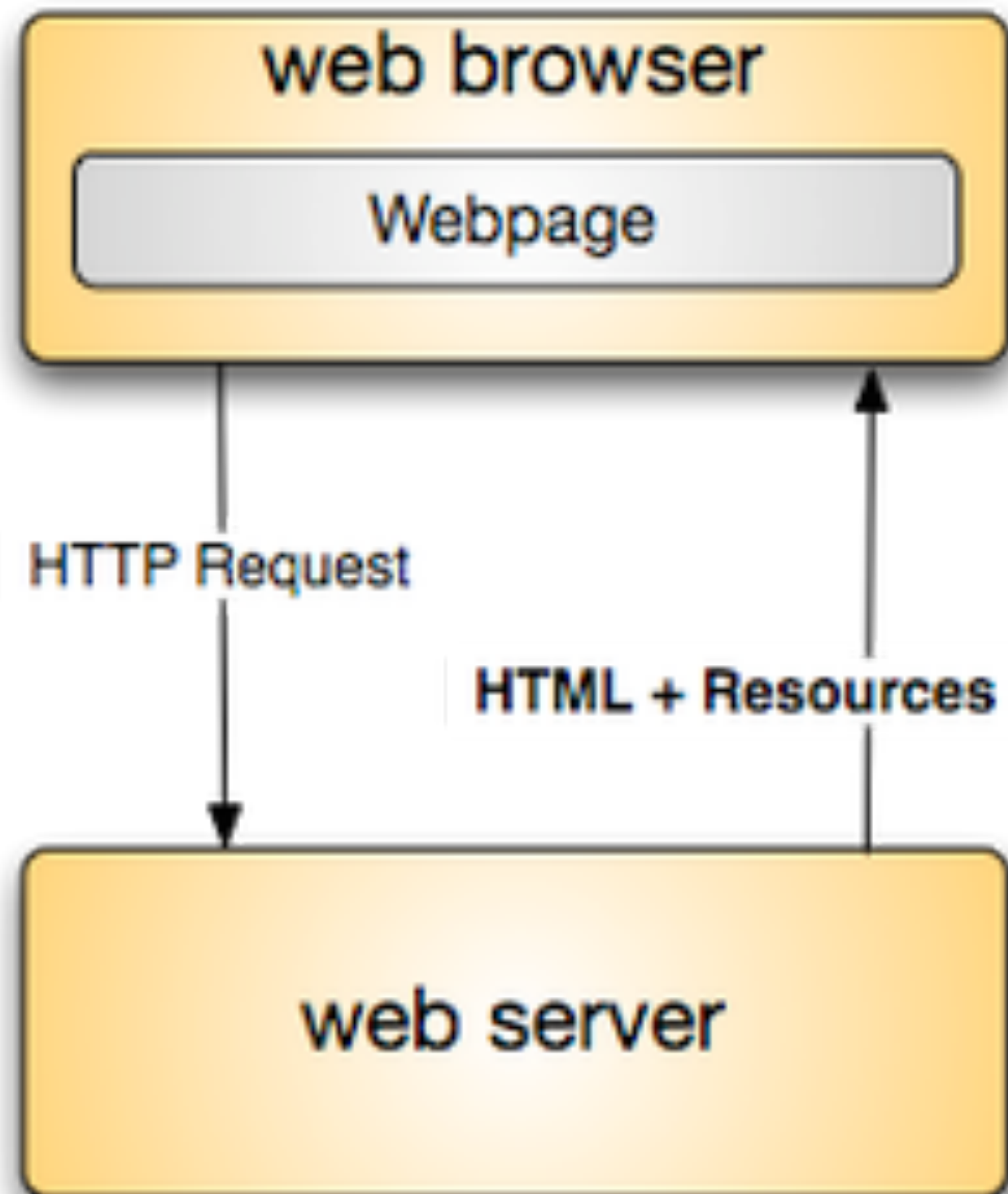
Request  
Response  
Headers  
Body

XmlHttpRequest

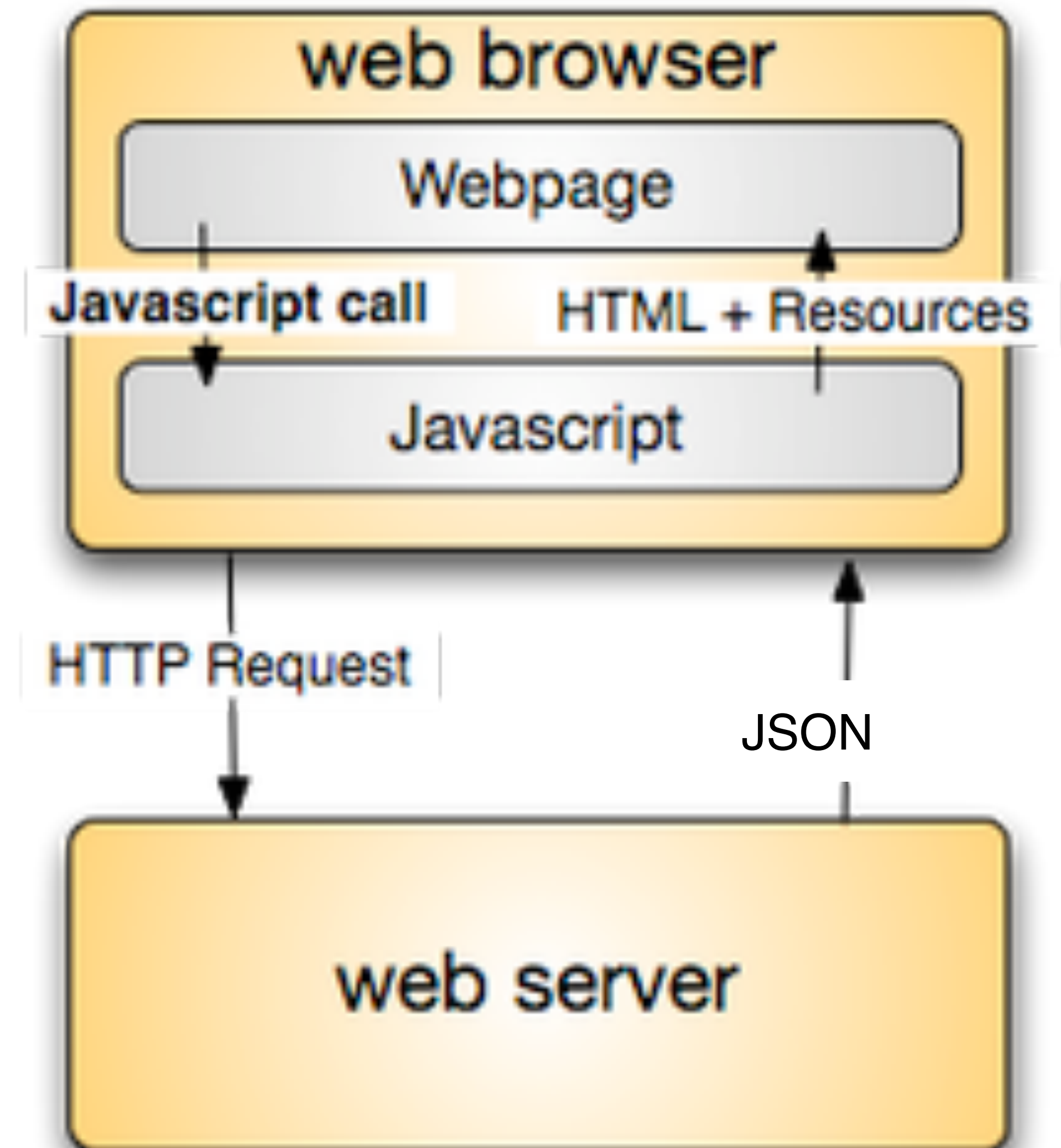


Fetch API

## Traditional web model



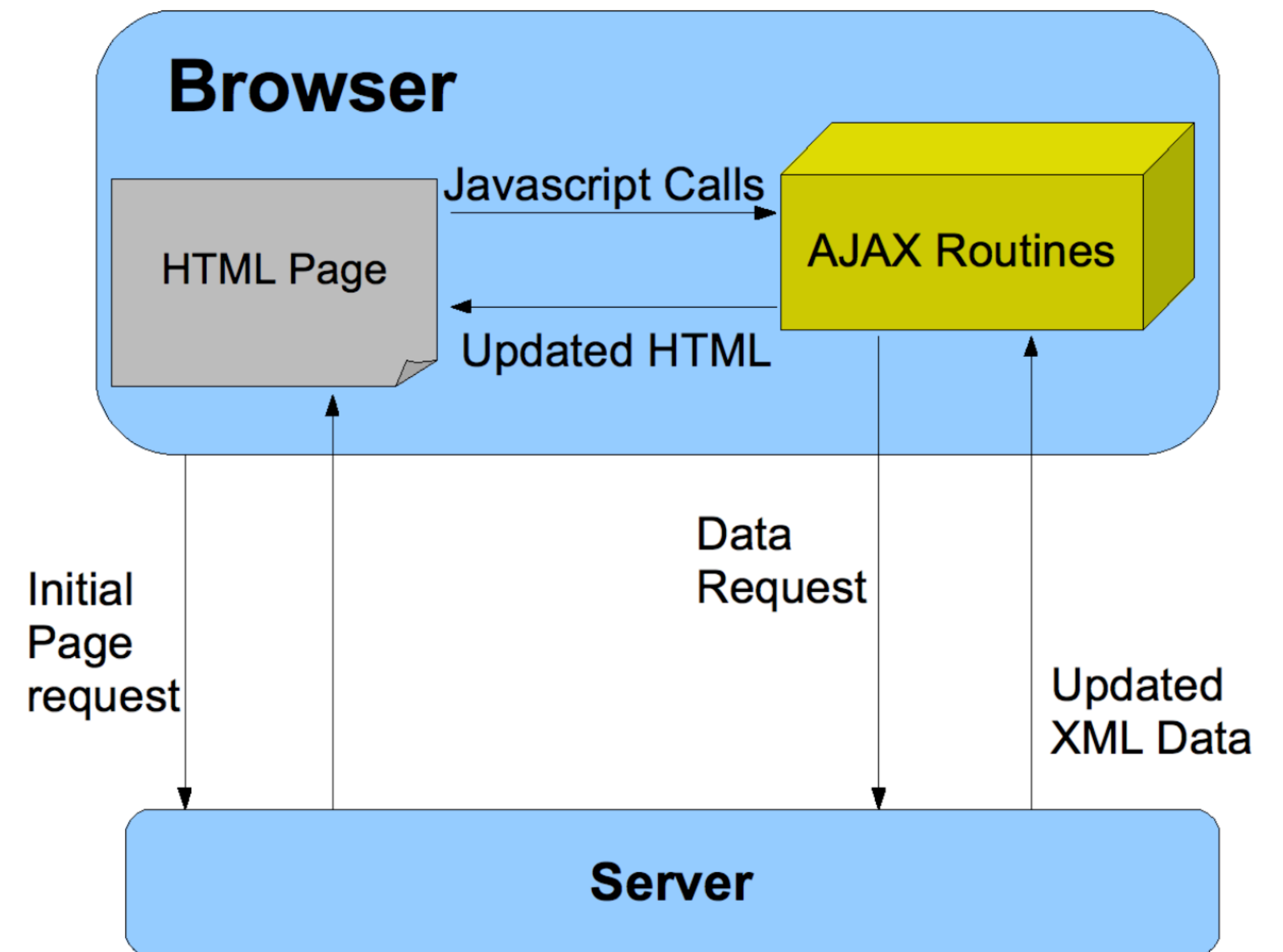
## AJAX web model





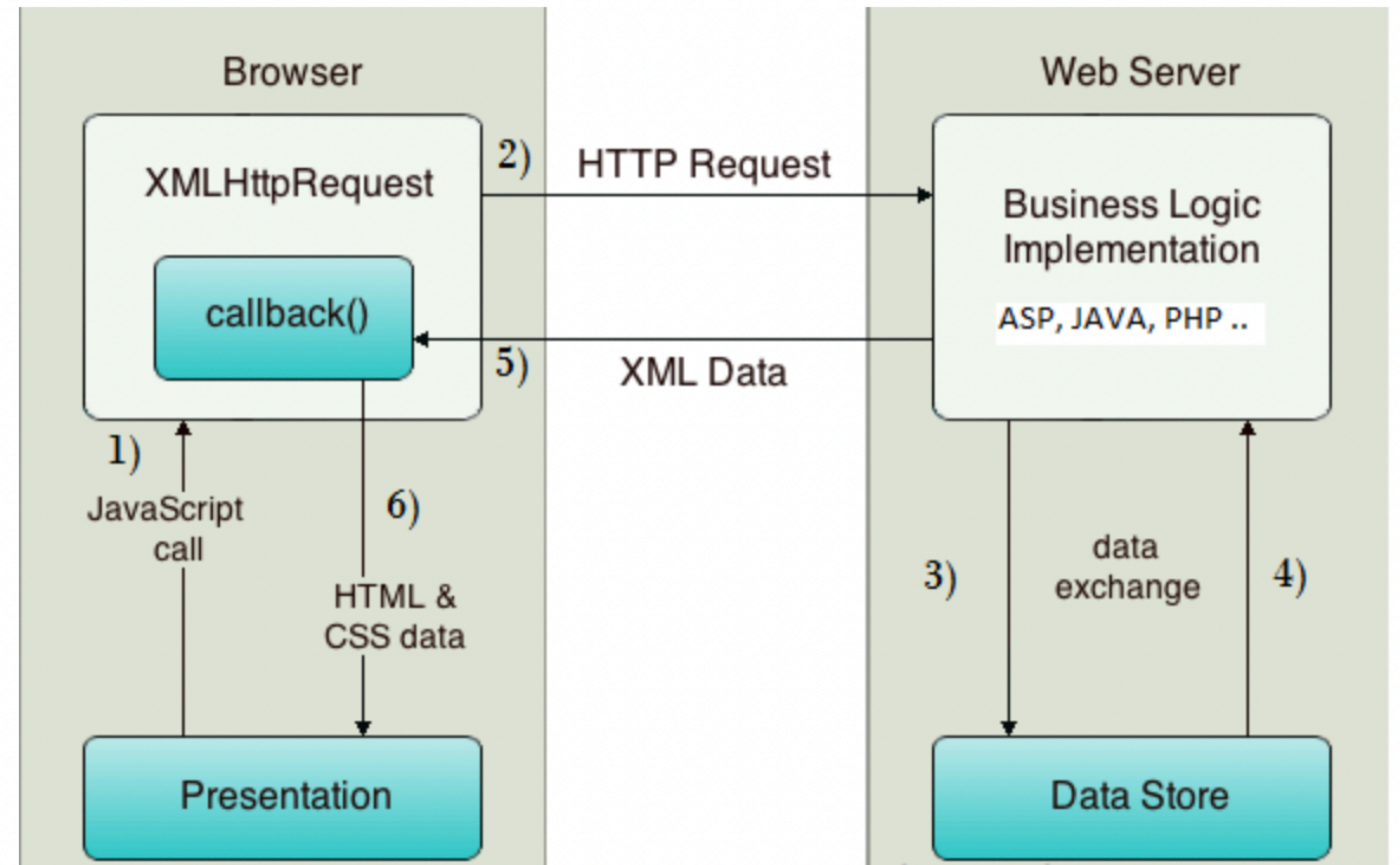
# Ajax

- Initially stood for :  
**Asynchronous JavaScript And XML,**
- A programming practice of building complex, dynamic webpages using a technology known as **XMLHttpRequest**.
- Ajax allows you to update parts of the DOM of an HTML page instead without the need for a full page refresh.
- Ajax also lets you work **asynchronously**:
  - ➡ code continues to run while the targeted part of the web page is trying to reload



# XMLHttpRequest

- XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full page refresh.
- This enables a Web page to update just part of a page without disrupting what the user is doing.
- XMLHttpRequest is used heavily in AJAX programming. .



# XMLHttpRequest

- Retrieval of data from XHR for the purpose of continually modifying a loaded web page is the underlying concept of Ajax design.
- Despite the name, XHR can be used with protocols other than HTTP and data can be in the form of not only XML but also JSON, HTML or plain text

```
var xmlhttp;  
  
if (window.XMLHttpRequest) {  
    xmlhttp = new XMLHttpRequest();  
    xmlhttp.open("GET", filepath, false);  
    xmlhttp.send(null);  
}
```

<https://en.wikipedia.org/wiki/XMLHttpRequest>



# Fetch API

- It is the newest standard for dealing with XMLHttpRequest
- The Fetch API provides an interface for fetching resources (including across the network).
- It uses XMLHttpRequest, but the API provides a more powerful and flexible feature set.

