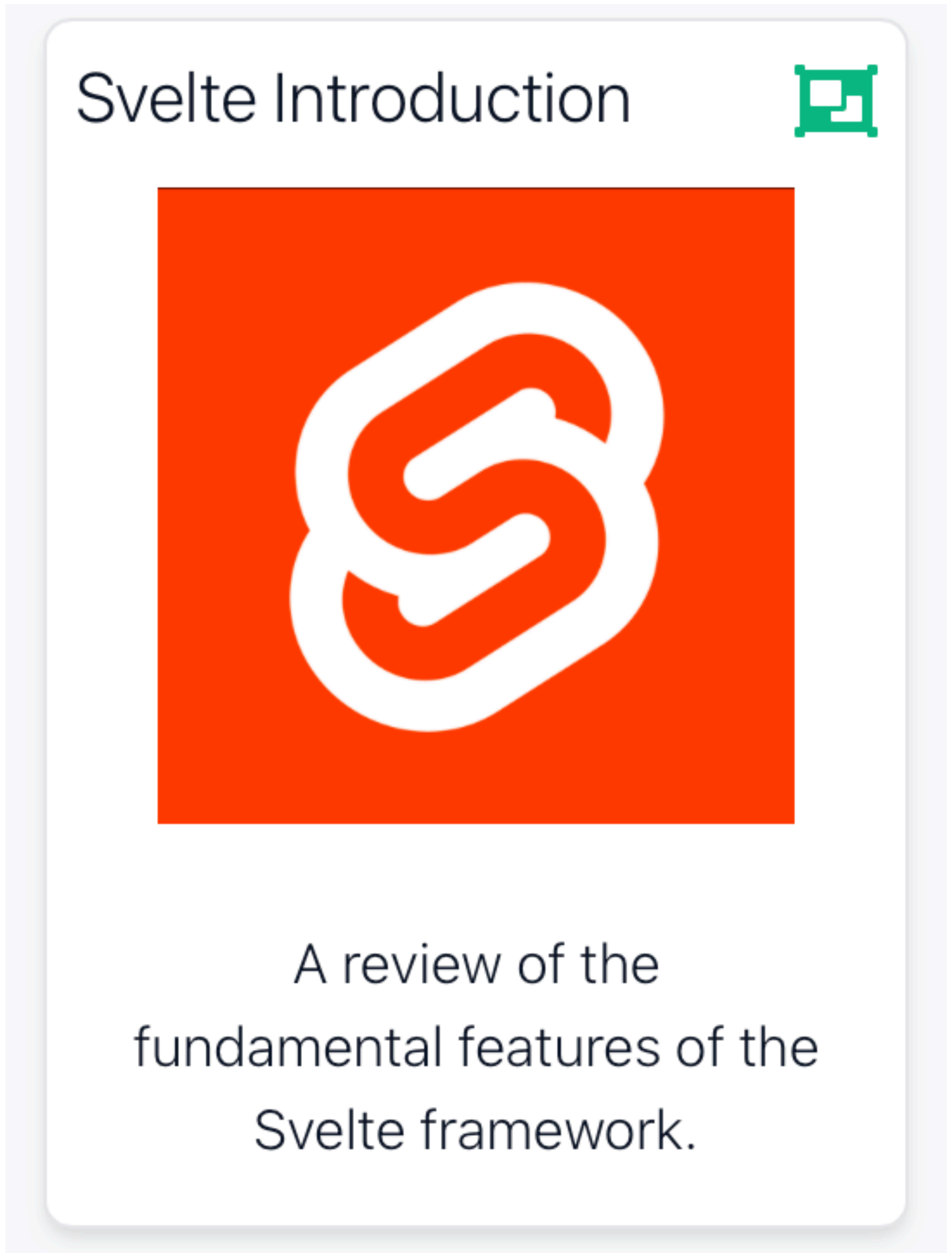
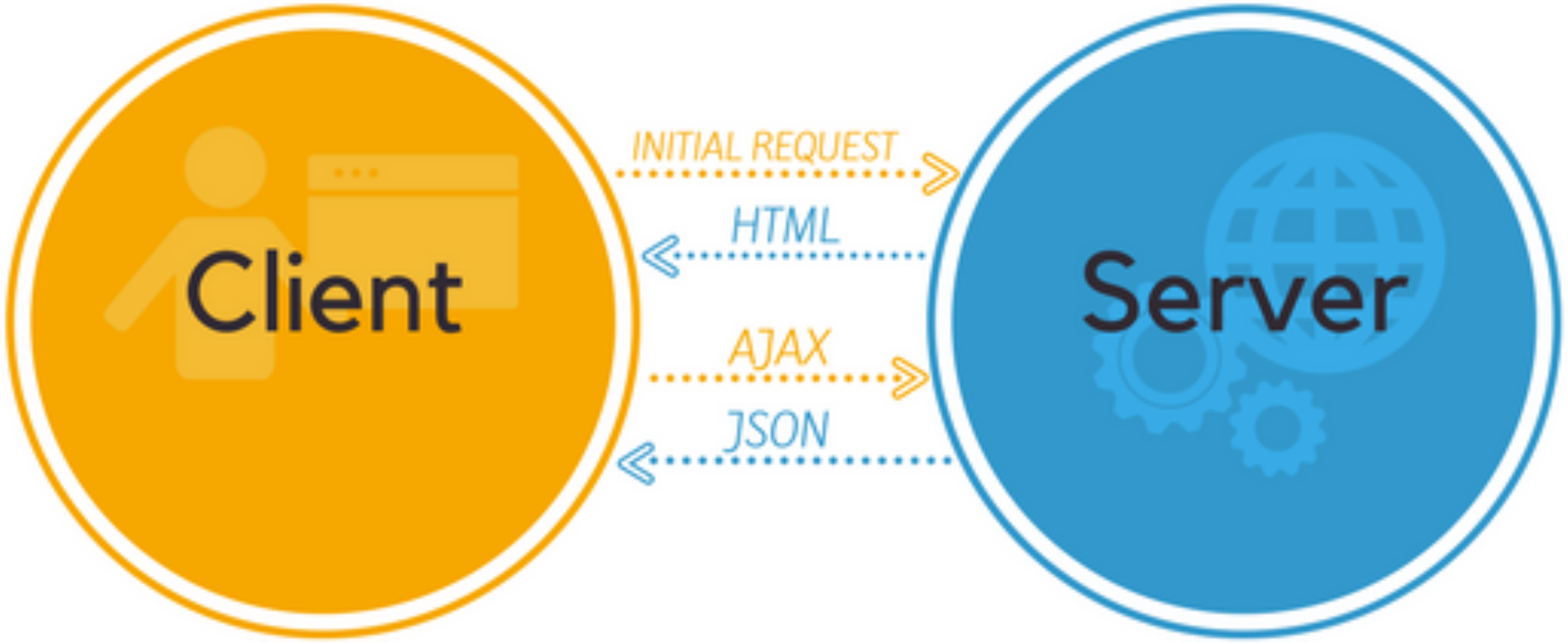


Svelte Introduction

A review of the fundamental features of the Svelte framework.

