Web Software Model

CS 4640
Programming Languages for Web Applications

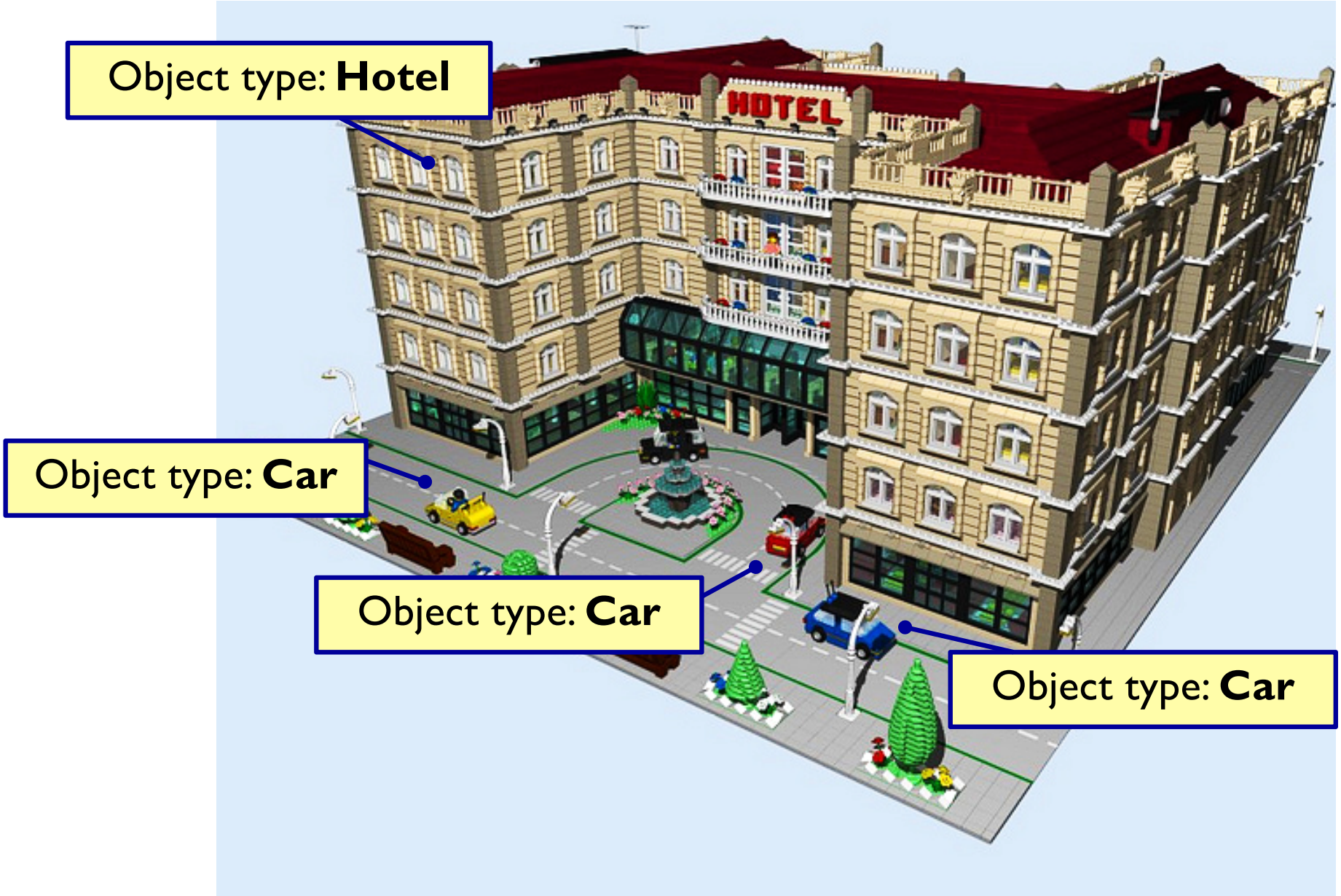
[Robert W. Sebesta, "Programming the World Wide Web"]

Web Applications

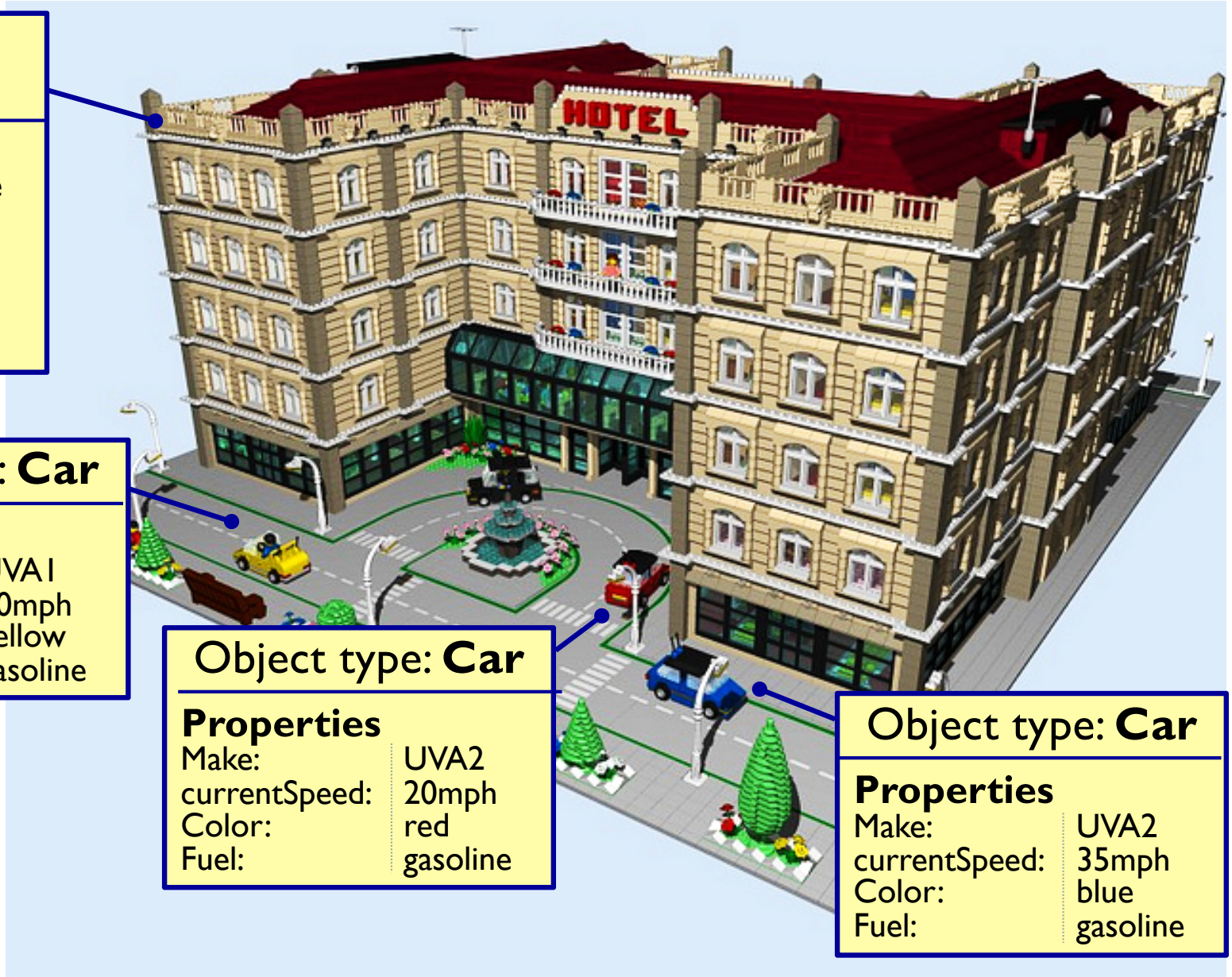
- User interactive software programs, deployed on a web server, accessed via a web browser
  - Browser features may affect the program's execution flow
- Use enabling technologies to
  - Make web site contents dynamic
  - Allow users of the system to implement business logic on the server
- Let users affect state on the server
- Constructed from diverse, distributed, and dynamically generated web components
  - Web components are software modules that implement different parts of the application's functionality

An enabling technology makes web pages interactive and responsive to user input

How do Web Apps fit in with the World Around Them?



