Asynchrous Servlet & XmlHttpRequest

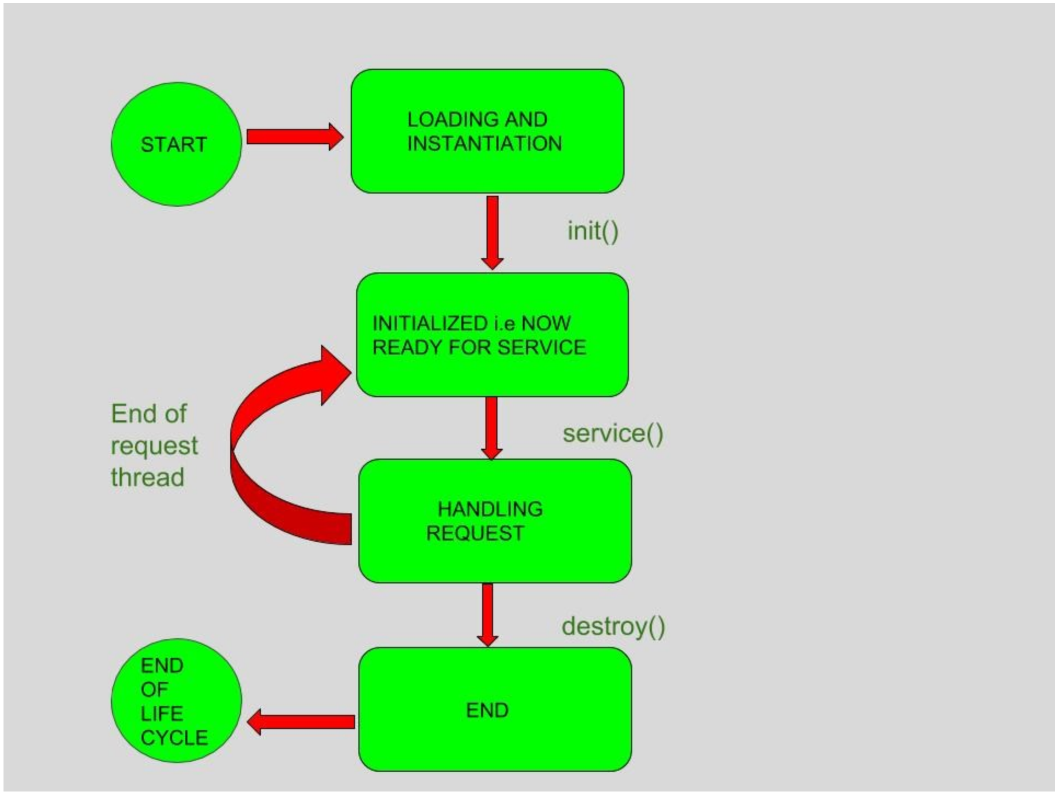
Presented by

Sagar
Java Consultant

Servlet Architecture – Single Thread Model

In traditional servlet architecture, a single thread handles a request from inception to completion.

This model becomes a bottleneck when dealing with long-running tasks or I/O-bound operations. To address this, asynchronous servlets were introduced.

