CptS 322- Programming Language Design

Web Development Overview

Instructor: Sakire Arslan Ay

Fall 2023

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

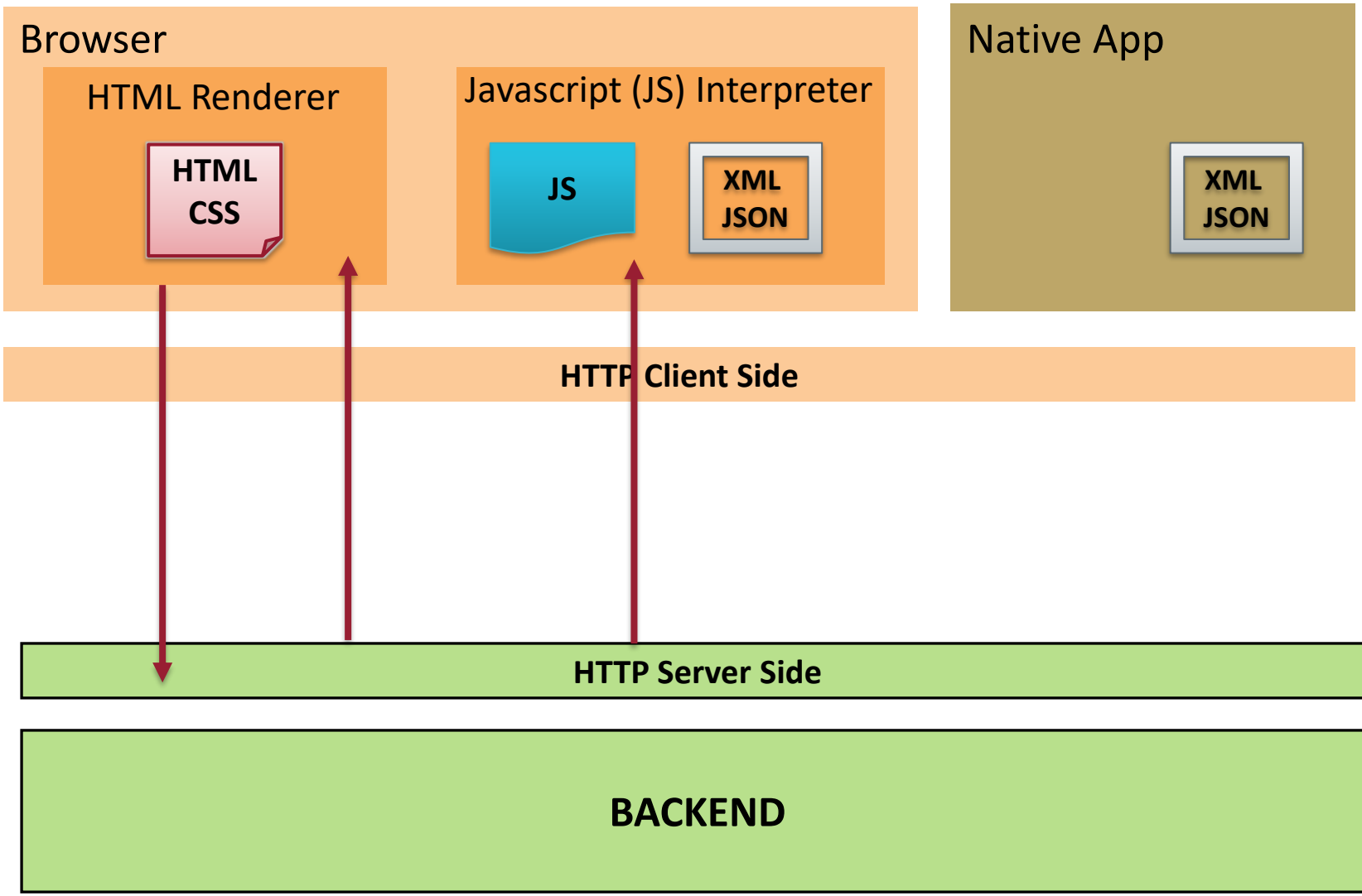
The Web

- Essentially a client-server architecture
- Clients (front-ends) are typically browsers, but can be native apps
  - Take the inputs from the user
  - Make requests for data to the server
  - Display the data

