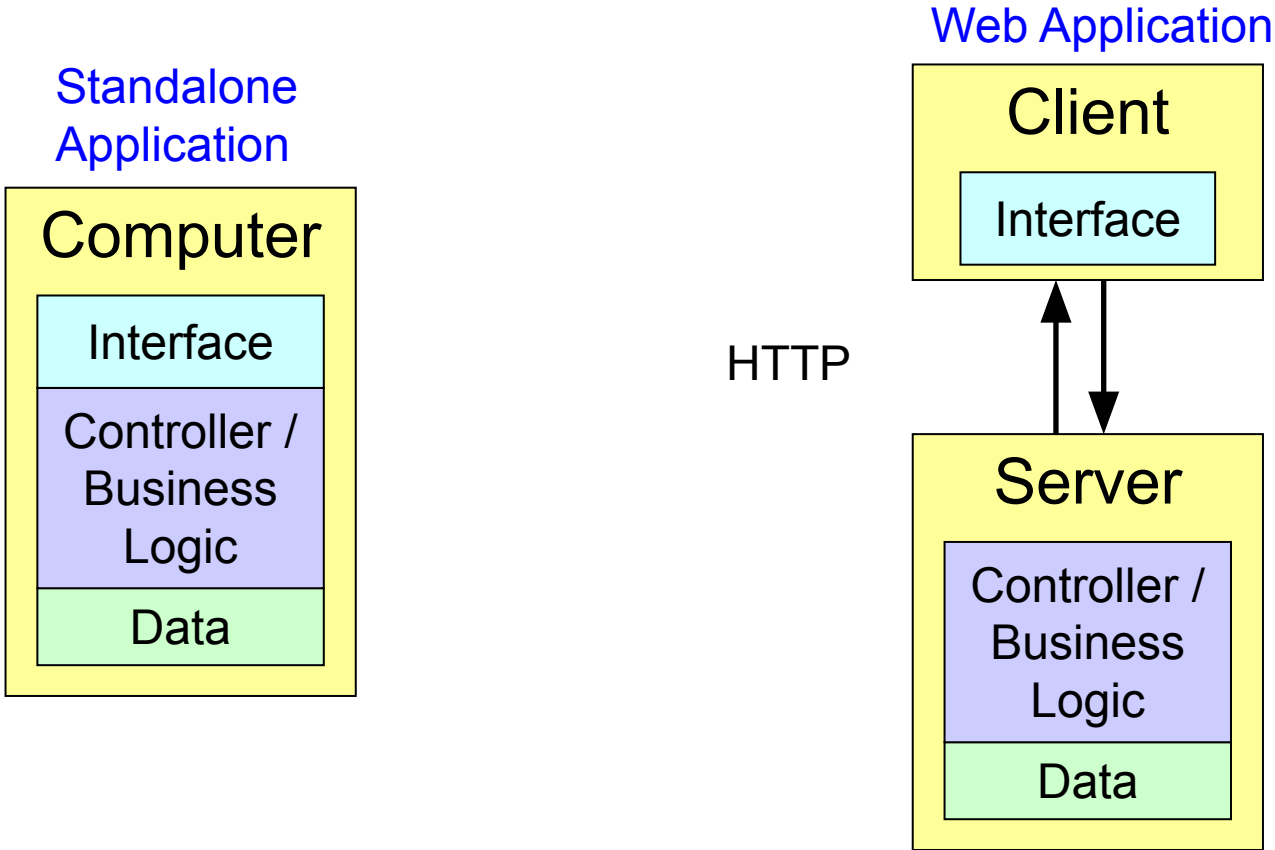
Rich Internet Application

Module 6

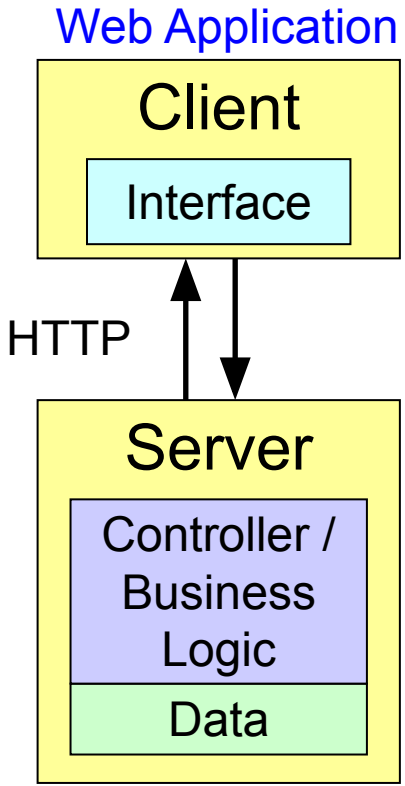
- Rich Internet applications (RIA) are Web-based applications that have some characteristics of graphical desktop applications.
- Built with powerful development tools, RIAs can run faster and be more engaging.
- They can offer users a better visual experience and more interactivity than traditional browser applications that use only HTML and HTTP.
- RIAs typically leverage advanced web technologies such as AJAX (Asynchronous JavaScript and XML), HTML5, CSS3, and JavaScript to provide features like interactivity, responsiveness, multimedia content, and offline capabilities.

- The term "rich Internet application" was introduced in a white paper of March 2002 by Macromedia (later merged into Adobe)
- Rich Internet Applications use a Rich Client deployment model rather than a thin-client-server model

How are web applications different from traditional desktop applications?



Web Applications



Characteristics of Rich Internet Applications include:

- **Interactivity**: RIAs allow users to interact with the application in real-time without needing to reload the entire webpage. This enables features like drag-and-drop, dynamic content updates, and form validation without page refreshes.
- **Rich Media**: RIAs can incorporate rich media elements such as audio, video, animations, and graphics to enhance the user experience and engagement.
- **Asynchronous Data Loading:** RIAs use techniques like AJAX to fetch data from the server asynchronously, enabling faster response times and a smoother user experience by updating only the necessary parts of the page.