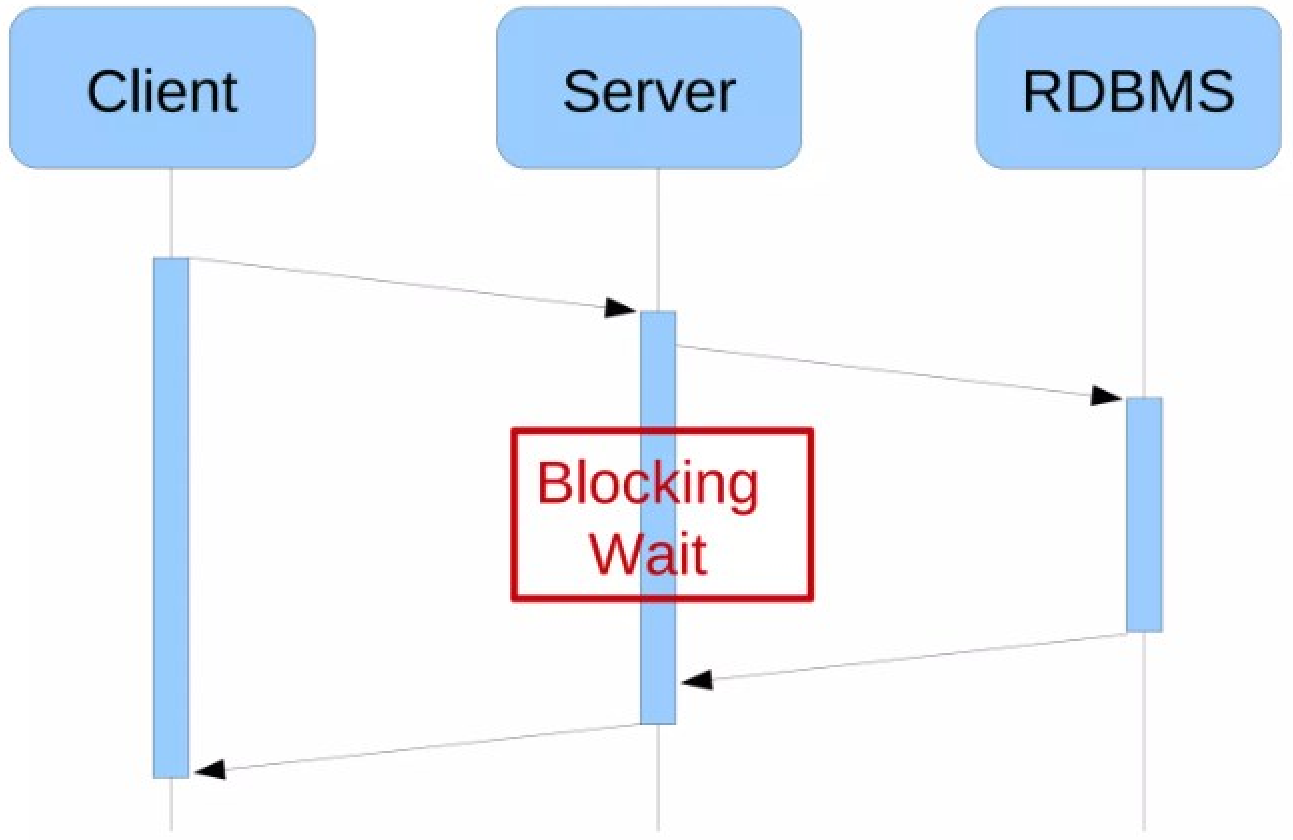


Asynchronous Servlets and Benefits

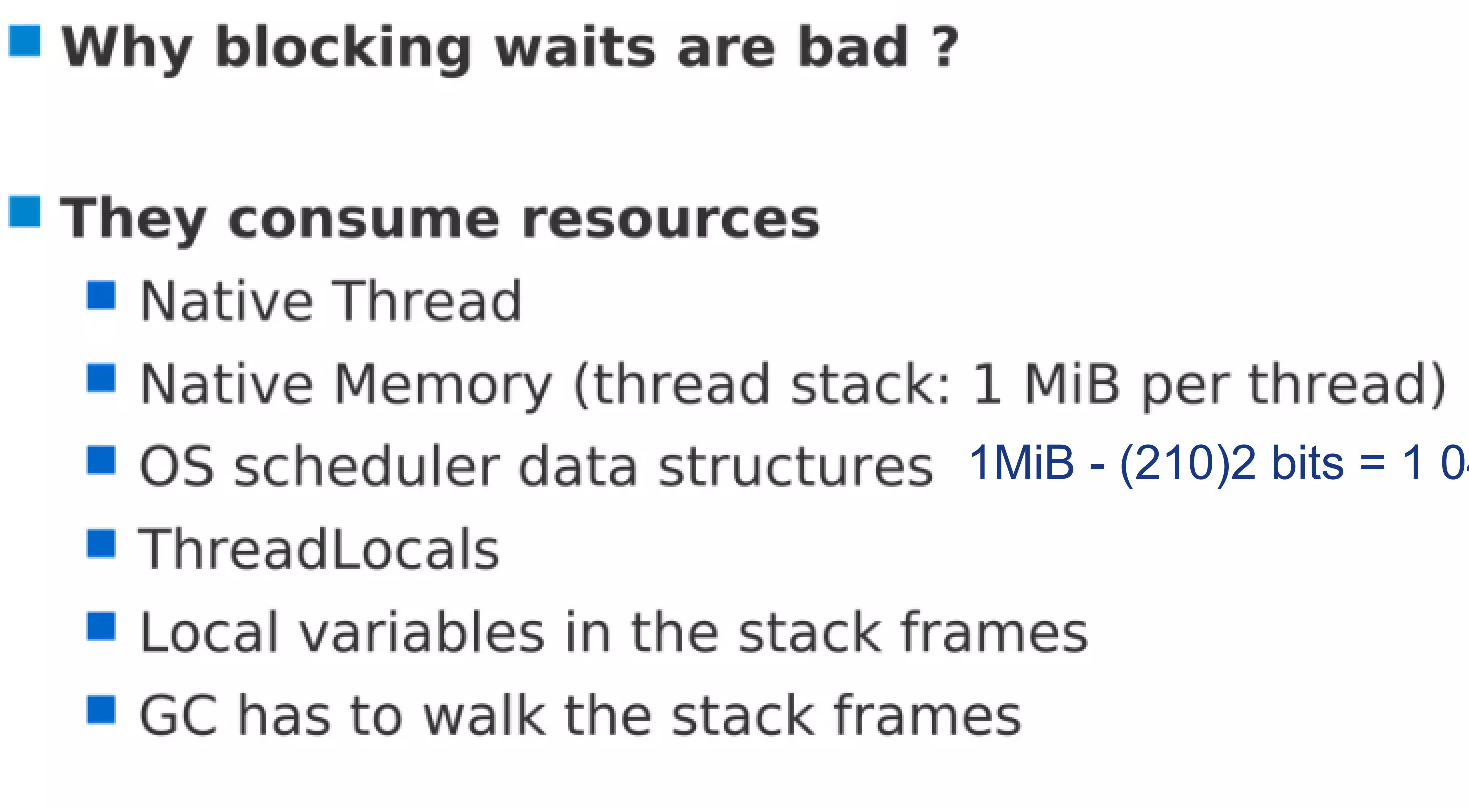
- **Asynchronous servlets** are a mechanism in Java Servlet technology that allows for non-blocking handling of requests, especially those involving long-running tasks or I/O operations.

- **Benefits :**
- Improved performance and scalability
- Non-blocking I/O operations
- Better resource utilization