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

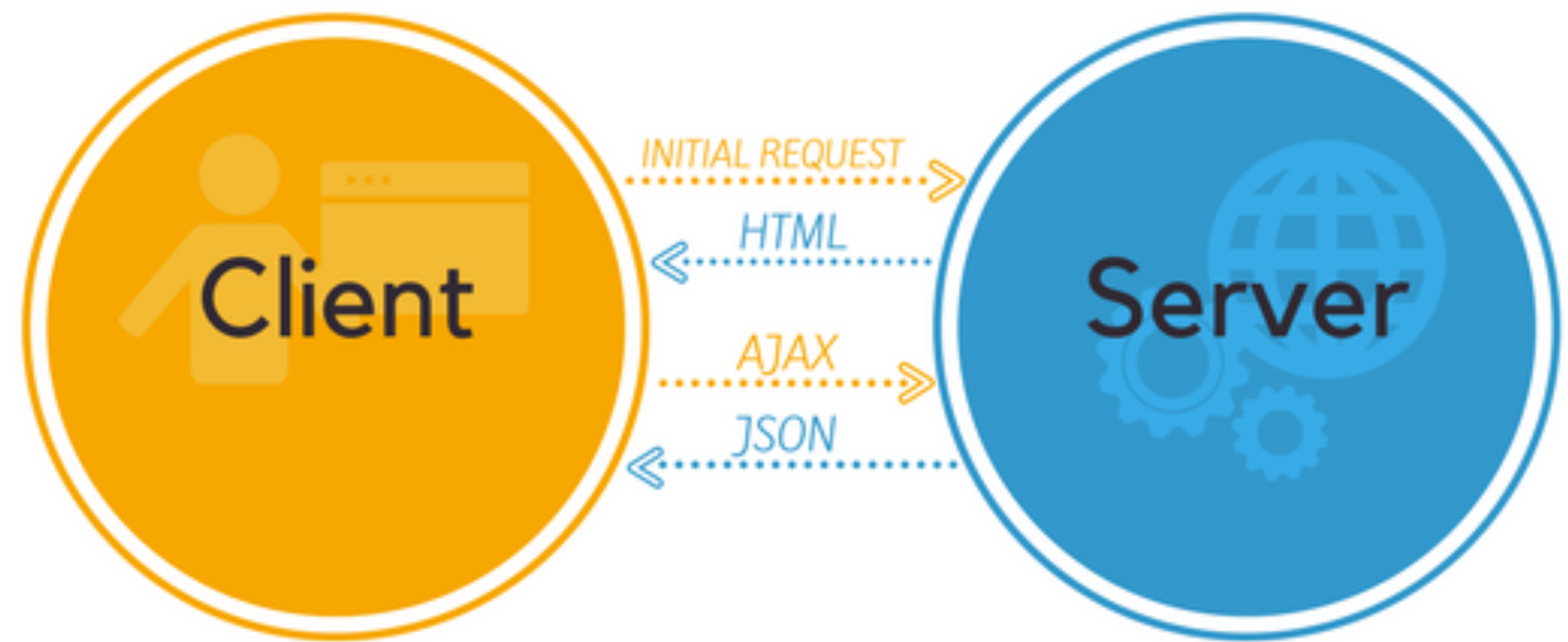


# Fetch API

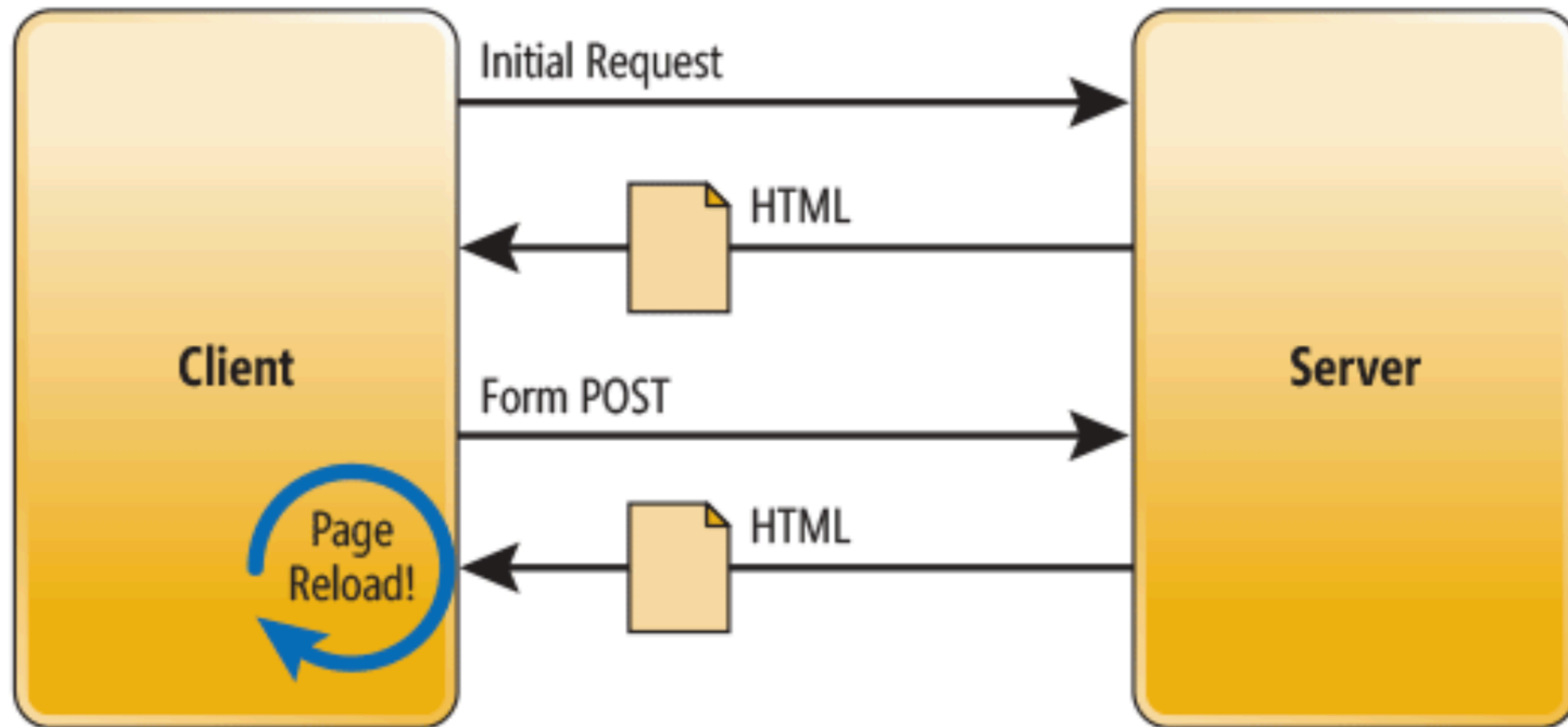
- Here we are fetching a JSON file across the network and printing it to the console.
- The simplest use of `fetch()` takes one argument — the path to the resource you want to fetch — and returns a promise containing the response (a `Response` object).
- This is just an HTTP response, not the actual JSON. To extract the JSON body content from the response, we use the `json()` method

