UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 4351 Fall 2023

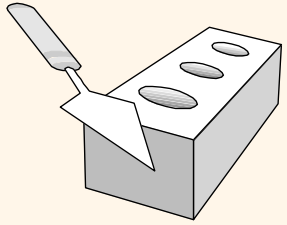
Software Engineering

M & W 4 to 5:30 PM

Prof. Victoria Hilford

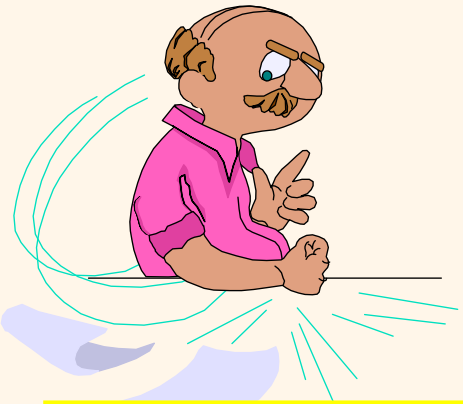
PLEASE TURN your webcam ON

NO CHATTING during LECTURE



COSC 4351

4 to 5:30



PLEASE

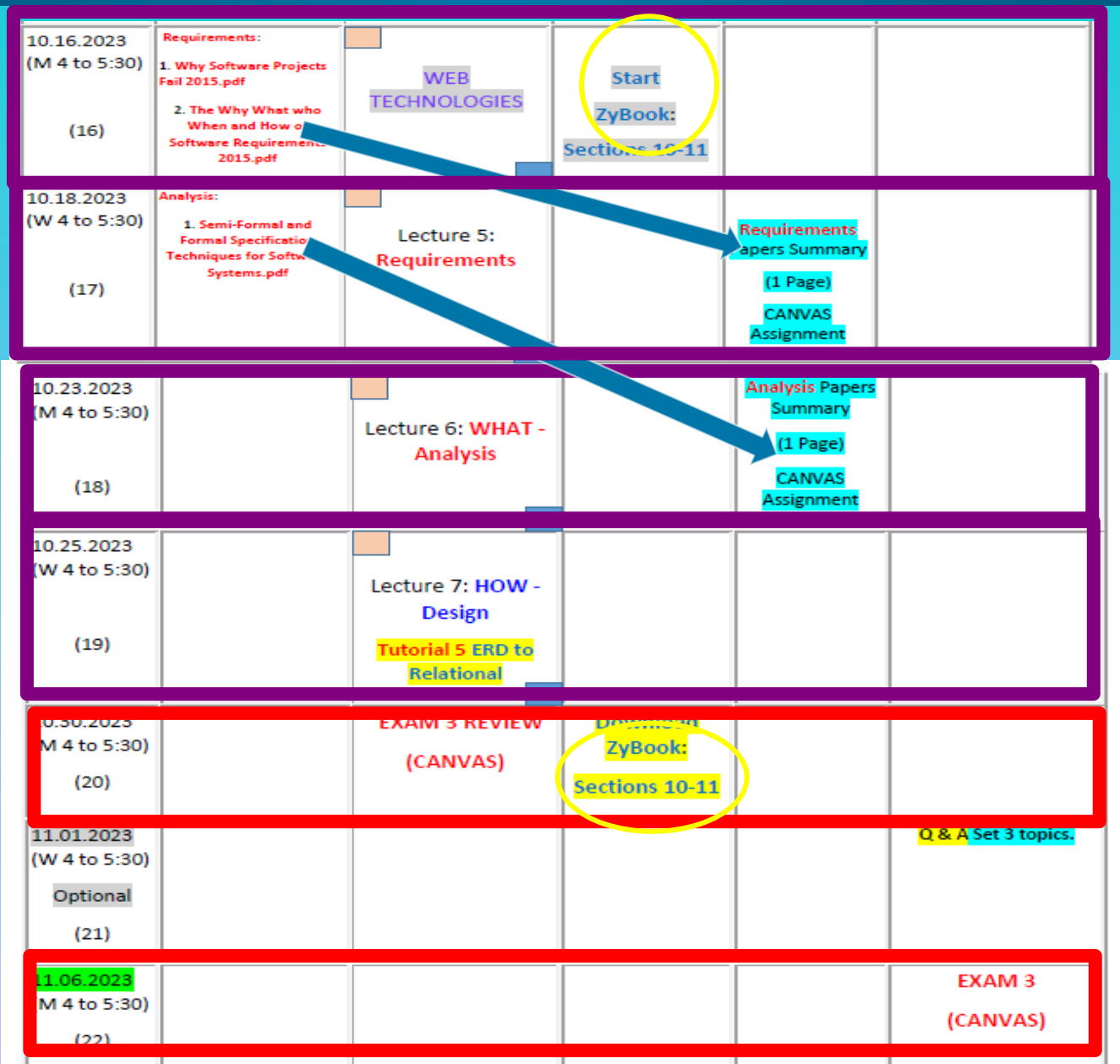
LOG IN

CANVAS

Youyi [A-L]

Kevin [M-Z]

Please close all other windows.


10.16.2023
(M 4 to 5:30)
(16)
Requirements:
1. Why Software Projects Fail 2015.pdf
2. The Why What who When and How o Software Requirements 2015.pdf
WEB TECHNOLOGIES
Start ZyBook:
Sectio 10-11
10.18.2023
(W 4 to 5:30)
(17)
Analysis:
1. Semi-Formal and Formal Specificatio Techniques for Softw Systems.pdf
Lecture 5: Requirements
Requirements apers Summary
(1 Page)
CANVAS Assignment
10.23.2023
(M 4 to 5:30)
(18)
Lecture 6: WHAT - Analysis
Analysis Papers Summary
(1 Page)
CANVAS Assignment
10.25.2023
(W 4 to 5:30)
(19)
Lecture 7: HOW - Design
Tutorial 5 ERD to Relational
M 4 to 5:30)
(20)
EXAM 3 REVIEW
(CANVAS)
ZyBook:
Sections 10-11
11.01.2023
(W 4 to 5:30)
Optional
(21)
Q & A Set 3 topics.
11.06.2023
M 4 to 5:30)
(22)
EXAM 3
(CANVAS)

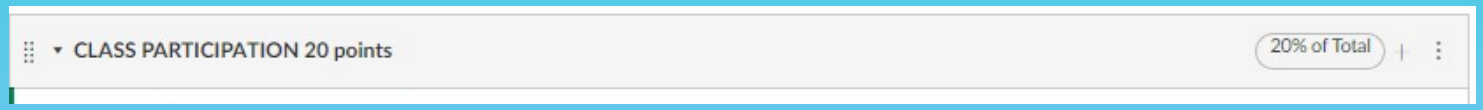
Class 16

WEB TECHNOLOGIES

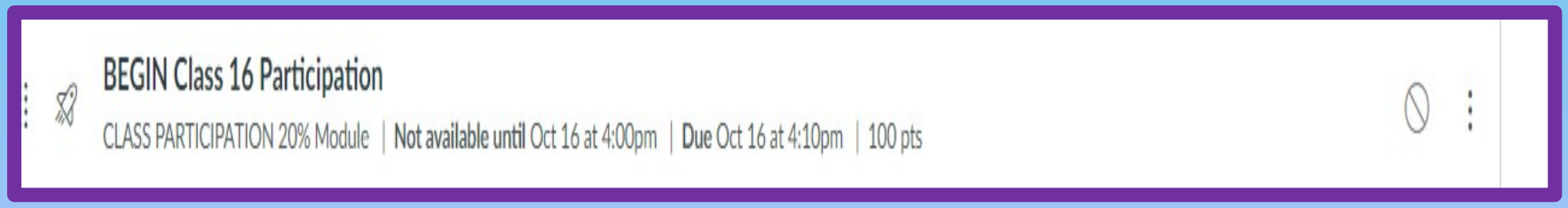
10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		

From 4:00 to 4:10 – 10 minutes.

10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		



PASSWORD: IN TEAMS

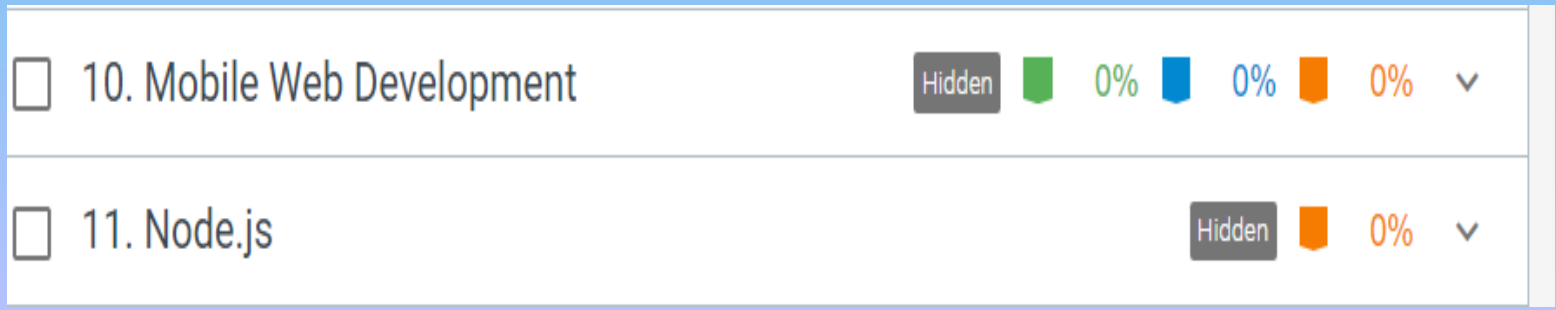


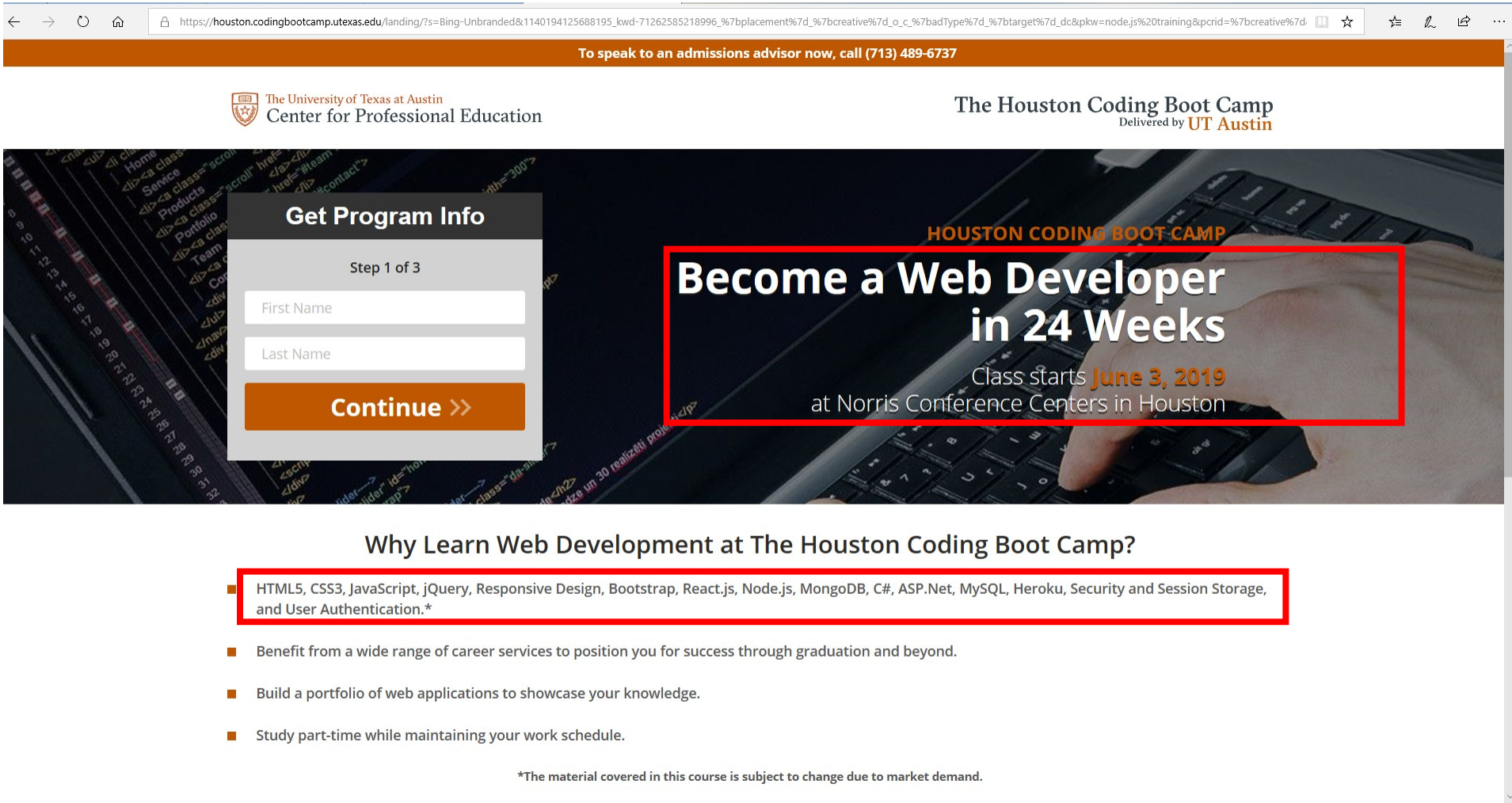
VH– Publish.

From 4:10 to 5:00 PM – 50 minutes.