JS
Javascript
Request
Response
Headers
Body
XmlHttpRequest
Fetch API


Client
INITIAL REQUEST
HTML
AJAX
JSON
Server

SVELTE

AJAX

(Asynchronous JavaScript and XML)

JS

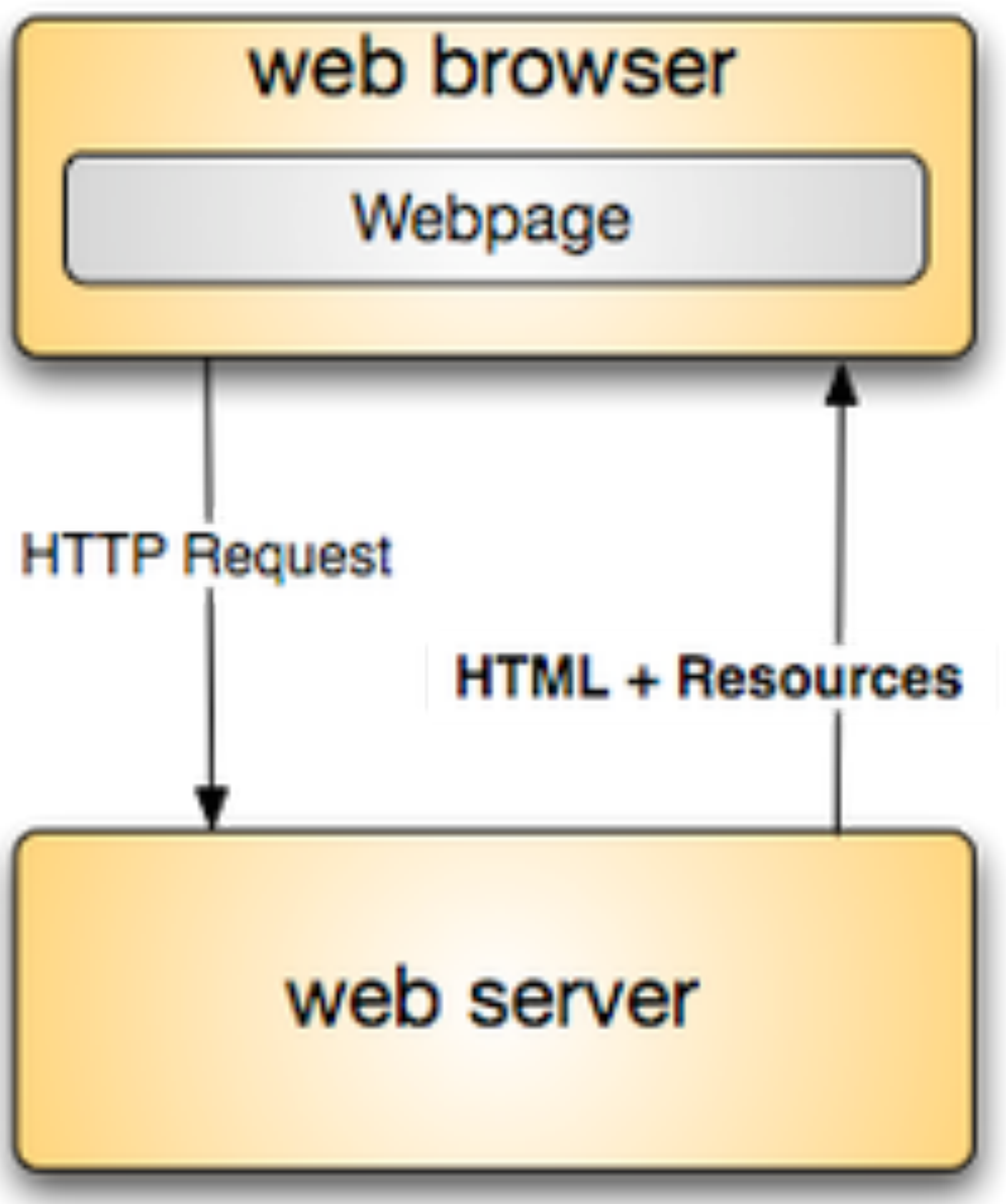
Javascript

Request
Response