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

# Single Page Applications

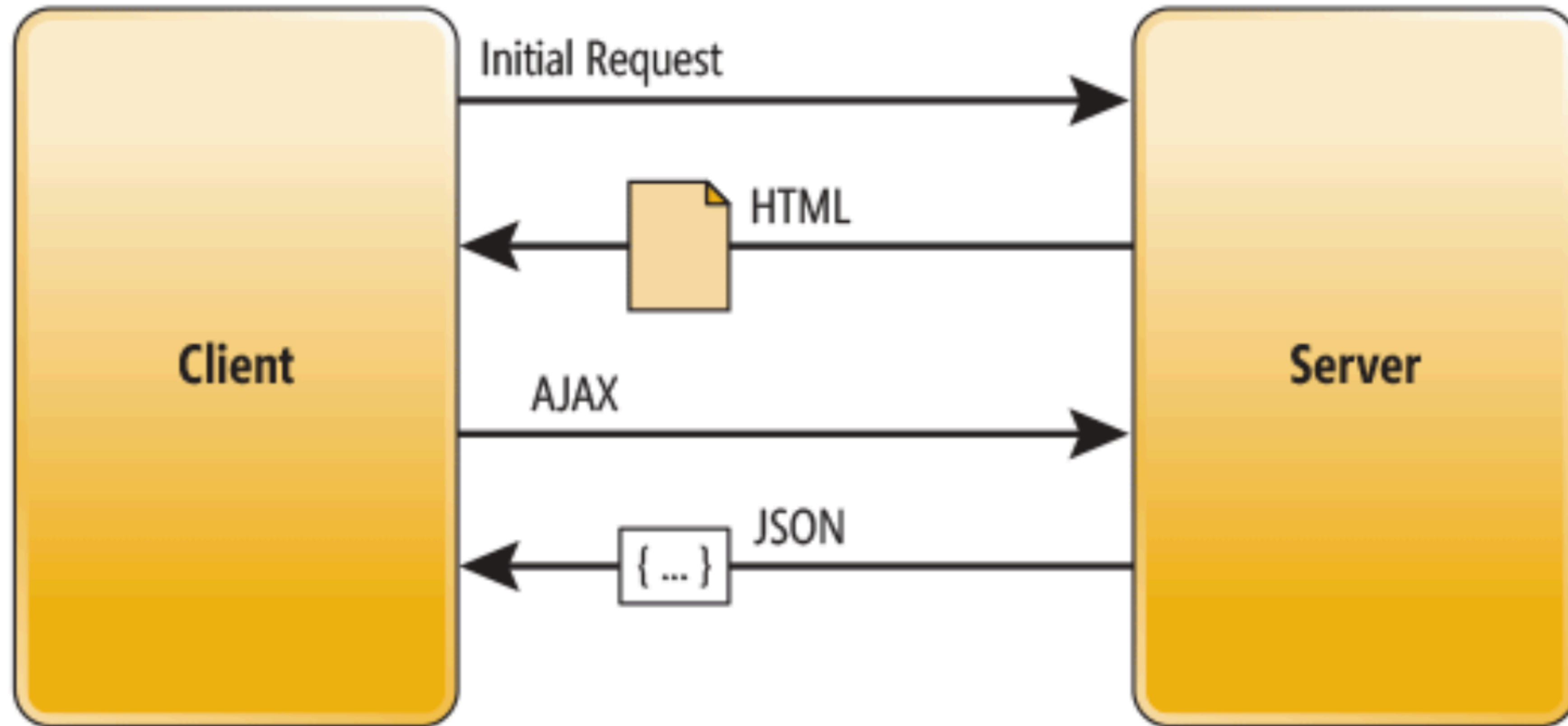


# Traditional Page Lifecycle