10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		

SET 3
Sections 10 - 11

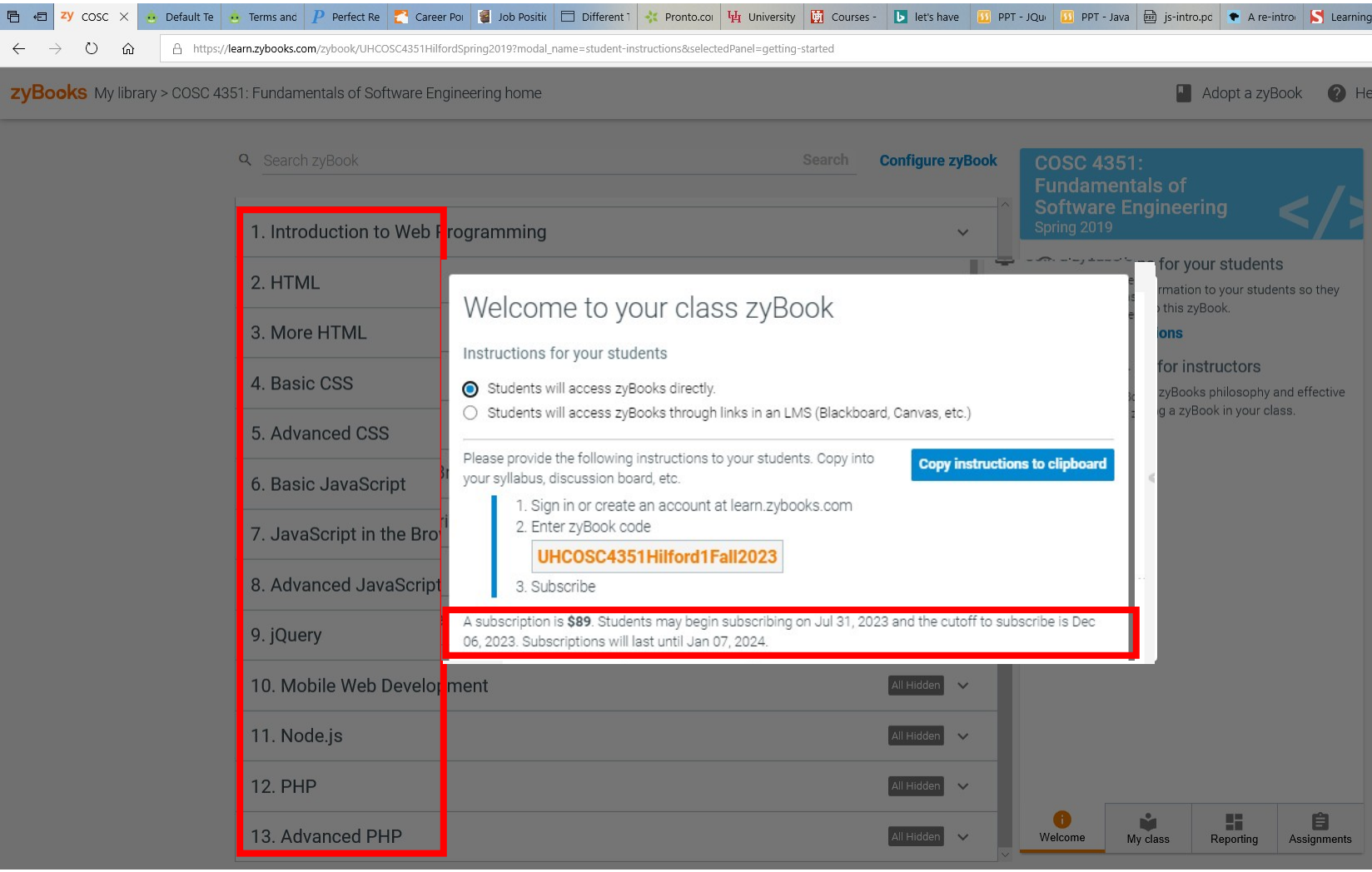




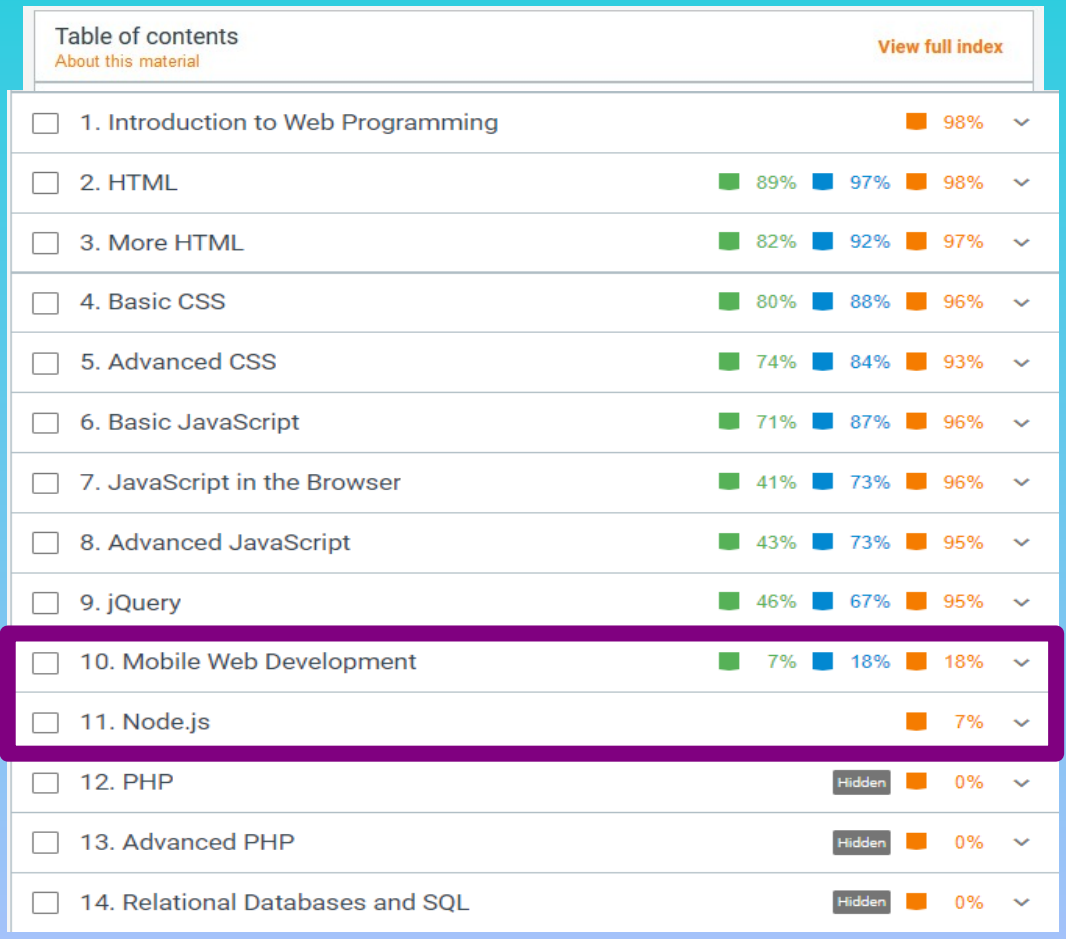
Why Learn Web Development at The Houston Coding Boot Camp?

- HTML5, CSS3, JavaScript, jQuery, Responsive Design, Bootstrap, React.js, Node.js, MongoDB, C#, ASP.Net, MySQL, Heroku, Security and Session Storage, and User Authentication.*
- Benefit from a wide range of career services to position you for success through graduation and beyond.
- Build a portfolio of web applications to showcase your knowledge.
- Study part-time while maintaining your work schedule.

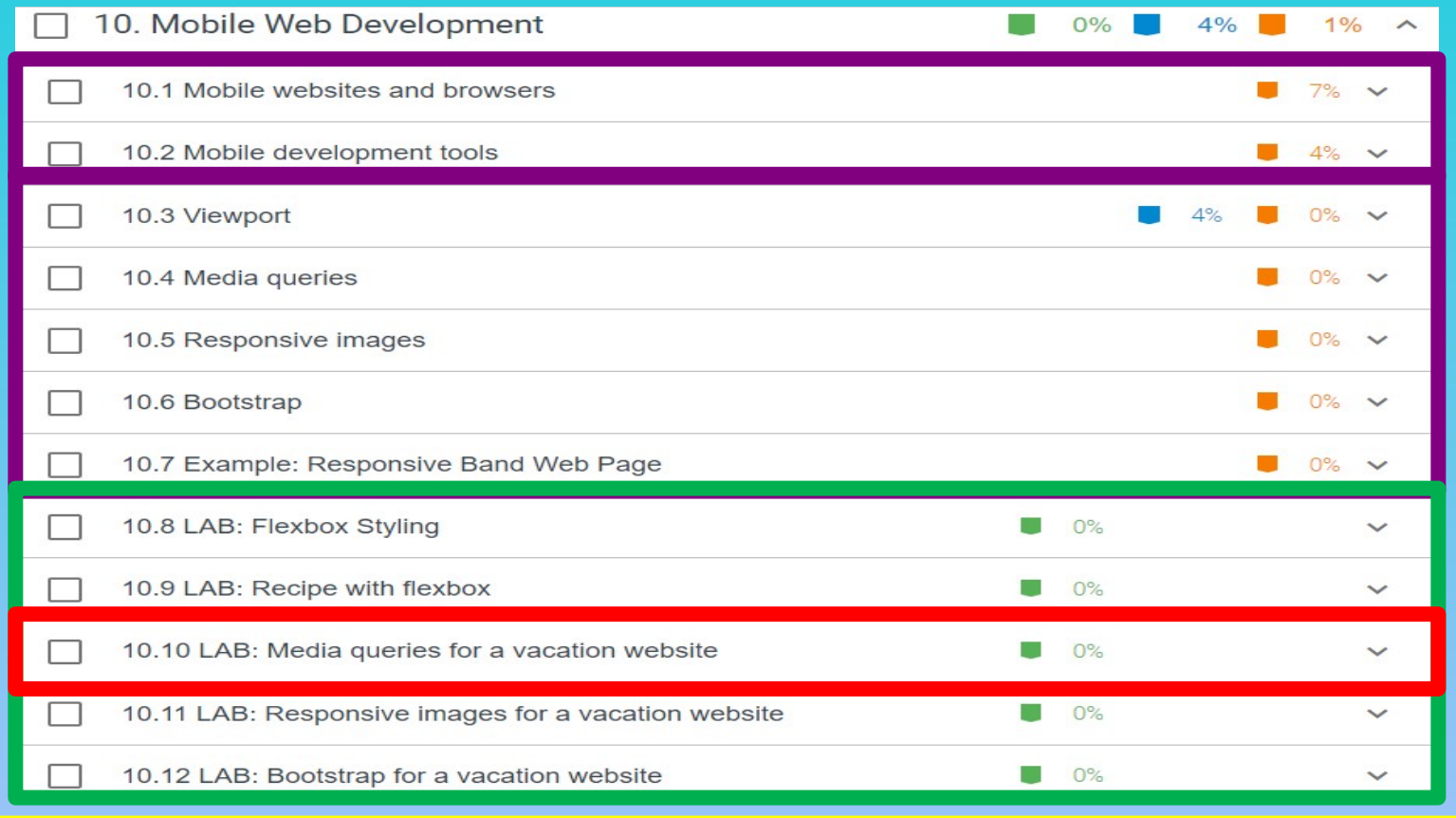
*The material covered in this course is subject to change due to market demand.


zyBooks My library > COSC 4351: Fundamentals of Software Engineering home
Adopt a zyBook
Help/FAQ
Victoria Hilford
Search zyBook
Search
Configure zyBook
COSC 4351: Fundamentals of Software Engineering
Spring 2019
1. Introduction to Web Programming
2. HTML
3. More HTML
4. Basic CSS
5. Advanced CSS
6. Basic JavaScript
7. JavaScript in the Bro
8. Advanced JavaScript
9. jQuery
10. Mobile Web Development
11. Node.js
12. PHP
13. Advanced PHP
All Hidden
Welcome to your class zyBook
Instructions for your students
Students will access zyBooks directly.
Students will access zyBooks through links in an LMS (Blackboard, Canvas, etc.)
Please provide the following instructions to your students. Copy into your syllabus, discussion board, etc.
Copy instructions to clipboard
1. Sign in or create an account at learn.zybooks.com
2. Enter zyBook code
UHCOSC4351Hilford1Fall2023
3. Subscribe
A subscription is $89. Students may begin subscribing on Jul 31, 2023 and the cutoff to subscribe is Dec 06, 2023. Subscriptions will last until Jan 07, 2024.
for your students
for instructors
Welcome
My class
Reporting
Assignments

ZyBook – Web Programming



10. Mobile Web Development



No DETAILED questions in the EXAM from this section.

10. Mobile Web Development

Mobile web browsers

Prior to 2007, a number of markup languages were used to create simple webpages for first-generation smartphones and personal digital assistants (PDAs). Ex: HDML, WML, cHTML, iHTML, and XHTML-MP. The introduction of the iPhone in 2007 inaugurated the smartphone era and led to the development of innovative mobile web browsers that could render the same webpages designed for desktop web browsers. A ***mobile web browser*** is a web browser designed for mobile devices that can display webpages using HTML, CSS, and JavaScript.

Today's mobile browsers are much like their desktop counterparts, although mobile browsers often lack the ability to use plugins developed for desktop browsers. The popularity of smartphones has encouraged web developers to create mobile websites. A ***mobile website*** is a website that is designed for mobile devices with smaller screen sizes and touch interfaces.

10. Mobile Web Development

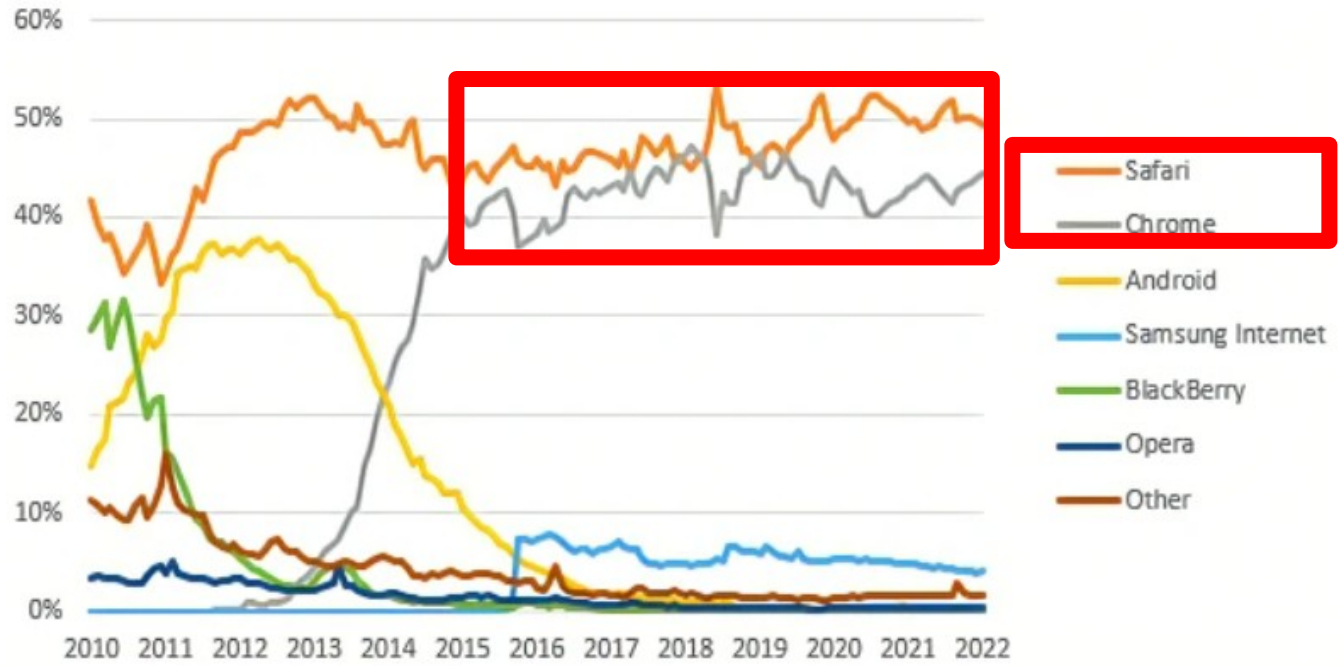
Figure 10.1.1: CNN.com for desktop and mobile browsers.