- **Cross-Platform Compatibility:** RIAs are designed to run on multiple platforms and devices, including desktop computers, laptops, tablets, and smartphones, without requiring platform-specific development.

- **Responsive Design**: RIAs can adapt to different screen sizes and orientations, providing a consistent user experience across devices through responsive design principles.
- **Offline Capabilities**: Some RIAs can continue to function even when the user is offline by leveraging technologies like local storage and caching. This allows users to access certain features and content without an active internet connection.
- **Client-Side Processing**: RIAs offload some processing tasks from the server to the client's browser, improving performance and reducing server load. This is often achieved using client-side scripting languages like JavaScript.
- **Integration with Web Services**: RIAs seamlessly integrate with web services and APIs to access and manipulate data from external sources, enabling features like social media integration, real-time updates, and third-party content embedding.

- **Security**: RIAs incorporate security features to protect user data and prevent unauthorized access, including encryption, secure communication protocols, input validation, and authentication mechanisms.

At the client's end

- A controlled and standardized environment for running the client portion of a web application
- Support richer UI and interaction

For the developers

- Improving productivity
- No need to worry about browser's incompatibilities
- Build UI the same way UIs are built for stand-alone applications (familiarity)
- Build UI using visual tools
- Ability to use other languages to write scripts to interact with UI components