Objects and Properties



Object type: Hotel

Properties

Name:	Awesome
Rating:	5
Rooms:	70
Bookings:	56
Pool:	true
Gym:	true

Object type: Car

Properties

Make:	UVA1
currentSpeed:	30mph
Color:	yellow
Fuel:	gasoline

Object type: Car

Properties

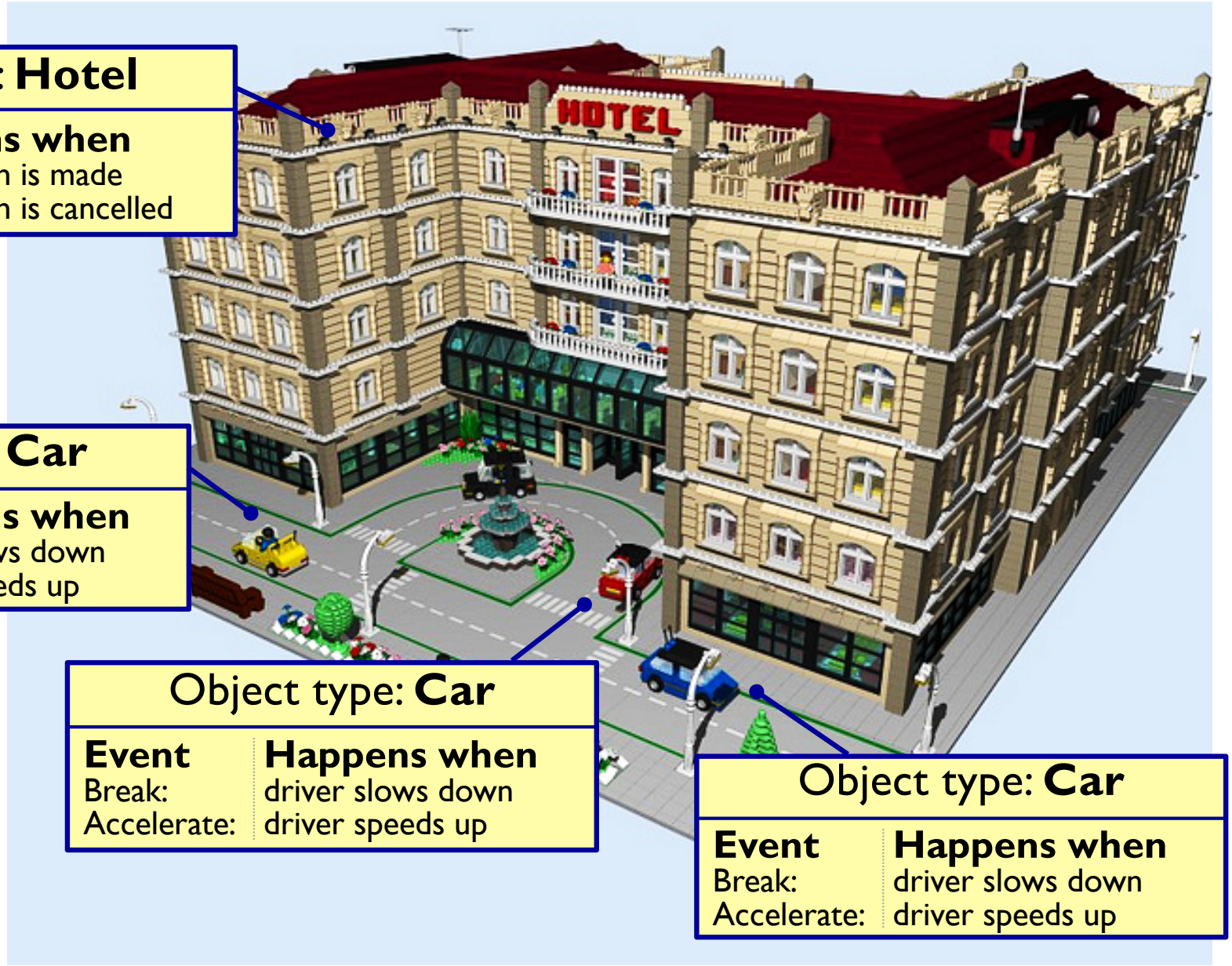
Make:	UVA2
currentSpeed:	20mph
Color:	red
Fuel:	gasoline

Object type: Car

Properties

Make:	UVA2
currentSpeed:	35mph
Color:	blue
Fuel:	gasoline

Objects and Events



Object type: **Hotel**	
Event	**Happens when**
Reserve	reservation is made
Cancel	reservation is cancelled

Object type: **Car**	
Event	**Happens when**
Break:	driver slows down
Accelerate:	driver speeds up

Object type: **Car**	
Event	**Happens when**
Break:	driver slows down
Accelerate:	driver speeds up

Object type: **Car**	
Event	**Happens when**
Break:	driver slows down
Accelerate:	driver speeds up

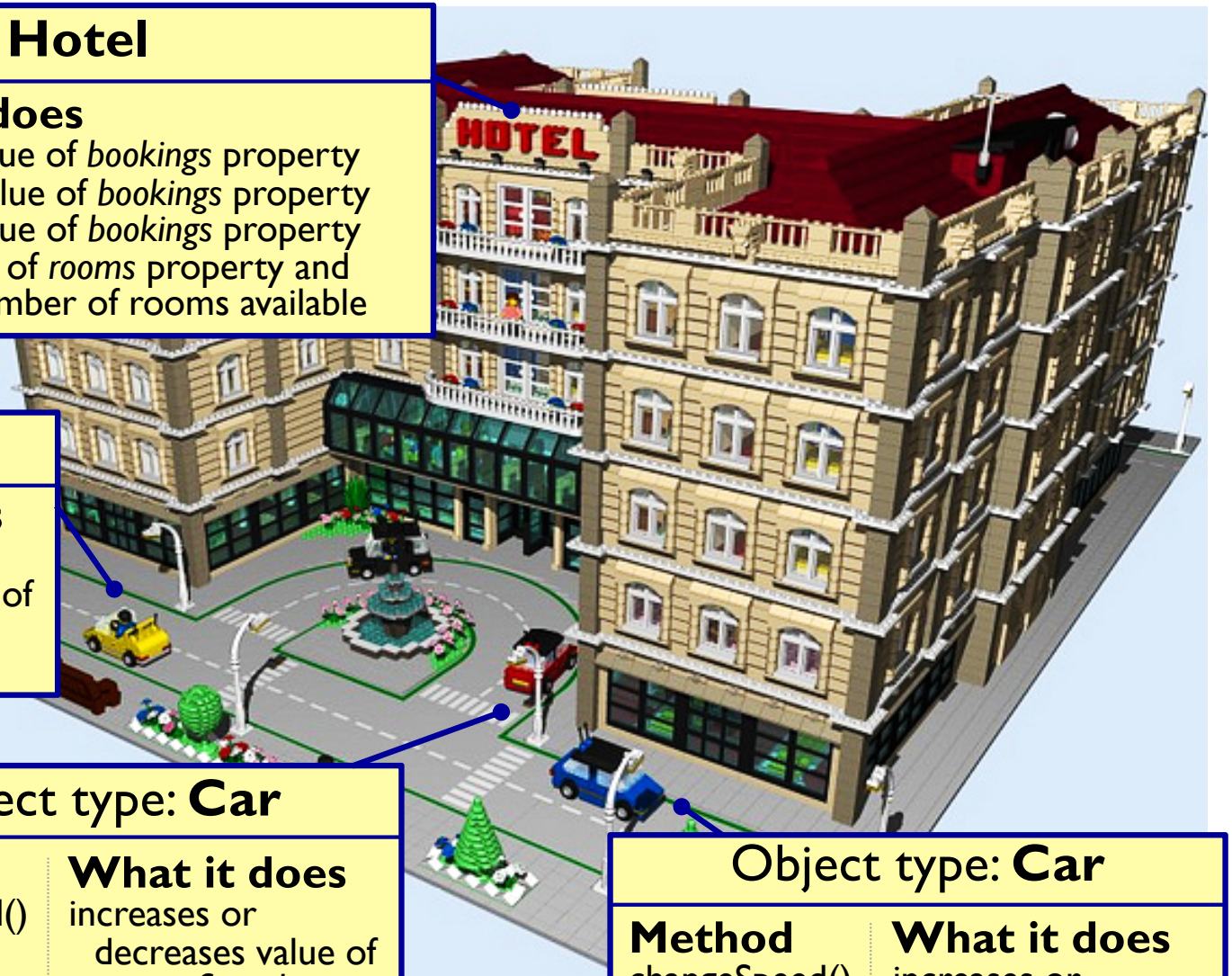
Objects and Methods

Object type: **Hotel**	
Method	**What it does**
makeReservation()	increases value of *bookings* property
cancelReservation()	decreases value of *bookings* property
checkAvailability()	subtracts value of *bookings* property from value of *rooms* property and returns number of rooms available

Object type: **Car**	
Method	**What it does**
changeSpeed()	increases or decreases value of *currentSpeed* property

Object type: **Car**	
Method	**What it does**
changeSpeed()	increases or decreases value of *currentSpeed* property

Object type: **Car**	
Method	**What it does**
changeSpeed()	increases or decreases value of *currentSpeed* property



All Together

Object type: **Hotel**

Event	Happens when	Method called
Reserve	reservation is made	makeReservation()
Cancel	reservation is cancelled	cancelReservation()

Method	What it does
makeReservation()	increases value of *bookings* property
cancelReservation()	decreases value of *bookings* property
checkAvailability()	subtracts value of *bookings* property from value of *rooms* property and returns number of rooms available

Properties

Name:	Awesome
Rating:	5
Rooms:	70
Bookings:	56
Pool:	true
Gym:	true

Object type: **Car**

Event	Happens when	Method called
Break	driver slows down	changeSpeed()
Accelerate	driver speeds up	changeSpeed()

Method	What it does
changeSpeed()	increases or decreases value of *currentSpeed* property