10. Mobile Web Development

Figure 10.1.2: Desktop vs. mobile access in North America.

10. Mobile Web Development

Figure 10.1.3 Top six mobile browsers in North America.



Source: StatCounter.com

10.2 Mobile Development tools

10.2 Mobile development tools

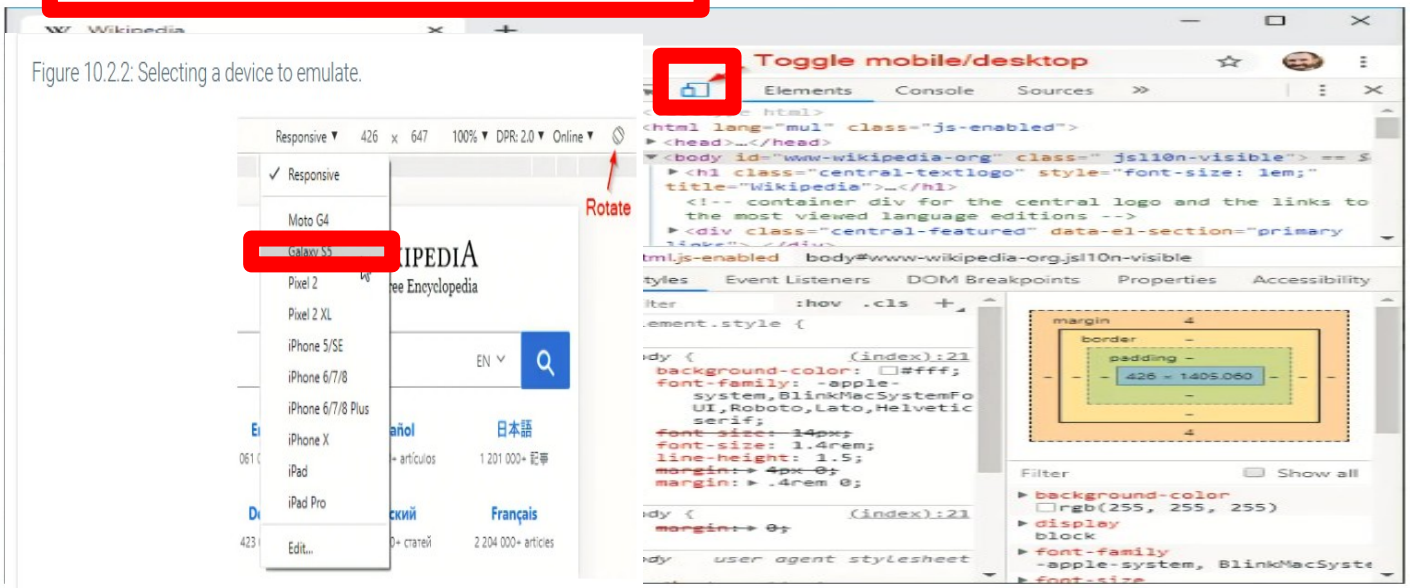
Present Note

Screen emulator

Developers must test their mobile websites on a variety of mobile devices to ensure the websites work properly for all users. However, many desktop browsers contain development tools that aid mobile website development without using a mobile device.

The Chrome desktop browser's DevTools provides a screen emulator that can mimic a wide range of smartphones and tablets. A ***screen emulator*** is software that simulates how mobile device screens operate. Developers access the DevTools screen emulator by typing Ctrl-Shift-I in Chrome (Windows) or Command-Option-I (Mac).

Figure 10.2.1 Chrome DevTools screen emulator.



Source: en.wikipedia.org

10.4 Media Queries

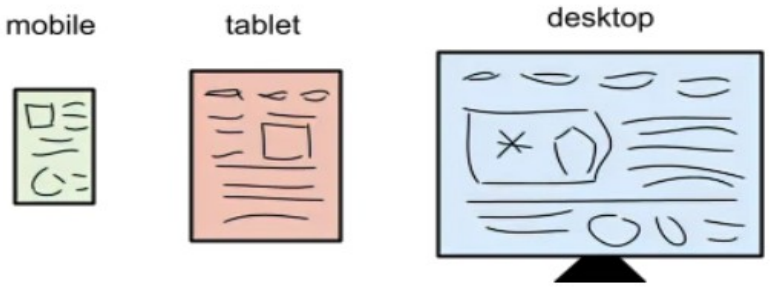
Designing for different screen sizes

Web developers typically design websites to make optimal use of the entire viewport. Viewports vary widely in size, so three platforms developers consider are: desktop, tablet, and mobile. Developers follow two general strategies to design websites for all three platforms:

1. ***Graceful degradation***: Design the desktop website first and modify the design to fit smaller screens.
2. ***Progressive enhancement***: A "mobile first" design methodology that begins with designing the website for the smallest device and then adapts the design for larger screens.

Good practice is to use progressive enhancement to build websites that show equivalent text and images and provide the same functionality on all platforms.

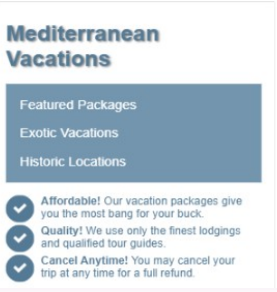
Figure 10.4.1: Designing for three platforms.



Feedback?

~10 minutes.

The provided Mediterranean Vacations webpage is designed for a mobile screen, as shown below. The navigation links near the top to Featured Packages, Exotic Vacations, and Historic Locations all link to sections within the webpage.



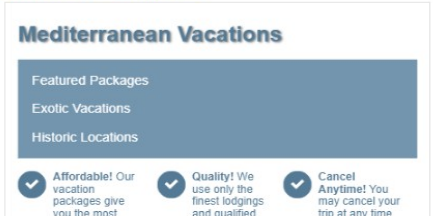
Add the necessary media queries to styles.css so the webpage scales appropriately to larger viewports. Do not modify the page's HTML.

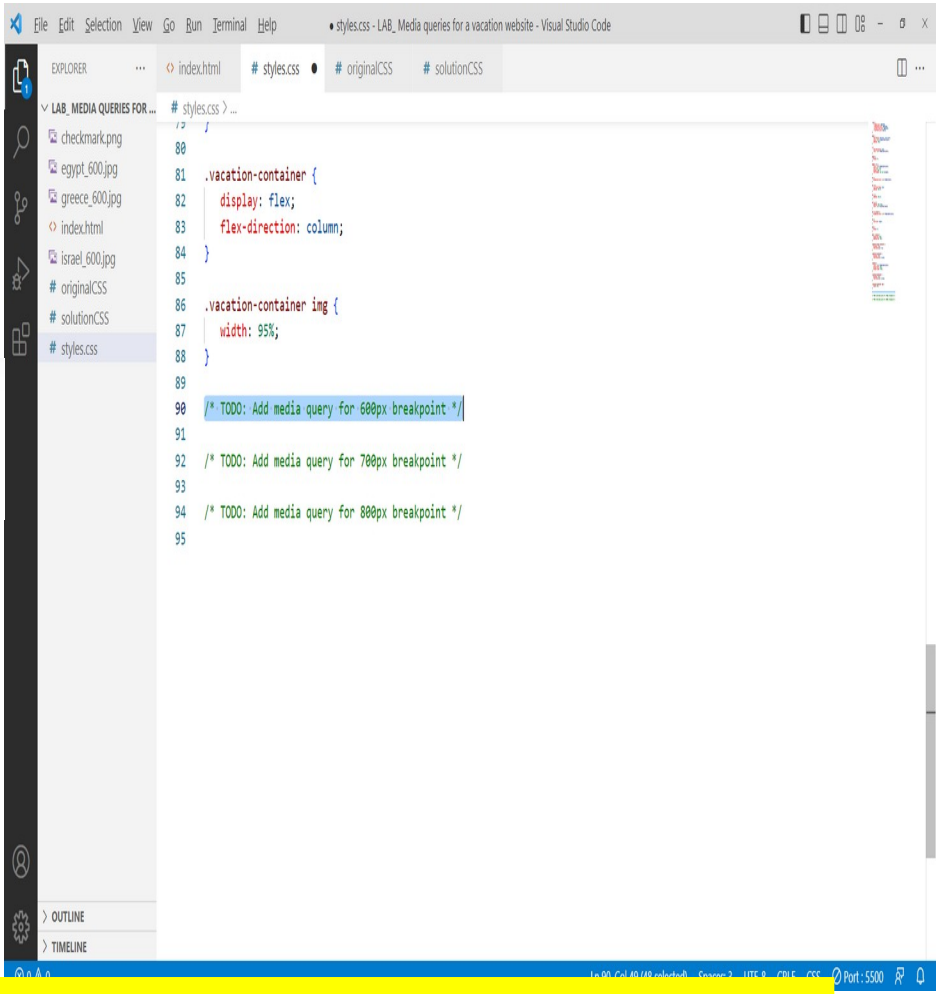
Add the necessary media queries to styles.css so the webpage scales appropriately to larger viewports. Do not modify the page's HTML.

600px breakpoint (3 points)

Add a media query that applies when the viewport's width is at least 600px. Use the `.benefits-container` selector to set `flex-direction: row;`

The screenshot below shows how the benefits change from being displayed in one column to being displayed in one row when the viewport is at least 600px wide.





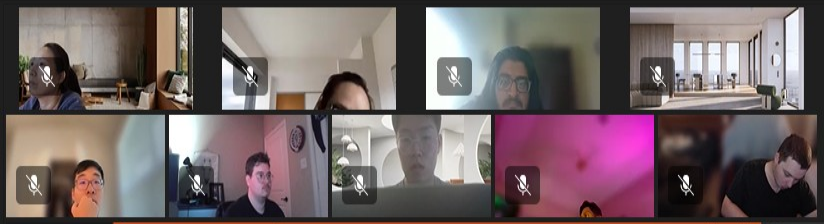
TA time (Kevin)

ZyLab 10.10: Media queries for a vacation website

COSC 4351 - 13236 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

27:07

Take control | Pop out | Chat | People 74 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave



KN

Nguyen, Kevin C

View all

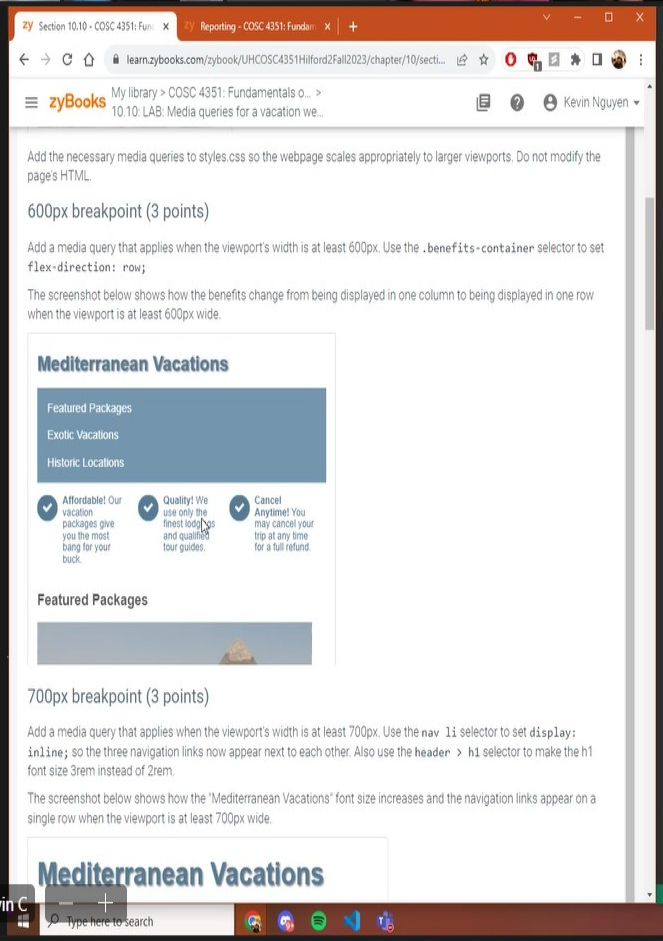
Participants

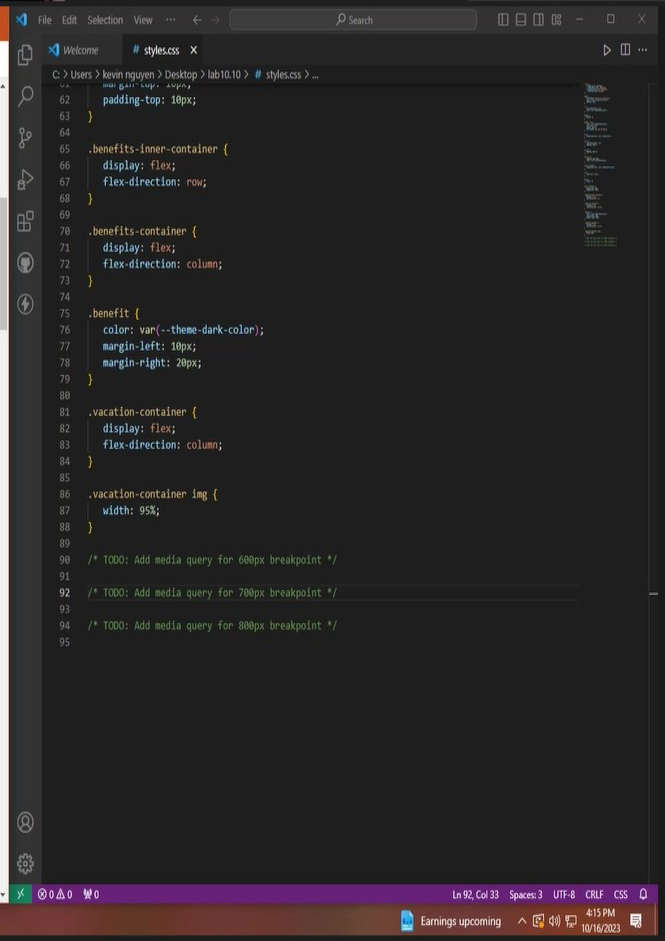
Invite someone or dial a number

Share invite

In this meeting (75) Mute all

- Hilford, Victoria — Organizer
- MA Adesiyun, Monboluwarin E
- IA Agha, Imaan
- SA Ahmed, Sami
- Akter, Sanzida
- JA Alegbe, Joshua A
- JB Batac, Joshua C
- Bonkoungou, Elie D
- WB Brieden, William A
- AC Catchings, Alexander H
- IC Chevez, Iliana G





