TOC: Front-end Frameworks: Bootstrap & Web Tooling

Tuesday, September 13, 2022 6:10 PM

Front-end Web UI Frameworks

Monday, August 29, 2022 2:40 PM

This module gives you a quick introduction to full-stack web development and the outline of the course. Then you will learn the basics of Bootstrap, setting up a web project using Bootstrap. You will learn about responsive design and the Bootstrap grid system. At the end of this module, you need to complete your first assignment.

Learning Objectives

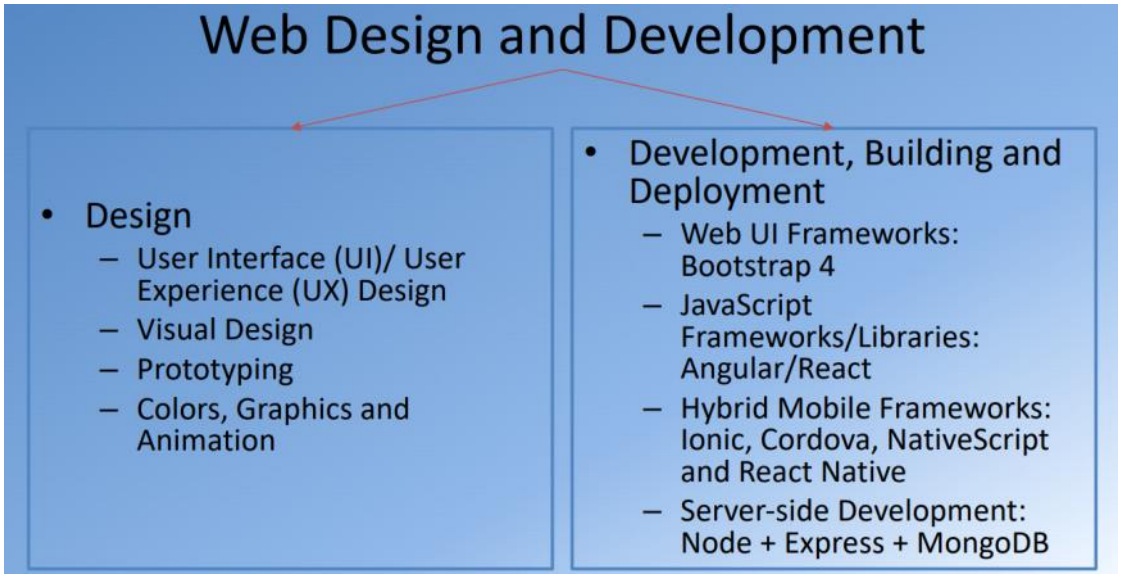
- Express what is meant by full stack in the context of web development
- Create a project with Bootstrap support
- Create a responsive website using the Bootstrap grid system
- Develop a website that can automatically adapt to different screen sizes and resolutions

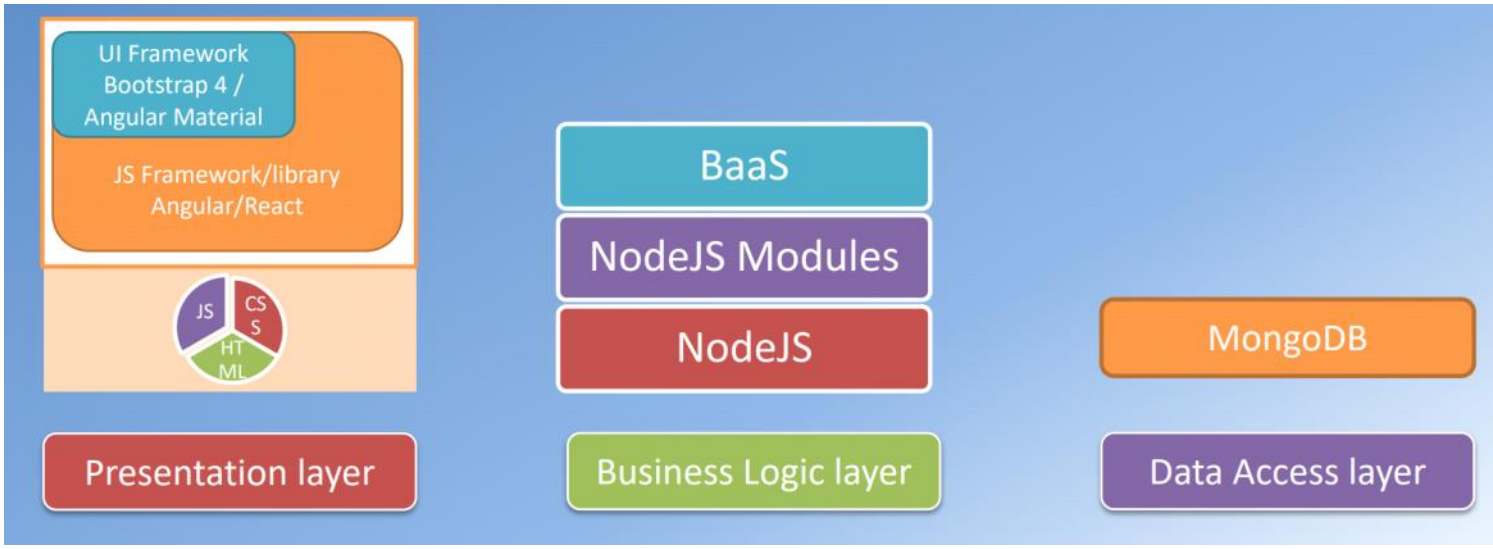
From <https://www.coursera.org/learn/bootstrap-4/home/week/1>

Welcome to front end Web UI Frameworks and Tools: Bootstrap 4.
I'm glad you decided to join this course to learn
about the most popular front end Web UI Framework: Bootstrap.
We'll look at various aspects of Bootstrap.
Through several examples, you will learn by doing exercises as part of this course.
Let's look at some details next.
Before you begin this course,
please make sure that you have sufficient background to
be able to succeed in this course.
You should have a good knowledge of HTML,
CSS, and JavaScript before you start the course.
From my perspective, web design and development consists of two aspects:
one is the design of the website and the web page,
the second one is the actual building and deployment of the web page itself.
From the design perspective,
what I mean is the design of the user interface and the user experience,
the visual design, the prototyping,
the elements of colors,
graphics and animation that might be of interest.
In this Specialization, we are not looking at this aspect of web development.
Instead, we are looking at that second aspect,
which is the development, building,
and deployment of websites and web pages, using technologies
like the front end web UI frameworks like Bootstrap,
maybe JavaScript framework like Angular or a Library like React
and hybrid mobile frameworks to design
mobile applications and also server-side development using Node,
Express, MongoDB, the main stack in more detail.
If you are looking at the Specialization,
thinking about the design aspect of website,
then you may wish to rethink again.
This Specialization concentrates on
the development building and deployment of website and web pages,
a purely technical knowledge on using the HTML,
CSS, and JavaScript-based skills for actual website development.
You're also probably wondering about what is meant by full stack web development.
We'll look at more details of full stack web development in the next lesson.
In particular, if you're looking at how this course is
positioned in the general context of full stack web development,
in this course, we are dealing with front end web UI development.
In particular, the UI Framework Bootstrap 4.
We'll be looking at the remaining aspects of
full stack web development in the rest of this specialization.
This course looks at Bootstrap in great detail.
We will also look at responsive web design and how Bootstrap supports
responsive web design through the Bootstrap Grid system.
We'll look at the CSS and JavaScript-based components in Bootstrap and how you
can make use of them in building your website and your web pages.
Along the way, we'll learn about that web development using the command line.
So we'll learn a lot of web tools that are
based on the command line and the Node.js ecosystem.
We'll briefly review Git,
add Node.js and look at Noje.js based tools including task runners like Grunt and Gulp.
This course, itself, is structured into full modules.

Each module roughly corresponding to one week of work.
In the first module,
we'll get the big picture view of full stack web development,
then you'll get a quick introduction to Git and Node.js.
Then we'll introduce you to the Bootstrap and then review the Bootstrap Grid system.
That will lead you up to the first assignment in this course.
The second module deals with Bootstrap CSS components.
We'll look at the design of the Bootstrap navigation bar
and how we can make use of it to support navigation.
We'll look at user input through
buttons and forms then we'll look at how we can display content using tables and cards.
Then we'll look at how we can include images and media into our Web page using images,
thumbnails, and media objects.
And then, finally, we look at how we alert users using tags,
alerts and progress bars.
This should lead you up to your second assignment.
The third module deals with Bootstrap JavaScript components.
We'll look at the big picture view of
how Bootstrap JavaScript components work, we'll review tabs,
pills and tabbed navigation,
then we'll look at how collapse and accordion can be used to show and hide content.
And then we'll look at the use of tooltips,
popovers and modals to reveal content to be displayed in your page.
And then we'll look at the carousel component,
which allows you to display sliding information on your web page.
This should lead you up to the third assignment in this course.
The last module deals with Bootstrap and the JQuery and various dev tools.
We'll, in particular, look at how Bootstrap and JQuery interact and how you can
write JQuery and JavaScript code in order to control your Bootstrap JavaScript component.
We look at the various methods that are supported by the Bootstrap JavaScript components,
which can be leveraged to write
JavaScript code to control the behavior of these components.
Then we'll review CSS pre-processing language is like LESS and Sass.
Then, finally, we'll look at how we can build and deploy our website
using NPM scripts or task runners like Grunt and Gulp.
This should take you all the way to the final assignment in this course.
I hope you will have a lot of fun doing
the various parts of this course and also enjoy the exercises that you'll
encounter at each stage that enable you to better
understand various aspects of the Bootstrap web UI framework.

From <https://www.coursera.org/learn/bootstrap-4/lecture/Rzoc7/front-end-web-ui-frameworks-and-tools-bootstrap-4-course-overview>




UI Framework
Bootstrap 4 /
Angular Material
JS Framework/library
Angular/React
JS
CS
S
HT
ML
BaaS
NodeJS Modules
NodeJS
MongoDB
Presentation layer
Business Logic layer
Data Access layer

Setting Up

Monday, August 29, 2022 3:46 PM

Setting up your Development Environment

Software Requirements

1. **Text editor of your choice**: Any text editor that you are already familiar with can be used for editing the project files. I will be using Visual Studio Code (https://code.visualstudio.com/) as the editor of choice in this specialization. You may also consider other editors such as Brackets (http://brackets.io/), Sublime Text (http://www.sublimetext.com/), or Atom (https://atom.io/).
2. **Browser of your choice**: You may use your preferred browser. I will be using Chrome as the browser in all the exercises. All the exercises and assignments in this course have been tested using Chrome v. 46. Please note that not all browsers may support all the HTML5 features to the same extent. You might encounter problems when using other browsers. I strongly urge you to use the latest Chrome browser for the exercises and assignments in this course so that any problems are minimized.
3. **Command line shell**: Familiarity with the command-line shell will be essential for the exercises. In Windows a cmd window or power shell with admin privileges would be needed. On a Mac or in Linux, a terminal window can be used. Please get familiar with the "sudo" command in OS X and Linux.
4. **Files required for the exercises**: We will provide additional starter files for the exercises wherever needed. Links to download the files will be provided inline in the **exercise instructions** that follow each exercise video. Please download the files provided there, if any, before beginning the exercise. The links are also available through the **Additional Resources** of the specific lesson.

Note: Please remember to retain the folders and all the files that you create in the exercises. Further exercises will build upon the files that you create in the preceding exercises. DO NOT DELETE the files at the end of the exercises, unless otherwise instructed. You may wish to set up your exercise folder as a Git repository and commit the files to the repository at the end of each exercise.

From <https://www.coursera.org/learn/bootstrap-4/supplement/vRljx/setting-up-your-development-environment>

Exercise (Instructions): Setting up Git

Objectives and Outcomes

In this exercise you will learn to install Git on your computer. Git is required for using all the remaining Node.js and Node based tools that we encounter in the rest of the course. At the end of this exercise, you would be able to:

- Install Git on your computer
- Ensure that Git can be used from the command-line or command-prompt on your computer
- Set up some of the basic global configuration for Git

Downloading and Installing Git

- To install Git on your computer, go to https://git-scm.com/downloads to download the Git installer for your specific computing platform.
- Then, follow the installation steps as you install Git using the installer.
- You can find more details about installing Git at https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. This document lists several ways of installing Git on various platforms.
- Installing some of the GUI tools like GitHub Desktop will also install Git on your computer.
- On a Mac, setting up XCode command-line tools also will set up Git on your computer.
- You can choose any of the methods that is most convenient for you.
  Some Global Configuration for Git

- Open a cmd window or terminal on your computer.
- Check to make sure that Git is installed and available on the command line, by typing the following at the command prompt:

```
git --version
```

- To configure your user name to be used by Git, type the following at the prompt:

```
git config --global user.name "Your Name"
```

- To configure your email to be used by Git, type the following at the prompt:

```
git config --global user.email <your email address>
```

- You can check your default Git global configuration, you can type the following at the prompt:

```
git config --list
```

Conclusions

At the end of this exercise you should have Git available on the command-line of your computer.

From <https://www.coursera.org/learn/bootstrap-4/supplement/0hPqo/exercise-instructions-setting-up-git>

Setting up your Development Environment: Git and Node: Additional Resources

PDFs of Presentations

Git.pdf
PDF File
Git-Exercises.pdf
PDF File
NodeJS.pdf
PDF File
Exercises-Node-NPM.pdf
PDF File

Additional Resources (Git)

- Git site http://git-scm.com.
- Installing Git chapter from Pro Git
- Git reference manual
- Quick reference guides: GitHub Cheat Sheet (PDF) | Visual Git Cheat Sheet (SVG | PNG)
- Atlassian comprehensive Git tutorial

Additional Resources (Node.js and NPM)

- Nodejs.org
- Npmjs.com
- Node API Documentation
- NPM Documentation
- lite-server

From <https://www.coursera.org/learn/bootstrap-4/supplement/GhmQK/setting-up-your-development-environment-git-and-node-additional-resources>

Exercise (Instructions): Setting up Node.js and NPM

Note: Make sure you have installed Git on your machine before you install Node.js. Please complete the previous Git installation exercise before proceeding with this exercise.

Objectives and Outcomes

In this exercise, you will learn to set up the Node.js environment, a popular Javascript based server framework, and node package manager (NPM) on your machine. To learn more about NodeJS, you can visit https://nodejs.org. For this course, you just need to install Node.js on your machine and make use of it for running some front-end tools. You will learn more about the server-side support using Node.js in a subsequent course. At the end of this exercise, you will be able to:

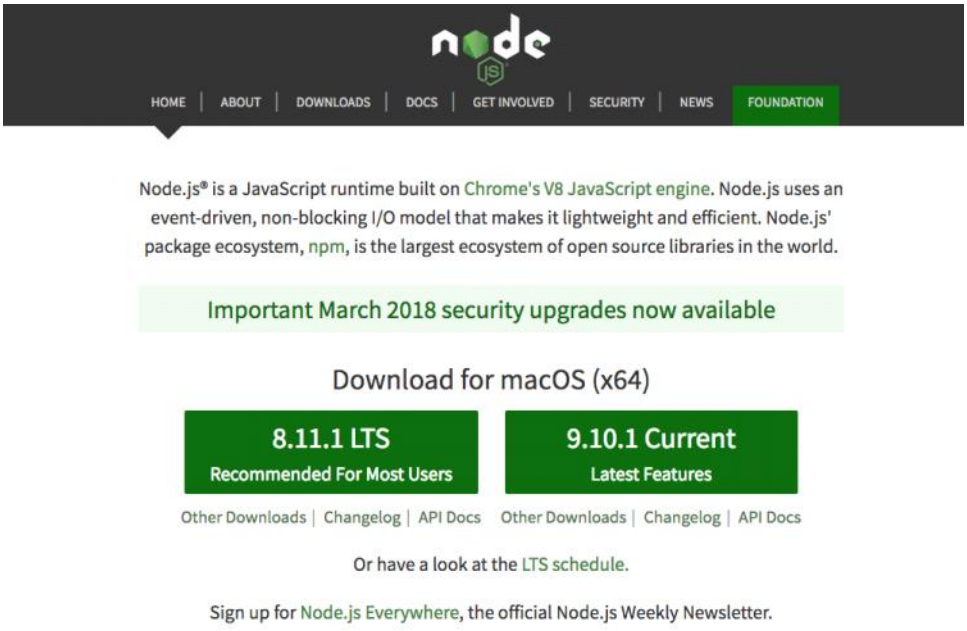
- Complete the set up of Node.js and NPM on your machine
- Verify that the installation was successful and your machine is ready for using Node.js and NPM.

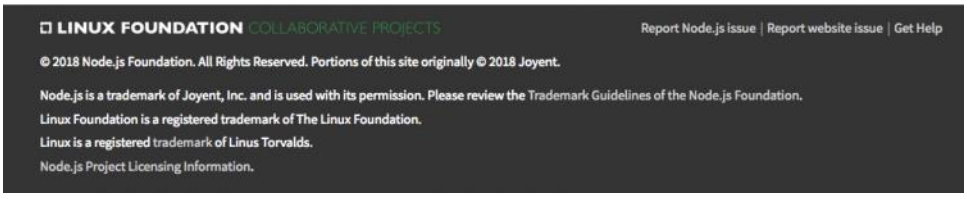
Installing Node

- To install Node on your machine, go to https://nodejs.org and click on the Download button. Depending on your computer's platform (Windows, MacOS or Linux), the appropriate installation package is downloaded.
- As an example, on a Mac, you will see the following web page. Click on the Download button. Follow along the instructions to install Node on your machine. (Note: Now Node gives you the option of installing a mature and dependable LTS version and a more newer stable version. You should to install the LTS version. I will use this version in the course.)
  Note: On Windows machines, you may need to configure your PATH environmental variable in case you forgot to turn on the add to PATH during the installation steps.

Verifying the Node Installation

- Open a terminal window on your machine. If you are using a Windows machine, open a cmd window or PowerShell window with **admin** privileges.
- To ensure that your NodeJS setup is working correctly, type the following at the command prompt to check for the version of **Node** and **NPM**






```
node -v

npm -v
```

Conclusions

At the end of this exercise, your machine is now ready with the Node installed for further development. We will examine web development tools next.

From <https://www.coursera.org/learn/bootstrap-4/supplement/2dirH/exercise-instructions-setting-up-node-js-and-npm>

Git Exercise

Monday, August 29, 2022 6:00 PM

Exercise (Instructions): Basic Git Commands

Objectives and Outcomes

In this exercise you will get familiar with some basic Git commands. At the end of this exercise you will be able to:

- Set up a folder as a Git repository
- Perform basic Git operations on your Git repository

Basic Git Commands

- At a convenient location on your computer, create a folder named **git-test**.
- Open this git-test folder in your favorite editor.
- Add a file named *index.html* to this folder, and add the following HTML code to this file:

```
<!DOCTYPE html>
<html>
    <head></head>

    <body>
        <h1>This is a Header</h1>
    </body>
</html>
```

Initializing the folder as a Git repository

- Go to the git-test folder in your cmd window/terminal and type the following at the prompt to initialize the folder as a Git repository:

```
git init
```

Checking your Git repository status

- Type the following at the prompt to check your Git repository's status:

```
git status
```

Adding files to the staging area

- To add files to the staging area of your Git repository, type:

```
git add .
```

Commiting to the Git repository

- To commit the current staging area to your Git repository, type:

```
git commit -m "first commit"
```

Checking the log of Git commits

- To check the log of the commits to your Git repository, type

```
git log --oneline
```

- Now, modify the *index.html* file as follows:

```
<!DOCTYPE html>
<html>
    <head></head>

    <body>
        <h1>This is a Header</h1>
        <p>This is a paragraph</p>
    </body>
</html>
```

- Add a sub-folder named **templates** to your **git-test** folder, and then add a file named *test.html* to the templates folder. Then set the contents of this file to be the same as the *index.html* file above.
- Then check the status and add all the files to the staging area.
- Then do the second commit to your repository
- Now, modify the *index.html* file as follows:

```
<!DOCTYPE html>
<html>
    <head></head>

    <body>
        <h1>This is a Header</h1>
        <p>This is a paragraph</p>
        <p>This is a second paragraph</p>
    </body>
</html>
```

- Now add the modified index.html file to the staging area and then do a third commit.

Checking out a file from an earlier commit

- To check out the index.html from the second commit, find the number of the second commit using the git log, and then type the following at the prompt:

```
git checkout <second commit's number> index.html
```

Resetting the Git repository

- To discard the effect of the previous operation and restore index.html to its state at the end of the third commit, type:

```
git reset HEAD index.html
```

- Then type the following at the prompt:

```
git checkout -- index.html
```

- You can also use *git reset* to reset the staging area to the last commit without disturbing the working directory.

Conclusions

At the end of this exercise you should have learnt some basic Git commands. Experiment with these commands until you fully understand how to use Git.

Exercise (Instructions): Online Git Repositories

Objectives and Outcomes

In this exercise you will learn about how to set up and use an online Git repository and synchronize your local Git repository with your online repository. At the end of this exercise, you will be able to:

- Set up the online repository as a remote repository for your local Git repository
- Push your commits to the online repository
- Clone an online Git repository to your computer

Setting up an Online Git repository

- Sign up for an account either at Bitbucket (https://bitbucket.org) or GitHub (https://github.com).
- Then set up an online Git repository named **git-test**. Note the URL of your online Git repository. Note that private repositories on GitHub requires a paid account, and is not available for free accounts.

Set the local Git repository to set its remote origin

- At the prompt, type the following to set up your local repository to link to your online Git repository:

```
git remote add origin <repository URL>
```

Pushing your commits to the online repository

- At the prompt, type the following to push the commits to the online repository:

```
git push -u origin master
```

Cloning an online repository

- To clone an online repository to your computer, type the following at the prompt:

```
git clone <repository URL>
```

Conclusions

In this exercise you have learnt to set up an online Git repository, synchronize your local repository with the remote repository, and clone an online repository.

From <https://www.coursera.org/learn/bootstrap-4/supplement/LkZXk/exercise-instructions-online-git-repositories>

From <https://www.coursera.org/learn/bootstrap-4/supplement/cgWKo/exercise-instructions-basic-git-commands>

Node Exercise

Monday, August 29, 2022 6:09 PM

Exercise (Instructions): Basics of Node.js and NPM

Objectives and Outcomes

In this exercise you will learn the basics of Node and NPM. At the end of this exercise, you will be able to:

- Set up package.json file in the project folder for configuring your Node and NPM for this project
- Install a NPM module and make use of it within your project

Initializing package.json

- At the command prompt in your **git-test** folder, type

```
npm init
```

- Follow along the prompts and answer the questions as follows: accept the default values for most of the entries, except set the entry point to index.html
- This should create a *package.json* file in your **git-test** folder.

Installing an NPM Module

- Install an NPM module, lite-server, that allows you to run a Node.js based development web server and serve up your project files. To do this, type the following at the prompt:

```
npm install lite-server --save-dev
```

- You can check out more documentation on lite-server here.
- Next, open package.json in your editor and modify it as shown below. Note the addition of two lines, line 7 and line 9.

```
  },
  "author": "",
  "license": "ISC",
  "homepage": "https://bitbucket.org/jogesh_k_muppala/git-test#readme",
  "devDependencies": {
    "lite-server": "^2.2.2"
  }
}
```

```
"scripts": {
```

```
    "start": "npm run lite",
    "test": "echo \"Error: no test specified\" && exit 1",
    "lite": "lite-server"
  },
```

- Next, start the development server by typing the following at the prompt:

```
npm start
```

- This should open your *index.html* page in your default browser.
- If you now open the *index.html* page in an editor and make changes and save, the browser should immediately refresh to reflect the changes.

Setting up .gitignore

- Next, create a file in your project directory named *.gitignore* (**Note**: the name starts with a period) Then, add the following to the .gitignore file

```
node_modules
```

- Then do a git commit and push the changes to the online repository. You will note that the node_modules folder will not be added to the commit, and will not be uploaded to the repository.

Conclusions

In this exercise you learnt to set up package.json, install a npm package and start a development server.

From <https://www.coursera.org/learn/bootstrap-4/supplement/JTkjO/exercise-instructions-basics-of-node-js-and-npm>

Bootstrap

Monday, August 29, 2022 8:27 PM

Introduction to Bootstrap: Objectives and Outcomes

In this lesson, you will be given a quick overview of front-end UI frameworks, and an introduction to Bootstrap. The exercises will introduce you to getting started with Bootstrap for your web project. At the end of this lesson, you will be able to:

- Identify the purpose of using front-end UI frameworks in web design and development
- Set up a project with Bootstrap support
- Configure a web project to use Bootstrap
- Become familiar with the basic features of Bootstrap

Note: For those of you who are already familiar with Bootstrap 3, here is an overview from the Bootstrap 4 documentation on the major changes in Bootstrap 4 compared to Bootstrap 3. While you will find Bootstrap 4 to have a lot of overlap in its classes with Bootstrap 3, several breaking changes have been introduced, including removing some components and introducing new components. This course covers Bootstrap 4 with the assumption that you are not familiar with Bootstrap.

From <https://www.coursera.org/learn/bootstrap-4/supplement/v3LOA/introduction-to-bootstrap-objectives-and-outcomes>

Let me do a quick introduction to Bootstrap before we go on to
learn how to make use of Bootstrap in designing our website.
If you go to the Bootstrap webpage,
you will see that Bootstrap claims to be the most popular HTML, CSS,
and JavaScript based framework that is in use today and specially designed
for developing responsive mobile first websites.
Indeed, Bootstrap stands up to its claim as the most popular framework.
It supports a number of CSS classes that enable you to design radius,
web components very effectively.
It supports consistent typography to be used on your webpage for rendering content.
It supports many different kinds of that components like buttons,
tables, navigation bars, modals,
images, carousels, and many other,
as well as many JavaScript enabled competence.
This enables you to design responsive websites with the mobile first approach.
We will talk more about this in the next lesson.
Bootstrap originated around 2011.
It was created by these two people from Twitter.
Hence, you might often hear people talking about Bootstrap as Twitter Bootstrap.
Mark Otto and Jacob Thornton from Twitter designed
Bootstrap for their own internal use and then ended up releasing
it for the rest of the web development community to use for consistent website design.
The current production version of Bootstrap is 4.0.
If you had taken my earlier specialization on Coursera,
I have dealt with Bootstrap 3.3.7 in the earlier Bootstrap course.
Now, it's time to move on to our first exercise where we will get our hands dirty with Bootstrap.

From <https://www.coursera.org/learn/bootstrap-4/lecture/SJK4s/introduction-to-bootstrap>

Exercise (Instructions): Getting Started with Bootstrap

Practice with Lab Sandbox

Exercise: Getting Started with Bootstrap

Exercise Resources

Bootstrap4-starter

ZIP File

Objectives and Outcomes

This exercise introduces the first set of steps to set up your web page to make use of Bootstrap classes and components. At the end of this exercise, you will be able to:

- Download Bootstrap using NPM and include it in your project
- Understand how to set up a web project to use Bootstrap
- Include the Bootstrap CSS and JS classes into a web page

Note: Please remember to retain the folder and all the files that you create in this exercise. Further exercises will build upon the files that you create in this exercise. DO NOT DELETE the files at the end of the exercise.

Setting up the Project Folder

- Go to a convenient folder location on your computer and download the ***Bootstrap4-starter.zip*** file using the link provided at the top of this page.
- Unzip the file to see a folder named ***Bootstrap4*** and a sub-folder under it named ***conFusion*** created. Move to the *conFusion* folder.
- Open a cmd window/terminal and move to the conFusion folder.
- At the prompt type

```
npm install
```

- This will install the lite-server node module to your project.
- Next, initialize a Git repository in the project folder, and then set up a .gitignore file with the contents as shown below:

```
node_modules
```

- Now do a commit of your project folder to the Git repository with the message "Initial Setup". You will be doing a commit of your project at the end of each exercise so that you retain the completed files of each exercise.
- Set up an online Git repository and synchronize your project folder with the online repository.

Downloading Bootstrap

- You will use npm to fetch the Bootstrap files for use within your project. Thereafter you need to install JQuery and Popper.js as shown below since Bootstrap 4 depends on these two. At the prompt, type the following to fetch Bootstrap files to your project folder:

```
npm install bootstrap@4.0.0 --save
npm install jquery@3.3.1 popper.js@1.12.9 --save
```

- This will fetch the Bootstrap files and store is in your node_modules folder in a bootstrap folder. The bootstrap->dist folder contains the precompiled Bootstrap CSS and JS files for use within your project.
- Open your project folder in your editor, and then open the index.html file in the *conFusion* folder. This is your starting web page for the project. We have already created the web page with some content to get you started. We will use Bootstrap to style this web page, and learn Bootstrap features, classes and components along the way.
- Start your lite-server by typing **npm start** at the prompt. The *index.html* file should now be loaded into your default browser.

Getting your Web page Bootstrap ready

- Open the *index.html* file in your favourite text editor. If you are using Visual Studio Code, Brackets, Sublime Text or similar editors, you can open the project folder in the editor and then view index.html.
- Insert the following code in the *<head>* of *index.html* file before the title.

```
    <!-- Required meta tags always come first -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="node_modules/bootstrap/dist/css/bootstrap.min.css">
```

- This will include Bootstrap CSS into your web page. Note the subtle change in the fonts of the content of the web page. This is the Bootstrap typography effect coming into play. The default Bootstrap typography sets the font to Helvetica Neue and selects the appropriate font size based on the choice of the heading style and paragraph style for the content.

- At the bottom of the page, just before the end of the body tag, add the following code to include the JQuery library, popper.js library and Bootstrap's Javascript plugins. Bootstrap by default uses the JQuery Javascript library for its Javascript plugins. Hence the need to include JQuery library in the web page.

```
    <!-- jQuery first, then Popper.js, then Bootstrap JS. -->
    <script src="node_modules/jquery/dist/jquery.slim.min.js"></script>
    <script src="node_modules/popper.js/dist/umd/popper.min.js"></script>
    <script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>
```

- Now, do a Git commit with the message "Intro. to Bootstrap". You may push the commit to your online repository.