The Web

- Servers (back-ends) contain the persistent data, and most of the business logic
  - Servers answer client requests
  - Servers contain many pieces:
    - HTTP server
    - Business logic
    - Databases
    - Caches
    - Load balancers

Web App Architecture



Web App Architecture

- **Browser**
  - The user interface (displays data, takes user inputs)
  - Gets and sends data to/from backend
  - Technologies: HTML and CSS and JavaScript
- **Network**
  - The transport (HTTP, URI, REST)
- **Backend**
  - Stores and processes data
  - The critical processing logic
  - The heavy processing, or shared-data processing
  - Technologies: DB, frameworks (Django, RoR, Node.js, Flask) + more

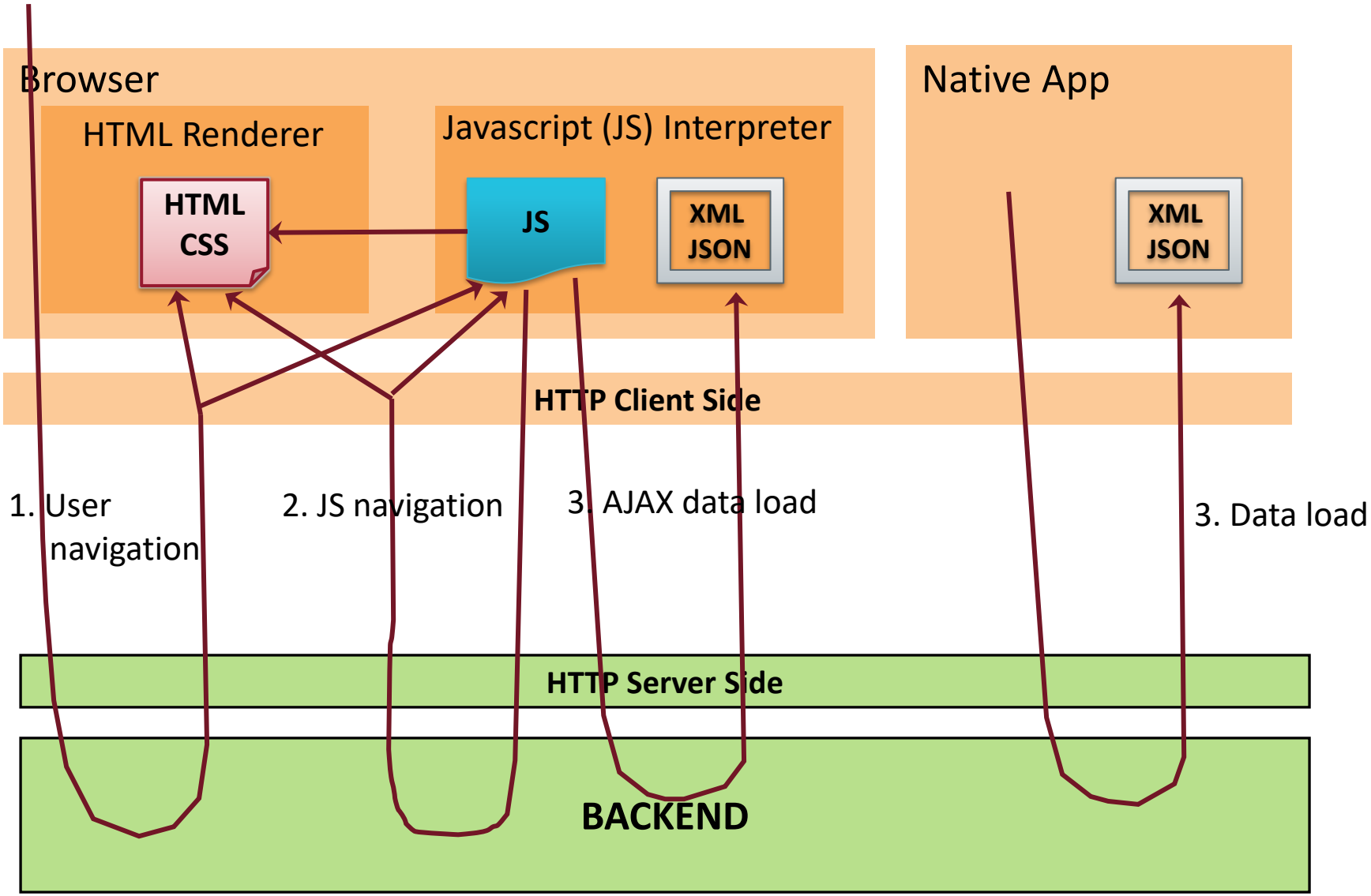
Evolution of Web Apps

- First web sites were static pages (1990)
- Dynamic pages (CGI, servlets) (1993)
- Then e-commerce (1995)
- Then services (email, search, apps)
  - Software-as-a-Service (SaaS)

HTML and CSS and Javascript

- HTML (hyper-text markup language)
  - A declarative language to describe content
- CSS (cascading style sheets)
  - A declarative language to describe the presentation of content
- Javascript
  - An imperative language to describe functionality
  - Changes content and presentation and issues data requests to backend

Web App Architecture



HTTP – Web Transfer Protocol

- A protocol specifying requests (from client) and responses (from server)
  - Built on top of TCP (reliable stream of bytes)
- By default uses port 80 on the sever
  - For testing you can use other ports
  - Port 80 is the most reliable through proxies and firewalls
- There is also HTTPS (default port 443)
  - Ensures server authenticity; requires paid server certificate.

HTTP – Web Transport Protocol

- **Example request:**

```
GET /index.html HTTP/1.0
User-agent: Safari 4.0
Referrer: http://test.com/testpage.html
If-modified-since: Fri, 31 Dec 2016 23:59:58 GMT
[blank line]
```

Request method type:
- **GET** – retrieve a resource (no side-effects)
- **POST** – send data in request; response may have side-effects
- **PUT** , **DELETE**

Describes OS/browser

Caching hints

- **Example response:**

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-type: text/html
Content-length: 19