Nguyen, Kevin C

Type here to search

Earnings upcoming

4:15 PM 10/16/2023

COSC 4351 - 13236 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

29:39

Take control | Pop out | Chat | People 73 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave

Unmute (Ctrl+Shift+M)

KN

Nguyen, Kevin C

View all

Participants

Invite someone or dial a number

Share invite

In this meeting (73) Mute all

- Hilford, Victoria — Organizer
- Adesiyun, Monboluwarin E
- Agha, Imaan
- Ahmed, Sami
- Akter, Sanzida
- Alegbe, Joshua A
- Batac, Joshua C
- Bonkoungou, Elie D
- Brieden, William A
- Catchings, Alexander H
- Chevez, Iliana G

Section 10.10 - COSC 4351: F | Reporting - COSC 4351: Fund | Media Query Lab

File | C:/Users/kevin%20nguyen/Desktop/lab10.10/index.html

Search lab10.10

Featured Packages

Exotic Vacations

Historic Locations

Affordable! Our vacation packages give you the most bang for your buck.

Quality! We use only the finest lodgings and qualified tour guides.

Cancel Anytime! You may cancel your trip at any time for a full refund.

Featured Packages

Egpyt

Egpyt is a land of enduring beauty and rich culture. Explore the capital city of **Cairo**, the **Great Pyramids of Giza**, and the iconic **Great Sphinx**!

Packages include airfare, lodging, and several guided tours!

Exotic Vacations

Nguyen, Kevin C

Type here to search

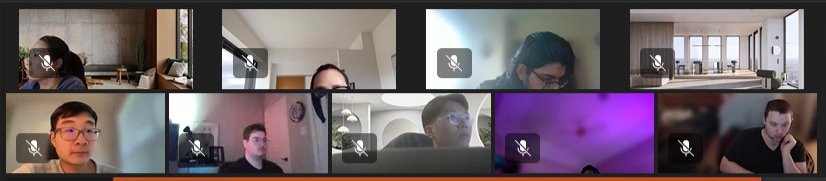
69°F Sunny 4:18 PM 10/16/2023

Type here to search 4:18 PM 10/16/2023

COSC 4351 - 13236 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

31:37

Take control | Pop out | Chat | People 73 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave



KN

Nguyen, Kevin C

View all

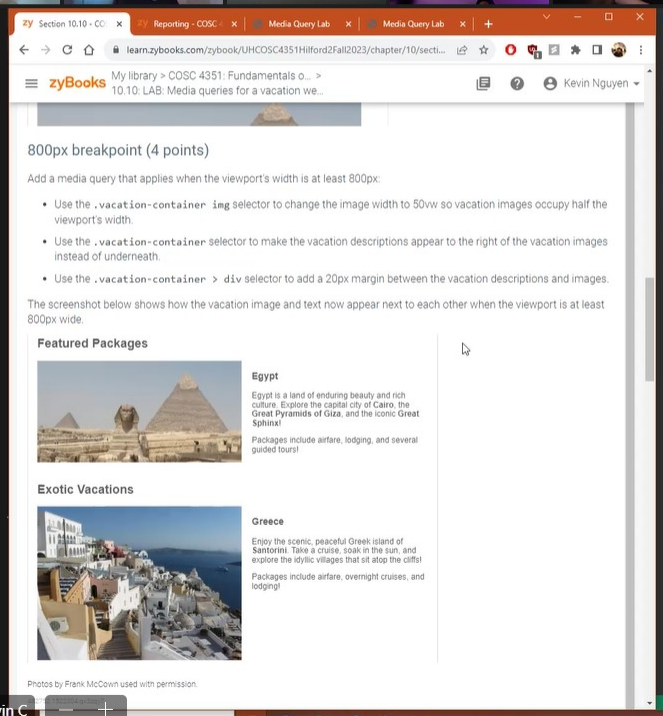
Participants

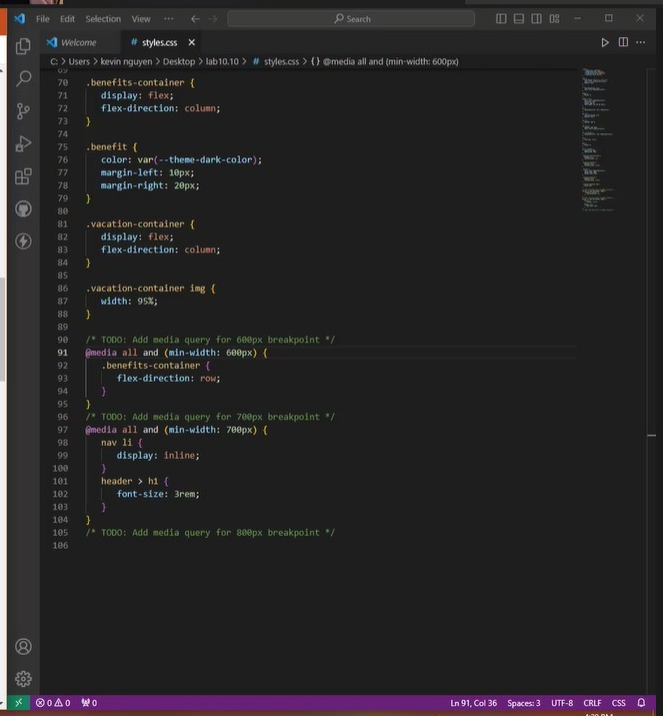
Invite someone or dial a number

Share invite

In this meeting (73) Mute all

- Hilford, Victoria — Organizer
- MA Adesiyun, Monboluwarin E
- IA Agha, Imaan
- SA Ahmed, Sami
- Akter, Sanzida
- JA Alegbe, Joshua A
- JB Batac, Joshua C
- Bonkoungou, Elie D
- WB Brieden, William A
- AC Catchings, Alexander H
- IC Chevez, Iliana G





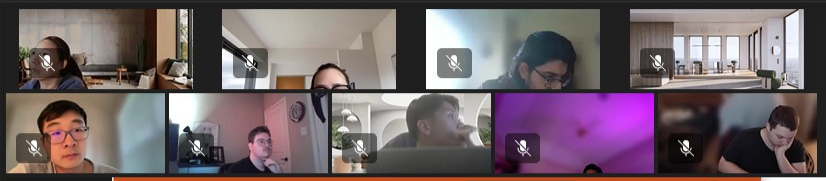
Nguyen, Kevin C

Type here to search | 69°F Sunny | 4:20 PM 10/16/2023

COSC 4351 - 13236 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

31:22

Take control | Pop out | Chat | People 73 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave



KN

Nguyen, Kevin C

View all

Participants

Invite someone or dial a number

Share invite

In this meeting (73) Mute all

Hilford, Victoria
Organizer

Adesiyun, Monboluwarin E

Agha, Imaan

Ahmed, Sami

Akter, Sanzida

Alegbe, Joshua A

Batac, Joshua C

Bonkoungou, Elie D

Brieden, William A

Catchings, Alexander H

Chevez, Iliana G

Section 10.10 - CC | Reporting - COSC 4 | Media Query Lab | Media Query Lab

File | C:/Users/kevin%20nguyen/Desktop/lab10.10/index.html

Mediterranian Vacations

Featured Packages | Exotic Vacations | Historic Locations

Affordable! Our vacation packages give you the most bang for your buck.

Quality! We use only the finest lodgings and qualified tour guides.

Cancel Anytime! You may cancel your trip at any time for a full refund.

Featured Packages

Egpyt

Egpyt is a land of enduring beauty and rich culture. Explore the capital city of **Cairo**, the **Great Pyramids of Giza**, and the iconic **Great Sphinx**!

Packages include airfare, lodging, and several guided tours!

Exotic Vacations

Search lab10.10

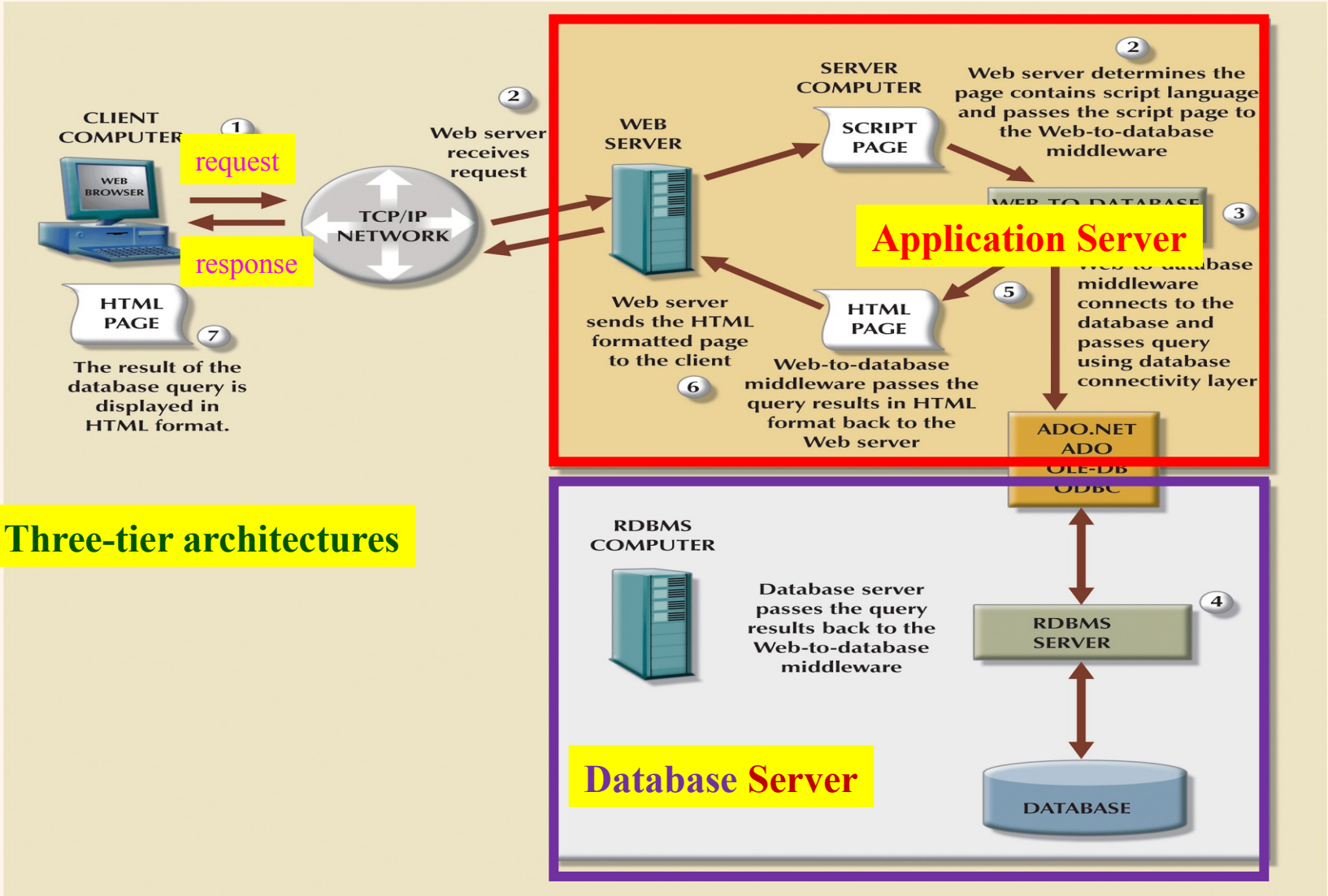
Nguyen, Kevin C

69°F Sunny 4:20 PM 10/16/2023

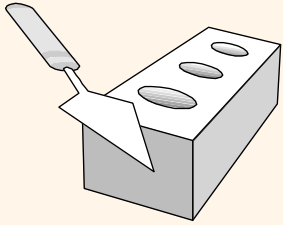
Type here to search 4:19 PM 10/16/2023

```
color);
}

nav li a:hover {
   background-color: var(--
theme-dark-color);
}

nav {
   font-size: larger;
}

main {
   margin: 0;
}

main > section {
   margin-top: 10px;
   padding-top: 10px;
}

.benefits-inner-container {
   display: flex;
   flex-direction: row;
}
```

Web Application Development



Three-tier architectures



3-tier Technologies (MVC)

