Web Application Development definitions & principles

CS 425

Instructions of this presentation

- The slides in this presentation go in pairs:
  - the first slide shows the concept to define (red background),
  - the second slide shows the answer (green background).

- The idea is to try to define the concept being discussed, and then reveal the solution by displaying the second slide afterward.

Question

#1 - Common web development tasks

Answer

#1 - Common web development tasks

- web design, web content development, client server side scripting, web server and network security configuration

Question

#2 - authoring (markup)

Answer

#2 - authoring (markup)

- preparing content for delivery

Question

#3 - styling

Answer

#3 - styling

- look and feel of a site's content

Question

#4 - Scripting (programming)

Answer

#4 - Scripting (programming)

- how a site works

Question

#5 - front end, back end, and full stack

Answer

#5 - front end, back end, and full stack

- three categories of web developers

Question

#6 - HTML, CSS, JavaScript

Answer

#6 - HTML, CSS, JavaScript

- 3 client side technologies used to build web pages that run on the client on the user's device

Question

#7 - client side programming

Answer

#7 - client side programming

- any aspect of the development process that relates to the browser only

Question

#8 - server side programming

Answer

#8 - server side programming

- applications that run on web servers and respond to requests from client side browsers (ordering products on Amazon, bidding on eBay auctions, banking online)

Question

#9 - web development process

Answer

#9 - web development process

- plan, design, build, beta, launch, support

Question

#10 - file system

Answer

#10 - file system

- hierarchy of directories that is used to organize applications and data; enables applications to store and retrieve files on storage devices

Question

#11 - file

Answer

#11 - file

- container for data that any application can find, inspect, and modify using standard interfaces provided by the operating system

Question

#12 - Windows File System
#13 - macOS File System

- Give an exemple of a file path and see what is different

Answer

#12 - Windows File System
#13 - macOS File System

- C:\Users\kroumina\Documents\CIS425
- /Users/juliechaumard/Downloads/CIS425

Question

#14 - General web development workflow

Answer

#14 - General web development workflow

- Create/open web doc in text editor, load web document onto browser, modify document's content in editor, refresh, commit document to repository when satisfied

Question

#15 - GIT

Answer

#15 - GIT

- allows for back tracking

Question

#16 - Describe the Components of a web application

Answer

#16 - Describe the Components of a web application

- Clients - use program known as web browsers to pull from server Web Server - serve the clients requests Network - connects the client and server

Question

#17 - Describe the four components of a URL

Answer

#17 - Describe the four components of a URL

- https://learn.schiller.edu/ultra/courses/_11303_1/outline
- protocol : https://
- sub-domain name : learn
- domain name : schiller.edu
- path : ultra/courses/_11303_1
- file name : outline

Question

#18 - distinguish between static and dynamic web pages, with the focus on the web server, application serve, and database server

Answer

#18 - distinguish between static and dynamic web pages, with the focus on the web server, application serve, and database server

- A static web page is a page that doesn't change each time it is requested (html)
- A dynamic web page is a page that's created by a program on an application server (.aspx)
- A static website is a website where all the elements on the page remain static-or the same. Reversely, a dynamic website can change itself based on the user, the time or more. The static websites are mostly built using HTML and CSS, while dynamic websites use server-side languages such as PHP, Ruby, server-side JavaScript, and Python.
- The applications for static websites could be portfolio websites, personal blogs and resumes, or documentation sites. On the other hand, the real-life application for dynamic websites could be E-commerce websites, social media sites such as Facebook, LinkedIn, and others.

Answer

Feature	Static	Dynamic
Content	Fixed content	Content changes based on user interaction, time, or database data
Technology used	Build using HTML, CSS, and sometimes JavaScript	Server-side languages: PHP, Python, ASP.NET, Node.js
Database	No database required	Use database (MySQL, MongoDB) to store and retrieve data
Interactivity	Limited	Highly interactive – logins, comments, shopping carts
Performance	Faster – pre-built pages	Slightly slower → real-time data fetching and processing
Maintenance	Easier to develop but harder to update	Easier to update since data and layout are separated
Hosting cost	Low cost	Higher cost
Example	Personal portfolio, company landing page	Social media platforms, e-commerce websites, blogs, online forums
Scalability	Less scalable	Highly scalable
User experience	Simple and static	Personalized and dynamic

Question

#19 - Describe these terms as they relate to HTML documents: element, block element, inline element, opening tag, content, closing tag, selfclosing tag, and attribute

Answer

#19 - Describe these terms as they relate to HTML documents: element, block element, inline element, opening tag, content, closing tag, selfclosing tag, and attribute

- Most HTML elements are made up of three parts. -Opening tag: marks the start of the element, consists of the element name (such as h1) plus one or more optional attributes (such as id or class) that provide additional information for the tag -Content: text or other data that makes up the element -Closing tag: marks the end of the element, consists of a slash followed by the element's name

Question

#20 - What is a self closing tag?