Headers
Body

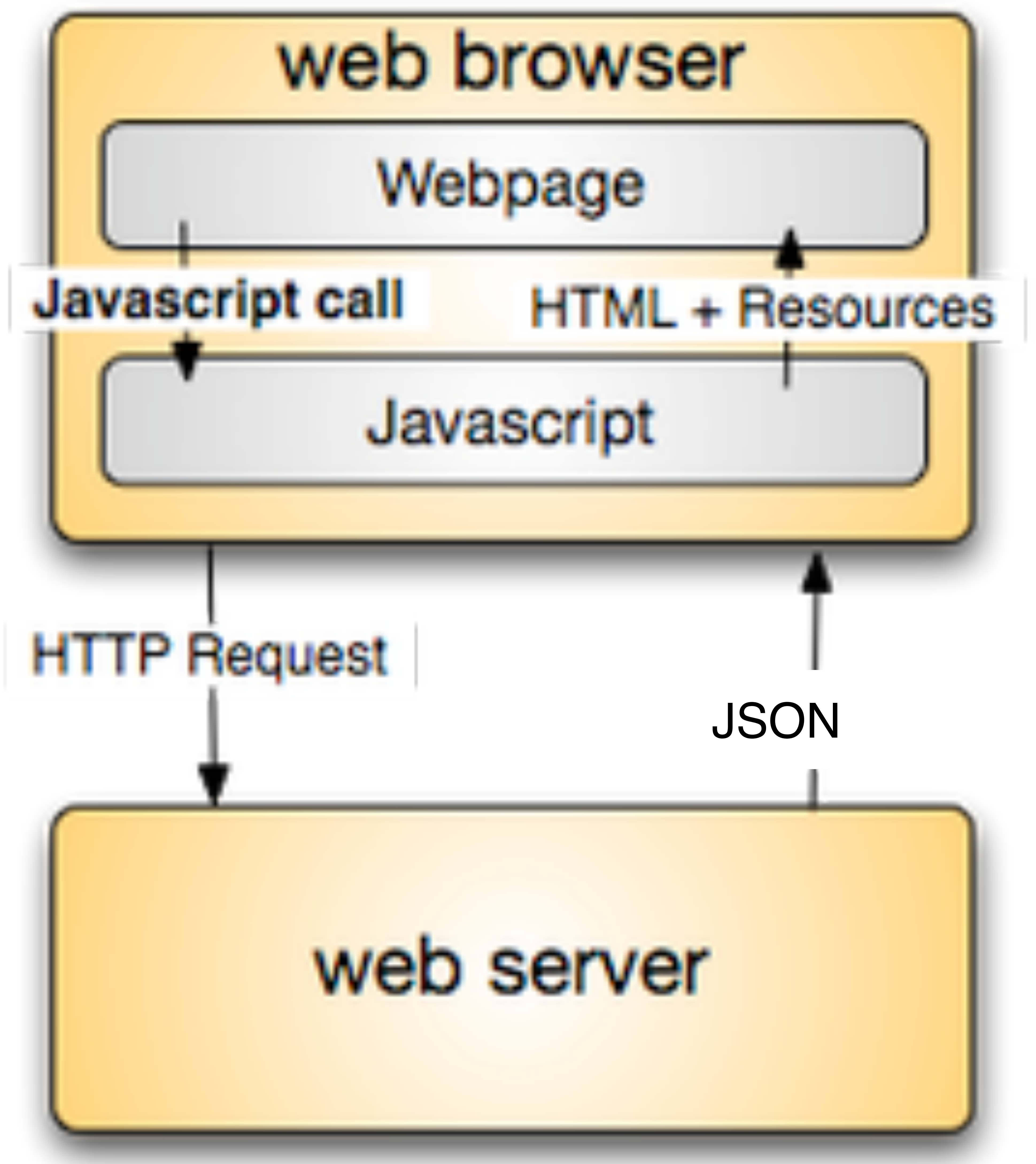
XmlHttpRequest

Fetch API

Traditional web model



AJAX web model



Ajax

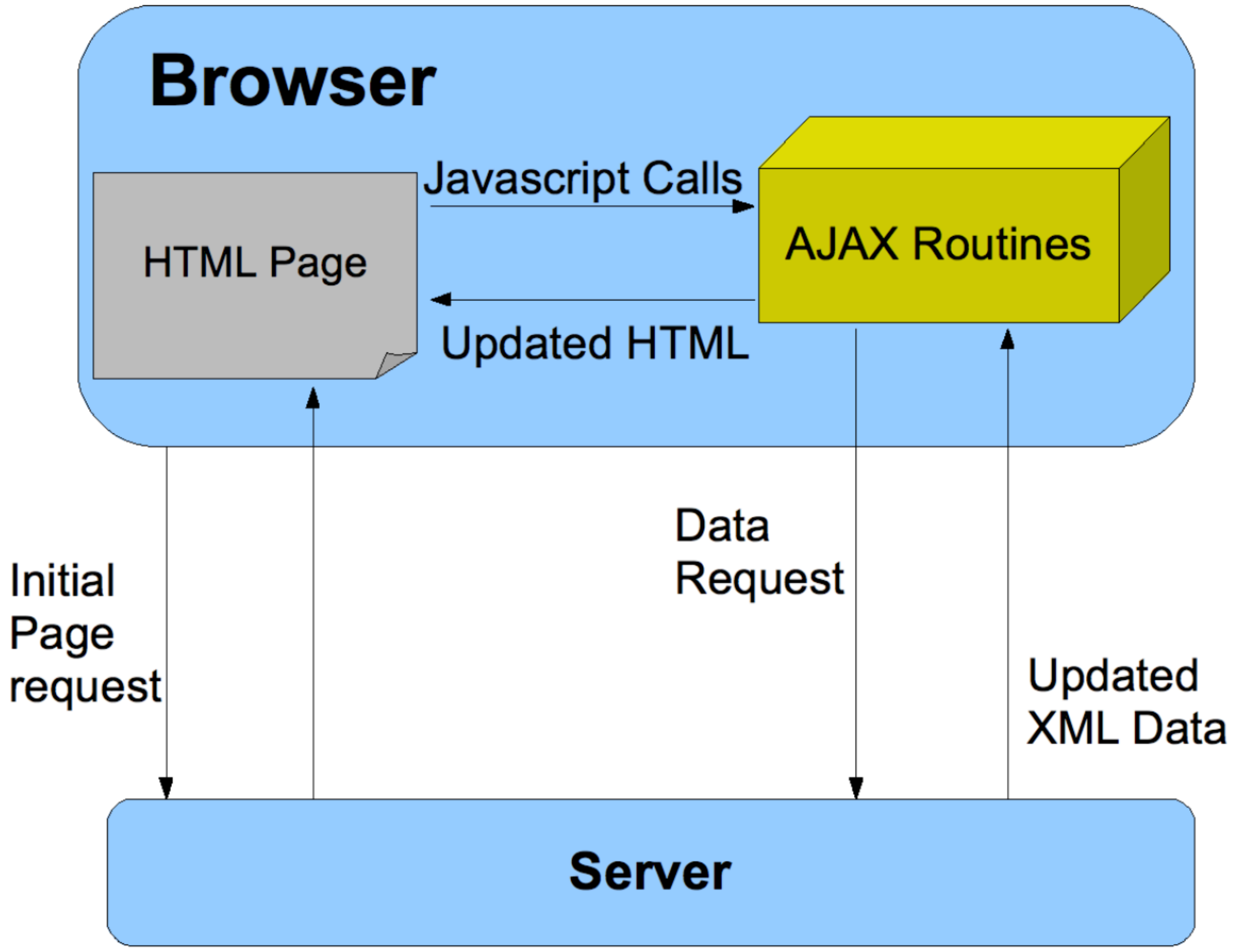
- Initially stood for :
  Asynchronous JavaScript And XML,