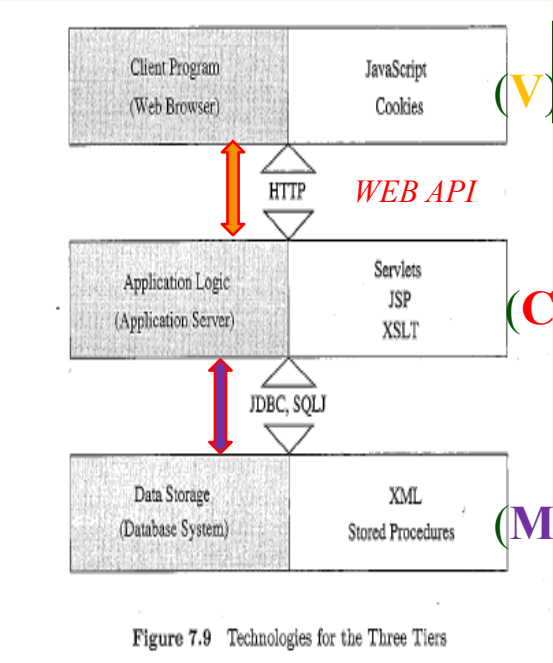


Figure 7.9 Technologies for the Three Tiers

(V) **Client** Program *(Web Browser)*

HTML
XML
Javascript
CSS
XSLT
Cookies

jQuery

(C) Application Server *(Tomcat, Apache, IIS)*

JSP
Servlets
CGI
ASP.NET
PHP

Node.js

(M) Database Server *(DB2)*

Relations/Tables
XML
Stored Procedures

11. Node.js

- [ ] 11. Node.js 2%
  - [ ] 11.1 Full-stack development (Node) 4%
  - [ ] 11.2 Getting started with Node.js 4%
  - [ ] 11.3 Express 4%
  - [ ] 11.4 Pug 4%
  - [ ] 11.5 Relational databases and SQL (Node) 4%
  - [ ] 11.6 MySQL (Node) 0%
  - [ ] 11.7 mysql module (Node) 0%
  - [ ] 11.8 MongoDB 0%
  - [ ] 11.9 Mongoose 0%
  - [ ] 11.10 Creating RESTful web APIs (Node) 4%
  - [ ] 11.11 Using RESTful web APIs (Node) 4%
  - [ ] 11.12 Third-party web APIs (Node) 0%
  - [ ] 11.13 Token-based user authentication (Node) 4%
  - [ ] 11.14 Password hashing (Node) 4%

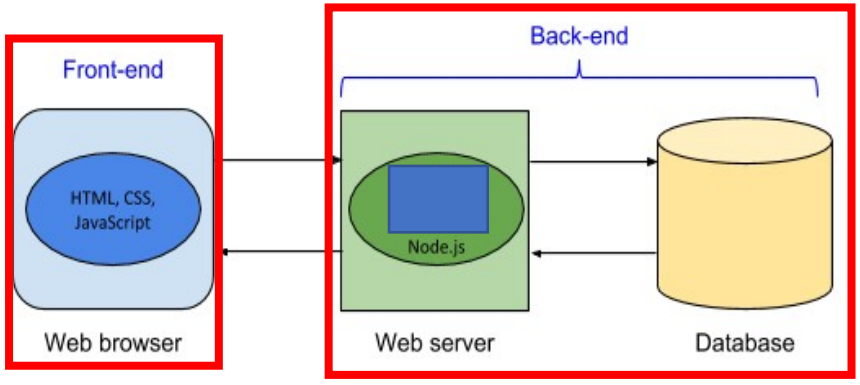
11.1 Full-stack development (Node)

Present Note

Overview of front-end and back-end development

Most websites and web applications require the development of client-side technologies that interact with server-side technologies. ***Client-side*** (or ***front-end***) refers to those technologies that run in the web browser like HTML, CSS, and JavaScript. ***Server-side*** (or ***back-end***) refers to those technologies that run on the web server like PHP, Python, Node.js, etc. and databases. Ex: Amazon uses server-side technologies to store information on millions of products and a client-side search interface that interacts with the web server so customers can find and purchase products.

Figure 11.1.1: Front-end and back-end technologies.



Feedback?

A ***front-end developer*** is a developer that is proficient in client-side technologies. A ***back-end developer*** is a developer that is proficient in server-side technologies. Many developers strive to be proficient in both front-end and back-end technologies and how the two sides work together. A ***full-stack developer*** is a developer who has expertise in all aspects of a website or web application's development, including client technologies, server technologies, data modeling, and user interfaces. The "stack" in "full-stack" refers to the various layers that compose websites and web applications. Technology stacks have increased in complexity over the years, so even "full-stack" developers typically specialize in a few areas of the technology stack.

Server-side programming

Web developers have a wide range of options when choosing a server-side programming platform or language. When choosing a server-side programming platform, developers must consider:

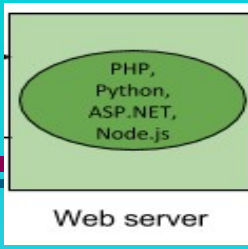
- Server platform: Some web servers support certain languages and not others. Ex: IIS supports ASP.NET, and Apache supports PHP.
- Tool support: Some tools are ideal for working with certain programming languages. Ex: PhpStorm is ideal for PHP development, and Visual Studio is ideal for ASP.NET.
- Developer experience: JavaScript developers may choose Node.js instead of learning a new language like C#. Developers who are new to web development might already know Java or Python and prefer those languages.
- Library support: Some languages may have pre-built libraries that support some web applications better than others.

(C) Application Server *(Tomcat, Apache, IIS)*

JSP
Servlets
CGI
ASP.NET
PHP

Node.js

Java

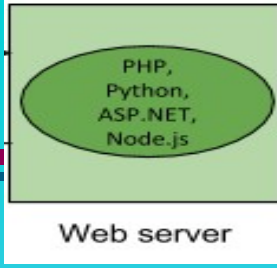


	Used to create applets on the front-end and servlets, JavaServer Pages, and web APIs on the back-end.

Java has been used for web development since the late 1990s. Java is still a popular server-side language, but most developers now use Java primarily for creating web APIs.

Your Answer?

PHP

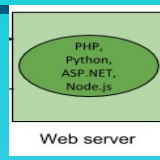


	Scripting language created in 1994 by Rasmus Lerdorf. Currently the most popular server-side language in use.

Facebook was originally created with PHP, and PHP remains one of the easiest server-side languages to learn.

Your Answer?

Python

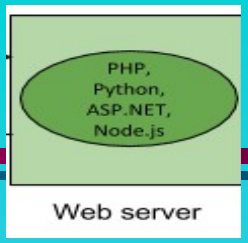


	General-purpose scripting language created by Guido van Rossum in the 1990s that uses frameworks like Django, web2py, and Flask to create web applications.

Python is **Google**'s language of choice. Python uses an easy-to-learn syntax without { curly braces }.

Your Answer?

Node.js

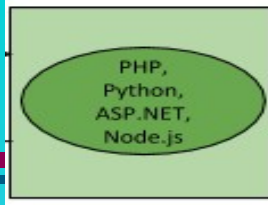


Runtime environment that uses modules written in JavaScript. Originally created in 2009 by Ryan Dahl.

Node.js is a relative new-comer, but many notable companies like **Walmart** and **LinkedIn** spout Node's ability to scale-up efficiently.

Your Answer?

Ruby on Rails

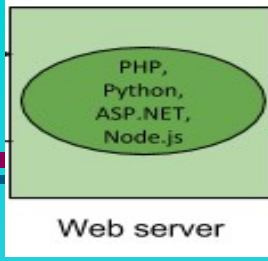


	Web application framework written in Ruby and created by David Heinemeier Hansson in 2004.

Twitter was originally created with Rails. Rails emphasizes Convention over Configuration (CoC), meaning developers are encouraged to write code in a manner that follows a specific convention.

Your Answer?

ASP.NET

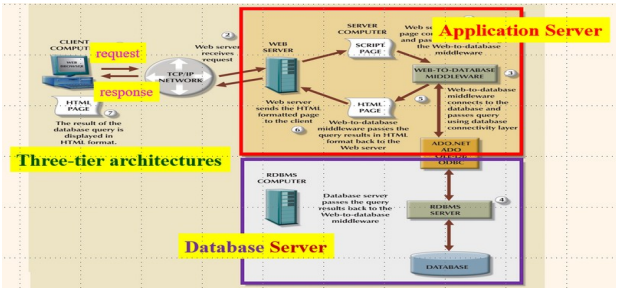


	Collection of web development technologies first released in 2002 by Microsoft that uses the C# or VB.NET programming languages.

ASP.NET is a powerful platform generally used on **Windows** servers.

Your Answer?

dynamic web page



Developers have traditionally used server-side technologies to generate dynamic web pages. A ***dynamic web page*** is a web page that is generated on the web server when requested, typically personalized to the user who requested the page. With advances in web browsers, developers have begun creating static web pages that are dynamically altered by JavaScript. In this new paradigm, server-side technologies are used primarily to respond to Ajax requests and send data to the front-end for rendering.

(C) Application Server
(Tomcat, Apache, IIS)

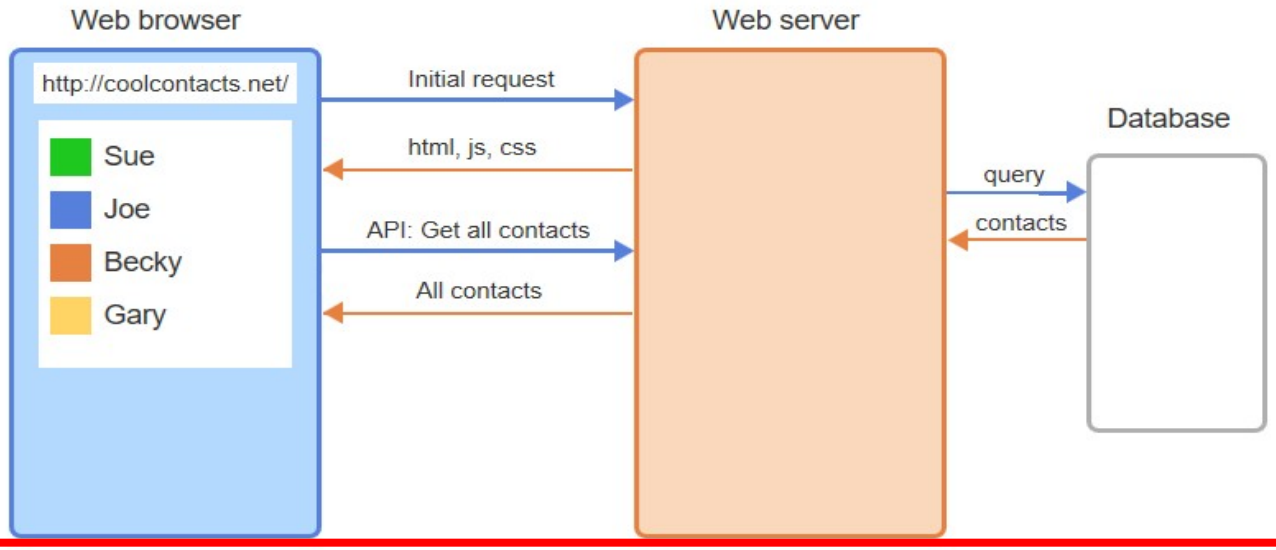
JSP
Servlets
CGI
ASP.NET
PHP

(M) Database Server
(DB2)

Relations/Tables
XML
Stored Procedures

Single Page Application (SPA)

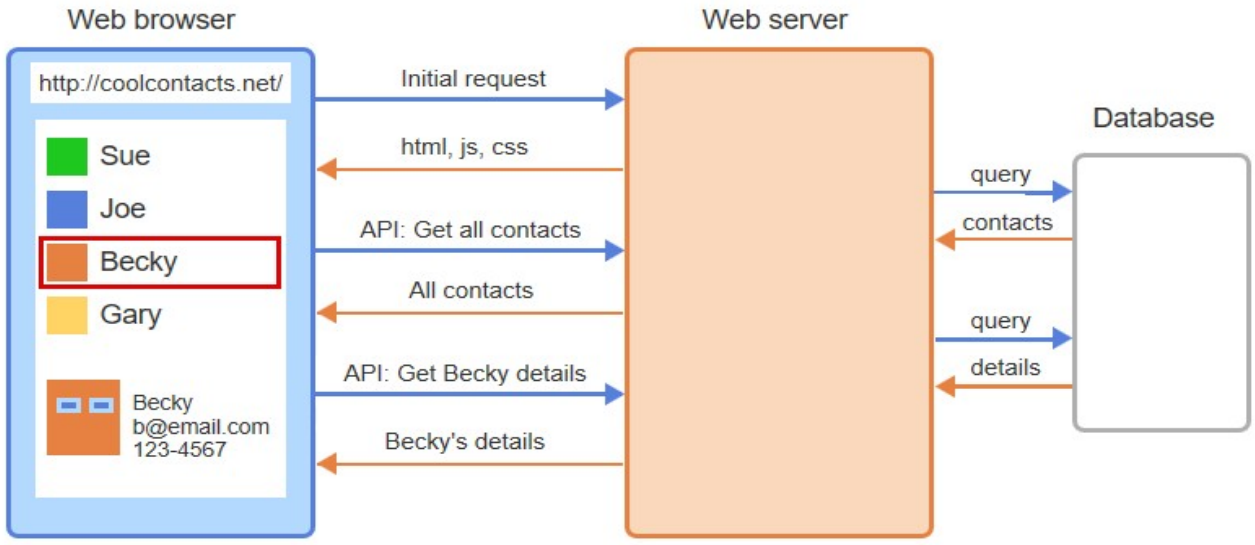
Single Page Applications are an example of modern web development. A ***Single Page Application (SPA)*** is a web application that provides a similar user experience as a desktop application, all in a single web page. Ex: Gmail, Google Docs, and Google Calendar are all SPAs. An SPA initially loads all of the application's resources so subsequent user interaction results in loading small pieces of content dynamically. Much of an SPA's programming logic is written in JavaScript, which loads data via Ajax calls to a web API. A ***web API*** is a collection of functions that are invoked using HTTP. Ex: An HTTP GET request to the URL http://linkedin.com/api/contacts may retrieve a list of all contacts from the web server.



All contacts are retrieved from the database and sent back to the web app for displaying.

Single Page Application (SPA)

1. Initial request for CoolContacts web app sent to the web server.
2. All resources needed for app are downloaded in multiple request-response messages.
3. JavaScript uses web API to request all contacts.
4. All contacts are retrieved from the database and sent back to the web app for displaying.
5. User selects a contact from the web app.
6. JavaScript uses web API to request details for selected contact.
7. Web server requests Becky's details from the database and sends the details back to the web app for displaying.

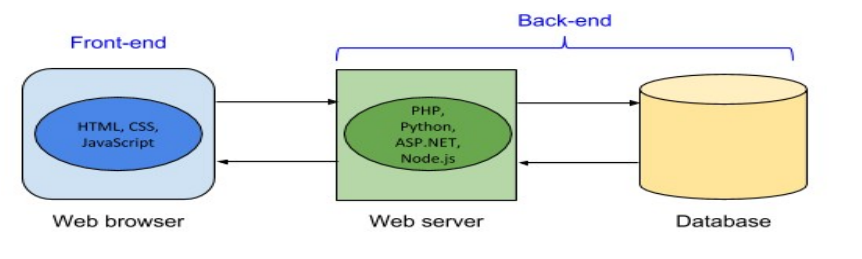


Server-side programming platforms.