Answer

#20 - What is a self closing tag?

- Don't have closing tags

Question

#21 - The use and benefits of external style sheets

Answer

#21 - The use and benefits of external style sheets

- The real power of using an external style sheet is that multiple web pages on our site can link to the same style sheet
- By editing the external style sheet, we can make site-wide changes (even to hundreds of pages) instantly

Question

#22 - Three ways to include CSS in a web page

Answer

#22 - Three ways to include CSS in a web page

- element type, class, and id inline style, internal style sheet, external style sheet

Question

#23 - List 3 frameworks of the web application development that are most used in the industry nowadays

Answer

#23 - List 3 frameworks of the web application development that are most used in the industry nowadays

- Front-end frameworks (react, angular, vue)
- Back-end frameworks (Node.js, Django, Flask, Laravel, ASP.NET core, Spring Boot, Ruby on Rails) manage data, logic and APIs

Answer

#	Framework	Programming Language	Common Tools	APIs	Libraries	DBMS Commonly Used
1	React.js	JavaScript	Visual Studio Code, npm, Git	RESTful APIs, GraphQL	Redux, Axios, React Router	Firebase, MongoDB
2	Angular	TypeScript / JavaScript	Angular CLI, VS Code, Git	RESTful APIs, RxJS	NgRx, Bootstrap, RxJS	MySQL, Firebase
3	Vue.js	JavaScript	Vue CLI, Webpack, npm	RESTful APIs, Axios	Vuex, Vuetify	Firebase, MySQL
4	Node.js (Express.js)	JavaScript	npm, Postman, GitHub	Express API, GraphQL	Mongoose, Lodash, Cors	MongoDB, PostgreSQL
5	Django	Python	PyCharm, VS Code, Docker	Django REST Framework	NumPy, Pandas, Bootstrap	PostgreSQL, SQLite
6	Flask	Python	PyCharm, VS Code, Postman	Flask-RESTful API	Jinja2, Requests	SQLite, MySQL
7	Laravel	PHP	Composer, Artisan CLI, XAMPP/WAMP	Laravel API Routes	Eloquent ORM, Blade	MySQL, PostgreSQL
8	Ruby on Rails	Ruby	RubyMine, Git, Heroku	Active Record API	RSpec, Capybara	PostgreSQL, SQLite
9	ASP.NET Core	C#	Visual Studio, IIS, GitHub	ASP.NET Web API	Entity Framework, LINQ	SQL Server, Azure SQL
10	Spring Boot	Java	IntelliJ IDEA, Maven, Eclipse	Spring REST API	Hibernate, Thymeleaf	MySQL, PostgreSQL

Question

#24 - List 2 inline elements and 2 block elements (do not use <img> as any of your answers - it's a special "inline-block" element).

Answer

#24 - List 2 inline elements and 2 block elements (do not use <img> as any of your answers - it's a special "inline-block" element).

- Possible inline elements: a, img, q, span Possible block elements: p, div, section, h1-6, blockquote

Question

#25 - Why do we always want to include an alt attribute on img tags?

Answer

#25 - Why do we always want to include an alt attribute on img tags?

- Users who cannot see the image due to vision impairment can have a textual description of the image (which can be spoken aloud by a screenreader)
- If the image fails to load (connection, broken path, etc.), the alt text is displayed instead
- SEO (Search Engine Optimization) benefits

Question

#26 - What’s the difference between margin, borders, and padding?

Answer

#26 - What's the difference between margin, borders, and padding?

- The margin, border, and padding are three parts of the CSS box model that define the spacing around and inside an element:
- Margin: The space outside the element's border. It separates the element from others around it.
- Border: The line that surrounds the padding and content. It can have color, width, and style.
- Padding: The space inside the element, between the content and the border.