Properties

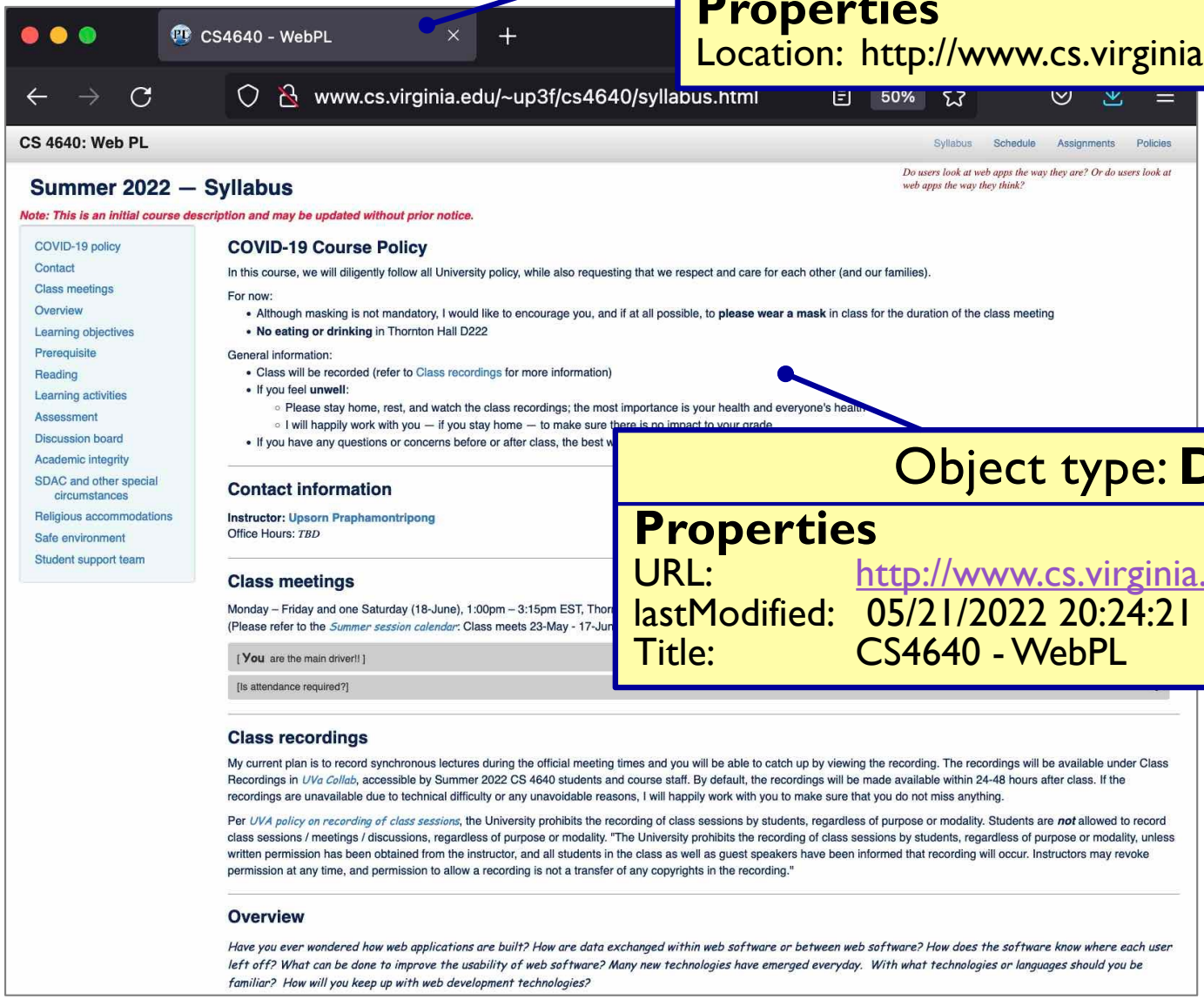
Make:	UVA1
currentSpeed:	30
Color:	yellow
Fuel:	gasoline

Let's Try

Activity:

"Thing" in the World and "Object" in Web App

Web Browsers and Objects



Object type: **Window**

Properties

Location: http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html

Object type: **Document**

Properties

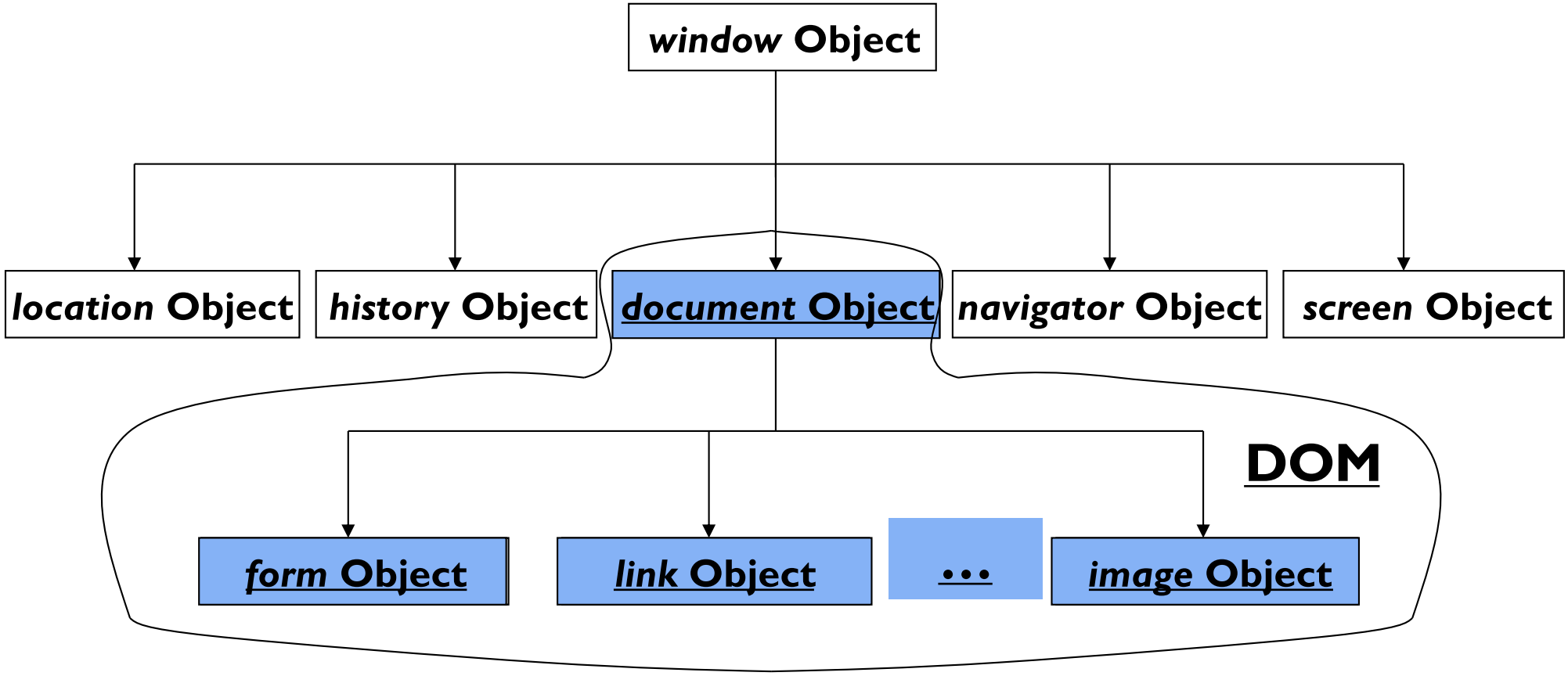
URL: http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html

lastModified: 05/21/2022 20:24:21

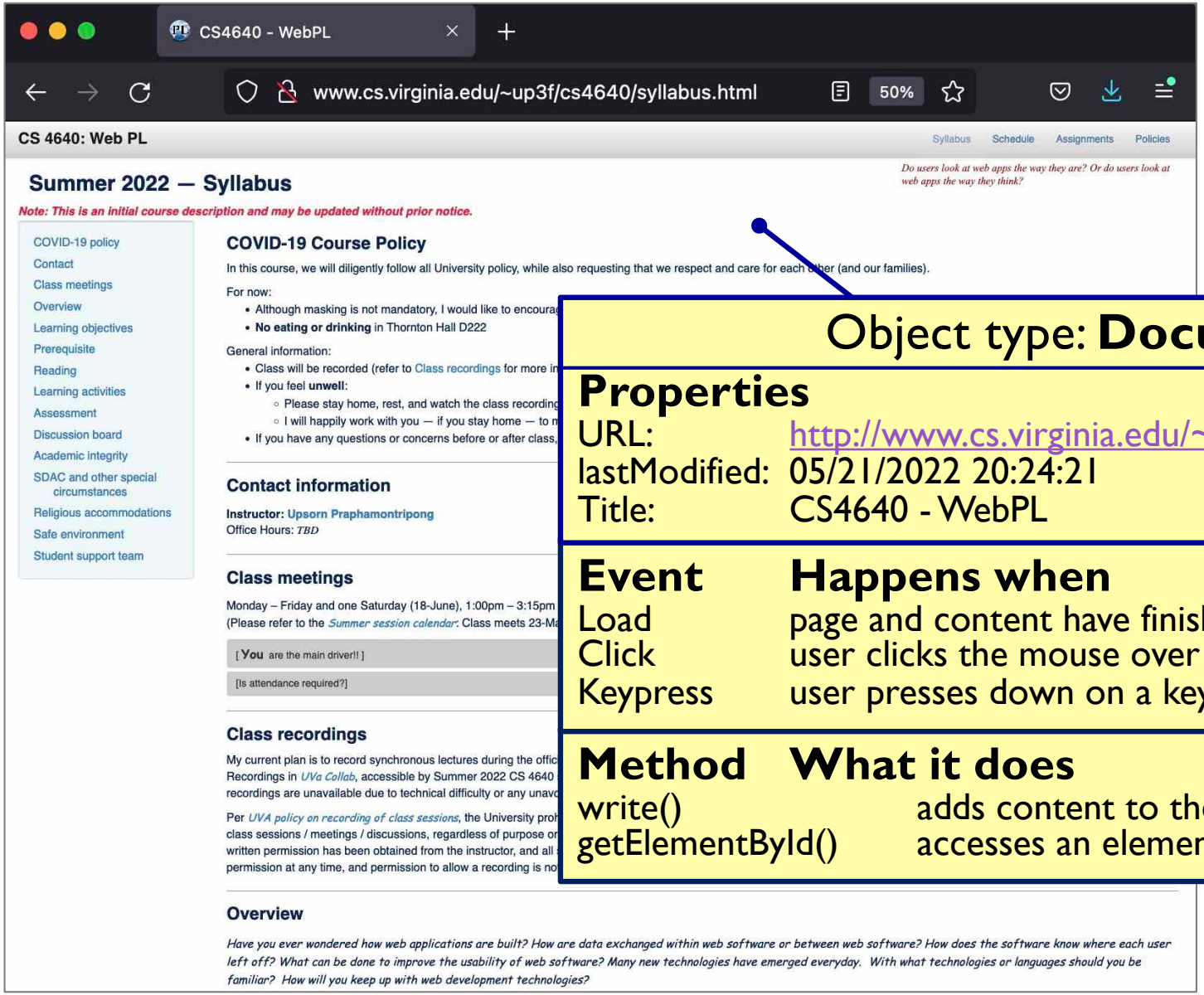
Title: CS4640 - WebPL

BOM: Browser Object Model

BOM – collection of objects that the browser makes available to us for use with JavaScript