A dynamic web page might look different for two different users who are accessing the same page.

- (●) True
- ( ) False

Dynamic web pages are often personalized to the user who requested the page.



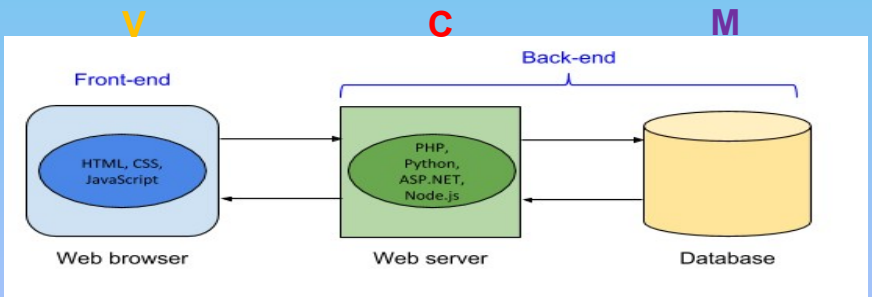
Your Answer?

The business logic of an SPA should generally be encoded in the front-end.

- ○ True
- ◉ False

MVC Design Pattern

A well-designed SPA implements as little business logic in the front-end as possible. The web API should implement the business logic.

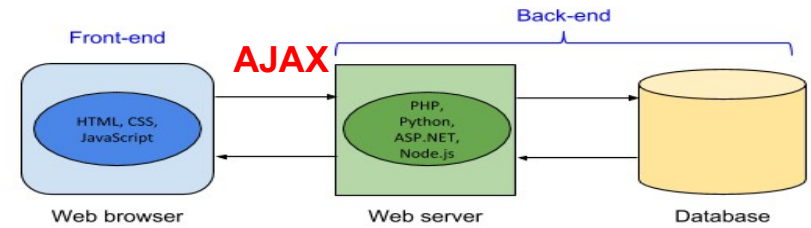


Your Answer?

SPAs generally result in less data being sent over the network than web applications developed with dynamically generated web pages.

- (●) True
- ( ) False

SPAs use web APIs to pass data between the web browser and web server, which are generally more network efficient.



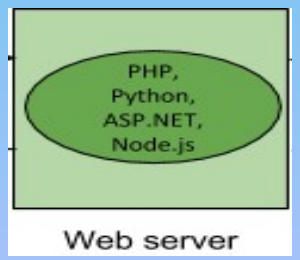
Your Answer?

Server-side programming platforms.

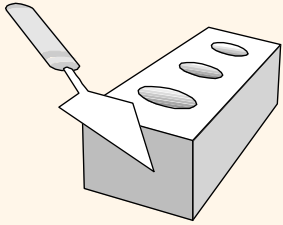
Developers use ASP.NET, Java, PHP, Python, Node.js, and Ruby on Rails to create web APIs.

- (●) True
- ( ) False

All server-side programming platforms or languages have libraries and frameworks that allow developers to create web APIs.



Your Answer?



3-tier Technologies (MVC)

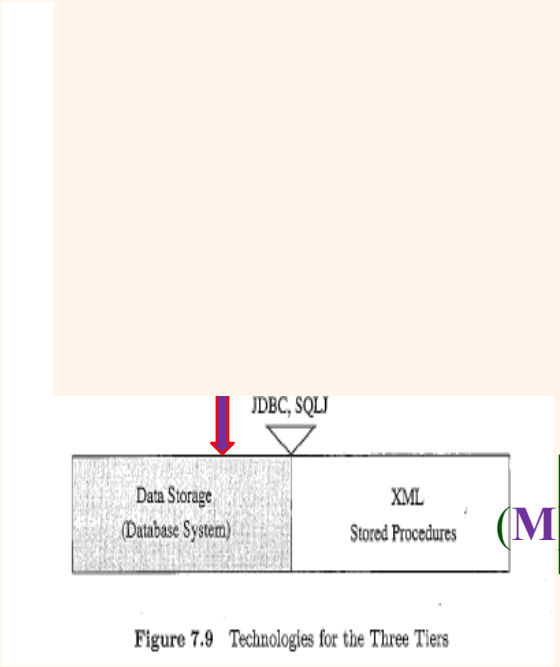


Figure 7.9 Technologies for the Three Tiers

(M)

Database Server
(DB2)

Relations/Tables

XML

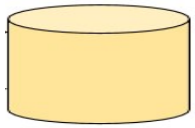
Stored Procedures

Databases

Websites and web applications normally store and retrieve information from a database and have traditionally used relational databases. A ***relational database*** stores data in relations (usually called tables). The ***Structured Query Language (SQL)*** is a language for creating, editing, selecting, and deleting data in a relational database. Popular relational database management systems (RDBMS) include MySQL, PostgreSQL, Oracle, and SQL Server.

Non-relational databases, sometimes called ~~***non-SQL***~~ or ~~***NoSQL***~~ databases, have become increasingly popular over the last few years. Non-relational databases use different methods to store and retrieve data using a variety of data access languages. Non-relational databases come in several flavors:

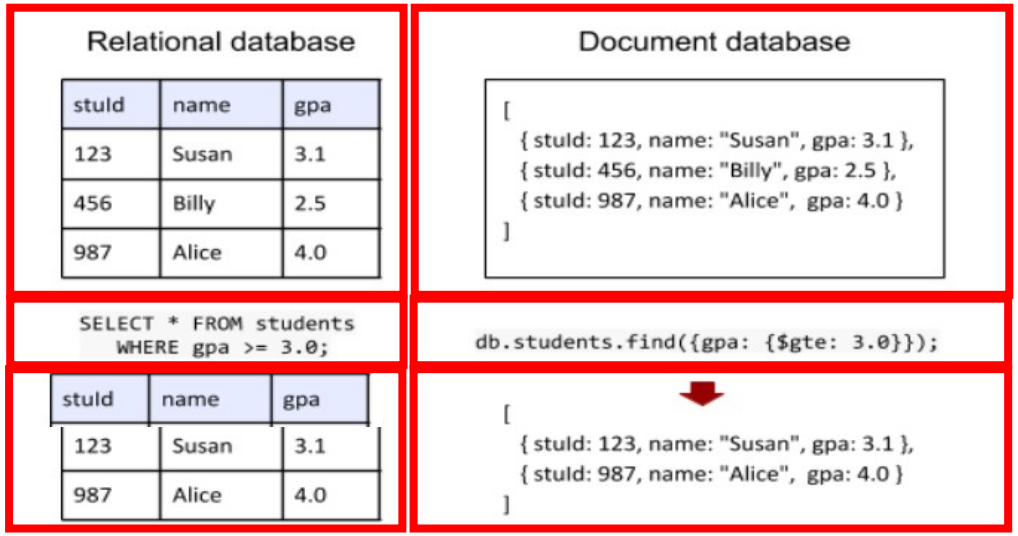
- Document database: For storing documents in JSON format with many levels of nesting. Ex: MongoDB.
- Key-value database. For storing values that are associated with unique keys. Ex: Redis.
- Object database: For storing objects created in object-oriented programming languages. Ex: Caché.
- Column database: For storing and processing large amounts of data using pointers that link to columns distributed over a cluster. Ex: HBase.
- Graph database: For storing graph structures with nodes and edges. Ex: Neo4j.



Databases

The figure below illustrates how information about students might be stored in a relational database with a table and in a document-oriented database using JSON-like documents. The "SELECT" statement is an SQL statement used to extract students with a 3.0 GPA or above from the table. The "db.students.find" statement is a MongoDB function used to extract the same information from the document database.

Figure 11.1.2: Relational model and document model for student data.



???

Relational databases will likely not be used for many web applications in the future.

- ○ True
- ◉ False

The popularity of relational databases and relative ease of using SQL ensures that relational databases will not go away soon.

Your Answer?

A relational database can be used to store documents, objects, graphs, and key-value pairs.

- (●) True
- ( ) False

Relational databases are very flexible and can store a wide range of data. Some developers prefer non-relational databases for documents, objects, graphs, and key-value pairs because the programming required to manipulate such data is often decreased, and in some cases non-relational databases are faster.

Your Answer?

Column databases are generally faster than relational databases accessing vast amounts of data.

- (●) True
- ( ) False

Column databases excel at certain types of queries, such as finding aggregations like totals and averages. However, column databases are slow when inserting a single row.

Your Answer?

Both relational and non-relational databases have been implemented with open source software.

- (●) True
- ( ) False

MySQL and PostgreSQL are popular open-source relational databases. MongoDB and Cassandra are popular open-source non-relational databases.

Your Answer?

Client-side technologies

The user interface (UI) governs the interaction between users and web applications. Developers use HTML, CSS, and JavaScript to create the UI. Various tools exist to aid UI development:

- HTML preprocessors: An ***HTML preprocessor*** is a program that converts a markup language into HTML. The markup languages supported by HTML preprocessors are generally easier to use and read than HTML. Ex: Haml, Markdown, Slim, Pug.
- CSS preprocessors: A ***CSS preprocessor*** is a program that converts a CSS-like language into CSS. CSS-like languages simplify the development of CSS stylesheets used in large projects. Ex: Sass, Less, Stylus.
- UI libraries: Libraries that create UI widgets like sliders, dialog boxes, and drop-downs or simplify DOM manipulation. Ex: jQuery UI, Bootstrap, YUI, Ext JS.
- CSS front-end frameworks: A ***CSS front-end framework*** is a framework that uses CSS or CSS pre-processors to aid in developing responsive websites that work well on every screen size. Ex: Bootstrap, YAML 4, Skeleton, Foundation.

Most modern web applications use an extensive amount of JavaScript, so developers use various tools to aid in JavaScript development

- Compile-to-JavaScript languages: A ***compile-to-JavaScript language*** is a programming language that is compiled into JavaScript. Compile-to-JavaScript languages provide benefits lacking in JavaScript like type safety, simplified class creation, and module creation. Ex: TypeScript, CoffeeScript, Haxe, Dart.
- JavaScript frameworks: A ***JavaScript framework*** is a JavaScript environment that dictates the organization of the application's JavaScript to simplify many programming tasks. JavaScript frameworks often dictate how UI widgets receive data or send data to the web server. Ex: AngularJS, Backbone, Ember.

Client-side technologies
Example use of HTML and CSS preprocessors and compile-to-JavaScript.

Haml

```
nav
   ul
      li
         a href='/home' Home
      li
         a href='/about' About
      li
         a href='/sales' Sales
```

Resulting HTML

```
<nav>
  <ul>
    <li><a href="html">Home</a></li>
    <li><a href="about">About</a></li>
    <li><a href="sales">Sales</a></li>
  </ul>
</nav>
```

Less

```
@nice-green: #097911;
@light-green: @nice-green + #111;

header {
   color: @light-green;
   .logo {
      width: 250px;
   }
}
```

Resulting CSS

```
header {
   color: #1a8a22;
}

header .logo {
   width: 250px;
}
```

CoffeeScript

```
math =
  root:   Math.sqrt
  square: (x) -> x * x
```

Resulting JavaScript

```
math = {
  root: Math.sqrt,
  square: function(x) {
     return x * x;
  }
};
```

CSS preprocessors allow developers to write much less code compared to writing straight CSS.

- ○ True
- ◉ False

CSS preprocessors convert special syntax into more verbose CSS rules.

Less	Resulting CSS

`@nice-green: #097911;`
`@light-green: @nice-green + #111;`
 	
`header {`
`  color: @light-green;`
`  .logo {`
`  width: 250px;`
`  }`
`}`
	`header {`
	`  color: #1a8a22;`
	`}`
	
	`header .logo {`
	`  width: 250px;`
	`}`

Your Answer?

UI libraries always use JavaScript to govern the behavior of the UI widgets.

- ○ True
- ◉ False

Some widgets require JavaScript, but many use special HTML tags to govern the widget behavior.

UI libraries: Libraries that create UI widgets like sliders, dialog boxes, and drop-downs or simplify DOM manipulation. Ex: jQuery UI, Bootstrap, YUI, Ext JS.

Your Answer?

CSS front-end frameworks are required to build responsive websites that work well on mobile devices.

- ○ True
- ◉ False

CSS front-end frameworks simplify the creation of responsive websites, but some developers write their own CSS code.

CSS front-end frameworks: A ***CSS front-end framework*** is a framework that uses CSS or CSS pre-processors to aid in developing responsive websites that work well on every screen size. Ex: Bootstrap, YAML 4, Skeleton, Foundation.

Your Answer?

JavaScript frameworks often simplify the use of web APIs in the browser.

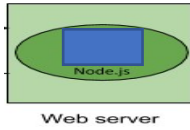
- (●) True
- ( ) False

JavaScript frameworks like AngularJS often provide built-in functionality for working with web APIs.

JavaScript frameworks: A ***JavaScript framework*** is a JavaScript environment that dictates the organization of the application's JavaScript to simplify many programming tasks. JavaScript frameworks often dictate how UI widgets receive data or send data to the web server. Ex: AngularJS, Backbone, Ember.

Your Answer?