Question

#27 - Exercise

- You are required to develop a website that allows users can order and buy products online. The requirement of the application is described as follows:
- Drawing a use-case diagram (who does what) to analyze user requirements with at least 3 actors, different use-cases for each and explanation
- Drawing a diagram of the system architecture that you use to implement the application based on an MVC model. Showing the techniques, framework, DBMS and explanation of interaction among them

Answer

- We'll use 3 main actors: 1. Visitor / Customer 2. Admin 3. Payment Gateway (external system like Stripe/PayPal)

Answer

- Actor 1: Visitor / Customer Use-cases:
  - Browse products
  - Search products
  - Register / Login
  - Add product to cart
  - View / Update cart (change quantities, remove items)
  - Checkout
  - Make payment (interacts with Payment Gateway)
  - View order history (only when logged in)
  - Update profile / delivery address
  - A visitor can see the product catalog and search products. If they want to buy, they create an account or log in. Then they add products to the shopping cart, review or edit the cart, and start checkout. During checkout they confirm the delivery address, shipping method and finally trigger the payment, which calls the external payment gateway. After the order is placed, a registered customer can view previous orders and update their personal information.
  - Checkout **includes** Calculate total and Choose shipping method
  - Make payment extends Checkout

Answer

- Actor 2: Admin Use-cases:
  - Login
  - Manage products (create, update, delete products, set price, stock, images)
  - Manage categories (create / edit categories, assign products to categories)
  - Manage orders (view orders, update status: pending, shipped, delivered, canceled)
  - Manage users (view customers, block/unblock accounts)
  - View reports / dashboard (sales reports, best-selling products, low stock alerts)
  - The admin logs into a back-office. They can create and edit products and categories, change prices and stock, and upload images. They can also see all orders, change their status and handle issues such as cancellations or refunds. The admin can manage customer accounts (e.g. deactivate them) and access simple reports to monitor sales and inventory.
  - Manage products includes Add product, Edit product, Delete product

Answer

- Actor 3: Payment Gateway (external system) Use-cases (from the system’s point of view):
  - Process payment
  - Send payment confirmation / failure
  - When the customer confirms checkout, the website sends the order amount and payment details to the payment gateway (e.g. Stripe). The gateway securely processes the card or wallet payment and returns a success or failure response. Based on this response, the system confirms the order or shows an error to the user.

Answer

- System architecture diagram (MVC model)
  - Implementation of the website with an MVC architecture.
  - Stack example :
  - Frontend (View): HTML, CSS, JavaScript (+ optional framework like React or Vue.js)
  - Backend (Controller + Model): Node.js with Express (MVC structure)
  - DBMS: MySQL

Answer

- The request goes down: **browser → controller → model → database.**

[User Browser]
  |
  v
[View layer]
 - HTML / CSS
 - JavaScript (e.g. Vue.js/React)
  |
  | HTTP requests (GET, POST, PUT, DELETE) via HTTPS / JSON
  v
[Controller layer - Express.js]
 - Routes: /products, /cart, /orders, /admin/...
 - Business logic: validation, security, sessions
  |
  v
[Model layer]
 - JS model classes / ORM (e.g. Sequelize, Prisma)
 - SQL queries
  |
  v
[DBMS - MySQL]
 - Tables: users, products, categories, orders, order_items, payments

Answer

- The response goes up: **database → model → controller → view → browser.**

[DBMS - MySQL]
- Tables: users, products, categories, orders, order_items, payments
|
| Result of the SQL query (rows)
v
[Model layer]
- JS model classes / ORM (e.g. Sequelize, Prisma)
- Transforms SQL rows into JavaScript objects
|
| Returns the objects to the controller (return products, user, etc.)
v
[Controller layer - Express.js]
- Receives the data from the model
- Applies final business logic if needed
- Chooses the response: HTML (render) or JSON (API)
|
| Sends the HTTP response (HTML or JSON) to the browser
v
[View layer]
- If server-side: template engine (EJS, Pug, Handlebars)
- If front-end framework: React / Vue.js receives JSON and updates the DOM
|
v
[User Browser]
- Displays the page with data retrieved from the database

Answer

- The **View** is the web interface in the browser, created with HTML, CSS and JavaScript (optionally Vue.js or React to build dynamic components). It displays products, forms and messages to the user.
- The **Controller** is implemented with **Express.js** (Node.js). Each route (for example /products, /cart, /checkout) receives HTTP requests from the browser. The controller validates input data, manages sessions, applies business rules and decides which model functions to call.
- The **Model** represents the data and database access. We use JavaScript model classes combined with SQL queries to interact with **MySQL**. Models know how to create, read, update and delete records in the tables users, products, orders, etc.
- The **DBMS (MySQL)** stores persistent data: users, encrypted passwords, products, stock, orders, payments.
- During **checkout**, the controller sends a request to an external **Payment Gateway** via a secure HTTP API. The gateway processes the payment and sends back a response. The controller updates the order status in the database and returns a confirmation view to the user.

💡 This separation of concerns makes the application easier to maintain:

- Views handle presentation,
- Controllers handle request/response and business logic,
- Models + DBMS handle data storage and consistency.