DOM: Document Object Model



Object type: **Document**

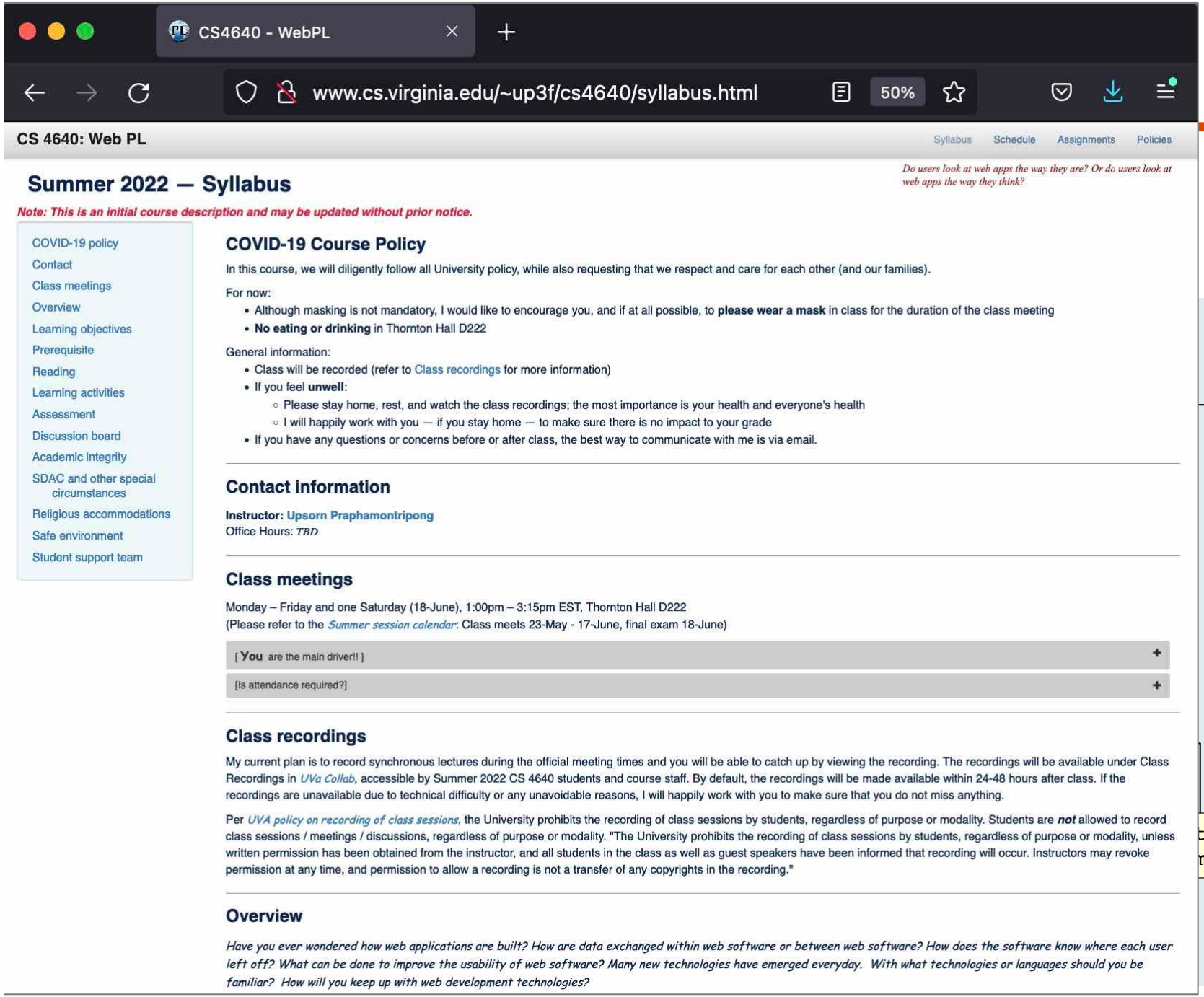
Properties

URL:	http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html
lastModified:	05/21/2022 20:24:21
Title:	CS4640 - WebPL

Event	**Happens when**
Load	page and content have finished loading
Click	user clicks the mouse over the page
Keypress	user presses down on a key

Method	**What it does**
write()	adds content to the document
getElementById()	accesses an element of a given *id* attribute

Page



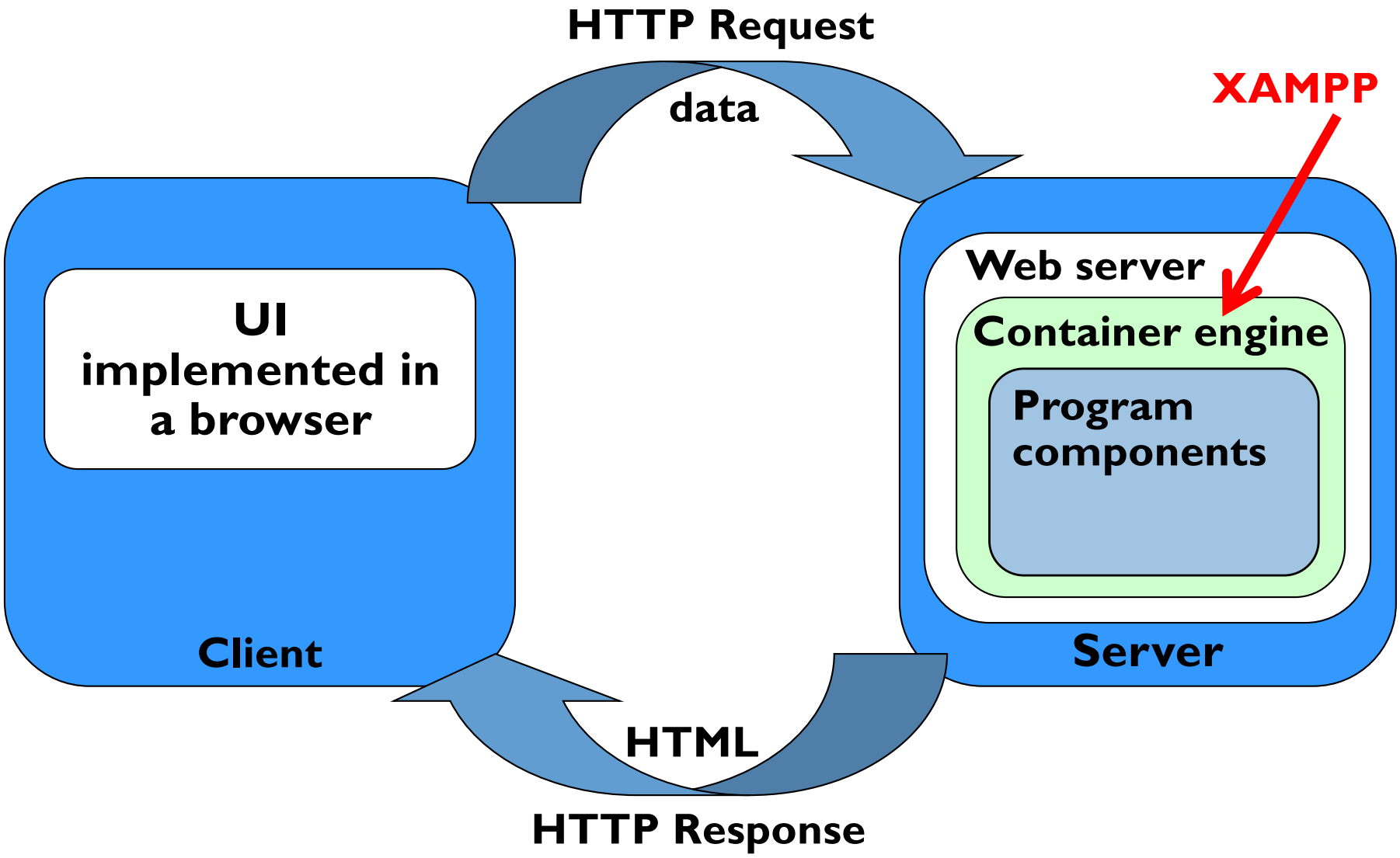
The browser receives an HTML page

It creates a model of the page and stores it in memory

```
container
bottom ...
```

It shows the page on screen using a rendering engine

Server Side Processing



Spaghetti Code

[ref: image by Ekarin Apirakthanakorn]

- Unstructured
- All-in-one
- Lack of style rules or patterns
- Hard to read, debug, maintain, reuse, evolve