Getting Started with Node.js



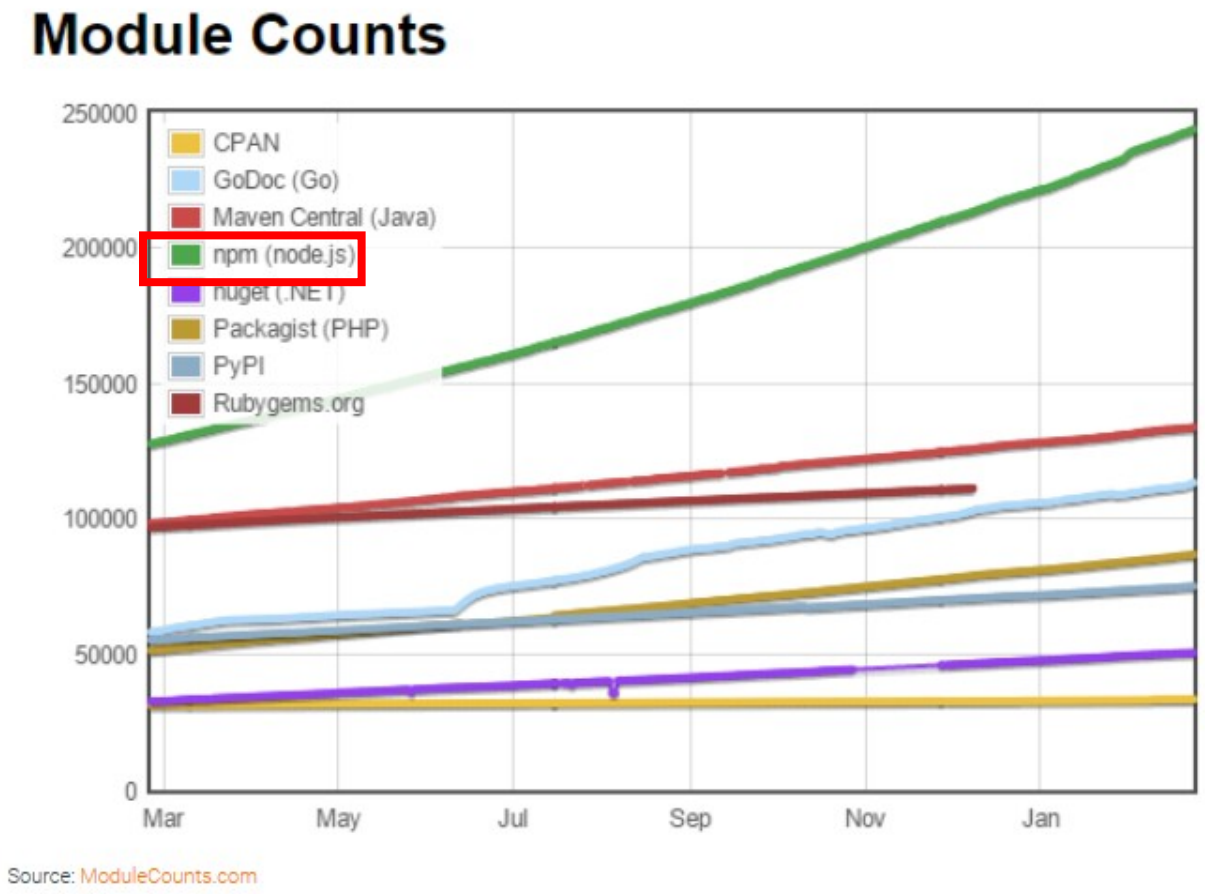
Node.js is a JavaScript runtime environment that is primarily used to run server-side web applications. Node.js has many benefits:

- The event-driven, non-blocking I/O architecture of Node.js allows Node.js to handle high loads.
- Node.js allows developers to write JavaScript on the server and client, simplifying some development tasks.
- Node.js provides a simple mechanism to create and distribute modules. A ***Node.js module*** is a self-contained collection of JavaScript code.
- Node.js works seamlessly with MongoDB, a document database that stores JSON and uses JavaScript as a query language. Web development is greatly simplified when JSON is used between the client and server, and between the server and database.

Companies using Node.js include Netflix, Walmart, Ebay, and LinkedIn. Adoption by these companies helped validate the Node.js approach and spur development of more Node.js packages.

node

Introduction to Node.js
Number of **modules** for the Node.js package manager (npm) far exceeds other languages.

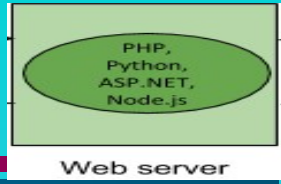


Node.js programs are always written in JavaScript.

- (●) True
- ( ) False

Node.js uses Google's V8 JavaScript engine to run JavaScript programs.

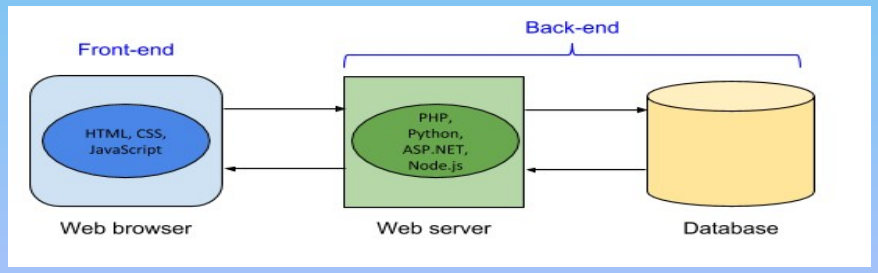
Your Answer?



Node.js programs run in the web browser.

- ○ True
- ◉ False

Node.js programs run on the web server.



Your Answer?

Introduction to Node.js

Developers may install Node.js using installers from the Node.js website for Windows, Mac OS X, and other operating systems. After installing Node.js, a developer can start the Node.js interactive shell and execute JavaScript statements. The figure below shows a command line prompt, from which the user started the Node.js interactive shell by entering "node".

Figure 11.2.2: Node.js interactive shell.

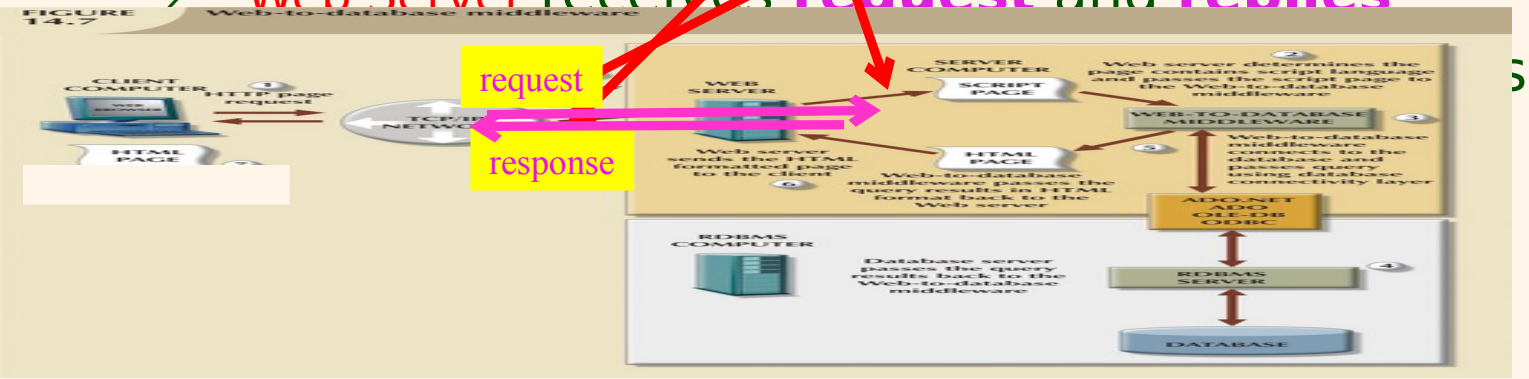
```
$ node
> console.log("Hello, Node.js!")
Hello, Node.js!
undefined
> x = 2
2
```

HyperText Transfer Protocol

- What is a communication protocol?
  - Set of **standards** that defines the structure of **messages**
  - Examples: TCP, IP, HTTP, HTTPS

- What happens if you click on www.cs.wisc.edu/~dbbook/index.html?

1. **Client** (web browser) sends HTTP **request** to Web Server
2. Web Server receives **request** and **replies**



s new

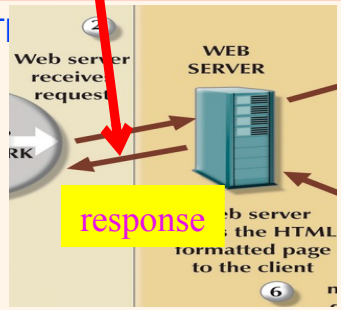
HTTP

Client request to Web Server:

GET ~/**index.html** HTTP/1.1
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg

Web Server replies **response** to **Client**:

HTTP/1.1 200 OK
Date: Wed, 26 Feb 2014 16:00:00 GMT
Server: Apache/1.3.0 (Linux)
Last-Modified: Wed, 04 Oct 2006 16:30:24 GMT
Content-Length: 1024
Content-Type: text/html
<HTML>
<HEAD></HEAD>
<BODY>
<h1>Barns and Nobble Internet Bookstore</h1>
Our inventory:
<h3>Science</h3>
The Character of Physical Law
...
</HTML>



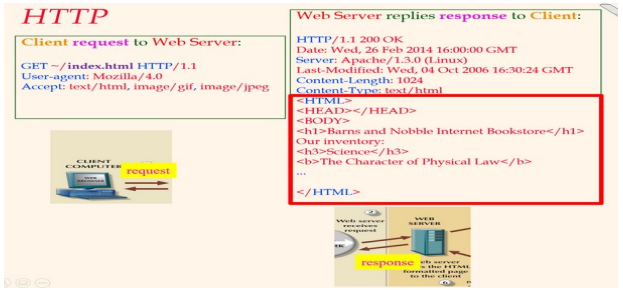
Introduction to Node.js

The ***http module*** allows a Node.js application to create a simple web server.

```
$ node server.js
```

```
// server.js
var http = require("http");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write("<h1>Hello, Node.js!</h1>");
    response.end();
}).listen(3000);
```

http



1. server.js is executed from the command line.
2. The require("http") command imports the "http" module.
3. createServer() creates a web server object that calls the provided callback function when an HTTP request is received.
4. listen() starts the web server listening on port 3000 for HTTP requests.
5. The user enters a URL to access the web server running on the same machine on port 3000.
6. The HTTP request is routed to the web server, causing the request callback function to execute.
7. writeHead() creates an HTTP response with a 200 status code and text/html content type.
8. write() sends the HTML to the HTTP response object.
9. end() sends the HTTP response to the web browser, which renders the HTML.

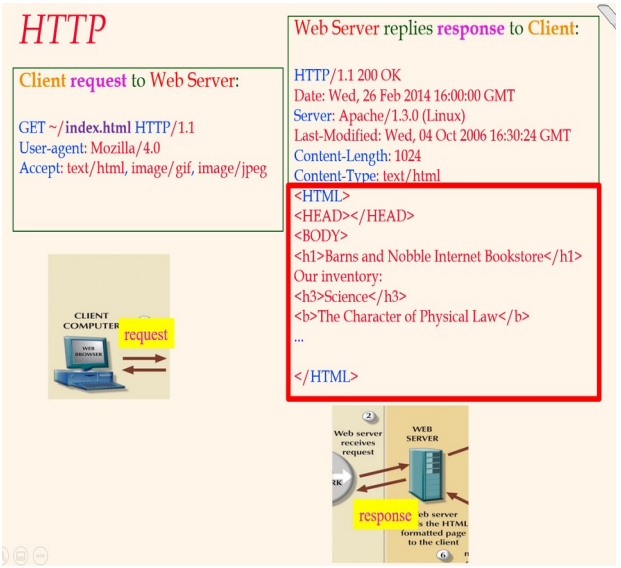
node Web Server

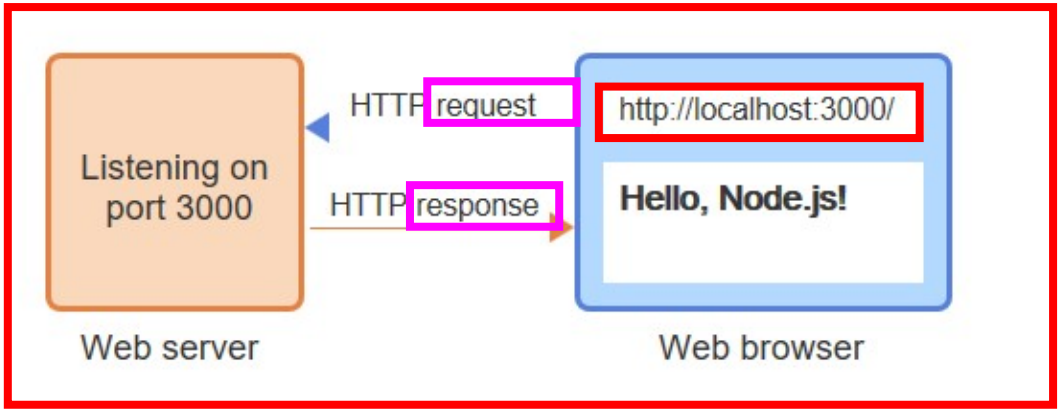
Introduction to Node.js

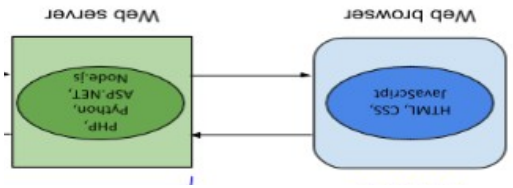
The ***http module*** allows a Node.js application to create a simple web server.

```
$ node server.js
```

```
// server.js
var http = require("http");

http.createServer(function(request, response) {
   response.writeHead(200, {"Content-Type": "text/html"});
   response.write("<h1>Hello, Node.js!</h1>");
   response.end();
}).listen(3000);
```





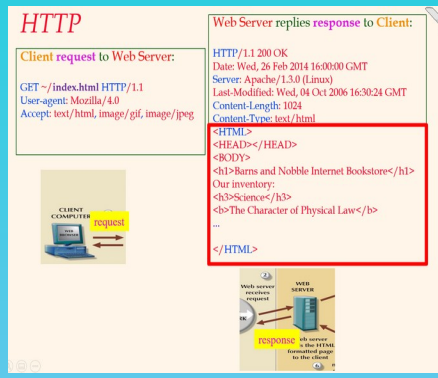


node Web Server

Node.js web server.

What method causes the web server to begin listening for HTTP requests?

- require()
- createServer()
- **listen()**



listen() starts the web server, after which the web server can respond to incoming requests.

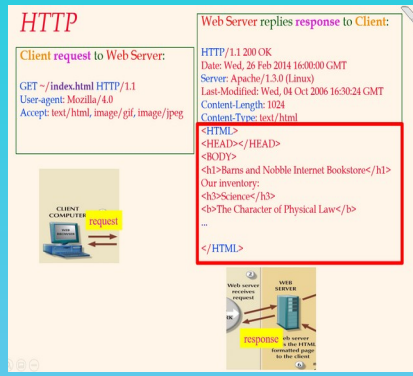
```
// server.js
var http = require("http");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write("<h1>Hello, Node.js!</h1>");
    response.end();
}).listen(3000);
```


Your Answer?

Node.js web server.

What URL accesses a web server running locally and listening on port 8080?