- 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

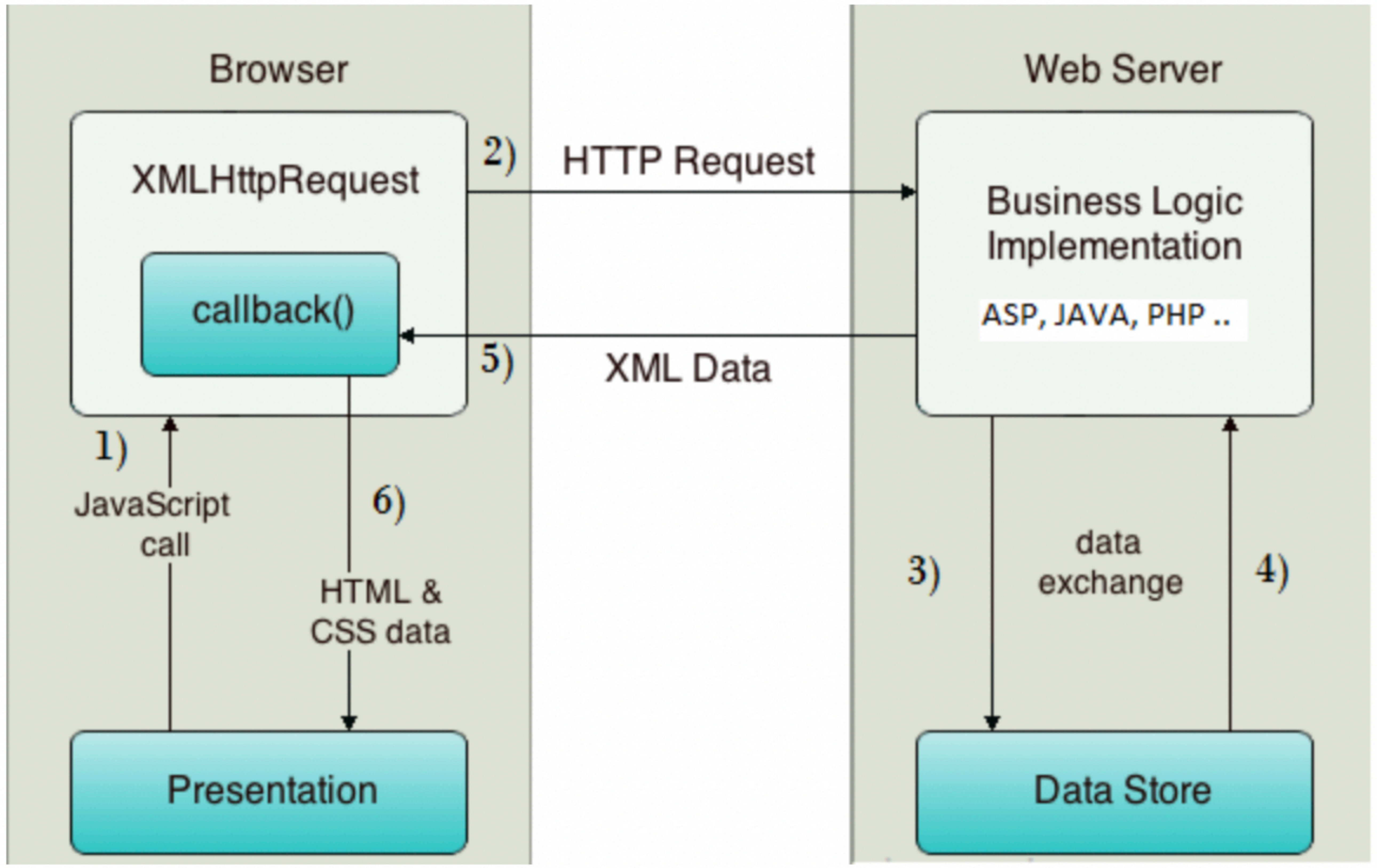


XMLHttpRequest

- 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 HTTPRequest

- 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

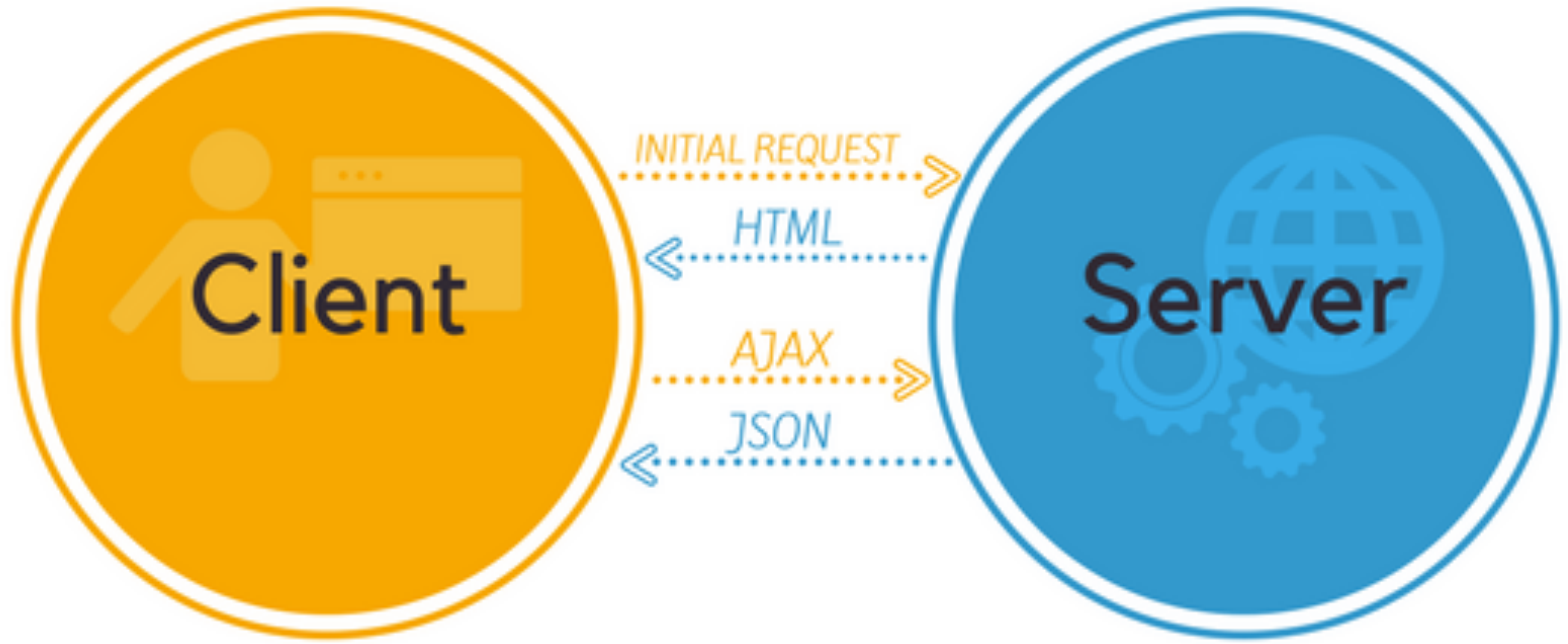
- Here we are fetching a JSON file across the network and printing it to the console.

- 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).

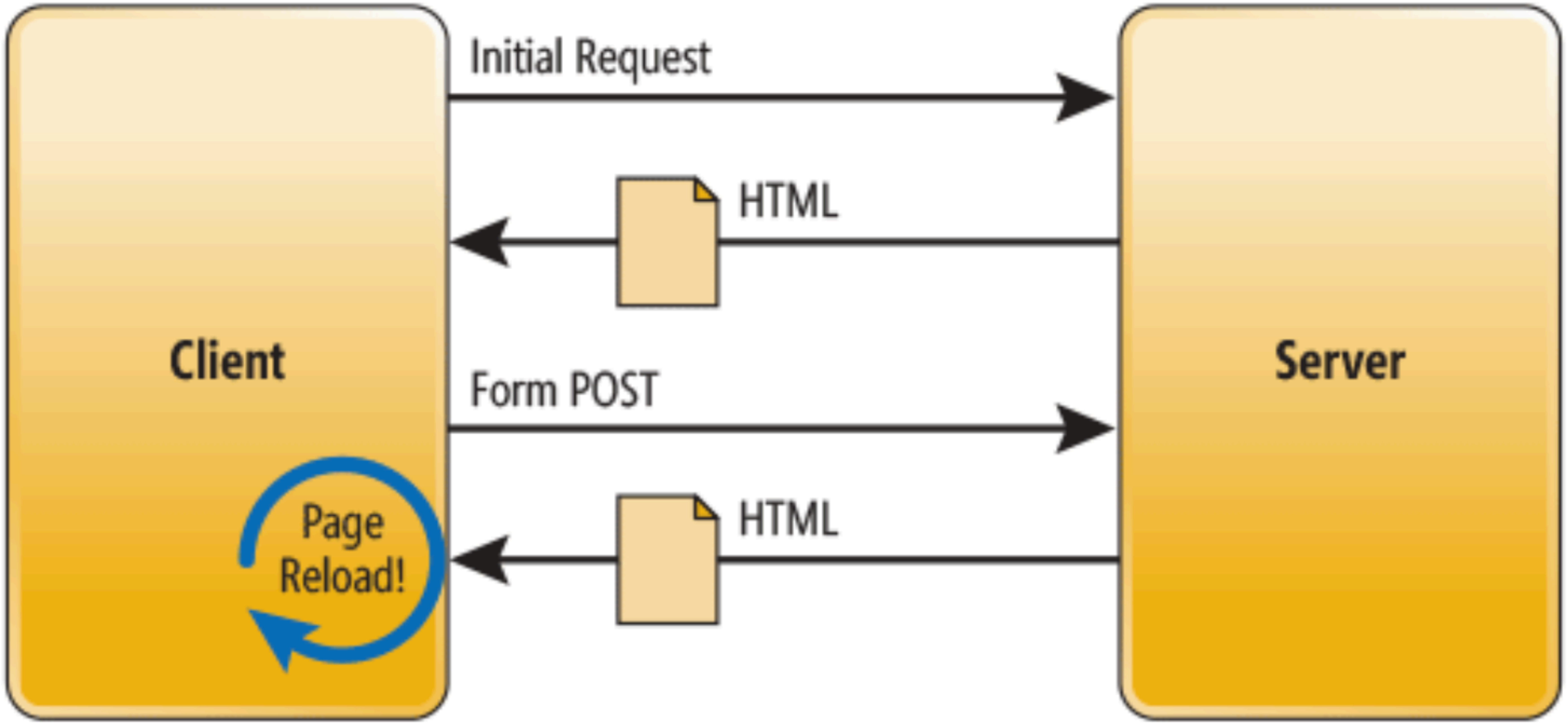
- This is just an HTTP response, not the actual JSON. To extract the JSON body content from the response, we use
the json() method