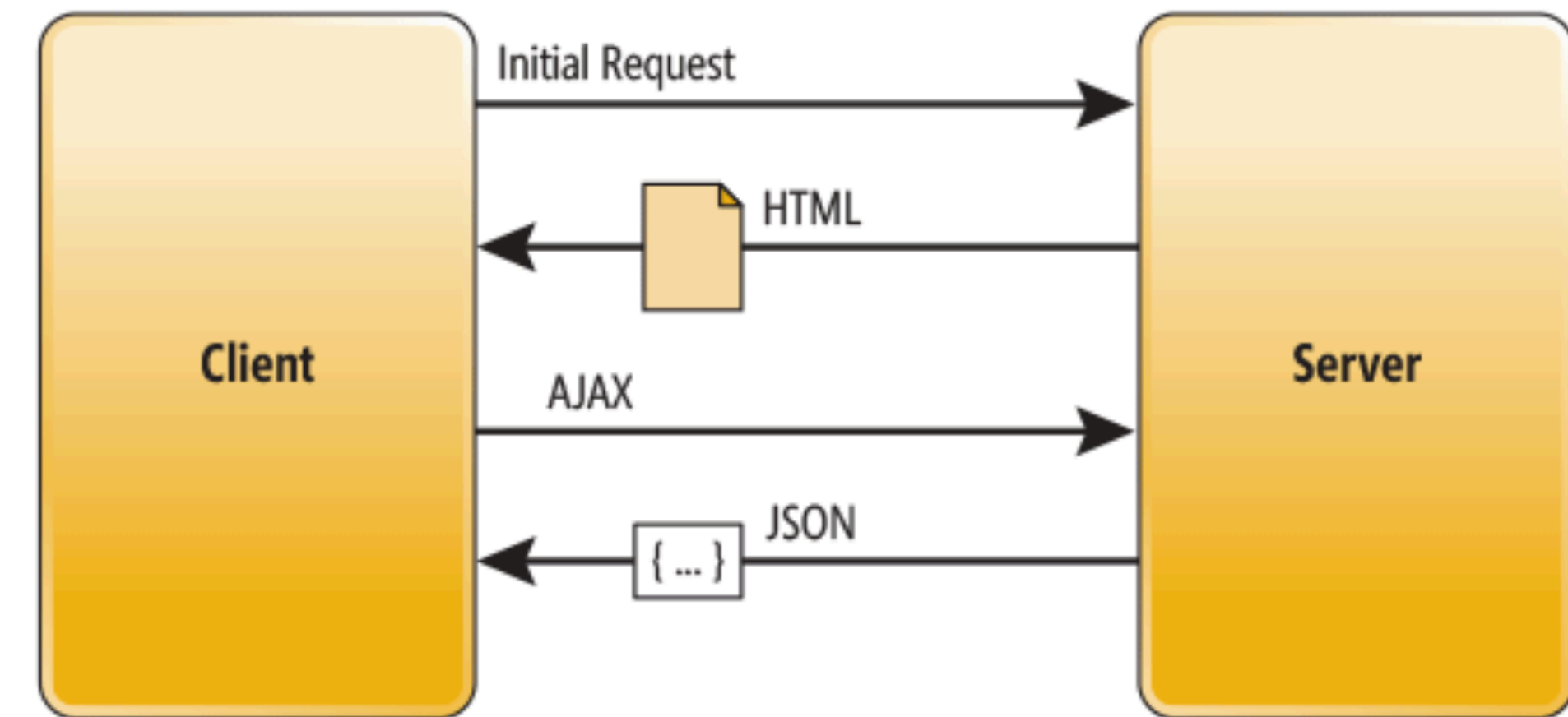


# Single Page Application Lifecycle



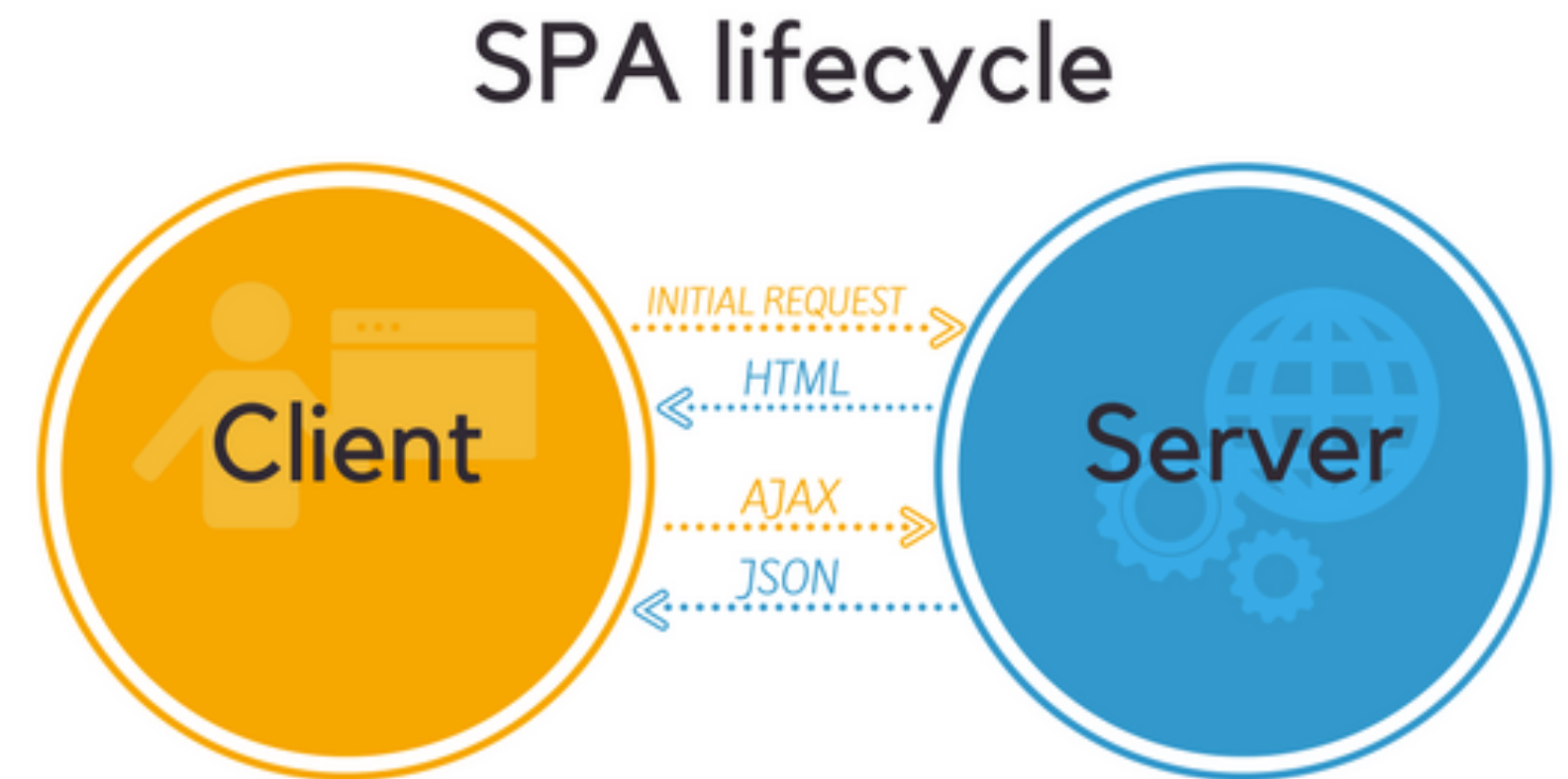
# Single Pages Apps (SPAs)