Synchronous Servlets



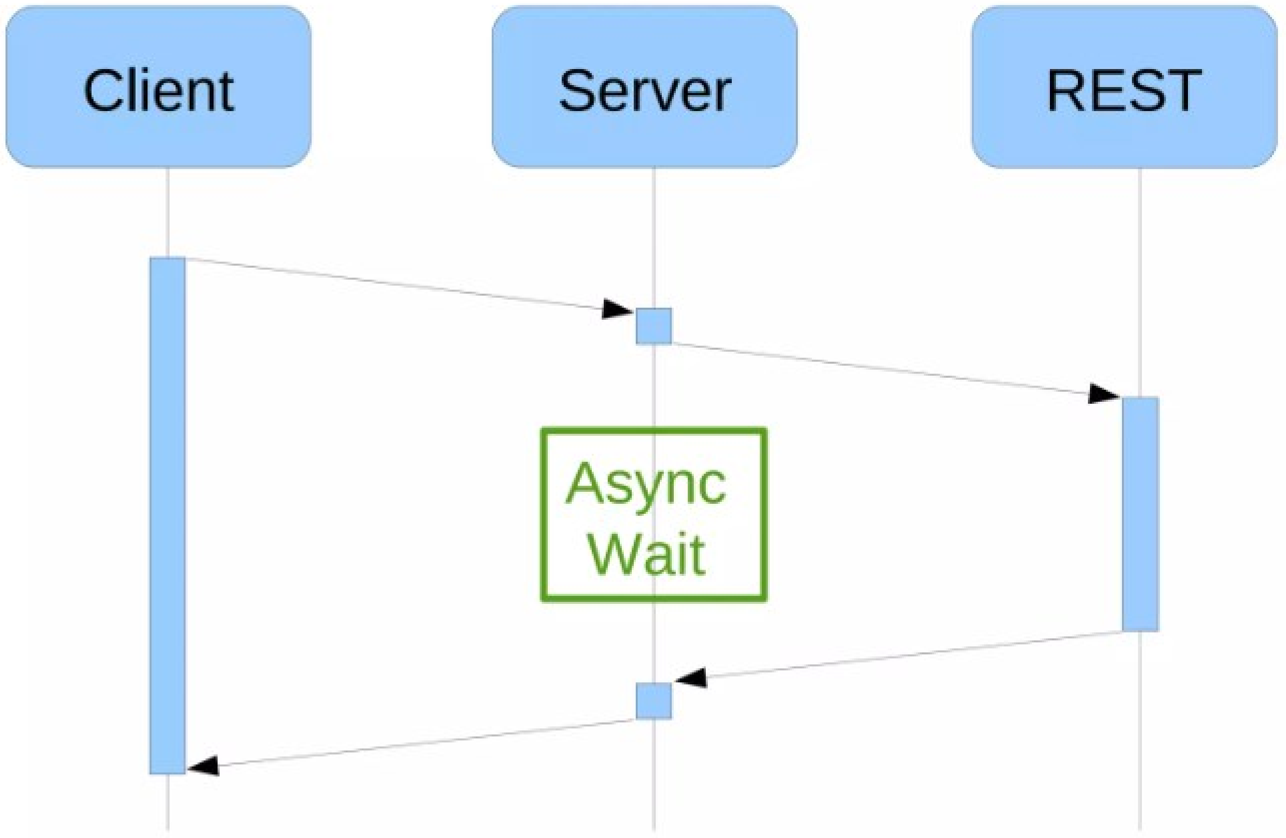
Drawbacks of Synchronous Servlets



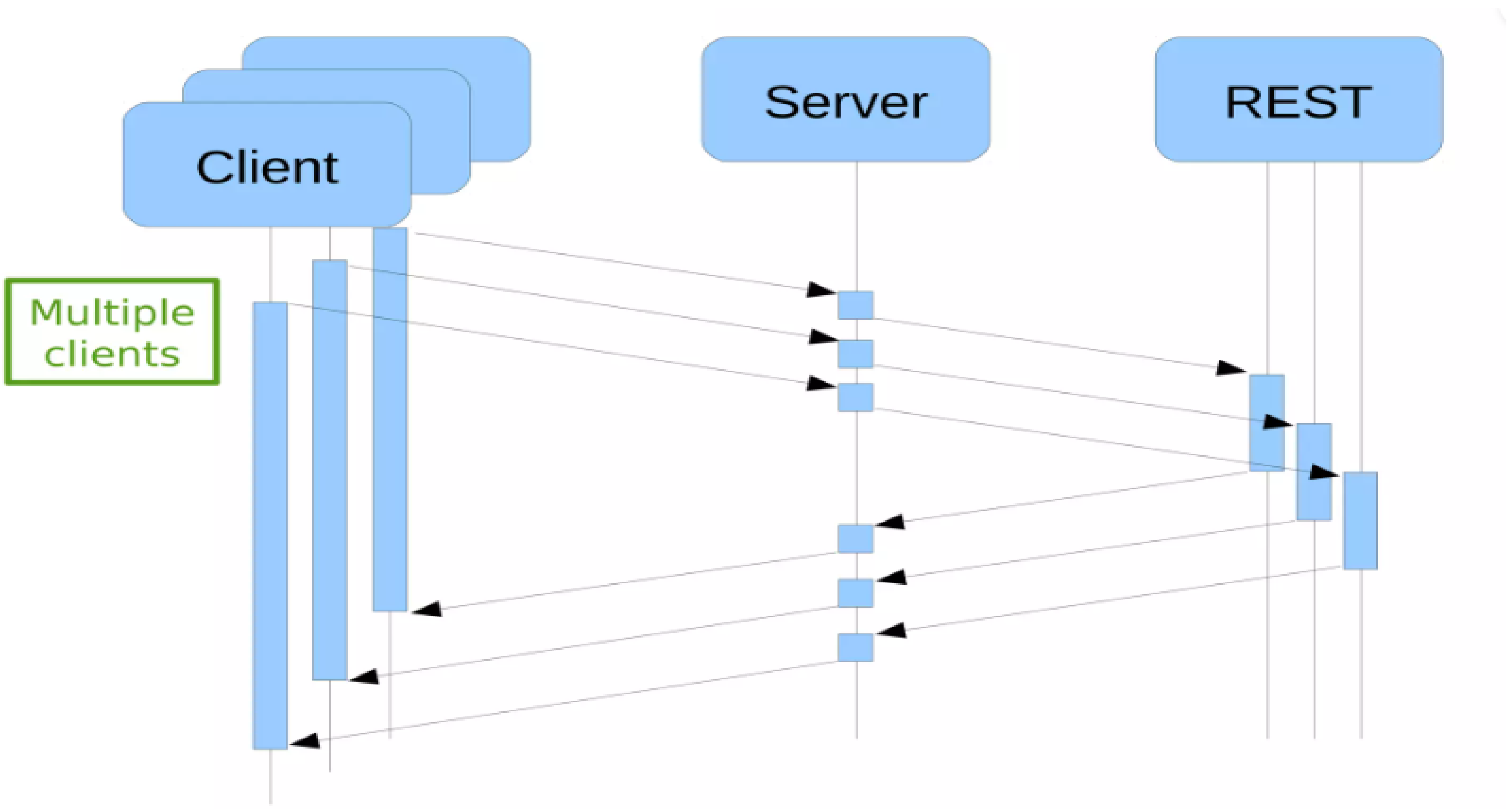
- **Why blocking waits are bad ?**
- **They consume resources**
  - Native Thread
  - Native Memory (thread stack: 1 MiB per thread)
  - OS scheduler data structures
  - ThreadLocals
  - Local variables in the stack frames
  - GC has to walk the stack frames