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

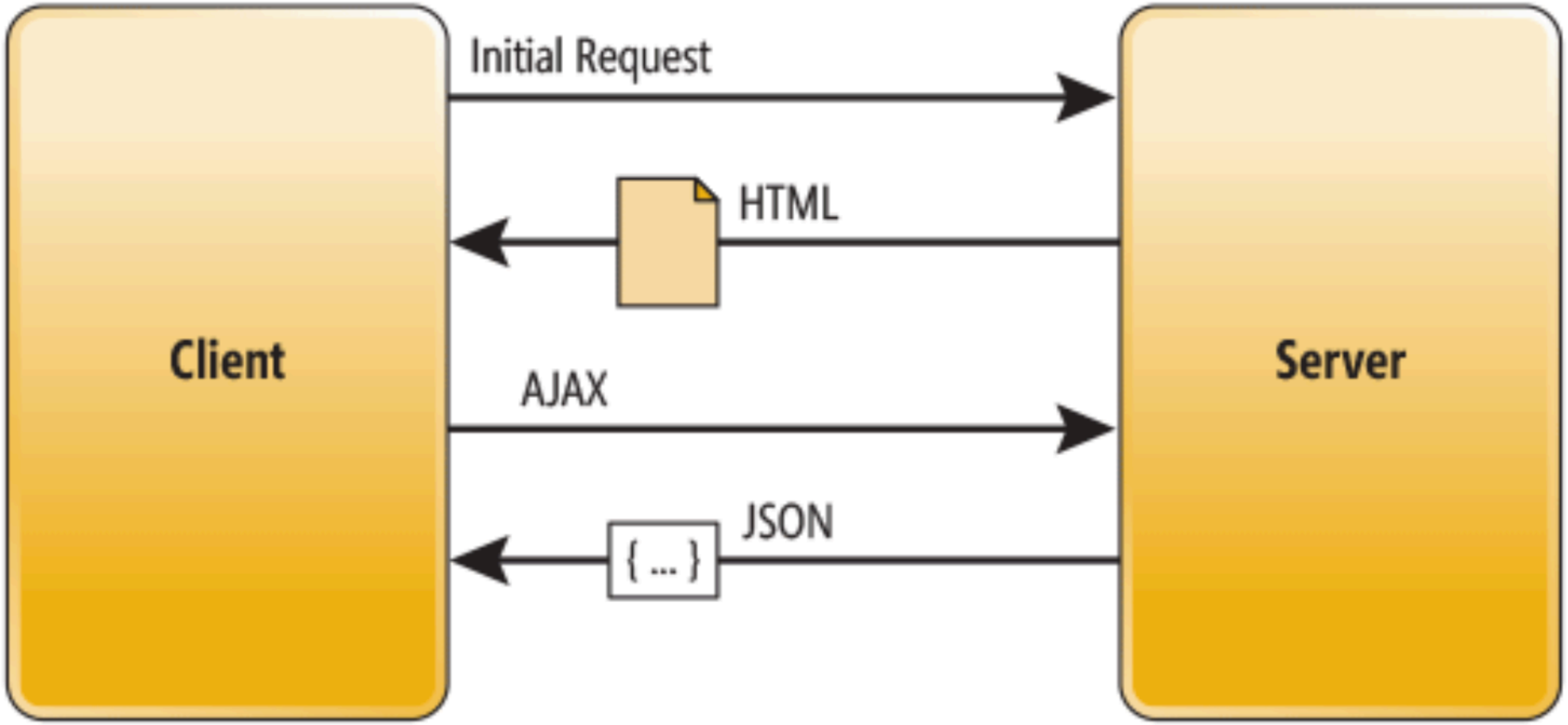
Single Page Applications



Traditional Page Lifecycle

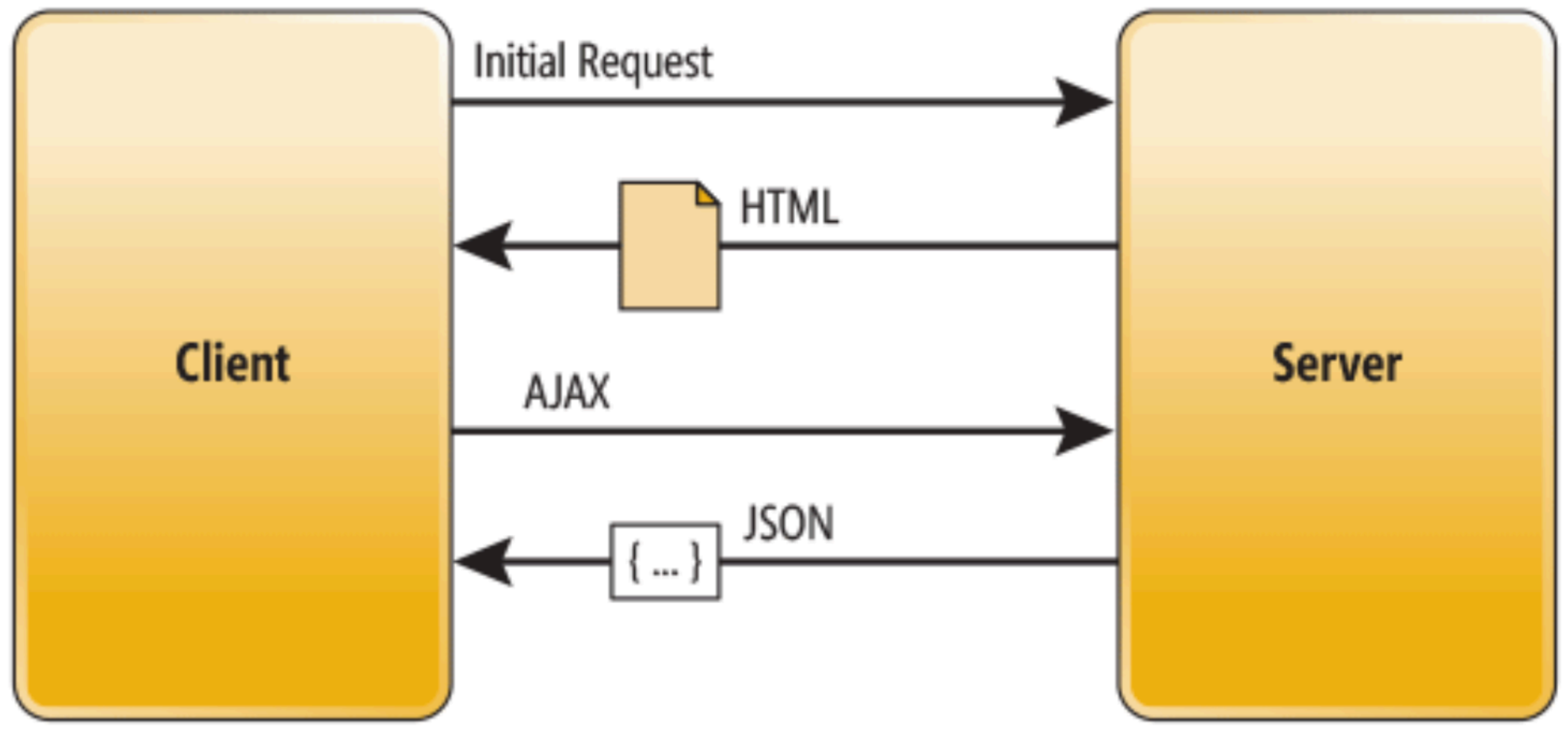


Single Page Application Lifecycle



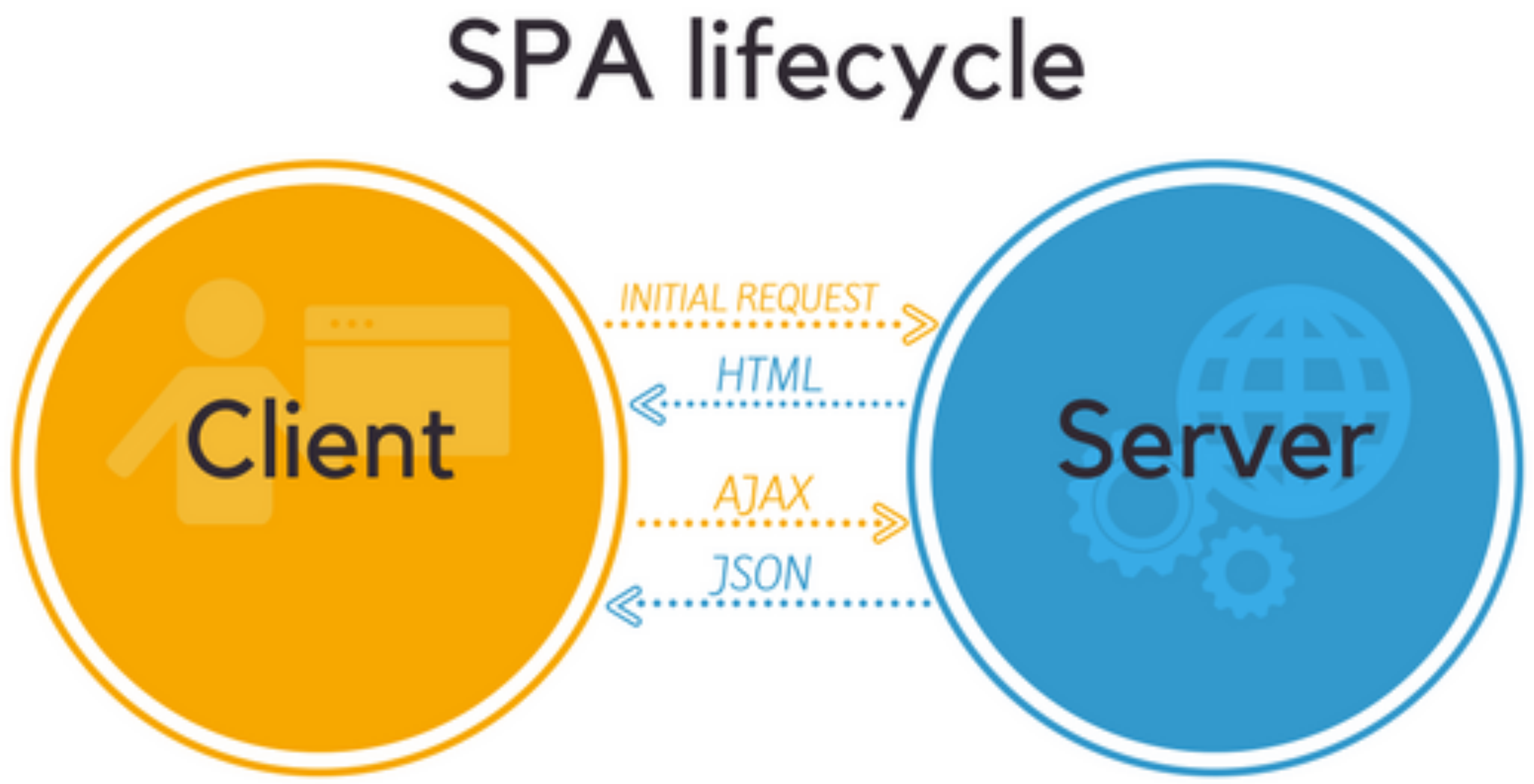
Single Pages Apps (SPAs)