Conclusion

We have now understood how to set up a web project to use Bootstrap. In the next lecture, we will explore further on responsive design and Bootstrap's grid system.

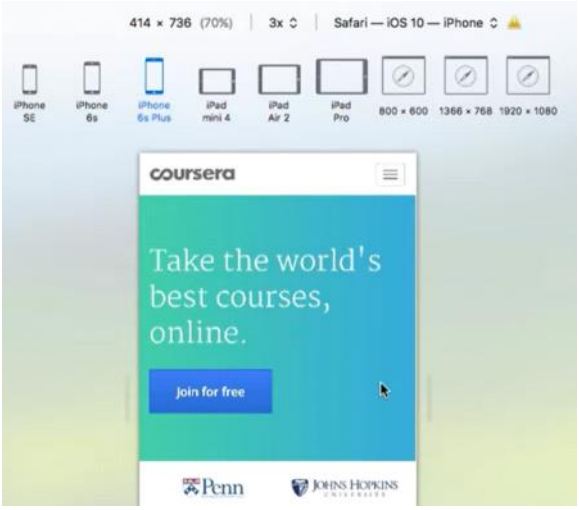
From <https://www.coursera.org/learn/bootstrap-4/supplement/kXErg/exercise-instructions-getting-started-with-bootstrap>

Responsive Design and Bootstrap Grid System: Objectives and Outcomes

In this lesson, you will be given an overview of responsive web design and an introduction to the Bootstrap grid system. The exercises will concentrate on enhancing your web project using the Bootstrap grid in order to make it responsive. At the end of this lesson, you will be able to:

- Understand the reasons for using responsive web design in a web project
- Use the Bootstrap grid system to design responsive websites
- Add your own custom CSS classes to a Bootstrap based web project

From <https://www.coursera.org/learn/bootstrap-4/supplement/jRGwX/responsive-design-and-bootstrap-grid-system-objectives-and-outcomes>



Foundation for Responsive Design

- Grid system
  - We'll deal with Bootstrap Grid system next
- Fluid images
  - We'll look at Bootstrap support later
- Media queries

Media Queries

- CSS technology to apply some styles based on the size of the viewport

```
e.g.,
@media (min-width: 992px) {
    /* CSS styles customized for desktop */
}
```

In the previous lecture,
we learnt about responsive design.
We also briefly mentioned that the Bootstrap grid system is

designed for supporting responsive websites.
How does Bootstraps grid support this?
And what exactly does Bootstrap grid involve,
that enables us to design responsive websites?
Let's talk a bit more detail about the Bootstrap grid system,
and how it supports responsiveness next.
We saw the use of the viewport meta tag in the previous lesson,
when we did the initial exercise on setting up Bootstrap in our index.html page.
I briefly referred to this particular line in the code then.
Let's try to understand why we use this in our index.html page.
What we are specifying here is that,
when our web page is being rendered by the browser in a particular device, then,
their rendering in the browser will take into account the size of the screen and
automatically adapt the rendering of the page to the devices screen width.
That way, then we have responsive classes,
CSS classes built into our UI framework,
then that will ensure that our web page is
correctly rendered for that particular screen size.
This is where the Bootstrap grid system comes to our rescue and
enables us to design responsive websites.
So again, emphasizing this point,
Bootstrap grid is designed to be responsive.
You have already seen what this means in the previous lecture,
and mobile first, which we have already discussed in the previous lecture and then fluid,
automatically adapting to the screen width.
The Bootstrap grid takes advantage of the CSS flexbox layout.
The CSS flexbox layout which is being supported by
the more recent versions of the various browsers,
enables a lot more simpler and flexible layout options in CSS.
Now, the actual discussion of
how CSS flexbox layout works is beyond the scope of this course,
we will restrict ourselves to how CSS flexbox is being
leveraged by Bootstrap for its grid system.
Earlier Bootstrap had its own grid system which was predating the CSS flexbox layout,
but the latest version of Bootstrap
has made the CSS flexbox layout as the standard for the Bootstrap grid.
This flexbox layout can easily handle
dynamic content and can adapt the containers to the dynamic content,
and also can easily adapt to unknown size of
the actual content enclosed inside the containers.
The flexbox layout also supports direction-agnostic layout.
What the flexbox layout brings to the Bootstrap grids is the ability
to do easy vertical alignment of the content within a parent element.
We"ll see the use of this in the exercise that follows.
Also, it allows easy reordering of the content
across different devices and screen resolutions with the help of media queries.
Again we'll see a little bit of that in the exercise on how Bootstrap
leverages that to support interesting ways of laying out content.
Also, that content itself,
the content containers can also be designed to be equal height columns so that
the content container with the largest height determines
the height of all the remaining containers that are laid out in the same row.
Let's now talk a little more details about the Bootstrap grid and how it actually works.
The way the Bootstrap grid works,
is by applying a container plus to the outermost layer.
We might use a div for
defining the element for which we apply the container class in general.
The container is the outermost unit within which the actual content is laid out.
Bootstrap supports both the container class which is a fixed width layout,
which means that the content itself is restricted to a fixed width on the screen.
This container class enables this fixed width to automatically adapt itself to
the size of the screen by using media queries so that
is where the responsive design comes into our rescue.
Later on, we will see how this container will adjust to different screen sizes.
Now, we also have the flexibility of using in
other classical container fluid which enables
the container to automatically adapt to the size of the screen.
But the fixed width container,
fixes the size of the actual layout width.

Inside the container, the content will be laid out in the form of rows,
so typically what we would do is inside
and out of div to which we apply the container class,
we"ll include an inner div to which we will apply the row class.
So, the content itself will be vertically divided into multiple rows.
And once this is divided into rows, within each row,
you can then layout the content using columns.
So, each row in Bootstrap,
will be divided into 12 equal sized columns.
Now, you can design your content to occupy any number of these 12 columns.
So, and the fact that the content itself adapts to
the screen width and the row itself being
enclosed inside the container will automatically adapt to the screen width,
and also the columns,
the 12 columns will be determined.
Their widths will be determined by the row by itself.
So, which means that for a different screen sizes,
whatever content you layout all of those 12 columns will automatically adjust
itself to the width that is allowed for the content.
So, this is how the responsiveness is built
into the Bootstrap grid way of laying out the content.
So, how exactly do we layout the content?
We'll talk about that in the next few slides.
The Bootstrap grid itself,
makes available to us five classes to specify different screen sizes by default.
There is a default class which targets all screen sizes,
starting from extra small to extra large.
So the entire range of screen widths in Bootstrap terms,
is divided into extra small, small,
medium, large, and extra large screen sizes.
Then look at how these different classes
of screen sizes are determined in detail in one of the later slides.
So, within your code,
you will identify the layout specification by
specifying sm for the small screen size,
md for medium, lg for large,
and xl for extra large screen sizes.
So, when you do the layout,
as we saw in the previous slide,
every row in a Bootstrap grid system is divided into 12 columns.
Now, we will layout content using what is called as the column classes.
For the column classes,
we can specify how the layout is done for different screen widths.
Look at the details of this in the subsequent slides later.
But let me quickly draw your attention to how we specify the column sizes.
They might specify the column sizes as col,
which is one of the classes they will say col-sm for
small and col-md for medium and large and extra large and so on.
Now, these column classes are used to specify
exactly how many columns in a row will each piece of content occupy.
How does this work?
We'll see that in the next slide.
So, going back to our Bootstrap Grid,
suppose we have a piece of content and we want to lay out the peace of content.
We can specify that that content is enclosed inside another div,
to which we apply column class,
like for example, col-sm-5.
So, in that case, what we are specifying is that for small to extra large screen sizes,
this piece of content will occupy five columns out of those 12 columns.
Now, you are going to immediately ask me,
what about the extra smocking?
If it is not explicitly specified for the extra smarted lockette
by the entire breadth of the 12 columns,
they can explicitly specify from which range to
which range the column layout will occupy how many of the columns.
So, in this particular case,
when I specify I only call them sm-5,
what I mean is that this particular content that is enclosed inside those div,
will occupy five columns for all screen sizes,

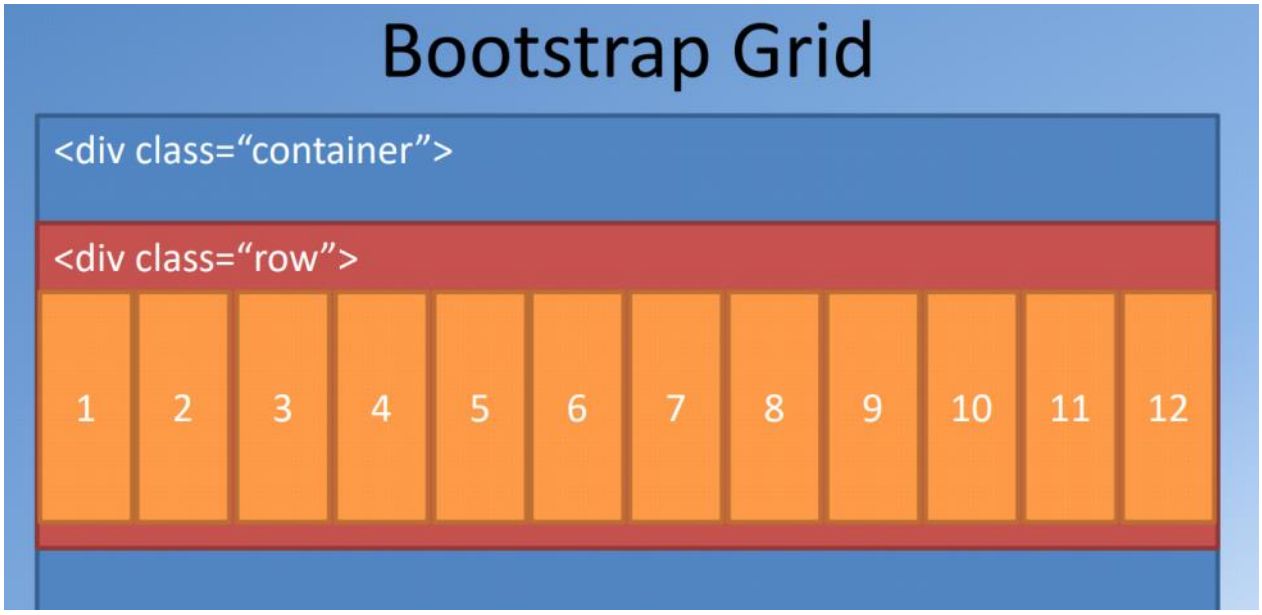
from small all the way to extra large.
Now, we now see that out of the 12 columns,
we have already occupied five columns.
There are still seven more columns left.
I could take another piece of content and then position it to the right of this,
by defending another div and then applying column sm-7.
Now, in this case,
you see that 5 plus 7 is equal to 12.
So, these two pieces of content that are enclosed in the two divs,
will now be positioned side by side,
and the sum total of them is occupying the entire 12 column width.
So, this is how we can specify and the work with the 12 columns
in defining the layout of the current tier.
With the use of the CSS flex box layout,
Bootstrap also supports what is called the Auto-layout Columns.
So, in this case,
I can simply specify
column sm without specifying how many columns it is supposed to occupy.
But in case I specify three Doobs width column sm in my heml 5,
then the bootstrap grid will automatically position these contents inside those divs,
side by side, so,
that each one of them gets one third of the total width.
So, since we have 12 columns here,
each of these three will get four columns each.
And this is automatically taken care of by the bootstraps layout mechanism.
Now, we can explicitly specify
the number of columns a particular piece of content occupy.
So, for example, suppose I specify three column classes,
but the middle one I specify column sm 6.
Then what happens is, when bootstrap does the layout,
the middle one will occupy six columns,
and then the left and the right pieces of content will occupy whatever is left off.
So, out of the 12 columns that we have,
if 6 columns that occupied by the middle piece of content,
then we have left with us 6 more columns and that will be equally
divided between the contents and the two sides of this middle column.
So, each of them will get free columns of space each.
So, that is how the layout is automatically done with Bootstrap.
In the exercise, we will see various combinations of use of these kind of
specifications of how many columns each piece of content will occupy in color layout.
This table summarizes the way Bootstrap treats
the different screen grids and correspondingly how you identify these screen grids.
So, as I specified earlier,
Bootstrap divides the entire screen width range into 5 classes.
The extra small screens are those whose grids are below 776 pixels.
Small screens are those that are between 576 and below 768 pixels.
Medium screens from 768 to 992,
large from 992 to 1200 and anything about 1200 pixels width
is treated to be an extra large screen.
So, this is how the default Bootstrap Grid is configured.
Now, once you get comfortable with Bootstrap,
you can also configure these divisions yourself.
But for this course,
we're going to stay with the default configuration that bootstrap supports.
Now, the grid behavior in this case,
is that whatever is laid out for extra small,
will be horizontal at all times,
for the higher bits they'll be collapsed to 2start width,
but horizontal about the break points.
We'll see how this works in a short while with some examples.
Now, we saw the use of the container in the previous slide.
For extra small screens,
the container width is automatically determined by the screen width.
But for small, medium,
large and extra large,
the container width is as specified.
So, if you use the div with a class container,
then for small screens it is 540 pixels.
So, you'll notice that if you have a screen width between 576 to 768,

your content will be laid out in 540 pixels,
will be centered in the screen width.
So, the leftover space will be left as margins on either side of this content.
Similarly, for medium it is 720 pixels and so on.
Now, then you specify how many columns each content occupies.
Then the column class prefixes that you will use
is.col for extra small,.col-sm for small,
all the way to extra large.
If you specify something with.col-md,
then that applies for medium to extra large screens.
So, whenever you specify any col- and some numbers,
then that applies to that particular screen size and everything about that screen size.
Now, in all screen size cases,
the number of columns is defined to be 12 columns.
The grid in case of bootstrap is designed to be 12 columns and that is configurable,
but the default value is 12 columns.
The reason for choosing 12 is that 12 is a good multiple of two, three,
four and so on.
So it gives you a lot of flexibility in terms of
how many columns you choose to lay out your content.
Between each column, so if you lay
out two pieces of content side by side with their divs,
between these two pieces of content they'll be a small gutter that will be left.
Empty white space that'll be left off width the gutter width,
which is just 30 pixels by default.
So, 15 pixels is from one piece of content and 15 pixels from other piece of content.
SumTotal together 30 pixels of
white space will be left between the two pieces of content.
Think about how a newspaper column layout
is done and you begin to see the correspondence between
the newspaper column layout is done and how
the bootstrap's grid does the layout of the content on that pitch.
Bootastrap's grid allows you to do next level content,
so you can enclose content inside content and then do nested content layout.
Also, it supports offsets.
We will see the use of offsets also in some examples later.
So let's look at our first example of how you would apply bootstraps column classes,
and how they would actually be rendered on different screen sizes.
Here is an example of a situation where I applied to the two divs,
column hyphen 12 and then column hyphen sm hyphen five for
the red piece of content and for
the other one I applied column hyphen 12 and column sm seven.
So the way this content will be laid out is for extra small screens,
the two pieces of content will be stacked one on top of the other.
So the red one will be stacked on top of the sea green colored one.
But for small to extra large screens,
the two pieces of content will be laid out side by side.
The reason for this is because we said column sm five and column sm seven for the two.
So for small to extra large they're laid out side by side so
that the red content will occupy the leftmost
five columns and the sea green content will occupy the right seven columns of your row,
all the way from small to extra large screen sizes.
So this is how we would specify content layout for different screen sizes.
Bootstrap also provides additional classes called as
the Order Hyphen Classes which allow you to reorder the content on the screen.
So for example, if you apply an order sm first,
an order some last class to the divs,
as is shown in the example here, in this case,
the div for which you apply order sm last will be pushed to
the right side of the screen and order
sm first div will be pushed to the left side of the screen.
So, applying these order classes allows you to reorder the content on the script.
Not only this, the order sm also allows you to specify same order sm one to
order sm 12 to specify the order in which the content needs to be rendered on the screen.
So using a larger number,
you could shift the content to the right side of
the row and a smaller number will shift the content to the left side of the row.
We will see an example of this also in the exercise.
The mixed interesting support that Bootstrap Grid brings with the use of

the Flexbox layout is vertical alignment of content.
The earlier Bootstrap versions,
like Bootstrap 3 and earlier,
did not have the ability to do vertical alignment of content.
With the use of the Flexbox for designing the Bootstrap Grid in Bootstrap four,
we get the flexibility that CSS Flexbox brings in terms of doing vertical alignment.
So if you wanted the content to be vertically-centered,
then to the row,
you would apply the class called "align items center".
So in this case, whatever content is laid out,
will be laid out vertically-aligned within that particular row.
So the content which occupies
the largest height will be the one to which the remaining ones will be
aligned when it is laid out in that row or if you pre-specify the height of the row,
then all the content will be centered vertically within that particular row there.
Not only that, Bootstrap's Grid also supports horizontal alignment of content.
Let's look at an example.
So if you specify your content like this.
Say, you have three columns and the first one you specify div class column three,
then the middle one you'll say div class column auto.
I'm going to come back to that column auto in a short while.
And the right one says div class column 3.
So in this case, what we are specifying is that the left and
the rightmost pieces of content will occupy three columns each.
The middle one, when I say, "column auto",
it means that the number of columns that this particular div
occupies will be automatically determined by the content that is enclosed inside there.
So based upon the content,
the number of columns occupied by the div will automatically adjust itself.
In this particular case, in this layout,
that particular content is being accommodated within four columns.
So now you have four columns being used by the center div,
three columns by the left and three columns by the right.
So sum total, you have 10 columns.
So you have two columns that are left empty.
Now, if you specify for the row,
you specify justify content center, then,
the entire content in this particular row will be centered with
respect to the row horizontal.
You can also have the content left-justified,
right-justified and a couple of other options.
Details are in the bootstraps documentation.
Bootstrap also allows you to do column offsets.
How does this work? Let's look at an example.
In this column offset,
we can specify a piece of content,
add to that if we apply the class as offset hyphen sm hyphen one.
What we are specifying is for small to extra large screens,
this piece of content should be right-shifted by one column.
So when this content is laid out, as you can see,
the leftmost one column is left blank and the content is shifted right by one column.
And you can layout the remaining content on the right side.
So sum total, you can see that the two divs will
occupy 11 columns but one column offsetted to the right.
So that is how we can control the layout of the content using a column offset.
So as you can see,
using the column size specifications,
the vertical and horizontal alignment,
the offset, the push and the pull,
and flex first and flex last,
we are able to get a lot of control on how we lay out
contents for different screen widths and different screen sizes.
Bootstrap goes even further by allowing you to nest content inside divs.
So for example, if you specify two divs,
as you see with column sm five and column sm seven,
the two contents will be laid out as seen here.
Now, inside the right div,
I can again go in and divide
that div's width into a row and then that row will automatically give me

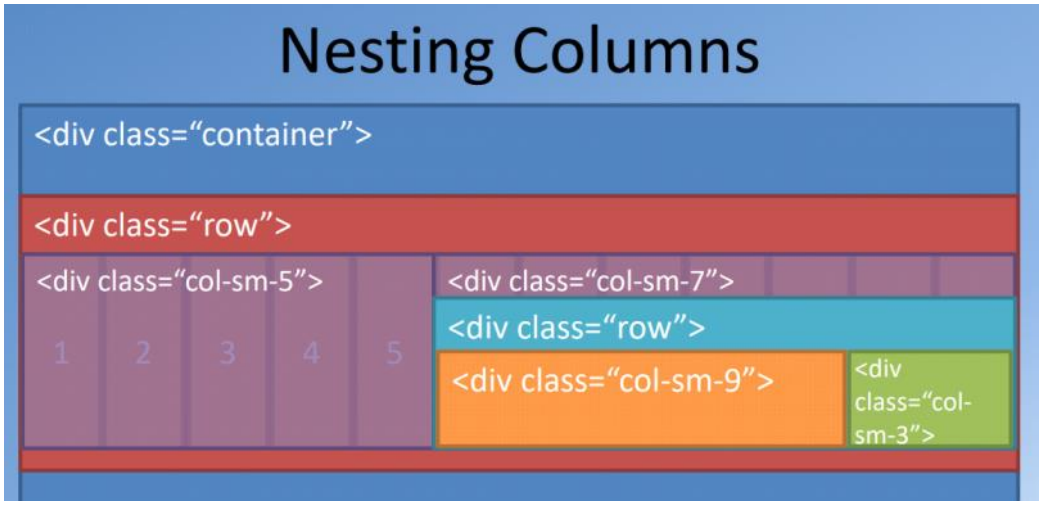
another 12 columns for that part of the screen and then I can then do the layout using
nesting of the divs and do the layout for different parts of that column.
So nesting like this,
brings you even more flexibility in the way you lay out content in your pitch.
With all the discussion about the Bootstrap Grid System and how
the Bootstrap Grid supports various ways of laying out content,
we're going to now, move on to our next set of exercises.
We're going to apply the Bootstrap grid to our index.html page in
order to do the layout of the content inside or index.html page.
We will also use some custom CSS classes.

From <https://www.coursera.org/learn/bootstrap-4/lecture/j7Gcb/bootstrap-grid-system>



Bootstrap Grid Details

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints			
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Offsets	Yes				



In the previous lectures,
we have learnt about responsive design and the bootstrap grid system.
We have seen how the bootstrap grid allows us to design responsive websites.
It's time for us to move onto the next exercise.
We'll begin to work with the index html page that we created in
our previous exercise and apply the bootstrap grid classes to make it responsive.
Taking a quick look at our web page in the browser we see that
it is still terrible although the fonts have improved a little bit,
and we are now using the bootstraps default fonts.
Let's now try and apply the bootstrap grid classes to index.html to improve its layout.
Going to the index.html page,
let me quickly draw your attention to this particular line in the
index.html page that we included already in the previous exercise.
I had referred to this line,
then we were talking about the bootstrap grid system.
So here, we see that we've specified the meta tags with a view port and the content,
width as device width,
and initial scale one and shrink to fit no.
So this meta tag allows us to make
our web page responsive by looking at the view port character.
Moving ahead to the next step.
We're going to roll down to the body of this index.html page,
and then look up the first div that comes right below the header tag there,
to this div you're going to apply the class container.
We have already learnt about the container class in the previous lecture.
So once we apply the container class,
then taking a look at our web page we see
that part of the content right there has already adapted itself.
Look at the difference between the content up here and the one below here.
Now this content down below here,
is in the footer of our index.html page and this content is in the header.
But that content on the middle is the actual content of our web page,
that is inside the div to which we apply the container.
So applying the container class you can immediately see that the width of the page within
which our content is laid out has now
shrunk and then you have some extra space left on either side,
so with this container width being a fixed size container,
our content will be fixed to this particular width on
the screen leaving enough margin on either side as white space.
Let's apply the same container class to both the header and the footer also,
and then see how the content changes.
Going back to our web page,
what I'm going to do now is for the inner content here,
the inner div inside the container,
I'm going to apply the class as the row class here,
to this first one,
and I'm going to simply copy this.
I'm going to apply the same to the second div here which is lined up for the div,
and also scroll down and then to the third div here,

I'm also applying the row class here.
So now, the content inside this outermost container class,
is now divided into three rows,
so whatever is inside here will be one row,
whatever is inside here,
will be the second row and the one in here will be the third row.
We'll later on apply the column classes to this.
Let us now move to the header in our index.html's body.
And to the header, I'm going to apply a class called as jumbotron.
The jumbotron component allows you to set
apart the content inside the jumbotron from the rest of the page.
We will see the result in a short while when we at the resulting web page.
Now, to the div inside this header,
I'm going to apply the class as container,
so that whatever is inside the content will
be now constrained by the container width there.
Similarly, moving down to the footer here,
Play video starting at :4:58 and follow transcript4:58
in the footer also,
the first div inside the footer,
I'm going to apply the class as container.
And the div inside there,
I'm going to apply the class as row there,
to the div inside there.
So this div will take us all the way to this particular div here.
And then the next div also that is right below that,
I will apply the class as row.
So now my footer contains two rows here,
we'll style the content using column,
classes in a short while.
Going to the header also,
I will apply the row class to the second div inside the container div here.
So this div which matches I've got this,
will be one single row in the header.
Let's take a look at our resulting web page.
Going to our web page,
you can now see that the header content is now set apart from
the rest inside this grey box on top here.
But the content itself is now lined up with the content down below here.
Now, this is the result of using the container.
Moving to the footer,
you can now see that the footer content is also now lined up inside the container.
But the page still looks not that great,
we'll now apply the column classes to the inner divs now.
Going back to the index.html page,
we'll now start applying the column classes.
So going to the header, to the second inner div inside the jumbotron here,
let me apply a class as column 12, column sm-6.
So, here the content in
this first div will occupy the entire row for extra small screen sizes,
and then will occupy half the row so that's why column
sm-6 for small to extra large screen sizes.
Now, similarly for the second div here,
although it doesn't contain any content there,
I'm going to define the column classes here as column 12 and then column sm.
So, notice that here by specifying column 12,
I'm explicitly stating that for extra small screen sizes whatever content is
inside this div will be stacked below the content about here.
And then for small to extra large screen sizes,
this content will occupy the leftover amount of
columns in the row for small to extra large screen sizes.
So here in this case since six columns are occupied by this content,
this will get the remaining six columns there.
Moving down to the content rows here,
for the content here,
so for this first content,
which is a label here,
I'm going to apply the classes as column 12,
column sm-4, column md-3.

So, stating that for extra small screen sizes,
this will occupy the entire row, for small,
it will occupy full columns,
and then for medium to extra large,
it will occupy three columns in the row.
Now, so the remaining part will be occupied by the content down below here,
so to that, I will apply the classes as col,
indicating that this will be stacked below
the content here in the extra small screen sizes,
and then I'll say column sm,
so which means that this will occupy the remaining eight columns here,
because four columns is taken up by this one,
and then we'll say column md,
so which means that this will occupy
the remaining nine columns because three columns are taken
up for medium to extra large screen sizes by the content above here.
Now that I have defined for the first content row,
I'm going to take the same column class application here,
and then apply that to the second row here,
and also the third content row here.
Similarly, I'm going to copy this one and then apply the same to
the content divs down below here.
So now we have configured both the header and the content columns there.
Let's now move down to the footer.
In the footer, you'll notice that this div contains three inner divs here,
so for each of these three inner divs here,
I'm going to apply the corresponding column classes.
So for the first one,
which contains links to the various pages that will form part of this website,
I'm going to apply a column class,
as column four, column sm-2 here.
So meaning that for extra small screen sizes,
this will occupy four columns and for small to extra large,
it will occupy just two columns here.
Then for the second one,
the second div in there I will apply the classes as column seven.
So notice that I have four here and then seven here,
so that total occupies 11 columns.
I have left one column empty.
I'll come back later to apply an offset to take care of the extra column.
So this is column seven,
and then col-sm-5 to the second one.
So this is two plus five, seven.
So I still have five columns left over,
which I can use for the third div here.
Now for extra small screen sizes,
these two contents will be positioned side by side,
and then this div which contains links to
a social media site will be in a separate row stacked below.
But for small to extra large screen sizes,
the first one will occupy two columns,
the second will occupy five and the remaining will be
taken up by social media links here.
So for the third one,
I'm going to apply the class
as col-12 col-sm-4 here.
So col-12 col-sm-4, meaning that this will occupy
a separate set of 12 columns
stacked below the previous content for extra small screen sizes,
but for small to extra large it will occupy four columns.
So here we are four plus five,
nine plus two 11.
So one column is still leftover for small to extra large.
So that's how I'm going to apply that column closest to this.
Now, we still have the one below here where we contain the copyrights to this.
I'm going to apply the class as column auto,
meaning that this content will occupy
just enough columns as is required by the content there.
Later on you will see that I will position this content in the center of

the screen thereby using yet another bootstrap CSS class.
We'll come to that in the next exercise.
So with this we have applied various column classes to the header,
the content, and the footer.
This is a good time for us to save the changes and then go and take
a look at the updated indexed or HTML page.
Taking a look at our index.html page,
you'll notice how in the header which is the Jumbotron,
the content is now occupying just half of the screen there.
The other half of the screen of this particular row is now left empty.
Coming down to the content rows,
you see that the label on the left side occupying
three or four columns depending upon which screen size were using,
and the remaining being occupied by the rest of the content.
So here we have one row,
the second row, and the third row.
And then the footer here you can see that the links are on the left side,
the others in the middle,
and then the social media links on the right side.
So this is for a medium sized screen.
Now if you want to look at the same view for an extra small or small screen sizes,
if you're using Chrome,
then Chrome has the developer tools that you can
turn on by clicking on the view developer tools,
and when the developer tools come up,
you will notice, let me reduce the size here.
You will notice this small to windows there.
So clicking on that will turn on the responsive view for these sites there.
So you can see that in here the view of this same web page on a pixel to site,
which is 411 by 731 in portrait mode,
so which is corresponding to an extra small screen size,
so you can see how the content is laid out.
So you can see the Jumbotron content there
and then the remaining part of the content down below here.
And then down to the footer.
So in the footer you can see how the links and ad does have laid out side by side.
Now reducing the screen size so that we can see how that footer is laid out here.
So the remaining part, you have the links on the left side,
you have the address on the right side,
and then the social media links down below in a separate row here,
and then the copyright at the bottom there.
So this is the interesting view that we see for extra small screen sizes.
Getting back to our code here,
I'm going to now apply the order and offset classes to
this content in order to display the content in a slightly different manner.
So going to the content row here,
for the two divs here which contain the content,
I'm going to apply the classes as
order-sm-last for the first row here.
And then for that one down below here I will apply order sm first to the row down here.
So which means that when this content is displayed,
this content will be pulled to the left side of the screen and then
this column will be pushed to the right side of the screen.
So this will be ordered to the right side and this will be ordered to the left side.
As I explained about the order clauses in the previous lecture.
Similarly from the second row,
I'm going to leave it as such but from the third row I'm going to apply
the same set of order classes.
So I'm going to go to the third row here,
and then apply the order-sm-last to this one,
and then I will apply the order-sm-first to this one.
And so this one again will be reordered
such that this content will appear to
the left side and this content will appear to the right side.
This is just a way of positioning the content in a bit more interesting way.
Now moving down into the footer here,
now I'm going to apply an offset class here.
So notice that I have mentioned that these two columns will occupy four plus seven, 11.
So one column is leftover.