1MiB - (210)2 bits = 1 048 576

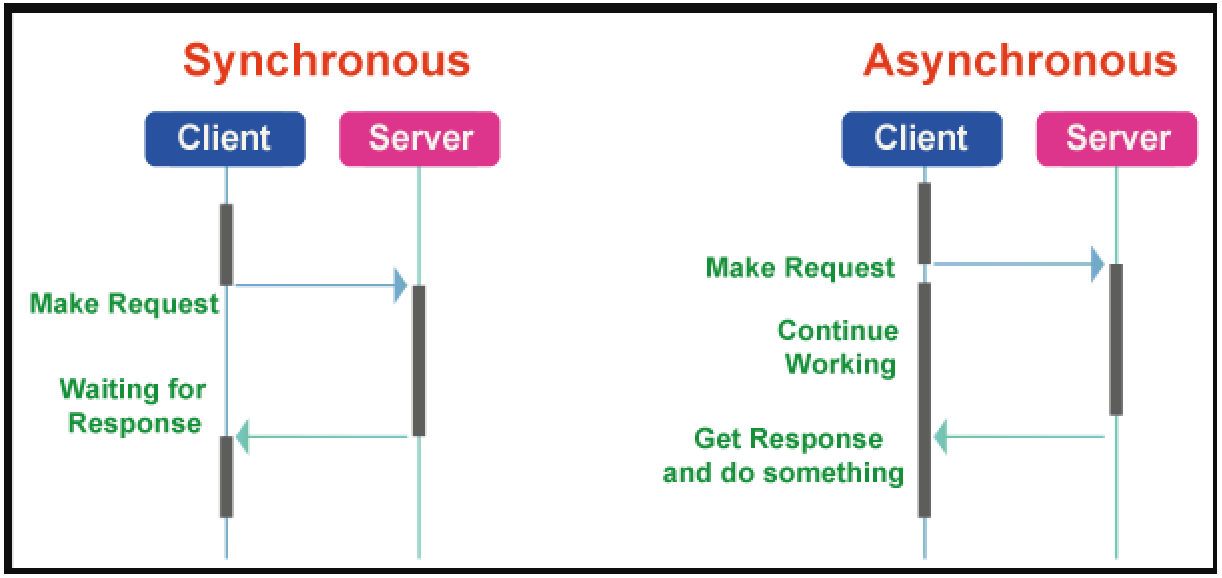
Asynchronous Servlets Architecture



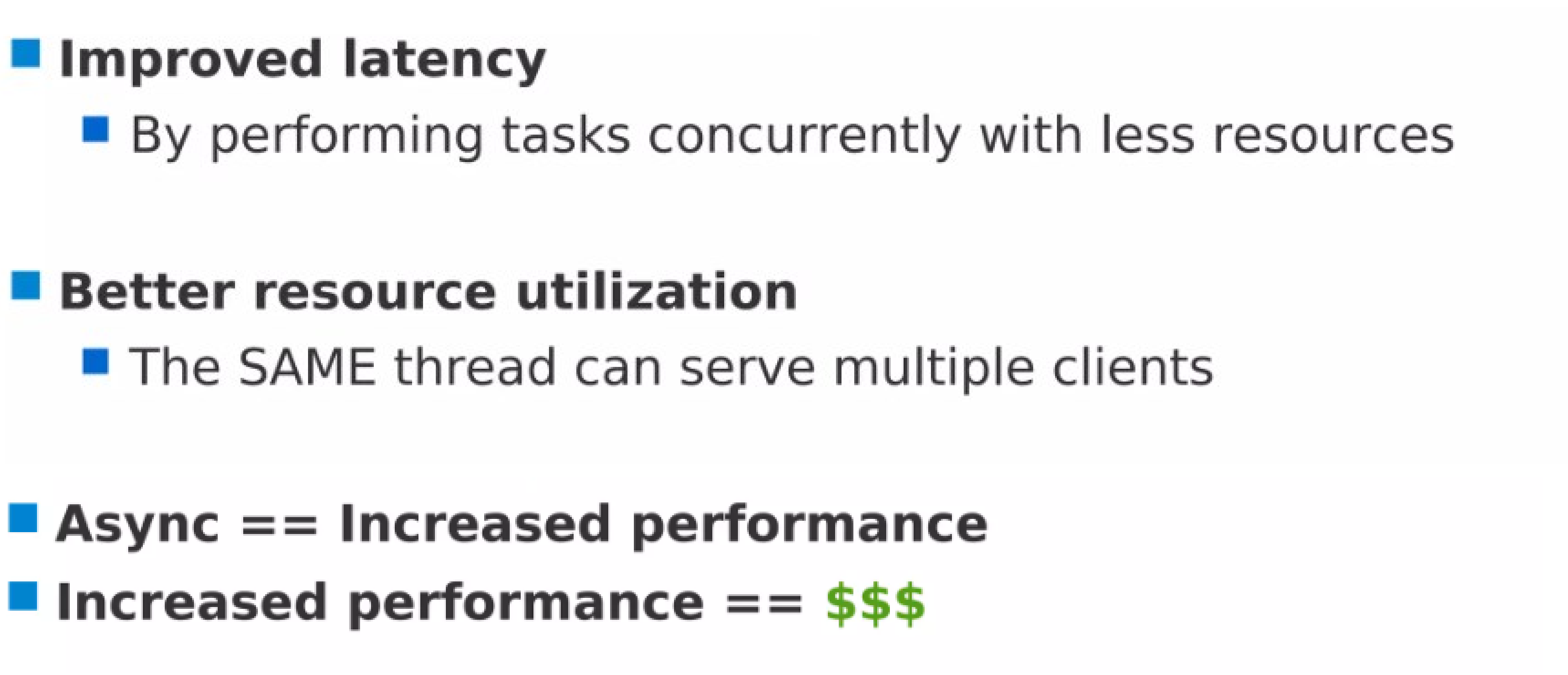
Asynchronous Servlets Architecture



Synch vs Asynch Request and Response



Async Servlet Benefits



- **Improved latency**
  - By performing tasks concurrently with less resources
- **Better resource utilization**
  - The SAME thread can serve multiple clients
- **Async == Increased performance**
- **Increased performance == $$$**

Async Servlet API …

AsyncContext is a crucial component in the realm of asynchronous servlet programming.

It represents the execution context for an asynchronous operation initiated on a ServletRequest.

Essentially, it's a bridge between the initial request handling thread and the background thread that will eventually process the request.