- 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

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:

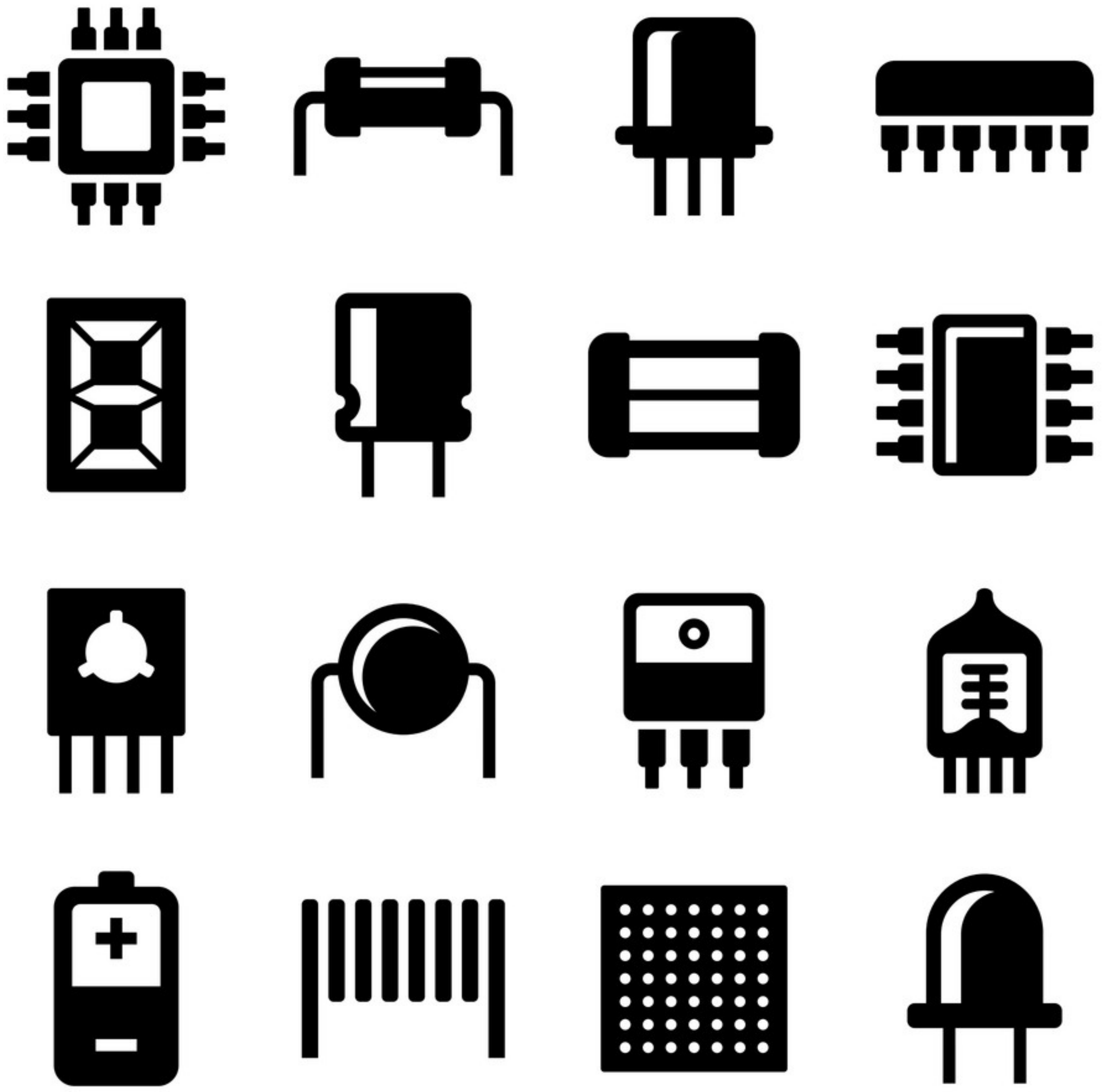
- Svelte converts your app into ideal JavaScript at build time, rather than interpreting your application code at run time.
- This means you don't pay the performance cost of the framework's abstractions, and you don't incur a penalty when your app first loads.

What is Svelte?

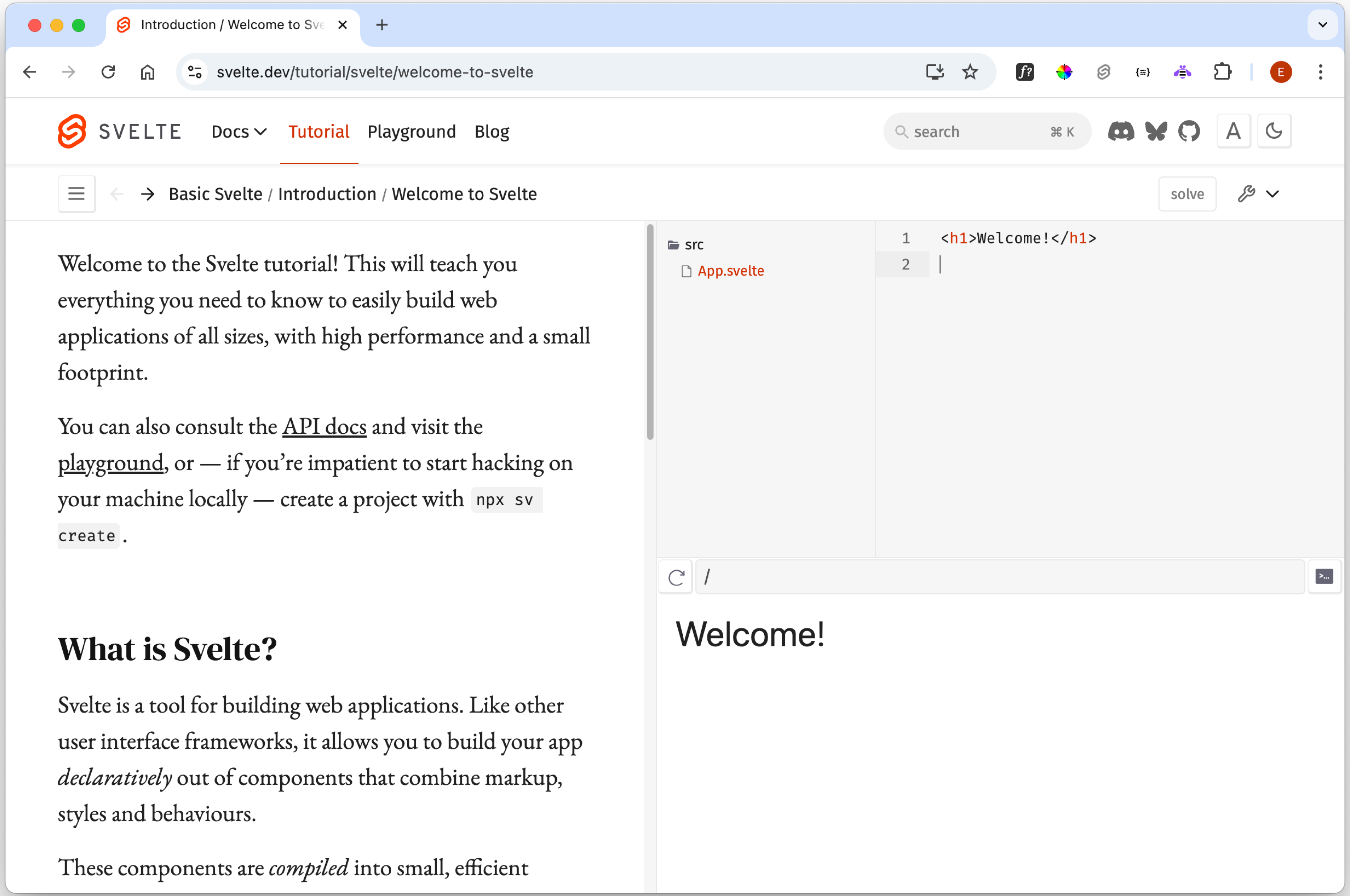
You can build your entire app with Svelte, or you can add it incrementally to an existing codebase. You can also ship components as standalone packages that work anywhere, without the overhead of a dependency on a conventional framework.