General Design Issues

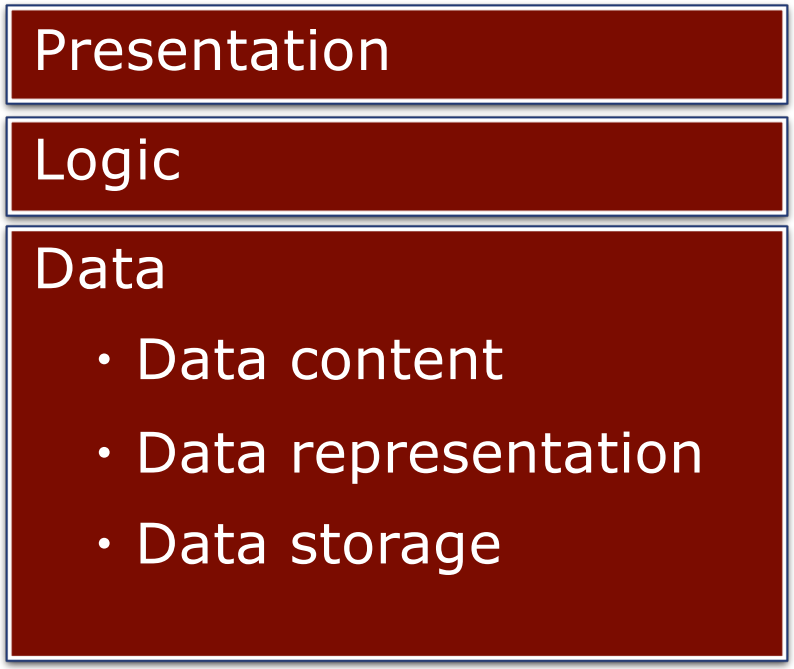
Separation of concerns: design goals of web app design

- Presentation
- Logic
- Data

Seven criteria the design should support:

- Reliability
- Usability
- Security
- Availability (and also Accessibility)
- Scalability
- Maintainability
- Performance and Time to market

Separation of Concerns



→ HTML, view

→ Functions and classes, software

→ Classes, objects

→ Data structure, classes, objects

→ Database and files, Oracle, SQL, NoSQL

Assumptions about data

- Data values: contents of memory – change very frequently (during execution)
- Data presentation: how the data are presented – change occasionally
- Data structure: types, organization and relationships of different data elements, changes infrequently

Architectural Styles

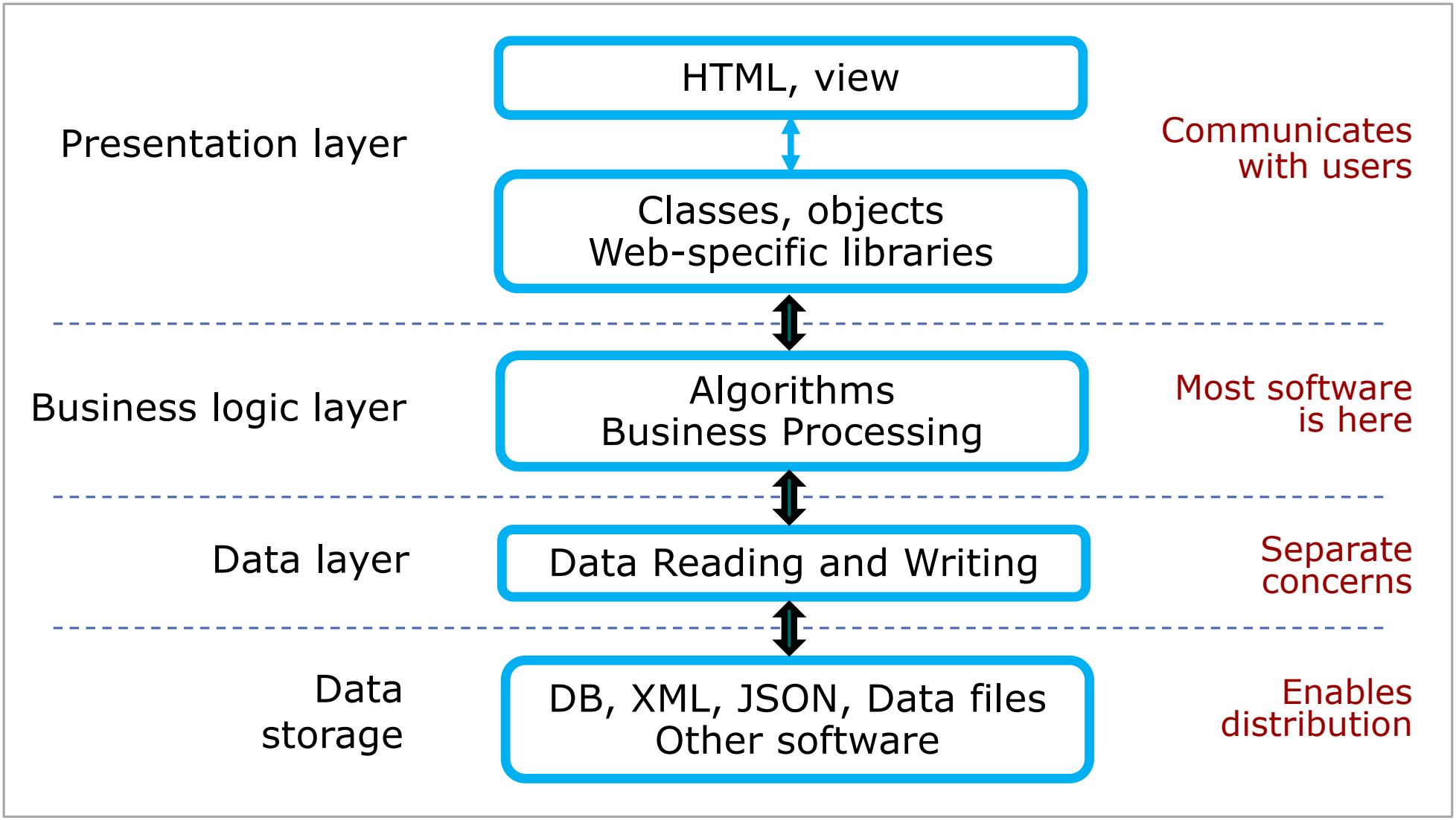
- How to partition a system
- How components identify and communicate with each other
- How information is communicated
- How elements in a system can evolve

Web architectural styles constantly change

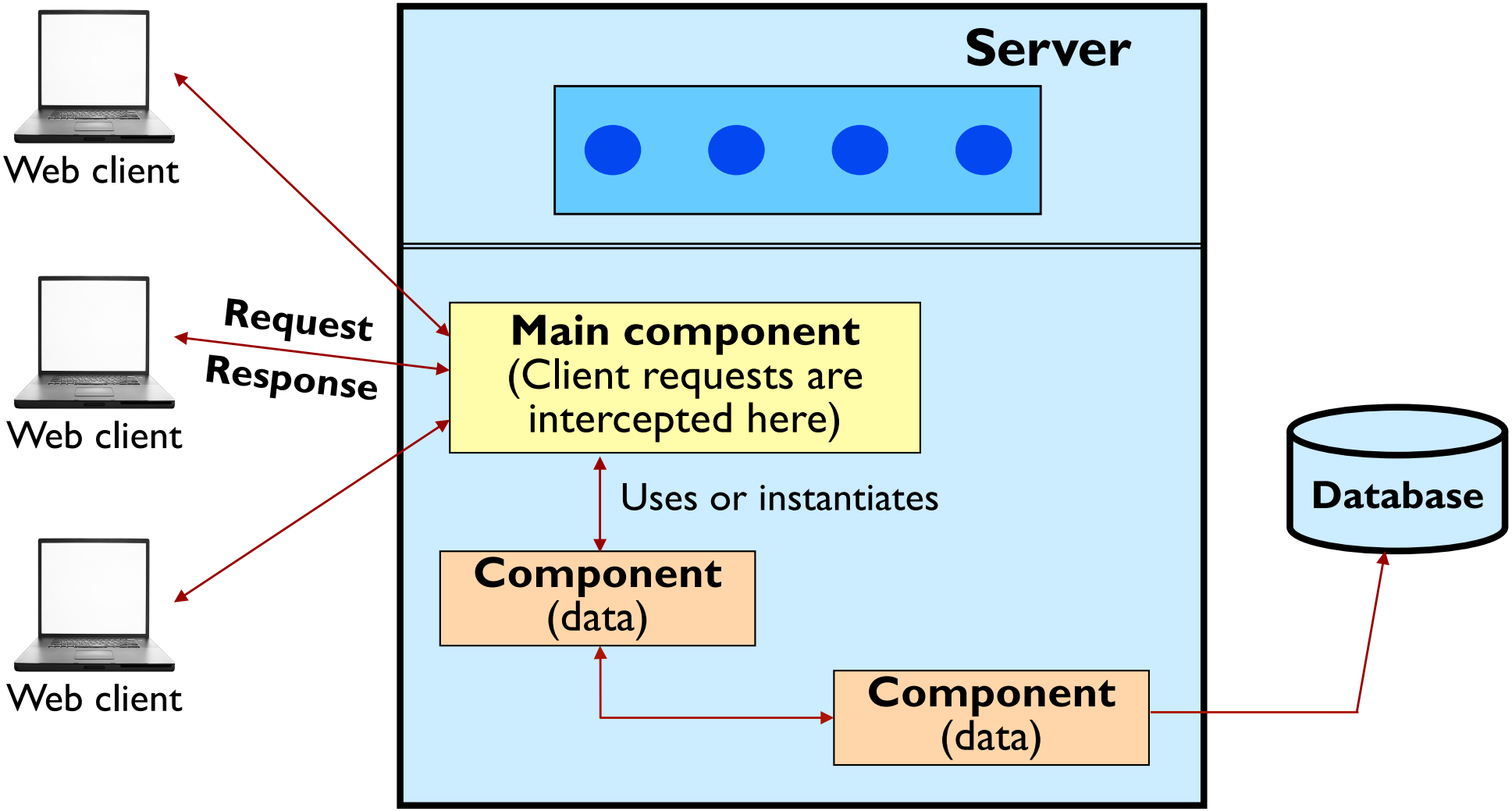
The goal is to **separate logic** from **presentation** and to **separate** as many **concerns** in the logic as possible – aiming to improve maintainability, reusability, scalability, responsiveness, and continuous development