Async Servlet API

startAsync() - The startAsync() method is a crucial component that initiates the asynchronous processing of a request.

It's invoked on a ServletRequest object to create an AsyncContext instance

---

start() : The start() method of the AsyncContext interface is used to initiate the asynchronous processing of a request.

It takes a Runnable object as a parameter.

---

complete() :

The complete() method in AsyncContext is the final step in the asynchronous request processing lifecycle.

It signifies that the asynchronous operation has finished, and the response is ready to be sent to the client. **Demo

XMLHttpRequest Usage

The XMLHttpRequest object is primarily associated with client-side JavaScript for making asynchronous HTTP requests to a server.

Process :

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

Implementation :

- Creating XMLHttpRequest object

- Handling asynchronous requests

- Processing server responses (XML parsing)

** Demo

AJAX

- AJAX = Asynchronous JavaScript And XML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes.

Page reloading – Traditional and AJAX Way

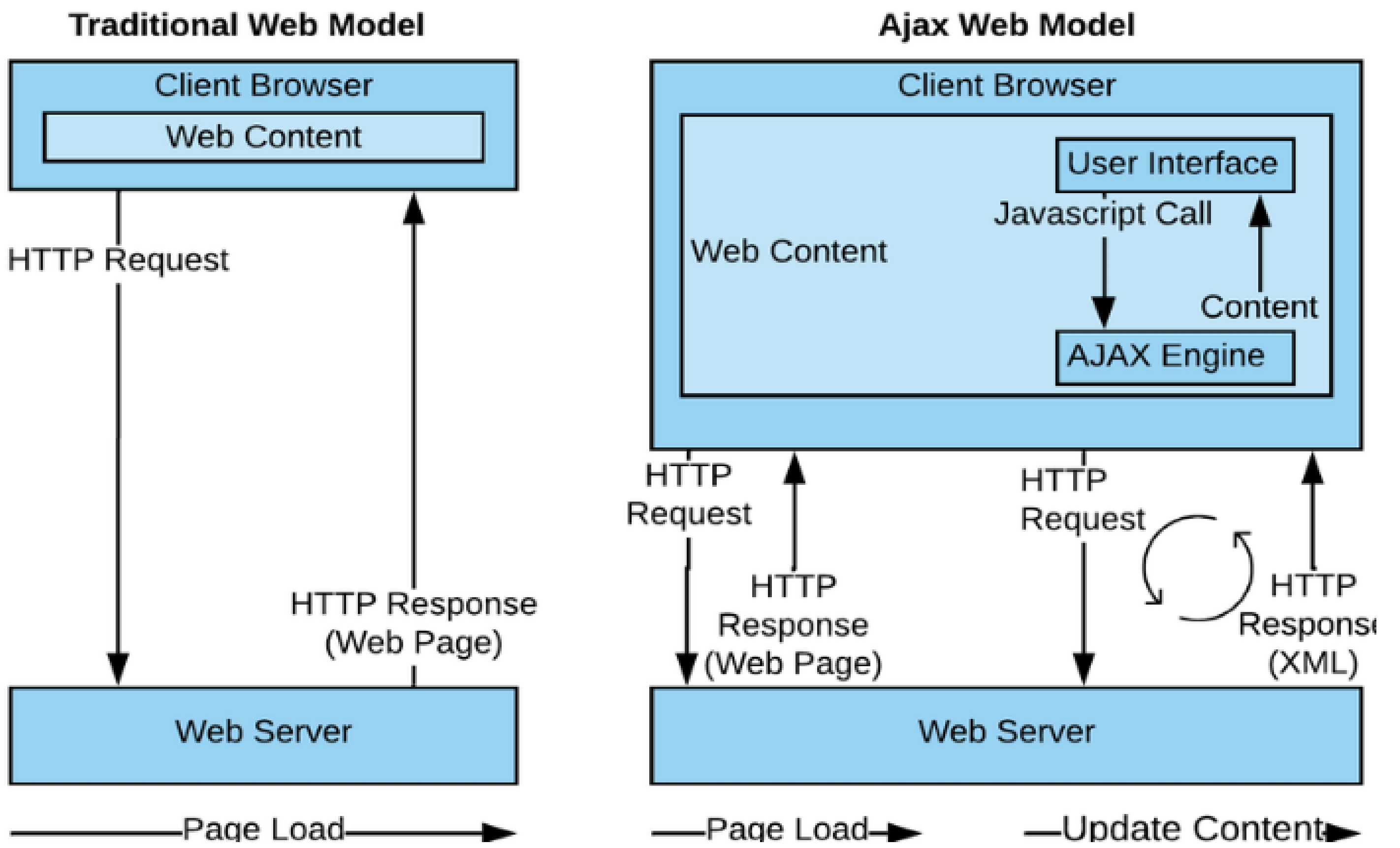
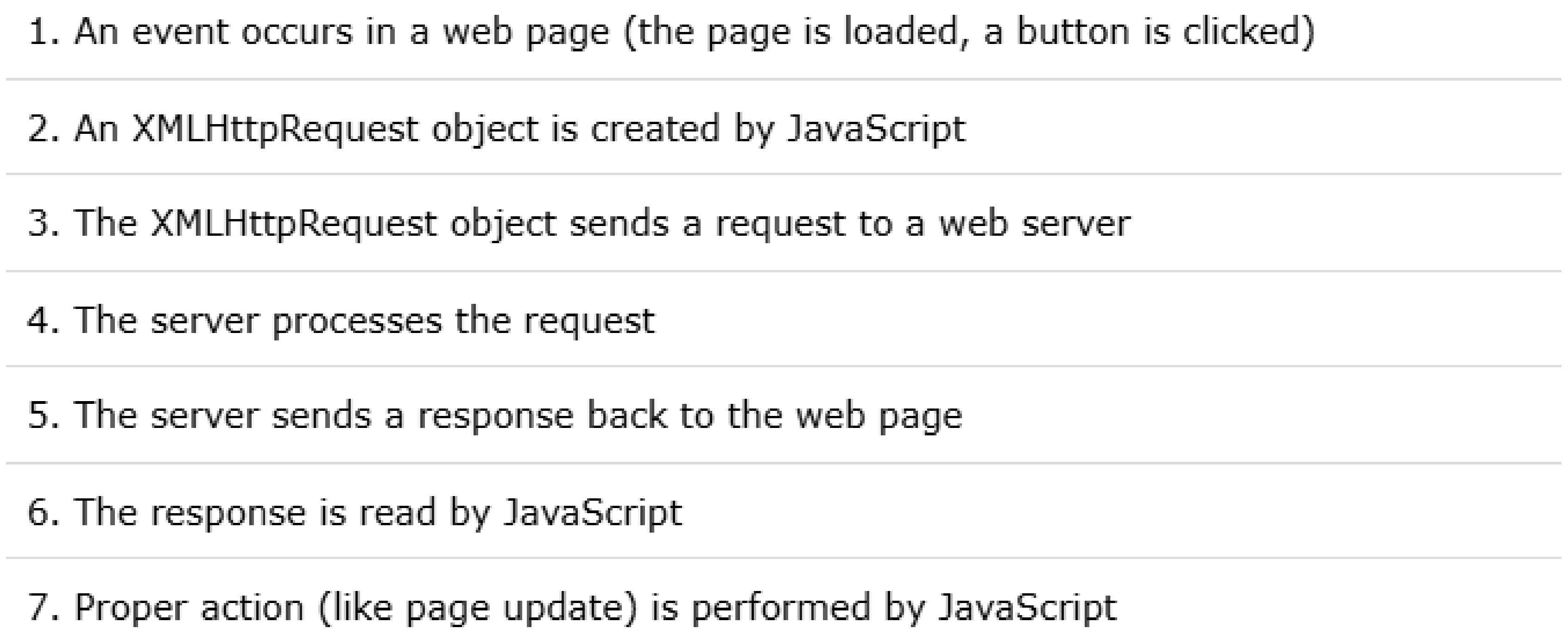


Illustration of Traditional Web Model vs AJAX Web Model. In traditional web models, an HTML request results in a full page refresh. In an AJAX Web model, the user requests a new content using XHR request and the respective contents/objects will be retrieved and displayed dynamically (an in-place update).

AJAX Flow



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

** Demo

Thank You