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.



https://svelte.dev/tutorial/svelte/welcome-to-svelte


Introduction / Welcome to Sve
svelte.dev/tutorial/svelte/welcome-to-svelte
SVELTE
Docs
Tutorial
Playground
Blog
search
⌘ K
A
Basic Svelte / Introduction / Welcome to Svelte
solve
src
App.svelte
/

Welcome to the Svelte tutorial! This will teach you everything you need to know to easily build web applications of all sizes, with high performance and a small footprint.

You can also consult the API docs and visit the playground, or — if you're impatient to start hacking on your machine locally — create a project with `npx sv create`.

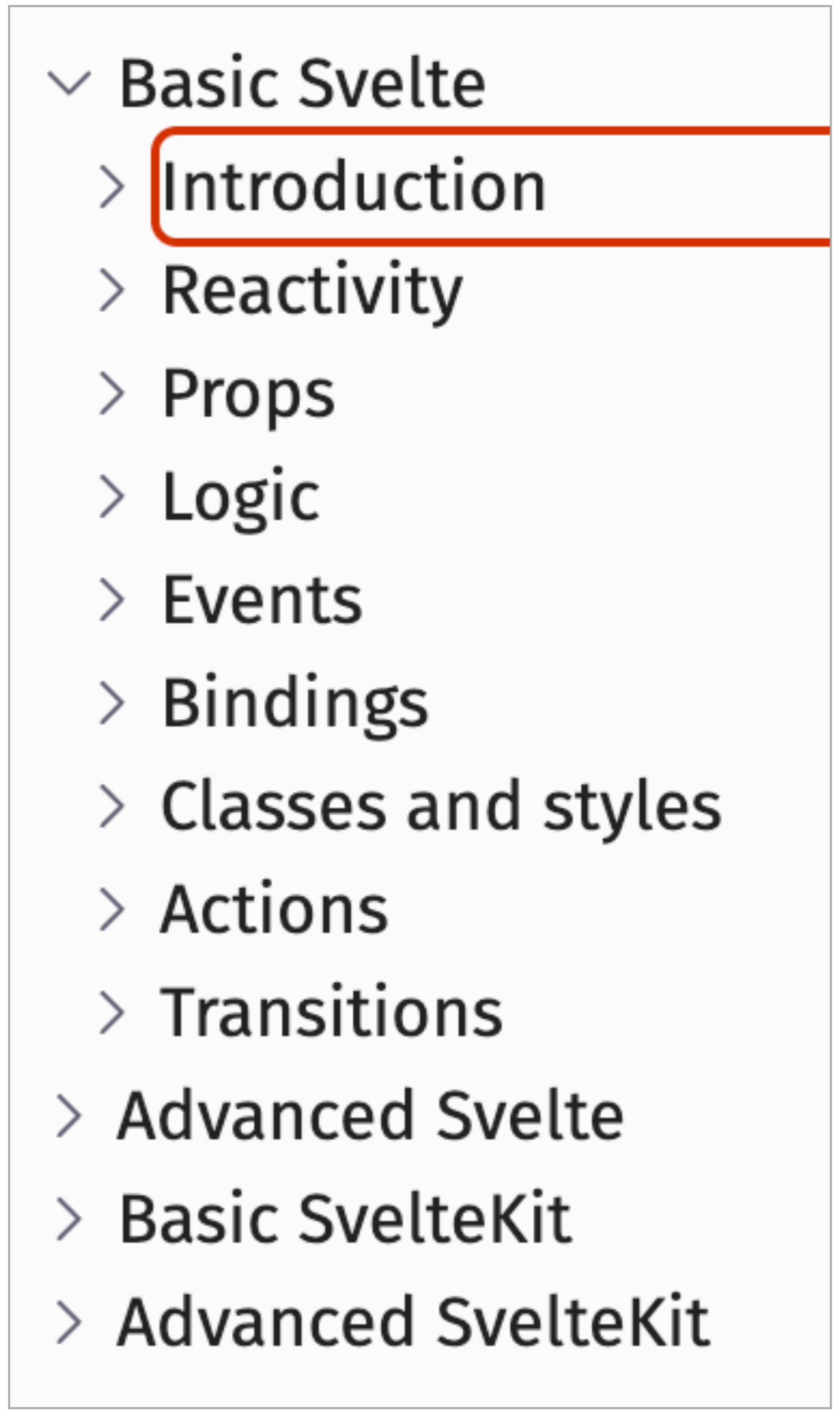
What is Svelte?

Svelte is a tool for building web applications. Like other user interface frameworks, it allows you to build your app *declaratively* out of components that combine markup, styles and behaviours.

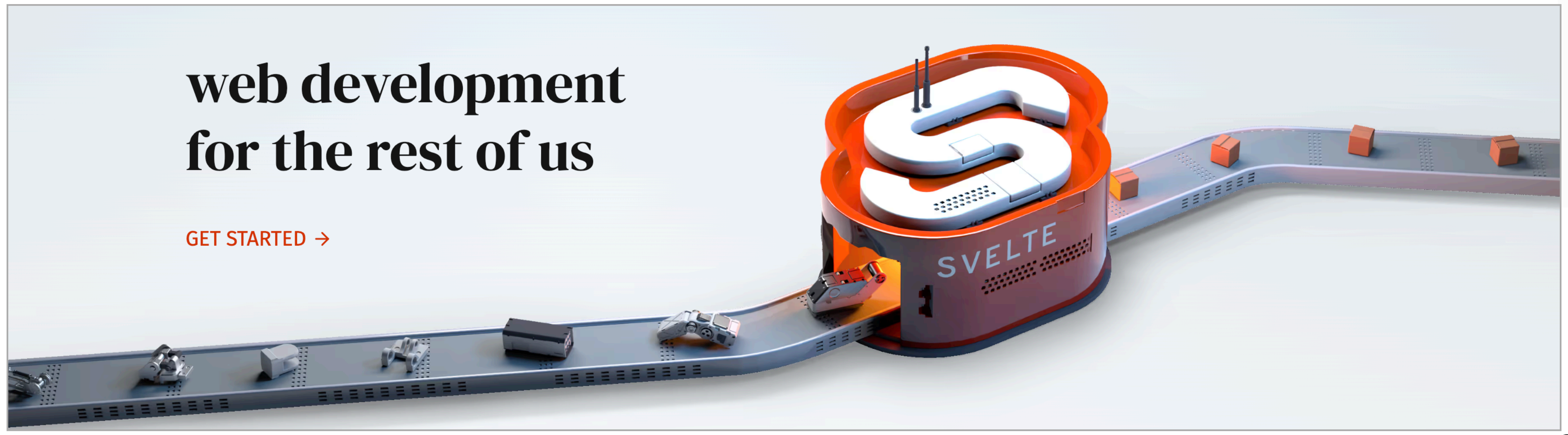
These components are *compiled* into small, efficient