PROs

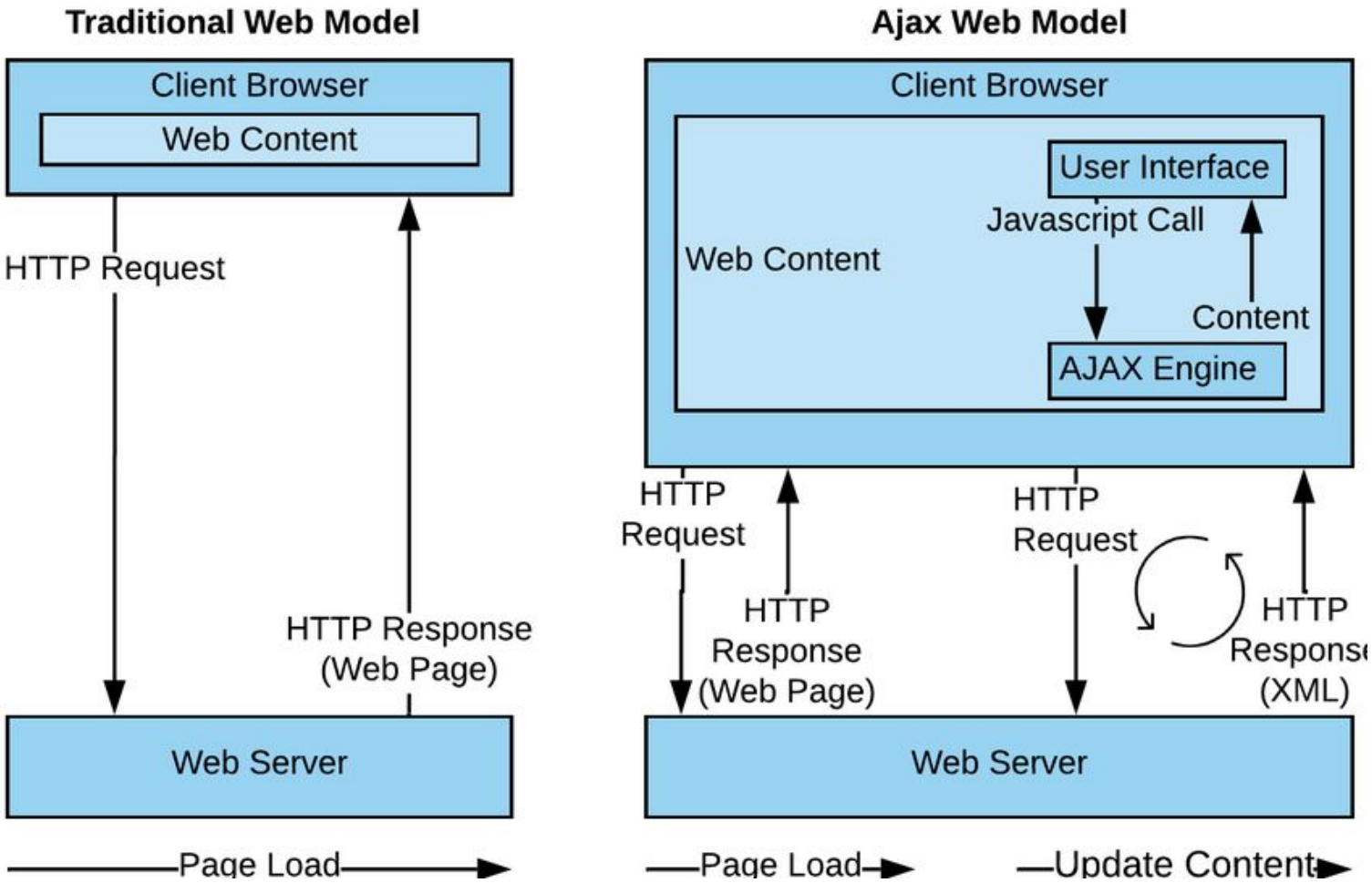
- Installation is not required
- Easy to upgrade
- Easily made available over internet/intranet
- Richer UI
- More responsive UI
- Client/Server Balance
- Asynchronous communication
- Network efficiency