- Single Page Applications (SPAs) are a web app served up as a single HTML request.
- One initial page load of HTML
- Dynamic features via sophisticated Javascript/AJAX incorporated into the page
- Built with a client-side library or framework (Angular, Ember, React, Vue, Svelte)
- Interact with Rest backends - using JSON default data format



# Key Features of SPAs

- Back-end language agnostic
- Apps usually driven by data and events
- Enhanced Performance & User Experience
- Decoupling/testability
- Easier to build/maintain
- Heavy JS lifting on the client, lighter back-end
- Easier to provide offline operation





## SPA - Performance

- Load time - One file each of HTML, CSS, JS, static files not dynamic
- Less data transfer: XHR calls only send raw data, not HTML markup
- Load distribution: dramatically less load on server, by distributing it to clients

## SPA - UX

- AJAX and SPAs have raised the bar for user expectations
- Besides actually being faster, JS interactions make apps feel more responsive
- Immediate feedback on click
- Smaller data transfer means faster responses



# Svelte is a tool for building fast web Front Ends

It is similar to JavaScript frameworks such as React and Vue, which share a goal of making it easy to build slick interactive user interfaces.

But there's a crucial difference:

- Svelte converts your app into ideal JavaScript at build time, rather than interpreting your application code at run time.
- This means you don't pay the performance cost of the framework's abstractions, and you don't incur a penalty when your app first loads.



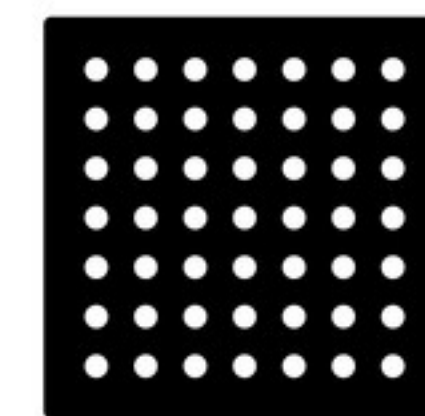
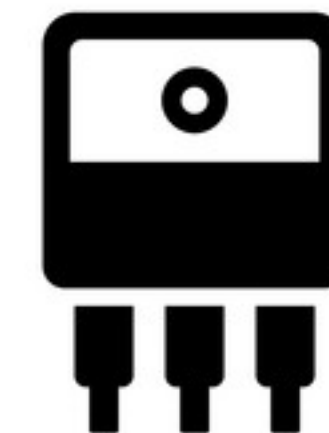
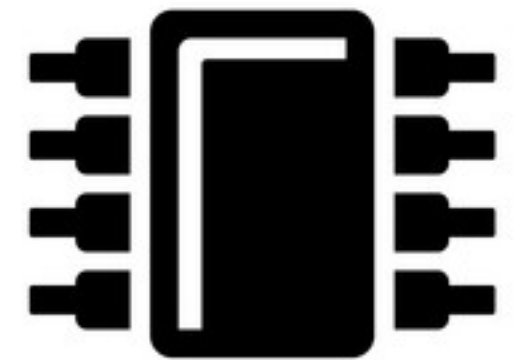
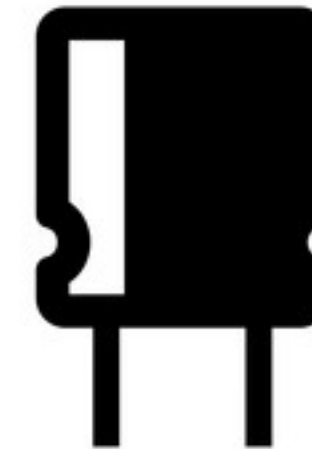
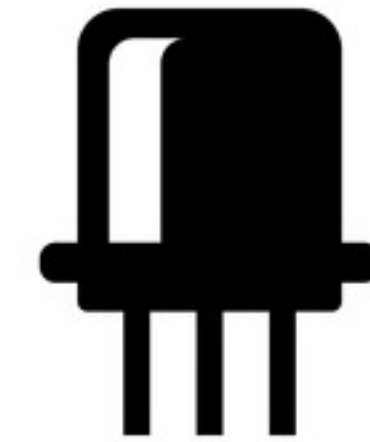
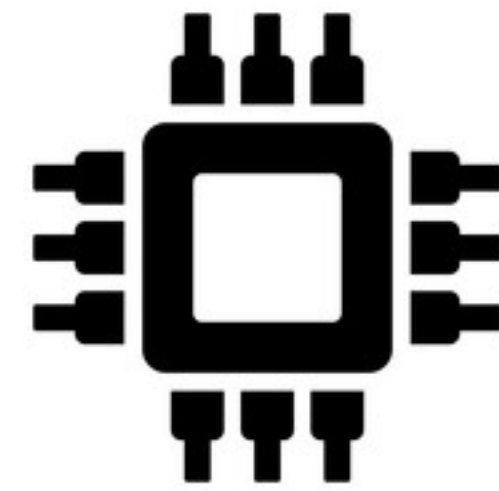
## What is Svelte?

*You can build your entire app with Svelte, or you can add it incrementally to an existing codebase. You can also ship components as standalone packages that work anywhere, without the overhead of a dependency on a conventional framework.*



# Svelte Components

- Modern Web development is very much focused on components,
- What is a component?
  - A component is an atomic part of the application that is self-contained and optionally references other components to compose its output.
  - It's a compartmentalized part of the application. A form can be a component. An input element can be a component. The whole application is a component.
- Svelte components contain all that's needed to render a piece of the UI.