- http://localhost/
- http://localhost:8080/
- http://8080/



localhost indicates the web server is running on the same machine as the web browser. ":8080" indicates the port to which HTTP requests are routed.

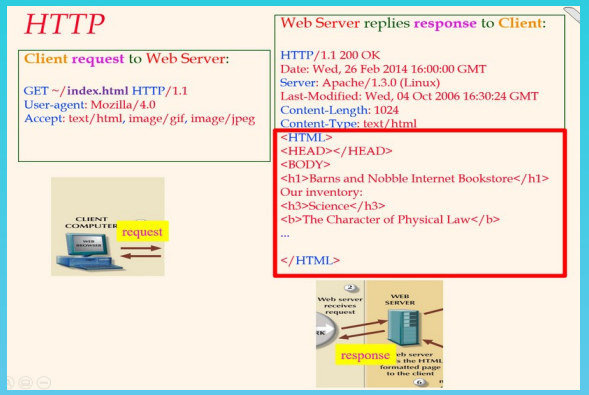
```
// server.js
var http = require("http");

http.createServer(function(request, response) {
   response.writeHead(200, {"Content-Type": "text/html"});
   response.write("<h1>Hello, Node.js!</h1>");
   response.end();
}).listen(3000);
```


Your Answer?

Node.js web server.

What method sets the status code for the HTTP response?

- (●) response.writeHead()
- ( ) response.write()
- ( ) response.end()



writeHead() sets the status code and HTTP headers on the HTTP response object.

```
// server.js
var http = require("http");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write("<h1>Hello, Node.js!</h1>");
    response.end();
}).listen(3000);
```

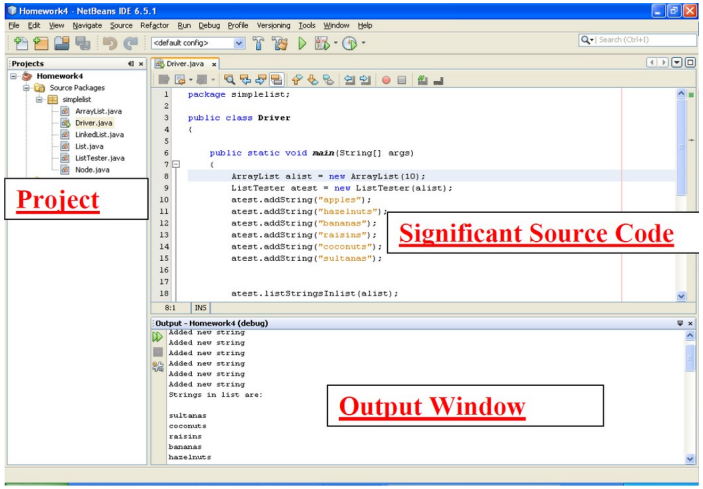

Your Answer?

Node.js project

Node.js programs are typically organized into projects. A ***Node.js project*** is a collection of JavaScript files, packages, configuration files, and other miscellaneous files that are stored in a directory.

Figure 11.2.4: Example Node.js project with a single JavaScript file.

```
myproject
└── server.js
```



IDE

node project structure

Node Package Manager (npm)

The ***Node Package Manager (npm)*** is the package manager for Node.js that allows developers to install and update packaged modules. npm is installed with Node.js and is executed from the command line.

Figure 11.2.5: Display npm's version.

```
$ npm -v
3.3.12
```

Version

npm can install packages in one of two modes:

Install

- Local mode: Packages are installed in a **node_modules** directory in the parent working directory. Ex: **npm install mypackage**
- Global mode: Packages are installed in a **{prefix}/node_modules** directory, where **{prefix}** is a location set in npm's configuration. The **--global** flag (or **-g**) directs npm to install in global mode. Ex: **npm install mypackage --global**

npm (node project manager) commands Summary

Table 11.2.1: Summary of npm commands.

Command	Description	Example
`config`	Manage npm configuration files	`npm config list` `npm config get prefix`
`install`	Install package locally or globally (-g)	`npm install nodemon -g`
`list`	List all installed local or global (-g) packages	`npm list`
`update`	Update a local or global (-g) package	`npm update lodash`
`uninstall`	Uninstall a local or global (-g) package	`npm uninstall lodash`

node package manager (npm) prefix

Local mode is ideal for installing project dependencies. A ***dependency*** is a package that a Node.js project must be able to access to run.
Global mode is typically for installing command-line tools.

Figure 11.2.6: Get npm's prefix directory where global packages are installed.

```
$ npm config get prefix
/usr/local
```

The prefix directory will be different for Windows users.

`config`	Manage npm configuration files	`npm config list` `npm config get prefix`

config

nodemon module

The figure below shows a developer installing the nodemon package globally. The ***nodemon module*** saves developers time by restarting a Node.js application whenever the files in the directory nodemon is started in are modified.

Figure 11.2.7: Installing "nodemon" as a global package.

```
$ npm install nodemon --global
/myproject
└─┬ nodemon@1.9.1
  ├─┬ chokidar@1.4.2
  │ └─┬ anymatch@1.3.0
etc...

$ nodemon server.js
[nodemon] 1.9.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
```

`install`	Install package locally or globally (-g)	`npm install nodemon -g`

nodemon – restart automatically

underscore module

The ***underscore module*** is a library of helpful functions that extends some built-in JavaScript objects. The figure below shows a developer changing to the `myproject` directory that stores a Node.js project, and then installing underscore as a local package and producing a list of all local packages. The underscore package is stored in `myproject/node_modules/underscore`.

Figure 11.2.8: Installing "underscore" as a local package.

```
$ cd myproject
$ npm install underscore
/myproject
└── underscore@1.8.3

$ npm list
/myproject
└── underscore@1.8.3
```

`install`	Install package locally or globally (-g)	`npm install nodemon -g`
`list`	List all installed local or global (-g) packages	`npm list`

underscore – get random numbers

require statement

The **require** statement imports the package so the package may be used in the Node.js program. Good practice is to assign imported packages to variables that are named similar to the package name. Ex: Variable **http** for the package "http". The figure below illustrates how underscore is used to produce a web page that displays 5 random dice rolls.

Figure 11.2.9: Using the "underscore" package to get random dice rolls

```
var http = require("http");
var _ = require("underscore");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write("<!DOCTYPE html>\n");
    response.write("<title>Dice Roll</title>\n");
    response.write("<body>\n");

    for (var i = 0; i < 5; i++) {
        // Use underscore to get a random number between 1 and 6
        var randNum = _.random(1, 6);

        response.write("<p>" + randNum + "</p>\n");
    }
    response.write("</body>\n</html>");
    response.end();
}).listen(3000);
```

```
<!DOCTYPE html>
<title>Dice Roll</title>
<body>
<p>3</p>
<p>5</p>
<p>5</p>
<p>6</p>
<p>1</p>
</body>
</html>
```

require === import

node package manager (npm) prefix

Where does the npm command below install the grunt package?

```
$ npm install grunt -g
```

- ○ The project's **node_modules** directory
- ◉ **{prefix}/node_modules** directory
- ○ **{prefix}** directory

All modules installed with -g or --global are installed globally in the same directory.

Your Answer?

node package manager (npm) prefix

Which command displays all the installed global npm packages?

- ○ npm install --global
- ○ npm list
- ◉ npm list --global

The list command with --global flag lists all globally installed packages.

Your Answer?

node package manager (npm) prefix

Which command updates the local mkdirp package?

- npm install mkdirp
- npm update mkdirp
- npm update mkdirp -g

mkdirp

The update command replaces the existing module with the most up-to-date version.

`update`	Update a local or global (-g) package	`npm update lodash`

Your Answer?

node package manager (npm) prefix

Which command uninstalls the local mkdirp package?

- npm update mkdirp
- npm uninstall mkdirp -g
- npm uninstall mkdirp

The uninstall command removes the local package from the project's node_modules directory.

`uninstall`	Uninstall a local or global (-g) package	`npm uninstall lodash`

Your Answer?

package.json file

Node.js projects use **`package.json`** to list information about the project. The ***package.json*** file contains JSON that lists the Node.js project's name, version, license, dependencies, and other information. Developers can manually create **`package.json`** or use **`npm init`**.

Figure 11.2.10: Example package.json file.

```
{
  "name": "my-web-server",
  "version": "0.0.1",
  "description": "A simple web server",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "underscore": "1.8.x"
  },
  "author": "",
  "license": "ISC"
}
```

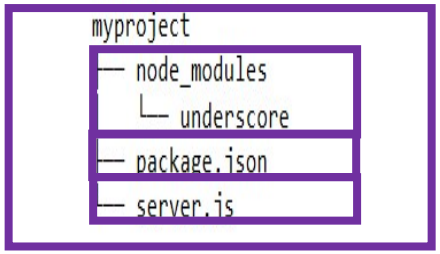
JSON – JavaScript Object Notation

package.json – project info

package.json file

When a project's **package.json** file is present, then all the project's dependencies can be installed with a single command: **npm install**.

Figure 11.2.11 Files composing Node.js project.



JSON – JavaScript Object Notation

package.json – project info helps install

Node.js project's package.json file.

A package.json file may list the project developers, homepage, and bugs.

- (●) True
- ( ) False

Much of the project's information may be included in a project's package.json.

```
{
  "name": "my-web-server",
  "version": "0.0.1",
  "description": "A simple web server",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "underscore": "1.8.x"
  },
  "author": "",
  "license": "ISC"
}
```


Your Answer?

node package manager (npm) prefix

Packages installed from npm occasionally have package.json files.

- ○ True
- ◉ False

npm requires packages to have a package.json file.

```
{
  "name": "my-web-server",
  "version": "0.0.1",
  "description": "A simple web server",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "underscore": "1.8.x"
  },
  "author": "",
  "license": "ISC"
}
```


Your Answer?

Node.js project's package.json file.

```
{
  "name": "my-web-server",
  "version": "0.0.1",
  "description": "A simple web server",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "underscore": "1.8.x"
  },
  "author": "",
  "license": "ISC"
}
```

The "scripts" value in the example package.json above allows the web server to be started with the command:

```
node start
```

- (●) True
- ( ) False

The **node start** command executes the instruction: **node server.js**.

Your Answer?

Node.js project's package.json file.

The following command installs the mkdirp module and adds mkdirp to the **<dependencies>** block of package.json:

```
npm install mkdirp --save
```

- (●) True
- ( ) False

save

The --save flag saves the module as a project dependency.

```
{
  "name": "my-web-server",
  "version": "0.0.1",
  "description": "A simple web server",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "underscore": "1.8.x"
  },
  "author": "",
  "license": "ISC"
}
```

Your Answer?

- [ ] 11.3 Express
- [ ] 11.4 Pug
- [ ] 11.5 Relational databases (Node)
- [ ] 11.6 MySQL (Node)
- [ ] 11.7 MongoDB
- [ ] 11.8 Mongoose
- [ ] 11.9 Creating RESTful web APIs (Node)
- [ ] 11.10 Using RESTful web APIs (Node)
- [ ] 11.11 Third-party web APIs (Node)
- [ ] 11.12 Token-based user authentication (Node)
- [ ] 11.13 Password hashing (Node)

ZyBook – Web Programming

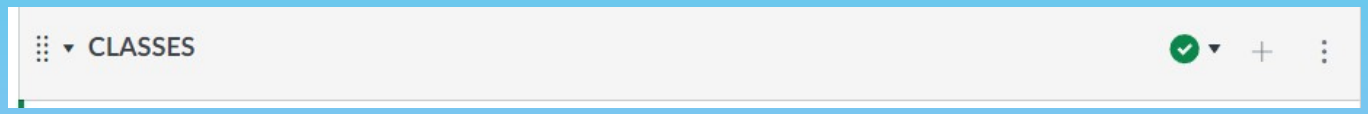
And so on...

Have fun!

ZyBooks takes you step by step

At 5:00 PM

10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		



VH– Publish.

10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		
10.18.2023 (W 4 to 5:30) (17)	Analysis: 1. Semi-Formal and Formal Specification Techniques for Software Systems.pdf	Lecture 5: Requirements		Requirements Papers Summary (1 Page) CANVAS Assignment	

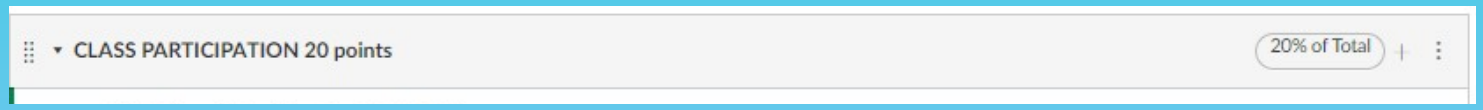
CLASS PARTICIPATION 20%

One Page Summary of Requirements Papers
Oct 18 | 100 pts

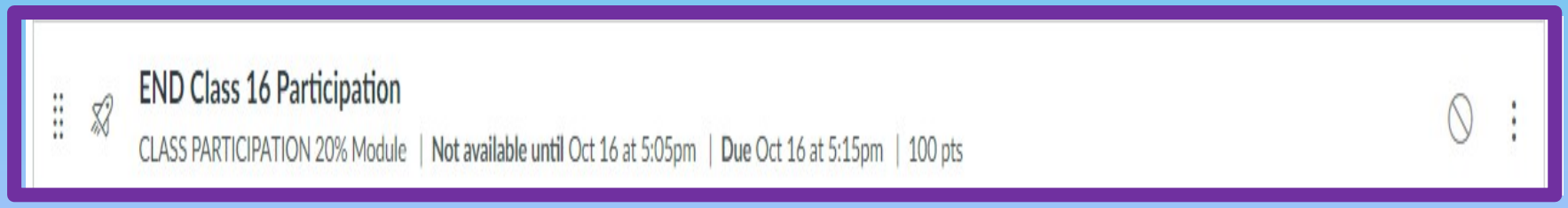
VH– Publish.

From 5:05 to 5:15 – 10 minutes.

10.16.2023 (M 4 to 5:30) (16)	Requirements: 1. Why Software Projects Fail 2015.pdf 2. The Why What who When and How of Software Requirements 2015.pdf	WEB TECHNOLOGIES	Start ZyBook: Sections 10-11		



PASSWORD: IN TEAMS



VH– Publish.

At 5:15.

End Class 16

VH, Download Attendance Report
Rename it:
10.16.2023 Attendance Report FINAL

VH, upload Class 16 to CANVAS.