Multi-Tiered Architectures

Each software layer only communicates with adjacent layers

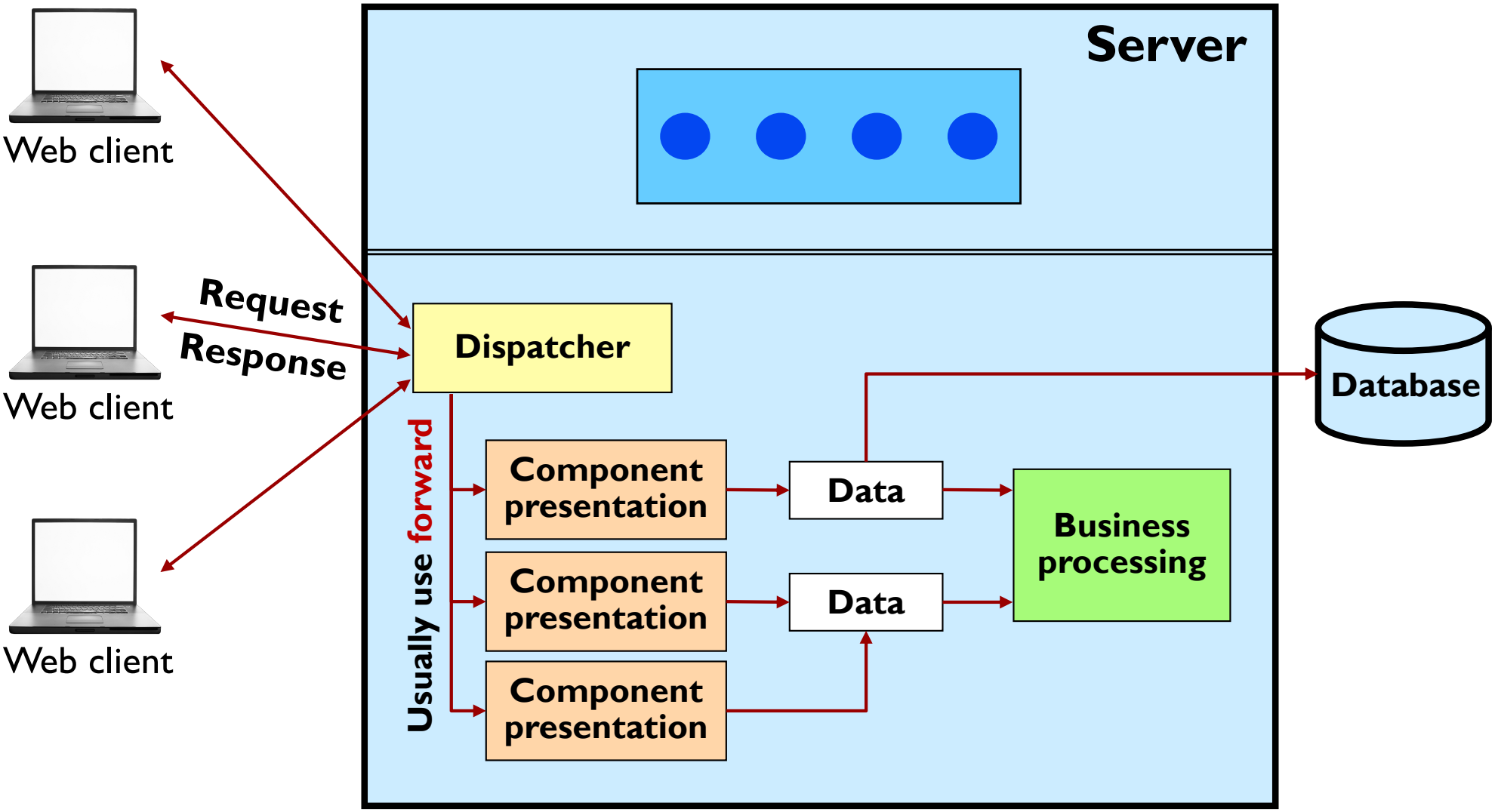


Page-centric Design



Requests are made to a main component and the main component response to clients – "client-server"

Dispatcher (N-tier) Design

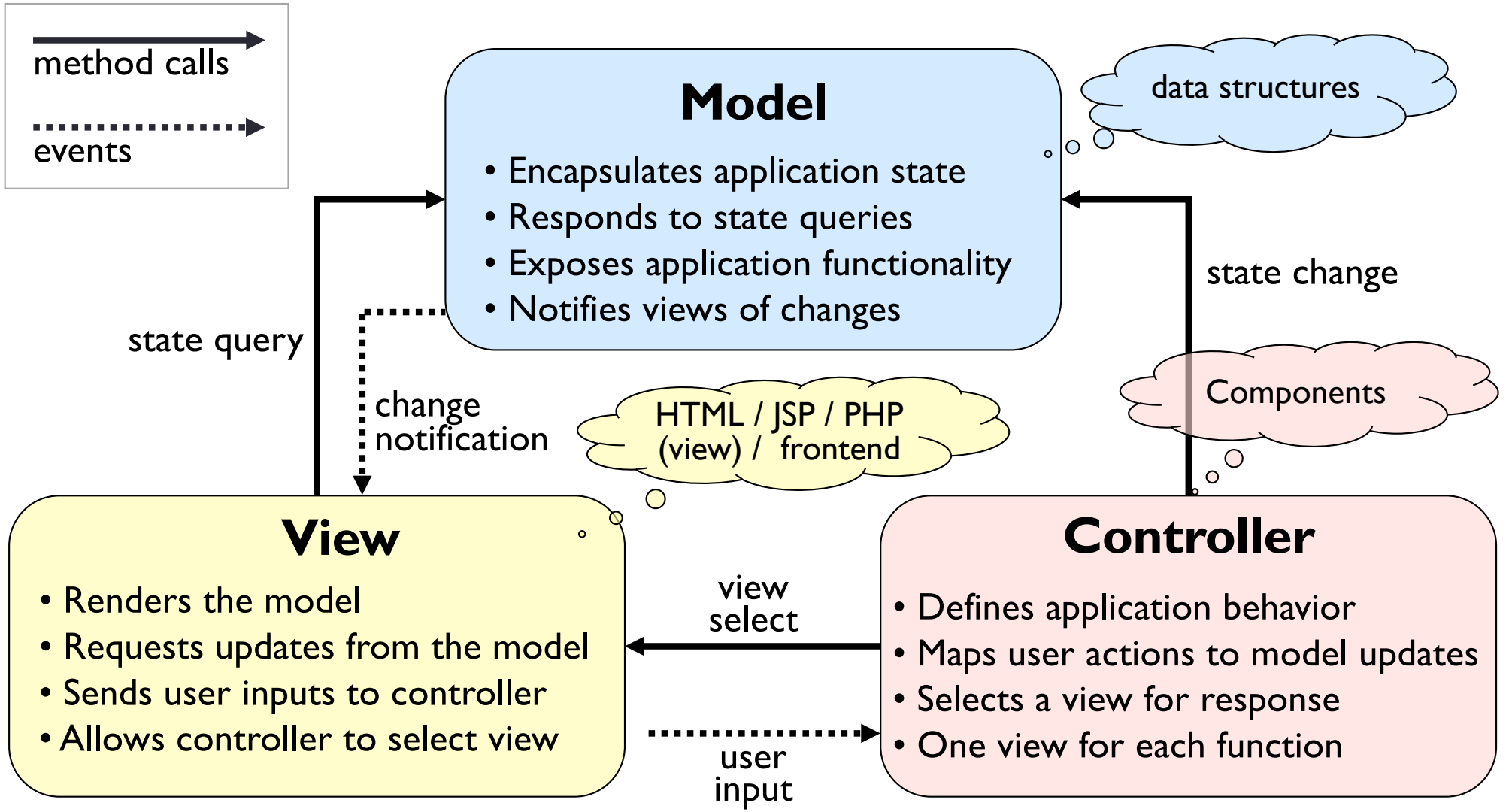


Requests are sent to a dispatcher that then forward the requests to another component (using *forward* or *redirect* control connection)

Model View Controller

- An abstraction frequently used in web app design
- Provide a way to divide the responsibilities of objects
- Decreases coupling between objects and layers (supports easier maintenance)
- Help divide the work (supports development expertise areas)

Model View Controller (2)



[Graphic from Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition, Nicholas Kassem et al., October 2000]

Server-Side Scripting

- Generate HTML on the server through scripts
- Early approaches emphasized embedding server code inside HTML pages
- Examples: PHP, JSP