State

Props

Logic

Events

Bindings

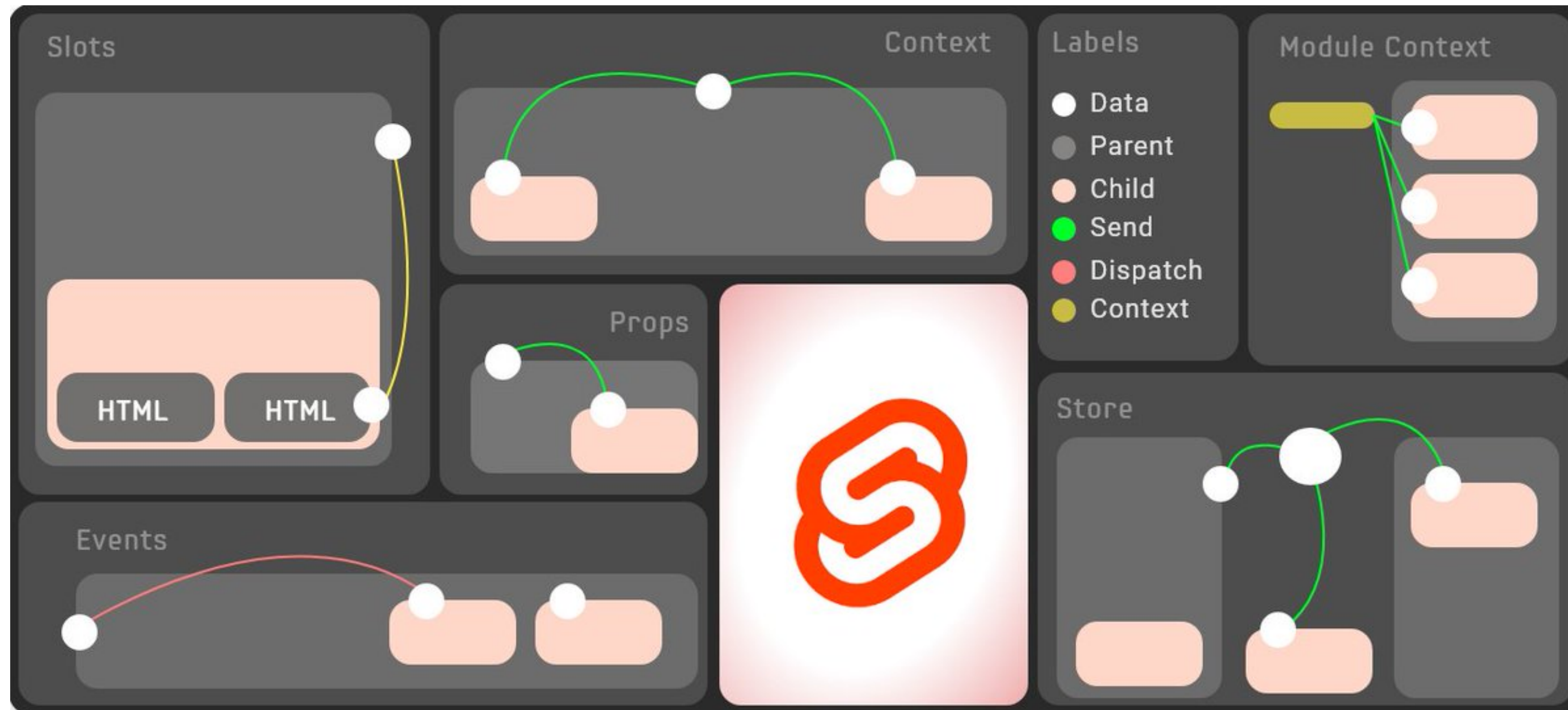
Stores

Context

Lifecycle

Reactivity

## Key Svelte Concepts



- **Components:** Primary building block for application. Must be named *SomeComponent.svelte*
- **State:** Any data that's needed to make the component render what it's meant to render
- **Props:** To pass data from a component down to its children
- **Logic:** To specify way of expressing logic, like conditionals and loops.
- **Events:** Define a listener for a DOM event directly
- **Bindings:** To create a two-way binding between data and the UI.

## Key Svelte Concepts

- **Context:** A feature to allow a component communicate with multiple descendants
- **Stores:** A feature to allow unrelated components to talk to each other
- **Lifecycle:** To specify way a component can be notified of important events in its lifecycle
- **Reactivity:** Be notified dynamically when changes to occur a variable



← Introduction / Basics →

## Introduction

Welcome to the Svelte tutorial. This will teach you everything you need to know to build fast, small web applications easily.

You can also consult the [API docs](#) and the [examples](#), or – if you're impatient to start hacking on your machine locally – the [60-second quickstart](#).


## What is Svelte?

Svelte is a tool for building fast web applications.

It is similar to JavaScript frameworks such as React and Vue, which share a goal of making it easy to build slick interactive user interfaces.

But there's a crucial difference: Svelte converts your app into ideal JavaScript at *build time*, rather than interpreting your application code at *run time*. This means you don't pay the performance cost of the framework's abstractions, and you don't incur a penalty when your app first loads.

<https://svelte.dev/tutorial/basics>



The cover of the Svelte Handbook features the title 'Svelte Handbook' at the top right, a teal flask icon, and a large orange Svelte logo in the center. The text 'THE SVELTE HANDBOOK' is written in a mix of bold sans-serif and handwritten fonts. Below the logo, the author's name 'Flavio Lopes' is written in a handwritten style on an orange background. At the bottom, it says 'The book is available at'.

<https://flaviocopes.com/page/download-svelte-handbook/>

## Svelte Introduction



A review of the  
fundamental features of the  
Svelte framework.