So I'm going to apply an offset-1 class to this one.
So which means that this content on this div will be pushed right by one column here.
Similarly since I said offset one,
this will be applied to extra small all the way up to the extra large screen sizes.
And so that is the use of the offset class.
So now that we have made the changes,
let's save the changes and then go and take a look at our web page in the browser.
Going back to that browser,
you now see how the header is as before,
but in the first content row,
you see that this content has been pushed to
the left side and the label to the right side.
The second row is different has being maintained just as before.
But for the third you'll see that this has been
pushed to the right and this has been pushed to the left.
So that is the use of the order-sm-last and order-sm-first class is there.
Now going to the footer,
you now see that the content in the first div here has been pushed right by one column,.
So you can see that there is one column of whitespace here,
and this isn't pushed right and the remaining ones have been formatted accordingly.
So with this we complete the changes
to our index.html page or additional page for this particular exercise.
With this we complete this particular exercise.
In this exercise we looked at the use of the container, row,
and column classes in order to design our web page a little bit nicer.
In the second part of the exercise which will follow this,
we're going to add more to try and improve the way these web pages are rendered.
It is still not to my satisfaction.
Obviously, there is still room for improvement.
This may be a good time for you to do a git comment with
the message Bootstrap Grid Part One.

From <https://www.coursera.org/learn/bootstrap-4/lecture/0H4jO/exercise-video-responsive-design-and-bootstrap-grid-system-part-1>

Exercise (Instructions): Responsive Design and Bootstrap Grid System Part 1

Practice with Lab Sandbox

Exercise: Responsive Design and Bootstrap Grid System Part 1

Objectives and Outcomes

This exercise introduces you to responsive design and Bootstrap support for mobile first responsive design through the use of the grid system. At the end of this exercise, you will be able to:

- Create responsive websites using the Bootstrap grid system
- Reordering content using push, pull and offset classes

Note: In this exercise we will continue to update the *index.html* file in the *conFusion* folder that we created and edited in the previous lecture.

Bootstrap Grid System and Responsive Design

Bootstrap is designed to be mobile first, meaning that the classes are designed such that we can begin by targeting mobile device screens first and then work upwards to larger screen sizes. The starting point for this is first through media queries. We have already added the support for media queries in the last lesson, where we added this line to the head:

```
1
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

The *viewport* meta tag ensures that the screen width is set to the device width and the content is rendered with this width in mind. This brings us to the second issue, designing the websites to be responsive to the size of the viewport. This is where the Bootstrap grid system comes to our aid. Bootstrap makes available four sizes, xs for extra small, sm for small, md for medium and lg for large screen sizes. We have already seen the basics of responsive design. In this exercise, we will employ the Bootstrap grid classes to design the websites. We would like our website to have the content stacked on extra small devices, but become horizontal within each row for

smaller devices and beyond. Towards this goal, we will make use of the classes .col-*, .col-sm-*, col-md-*, and .col-lg-* for defining the layouts for the various device sizes. We can specify how many columns each piece of content will occupy within a row, all adding up to 12 or a multiple thereof.

Using a Container class

- We use the container class to keep content within a fixed width on the screen, determined by the size of the screen. The alternative is to use the container-fluid class to make the content automatically to span the full width of the screen. We will discuss further about this when we discuss the Bootstrap grid system in the next lecture. Add the container class to the first div right after the </header> in the file as follows.

```
<div class="container"> ...
```

Dividing the content into rows

- Let us now add the class *row* to the first-level inner *div* elements inside the container. This organizes the page into rows of content. In the next exercise, we will see how we can add other classes to the rows.

```
    <div class="row"> ...
```

Creating a Jumbotron

- Let us add the class jumbotron to the header class as shown below. This turns the header element into a Bootstrap component named Jumbotron. A jumbotron is used to showcase key content on a website. In this case we are using it to highlight the name of the restaurant.

```
        <header class="jumbotron"> ...
```

- In the header add a **container** class to the first inner div and a row class to the second inner div.

Creating a footer

- Finally, in the footer add a **container** class to the first inner div and a row class to the second inner div.

Applying column classes within each row

- In the header row, we will display the restaurant name and the description to occupy 6 columns, while we will leave six columns for displaying the restaurant logo in the future. Let us go into the jumbotron and define the classes for the inner divs as follows:

```
                <div class="col-12 col-sm-6"> ... </div>

                <div class="col-12 col-sm"> ... </div>
```

- For the remaining three div rows that contain the content, let us define the classes for the inner divs as follows:

```
            <div class="col-12 col-sm-4 col-md-3"> ... </div>

            <div class="col col-sm col-md"> ... </div>
```

- For the footer, let us define the classes for the inner divs as follows:

```
            <div class="col-4 col-sm-2"> ... </div>

            <div class="col-7 col-sm-5"> ... </div>

            <div class="col-12 col-sm-4"> ... </div>

            <div class="col-auto"> ... </div>
```

Now you can see how the web page has been turned into a mobile-first responsive design layout.

Using Order and Offset with column layout classes

- In the content rows, we would like to have the title and description to alternate so that it gives an interesting look to the web page. For extra small screens, the default stacked layout works best. This can be accomplished by using the .order-sm-last and .order-sm-first for the first and the third rows as follows:

```
            <div class="col-12 col-sm-4 order-sm-last col-md-3"> ... </div>

            <div class="col col-sm order-sm-first col-md"> ... </div>
```

- For the div containing the <ul> with the site links, update the class as follows:

```
                <div class="col-4 offset-1 col-sm-2">
```

- After saving all the changes, you can do a Git commit with the message "Bootstrap Grid Part 1" and push your changes to the online repository.

Conclusion

In this exercise, we reviewed responsive design and the Bootstrap grid system.

From <https://www.coursera.org/learn/bootstrap-4/supplement/oWCkZ/exercise-instructions-responsive-design-and-bootstrap-grid-system-part-1>

We'll continue with the bootstrap grid exercise
that we started out in the previous exercise.
We're going to do a few more things to our index.html
page in order to improve its layout.
We're going to use some custom CSS classes and add some color
to our index.html page using custom CSS classes.
The next thing that I'm going to do is go down to the footer and for this UL here,
you'll notice that this list when you look at
our web page is displayed in the web page using this bulleted list here.
I'm not too happy with this bulleted list,
I want to remove these bullet points and then simply show them as only the links there.
So to do that, I'm going to use another list style called as
list-unstyled and apply that to that UL tag there.
Going back to our web page,
to this UL tag in the footer,
I'm going to apply the class as list-unstyled,
Play video starting at :1:23 and follow transcript1:23
and this would remove those bullets in front of these links there.
Now, let me add a few custom CSS classes to the index.html page.
To do that, we will now create a folder here and name it as the CSS folder here.
Inside the CSS folder,
I'm going to create a new file and name the file as styles.css.
Now, inside the styles.css file,
I can now add some CSS classes in order to style our page.
Let me add some CSS classes here.
So, I would add my first class as row header.
So as the name implies,
this class I am applying to the header.
And then, I would apply the margin as
zero pixel auto and
then padding as zero pixel auto.
So, this is for the row header.
So, I'm going to have zero margin and zero padding for the row header.
And then for the row contents,
so I'm going to add one more CSS class called row-content.
And then for this, I will add margin zero pixel auto,
and then padding have it at 50 pixel,
zero pixel, 50 and zero pixel.
Then I'm going to add a border at
the bottom of my row here.
And then, the border I will give it one pixel and then call it a ridge.

And I will set the minimum height to be 400 pixel for the content here.
That way, my web page will look like more nicer on the screen.
Now this again, you have to try and
see until you are satisfied with the way the page is laid out.
Now also, I'm going to include
one more class called as the footer which as you might have already guessed,
I'm going to apply to the footer and I
will apply a background color to the footer as D1C4E9.
Play video starting at :4:29 and follow transcript4:29
If you ask me how I selected the color,
I just looked up the color and then I was happy with that color.
You can choose any color you want to apply,
but this color seems to be good for me.
So, I added that color.
And for the padding, for the footer,
I would add 20 pixel and zero pixels.
Now, once I have added this styles to my CSS file,
let me apply the CSS file to the index.html page.
I'm going to index.html page.
Obviously, I need to use that CSS file inside my index.html page.
So right after the bootstrap CSS file,
I'm going to do a link and style sheet,
and this is style.css file there.
So that is where my CSS file is stored.
So, I'm going to include that into my index.html page.
Now, I am going to go into the body and apply the classes that I have just created.
So going to the header,
I will apply the row header class to the div in the header.
Then, for the content,
I would apply the row content classes to the rows there.
So, to the three rows in the content,
I apply the row content class and then going down to the footer,
to the footer tag,
I will apply the class footer here,
and then save the changes.
Let's go and have a look at that page now.
Going to your web page,
we're already beginning to see some interesting changes to the web page.
So, you can now see that the content in
the web page has been laid out a bit more cleaner.
That is the border bottom that we added to the row content,
so you can see that there is enough separation between the rows although
the contents of the rows are still positioned towards the top in the rows,
so you have these three rows and then look at the footer,
the footer now has acquired the background color that I applied,
and notice how the links are styled with the list-unstyled on the left side here.
We are not done yet, we got to do a few more changes to the index.html page,
and then come back and have a look at the final result of this exercise.
Going back to your Editor,
I'm going to add a few more CSS classes into my style structure,
it is a file, so I'm going to go in there and then add a class called Jumbotron.
So, you'll see that we have already applied the Jumbotron to our header there.
Now, if I add more properties to this Jumbotron class here,
this will be applied to the Jumbotron,
in addition to the standard default Jumbotron that bootstrap already includes.
I'm going to add in a few more things,
I'm going to add in a padding of
70 and 30 pixel on its surroundings,
and then a margin of zero pixel all around,
and I will set the background of this to 9575CD.
This is somewhat of a darker purple color which seems to go very well on the Jumbotron.
Again, by trial and error,
I selected that color,
and also the text color as floral white.
Play video starting at :9:13 and follow transcript9:13
So for the Jumbotron,
we are going to apply those classes, and also,
I'm going to apply one more class to the address class.
We have an address in the footer,
so I'm going to change the font size of that to 80 percent,

a slightly smaller font than the normal font,
and the margin, I will give it a zero pixel,
and the color, I'm going to use zero F,
zero F, zero F,
and then save the changes.
So couple more CSS classes added,
switching back to index.html.
What I'm going to do now,
is to go to the content rows and then
I'm going to try and position this content in the middle,
vertically in that row.
This is where I'm going to take the help of
the vertical alignment that bootstrap grid supports through the Flexbox support.
To do that, I would say align-items-center,
so this is the other class that I need to add, align-items-center.
And I'm going to add that to the remaining two rows also,
align-items-center to the second and the third row,
Play video starting at :10:53 and follow transcript10:53
and that should do a vertical justification of the content of the rows.
We'll see the result in a minute.
Going down to the footer.
In the footer, you see that we have used
the column auto for this copyright content there.
Now, I'm going to position this right in the middle horizontally
using that justify-content-center class,
so there is a justify content and a few other classes there.
I'm going to use to justify-content-center class for that.
That will essentially position that column in the center of the row there.
That applied to that, and then,
one little change that I'm going to do,
is to this inner div there,
I'm going to apply a class as text-center to that,
so that these links are centered on the screen.
With these changes, let's go and take a look at the web page,
at end of this exercise.
Going to our web page,
our web page is already starting to look even better now,
so you can now see that the Jumbotron is now styled with a new background color,
which is a little dark purple there,
and the lettering is now in floral white color,
little bit indented there,
then you can now see that the content in the content rows are now vertically centered,
for the three content rows.
This is the use of the align-items-center that we applied to the rows there.
Then going to the footer,
you can now see that this copyright is now justified to the center of the screen.
Look at how the links here have been centered.
I realized that I want to also position this vertically centering to this particular row,
so let's go and apply one more class to move this to
the center of this particular row vertically.
So to do that, going back to index.
html, in the footer for that particular column which contains those,
I will apply the class as align-self-center.
This should be applied to the div which uses the column class.
With this, that particular content will be aligned to the center of the row.
The remaining content will still remain at the top.
Let's take a final look at the footer.
Going to the footer in our web page,
you can now see how this has been positioned vertically in the center.
These two pieces of content are still aligned to the top of the row,
that is fine because that looks okay on in there,
but this one I have dragged it down vertically to the center of this particular row.
With this, we complete this exercise.
In this exercise, we saw the use of custom CSS classes,
and also used some of the classes for
justifying the content horizontally and vertically in our rows.
This is a good time for you to do a good comment with the message,
Bootstrap grid, part two.

From <https://www.coursera.org/learn/bootstrap-4/lecture/FLjFv/exercise-video-responsive-design-and-bootstrap-grid-system-part-2>

Exercise (Instructions): Responsive Design and Bootstrap Grid System Part 2

Practice with Lab Sandbox

Exercise: Responsive Design and Bootstrap Grid System Part 2

Objectives and Outcomes

This exercise continues the examination of responsive design and Bootstrap support for mobile first responsive design through the use of the grid system. We also learn how to customize some of the Bootstrap classes through defining our own modifications in a separate CSS file. At the end of this exercise, you will be able to:

- Customize the CSS classes through your own additions in a separate CSS file
- Centering the content both vertically and horizontally within a row

List styles

- You can use several list styles to display lists in different formats. In this exercise, we will use the unordered list style *list-unstyled* to display the links at the bottom of the page without the bullets. To do this, go to the links in the footer and update the ul as follows

```
<ul class="list-unstyled"> ... </ul>
```

Using Custom CSS classes

We can define our own custom CSS classes in a separate CSS file, and also customize some of the built-in CSS classes. We will now attempt to do this in this part of the exercise.

- Create a folder named ***css***. Then create a file named *styles.css* in the ***css*** folder. Open this file to edit the contents. Add the following CSS code to the file:

```
.row-header{
    margin:0px auto;
    padding:0px;
}

.row-content {
    margin:0px auto;
    padding: 50px 0px 50px 0px;
    border-bottom: 1px ridge;
    min-height:400px;
}

.footer{
    background-color: #D1C4E9;
    margin:0px auto;
    padding: 20px 0px 20px 0px;
}
```

- Include the *styles.css* file into the head of the *index.html* file as follows:

```
    <link href="css/styles.css" rel="stylesheet">
```

- Then add these classes to the corresponding rows in the *index.html* file as follows. See the difference in the *index.html* file in the browser. The first one is for the row in the <header>, the next three for the rows in the content, and the last one directly to the <footer> tag.

```
    <div class="row row-header"> ... </div>

    <div class="row row-content"> ... </div>

    <div class="row row-content"> ... </div>

    <div class="row row-content"> ... </div>

    <footer class="footer"> ... </footer>
```

- Our next set of customization is to the jumbotron and the address. Add the following to *styles.css* file:

```
.jumbotron {
    padding:70px 30px 70px 30px;
    margin:0px auto;
    background: #9575CD ;
    color:floralwhite;
}

address{
    font-size:80%;
    margin:0px;
    color:#0f0f0f;
}
```

Vertically Centering the Content

- In the content section, update all the rows as follows:

```
        <div class="row row-content align-items-center">
```

- In the footer, update the third column div that contains the social media links as follows:

```
                <div class="col-12 col-sm-4 align-self-center">
```

Horizontally Centering the Content

- Update the copyright paragraph as follows:

```
            <div class="row justify-content-center">
                <div class="col-auto">
```

- Update the inner div containing the social media links as follows:

```
                    <div class="text-center">
```

- After saving all the changes, you can do a Git commit with the message "Bootstrap Grid Part 2" and push your changes to the online repository.

Conclusion

In this exercise, we continued our review of responsive design and the Bootstrap grid system. We also learnt how to customize using our own CSS classes.

From <https://www.coursera.org/learn/bootstrap-4/supplement/OPtYt/exercise-instructions-responsive-design-and-bootstrap-grid-system-part-2>

Assignments

Wednesday, August 31, 2022 5:14 PM

Peer-graded Assignment: Assignment 1: Bootstrap and Responsive Design

Ready for the assignment?
You will find instructions below to submit.

Coursera Lab Sandbox

BETA

- Easily launch Coursera's preconfigured environment for Bootstrap 4
- Get access to all dependencies (libraries and packages) for VSCode—no local software installation required
- Practice Bootstrap 4, run test cases, and work on assignments from your browser

Open Lab Sandbox

1. **Instructions**
2. **My submission**
3. **Discussions**

In this assignment you will add a second page, *aboutus.html*, to your website. You will make use of the Bootstrap skills learnt in this module to prepare this web page for integration into the website.

Assignment Resources

aboutus.html
ZIP File

Objectives and Outcomes

In this assignment, you will continue to work with the website that you have been developing in the exercises. You will add the ***About Us*** web page to the website. To get you started, you are provided with a partially formatted *aboutus.html.zip* file given above that you need to download, unzip and move the *aboutus.html* to the conFusion folder that contains your website. At the end of this assignment, you should have completed the following tasks:

- Updated the page to make use of Bootstrap classes and Bootstrap grid
- Formatted the contents of the web page using the container, row and column classes
- Use the responsive utilities (hidden-* classes) to enable hiding of the detailed descriptions in the extra small screen size devices

Assignment Requirements

This assignment requires you to complete the following tasks. Detailed instructions for each task are given below. The picture of the completed web page included below indicates the location within the web page that will be updated by the three tasks.

Task 1

In this task you will be updating the *aboutus.html* page to make use of the Bootstrap classes and components:

- Update the page to make use of the Bootstrap CSS classes.
- Update the page to also use your custom styles defined in your *styles.css* file, and
- Update the page to make use of all the Bootstrap JS components.

Task 2

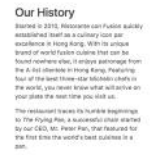
In this task you will be adding appropriate formatting to the web page contents using container, row and column classes using the Bootstrap grid so that the web page is formatted to look like the figure given below.

- The "About Us" title should stretch the entire width of the row.
- The "Our History" part should occupy only half the width of the row for small to extra large screens, leaving space on the right side for more content to be added later. The content should be stacked for extra small screens.
- The "Corporate Leadership" section should stretch the entire width of the row.

Task 3

In this task you will use some responsive utilities provided by Bootstrap to hide some of the content only for extra small screens. You will make use of the *d-none* and *d-sm-block* CSS classes provided

by Bootstrap. To understand how to use these classes, please read the documentation here (in particular see how the combination of classes shown here enables you to hide the content for xs screen sizes) to learn how to apply the *d-none* and *d-sm-block* classes. This will get you into the habit of consulting the Bootstrap documentation whenever you need to learn more about the various components and classes of Bootstrap. You should apply the classes so that the <p> elements containing the detailed descriptions of the corporate leadership is hidden only for extra small screens. Thus, your page should look like the figure below on extra small screens.



More details about the *d-none* and *d-sm-block* CSS classes can be found at http://getbootstrap.com/docs/4.0/utilities/display/.

While you are at it, please also apply the same classes to the descriptions in the *index.html* page. This is not part of the assignment, but should be completed to update your website.

Review criteria

less

Upon completion of the assignment, your submission will be reviewed based on the following criteria:

Task 1:

- The page is update to correctly use the Bootstrap CSS classes
- The page is updated correctly to use the custom CSS classes from styles.css
- The page has been updated to use the all the necessary JavaScript classes

Task 2:

- The container class has been applied to the content at the correct location.
- Row class, including the row-content class has been applied to the rows of the content. Do not apply row-content to the row containing the page heading
- Column classes have been appropriately applied to the content within each row to provide responsive layout of the content.

Task 3:

- The d-none and d-sm-block classes are correctly applied to the content in the Corporate Leadership section to hide the detailed description of the corporate leaders.

You are required to include two full-page screenshots of your completed web pages, one for normal screen size, and one for extra small screens. To take a full-page screenshot of your page use the Chrome extension: Full Page Screen Capture.

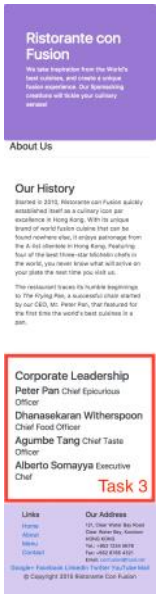
From <https://www.coursera.org/learn/bootstrap-4/peer/e9axj/assignment-1-bootstrap-and-responsive-design>

Assignment 1 Resources

Assignment 1 Starter Files

aboutus.html
ZIP File

Assignment 1 Screenshots



Bootstrap Resources

- Bootstrap grid
- Bootstrap display utilities (documentation here about the d-none and d-sm-block classes)

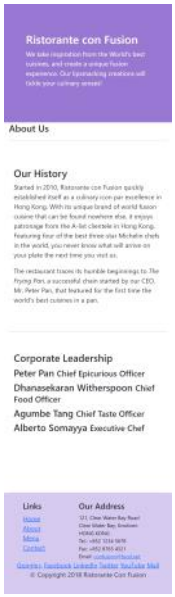
Chrome extension

- Full Page Screen Capture

From <https://www.coursera.org/learn/bootstrap-4/supplement/ydbEc/assignment-1-resources>

Assignments submitted:

aboutus_full page_nor...



Ristorante con Fusion

We take inspiration from the World's best cuisines, and create a unique fusion experience. Our lipsmacking creations will tickle your culinary senses!

About Us

Our History

Started in 2010, Ristorante con Fusion quickly established itself as a culinary icon par excellence in Hong Kong. With its unique brand of world fusion cuisine that can be found nowhere else, it enjoys patronage from the A-list clientele in Hong Kong. Featuring four of the best three-star Michelin chefs in the world, you never know what will arrive on your plate the next time you visit us.

The restaurant traces its humble beginnings to *The Frying Pan*, a successful chain started by our CEO, Mr. Peter Pan, that featured for the first time the world's best cuisines in a pan.

Corporate Leadership

Peter Pan Chief Epicurious Officer

Our CEO, Peter, credits his hardworking East Asian immigrant parents who undertook the arduous journey to the shores of America with the intention of giving their children the best future. His mother's wizardy in the kitchen whipping up the tastiest dishes with whatever is available inexpensively at the supermarket, was his first inspiration to create the fusion cuisines for which *The Frying Pan* became well known. He brings his zeal for fusion cuisines to this restaurant, pioneering cross-cultural culinary connections.

Dhanasekaran Witherspoon Chief Food Officer

Our CFO, Danny, as he is affectionately referred to by his colleagues, comes from a long established family tradition in farming and produce. His experiences growing up on a farm in the Australian outback gave him great appreciation for varieties of food sources. As he puts it in his own words, *Everything that runs, wins, and everything that stays, pays!*

Our CFO, Danny, as he is affectionately referred to by his colleagues, comes from a long established family tradition in farming and produce. His experiences growing up on a farm in the Australian outback gave him great appreciation for varieties of food sources. As he puts it in his own words, *Everything that runs, wins, and everything that stays, pays!*

Agumbe Tang Chief Taste Officer

Blessed with the most discerning gustatory sense, Agumbe, our CTO, personally ensures that every dish that we serve meets his exacting tastes. Our chefs dread the tongue lashing that ensues if their dish does not meet his exacting standards. He lives by his motto, *You click only if you survive my lick.*

Alberto Somayya Executive Chef

Award winning three-star Michelin chef with wide International experience having worked closely with whos-who in the culinary world, he specializes in creating mouthwatering Indo-Italian fusion experiences. He says, *Put together the cuisines from the two craziest cultures, and you get a winning hit! Amma Mia!*

aboutus

Peer-graded Assignment: Assignment 2: Bootstrap CSS Components

Ready for the assignment?
You will find instructions below to submit.

Coursera Lab Sandbox

BETA

- Easily launch Coursera's preconfigured environment for Bootstrap 4
- Get access to all dependencies (libraries and packages) for VSCode—no local software installation required
- Practice Bootstrap 4, run test cases, and work on assignments from your browser

Open Lab Sandbox

4. **Instructions**
5. **My submission**
6. **Discussions**

Objectives and Outcomes

In this assignment, you will continue to work with the website that you have been developing in the exercises. You will edit the *home* page (*index.html*). You will start with the current home page at the end of the last exercise in this module. At the end of this assignment, you should have completed the following tasks:

- Designed a form to enable users to submit a reservation request for a table. Note that at this stage the form will be inactive. This form should have been included in a new content row that you create just before the footer of the page.
- Formatted the contents of the second row of the page using media class. The content column of the row should have been converted to a media object. In addition it should include a badge.
- Added a button to the Jumbotron to enable users to access the form to reserve a table at the restaurant. Clicking on this button should take you to the reservation form at the bottom of the page.

Assignment and Requirements

This assignment requires you to complete the following three tasks. Detailed instructions for each task are given below. The picture of the completed web page included below indicates the location within the web page that will be updated by the three tasks.

Task 1

In this task you will be adding another content row to the page. The content row should contain the following:

- You should create a reservation form for the user to reserve a table. The reservation form should contain a field using radio that enables the users to specify the number of guests (1-6).
- The form should contain fields to specify the date and time of the reservation. The fields should contain appropriate placeholder information to identify the purpose of the fields.
- The form should contain a button named Reserve to initiate reservation of the table.
- The form should be enclosed inside a card with the heading "Reserve a Table". The card should occupy 8 columns and centred in the row for small to extra-large screens. For extra-small screens, the card should span the entire row.

Task 2

- The form should be enclosed inside a card with the heading "Reserve a Table". The card should occupy 8 columns and centred in the row for small to extra-large screens. For extra-small screens, the card should span the entire row.

Task 2

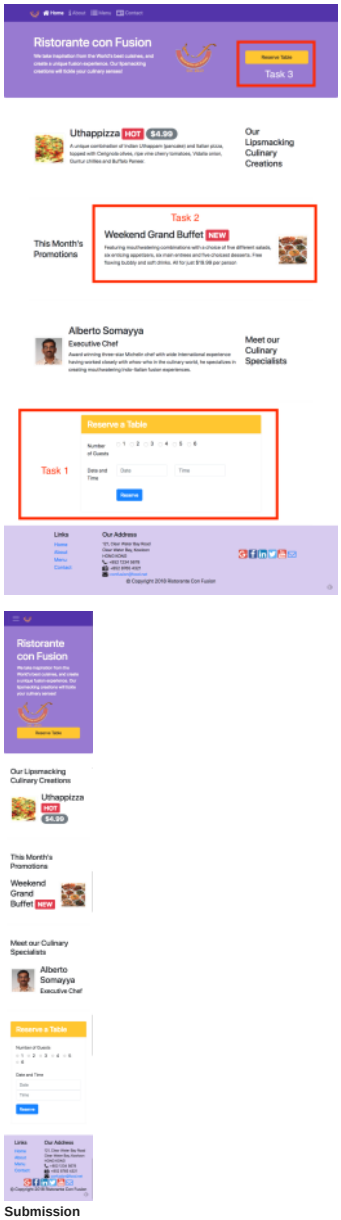
In this task you will be formatting the content in the second row of the page. The formatting should result in the following:

- Format the content of the second column with the media class together with the media object class. Use the *buffet.png* image file provided for you in the *img* folder. The image should displayed to the right of the content description. See figure below.
- Add a badge with the word “NEW” to the content as shown in the figure below.

Task 3

In this task you will be adding a block-sized button to the Jumbotron to the right of the restaurant logo:

- The block-level button and the restaurant logo should share the right six columns of the row. The restaurant name and description can now be reduced to occupy the left six columns. Use the small button (btn-sm).
- Clicking on the button should take you down to the form for reserving a table.



Submission

1. You should submit the updated *index.html* file with all the tasks completed. A reviewer should easily be able to take your file and substitute it into their own web project and see it working correctly.
2. Also upload a screenshot of your browser window showing the completed *index.html* page in png or jpg format.

Review criteria

less

Upon completion of the assignment, your submission will be reviewed based on the following criteria:

Task 1

less

Upon completion of the assignment, your submission will be reviewed based on the following criteria:

Task 1

- The new content row is correctly formatted and includes the reservation form and the “Reserve a Table” card header.
- The form contains the radios to enable specification of the number of guests.
- The form includes a date field
- The form includes a time field
- The form contains a reserve button.

Task 2

- The content in the row has been correctly formatted using the media class
- The image is displayed to the right of the content using the correct media-* classes and at the correct position.
- The badge is correctly displayed in the second row.

Task 3

- The Reserve Table button is correctly included in the Jumbotron and is a block size button.
- The button is correctly enclosed inside a div with the correct column specification.
- The reserve button when clicked takes us to the form. The link in the button should be set up correctly to take us to the form.

From <https://www.coursera.org/learn/bootstrap-4/peer/omyzt/assignment-2-bootstrap-css-components>

index

Peer-graded Assignment: Assignment 3: Bootstrap JavaScript Components

Ready for the assignment?
You will find instructions below to submit.

Coursera Lab Sandbox

BETA

- Easily launch Coursera's preconfigured environment for Bootstrap 4
- Get access to all dependencies (libraries and packages) for VSCode—no local software installation required
- Practice Bootstrap 4, run test cases, and work on assignments from your browser

Open Lab Sandbox

7. **Instructions**
8. **My submission**
9. **Discussions**

In this assignment you will remove the tooltip from the Reserve table button. Then you will move the table reservation form into a modal that will be shown when the Reserve Table button is clicked. The updated reservation form will include a new radio button group allowing you to select the smoking/non-smoking section of the restaurant.

Objectives and Outcomes

In this assignment, you will continue to work with the website that you have been developing in the exercises. You will edit the home page (index.html). You will start with the current home page at the end of the last exercise in this module. At the end of this assignment, you should have completed the following tasks:

- Moved the table reservation form from the last content row into a modal.
- Included a radio button group in the table reservation form to enable diners to ask for a table in the smoking/non-smoking section of the restaurant.
- Removed the tooltip from the Reserve Table button.
- Updated the Reserve Table button to show the modal containing the table reservation form when the button is clicked.

Assignment Requirements

This assignment requires you to complete the following four tasks. Detailed instructions for each task are given below. The picture of the completed web page included below indicates the location within the web page that will be updated by the four tasks.

Task 1

In this task you will move the table reservation form from the last content row into a modal. You should also remove the last content row.

- The form should be completely shifted to a modal.
- Add a Cancel button in the form that will dismiss the modal when clicked.
- The modal header should contain a X button to dismiss the modal.

Task 2

In this task you will be adding a radio button group to the form to allow the selection of the smoking/non-smoking section of the restaurant.

- The radio button group should start out with the non-smoking section selected by default.
- The row containing the button group will have the label Section displayed preceding it in the form.

Note: Read Bootstrap Buttons Checkbox/Radio for more information on how to design checkbox/radio buttons.

- The row containing the button group will have the label Section displayed preceding it in the form.

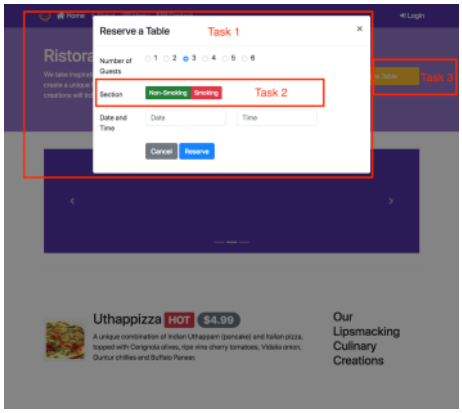
Note: Read Bootstrap Buttons Checkbox/Radio for more information on how to design checkbox/radio buttons.

Task 3

In this task you will be updating the Reserve Table button in the Jumbotron:

- Remove the tooltip from the button. This is to facilitate the button to be used to trigger the modal containing the table reservation form in the later tasks. A single button can support only one Javascript plugin via the data-* attributes. Make sure to remove the JavaScript script at the bottom of the page. Also remove the corresponding JavaScript code for the tooltip from the bottom of the page.
- You will update the Reserve Table button to show the modal containing the table reservation form when the button is clicked.

At the end of this assignment, your *index.html* file should look like this:



Submission

1. You should submit the updated index.html file with all the tasks completed. A reviewer should easily be able to take your file and substitute it into their own web project and see it working correctly.
2. Also upload a screenshot of your browser window showing the completed index.html page with the modal containing the table reservation form overlayed on top of the web page in png or jpg format.

Review criteria

less

Upon completion of the assignment, your submission will be reviewed based on the following criteria:

Task 1:

- The form has been moved into a modal.
- The modal HTML code is included towards the top of the body of the page near the other modal code.
- The modal includes a Cancel button to dismiss the modal.
- The modal header includes a X button to dismiss the modal.

Task 2:

- A correct radio button group with the labels non-smoking in green and smoking in red is included in the form. Use the correct button color classes for the buttons.
- The label of the row containing the buttons is set to Section.
- The non-smoking button is checked by default.

Task 3:

- The tooltip has been removed from the Reserve Table button and the corresponding JavaScript code has been removed from the bottom of the page.
- The Reserve Table button will show the modal containing the table reservation form when clicked.

From <https://www.coursera.org/learn/bootstrap-4/peer/aRDkz/assignment-3-bootstrap-javascript-components>

index

Peer-graded Assignment: Assignment 4: Bootstrap, JQuery and Sass

Review fellow learners

Congrats on submitting your assignment! Your peers can now review it and give you constructive feedback. You can do your part and help other learners complete this course by giving reviews.
Review assignments

Coursera Lab Sandbox

BETA

- Easily launch Coursera's preconfigured environment for Bootstrap 4
- Get access to all dependencies (libraries and packages) for VSCode—no local software installation required
- Practice Bootstrap 4, run test cases, and work on assignments from your browser

Open Lab Sandbox

10 **Instructions**

- Get access to all dependencies (libraries and packages) for VSCode–no local software installation required
- Practice Bootstrap 4, run test cases, and work on assignments from your browser

Open Lab Sandbox

10. **Instructions**
11. **My submission**
12. **Discussions**

In this assignment you will use the JavaScript methods to control the showing and hiding of the modals in the index.html page. You will replace the data-* attributes of the buttons and instead add JavaScript method to control the modals. In addition, you will write some SCSS code to style the modal background.

Step-By-Step Assignment Instructions

less

Objectives and Outcomes

In this assignment, you will continue to work with the website that you have been developing in the exercises. You will edit the home page (index.html), the JavaScript code (scripts.js) and the SCSS code (styles.scss). You will start with the current home page at the end of the last exercise in this module. At the end of this assignment, you should have completed the following tasks:

- Removed the data-* attributes from the Reserve Table button and the Login link in the Navbar that control the two modals.
- Updated the button and the link so that they will trigger the appropriate JavaScript code when clicked.
- Included appropriate JavaScript code using the modal methods to toggle the showing and hiding of the modals when the two buttons are clicked.
- Add SCSS code to style the modal with colors.

Assignment Requirements

This assignment requires you to complete the following **four** tasks. Detailed instructions for each task are given below.

Task 1

In this task you will be removing the data-* attributes from the *Reserve Table* button in the Jumbotron that enable the toggling of the Reserve Table modal. Similarly you will also remove the data-* attributes from the Login link in the Navbar that triggers the Login modal.

Task 2

In this task you will add appropriate JavaScript code to the page so that the *Reserve Table* modal will be toggled when the Reserve Table button is clicked:

- You will add the appropriate JavaScript code to the script.
- You will update the button so that the modal is triggered when the button is clicked.
- Note: Read Bootstrap Modal Methods for more information on how to implement the JavaScript methods.

Task 3

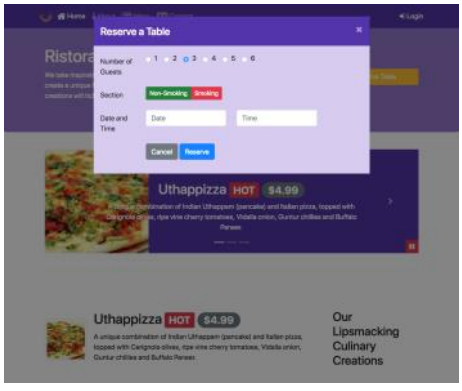
In this task you will add appropriate JavaScript code to the page so that the Login modal will be toggled when the *Login* link is clicked:

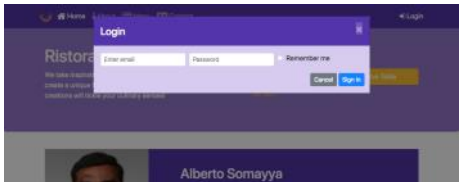
- You will add the appropriate JavaScript code to the script.
- You will update the Login link so that the modal is triggered when the link is clicked.

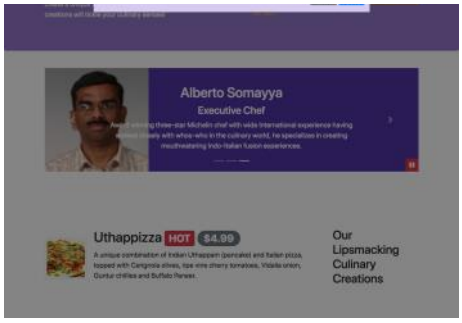
Task 4

In this task you will add appropriate SCSS code to styles.scss whereby the colors of the Modal are appropriately styled. These colors are already set up in the styles.scss file.

- You will set the modal header to dark background and change the text color to floralwhite, and use nesting to change the color of the close button to floralwhite.
- You will set the modal body to the pale background







Submission

- You should submit the updated index.html file and styles.scss with all the tasks completed. A reviewer should easily be able to take your files and substitute it into their own web project and see it working correctly.

Review criteria

less

Upon completion of the assignment, your submission will be reviewed based on the following criteria:

Task 1:

- The data-* attributes have been removed from the *Reserve Table* button.
- The data-* attributes have been removed from the *Login* link.

Task 2:

- Appropriate JavaScript code has been added to the page to trigger the *Reserve Table* modal.
- The *Reserve Table* button has been appropriately updated to enable the triggering of the modal.

Task 3:

- Appropriate JavaScript code has been added to the page to trigger the Login modal.
- The *Login* link has been appropriately updated to enable the triggering of the modal.

Task 4:

- Appropriate SCSS code has been added to styles.scss to set the modal header and body background colors appropriately.
- Set up the modal header text color to floralwhite.
- Set up the close button color to floralwhite.

From <https://www.coursera.org/learn/bootstrap-4/peer/FNIDB/assignment-4-bootstrap-jquery-and-sass>

index

styles

styles

styles

Bootstrap CSS Components

Thursday, September 1, 2022 6:50 PM

This module concentrates on Bootstrap's components that are designed using pure CSS classes. You will learn about Navigation and Navigation bar. Then, you will learn about buttons, forms, tables, cards, images and media, tags, alerts and progress bars. At the end of this module you need to complete your second assignment.

Learning Objectives

- Express the need for including navigation in your website
- Build navigation features into your website
- Build data presentation using tables and cards
- Prepare media to be included into your website
- Build means for providing user interaction with your website using forms and buttons

From <https://www.coursera.org/learn/bootstrap-4/home/week/2>

Navigation and Navigation Bar: Objectives and Outcomes

In this lesson, you will be given an overview of navigation design and the importance of providing appropriate navigation support within your website. You will learn about support for navigation design elements available in Bootstrap, including the Navbar and Breadcrumbs. Other navigation aids will be covered in subsequent modules. In addition, the use of icon fonts in web page design will be covered. The exercises will concentrate on adding a responsive navigation bar to the website. At the end of this lesson, you will be able to:

- Understand the need for navigation support in a web project
- Use the Bootstrap navigation features including the Navbar and breadcrumbs in providing navigation support in websites
- Use icon fonts for decorating your website with meaningful graphical elements

From <https://www.coursera.org/learn/bootstrap-4/supplement/xKBat/navigation-and-navigation-bar-objectives-and-outcomes>

Exercise (Instructions): Navbar and Breadcrumbs

Objectives and Outcomes

In this exercise, we will examine the navigation support that we can build into a web page using the Navbar in Bootstrap. At the end of this exercise, you will be able to:

- Create a navigation structure for your website using the Navbar
- Add breadcrumbs to the website
- Include additional CSS classes into your project

Create a basic navigation bar

- We will now add a simple navigation bar to the web page so that it provides links to the other pages on the website. Start by adding the following code to the body just above the header jumbotron.