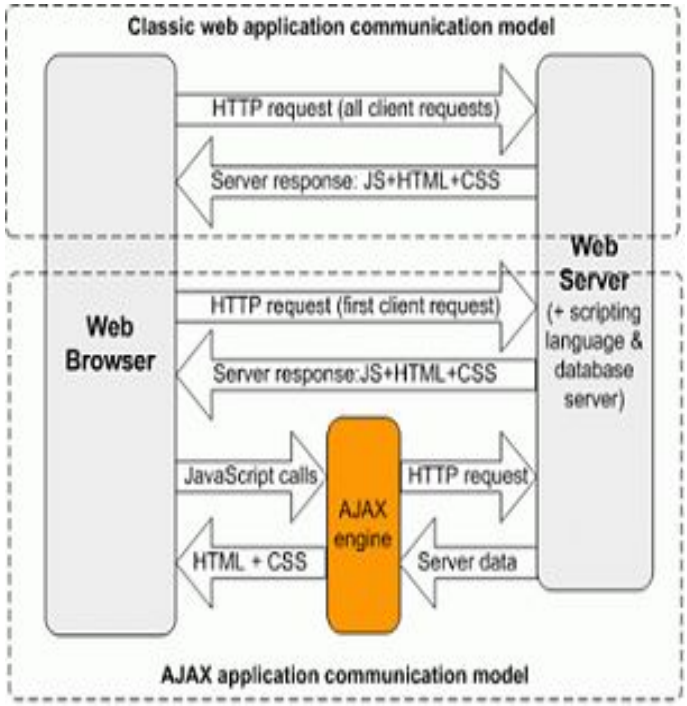
CONs

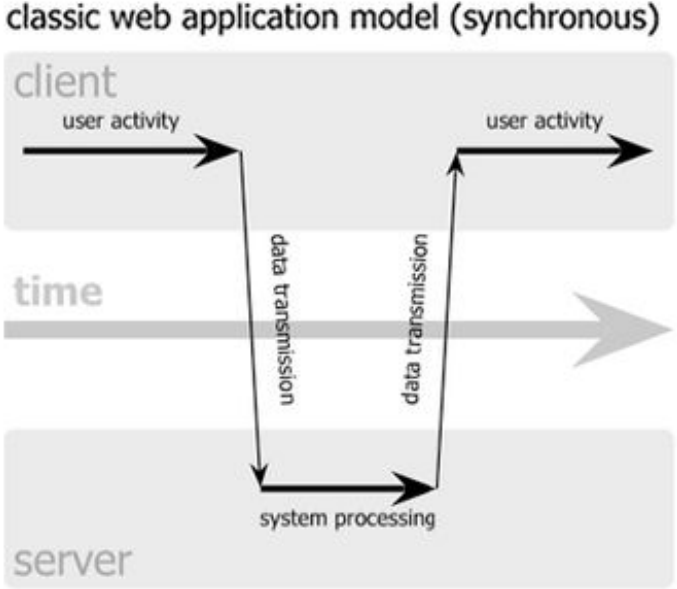
- Loss of visibility to search engines
- Software development complications (What to cache or not to cache at client's computer?)
- RIA architecture breaks the Web page paradigm