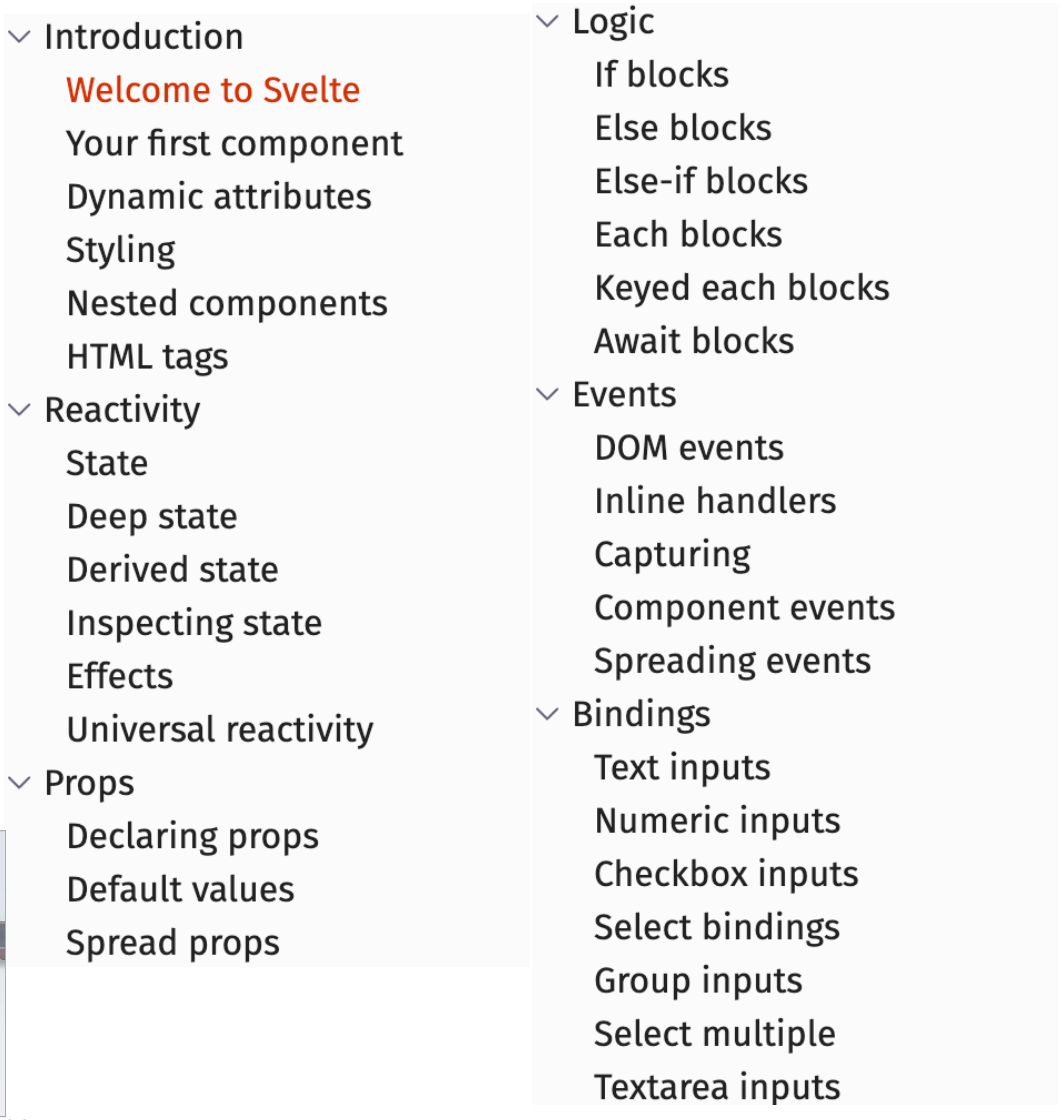
```
<h1>Welcome!</h1>

```

Welcome!

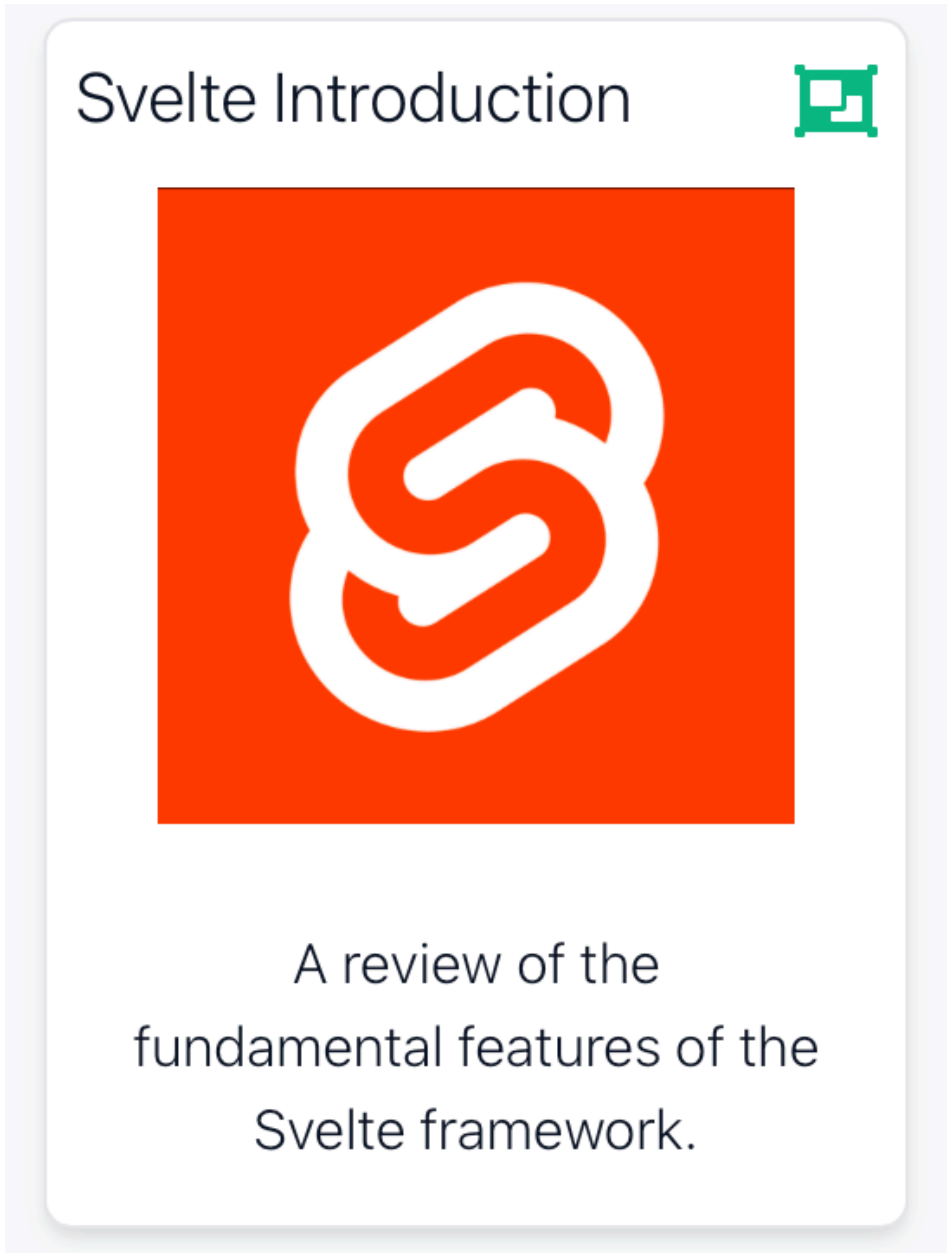


- Basic Svelte
  - Introduction
  - Reactivity
  - Props
  - Logic
  - Events
  - Bindings
  - Classes and styles
  - Actions
  - Transitions
- Advanced Svelte
- Basic SvelteKit
- Advanced SvelteKit





- Introduction
  - Welcome to Svelte
  - Your first component
  - Dynamic attributes
  - Styling
  - Nested components
  - HTML tags
- Reactivity
  - State
  - Deep state
  - Derived state
  - Inspecting state
  - Effects
  - Universal reactivity
- Props
  - Declaring props
  - Default values
  - Spread props
- Logic
  - If blocks
  - Else blocks
  - Else-if blocks
  - Each blocks
  - Keyed each blocks
  - Await blocks
- Events
  - DOM events
  - Inline handlers
  - Capturing
  - Component events
  - Spreading events
- Bindings
  - Text inputs
  - Numeric inputs
  - Checkbox inputs
  - Select bindings
  - Group inputs
  - Select multiple
  - Textarea inputs



Svelte Introduction

A review of the fundamental features of the Svelte framework.