```
    <nav class="navbar navbar-dark navbar-expand-sm bg-primary fixed-top">
        <div class="container">
           <a class="navbar-brand" href="#">Ristorante Con Fusion</a>
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active"><a class="nav-link" href="#">
Home</a></li>
                    <li class="nav-item"><a class="nav-
link" href="./aboutus.html">About</a></li>
                    <li class="nav-item"><a class="nav-link" href="#">Menu</a>
</li>
                    <li class="nav-item"><a class="nav-link" href="#">Contact</a>
</li>
                </ul>
        </div>
    </nav>
```

In the above code, we can see the use of the nav element to specify the navigation information for the website. This nav element is styled by the *navbar* that declares it as a navigation bar, and the *navbar-dark* class to specify that the page should use the dark navigation bar. You will now notice the addition of a link with the name of the restaurant. This is the brand name for the website. You can replace this with the logo for the website. This is created by the *<a class="navbar-brand">* tag. In addition the inner *ul* is used to specify the items to be put in the navigation bar. This *ul* is styled with *navbar-nav* class to specify that the items should be displayed inline inside the navigation bar. We also use the container class inside the navigation bar.

Creating a responsive navigation bar

- We would like the navigation bar elements to collapse for shorter screens, to be replaced by a toggle button so that the items can be toggled on or off when required on small and extra small screens. This can be achieved by adding the following code to the navigation bar, just below the container div

```
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#Navbar">
```

```
                <span class="navbar-toggler-icon"></span>
            </button>
```

This creates a button with three horizontal lines. For medium to extra large screens, this button is hidden. For small and extra small screens, this button becomes visible. This button will act as the toggle for the navbar items.

- To hide the items from the navigation bar for the small screens, we need to enclose the *ul* within another div as follows:

```
            <div class="collapse navbar-collapse" id="Navbar">
                <ul class="navbar-nav mr-auto"> ... </ul>
            </div>
```

By doing this, we are specifying that this div with *collapse* and *navbar-collapse* classes and with the id *Navbar* will be collapsed on small and xs screens, but can be toggled on or off when the toggle button is clicked. Note the use of *data-toggle="collapse" data-target="#Navbar"* within the button above. This specifies that the menu items are collapsed on small and xs screens when the toggle button is visible. They can be displayed or hidden by clicking the toggle button.

- Copy and paste the entire navbar code also into *aboutus.html* to add the navigation also to that page. Make sure to change the <li> corresponding to "About" to *active*, and remove the *active* class from the Home link. Also, update the home link to take you back to *index.html*. Update the navbar-brand tag also to take you back to *index.html*.

Modifications to the CSS styles

- We would like to have the navigation bar displayed in darker purple color, instead of the current color. In addition, when we use the fixed navigation bar, we should give the body of the page an upper margin of 50px, so that the top 50px of the page does not get hidden under the navigation bar. We accomplish these by adding these CSS customisations to the *styles.css* file

```
body{
    padding:50px 0px 0px 0px;
    z-index:0;
}

.navbar-dark {
     background-color: #512DA8;
}
```

- Remember to delete the *bg-primary* class from the <nav> element in both *index.html* and *aboutus.html*.
- We are already beginning to see the page format close to the final format for this module.

Adding Breadcrumbs

- To add breadcrumbs to our pages, we take the help of the breadcrumb and breadcrumb-item classes to add the following to the row containing the About Us title in *aboutus.html*.

```
<ol class="col-12 breadcrumb">
    <li class="breadcrumb-item"><a href="./index.html">Home</a></li>
    <li class="breadcrumb-item active">About Us</li>
</ol>
```

- Save all the changes and commit to your Git repository with a message "Navbar and Breadcrumbs".

Conclusions

In this exercise you learnt about adding a navigation bar and breadcrumbs to your website.

From <https://www.coursera.org/learn/bootstrap-4/supplement/39PTK/exercise-instructions-navbar-and-breadcrumbs>

Let us now talk about icon fonts.
What are they?
How are they useful?
And how do we make use of them in designing our website?
Icon fonts provide a very flexible way of including tiny
images into our web pages that can be styled just like text.
So that is the reason icon fonts prove very,
very useful in designing websites.
Using icon fonts, you can take a web page like this and
quickly turn it into a web page like this.
Note the use of icons together with the text in the nav bar,
and then when you scroll down to the footer,
you can see more icons being used in the footer.
And also your social media links are now replaced by social media buttons here.
So these are the changes that icon fonts will enable you to do to your website.
So as I briefly mentioned earlier, icon fonts are a set of symbols or
glyphs that can be used in your website design.
These can be used just like regular fonts,
just like regular text fonts that you use in your website.
Their advantage is that you can style them and
then expand and contract them and use all typical
stylings that you use on your text, for your icons also.
They are seen as a popular lightweight replacement for
simple graphics that you can use in your web pages,
simply graphics or images that you can use in your web pages.
And they are very, very useful in designing your web pages.
There are many icon font packs that are available in the market.
One of the most popular ones is Font Awesome,
which I'm going to talk about next.
Play video starting at :2:20 and follow transcript2:20
Font Awesome, as I say, is a very popular icon font.
It is available free for use in designing your page design.
It has been extensively used by many different websites.
We'll learn how to download Font Awesome and
then use it in web page in the exercise that follows this lecture.
Play video starting at :2:44 and follow transcript2:44
Using Font Awesome in your webpage is as simple as
using some classes that are applied to an i,

or a span tag in your web page HTML code.
So once you include the CSS and
the font files that are available through Font Awesome,
then it is very straightforward to make use of Font Awesome in your pages.
Play video starting at :3:15 and follow transcript3:15
Another common feature that you would see on web
pages is the use of social site buttons there like Facebook,
Twitter, Google+, and YouTube and many others.
Play video starting at :3:34 and follow transcript3:34
Now fortunately for us, there is this other CSS
file called bootstrap-social that you can download and
make use of, including various social
media site buttons into your web page.
We will see the use of bootstrap-social in the exercise that follows.
Once you download the bootstrap-social css file, you include that in your web page.
And then together with the Font Awesome icons,
you can combine that with bootstrap social classes
to include various social media buttons on your web pages.
So without further ado, let's move to the next exercise where
we're going to use both Font Awesome and bootstrap-social,
a modified version of bootstrap-social.css file that I
provide on the exercise instructions that you should download and
include in your confusion project and make use of it in the exercise.

From <https://www.coursera.org/learn/bootstrap-4/lecture/0TfhV/icon-fonts>

Exercise (Instructions): Icon Fonts

Objectives and Outcomes

In this exercise, we will learn the use of icons in web page design using Font Awesome icons, and bootstrap-social icons. At the end of this exercise, you will be able to:

- Use icons within your website to represent various entities making use of the glyphicons, font-awesome icons and bootstrap-social icons
  Note: Some people have pointed out that if they have AdBlocker installed, then the font icons are not showing up in their browser.

Using Icon Fonts and Other CSS classes

- One of the most popular icon font toolkit is Font Awesome. Go to its website http://fontawesome.io/ to check out more details about this icon font. You can get Font Awesome using npm by typing the following at the prompt:

```
npm install font-awesome@4.7.0 --save
```

- Another module that we install is Bootstrap Social that enables the addition of Social buttons to our site. You can find more information about it at https://lipis.github.io/bootstrap-social/. To install it using npm, type the following at the prompt:

```
npm install bootstrap-social@5.1.1 --save
```

- We now need to include the CSS files for font awesome and bootstrap-social in the index.html file. Add the following code to the head of the file after the links for importing Bootstrap CSS classes. Do the

same change to aboutus.html file:

```
    <link rel="stylesheet" href="node_modules/font-awesome/css/font-awesome.min.css">
    <link rel="stylesheet" href="node_modules/bootstrap-social/bootstrap-social.css">
```

- Let us now use some font icons in our web page and decorate it. Update the navbar's ul list items as follows in index.html:

```
                    <li class="nav-item active"><a class="nav-link" href="#"><span class="fa fa-home fa-lg"></span> Home</a></li>
                    <li class="nav-item"><a class="nav-link" href="./aboutus.html"><span class="fa fa-info fa-lg"></span> About</a></li>
                    <li class="nav-item"><a class="nav-link" href="#"><span class="fa fa-list fa-lg"></span> Menu</a></li>
                    <li class="nav-item"><a class="nav-link" href="#"><span class="fa fa-address-card fa-lg"></span> Contact</a></li>
```

- Similarly update the navbar's ul list items as follows in aboutus.html:

```
                    <li class="nav-item"><a class="nav-link" href="./index.html"><span class="fa fa-home fa-lg"></span> Home</a></li>
                    <li class="nav-item active"><a class="nav-link" href="#"><span class="fa fa-info fa-lg"></span> About</a></li>
                    <li class="nav-item"><a class="nav-link" href="#"><span class="fa fa-list fa-lg"></span> Menu</a></li>
                    <li class="nav-item"><a class="nav-link" href="#"><span class="fa fa-address-card fa-lg"></span> Contact</a></li>
```

- Next, in both index.html and aboutus.html, go down to the address in the footer of the page and replace the "Tel.", "Fax" and "Email" with the corresponding font awesome based icons as follows:

```
                    <i class="fa fa-phone fa-lg"></i>: +852 1234 5678<br>
                    <i class="fa fa-fax fa-lg"></i>: +852 8765 4321<br>
                    <i class="fa fa-envelope fa-lg"></i>:
                    <a href="mailto:confusion@food.net">confusion@food.net</a>
```

- Finally, let us use the bootstrap-social CSS classes to create the social buttons in the footer in both index.html and aboutus.html, by replacing the social sites' links with the following code:

```
                    <div class="text-center">
                        <a class="btn btn-social-icon btn-
google" href="http://google.com/+"><i class="fa fa-google-plus"></i></a>
                        <a class="btn btn-social-icon btn-
facebook" href="http://www.facebook.com/profile.php?id="><i class="fa fa-
facebook"></i></a>
                        <a class="btn btn-social-icon btn-
linkedin" href="http://www.linkedin.com/in/"><i class="fa fa-linkedin"></i></a>
                        <a class="btn btn-social-icon btn-
twitter" href="http://twitter.com/"><i class="fa fa-twitter"></i></a>
                        <a class="btn btn-social-icon btn-
google" href="http://youtube.com/"><i class="fa fa-youtube"></i></a>
                        <a class="btn btn-social-icon" href="mailto:">
<i class="fa fa-envelope-o"></i></a>
                    </div>
```

- Save all the changes and commit to your Git repository with the message "Icon Fonts".

Conclusions

We learnt about using icon fonts in a web project.

From <https://www.coursera.org/learn/bootstrap-4/supplement/36yqj/exercise-instructions-icon-fonts>

User Input: Buttons and Forms: Objectives and Outcomes

In this lesson we review the support for user input through the use of buttons and forms in a web page. We review Bootstrap button classes and Forms classes. At the end of this lesson you will be able to:

- Create and style buttons on a web page using Bootstrap button classes
- Create and style forms on a web page using Bootstrap form classes

From <https://www.coursera.org/learn/bootstrap-4/supplement/TZ8Au/user-input-buttons-and-forms-objectives-and-outcomes>

Gone are the days when websites were purely used for
delivering information to the users.
These days on most websites, users will be able to interact with the website to
supply information, for example by clicking buttons or by filling in forms
and by typing something into search boxes, and so on.
So how do we support these kinds of interactions with the users?
So that is what we're going to deal with in this particular lecture and
the exercise that follows.
As an example, if you visit the Coursera website, you will see that you have
buttons at the top here that you could click to reach different places.
You could have a search box, into which you can type in information,
to search for, say for example, full stack web development and, so on.
So, what we notice is that user interaction needs to be supported
on websites using many different approaches including buttons,

forms, text boxes, check boxes, and many others.
Early interactions with websites were provided mainly through hyperlinks.
So you could click on a hyperlink and go on to other places and so on, but
this is obviously,
hyperlink is just one of the many methods of interacting with your website.
You could have buttons included on the website which when clicked,
will result in some action being taken on the website.
You could have forms that you fill out to supply information to the website.
So when you look at how you would include such user
Interaction features into your website, you will see that you could use something
like the a tags that are useful for providing hyperlinks,
then you have the button tags that enable you to include buttons in your website.
Now what we're going to look at is how do we format with a tags on
the button tags using bootstrap classes so that you can style them
Play video starting at :2:20 and follow transcript2:20
to fake the general theme of bootstrap.
Play video starting at :2:25 and follow transcript2:25
HTML already includes the form elements and the input elements in there.
Now, we can look at how bootstrap enhances these elements by providing
styling features for forms, and various elements that go into the forms.
Buttons obviously provide a simple way of interacting with your website.
So when you have a button on your website, you might hover on the button,
you might click on the button and expect something to happen in the process.
The button behavior depends upon where it is positioned in bootstrap.
If the button is inside a form, for example,
typically the button clicking will result in the form being submitted to a server,
or the form information being canceled.
A button outside a form could have other ways of providing interaction.
Similarly, the < a > tag that we have traditionally associated with
Play video starting at :3:25 and follow transcript3:25
hyperlinks could also be hijacked to be styled and
presented in the form of a button.
So here we will see how bootstrap provides classes that can be
used to format the <a> tag into a button to be presented on your website.
We'll also look at various form elements, like the input elements,
select, the button and also the text AD elements, and
how we make use of them within our forms to design and
construct a form that can be included in our website.
So the exercise that follows this particular lecture will introduce you to
bootstrap support for styling buttons, and
also look at how various form elements can be enhanced using
bootstrap classes to present classic Bootstrap forms.

From <https://www.coursera.org/learn/bootstrap-4/lecture/3sQ7s/user-input>

Exercise (Instructions): Buttons

Practice with Lab Sandbox

Exercise: Buttons

Objectives and Outcomes

In this exercise, we will examine user input for a website through the use of Buttons support in

Bootstrap. At the end of this exercise, you will be able to:

- Create, style and activate buttons in a web page using the button classes
- Use a Button Group to group together related buttons.

Exercise Resources

contactus.html
ZIP File

Set up for the Exercise

- Download the *contactus.html.zip* file given above, unzip it and move the *contactus.html* to the ***conFusion*** folder. This file is already pre-formatted with some content.
- Set up the links in the navigation bars for all the three pages, *index.html, aboutus.html* and *contactus.html* so that we can navigate from one to the other with ease.
- Also set up the links in the footer correctly to point to the appropriate pages.

Adding a Button Bar

- We are now going to add content to *contactus.html* file to learn more about buttons and button bars. Go to the div where we specify "Button group goes here", and replace it with the following code to create a button bar containing three buttons:

```
                <div class="btn-group" role="group">
                    <a role="button" class="btn btn-primary" href="tel:+
85212345678"><i class="fa fa-phone"></i> Call</a>
                    <a role="button" class="btn btn-info"><i class="fa fa-skype">
</i> Skype</a>
                    <a role="button" class="btn btn-
success" href="mailto:confusion@food.net"><i class="fa fa-envelope-o">
</i> Email</a>
                </div>
```

Note how we define the button bar using the *btn-group* class, and then add the three buttons using the <*a*> tag. In this case, the three buttons are hyperlinks that cause an action and have an *href* associated with them. So we decided to use the <*a*> tag instead of the <*button*> tag. Note how the <*a*> tags have been styled using the *btn* class.

- Remember to do a Git commit with the message "Buttons"

Conclusions

We have learnt how to add buttons and button groups to a web page.

From <https://www.coursera.org/learn/bootstrap-4/supplement/zZFV1/exercise-instructions-buttons>

Exercise (Instructions): Forms

Practice with Lab Sandbox

Exercise: Forms

Objectives and Outcomes

In this exercise, we will examine user input for a website through the use of Forms support in Bootstrap. At the end of this exercise, you will be able to:

- Design a form using various form elements and style the form using Bootstrap form classes

Adding a Basic Form

- We will add a simple form to the page at the location identified by "Form goes here". Add the following code to page to create a simple horizontal form with two fields:

```
                <form>
                    <div class="form-group row">
                        <label for="firstname" class="col-md-2 col-form-label">
First Name</label>
                        <div class="col-md-10">
                            <input type="text" class="form-
control" id="firstname" name="firstname" placeholder="First Name">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label for="lastname" class="col-md-2 col-form-label">
Last Name</label>
                        <div class="col-md-10">
                            <input type="text" class="form-
control" id="lastname" name="lastname" placeholder="Last Name">
                        </div>
                    </div>
                </form>
```

This creates a form with two elements in the form. Note that the class *row* in the form enables us to use the Bootstrap grid system. Hence we can style the contents using the column classes as

appropriate.

- Let us add fields to seek user's telephone number and email:

```
                    <div class="form-group row">
                        <label for="telnum" class="col-12 col-md-2 col-form-
label">Contact Tel.</label>
                        <div class="col-5 col-md-3">
                            <input type="tel" class="form-
control" id="areacode" name="areacode" placeholder="Area code">
                        </div>
                        <div class="col-7 col-md-7">
                            <input type="tel" class="form-
control" id="telnum" name="telnum" placeholder="Tel. number">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label for="emailid" class="col-md-2 col-form-label">
Email</label>
                        <div class="col-md-10">
                            <input type="email" class="form-
control" id="emailid" name="emailid" placeholder="Email">
                        </div>
                    </div>
```

Adding a Checkbox and Select

- We now see the addition of a checkbox and a select element to the form. Note the styling of these elements using Bootstrap classes:

```
                    <div class="form-group row">
                        <div class="col-md-6 offset-md-2">
                            <div class="form-check">
                                <input type="checkbox" class="form-check-
input" name="approve" id="approve" value="">
                                <label class="form-check-label" for="approve">
                                    <strong>May we contact you?</strong>
                                </label>
                            </div>
                        </div>
                        <div class="col-md-3 offset-md-1">
                            <select class="form-control">
                                <option>Tel.</option>
                                <option>Email</option>
                            </select>
                        </div>
                    </div>
```

Adding a textarea

- Next we add a textarea for the users to submit their feedback comments as follows:

```
                    <div class="form-group row">
                        <label for="feedback" class="col-md-2 col-form-label">
Your Feedback</label>
                        <div class="col-md-10">
                            <textarea class="form-
control" id="feedback" name="feedback" rows="12"></textarea>
                        </div>
                    </div>
```

Adding the Submit Button

- Finally, we add the submit button to the form as follows:

```
                    <div class="form-group row">
                        <div class="offset-md-2 col-md-10">
                            <button type="submit" class="btn btn-primary">
Send Feedback</button>
                        </div>
                    </div>
```

Note the declaration of the type for the button to *submit*.

- Remember to do a Git commit with the message "Forms"

Conclusions

We have learnt how to add a form and style the form using Bootstrap form classes.

From <https://www.coursera.org/learn/bootstrap-4/supplement/Ow3tC/exercise-instructions-forms>

Displaying Content: Tables and Cards: Objectives and Outcomes

In this lesson we will be reviewing the support for tables in Bootstrap. In addition we will look at a versatile component called card that enables the display of content in myriad ways. At the end of this lesson you will be able to:

- Present and style tabular data in a table form using Bootstrap support for tables
- Display content using a card on a web page.

From <https://www.coursera.org/learn/bootstrap-4/supplement/1zyjj/displaying-content-tables-and-cards-objectives-and-outcomes>

Let's briefly discuss about bootstrap tables.
What does bootstrap bring to styling tables that we include in our web pages?
We'll also look at another bootstrap component called as bootstrap card,
which is a very versatile component for displaying content.
Play video starting at ::26 and follow transcript0:26
There may be many occasions where you want to present tabular content
on your web page.
Tables were essentially designed for
that purpose, but soon there after web page
designers hijacked tables into using for
designing and laying out content in the pages.
Because inherently, the original HTML did not have
the ability to properly lay out content in a web page.
Of course, with the arrival of grid-based layouts like the bootstrap grid,
the tables have been turned on tables.
And increasingly, people are preferring not to use tables
as a main sort of building content layout in web pages.
Grid-based layouts are better designed for this purpose.
Here is an example of what we can do to a standard HTML table
by using several classes that bootstrap provides for styling tables.
So here, you can see that the set apart by displaying
it in darker color and then the rows themselves are highlighted
by displaying alternate rows with different backups.
The table used in bootstrap based that page
is similar to how you use tables in standard HTML pages.
But of course, you apply the class table which is
supported in bootstrap to a standard table tech.
In addition, bootstrap provides many different classes for styling tables like,
table striped for zebra striped tables as we saw in the example earlier.

Similarly, we have table bordered for borders.
Tables, table-hover, table-sm for tables with cell padding cut in half.
And similarly, table-responsive can be applied to a div around tables, so
that the tables can be made responsive where it is shown on smaller screen sizes.
The tables will then be able to scroll horizontally and vertically.
Also, you could apply backgrounds to individual row of tables or
entire tables themselves.
Here are a couple of examples of how you could apply a background
color to a particular row or a particular cell in table.
Another versatile component that is available in bootstrap
is a bootstrap card.
Sometimes, you may want to highlight some content on your web page.
So this is where bootstrap comes come to your aid.
In Bootstrap card is a versatile component that allows you to
display content in myriads of ways.
You would notice that it allows you to have headers for cards.
You can also include images in cards.
You can include images in the background of cards.
You can even change the background colors of cards.
Many features are supported by cards.
Play video starting at :3:53 and follow transcript3:53
Here are a couple of examples of cards that we will see in the exercises later.
We will also see more versatile us of cards in later exercises and
other courses in this specialization.
Card is a very important company that can be effectively used for
displaying content.
It's time to move on to the next exercise where we will use tables and
cards to display some content in our web page.

From <https://www.coursera.org/learn/bootstrap-4/lecture/IgIYO/bootstrap-tables-and-cards>

Exercise (Instructions): Displaying Content: Tables and Cards

Objectives and Outcomes

In this exercise, we will examine tables and Bootstrap classes for styling tables. We will also examine Bootstrap cards and their use for displaying content. At the end of this exercise, you will be able to:

- Create, style and present tabular data in tables in a web page using the Bootstrap table classes
- Display content in a web page using Bootstrap cards

Set up for the Exercise

- In this exercise we will be modifying the *aboutus.html* page to add a table, a card with some content and a card with a quotation.
- Let us get started by opening *aboutus.html* page in a text editor.

Bootstrap Tables

- In this part, we will add a new row of content after the Corparate Leadership row in the page. We first start by adding a row and columns to the page as follows:

```
        <div class="row row-content">
            <div class="col-12 col-sm-9">
                <h2>Facts &amp;amp; Figures</h2>
            </div>
             <div class="col-12 col-sm-3">
            </div>
        </div>
```

- Inside the first column of this row, insert the table as follows:

```
                <div class="table-responsive">
```

```
                    <table class="table table-striped">
                        <thead class="thead-dark">
                            <tr>
                                <th> </th>
                                <th>2013</th>
                                <th>2014</th>
                                <th>2015</th>
                            </tr>
                        </thead>
                        <tbody>
                            <tr>
                                <th>Employees</th>
                                <td>15</td>
                                <td>30</td>
                                <td>40</td>
                            </tr>
                            <tr>
                                <th>Guests Served</th>
                                <td>15000</td>
                                <td>45000</td>
                                <td>100,000</td>
                            </tr>
                            <tr>
                                <th>Special Events</th>
                                <td>3</td>
                                <td>20</td>
                                <td>45</td>
                            </tr>
                            <tr>
                                <th>Annual Turnover</th>
                                <td>$251,325</td>
                                <td>$1,250,375</td>
                                <td>~$3,000,000</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
```

Note the use of *table-responsive* class to create a responsive table, and the *table-striped* and *thead-inverse* classes for styling the table.

Bootstrap Cards

- Next we add a card to the second div in the first content row as follows, updating the div first by adding the classes col-12 col-sm-6 to it and then adding the card:

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
            <div class="col-12 col-sm-6">
                <div class="card">
                    <h3 class="card-header bg-primary text-white">
Facts At a Glance</h3>
                    <div class="card-body">
                        <dl class="row">
                            <dt class="col-6">Started</dt>
                            <dd class="col-6">3 Feb. 2013</dd>
                            <dt class="col-6">Major Stake Holder</dt>
                            <dd class="col-6">HK Fine Foods Inc.</dd>
                            <dt class="col-6">Last Year's Turnover</dt>
                            <dd class="col-6">$1,250,375</dd>
                            <dt class="col-6">Employees</dt>
                            <dd class="col-6">40</dd>
                        </dl>
                    </div>
                </div>
            </div>
```

- Next, we add a Bootstrap card and include a quotation in the card using the blockquote typography style:

```
            <div class="col-12">
                <div class="card card-body bg-light">
                    <blockquote class="blockquote">
                        <p class="mb-0">
You better cut the pizza in four pieces because
                            I'm not hungry enough to eat six.</p>
                        <footer class="blockquote-footer">Yogi Berra,
                            <cite title="Source Title">
The Wit and Wisdom of Yogi Berra,
                            P. Pepe, Diversion Books, 2014</cite>
                        </footer>
                    </blockquote>
                </div>
            </div>
```

Note the use of the *<blockquote>* tag to create a block quote in the card. We can use a *<footer>* inside the block quote to specify the attribution of the quote to its origin.

- Remember to commit the changes to your Git repository with the message "Tables and Cards"

Conclusions

In this exercise, we constructed a table and styled it with the Bootstrap table classes. Thereafter, we added two cards to the web page. We also saw the use of the description list and the block quote in the content.

From <https://www.coursera.org/learn/bootstrap-4/supplement/5dEHJ/exercise-instructions-displaying-content-tables-and-cards>

Images and Media: Objectives and Outcomes

In this lesson we will look at the use of images and media on websites. In particular we will review the Bootstrap classes to support the inclusion of images and media, supporting responsiveness of images and media, and the use of these as thumbnails and part of other components, in particular the media component. At the end of this lesson you will be able to:

- Use images and media and include them in your website
- Support responsive images and media using responsive Bootstrap classes for images and media
- Use thumbnails and media components using Bootstrap classes

From <https://www.coursera.org/learn/bootstrap-4/supplement/daVyr/images-and-media-objectives-and-outcomes>

No self-respect webpage designers,
would today design webpages without including
some form of images or media on their websites.
Computer science professors are an exception.
At this stage, you are probably expecting me to say this,
but that's not what I'm going to say.
I'm going to say that,
"A picture is worth $12 on iStockPhoto.com.
Bootstrap provides extensive support for including
images and various kinds of media in your websites and your web pages.
The image HTML tag is obviously going to be used for including images in your webpages.
Bootstrap also provides a bunch of classes that you can apply to
the image tag in order to further enhance the way images are included in your webpage.
So you could use a image class called img-fluid which will make your images responsive,
meaning that the size of the image will automatically adjust to different screen sizes.
You can also use an img-thumbnail which will put a white border around your image.
You can also deal with the shapes and sizes of images like you
see in this picture here so you can have images for which you create rounded corners,
either all the four corners,
or any chosen pair of corners.
So you can apply the class around it to
create a image with rounded corners or you can see
a rounded hyphen top bottom left or right to put
the rounded corners in any one of these four edges of your image.
You can also apply rounded circle to create circular images.
So, when you visit websites,
you would see images being used in many of these ways already.
You'll remember we saw the Card class in the previous lesson and the exercise.
Here, I illustrate the use of the Card class with an image included.

So in this case, we include an image with the class card image top and if you include large enough size the image that image will be included at the top and then enclosed inside a card like this.
If the image is too small, then it may be positioned to one corner in your card.
So use a large enough image to be able to define a clean card like this.
So here you'll see the image being described with, the classes applied to that including card image top and img-fluid to make the image responsive automatically.
So, this is how you can use the images with the Card class to define cards that you use on your webpages.
Another way of including images and related content in your webpages is a media object.
A media object allows you to specify an image that you can position either to the left or to the right of the description.
Also, you can include a media body which contains the description that goes together with the image.
So as you see in this example here, the media body itself encloses any HTML formatted content here.
So here I am using an h2 and h4 together with a p class inside the media body to define the actual content.
We will use the media object in the exercise that follows.
You can also do a responsive embed of content.
So for example, if you have a video that you want to enclose in your webpage, then you can use an iframe class to frame the video, and then enclose it inside the div with the classes embedded responsive giving sizes so you can 16by9, 4by3 or video, content to be displayed.
So responsive embed, you can use the embed or iframe or video or object HTML tags and then enclose it inside a div.
Applying the embed-responsive and also together with the embed-responsive, specifying the dimensions which you want to use for 4by3 or 16by9 dimensions.
So this is how you can also enclose video content in your webpages.
Now, that we have seen the support for images and media, it is time to go onto an exercise where we will look at how we can include images in the webpage or the website that we have been working on.
We'll make some changes to the index.html page by including some images and content using the media object.

From <https://www.coursera.org/learn/bootstrap-4/lecture/banH5/images-and-media>

Exercise (Instructions): Images and Media

Objectives and Outcomes

In this exercise, we will explore the Bootstrap classes to support image and media on a website. In particular, we will look at how to include images on a website, how to make use of images within a media objects. At the end of this exercise you will be able to:

- Use Bootstrap classes to include a responsive images in a website
- Use a media object to include images and description on a website

Exercise Resources

img
ZIP File

Set up for the Exercise

- Download the img.zip file that we provide above and unzip it in the *conFusion* folder. This should create a folder named *img* there.
- We will now update the *index.html* file to include images and media objects on the web page.

Adding the Restaurant Logo

- We will now add the restaurant logo to the Jumbotron. In index.html go to the header row inside the jumbotron and replace the second column with the following code:

```
                <div class="col-12 col-sm align-self-center">
                    <img src="img/logo.png" class="img-fluid">
                </div>
```

You will immediately notice the restaurant logo being displayed in the jumbotron.

- Next, we will add the logo to the navbar where we display the restaurant brand. Go to the navbar and replace the code there for the <a> tag with the "navbar-brand" class with the following code:

```
            <a class="navbar-brand mr-auto" href="#">
<img src="img/logo.png" height="30" width="41"></a>
```

Note the inclusion of the logo in the navbar.

- Repeat the above two steps for the *aboutus.html* and the *contactus.html* page also to update their navbars and jumbotrons.

Adding Media Objects

- Next we will work with the content on the web page and use the media object classes to style the content in the content rows.
- Go to the first content row, and replace the content in the second column containing the description of Uthappizza with the following code:

```
                <div class="media">
                    <img class="d-flex mr-3 img-thumbnail align-self-center"
                            src="img/uthappizza.png" alt="Uthappizza">
                    <div class="media-body">
                        <h2 class="mt-0">Uthappizza</h2>
                        <p class="d-none d-sm-block">
A unique combination of Indian Uthappam (pancake) and
                            Italian pizza, topped with Cerignola olives, ripe vin
e
                            cherry tomatoes, Vidalia onion, Guntur chillies and
                            Buffalo Paneer.</p>
                    </div>
                </div>
```

Note the use of the *media* class and the related Bootstrap classes to style the content.

- Next, we will go to the third row and replace the contents of the second column containing the description about Alberto with the following content:

```
                <div class="media">
                    <img class="d-flex mr-3 img-thumbnail align-self-center"
                            src="img/alberto.png" alt="Alberto Somayya">
                    <div class="media-body">
                        <h2 class="mt-0">Alberto Somayya</h2>
                        <h4>Executive Chef</h4>
                        <p class="d-none d-sm-block">Award winning three-
star Michelin chef with wide
                            International experience having worked closely with
                            whos-who in the culinary world, he specializes in
                            creating mouthwatering Indo-
Italian fusion experiences.
                            </p>
                    </div>
                </div>
```

- Finally, do a Git commit with a message "Images and Media".

Conclusions

In this exercise, we learnt about the Bootstrap classes to support images and media in a web page. We saw how we can include responsive images on a web page. In addition, we saw the use of images within a media object to style and display content.

From <https://www.coursera.org/learn/bootstrap-4/supplement/7UjCF/exercise-instructions-images-and-media>

Alerting Users: Objectives and Outcomes

In this lesson we examine various ways of delivering alert information to users. We examine labels, badges, alerts and progress bars. At the end of this lesson, you will be able to:

- Include labels and badges in your web page
- Create, style and include alerts in your web page
- Appreciate the use of progress bars and controlling the state of the progress bars.

From <https://www.coursera.org/learn/bootstrap-4/supplement/Hze74/alerting-users-objectives-and-outcomes>

You may have several reasons for drawing the attention of visitors
to certain parts of your website or maybe you want to alert
the users about some information or you may want to keep
the users informed about the progress of an operation that they initiate on your website.
Let's look at some different mechanisms that
bootstrap provides for these kind of operations.
Bootstrap has many different components that enable us to keep users informed.
We have badges which allow us to draw attention of
users to recent updates to our web page.
We have alert, error,
and warning messages that can be delivered to users should
they perform operations that are not allowed.
And also, we may use progress bars to keep users informed
about the progress of whatever operations that they initiate on our website.
Let's look at this in a bit more detail next.
Badges are an easy way of adding small amount of
information to your website to attract the attention of visitors.
As an example, you can see that we have added a couple of badges to
this description of the dish here on our main page so you
can see that we have the red colored badge drawing attention of the users to
the fact that this is a hot item on the menu
and perhaps even informing them about the price of the menu.
These kind of small pieces of information can be added to your website using a badge.
A badge for example,
the red one here is created by using a span with the classes badge and badge-danger.
You can use the various other colors that are built into
the bootstrap framework for creating badges of several different colors.
Similarly, you can use a badge-pill which creates a rounded rectangle as you see here,
and that can be done by applying the badge,
badge-pill class to it along with the badge class.
Alerts are another way of informing users about information.
So for example, an alert can be created using the alert component
which simply can be created by applying the alert with alert-warning for example,
in this example that you see warning creating a yellow colored alert.
You can apply again danger primary and other bootstrap colors there.
And then alert-dismissible class,
allows you to include a class there which the user can click to dismiss an alert.
To support that operation,

you need to also include a button into your alert as you can see inside this div,
we included this button here.
And the contents of the alert can be formatted using standard HTML as you can see here.
So, this enables us to create an alert on our web page.
So, to summarize, the Alert can be created by a appling the alert class
together with additional qualifying alert classes,
which enable you to create alerts in different colors.
We can also include links in our alerts.
We can also make the alert dismissible by using
the alert dismissible class and including a cross button inside our alert.
Similarly, progress bars are included in websites to keep
users informed about the progress of whatever we have initiated on the website.
So for example, if you're uploading a file to a website,
then you may want to keep the user informed about how much of the file has
completed upload by showing a progress bar on the website.
So, creating a progress bar in bootstrap is
done by applying the progress class to a div and inside that we can apply
another inner div with the class progress bar and
also allow that to be further styled by using things like
progress bar striped and then you can apply the color for
the progress bar by specifying the color as BG hyphen and the color.
So, in this case the bg-danger being applied to this progress bar.
We can also make the progress part striped,
we can also animate the progress bar,
and also specify how much of that progress bar needs to be in form.
So, by changing this width value
dynamically you can create a progress bar that will keep updating itself on the web page.
So, to summarize, a progress bar can be created by
applying a progress class to the div and inside
there an inner div to which you apply
the progress bar class will enable you to create progress bar.
You can even stack multiple progress bars together to create
a stacked progress bar and also apply different colors to the progress bar and also
can have different appearance for the progress bar using
the progress bar striped class and also progress bar
animated class for animating the stripes on our progress bar.
Now that we have looked at various ways of alerting users,
let's do a simple exercise next and use badges
in our web page to draw the attention of users.

From <https://www.coursera.org/learn/bootstrap-4/lecture/BHN4S/alerting-users>

Exercise (Instructions): Alerting Users

Objectives and Outcomes

In this short exercise we will examine the use of badges as a way of alerting users. At the end of this exercise, you will be able to:

- Add a badge to your web page using the Bootstrap badge class

Adding Badges

- We will continue to edit the *index.html* file. In this file, we will add a badge *HOT* next to the name of the dish Uthappizza in the first content row. To do this, add the following code inside the <*h2*> containing the name of the dish:

```
<span class="badge badge-danger">HOT</span>
```

- Next we will add a badge as a badge-pill right next to the earlier tag in the web page. Add the following code to the <*h2*> tag:

```
<span class="badge badge-pill badge-secondary">$4.99</span>
```

- Remember to commit the changes to the Git repository with message "Alerting Users".

Conclusions

In this short exercise, we learnt how to add badges to our web page.

From <https://www.coursera.org/learn/bootstrap-4/supplement/DP5FX/exercise-instructions-alerting-users>

Bootstrap JavaScript Components

Monday, September 5, 2022 2:08 AM

This module concentrates on Bootstrap's JavaScript based components. You will learn about tabs, pills and tabbed navigation, collapse, accordion, scrollspy, affix, tooltips, popovers, modals and the carousel. At the end of this module you need to complete the third assignment.

Learning Objectives

- Recognize how the JavaScript components provide dynamic behavior to your website
- Create various navigational elements in your website using Tabs and Accordion
- Create modals, tooltips and popovers to reveal content in your website
- Build a carousel to show dynamic content on your website

From <https://www.coursera.org/learn/bootstrap-4/home/week/3>

In the previous module,
we examined the number of Bootstrap components that are purely based around CSS.
So, by applying the CSS classes for these components,
we were able to create them on our webpage.
In this module, we are going to examine more of Bootstrap's JavaScript-based components.
So summarizing, as we saw in the previous module,
Bootstrap supports a number of CSS classes that can be applied to
your HTML markup in your webpages to activate and make use of these CSS classes.
So, you could see how the CSS classes modified the generic HTML tags and
provided Bootstrap's own way of presenting the same information.
So, the components that are purely based upon CSS can easily be activated by
applying the bootstrap CSS classes to
the various HTML tags that are there in your webpages.
Bootstrap by itself can be used purely for its CSS classes.
One other aspect of Bootstraps CSS classes that we saw was
that most of these components started with a base class that you applied.
Like for example, when you look at button,
you saw that the base class was the BTN class.
Then you would apply something like
a BTN primary which would add their colors to the BTN,
then you could define the size of the button by saying BTN SM.
Then you could also define the button to be an entire block by using the BTN block class.
So, all these subsequent classes and modifiers to the base BTN class.
So this approach, you will see it
over and over again in many of the Bootstraps components.
So you will have a base class then set of modifier classes that can be
applied to modify the behavior of the base component.
This facilitates easy design of your webpage.
Taking another example, we can look at the Nav class.
We use the Nav class in the Nav-bar that
we included in our webpage in the previous module.
The Nav class, we modify that by applying the Nav-bar Nav class to it.
So that provided us with a certain way of presenting the navigation information.
In this module, we will examine Nav-tabs and
Nav-pills which are yet another way of presenting navigation information for the webpage.
The tabs and the pills we are presenting navigation again,
supported with a JavaScript plugin,
provides an enhanced way of providing navigation within your content.
So, we'll examine that in this particular module.
I use this as an example to illustrate to you how started from a base class,

you can modify by applying the modifier classes to the base class.
Of course, if you want to bring in
the JavaScript part of your Bootstrap components into action,
then you should start applying the JavaScript classes.
So, Bootstrap supports a number of components that
are supported through JavaScript plugins.
Now, when you use these plugins in your webpage,
you can include individual plugins that you are actually going to use in your website,
or the entire set of plugins.
Now, what we're going to do in the exercises is,
include the entire set of plugins because we just want explore all of them.
But if you are designing a website,
you may include only a subset of these plugins
that you're actually employing in your website,
thereby reducing the amount of JavaScript code that needs
be sent over to a user that is viewing your webpage.
This is the way I look at Bootstraps approach to using JavaScript in
its own support for Bootstrap components that are enabled with JavaScript.
If you look at JavaScript on its own,
it provides a lot of flexibility and enables you to do lot of interesting things.
But that also entails
writing JavaScript code to be used when you are designing your website.
Now, I look at JavaScript as a big sumo wrestler with a lot of capability,
a lot of power, but untamed.
Now, JQuery as a library that is built on top of JavaScript,
essentially takes the sumo wrestler and then packages it in a better manner to make
this hundred lbs gorilla into a more manageable size,
more easier to approach and more easier to
include and make use off in your website design.
So, if you are using JQuery library directly,
you can definitely leverage many of
the JQuery components and make use of them in your webpage.
Now, that is an approach that you can always take because JQuery
is already going to be included in your Bootstrap webpage,
if you are going to leverage the JavaScript-based components.
Bootstrap takes this one step further,
and then looking at the picture
Bootstrap's approach is to take the sumo wrestler and then put him into a box.
Essentially standing for packaging
your JavaScript plugins into a component that you can more easily use in your webpages.
So, the Bootstrap JavaScript-based components essentially take JQuery-based support,
but then package them in a way that you can employ them in
your webpage even without writing a single line of JavaScript code.
This is where the Bootstrap JavaScript component
can be used without writing a single line of code.
The way it is supported in Bootstrap is that,
the JavaScript components support a number of data hyphen
star attributes that you can apply to your HTML tags.
So, you will see me using things like data-toggle,
data-spy, data-target, and so on.
If you recall, when we designed the Nav-bar,
you had actually seen a couple of these attributes.
There we designed the button for the extra small screens.
Now, it is time for us to [inaudible] why we did it in the previous module.
So, we will explore this approach in great detail in this module,
looking at several Bootstrap components that leverage JavaScript support.
Of course, you can leverage
the full JavaScript API that is
available for all these components by actually writing JavaScript code.

We will defer this to the next module,
where we will examine how you can write simple JavaScript code based upon
the JQuery syntax to add
more functionality to your JavaScript-based Bootstrap components.

From <https://www.coursera.org/learn/bootstrap-4/lecture/rKmok/bootstrap-javascript-components>

Tabs and Tabbed Navigation: Objectives and Outcomes

In this lesson, we examine tabs and tabbed navigation. Tabs require Javascript support to be enabled for navigating the content.
At the end of this lesson you will be able to:

- Design a tabbed navigation for your content
- Use tab panes to organize the content and navigate the content using tabbed navigation

From <https://www.coursera.org/learn/bootstrap-4/supplement/HB1V9/tabs-and-tabbed-navigation-objectives-and-outcomes>

In the previous lesson I referred to the nav class that is used in Bootstrap for
creating navigation structures.
We explored the navbar nav already in the previous module.
In this lecture,
we're going to explore nav tabs and nav pills in a bit more detail.
And the exercise that follows this, we'll
look at how we make use of this to create a navigation structure for our web page.
Play video starting at ::36 and follow transcript0:36
Tabs and pills provide a interesting navigation structure for our web page.
Play video starting at ::42 and follow transcript0:42
They will allow you to enclose content inside your web page that
the user can navigate among the content without reloading
the web page to a different web page.
So if you have a certain amount of content to be presented in your web page,
but want to show only part of the content on your web page, but
still provide user with an ability to navigate to other parts of the content,
then tabs and pills can be very useful for this purpose.
Play video starting at :1:14 and follow transcript1:14
What's better than to just go and take a look at the use of tabs and
pills in navigation?
Play video starting at :1:20 and follow transcript1:20
Taking a look at an example of the use of tabs in navigation,
you'll see that in this particular case, you have
Play video starting at :1:29 and follow transcript1:29
the Corporate Leadership information in our aboutus.html page,
enclosed inside tabs with the tab navigation built on top.
So you can see that you have the four different corporate leaders
represented in their own tab here.
And the content currently revealed on the web page is only
the details about the first corporate leader.
So if you want to switch to others, then you simply click on
the corresponding tab, and then reveal that content in your web page.
So notice that we are not navigating to a different web page, but
we are simply revealing the content using this tab navigation here.
Play video starting at :2:15 and follow transcript2:15
Instead, in some cases you may find
a pill-based navigation more useful for your purpose.
So a pill-based navigation works like this.
So you still have the same navigation structure on top here, but
the currently displayed information will be highlighted as a pill here.
So if you switch to another part of your navigation, notice how

that particular item is being highlighted in the web page,
and the corresponding information being displayed in the tab content down below.
So this is a pill-based navigation.
So this uses nav pills instead of nav tabs for
the class being applied to the nav class there.
Now that you have seen an example of the use of tabbed and pull navigation
in a web page, I'm sure you're curious to know how it is actually constructed.
That is what we're going to explore in the exercise next.
So we will use tabbed navigation and then use tab content and
tab panes to organize the actual content that will be displayed in our web page.

From <https://www.coursera.org/learn/bootstrap-4/lecture/UIzBq/tabs-pills-and-tabbed-navigation>

Exercise (Instructions): Tabs

Objectives and Outcomes

In this exercise we will explore Bootstrap tabs and tabbed navigation. In particular we will learn about the use of tabs for organizing the content. At the end of this exercise you will be able to:

- Design a web page to use tabbed navigation for organizing the content
- Use tab panes and organize the content into the panes
- Facilitate navigation among the tab panes using the tabbed navigation elements

Adding Tab Navigation Elements

- Open the *aboutus.html* page and move to the second content row containing the details of the corporate leadership of the restaurant.
- Right after the Corporate Leadership heading, introduce the following code to set up the tabbed navigation:

```
                <ul class="nav nav-tabs">
                    <li class="nav-item">
                      <a class="nav-link active" href="#peter"
                        role="tab" data-toggle="tab">Peter Pan, CEO</a>
                    </li>
                    <li class="nav-item">
                      <a class="nav-link" href="#danny" role="tab"
                        data-toggle="tab">Danny Witherspoon, CFO</a>
                    </li>
                    <li class="nav-item">
```

```
                    <a class="nav-link" href="#agumbe"role="tab"
                      data-toggle="tab">Agumbe Tang, CTO</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="#alberto" role="tab"
                      data-toggle="tab">Alberto Somayya, Exec. Chef</a>
                  </li>
              </ul>
```

Note the use of the *<ul>* tag with the *nav* and *nav-tabs* classes to set up the tab navigation. Each list item within the list acts as the tab element. Within each list item, note that we set up the *<a>* tags with the *href* pointing to the *id* of the tab pane of content to be introduced later. Also note that the *<a>* tag contains the *data-toggle=tab* attribute. The first list element's *<a>* tag contains the class *active*. This tab will be the open tab when we view the web page. We can switch to the other tabs using the tabbed navigation that we just set up.

Adding Tab Content

- The details about the various corporate leaders should now be organized into various tab panes. To begin this, we will enclose the entire content into a div element with the class tab-content as specified below:

```
1
2
3
4
5
```

```
              <div class="tab-content">

                  ...

              </div>
```

- Then we take the name and description of the CEO of the company and enclose it within a tab-pane as follows

```
1
2
3
4
```

```
                  <div role="tabpanel" class="tab-pane fade show active" id="peter">
                      <h3>Peter Pan <small>Chief Epicurious Officer</small></h3>
                      <p> ... </p>
                  </div>
```

Note the use of the *tab-pane*, *fade*, *show*, and *active* classes and with *peter* as the id. This is the same id used as the *href* in the *<a>* link in the navigation.

- The remaining content is also similarly enclosed inside appropriate divs with the correct ids and the classes specified as above. Only the first tab pane will have the *show* and *active* classes specified to indicate that the content should be visible on the web page by default.

Modifying the tab-content CSS

- We now modify the CSS styles for the tab-content class in the *mystyles.css* file as follows:

```
1
2
3
4
5
6
```

```
.tab-content {
    border-left: 1px solid #ddd;
    border-right: 1px solid #ddd;
    border-bottom: 1px solid #ddd;
    padding: 10px;
}
```

This modification adds a 1px border to the tab content which joins with the upper border introduced by the tab navigation element to give a clean tab like appearance.

- Finally do a Git commit with the message "Tabs".

Conclusions

In this exercise we learnt the use of tabbed navigation, tab content and tab panes and their use in organizing and navigating within the content in a page.

From <https://www.coursera.org/learn/bootstrap-4/supplement/Lt62U/exercise-instructions-tabs>

Hide and Seek: Objectives and Outcomes

In this lesson we learn about the collapse javascript plugin that allows us to hide and reveal content. In particular we explore its use in creating an accordion. At the end of this lesson, you will be able to:

- Use the collapse plugin to hide/reveal content
- Construct the accordion using cards

From <https://www.coursera.org/learn/bootstrap-4/supplement/RrvHW/hide-and-seek-objectives-and-outcomes>

Let's now talk about a very useful plugin in Bootstrap called as the Collapse.
We have seen its application in the navigation bar earlier.
We'll revisit that code and take a quick look at it in more detail.
A related component is called as the Accordion.
The Accordion makes use of the Collapse plugin for its functioning.
And we will see that in detail in the exercise that follows.
Play video starting at ::31 and follow transcript0:31
The Collapse plugin provides an easy way of revealing and
hiding content on your web page.
Play video starting at ::40 and follow transcript0:40
This revealing and hiding of the content is usually triggered
by the user clicking on a button or a link in your web page.
Now you will see many places where the Collapse kind of
behavior is very well leveraged to show content.
We'll revisit our navbar to look at the Collapse plugin in action, and
then we'll also see an Accordion example next.
Play video starting at :1:8 and follow transcript1:08
Going back to our web page, you'll recall that when we created the navbar
in our web page for extra-small screens, the navbar was collapsed and
then it was revealed when we clicked on the button here.
Now this is an example of the use of the Collapse plugin here.
When we look at the code, we will see that we actually apply the Collapse plugin to
the div that contains this navigation bar.
Play video starting at :1:39 and follow transcript1:39
Taking a look at the code for
our navigation bar that we designed earlier, you saw that in here,
inside the container, we created this button which is displayed for
the extra-small screen sizes by using the navbar-toggler class.
Look at the div that follows this button here.
For the div that follows this button, we used the collapse class.
So this is the use of the Collapse plugin in our navbar.
So what this creates is that this particular content,
our navbar that is enclosed inside this div, will be collapsed for
the extra-small screen sizes.
And then together with the Collapse plugin, this button that appears
there will trigger the showing and hiding of this navbar here.