[blank line]
<html> Hello</html>
```

Status code:
200 – Ok
3xx – alternate (redirect, cached)
4xx, 5xx – Errors

Tells client how to interpret the content

Headers end with blank line. The data follows

URL Format

- URL = Uniform Resource Locator
  - Identifies files/resources available from web servers
- Example URL:
    - http://server.com:8888/path/file?p1=v1&p2=v2#x
  - Protocol (scheme): http, https
  - Host: server.com
  - Port: 8888 (optional, default 80)
  - Path: /path/file
  - Query: p1=v1&p2=v2 (optional)
  - Anchor: #x (optional, NOT sent to the server)

REST (Representational State Transfer)

- Principles for designing URL-based addressing scheme for resources
  - Examples:
    - Collection URL: /users
    - Element URL: /users/<userid>/action
      - In the above path <userid> will be substituted by the “id” of the user
- What HTTP method to use:
  - GET vs. POST vs. PUT vs. DELETE
- Convention:
  - GET requests may be cached, others are not cached

REST (Representational State Transfer)

Resource	GET	PUT	POST	DELETE
	No side-effects	Idempotent (multiple times - ok)		Idempotent (multiple times - ok)
Collection URL: **/users**	List of collection elements	Replace the entire collection	Create element. Return element.	Delete the entire collection
Element URL: **/users/15**	Retrieve the data for one element	Update the element in the collection	Not generally used.	Delete the element from the collection
Element URL: **/users/15/action**	Retrieve some data for one element		Perform an action, return result.	

REST (Representational State Transfer)

- How do we return error information?
  - Use status codes
    - E.g., 200 Ok, 404 not found, 403 denied, 500 bug, etc.
    - Limited choice of status codes
    - Web server and proxies may interpret the codes

Encoding Data: XML and JSON

- How do you encode data in requests/responses?
- XML was first, lots of tool support

```
<book>
 <title>How to build a web app</title>
 <price>15.00</price>
</book>
```

- JSON is gaining popularity: simple, short, readable

```
{
 "title" : "How to build a web app",
 "price" : 15.0 }
}
```

- There are binary proprietary formats (Protobuf, Thrift)
  - Compatibility issues (browsers, applications, debuggers)

Basic Flask App Architecture