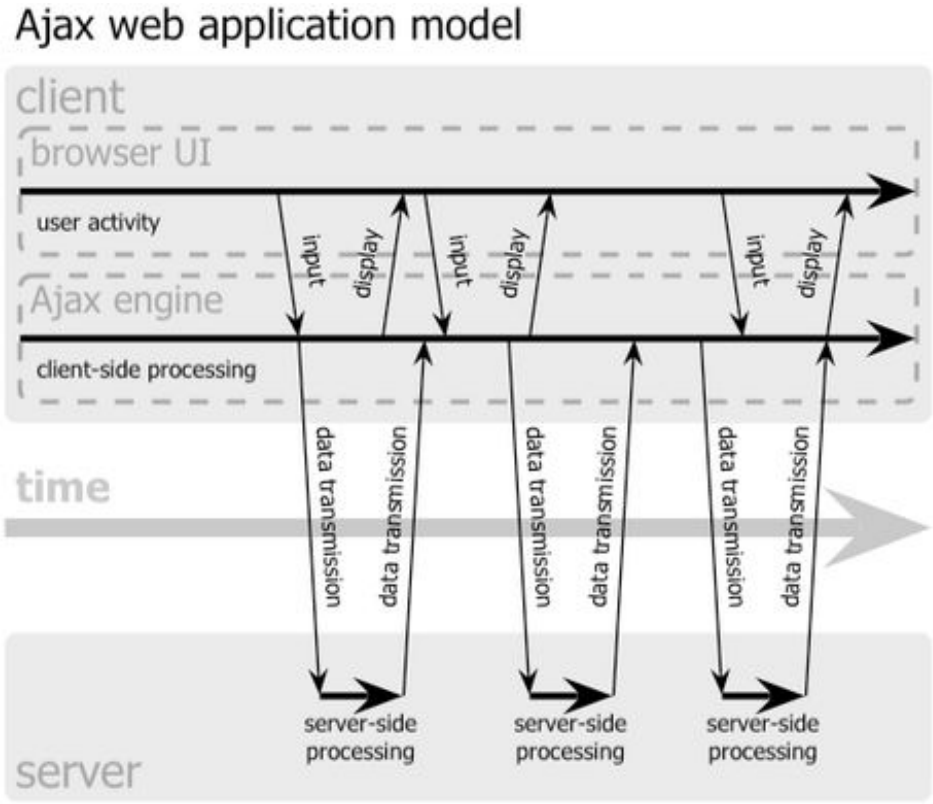
AJAX

- stands for Asynchronous Javascript And XML
- Allows incremental update of Web pages.
- Built using standard web technologies - HTTP, (X)HTML, CSS, JavaScript, Document Object Model (DOM), XML
- created by Jesse James Garrett in 2005.
- Ajax isn’t a single technology.
- It’s really several independent technologies coming together in new ways.
- Ajax incorporates:
    - standards based presentation using XHTML and CSS
    - dynamic display and interaction using the Document Object Model
    - data interchange and manipulation using XML and XSLT
    - asynchronous data retrieval using XMLHttpRequest
    - JavaScript binding everything together


Traditional Web Model
Client Browser
Web Content
HTTP Request
HTTP Response
(Web Page)
Web Server
Page Load
Ajax Web Model
Client Browser
User Interface
Javascript Call
Web Content
Content
AJAX Engine
HTTP
Request
HTTP
Response
(Web Page)
HTTP
Request
HTTP
Response
(XML)
Web Server
Page Load
Update Content


Classic web application communication model
HTTP request (all client requests)
Server response: JS+HTML+CSS
Web Server
(+ scripting language & database server)
Web Browser
HTTP request (first client request)
Server response:JS+HTML+CSS
JavaScript calls
AJAX engine
HTTP request
HTML + CSS
Server data
AJAX application communication model


classic web application model (synchronous)
client
user activity
user activity
data transmission
data transmission
time
system processing
server


Ajax web application model
client
browser UI
user activity
input
display
input
display
input
display
Ajax engine
client-side processing
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
time
server-side processing
server-side processing
server-side processing
server

How does AJAX work?

1. A JavaScript creates an XMLHttpRequest object, initializes it with relevant information as necessary, and sends it to the server. The script (or web page) can continue after sending
   it to the server.
2. The server responds by sending the contents of a file or the output of a server side program (for example, in PHP).
3. When the response arrives from the server, a JavaScript function is triggered to act on the data supplied by the server.
4. This JavaScript response function typically refreshes the display using the DOM, avoiding the requirement to reload or refresh the entire page.

Property	Description
onreadystatechange	Event handler for an event that fires at every state change
readyState	Object status integer: 0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
responseText	String version of data returned from server process
responseXML	DOM-compatible document object of data returned from server process
status	Numeric code returned by server, such as 404 for "Not Found" or 200 for "OK"
statusText	String message accompanying the status code

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

What are the Issues with AJAX?

- User does not know updates will occur.
- User does not notice an update.
- User can not find the updated information.
- Unexpected changes in focus.
- Loss of Back button functionality*.
- URIs can not be bookmarked*.