That is why when you look at the button, you will see that right there,
we supply the data-toggle attribute with the collapse in here,
and then the data-target as #Navbar here.
And notice that the id for this div that we applied here is Navbar.
So we are specifying essentially that this button is going to act as the triggering
mechanism for this Collapse plugin to work on this div here.
So this is a use of the Collapse plugin in your web page.
Play video starting at :3:4 and follow transcript3:04
Also, we'll look at the other component, which is the Accordion in Bootstrap.
How does an Accordion behave?
So this is an example of an Accordion that we're going to construct in the exercise
that follows.
So here, you'll see that I have replaced the tabbed navigation that we
did in the previous exercise by using an Accordion here.
The way the Accordion works is that one piece of content is revealed and
the remaining three are hidden.
So this uses the Collapse plugin for this purpose.
So when you click on any one of these other corporate leader's name,
then the details of that corporate leader gets revealed and
the remaining three gets hidden.
So this is the Accordion behavior here.
If you are familiar with an accordion,
the musical instrument, you know how the bellows of the accordion work.
So this is, in some sense, behaving like the bellows of the accordion.
So that's the reason why this component is referred to as an Accordion in Bootstrap.
One more example of the Collapse plugin in use is on your own course website.
So if you go to your course website, and
then you go into the content of your course website and
start viewing the content here, you will notice that on the left side,
you have a navigation that comes into picture on the left side.
So this is the navigation there.
So when you click on each one of them, you notice how the content in here
is hidden and revealed by clicking on each of these links there.
So this is like the Collapse plugin that we saw in Bootstrap.
Although, of course, the Coursera page actually doesn't use Bootstrap.
We'll move to the next exercise where we will see how we construct
the Accordion for showing our corporate leadership information.

From <https://www.coursera.org/learn/bootstrap-4/lecture/PjhJs/collapse-and-accordion>

Exercise (Instructions): Accordion

Objectives and Outcomes

In this exercise we explore the use of the collapse Javascript plugin together with card component to create an accordion to show/hide content in a web page. At the end of this exercise, you will be able to:

- Design an accordion using the collapse plugin together with the card component.

Converting Tabs to Accordion

- First delete the <ul> class that was introduced for the tabbed navigation.
- Then the turn the *tab-content* div into a *accordion* div. Use the code structure as shown below:

```
1
2
3
```

```
                <div id="accordion">
```

```
                    . . .
                </div>
```

- Then, convert the first tab-pane into a card such that the name appears as a card heading, and the <p> will be in the card body. Use the structure of the code as shown below:

```
                <div class="card">
                    <div class="card-header" role="tab" id="peterhead">
                    <h3 class="mb-0">
                        <a data-toggle="collapse" data-target="#peter">
                        Peter Pan <small>Chief Epicurious Officer</small>
                        </a>
                    </h3>
                    </div>
                    <div class="collapse show" id="peter" data-parent="#accordion">
                        <div class="card-body">
                            <p class="d-none d-sm-block">. . .</p>
                        </div>
                    </div>
                </div>
```

- For the remaining three leaders, use the same structure as above, with the appropriate ids set up for the cards, as shown in the code structure below:

```
                    <div class="card">
                            <div class="card-header" role="tab" id="dannyhead">
                            <h3 class="mb-0">
                                <a class="collapsed" data-toggle="collapse" data-target="#danny">
                                Dhanasekaran Witherspoon <small>Chief Food Officer</small>
                                </a>
                            </h3>
                        </div>
                        <div class="collapse" id="danny" data-parent="#accordion">
                            <div class="card-body">
                                <p class="d-none d-sm-block">. . .</em></p>
                            </div>
                        </div>
                    </div>
                    <div class="card">
                            <div class="card-header" role="tab" id="agumbehead">
                            <h3 class="mb-0">
                                <a class="collapsed" data-toggle="collapse" data-target="#agumbe">
                                Agumbe Tang <small>Chief Taste Officer</small>
                                </a>
                            </h3>
                        </div>
                        <div class="collapse" id="agumbe" data-parent="#accordion">
                            <div class="card-body">
                                <p class="d-none d-sm-block">. . .</p>
                            </div>
                        </div>
                    </div>
                    <div class="card">
                            <div class="card-header" role="tab" id="albertohead">
                            <h3 class="mb-0">
                                <a class="collapsed" data-toggle="collapse" data-target="#alberto">
                                Alberto Somayya <small>Executive Chef</small>
                                </a>
                            </h3>
                        </div>
                        <div class="collapse" id="alberto" data-parent="#accordion">
                            <div class="card-body">
                                <p class="d-none d-sm-block">. . .</em></p>
                            </div>
```

- After completing the update, check the behavior of the accordion on the web page.
- Finally do a Git commit with the message "Accordion".

Conclusions

In this exercise we constructed the accordion using the collapse plugin together with the card component.

From <https://www.coursera.org/learn/bootstrap-4/supplement/WPs62/exercise-instructions-accordion>

Revealing Content: Objectives and Outcomes

In this lesson we look at several ways of presenting information to users by overlaying the information on top of the page. In particular, we look at tooltips, popovers and modals. At the end of this lesson, you will be able to:

- Set up a tooltip to be displayed when the user hovers over an area of the page
- Enable popovers when the user clicks on a link or button
- Reveal and hide modals when the user clicks on a link or button

From <https://www.coursera.org/learn/bootstrap-4/supplement/M44Lt/revealing-content-objectives-and-outcomes>

Let us now try to understand several mechanisms to enable us to
display information to the users overlaying the content of your web page.
So here we'll look at three different constructs that are available in Bootstrap
called tooltips, popovers, and modals.
Play video starting at ::25 and follow transcript0:25
So what are tooltips, popovers and modals and how are they useful?
We'll look at some basic ideas first ,and then we'll go on to look at some examples.
In the exercise that follows, we'll use tooltips and
modals in our web page, and we'll look at an example of popovers.
So, as I mentioned, tooltips, popovers, and modals are a way of revealing content
to the users, when the user interacts with certain elements on your web page.
Say for example when the user's mouse
clicks on a button, or hovers over a button, or clicks on a link,
or reaches a certain point on your web page.
So all these will trigger information to be displayed to the users.
So in this case,
the information is displayed as an overlay on top of your web page.
So the underlying content of the web page is still there, but
this is laid out on top of the underlying content.
So in terms of flexibility, tooltips are the simplest to implement, but
at the same time have limited flexibility in how they can display information.
Popovers are more flexible than tooltips, but they also have their own limitations.
Modals give you the most extensive support for
displaying content in a wide variety of ways.
Play video starting at :1:56 and follow transcript1:56
As an example, let's go to our webpage that we have been working on.
You see that when we hover our mouse pointer onto this button.
You see this message popping up on the screen here,
with some additional information.
This is an example of a tooltip.
Play video starting at :2:20 and follow transcript2:20
This allows you to display smaller amounts of information to the users.
So for example if you are trying to guide users through your website and
want them to know what happens when you click various locations any web page,
these maybe a good way of reminding them of what is expected.
So you could easily design for example,
walkthroughs of your website using these tooltips to indicate to users.
If you want a bit more detailed information,
then popovers would be more useful.
Play video starting at :2:52 and follow transcript2:52
The same example, implement and using a popover would look like this.
Now in this case,
you will have to explicitly click on the button to show the popover.
So in that case, the popover is shown with some title information, and
then the actual content at the bottom in that popover.
Now, dismissing the popover will require you again to click on the bottom there.
So this is the behavior of a popover.
In some circumstances, popovers are more useful than tooltips.
Our third kind of data overlay is the modal.
A modal allows you to present more

detailed information to the users than a tooltip and popover.
The content of the modal is itself divided into a header, body, and the footer.
And the modal itself can contain a lot more detailed information.
And you can use the entire Bootstrap grid,
inside the modal body, to organize the actual content.
We look at a couple of examples of the use of modals next.
Going to our web page, you will see that on the right hand side,
here we have a link here called Login.
So when you click on that link, you will notice that this modal
with their login form is popped up on the screen.
So this is the typical behavior of a modal.
And so here you can type in the information, and
then click on the Sign In button to sign in to your website.
Play video starting at :4:29 and follow transcript4:29
Going to your Coursera page, here is a real life example of the use of a modal.
So for example, if you click on the Log In button here,
you can see that on Coursera, a form pops up on the screen.
So this is another use of a modal in your web page.
Play video starting at :4:49 and follow transcript4:49
Now that you have seen examples of tooltips, popovers, and modals,
let's go to the next exercise, where we'll create a tooltip on our index.html page.
We'll also create a modal that allows the user to type in information for
logging in into our web page.

From <https://www.coursera.org/learn/bootstrap-4/lecture/6TTM1/tooltips-popovers-and-modals>

Exercise (Instructions): Tooltips and Modals

Objectives and Outcomes

In this exercise we will examine how to add tooltips to a web page. In addition we look at adding modals to a web page. At the end of this exercise, you will be able to:

- Add tooltips to a web page
- Add modals that are revealed when the user clicks on a link or a button in the web page.

Adding a Tooltip

- Let us now switch to the *index.html* page. We will now add a tooltip to this page. The tooltip will be added to the "Reserve Table" button that is in the jumbotron. We will update the *<a>* tag for the button as follows:

```
                    <a role="button" class="btn btn-block nav-link btn-warning"
                        data-toggle="tooltip" data-html="true"  title="Or Call us at  <br><strong>+852 12345678</strong>"
                        data-placement="bottom" href="#reserveform">Reserve Table</a>
```

As you can see from the code, we add a *data-toggle*, *data-placement* and a *title* attribute to the <a> tag in order to introduce a tooltip.

- The tooltip needs to be activated by adding a small Javascript code to the bottom of the page as follows:

```
    <script>
        $(document).ready(function(){
```

```
            $('[data-toggle="tooltip"]').tooltip();
        });
    </script>
```

This script is added right after the line that imports the bootstrap.min.js file.

Adding a Modal

- In the next step we introduce the modal to the web page. To set up the modal, add the following code right after the navbar at the top of the page.

```
    <div id="loginModal" class="modal fade" role="dialog">
        <div class="modal-dialog modal-lg" role="content">
            <!-- Modal content-->
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Login </h4>
                    <button type="button" class="close" data-dismiss="modal">&times;</button>
                </div>
                <div class="modal-body">
                    <form>
                        <div class="form-row">
                            <div class="form-group col-sm-4">
                                    <label class="sr-only" for="exampleInputEmail3">
Email address</label>
                                    <input type="email" class="form-control form-control-
sm mr-1" id="exampleInputEmail3" placeholder="Enter email">
                                </div>
```

```
                            <div class="form-group col-sm-4">
                                <label class="sr-only" for="exampleInputPassword3">Password</label>
                                <input type="password" class="form-control form-control-
sm mr-1" id="exampleInputPassword3" placeholder="Password">
                            </div>
                            <div class="col-sm-auto">
                                <div class="form-check">
                                    <input class="form-check-input" type="checkbox">
                                    <label class="form-check-label"> Remember me
                                    </label>
                                </div>
                            </div>
                        </div>
                        <div class="form-row">
                            <button type="button" class="btn btn-secondary btn-sm ml-auto" data-
dismiss="modal">Cancel</button>
                            <button type="submit" class="btn btn-primary btn-sm ml-1">
Sign in</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

- Next we introduce another link on the right side of the navbar in order to trigger the display of the modal. To do this, add the following code in the navbar after the :

```
1
2
3
4
```

```
                <span class="navbar-text">
                    <a data-toggle="modal" data-target="#loginModal">
                    <span class="fa fa-sign-in"></span> Login</a>
                </span>
```

We are introducing another link to the right of the navbar using the *navbar-text*. This contains a link with an *<a>* tag with the *data-toggle="modal"* and *data-target="#loginModal"* attributes.

- Save all the changes and do a Git commit with the message "Tooltip and Modal".

Conclusions

In this exercise we explored tooltips and modals as two ways of revealing content for the user upon clicking on a button or a link.

From <https://www.coursera.org/learn/bootstrap-4/supplement/StwXQ/exercise-instructions-tooltips-and-modals>

Carousel: Objectives and Outcomes

In this lesson we will examine the use of the Carousel component in our web page. We will examine the configuration of the various aspects of the carousel. At the end of this lesson you will be able to:

- Use a carousel component in your web page
- Configure various aspects of the carousel
- Add controls to the carousel to manually control it

From <https://www.coursera.org/learn/bootstrap-4/supplement/LlqzF/carousel-objectives-and-outcomes>

Let's take a look at yet another interesting component in bootstrap,

called as the bootstrap carousel.
The carousel allows you to display multiple items on
your web page and provides an animated display of these items.
As I mentioned, a carousel is an interesting component
that allows you to include a slideshow on your web page,
and can also provide manual controls for the slides.
So, as the slides slide from left to right on your web page,
you can control and choose which particular item to view,
and also you can manually scroll the items left and right.
Carousel finds a lot of applications in many websites that you might have visited.
Let's take a look at an example of a carousel.
Go into your web page,
you see a carousel included on this web page.
So, you see that this carousel provides
interesting transitions from one slide to another,
and also provides manual controls that enable you to select which item to view.
So for example, you can manually scroll right and left in the carousel.
You can also select an item from the carousel
using these indicators that are provided at the bottom of the carousel.
I'm sure you have seen many websites using this kind of component.
Now that I have gotten you interested in the carousel,
I'm sure you're curious about how you create one for your web page.
That is what we're going to examine in this next exercise,
where we will add a carousel to our web page,
the index.html page and then create the manual controls for it.

From <https://www.coursera.org/learn/bootstrap-4/lecture/1YMc3/carousel>

Exercise (Instructions): Carousel

Objectives and Outcomes

In this exercise we will examine the carousel component and add it to the web page. We will examine the configuration of the carousel and adding controls to the carousel. At the end of this exercise you will be able to:

- Use a carousel component in your web page
- Configure various aspects of the carousel
- Add controls to the carousel to manually control it

Adding a row for the carousel

- The carousel will be added to the *index.html* page. In this page, go to the top of the container div that contains the content of the page and add a new content row and an inner div spanning all the 12 columns as follows:

```
1
2
3
4
5
6
7
```

```
<div class="row row-content">
    <div class="col">

    </div>
</div>
```

Adding a Carousel

- Next, add the basic carousel div inside the content row that you just added as follows:

```
            <div id="mycarousel" class="carousel slide" data-ride="carousel">

            </div>
```

Adding Carousel Content

- Next add the content inside the carousel as follows:

```
                    <div class="carousel-inner" role="listbox">
                        <div class="carousel-item active">
                            <img class="d-block img-fluid"
                                src="img/uthappizza.png" alt="Uthappizza">
                            <div class="carousel-caption d-none d-md-block">
                                <h2>Uthappizza <span class="badge badge-danger">
HOT</span> <span class="badge badge-pill badge-default">$4.99</span></h2>
                                . . .
                            </div>
                        </div>
                        <div class="carousel-item">
                                . . .
                        </div>
                        <div class="carousel-item">
                                . . .
                        </div>
                    </div>
```

Note that the first item has been set up partially. Fill in the remaining parts from the content rows below.

Adding CSS Classes

- Add the following CSS classes to the *styles.css* file:

```
.carousel {
    background:#512DA8;
}

.carousel-item {
  height: 300px;
}

.carousel-item img {
    position: absolute;
    top: 0;
    left: 0;
    min-height: 300px;
}
```

Adding Carousel Controls

- Next, we will add manual controls to the carousel so that we can manually move among the slides. Add the following code to the bottom after the carousel items in the div of the carousel to add slide indicators that enable us to select a specific slide:

```
                    <ol class="carousel-indicators">
                        <li data-target="#mycarousel" data-slide-to="0" class="active"></li>
                        <li data-target="#mycarousel" data-slide-to="1"></li>
                        <li data-target="#mycarousel" data-slide-to="2"></li>
                    </ol>
```

- Then, add the left and right controls to the carousel that enable us to move to the previous and next slide manually. Add this to the bottom of the carousel div:

```
                    <a class="carousel-control-prev" href="#mycarousel" role="button" data-
slide="prev">
                        <span class="carousel-control-prev-icon"></span>
                    </a>
                    <a class="carousel-control-next" href="#mycarousel" role="button" data-
slide="next">
                        <span class="carousel-control-next-icon"></span>
                    </a>
```

- Do a Git commit with the message "Carousel".

Conclusions

In this exercise we learnt about the carousel component and how to add it to a web page. We also learnt about introducing manual controls to the carousel.

From <https://www.coursera.org/learn/bootstrap-4/supplement/z46f4/exercise-instructions-carousel>

Bootstrap and JQuery: Objectives and Outcomes

In this lesson we examine the use of Bootstrap, JQuery and JavaScript together to control Bootstrap JS components. Many of the Bootstrap's JS components provide JavaScript methods and events to control the behavior. At the end of this lesson, you will be able to:

- Understand how to use JQuery and JavaScript and Bootstrap's JS component methods to control the behavior of the components.
- Write JavaScript code taking advantage of the Bootstrap's JS component methods and JQuery methods for controlling Bootstrap JS components

From <https://www.coursera.org/learn/bootstrap-4/supplement/sVpOL/bootstrap-and-jquery-objectives-and-outcomes>

Let us now spend some time trying to understand how Bootstrap and
jQuery work together to support the Bootstrap's JavaScript components.
Play video starting at ::16 and follow transcript0:16
As we have understood from the previous module, Bootstrap has
a number of interesting JavaScript-based components.
And we also learned that Bootstrap's JavaScript-based components
are all enabled using jQuery as the support.
So many of these plugins are written in jQuery.
Their plugins themselves can be individually included or
you can include all the plugins together as a single unit.
In the exercises, we have been including all the Bootstrap's JavaScript
plugins into our webpage.
Play video starting at ::55 and follow transcript0:55
We also saw this graphic in the previous module
to help us understand the relationship between JavaScript, jQuery, and
how the Bootstrap's JS-based plugins are implemented.
So we saw that Bootstrap's JS components ox in the jQuery-based components and
make it easy for us to make use of them in our webpage.
We also learned that the Bootstrap's JavaScript-based components can be used
in your webpage without writing a single line of JavaScript code.
So this is where the data-* attributes come to our rescue.
So you can use the various attributes that we saw in the previous
module to enable many of our Bootstrap's JavaScript-based components.
In case you want more flexibility with your JavaScript components,
then the full flexibility of that JavaScript-based API
is available for all the Bootstrap JavaScript components.
You can write code using the jQuery syntax and then use that
together to control your Bootstrap's JS components.
We're going to explore that in a little more detail in this module.
Before we proceed further, let me give you a quick tour of jQuery and its syntax so
that we understand some of the code that we'll write in this module.
jQuery is a very powerful,
lightweight JavaScript-based library that provides a number of different components.
It is a feature rich library that enables writing code for
doing HTML or DOM manipulation.
It allows you to do CSS manipulation.
So, for example,
you can apply CSS classes to various HTML elements through the jQuery code.
It allows you to handle HTML events,
and when those events occur you can implement methods that are being executed
in response to the occurrence of these events.
jQuery also supports various effects and
animations that can be applied to your HTML elements.
Also jQuery enables you to interact with a back end server using AJAX.

Although we won't to explore that in too much detail in this course.
We also learned that Bootstrap's JavaScript components
are built upon jQuery.
This components make use of many of the jQuery methods that are available to
implement the various features that this Bootstrap JavaScript components support.
Let's briefly understand jQuery Syntax.
Now if you have seen some of the code that we included
in the previous module, specifically for
the tool tip, you remember that there was something that started with a dollar sign.
So, the jQuery Syntax is all implemented using the dollar sign.
The dollar sign at the start of a sentence implies that this defines and
accesses jQuery's library plugins that are available.
Now whenever we use the dollar sign, you also supply a selector.
The selector is used to query and find those
HTML elements within your DOM to which you want to apply this manipulation.
There are various ways of doing selections.
We'll look at them in the subsequent slide.
Play video starting at :4:47 and follow transcript4:47
Then the third aspect of a jQuery statement is the action that you specify.
Now that you select an HTML element, what kind of an action you want to be performed
on that element, so that is the third part that you will see.
So as an example you can specify a jQuery statement
like $("p") within codes, and then .hide.
So in this case what it implies is that, select all those HTML elements
which are the paragraph elements,
starting with the p tag, and then those elements hide those elements.
So the action to be performed is the height,
so this will result in all the p elements being hidden from your
Play video starting at :5:39 and follow transcript5:39
DOM, and consequently from the webpage that is rendered.
Play video starting at :5:44 and follow transcript5:44
Similarly, you will see later on us using
a statement like saying dollar, and
then within quotation marks, #mycarousel,
implying that you are specifying the ID of a specific HTML element,
and then specifying carousel and pause.
We'll look at more details of what this actually does
in one of the subsequent slides there.
But note the structure of the jQuery statement syntax there.
You specify the selector, and then you specify the corresponding action to be
performed on the element that is selected by these selectors.
Let's take a look at Bootstrap jQuery example.
We'll go back to the previous example that we have seen earlier,
where we introduce the tooltip.
So, when we introduce the tooltip, we specified script and
then inside the script, we said $(document).
In this case the document means the entire document.
So the selected here is for the entire document.
So for the entire document, and then we'll specify ready.
Ready is the action to be taken.
So when the document is ready, then execute this function that is
specified as a parameter for this ready action there.
So the function that we saw being used for the tooltip specified it as $, and
then within brackets, it said, data-toggle=" tooltip".
Now here we are specifying that select those elements for
which there is an attribute with data-toggle="tooltip".
And then for those elements that match this criteria,perform this action
called enable tooltips for those elements.
And we close the function there.
So this specifies that this particular script will

be activated for those elements for which you have applied the data-toggle tooltip.
So basically for those HTML elements on which you have defined tooltips,
you want the tooltip to be enabled there.
So this is how we interpret this syntax of this jQuery example here.
Let's look at the various ways of specifying selectors.
So as we realize from the jQuery statement syntax,
we always follow dollar and then, within brackets, we specify a selector.
How do we specify selectors, what are the various ways you can specify selectors?
Here are some possibilities.
You can specify a selector by specifying any specific
HTML element by specifying the tag.
So you can say p, button, h4,
h3, or any of the HTML tags directly.
And so in that case the HTML tag name is specified within quotes and
that'll form your selector.
When you apply a selector like this, you're saying all the elements that match
this criteria will be selected.
Play video starting at :9:6 and follow transcript9:06
Then you can also specify a specific HTML DOM
element by specifying the ID of that element by using the #id.
So, for example, we use #myCarousel.
So in this case you're saying select that particular
HTML element for which the ID is myCarousel.
So that's the other way of selecting, by specifying an ID for an element.
Play video starting at :9:37 and follow transcript9:37
The third possibility is to select
elements by the classes which have applied to that.
So, for example, you can see within brackets if you say .btn,
meaning all the HTML DOM elements for
which you have applied the button class will be selected.
Or you can even further qualify by specifying a group of classes by
saying .btn.btn-default, meaning that those elements for
which button class and the button default class have both been applied.
Play video starting at :10:15 and follow transcript10:15
So that's another way,
by using the classes that are applied to the HTML element.
Play video starting at :10:20 and follow transcript10:20
The other possibility is to specify an attribute that has been applied
to HTML element.
So for example, you can say attributes are specified within square brackets here.
So you can see within square brackets href, and include it in the quotation so
which means that all those HTML elements for which href attribute has been applied.
Similarly, you can say data-toggle="tooltip",
which essentially saves all those elements for
which the data-toggle tooltip attribute has been applied.
Also, selecting the current element for
which you want to do something by saying $(this).
So meaning, for the current element that has already been selected, for
this, do something.
Play video starting at :11:11 and follow transcript11:11
And a lot more other possibilities.
So these are some examples, so you will see me using some of these
in the examples and the exercise that follows.
Play video starting at :11:24 and follow transcript11:24
Not only that, you can also specify jQuery events,
events based on which you respond.
So, for example, the user interactions
with various elements on your webpage will cause DOM events.
So, for example, with a mouse, the user may click at a particular location.
Or double-click, or when the mouse enters and leaves a specific location.
For keyboard you can respond to key presses, keydown and keyup.

Play video starting at :11:57 and follow transcript11:57
Events for forms, when the form is submitted,
when there is a change in a particular input element
Play video starting at :12:7 and follow transcript12:07
value, and when a particular element is focused upon, and so on.
We can even talk about the entire document.
So, upon the loading of the document, upon resizing the document,
upon scrolling or unloading the document, you can respond to those events.
So, in that case, the jQuery Event Methods that are supported include ready,
click, dblclick, mousedown, on, and so on.
So these are all the event methods that
will be executed upon occurrence of any of these events there.
Play video starting at :12:43 and follow transcript12:43
Let's take an example of the Bootstrap's Carousel to see how
JavaScript code can be written to control the carousel actions.
Play video starting at :12:54 and follow transcript12:54
So, for a carousel, you saw already from the previous module, index size.
We used all these attributes.
So we used the attribute data-slide="prev|next", or
we said data-slide-to and then specified the specific slide number,
where it says data-ride+"carousel".
And then we've specified data interval, the interval for
the sliding action to take place.
So for the carousel you can do things like, for
example, you can specify JavaScript based controls.
You can see $, and within brackets specify to select those elements,
the carousels, that are included in your page by specifying .carousel.
Meaning all those elements for which the carousel class has been applied,
and then following that you use the .carousel to specify something.
So as an example you will see me using something like this
in the exercise that follows.
We'll say .carousel and carousel, and and
inside there you would specify interval:2000,
meaning set the interval for the sliding to be 2000 milliseconds.
Or two seconds, in this case.
So that way you control or modify a particular property
of the carousel JavaScript element there.
The carousel also supports many other controls.
So, for example, you can say carousel('cycle'),
meaning start cycling the items from left to right.
You can say carousel('pause'), to pause the sliding action of the carousel.
Then you can say carousel(number), so
it cycles the carousel to that particular carousel item.
And then you can say carousel('prev'),
carousel('next') to go to the previous item and the next item in my carousel.
So, these can be invoked directly from our JavaScript code.
Similarly, when the JavaScript carousel
Play video starting at :15:12 and follow transcript15:12
item is in your webpage, it causes various events.
And based on the occurrence of these events, you can respond.
So, for example, you can say slide.bs.carousel.
This particular event will be fired when the slide instance method is invoked.
Play video starting at :15:29 and follow transcript15:29
Similarly, slid.bs.carousel means
this event is fired when it has completed the slide transition to the next item.
So within your code you can specify something
Play video starting at :15:43 and follow transcript15:43
like $("#myCarousel").on('slide.bs.carousel'.
So meaning that when that slide action starts, then invoke this function,
and then do something inside that function there.
So this kind of code can be returned also for

responding to the events that are caused by your carousel's behavior.
Play video starting at :16:7 and follow transcript16:07
Having considered some of these examples, we'll go on to an exercise where we
will actually write some JavaScript based code to control our carousel.
We will write code to include a couple of buttons within our carousel,
which will be used to control the sliding action of the carousel.
So, which means that we can pause and resume the sliding action of our carousel.
And we'll activate these buttons from JavaScript.

From <https://www.coursera.org/learn/bootstrap-4/lecture/OaEAe/bootstrap-and-jquery>

Exercise (Instructions): Bootstrap and JQuery

Objectives and Outcomes

In this exercise we learn about using Bootstrap's JS component methods together with JQuery and JavaScript to write JavaScript code to control the JS component. We will use the Carousel as an example of a component that can be controlled. At the end of this exercise you will be able to:

- Use Bootstrap's JS component methods together with JQuery and Javascript
- Use JS code to control the Bootstrap JS component

Adding the Carousel Control Buttons

- We will introduce two new buttons into the carousel component that we already included in the index.html page. To add the two buttons to the carousel, add the following code to the end of the carousel:

```
<div class="btn-group" id="carouselButton">
    <button class="btn btn-danger btn-sm" id="carousel-pause">
        <span class="fa fa-pause"></span>
    </button>
    <button class="btn btn-danger btn-sm" id="carousel-play">
        <span class="fa fa-play"></span>
    </button>
</div>
```

We are adding the two buttons inside a button group with the ID carouselButtons. The two buttons contain the pause and play glyphicons to indicate their corresponding actions.

Adding CSS Class for the Buttons

- Next, we add the following CSS class to styles.css file to position the buttons at the bottom-right corner of the carousel:

```
#carouselButton {
```

```
    right:0px;
    position: absolute;
    bottom: 0px;
    z-index: 1;
}
```

Adding JavaScript Code

- Finally we add the following JavaScript code to activate the buttons:

```
    <script>
        $(document).ready(function(){
            $("#mycarousel").carousel( { interval: 2000 } );
            $("#carousel-pause").click(function(){
                $("#mycarousel").carousel('pause');
            });
            $("#carousel-play").click(function(){
                $("#mycarousel").carousel('cycle');
            });
        });
    </script>
```

- Do a Git commit with the message "Bootstrap JQuery"

Conclusions

In this exercise we learnt about Bootstrap's JS component methods and how they can be used together with JQuery and JavaScript to control the behavior of a Bootstrap JS component.

From <https://www.coursera.org/learn/bootstrap-4/supplement/ZjMEL/exercise-instructions-bootstrap-and-jquery>

Exercise (Instructions): More Bootstrap and JQuery

Objectives and Outcomes

In this exercise we extend the previous exercise of controlling the carousel by using more JQuery and JavaScript to write JavaScript code to control the JS component. At the end of this exercise you will be able to:

- Use Bootstrap's JS component methods together with JQuery and Javascript
- Use JS code to control the Bootstrap JS component

Modifying the Carousel Control Buttons

- We will modify the carousel control buttons in the carousel component that we already included in the index.html page. Instead of two buttons, we will use a single button that will indicate if the carousel is currently cycling or paused. Furthermore we can use the button to toggle the carousel cycling behavior:

```
<button class="btn btn-danger btn-sm" id="carouselButton">
    <span id="carousel-button-icon" class="fa fa-pause"></span>
</button>
```

We are adding a single button inside a button group with the ID carouselButton. The buttons will show either as a pause or play button based on the current behavior of the carousel.

Modifying JavaScript Code

- Finally we modify the JavaScript code to control the behavior of the carousel and also show the appropriate button:

```
$("#carouselButton").click(function(){
    if ($("#carouselButton").children("span").hasClass('fa-pause')) {
        $("#mycarousel").carousel('pause');
        $("#carouselButton").children("span").removeClass('fa-pause');
        $("#carouselButton").children("span").addClass('fa-play');
    }
    else if ($("#carouselButton").children("span").hasClass('fa-play')){
        $("#mycarousel").carousel('cycle');
        $("#carouselButton").children("span").removeClass('fa-play');
        $("#carouselButton").children("span").addClass('fa-pause');
    }
});
```

- Do a Git commit with the message "More Bootstrap JQuery".

Conclusions

In this exercise we learnt more about Bootstrap's JS component methods and how they can be used together with JQuery and JavaScript to control the behavior of a Bootstrap JS component.

From <https://www.coursera.org/learn/bootstrap-4/supplement/7V2hr/exercise-instructions-more-bootstrap-and-jquery>

CSS Preprocessors: Objectives and Outcomes

In this lesson we explore CSS preprocessors, Less and Sass. We learn the salient features of both the processors and see how we can generate CSS from the Less and Sass code. At the end of this lesson, you will be able to:

- Write Less and Sass code to define your CSS classes
- Compile the Less and Sass code into the corresponding CSS classes

From <https://www.coursera.org/learn/bootstrap-4/supplement/kODXd/css-preprocessors-objectives-and-outcomes>

This particular lesson deals with CSS preprocessors,
in particular Less and Sass.
Now obviously, these topics have more to do with CSS and
defining CSS classes, but it would be inappropriate for
me to complete a Bootstrap course without
delving a little bit into CSS preprocessors.
Simply because Bootstrap is built using Sass for its source.
So, we'll briefly examine CSS preprocessors,
what they are, how they are useful in defining CSS classes,
and why Bootstrap makes use of them.
Play video starting at ::52 and follow transcript0:52
As you probably understand from your previous experience with CSS,
CSS is great for defining styles and
repeatedly applying these styles to various HTML elements.
And that's the way we use CSS when we define our web pages.
But when you look at the CSS code itself,
you begin to quickly realize the limitations of CSS.
Especially if you are coming from a programming background, you realize that
CSS doesn't have what you typically expect in a programming language, like variables,
nesting of selectors, variables, expressions, and functions.
Play video starting at :1:36 and follow transcript1:36
This means that writing CSS code becomes cumbersome, and
maintaining CSS code becomes cumbersome because of
lack of the traditional programming language-like syntax.
Now, this is where the CSS preprocessors come to our rescue.
Play video starting at :1:55 and follow transcript1:55
There are several popular CSS preprocessors that try to address some of
the shortcomings of CSS by supporting many of these features.
Two in particular that is of interest to us is Less and Sass.
Play video starting at :2:14 and follow transcript2:14
We will look at these two in more detail in this lesson.
Now, when you define your CSS classes using one
of these preprocessor languages like Less or
Sass, then they eventually have to be converted into CSS,
but this can be done automatically before the CSS gets used in your web page.
What these preprocessing languages bring to us is
more programming language-like syntax, as we will see in the next few slides.
We'll examine Less and Sass briefly.
Play video starting at :2:54 and follow transcript2:54
In particular, the reason for us to take a look at this is because Bootstrap itself
uses Sass for defining its source for its CSS classes.
And so if you go into customizing Bootstrap,
then you would have to work with Sass code.
Bootstrap 3, which was the preceding version of Bootstrap, used Less,
and hence, I thought it would be an appropriate thing to cover both Less and
Sass, because they are pretty much similar to each other
in terms of their capabilities and the way the syntax is defined.
The typical features that CSS preprocessors bring is the support for
variables, nesting selectors, expressions, functions and mixins.
So we will look at some of these with some examples in the next few slides.
The first thing that we'll look at is variables.
Now in many CSS classes that you define,
you might have some repeated use of certain quantities.
It might be more worthwhile if you define some variables that
hold these quantities and use these variables in defining your CSS classes.
So that's where variables come to your rescue.
Here, we'll look at both Less and Scss code.
Scss is a version of Sass,
more popular version of Sass syntax, so that's why I concentrate on Scss here.

So, if you define variables in Less,
you will proceed the name of the variable with an @ sign.
In case of Scss, you will use a dollar sign before the variable name.
And once you define those variables,
you can then use those variables when you define the classes.
As you see in the example, where we define the navbar-inverse as background,
or the carousel-item's height as variables.
Now, the advantage of defining variables at the start of your CSS class
is that there is a single point where you can update a value, and
it will automatically update all the CSS classes that use this variable.
Typically the way we use variables in programming languages.
Play video starting at :5:32 and follow transcript5:32
Variables in the CSS preprocessor languages can also have their own scope.
Play video starting at :5:39 and follow transcript5:39
Very often when you define CSS classes, especially when you
have to define classes that are nested inside other classes,
then, pretty soon your CSS code gets very cumbersome.
So this is where nesting is supported by your CSS preprocessors.
So as you can see, you can define for example, a carousel class, and
inside a carousel class, you can define a carousel item class.
And similarly, the image subclass inside the carousel item class.
So here you can see that each of these is nested inside a prior class.
Play video starting at :6:19 and follow transcript6:19
With variables, you can hold one value at a time in a variable.
Suppose you have a group of variables that summed up together
declare a set of CSS declarations, that is where we make use of mixins.
In Less you define a mixin by giving it a name,
and and in Sass you simply precede that with an @mixin in
front of the name of the mixin declaration.
And inside a mixin, you can define a bunch of CSS declarations
that can then be reused for various CSS classes.
As you see in this example here, we define the zero margin
as a mixin, both in Less and Sass.
Note the minor variation in the syntax in each of those cases.
And then, you can then use this mixin in defining additional CSS classes.
So here you can see that for the row header, we define the zero margin as
a mixin in the row header, which means that all
these properties from the zero margin will be inherited by that row header class.
Mixins themselves can take on additional parameters if you so wish to define them.
So in this case, I'm defining
a variation of the zero margin mixin that we saw in the previous slide.
Here, this zero margin mixin takes two parameters,
pad up down and pad left right.
And with Less, you can even specify the default value for it,
but with Scss you need to explicitly specify the values.
So here we are defining two different CSS properties, the margin and
padding, and then the padding itself uses the parameters for our mixin there.
So in that case, you can use these mixins as you see in
the two CSS classes defined below the row header and
the row content by specifying their parameter values
when you include that mixin into your CSS cluster.
Play video starting at :8:47 and follow transcript8:47
Not only that, you can even perform mathematical operations on predefined
variables when you use them inside your CSS classes.
So here you can see that I have defined two different carousel items
with two different sizes, and so
you can see that the heights of each one of them is defined differently.
In one case I'm using the predefined height, in another case I
am multiplying that height and then using it to define the height
property inside my carousel item for the item-large class.

So this kind of mathematical operations on
variables is also allowed in CSS preprocessor languages.
Play video starting at :9:35 and follow transcript9:35
Other features that the CSS preprocessors include
are functions that allow you to define mathematical functions, Less strings.
You can also do color operations and color blending operations using these functions
and make use of them when you define your CSS classes.
This is a bit advanced so I'm not going into too much of detail there.
Play video starting at :10:3 and follow transcript10:03
Also, you can input predefined CSS preprocessor classes into other classes.
So for example, if you've a Less class as defined, and
you can input that file into another Less file.
Similarly, if you have defined Scss files,
then you can import them into other Scss files using the import operation.
The syntax is the same for both Less and Sass.
Play video starting at :10:33 and follow transcript10:33
Now that we have seen some features of the CSS preprocessor languages, let's
now examine how we can actually make use of them by doing a couple of exercises.
First, we'll do a exercise based upon Less.
Then we'll follow that up with an exercis

From <https://www.coursera.org/learn/bootstrap-4/lecture/BFI1E/css-preprocessors-less-and-sass>

Exercise (Instructions): Less

Objectives and Outcomes

In this exercise you will learn to write Less code and then automatically transform it into the corresponding CSS code. At the end of this exercise you will be able to:

- Write Less code using many of the features of Less
- Automatically convert the Less code into CSS

Adding Less Variables

- Open the *conFusion* project in a text editor of your choice. In the css folder, create a file named *styles.less*. We will add the Less code into this file.
- Add the following Less variables into the file:

```
@lt-gray: #ddd;
@background-dark: #512DA8;
@background-light: #9575CD;
@background-pale: #D1C4E9;

// Height variables
@carousel-item-height: 300px;
```

We have just added a few color and a height variable. We will make use of these variables while defining the classes.

Less Mixins

- Next we add a mixin into the file as follows:

```
.zero-margin (@pad-up-dn: 0px, @pad-left-right: 0px) {
    margin:0px auto;
    padding: @pad-up-dn @pad-left-right;
}
```

We will make use of this to define several row classes next.

- Using the variables and Mixin class that we defined earlier, add the following row classes to the file:

```
.row-header{
    .zero-margin();
}

.row-content {
    .zero-margin(50px,0px);
    border-bottom: 1px ridge;
```

```
    min-height:400px;
}

.footer{
    background-color: @background-pale;
    .zero-margin(20px, 0px);
}

.jumbotron {
    .zero-margin(70px,30px);
    background: @background-light ;
    color:floralwhite;
}

address{
    font-size:80%;
    margin:0px;
    color:#0f0f0f;
}

body{
    padding:50px 0px 0px 0px;
    z-index:0;
}

.navbar-dark {
     background-color: @background-dark;
}

.tab-content {
    border-left: 1px solid @lt-gray;
    border-right: 1px solid @lt-gray;
    border-bottom: 1px solid @lt-gray;
```

Note the use of the variables and the mixin with various parameters in defining the classes.

Nesting Selectors

- Next we add a carousel class to illustrate the use of nesting of classes in Less, as follows:

```
.carousel {
    background:@background-dark;

    .carousel-item {
```

```
        height: @carousel-item-height;
        img {
            position: absolute;
            top: 0;
            left: 0;
            min-height: 300px;
        }
    }
}

#carouselButton {
    right:0px;
    position: absolute;
    bottom: 0px;
    z-index: 1;
}
```

Installing and using the lessc Compiler

- Now we install the node module to support the compilation of the Less file. To do this, type the following at the command prompt:

```
1
2
```

```
npm install -g less@2.7.2
```

This will install the *less* NPM module globally so that it can be used by any project. **Note: if you are executing this on a Mac or Linux machine, you may need to add "sudo" to the beginning of this command**. This will make available the *lessc* compiler for us so that we can compile Less files.

- Next, go to the CSS folder on your machine and rename the *styles.css* file that you have there as *styles-old.css*. This is to save the CSS file that we have been using so far. We will be creating a new *styles.css* file by compiling the Less file.
- Next type the following at the command prompt to compile the Less file into a CSS file:

```
1
2
```

```
lessc styles.less styles.css
```

- You can now do a Git commit with the message "Less".

Conclusions

In this exercise you learnt to write Less code and then automatically generating the CSS file by compiling the Less code.

From <https://www.coursera.org/learn/bootstrap-4/supplement/wUjIf/exercise-instructions-less>

Exercise (Instructions): Scss

Objectives and Outcomes

In this exercise you will learn to write Scss code and then automatically transform it into the corresponding CSS code. At the end of this exercise you will be able to:

- Write Scss code using many of the features of Scss
- Automatically convert the Scss code into CSS

Adding Scss Variables

- Open the *conFusion* project in a text editor of your choice. In the css folder, create a file named *styles.scss*. We will add the Scss code into this file.
- Add the following Scss variables into the file:

```
$lt-gray: #ddd;
$background-dark: #512DA8;
$background-light: #9575CD;
$background-pale: #D1C4E9;

// Height variables
$carousel-item-height: 300px;
```

We have just added a few color and a height variable. We will make use of these variables while defining the classes.

Scss Mixins

- Next we add a mixin into the file as follows:

```
@mixin zero-margin($pad-up-dn, $pad-left-right) {
    margin:0px auto;
    padding: $pad-up-dn $pad-left-right;
}
```

We will make use of this to define several row classes next.

- Using the variables and Mixin class that we defined earlier, add the following row classes to the file:

```
.row-header{
    @include zero-margin(0px,0px);
}

.row-content {
    @include zero-margin(50px,0px);
    border-bottom: 1px ridge;
    min-height:400px;
}

.footer{
    background-color: $background-pale;
    @include zero-margin(20px, 0px);
}

.jumbotron {
    @include zero-margin(70px,30px);
    background: $background-light ;
    color:floralwhite;
}

address{
    font-size:80%;
    margin:0px;
    color:#0f0f0f;
}

body{
    padding:50px 0px 0px 0px;
    z-index:0;
}

.navbar-dark {
     background-color: $background-dark;
}

.tab-content {
    border-left: 1px solid $lt-gray;
    border-right: 1px solid $lt-gray;
    border-bottom: 1px solid $lt-gray;
```

Note the use of the variables and the mixin with various parameters in defining the classes.

Nesting Selectors

- Next we add a carousel class to illustrate the use of nesting of classes in Scss, as follows:

```
1
```

```
.carousel {
    background:$background-dark;

    .carousel-item {
        height: $carousel-item-height;
        img {
            position: absolute;
            top: 0;
            left: 0;
            min-height: 300px;
        }
    }
}

#carouselButton {
    right:0px;
    position: absolute;
    bottom: 0px;
    z-index: 1;
}
```

Installing and using the node-sass module

- Now we install the node module to support the compilation of the Scss file to a CSS file. To do this, type the following at the command prompt:

```
    npm install --save-dev node-sass@4.7.2
```

  This will install the *node-sass* NPM module into your project and also add it as a development dependency in your package.json file.

- Next open your package.json file and add the following line into the scripts object there. This adds a script to enable the compilation of the Scss file into a CSS file:

```
    "scss": "node-sass -o css/ css/"
```

- In order to transform the Scss file to a CSS file, type the following at the prompt:

```
    npm run scss
```

- You can now do a Git commit with the message "Sass".

Conclusions

In this exercise you learnt to write Scss code and then automatically generating the CSS file by compiling the Scss code.

From <https://www.coursera.org/learn/bootstrap-4/supplement/Q2amj/exercise-instructions-scss>

Web Tools

Wednesday, September 7, 2022 5:29 PM

This module rounds out our discussions on Bootstrap JavaScript components. Then we discuss CSS preprocessors, Less and Sass. Finally, we discuss building and deployment of our Web projects through task automation using NPM scripts, and task runners like Grunt and Gulp. The final assignment of this course needs to be completed at the end of this module.

Learning Objectives

- Write Bootstrap JS component controls using jQuery and JavaScript
- Develop CSS code using preprocessors like Less and Sass
- Develop task automation using NPM Scripts
- Formulate task automation using Task runners like Grunt and Gulp

From <https://www.coursera.org/learn/bootstrap-4/home/week/4>

Building and Deployment: Objectives and Outcomes

In this lesson you will learn about building and deploying your web project. You will learn to automate your web development tasks using NPM scripts. At the end of this lesson, you will be able to:

- Configure NPM scripts and automate your web development
- Prepare your project for being hosted on a web server

From <https://www.coursera.org/learn/bootstrap-4/supplement/SKMQe/building-and-deployment-objectives-and-outcomes>

So far, we have been concentrating on designing and
implementing our website, be it HTML, CSS, or JavaScript code.
Play video starting at ::16 and follow transcript0:16
Once your website is ready, the next step is to be able to build your website and
to deploy it to a web server, so that it becomes publicly available.
Play video starting at ::27 and follow transcript0:27
Then our second set of steps that we need to undergo before
your website is ready for deployment on a web server.
This is what we're going to look at in this and the next lesson.
Play video starting at ::40 and follow transcript0:40
As I mentioned, web development and
deployment involves a lot of repetitive tasks.
Obviously, the writing of the HTML/CSS and JavaScript code is one part of it.
But the other part is the fact that if you write
Play video starting at ::59 and follow transcript0:59
the CSS code using one of the CSS preprocessing languages, like Sass or
Less, then you need to convert that code into the corresponding CSS code.
Thereafter, you need to do additional processing on your
CSS files like minification, compaction, and concatenation.
We'll talk a little bit more about these in the next few slides.
Similarly, with your JavaScript code, you need to do
JSHinting, checking for potential errors.
Then concatenation of various script files and

then even uglification and the mangling of the code.
We'll talk a little bit about that in the next few slides.
All these tasks are repetitive tasks, which we would like to automate
as far as possible, so that we can concentrate on the actual design and
building of our website, rather than these repetitive tasks.
So that the DRY principle, do not repeat yourself principle,
is very essential in this case.
We don't want to be wasting our time on such repetitive combustion tasks,
and instead, try to automate them as far as possible so
that they can be executed whenever required.
Let's talk about some of these repetitive tasks in a bit more detail.
Play video starting at :2:28 and follow transcript2:28
Let's take the example of CSS as a case.
When we write CSS code, we often write the code
using one of the preprocessing languages likes Less or Sass.
Now once the code is written, then it needs to be converted into CSS by using
one of the preprocessors as we have seen in the previous lesson.
Play video starting at :2:54 and follow transcript2:54
There, we may need to do some vendor specific prefixing for
our CSS code to address the issues that might arise with various browsers.
So this is where the task of doing auto-prefixing is used,
whereby this can be automatically done for us.
Similarly, once your CSS code is written, obviously,
the way we write CSS code is to be humanly readable.
But for a machine, it doesn't really care whether there are enough spaces
between the code or whether it is properly formatted or not.
Play video starting at :3:35 and follow transcript3:35
So, minification is the process of removing all the unnecessary characters,
the white space, newlines, comments, from your CSS code.
So that your end result is a very compact file with minimum number
of characters, so that may be served up very, very quickly.
But at the same time,
you want to preserve the functionality that you designed in your CSS code.
Play video starting at :4:4 and follow transcript4:04
Similarly, you might distribute your CSS code into many files,
while you are designing and building your website.
You may want to concatenate all of them into a single CSS file at the end,
so that only a single CSS file needs to be downloaded
by the browser when it is rendering your website.
So concatenation is yet
another task that is involved when you're building your website.
Play video starting at :4:33 and follow transcript4:33
Similarly, when you write JavaScript code, whether it is pure JavaScript or
jQuery or one of the frameworks that we will use in
the future courses of this specialization, you would need to write JavaScript code.
So once you've written the JavaScript code, you'll want to be able to check
the JavaScript code for errors and potential problems.
Things like missing semicolons,
Play video starting at :5:2 and follow transcript5:02
Improper use of the language, and so on, so what we call as static code analysis.
So if you want to be able to perform this, even before we deploy our
Play video starting at :5:14 and follow transcript5:14
website on the web server.
Similarly, we might organize our code into multiple JavaScript files.
When we actually deploy, we may want to concatenate all these files into
a single JavaScript file and then use that in our web pages.

And this concatenation can be done automatically.
Similarly, the uglification of the JavaScript code, which stands for
minification, meaning removing all the unnecessary white space and
comments and so on.
And mangling of the code, meaning reducing the names of the local
variables to single letters wherever feasible.
Now, from a perspective of a computer,
it doesn't really care what the code looks like as long as it works correctly.
For human readable format, we obviously write code in a lot more
elaborate manner, so that it's easier for us to follow what the code is doing.
So when you actually deploy,
you may wish to remove all the extraneous features from your code.
And then compact it so that the minimum amount of code will be served up.
At the same time, upon completion of the JSHint concatenation and
uglification process,
you may still want to make sure that your resulting code will still work correctly.
So, you may want to recheck your code for potential errors.
Play video starting at :6:47 and follow transcript6:47
CSS and JavaScript are two major aspects of your web development that you
obviously need to pay a lot of attention to.
But there are many other smaller tasks that you need to perform
before your website is ready for deployment.
You might include a lot of images into your web pages.
Once your website is ready, you may want to compact those images so
that you optimize the file sizes, so
that their images will be minimum sized files to be deployed.
Play video starting at :7:24 and follow transcript7:24
Similarly, while you're doing development, you may be making modifications to,
for example, your Sass files or your JavaScript code.
When such modifications are done,
you want to be able to automatically carry out those tasks,
like concatenation, minification, and uglification tasks.
So we could use watch tasks,
whose main job is to keep a watch on all these files.
And if any changes are done to these files,
the tasks will be automatically executed.
This will free up a lot of our time from doing repetitive tasks manually.
Play video starting at :8:10 and follow transcript8:10
We'll look at some of this in more detail in the exercises that follow.
Play video starting at :8:16 and follow transcript8:16
One other aspect, while you're doing your development,
is to be able to serve up your code and
watch the code in your browser.
So we have seen the use of the live server in our previous development,
where we had the server up and running and serving up the code.
So that we can see the changes that we make instantaneously
Play video starting at :8:48 and follow transcript8:48
being rendered in the browser.
So, for this, we need server and livereload mechanism, and
live server that we saw earlier is one such example of how we can achieve this.
Play video starting at :9:2 and follow transcript9:02
Finally, if you are writing code, you obviously need to carry out testing
of your code, which, in Bootstrap's case, is a lot less of consideration.
But as you go on to use various JavaScript frameworks,
testing becomes an equally important task.
Play video starting at :9:23 and follow transcript9:23

Finally, you want to be able to accomplish all these tasks and
then build up your final deployment code that can then be uploaded
to your web server to make your website available for the general public.
Play video starting at :9:42 and follow transcript9:42
The steps involved in building up your site for deployment, what we call
as building up the distribution files, is also an equally important task.
We'll look at some of these through examples in the next exercise and
also the next lesson where we will look at task runners.

From <https://www.coursera.org/learn/bootstrap-4/lecture/4GHx2/building-and-deployment>

As a first approach to automating our tasks,
let's look at the example of using NPM scripts.
Indeed, we had already started using NPM scripts in this course in the earlier lessons.
Let's quickly revisit our package.json file to see how we have used a couple of
NPM scripts in our package.json file to automate a couple of tasks.
Taking a look at our package.json file,
you see that in the package.json file,
we had this JSON object called scripts.
Inside the scripts, we had this property called start that we defined here which we
specified as NPM run lite and also we specified this lite as lite-server.
So, then you start up your web development.
We also started up our lite-server by typing on NPM start at the prompt.
Now, what this enabled us is to start up the lite-server in
our project directory and start saving up the files from the directory,
so that we can view it in a browser.
We found this to be a useful approach to be able to watch live
the changes that we make to our files being reflected immediately in the browser.
Now, in other tasks that we configured and used in the previous lesson was SCSS,
which is used for transforming the SCSS code into the corresponding CSS code.
So, we wrote this quote unquote NPM script called as SCSS which executed the node sass
by looking for the SCSS files
in the CSS folder and then transforming them into the corresponding CSS code.
And to execute this,
we typed NPM run SCSS at the command prompt.
So, we've already see that we have used a couple of
NPM scripts to automate some tasks during our web development.
So to summarize, NPM scripts are supported through
this scripts property in the package.json file as we have seen in the example earlier.
There are several scripts that are supported.
One in particular that is of interest to us is the start-script,
so that you can add the prompt type NPM start,
and then the corresponding script referred to by the start, will be started.
Now, we can define arbitrary scripts in
the scripts property and then run them by saying NPM run and the name of the script,
as you saw the use of NPM run SCSS or NPM run lite and so on.
We are going to leverage this to be able to develop a few additional scripts that will
automate a lot of those tasks that we talked about in the previous lecture.
So, to help us understand how we can leverage NPM scripts to
automate a lot of our web development tasks,
we will learn how to configure the NPM scripts in the package.json file,
and then execute the scripts,
sometimes automatically or sometimes manually invoke
the scripts in order to do the building and deployment of our website.

From <https://www.coursera.org/learn/bootstrap-4/lecture/H4Cyz/npm-scripts>

Exercise (Instructions): NPM Scripts Part 1

Objectives and Outcomes

In this exercise, you will learn to set up NPM scripts by modifying the *package.json* file. At the end of this exercise, you will be able to:

- Watch for changes to the *styles.scss* file and automatically compile it to the css file.
- Run multiple NPM scripts in parallel using parallelshell NPM module.

Moving JS to Script file

- Create a folder named *js* and in that folder create a file named *scripts.js*.
- Open *index.html* and from this file cut out all the JQuery script that we added to it and move the code to the *scripts.js* file that we created above.
- Then, update the *index.html* file to include the *scripts.js* file by adding the following line:

```
    <script src="js/scripts.js"></script>
```

- Add the same line to the scripts block in *aboutus.html* and *contactus.html*:

Watching for Changes and Parallelshell

- First, we install two NPM packages *onchange* and *parallelshell* as follows:

```
npm install --save-dev onchange@3.3.0 parallelshell@3.0.2
```

- Then, add the following two script items to *package.json* if you are doing the exercise on a MacOS computer or a Linux computer:

```
    "watch:scss": "onchange 'css/*.scss' -- npm run scss",
    "watch:all": "parallelshell 'npm run watch:scss' 'npm run lite'"
```

- **NOTE:** If you are doing the exercise on a Windows computer, please use the following two script items instead of the above:

```
    "watch:scss": "onchange \"css/*.scss\" -- npm run scss",
    "watch:all": "parallelshell \"npm run watch:scss\" \"npm run lite\""
```

- You will also update the start script as follows:

```
    "start": "npm run watch:all",
```

- Then, type the following at the prompt to start watching for changes to the SCSS file, compile it to CSS, and run the server:

```
npm start
```

- Now, whenever you make any changes to *styles.scss* file, it will automatically be compiled to the corresponding css file.
- Do a Git Commit with the message "NPM Scripts Part 1".

Conclusions

In this exercise, you learnt how to set up a watch task to watch for changes to a file and automatically run tasks upon detecting changes.

From <https://www.coursera.org/learn/bootstrap-4/supplement/nIb73/exercise-instructions-npm-scripts-part-1>

Exercise (Instructions): NPM Scripts Part 2

Objectives and Outcomes

In this exercise you will learn to build a distribution folder containing the files that can be deployed on a web server hosting your project. This distribution folder would be built from your project files using various NPM packages and scripts. At the end of this exercise, you will be able to:

- Clean out a folder using the clean NPM module.
- Copy files from one folder to another
- Prepare a minified and concatenated css file from all the css files used in your project
- Prepare an uglified and concatenated JS file containing all the JS code used in your project

Cleaning up a Distribution Folder

- Install the *rimraf* npm module by typing the following at the prompt:

```
npm install --save-dev rimraf@2.6.2
```

- Then, set up the following script:

```
    "clean": "rimraf dist",
```

Copying Fonts

- Your project uses font-awesome fonts. These need to be copied to the distribution folder. To help us do this, install the *copyfiles* NPM module globally as follows:

```
npm -g install copyfiles@2.0.0
```

Remember to use *sudo* on mac and Linux.

- Then set up the following script:

```
    "copyfonts": "copyfiles -f node_modules/font-awesome/fonts/* dist/fonts",
```

Compressing and Minifying Images

- We use the *imagemin-cli* NPM module to help us to compress our images to reduce the size of the images being used in our project. Install the *imagemin-cli* module as follows:

```
npm -g install imagemin-cli@3.0.0
```

Remember to use *sudo* on mac and Linux. **NOTE:** Some students have encountered issues with imagemin-cli not installing its plugins due to issues with global permissions on Mac. In that case try

```
sudo npm install -g imagemin-cli@3.0.0 --unsafe-perm=true --allow-root
```

- Then set up the following script:

```
    "imagemin": "imagemin img/* --out-dir='dist/img'",
```

Preparing the Distribution Folder

- Open *.gitignore* and update it as follows. We do not want the dist folder to be checked into the git repository.