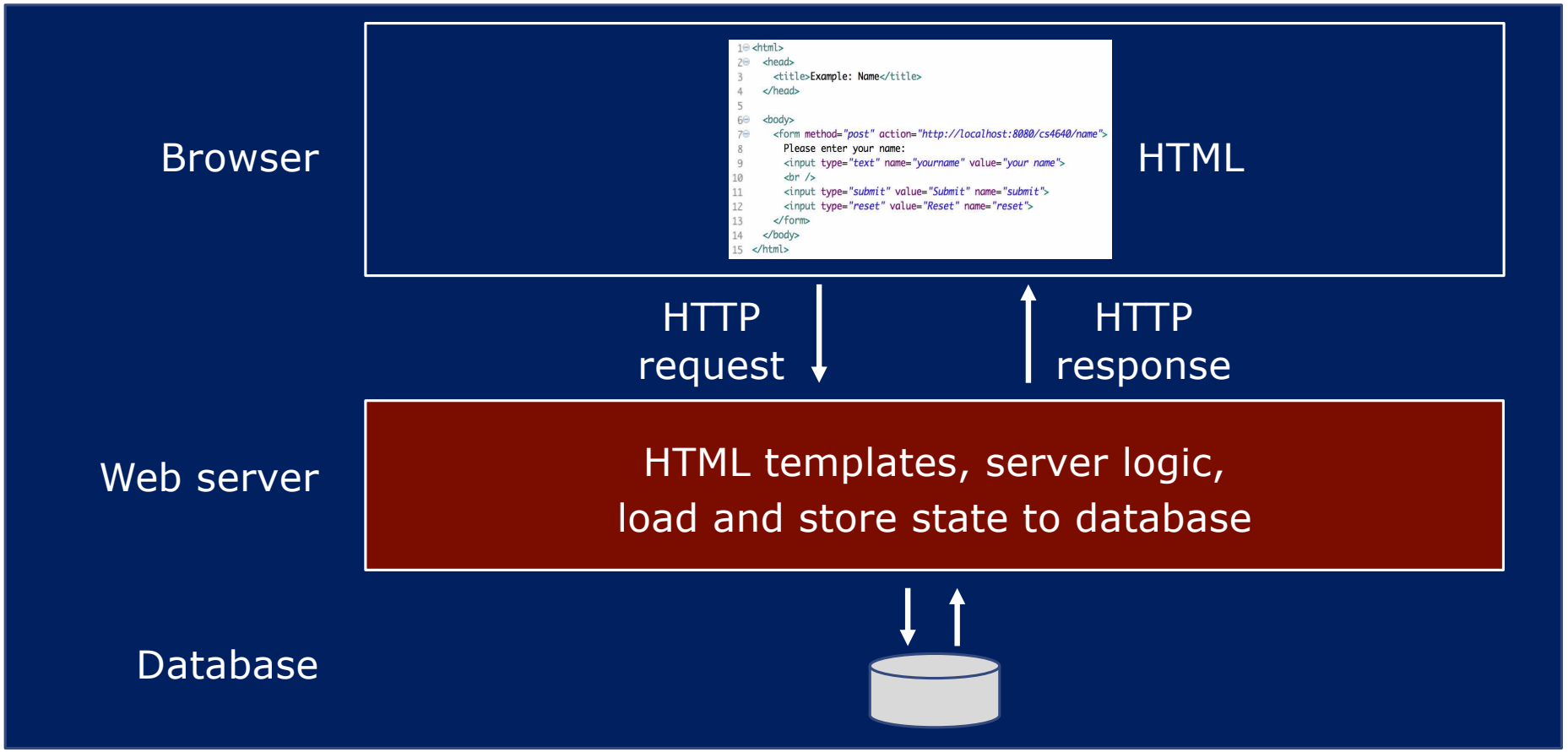
JSP

```jsp
<html>
  <head>
    <title>Counting with a JSP</title>
  </head>
  <body>
    <!-- Set global information for the page -->
    <%@ page language="java" %>

    <!-- Declare the variable -->
    <%! int count = 0; %>

    <!-- Scriptlet - Java code -->
    <%
      for (int i = 0; i < 10; i++)
      {
        count = count+1;
    %>
      <br />
      The counter value is: <%= count %>
    <% } %>
  </body>
</html>
```

PHP

```php
<!doctype html>
<html>
  <head>
    <title>Login example</title>
  </head>
  <body>
    You logged in as <font color="green"><b><?php echo $_POST["name"]; ?></b></font><br>
    with password <font color="green"><b><?php echo $_POST["pwd"]; ?></b></font>
  </body>
</html>
```

Server-Side Scripting Site



Advantages

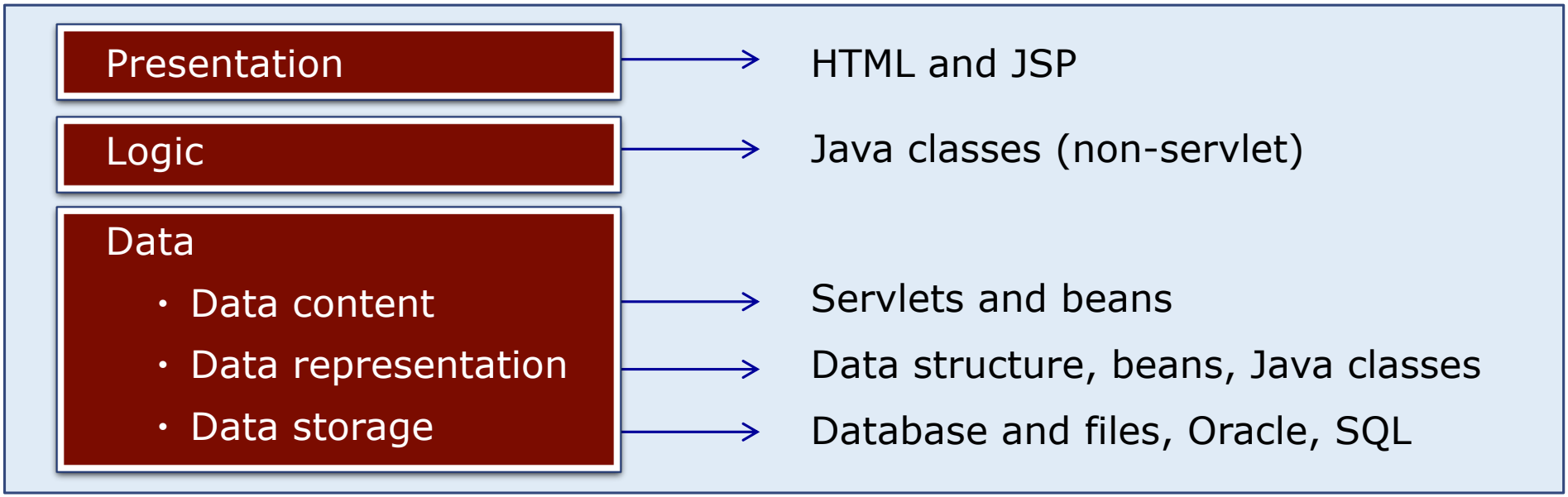
Server-side processing, browser independent, search optimization improvement, increased security

Disadvantages

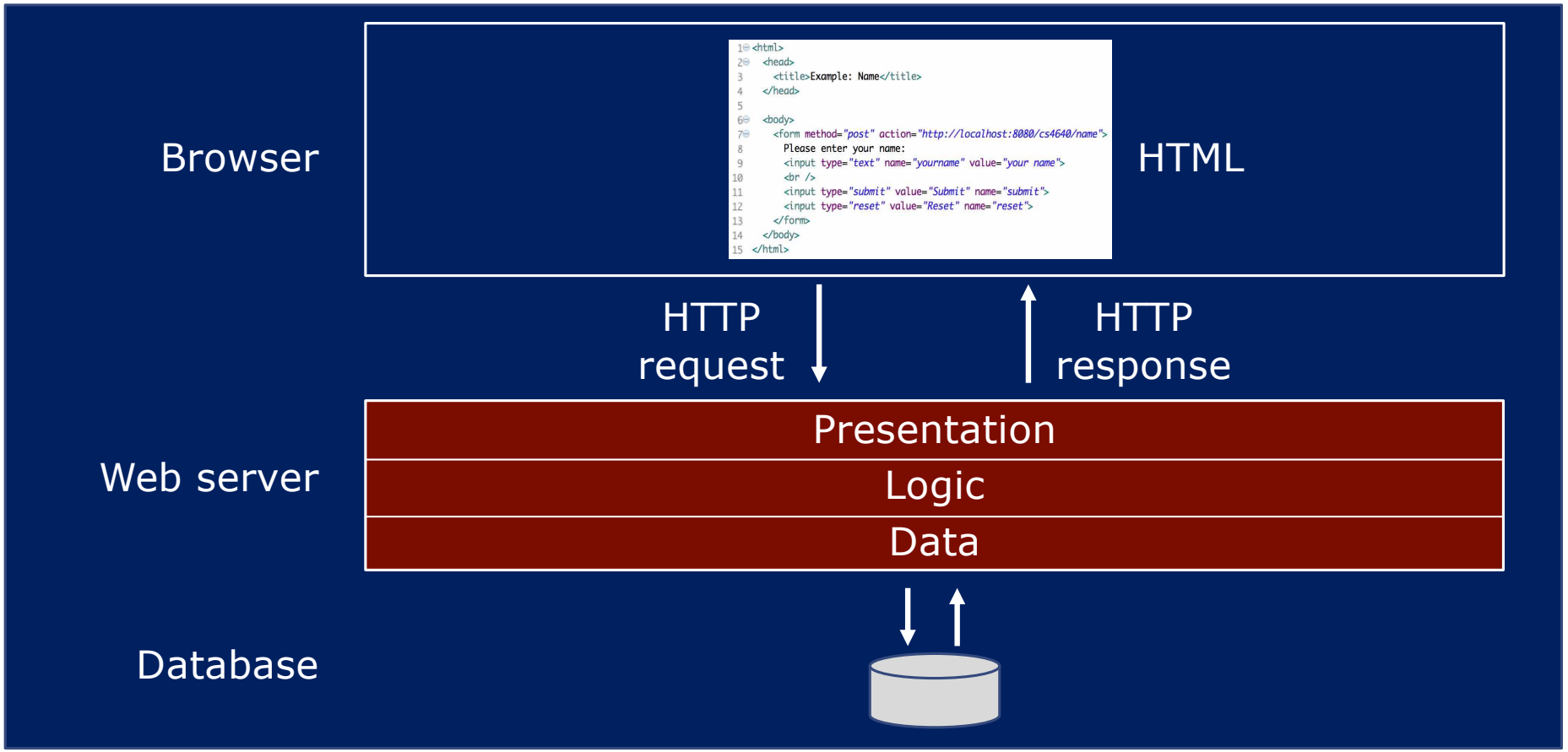
Poor modularity, hard to understand, difficult to maintain

Server-Side Framework

- Structure server into tiers, organizes logic into classes
- Execution on the server
- Can be single-page or multiple pages with page-centric design, dispatcher design, model-view-control design, or combination of any architectures
- Example: JSP (with separation of concerns)



Server-Side Framework Site



Advantages

Separation of concerns, maintainable and reusable

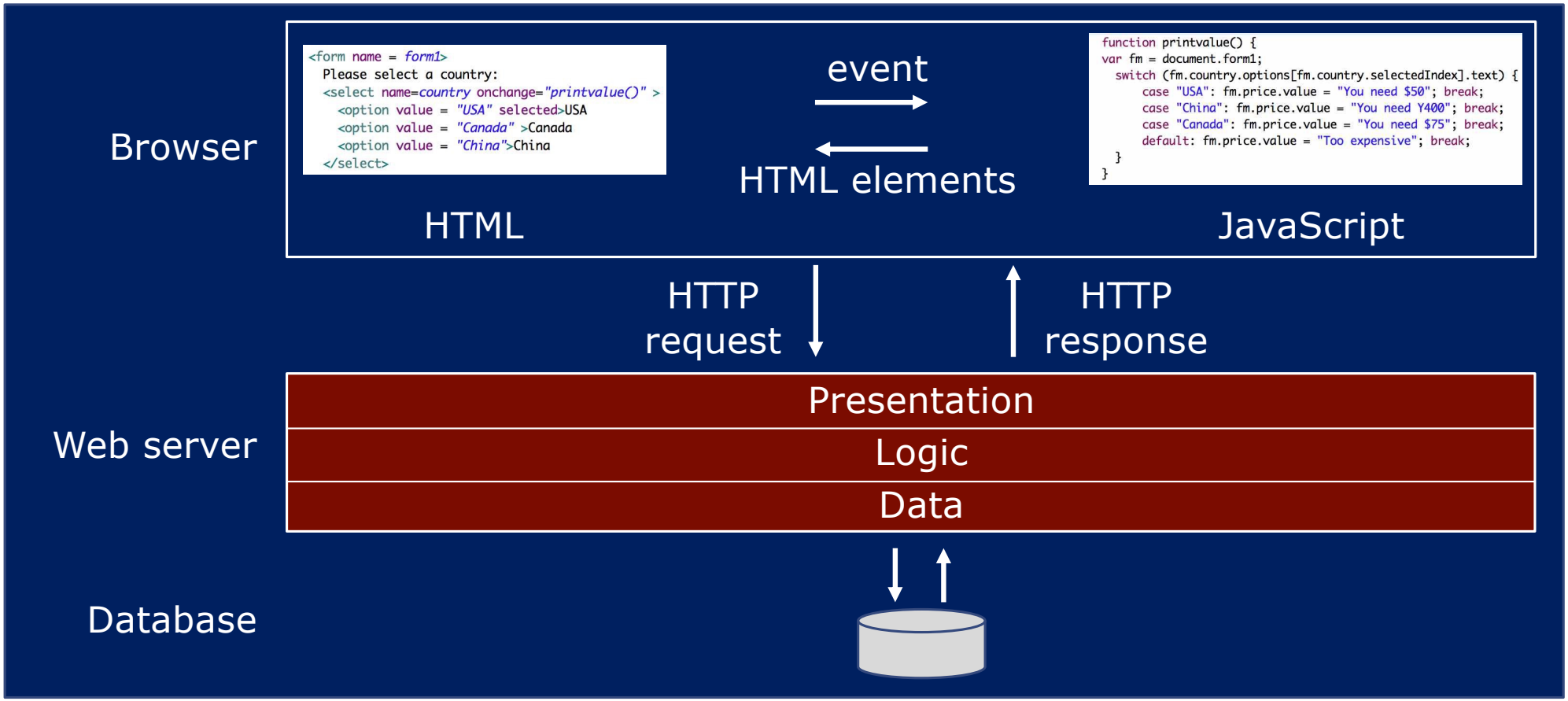
Disadvantages

Need to load an entire page to get new data

Single Page Application (SPA)

- Client-side logic sends messages to server, receives response
- Logic is associated with a single HTML page, typically written in JavaScript
- HTML elements dynamically added and removed through DOM manipulation
- Enabling technologies
  - AJAX – set of technologies for sending asynchronous request and receiving response
  - DOM Manipulation – methods for updating HTML elements
  - JSON – standard syntax for describing and transmitting JavaScript data object
  - JQuery – wrapper library built on HTML standard
- Example: Gmail

Single Page Application Site



Advantages

Fast (load most resources once; only data are transmitted)

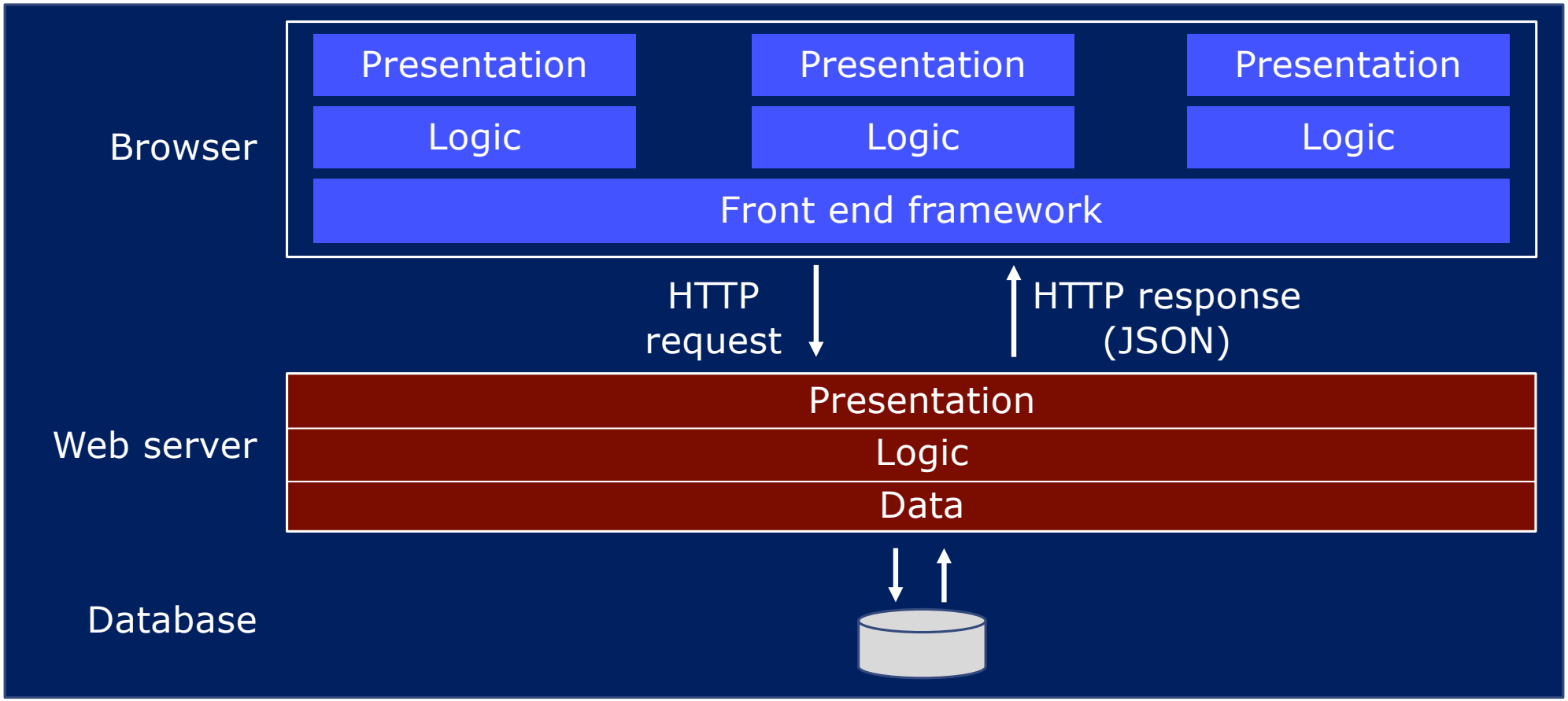
Disadvantages

Poor modularity client-side, tightly coupled logic to HTML elements (leading to redundant code), hard to maintain and reuse, cross-site scripting

Front-End Frameworks

- Client is organized into separate components, capturing model of web application data
- Components separate logic from presentation
- Components dynamically generate corresponding code based on component state
- Example: Angular, React

Front-End Framework Site



Advantages

Code organization, reuse, quick and easy to develop

Disadvantages

Duplicate logic in client and server, tight coupling – changes to server logic require changes to client logic, difficult to reuse server logic

Summary: Web Apps

- Deploy across the Web
  - Other (software) deployment methods include bundling, shrink-wrapping, embedding, and contracting
- New technologies
- New conceptual language constructs for programming
  - Integration
  - Data management
  - Control connections
- Browser features may affect the execution flow
- Architectures – no silver bullet

These differences affect every aspect of how to engineer high quality software