```
node_modules
dist
```

- Then, install the *usemin-cli, cssmin, uglifyjs* and *htmlmin* NPM packages as follows:

```
npm install --save-dev usemin-cli@0.5.1 cssmin@0.4.3 uglifyjs@2.4.11 htmlmin@0.0.7
```

- Add the following two scripts to the package.json file:

```
    "usemin": "usemin contactus.html -d dist --htmlmin -o dist/contactus.html && usemin aboutus.html -d dist --htmlmin -o dist/aboutus.html && usemin index.html -d dist --htmlmin -o dist/index.html",
    "build": "npm run clean && npm run imagemin && npm run copyfonts && npm run usemin"
```

- Open *index.html* and surround the css links inclusion code as follows:

```
    <!-- build:css css/main.css -->
    <link rel="stylesheet" href="node_modules/bootstrap/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="node_modules/font-awesome/css/font-awesome.min.css">
    <link rel="stylesheet" href="node_modules/bootstrap-social/bootstrap-social.css">
    <link href="css/styles.css" rel="stylesheet">
    <!-- endbuild -->
```

- Do the same change in *aboutus.html* and *contactus.html*
- Similarly, open *index.html* and surround the js script inclusion code as follows:

```
    <!-- build:js js/main.js -->
    <script src="node_modules/jquery/dist/jquery.slim.min.js"></script>
    <script src="node_modules/popper.js/dist/umd/popper.min.js"></script>
    <script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>
```

```
    <script src="js/scripts.js"></script>
    <!-- endbuild -->
```

- Do the same change in *aboutus.html* and *contactus.html*
- To build the distribution folder, you can type the following at the prompt:

```
npm run build
```

- This will build the *dist* folder containing the files that are a self-contained version of your project. You can now copy the contents of this folder to a web server that hosts your website.
- After verifying that the dist folder is built correctly, you can now do a git commit with the message "NPM Scripts Part 2"

Conclusions

In this exercise, you learnt the various steps to build the project for deployment using NPM scripts.

index

package

scripts

processBlocks

From <https://www.coursera.org/learn/bootstrap-4/supplement/hWE5b/exercise-instructions-npm-scripts-part-2>

Building and Deployment: Task Runners: Objectives and Outcomes

In this lesson you will learn about JavaScript based Task runners, Grunt and Gulp. You will learn to automate your web development tasks using these tools. At the end of this lesson, you will be able to:

- Configure Grunt tasks and automate your web development using Grunt
- Define Gulp tasks in code to automate the web development using Gulp

From <https://www.coursera.org/learn/bootstrap-4/supplement/LNtXV/building-and-deployment-task-runners>

As we understood from the previous lesson building and
deployment of websites involves a lot of steps.
And we looked at the use of [INAUDIBLE] scripts for
doing the building and deployment tasks.
In this lesson, we will concentrate on task renders.
Two in particular, Grunt and Gulp, and try to understand how
they facilitate their automation of the various web developement tasks.
Play video starting at ::41 and follow transcript0:41
The primary task of a Task Runner is to enable to us to configure the tasks and
then rerun them automatically as it when request.
Play video starting at ::51 and follow transcript0:51
In the previous Lesson, we show the user NPM scripts that we set up in
the package.json file in order to automate some of these tasks,
including the use of the on change to automatically
run some of these tasks based upon changes to files that are made.
Play video starting at :1:15 and follow transcript1:15
Now, thinking back, we realize that we are in the Node ecosystem.
And what Node enables us to do is to run JavaScript on the desktop.
Now, better luck than people, the programmers as you know of them,
Play video starting at :1:36 and follow transcript1:36
took on the task of using JavaScript itself to write
Play video starting at :1:43 and follow transcript1:43
applications that enable us to set up and run these tasks automatically.
So this is where the task runners came into the picture.
Grunt, Gulp, Cake, Brunch, Broccoli.
What's this obsession with food?
Play video starting at :2:1 and follow transcript2:01
Well when you're sitting in a single place for a long time, you develop a big
appetite, Grunt and Gulp being the two most successful in this field.
We'll look at both of them in more detail in this lesson.
Play video starting at :2:17 and follow transcript2:17
These enable us to double up automated tasks for
executing and building and deploying our website.
Play video starting at :2:27 and follow transcript2:27
As I mentioned, the JavaScript based task runners, the two
of them in particular that we will deal with in this lesson are Grunt and Gulp.
Grunt operates based on doing configuration
whereas Gulp concentrates more on code.
We'll look at the two of them in more detail as we go along.
But both of them pretty much enable us to accomplish the same kind of tasks.
Both are built around plugins.
So, both Grunt and Gulp provide a framework for which
you write various plugins that enable you to perform these kinds of tasks.
So Grunt and Gulp together with their plugins enable us to configure and
execute various tasks.
If you ask me, is Grunt better or is Gulp better, that's a tough question to answer.
Play video starting at :3:26 and follow transcript3:26
The Grunt people are grunting all over,
Play video starting at :3:29 and follow transcript3:29
saying that theirs is the best way of doing it.
Play video starting at :3:33 and follow transcript3:33
Which the Gulp community finds hard to swallow.
Play video starting at :3:38 and follow transcript3:38
Both these communities can easily give your televangelists a run for their money.

Play video starting at :3:45 and follow transcript3:45
But, from my perspective,
Play video starting at :3:49 and follow transcript3:49
I look at any of these tools as just mechanisms to get my work done.
Play video starting at :3:55 and follow transcript3:55
So long as it gets my work done, I'm happy with it.
If one makes it easier to accomplish a set of tasks that has,
then I choose the one over the other for that particular project.
Play video starting at :4:10 and follow transcript4:10
So my take is event VM scripts satisfies you, stable them.
If Grunt is the way to go, be my guest.
If Gulp is worth satisfies you, take you a big gulp and you'll be satisfied.
Play video starting at :4:27 and follow transcript4:27
So having said that, let's take a look at both Grunt and Gulp.
I will show you how you perform the same set of tasks
that I did with the npm scripts in the previous lesson and the exercise.
Play video starting at :4:45 and follow transcript4:45
By showing you the Grunt and the Gulp way of doing the same
sort of tasks in the exercises that follow this lecture.
[MUSIC]
: Added to Selection. Press [CTRL + S] to save as a note

From <https://www.coursera.org/learn/bootstrap-4/lecture/SDnS9/task-runners>

Exercise (Instructions): Grunt Part 1

Objectives and Outcomes

In this exercise, you will learn to use Grunt, the task runner. You will install Grunt CLI and install Grunt packages using NPM. Thereafter you will configure a Grunt file with a set of tasks to build and serve your web project. At the end of this exercise, you will be able to:

- Install Grunt CLI and Grunt packages in your project
- Configure a Grunt file with a set of tasks to build a web project from a source, and serve the built project using a server.

Installing Grunt

- At the command prompt, type the following to install Grunt command-line interface (CLI):

```
npm install -g grunt-cli@1.2.0
```

This will install the Grunt CLI globally so that you can use them in all projects.

- Next install Grunt to use within your project. To do this, go to the *conFusion* folder and type the following at the prompt:

```
npm install grunt@1.0.2 --save-dev
```

This will install local per-project Grunt to use within your project.

Creating a Grunt File

- Next you need to create a Grunt file containing the configuration for all the tasks to be run when you use Grunt. To do this, create a file named *Gruntfile.js* in the *conFusion* folder.
- Next, add the following code to Gruntfile.js to set up the file to configure Grunt tasks:

```
'use strict';

module.exports = function (grunt) {
  // Define the configuration for all the tasks
  grunt.initConfig({

  });
};
```

This sets up the Grunt module ready for including the grunt tasks inside the function above.

Compiling SCSS to CSS

- Next, we are going to set up our first Grunt task. The SASS task converts the SCSS code to CSS. To do this, you need to include some Grunt modules that help us with the tasks. Install the following modules by typing the following at the prompt:

```
npm install grunt-sass@2.1.0 --save-dev
npm install time-grunt@1.4.0 --save-dev
npm install jit-grunt@0.10.0 --save-dev
```

The first one installs the Grunt module for SCSS to CSS conversion. The time-grunt module generates time statistics about how much time each task consumes, and jit-grunt enables us to include the necessary downloaded Grunt modules when needed for the tasks.

- Now, configure the SASS task in the Gruntfile as follows, by including the code inside the function in *Gruntfile.js*:

```
'use strict';

module.exports = function (grunt) {
    // Time how long tasks take. Can help when optimizing build times
    require('time-grunt')(grunt);

    // Automatically load required Grunt tasks
    require('jit-grunt')(grunt);

    // Define the configuration for all the tasks
    grunt.initConfig({
        sass: {
            dist: {
                files: {
                    'css/styles.css': 'css/styles.scss'
                }
            }
        }
    });

    grunt.registerTask('css', ['sass']);

};
```

- Now you can run the grunt SASS task by typing the following at the prompt:

```
grunt css
```

Watch and Serve Tasks

- The final step is to use the Grunt modules watch and browser-sync to spin up a web server and keep a watch on the files and automatically reload the browser when any of the watched files are updated. To do this, install the following grunt modules:

```
npm install grunt-contrib-watch@1.0.0 --save-dev
npm install grunt-browser-sync@2.2.0 --save-dev
```

- After this, we will configure the browser-sync and watch tasks by adding the following code to the Grunt file:

```
'use strict';

module.exports = function (grunt) {
    // Time how long tasks take. Can help when optimizing build times
    require('time-grunt')(grunt);

    // Automatically load required Grunt tasks
    require('jit-grunt')(grunt);

    // Define the configuration for all the tasks
    grunt.initConfig({
        sass: {
            dist: {
                files: {
                    'css/styles.css': 'css/styles.scss'
                }
            }
        }
    });

    grunt.registerTask('css', ['sass']);

};
```

```
,
        watch: {
            files: 'css/*.scss',
            tasks: ['sass']
        },
        browserSync: {
            dev: {
                bsFiles: {
                    src : [
                        'css/*.css',
                        '*.html',
                        'js/*.js'
                    ]
                },
                options: {
                    watchTask: true,
                    server: {
                        baseDir: "./"
                    }
                }
            }
        }
```

- Then add the following task to the Grunt file:

```
    grunt.registerTask('default', ['browserSync', 'watch']);
```

- Now if you type the following at the command prompt, it will start the server, and open the web page in your default browser. It will also keep a watch on the files in the css folder, and if you update any of them, it will compile the scss file into css file and load the updated page into the browser (livereload)

```
grunt
```

- Do a Git commit with the message "Grunt Part 1".

Conclusions

In this exercise you have learnt how to configure a Grunt file to perform several tasks. You were able to start a server with livereload to serve the web page.

From <https://www.coursera.org/learn/bootstrap-4/supplement/6uMvP/exercise-instructions-grunt-part-1>

Exercise (Instructions): Grunt Part 2

Objectives and Outcomes

In this exercise, you will continue to learn to use Grunt, the task runner. You will configure the Grunt file with a set of additional tasks to build your web project. At the end of this exercise, you will be able to:

- Configure a Grunt file with a set of tasks to build your web project from a source.

Copying the Files and Cleaning Up the Dist Folder

- Next you will install the Grunt modules to copy over files to a distribution folder named dist, and clean up the dist folder when needed. To do this, install the following Grunt modules:

```
npm install grunt-contrib-copy@1.0.0 --save-dev
npm install grunt-contrib-clean@1.1.0 --save-dev
```

- You will now add the code to perform the copying of files to the dist folder, and cleaning up the dist folder. To do this, add the following code to *Gruntfile.js*. This should be added right after the configuration of the SASS task.:

```
,
        copy: {
            html: {
                files: [
                {
                    //for html
```

```
                    expand: true,
                    dot: true,
                    cwd: './',
                    src: ['*.html'],
                    dest: 'dist'
                }]
            },
            fonts: {
                files: [
                {
                    //for font-awesome
                    expand: true,
                    dot: true,
                    cwd: 'node_modules/font-awesome',
                    src: ['fonts/*.*'],
                    dest: 'dist'
                }]
            }
        },

        clean: {
            build: {
                src: [ 'dist/']
            }
        }
```

- Remember to add the comma after the end of the SASS task.

Compressing and Minifying Images

- Next we install the grunt-contrib-imagemin module and use it to process the images. To install this module type at the prompt:

```
npm install grunt-contrib-imagemin@2.0.1 --save-dev
```

- Then, configure the imagemin task as shown below in the Gruntfile:

```
,
        imagemin: {
            dynamic: {
                files: [{
                    expand: true,                  // Enable dynamic expansion
                    cwd: './',                   // Src matches are relative to t
his path
                    src: ['img/*.{png,jpg,gif}'],   // Actual patterns to match
                    dest: 'dist/'                  // Destination path prefix
                }]
            }
```

```
        }
```

Preparing the Distribution Folder and Files

- We are now going to use the Grunt *usemin* module together with *concat*, *cssmin*, *uglify* and *filerev* to prepare the distribution folder. To do this, install the following Grunt modules:

```
npm install grunt-contrib-concat@1.0.1 --save-dev
npm install grunt-contrib-cssmin@2.2.1 --save-dev
npm install grunt-contrib-htmlmin@2.4.0 --save-dev
npm install grunt-contrib-uglify@3.3.0 --save-dev
npm install grunt-filerev@2.3.1 --save-dev
npm install grunt-usemin@3.1.1 --save-dev
```

- Next, update the task configuration within the Gruntfile.js with the following additional code to introduce the new tasks:

```

```

```
,
        useminPrepare: {
            foo: {
                dest: 'dist',
                src: ['contactus.html','aboutus.html','index.html']
            },
            options: {
                flow: {
                    steps: {
                        css: ['cssmin'],
                        js:['uglify']
                    },
                    post: {
                        css: [{
                            name: 'cssmin',
                            createConfig: function (context, block) {
                            var generated = context.options.generated;
                                generated.options = {
                                    keepSpecialComments: 0, rebase: false
                                };
                            }
                        }]
                    }
                }
            }
        },

        // Concat
        concat: {
            options: {
                separator: ';'
            },

            // dist configuration is provided by useminPrepare
            dist: {}
        },

        // Uglify
        uglify: {
```

- Next, update the jit-grunt configuration as follows, to inform it that useminPrepare task depends on the usemin package:

```
require('jit-grunt')(grunt, {
  useminPrepare: 'grunt-usemin'
});
```

- Next, update the Grunt build task as follows:

```
    grunt.registerTask('build', [
        'clean',
        'copy',
        'imagemin',
        'useminPrepare',
        'concat',
        'cssmin',
        'uglify',
        'filerev',
        'usemin',
        'htmlmin'
    ]);
```

- Now if you run Grunt, it will create a dist folder with the files structured correctly to be distributed to a server to host your website. To do this, type the following at the prompt:

```
grunt build
```

Conclusions

In this exercise you have learnt how to configure a Grunt file to perform several tasks. You were able to build a distribution folder for your web project.

From <https://www.coursera.org/learn/bootstrap-4/supplement/SIHkS/exercise-instructions-grunt-part-2>

Exercise (Instructions): Gulp Part 1

Objectives and Outcomes

In this exercise, you will learn to use Gulp, the task runner. You will install Gulp CLI and install Gulp plugins using NPM. Thereafter you will configure a Gulp file with a set of tasks to build and serve your web project. At the end of this exercise, you will be able to:

- Install Gulp CLI and Gulp plugins in your project
- Configure a Gulp file with a set of tasks to build a web project from a source, and serve the built project using a server.

Installing Gulp

- At the command prompt, type the following to install Gulp command-line interface (CLI) globally:

```
    npm install -g gulp-cli@2.0.1
```

This will install the Gulp globally so that you can use it in all projects.

- Next install Gulp to use within your project. To do this, go to the *conFusion* folder and type the

following at the prompt:

```
    npm install gulp@3.9.1 --save-dev
```

This will install local per-project Gulp to use within your project.

Install Gulp Plugins for SASS and Browser-Sync

- Install all the Gulp plugins that you will need for this exercise. To do this, type the following at the command prompt:

```
npm install gulp-sass@3.1.0  browser-sync@2.23.6 --save-dev
```

Creating a Gulp File

- Next you need to create a Gulp file containing the tasks to be run when you use Gulp. To do this, create a file named *gulpfile.js* in the *conFusion* folder.

Loading Gulp Plugins

- Load in all the Gulp plugins by including the following code in the Gulp file:

```
'use strict';

var gulp = require('gulp'),
    sass = require('gulp-sass'),
    browserSync = require('browser-sync');
```

Adding Gulp Tasks for SASS and Browser-Sync

- Next, we will add the code for the SASS task, the Browser-Sync task and the default task as follows:

```
gulp.task('sass', function () {
  return gulp.src('./css/*.scss')
    .pipe(sass().on('error', sass.logError))
    .pipe(gulp.dest('./css'));
});

gulp.task('sass:watch', function () {
  gulp.watch('./css/*.scss', ['sass']);
});

gulp.task('browser-sync', function () {
   var files = [
      './*.html',
      './css/*.css',
      './img/*.{png,jpg,gif}',
      './js/*.js'
   ];

   browserSync.init(files, {
      server: {
         baseDir: "./"
      }
   });

});

// Default task
gulp.task('default', ['browser-sync'], function() {
    gulp.start('sass:watch');
});
```

- Save the Gulp file

Running the Gulp Tasks

- At the command prompt, if you type *gulp* it will run the default task:

```
gulp
```

- Do a Git commit with the message "Gulp Part 1".

Conclusions

In this exercise, you learnt to use Gulp, install Gulp plugins, configure the gulpfile.js and then use Gulp to automate the web development tasks.

From <https://www.coursera.org/learn/bootstrap-4/supplement/mLaHL/exercise-instructions-gulp-part-1>

Exercise (Instructions): Gulp Part 2

Objectives and Outcomes

In this exercise, you will continue to learn to use Gulp. Thereafter you will configure a Gulp file with a set of tasks to build and serve your web project. At the end of this exercise, you will be able to:

- Configure the Gulp file with a set of tasks to build the distribution folder for the web project.

Copying the Files and Cleaning up the Dist Folder

- We will now create the tasks for copying the font files and cleaning up the distribution folder. To do this we will first install the *del* Node module and require it in the Gulp file as follows:

```
npm install del@3.0.0 --save-dev
```

```
var ...
    del = require('del'),
    ...
```

- Next, we will add the code for the Clean task and the copyfonts task as follows:

```
// Clean
gulp.task('clean', function() {
    return del(['dist']);
});

gulp.task('copyfonts', function() {
   gulp.src('./node_modules/font-awesome/fonts/**/*.{ttf,woff,eof,svg}*')
   .pipe(gulp.dest('./dist/fonts'));
});
```

Compressing and Minifying Images

- We will now install the *gulp-imagemin* plugin and configure the *imagemin* task. To do this we install the plugin and require it as follows:

```
npm install gulp-imagemin@4.1.0 --save-dev
```

```
var ...
    imagemin = require('gulp-imagemin'),
    ...
```

- Next, we create the *imagemin* task as follows:

```
// Images
gulp.task('imagemin', function() {
  return gulp.src('img/*.{png,jpg,gif}')
    .pipe(imagemin({ optimizationLevel: 3, progressive: true, interlaced: true })
)
    .pipe(gulp.dest('dist/img'));
});
```

Preparing the Distribution Folder and Files

- We now install the gulp-usemin and other related Gulp plugins and require them as follows:

```
npm install gulp-uglify@3.0.0 gulp-usemin@0.3.29 gulp-rev@8.1.1 gulp-clean-css@
3.9.3 gulp-flatmap@1.0.2 gulp-htmlmin@4.0.0 --save-dev
```

```
var ...
    uglify = require('gulp-uglify'),
    usemin = require('gulp-usemin'),
    rev = require('gulp-rev'),
    cleanCss = require('gulp-clean-css'),
    flatmap = require('gulp-flatmap'),
    htmlmin = require('gulp-htmlmin');
```

- We configure the usemin and the build task as follows:

```
gulp.task('usemin', function() {
  return gulp.src('./*.html')
  .pipe(flatmap(function(stream, file){
      return stream
        .pipe(usemin({
            css: [ rev() ],
            html: [ function() { return htmlmin({ collapseWhitespace: true })} ],
            js: [ uglify(), rev() ],
            inlinejs: [ uglify() ],
            inlinecss: [ cleanCss(), 'concat' ]
        }))
    }))
    .pipe(gulp.dest('dist/'));
});

gulp.task('build',['clean'], function() {
    gulp.start('copyfonts','imagemin','usemin');
});
```

- Save the Gulp file

Running the Gulp Tasks

- At the command prompt, if you type *gulp build* it will run the build task:

```
gulp build
```

- Do a Git commit with the message "Gulp Part 2"

Conclusions

In this exercise, you learnt to use Gulp, install Gulp plugins, configure the gulpfile.js and then use Gulp to automate the web development tasks.

From <https://www.coursera.org/learn/bootstrap-4/supplement/YL7cj/exercise-instructions-gulp-part-2>

Conclusion

Tuesday, September 13, 2022 1:51 AM

As we come to the conclusion of this course, it's time to look back and
reflect about what we've covered in this course so far.
And what is the way forward from
this course towards the other courses of this specialization.
Play video starting at ::21 and follow transcript0:21
Looking back at the big picture view of full stack web development.
We have started out our journey by looking at the front-end web UI frameworks,
in particular, Bootstrap 4, in this course.
We will cover the remaining parts of the full stack web development stack
Play video starting at ::40 and follow transcript0:40
in the rest of the courses of this specialization.
In this course, we covered Bootstrap in great detail.
We looked at responsive web design and
how the Bootstrap grid system enables us to achieve responsive websites.
We also looked at various Bootstrap CSS and JavaScript components.
Along the way, we learned how to operate within the Node ecosystem for
doing development.
We learned web tools like Git, and we installed Node and
learned about the NPM package manager.
We also learned about task runners like Grunt and Gulp.
And learned how to set up NPM scripts for
automating frequently performed tasks in web development.
As we move towards the next course of the specialization,
in the next course we'll be concentrating on Javascript frameworks.
So this is the other piece of the puzzle on front-end.
So together with UI frameworks, the Javascript frameworks facilitate
development of dynamic websites, especially the front-end.
We'll look at the remaining parts, including hybrid development and
server-side, in the remaining courses of the specialization.
I hope you enjoyed going through this course.
And I hope I'll see you back again in the next course of the specialization.

From <https://www.coursera.org/learn/bootstrap-4/lecture/sHoTk/front-end-web-ui-frameworks-bootstrap-4-conclusions>

Resources

Monday, August 29, 2022 3:26 PM

Welcome to Front-End Web UI Frameworks and Tools: Bootstrap 4: Additional Resources

Bootstrap Resources

- Bootstrap Site

Coursera Resources

- Coursera Learner Help
- Switching to a Different Session

From <https://www.coursera.org/learn/bootstrap-4/supplement/JINBo/welcome-to-front-end-web-ui-frameworks-and-tools-bootstrap-4-additional>

Full Stack Web Development: Additional Resources

PDFs of Presentations

FSWD-BigPicture.pdf

PDF File

Useful Links

- What is a Full Stack developer?
- Wait, Wait… What is a Full-stack Web Developer After All?
- The Myth of the Full-stack Developer
- Multi-tier Architecture
- What is the 3-Tier Architecture?

From <https://www.coursera.org/learn/bootstrap-4/supplement/xl0rj/full-stack-web-development-additional-resources>

Additional Resources (Git)

- Git site http://git-scm.com.
- Installing Git chapter from Pro Git
- Git reference manual
- Quick reference guides: GitHub Cheat Sheet (PDF) | Visual Git Cheat Sheet (SVG | PNG)
- Atlassian comprehensive Git tutorial

Additional Resources (Node.js and NPM)

- Nodejs.org
- Npmjs.com
- Node API Documentation
- NPM Documentation
- lite-server

Introduction to Bootstrap: Additional Resources

PDFs of the Presentations

1-Web-UI-Frameworks.pdf

PDF File

2-Intro-Bootstrap.pdf

PDF File

Exercise Resources

- (required for the exercise)
  Bootstrap4-starter

  ZIP File

  Bootstrap Official Resources

- Bootstrap 4 Home Page
- Bootstrap typography
- Migrating from Bootstrap 3 to Bootstrap 4

Front-end Web UI Frameworks

- Top 10 Front-End Frameworks of 2018
- The 5 Most Popular Front-end Frameworks Compared

From <https://www.coursera.org/learn/bootstrap-4/supplement/QjlKC/introduction-to-bootstrap-additional-resources>

Responsive Design and Bootstrap Grid System: Additional Resources

PDFs of Presentations

3-Responsive-Design.pdf

PDF File

4-Bootstrap-Grid.pdf

PDF File

Bootstrap Official Documentation

- Bootstrap Grid System

Responsive Design and Bootstrap Grid Resources

- CSS Flexible Box Layout Module Level 1 (W3C Documentation)
- A Complete Guide to Flexbox
- A Visual Guide to CSS3 Flexbox Properties
- The Bootstrap 4 Grid: What's New?
- How the Bootstrap Grid Really Works
- The Subtle Magic Behind Why the Bootstrap 3 Grid Works (a detailed explanation of why the Bootstrap grid system works the way it does, a delight to read!)
- What The Heck Is Responsive Web Design? (a short presentation that introduces responsive web design)
- Beginner's Guide to Responsive Web Design (simple introduction to responsive web design)
- The 2014 Guide to Responsive Web Design (an updated guide to responsive design)

From <https://www.coursera.org/learn/bootstrap-4/supplement/Z7VfT/responsive-design-and-bootstrap-grid-system-additional-resources>

Navigation and Navigation Bar: Additional Resources

PDFs of Presentations

1-Navigation.pdf

PDF File

2-Icon Fonts.pdf

PDF File

Official Bootstrap Resources

- Navbar
- Breadcrumbs

General

- Accessible Rich Internet Applications (ARIA) (Accessibility support and screen reader support)

Information Architecture

- Information Architecture 101: Techniques and Best Practices (Quick introduction to Information architecture with respect to website design)
- Web Site Information Architecture models (Another good resource on information architecture)
- What is information architecture? (Good definition and explanation about the topic)
- Information Architecture Tutorial (Comprehensive look from a website design perspective)

Navigation Bar Design

- Designing A Winning Navigation Menu: Ideas and Inspirations (Good suggestions on how to design navigation for a website)
- Are You Making These Common Website Navigation Mistakes? (Worth reading at least to learn what not to do)
- 3 Reasons We Should Stop Using Navigation Bars (A provocative view on navigation bars)

Breadcrumbs

- Breadcrumb Navigation Examined: Best Practices & Examples (Great suggestions on using breadcrumbs for navigation)
- Breadcrumb Navigation: A Guide On Types, Benefits And Best Practices (Another great resource on types and usage of breadcrumbs)

Icon Fonts

- Why And How To Use Icon Fonts (a good overview of icon fonts)
- Icon Fonts are Awesome (another good introduction to icon fonts)
- Font Awesome (one of the most popular icon fonts)
- Get started with FontAwesome (good official help)
- Bootstrap-Social
- The Final Nail in the Icon Fonts Coffin? (a controversial opinion piece on icon fonts)
- Using SVGs (alternative to icon fonts)

From <https://www.coursera.org/learn/bootstrap-4/supplement/Gbp6X/navigation-and-navigation-bar-additional-resources>

User Input: Additional Resources

PDFs of Presentations

3-User-Input.pdf

PDF File

Exercise Resources

contactus.html

ZIP File

Bootstrap Resources

- Bootstrap Buttons
- Bootstrap Button Groups
- Bootstrap Forms

Other Useful Resources

- The Difference Between Anchors, Inputs and Buttons (Semantic differences in the usage)
- When To Use The Button Element (The multifaceted button element)

From <https://www.coursera.org/learn/bootstrap-4/supplement/ePph0/user-input-additional-resources>

Displaying Content: Additional Resources

PDFs of Presentations

4-Bootstrap-Tables.pdf
PDF File
5-Bootstrap-Cards.pdf
PDF File

Bootstrap Classes

- Bootstrap Tables
- Bootstrap Card
- Bootstrap Blockquote

From <https://www.coursera.org/learn/bootstrap-4/supplement/V6S2V/displaying-content-additional-resources>

PDFs of Presentations

6-Images-Media.pdf
PDF File

Exercise Resources

img
ZIP File
- (download this to conFusion, unzip it to create an "img" folder there)

Bootstrap Resources

- Bootstrap Image Classes
- Bootstrap Media Object Classes

From <https://www.coursera.org/learn/bootstrap-4/supplement/ACjDu/images-and-media-additional-resources>

Alerting Users: Additional Resources

PDFs of Presentations

7-Alerting-Users.pdf
PDF File

Bootstrap Resources

- Bootstrap Badge
- Bootstrap Alerts
- Bootstrap Progress

From <https://www.coursera.org/learn/bootstrap-4/supplement/CBRIz/alerting-users-additional-resources>

Bootstrap Resources

- Navbar
- Breadcrumbs
- Bootstrap Buttons
- Bootstrap Button Groups
- Bootstrap Forms
- Bootstrap Tables
- Bootstrap Card
- Bootstrap Blockquote
- Bootstrap Image Classes
- Bootstrap Media Object Classes
- Bootstrap Responsive Helpers
- Bootstrap Badges
- Bootstrap Alerts
- Bootstrap Progress

From <https://www.coursera.org/learn/bootstrap-4/supplement/J1tO2/assignment-2-resources>

Bootstrap JavaScript Components: Additional Resources

PDFs of Presentations

1-Bootstrap JS.pdf

PDF File

Bootstrap Resources

- Bootstrap and JavaScript
- Bootstrap JS Data Attributes
- Bootstrap Programmatic API

From <https://www.coursera.org/learn/bootstrap-4/supplement/nQrDn/bootstrap-javascript-components-additional-resources>

Tabs and Tabbed Navigation: Additional Resources

PDFs of Presentations

2-Tabs-Pills-Navigation.pdf

PDF File

Bootstrap Resources

- Bootstrap Navs
- Bootstrap Tabs
- Bootstrap Pills
- Bootstrap Tabs Javascript Behavior

From <https://www.coursera.org/learn/bootstrap-4/supplement/vxnqX/tabs-and-tabbed-navigation-additional-resources>

Hide and Seek: Additional Resources

PDFs of Presentations

3-Collapse.pdf

PDF File

Bootstrap Resources

- Bootstrap Collapse
- Bootstrap Accordion Example

From <https://www.coursera.org/learn/bootstrap-4/supplement/GzzMf/hide-and-seek-additional-resources>

Revealing Content: Additional Resources

PDFs of Presentations

4-Tooltips-Popovers-Modals.pdf

PDF File

Bootstrap Resources

- Bootstrap Tooltips
- Bootstrap Popovers
- Bootstrap Modals

From <https://www.coursera.org/learn/bootstrap-4/supplement/SRDxs/revealing-content-additional-resources>

Carousel: Additional Resources

PDFs of Presentations

5-Carousel.pdf
PDF File

Bootstrap Resources

- Bootstrap Carousel

From <https://www.coursera.org/learn/bootstrap-4/supplement/u1Oc8/carousel-additional-resources>

Bootstrap and JQuery: Additional Resources

PDFs of Presentations

1-Bootstrap-JQuery.pdf
PDF File

Bootstrap Resources

- Bootstrap Carousel Methods

JQuery

- JQuery
- W3Schools JQuery

From <https://www.coursera.org/learn/bootstrap-4/supplement/6Ogtq/bootstrap-and-jquery-additional-resources>

CSS Preprocessors: Additional Resources

PDFs of Presentations

2-CSS-Preprocessors.pdf
PDF File

Less and Sass Resources

- Less Getting Started
- Sass Basics
- Getting Started with Less Tutorial
- Getting Started with Sass Tutorial
- Less NPM package
- Node-sass NPM package

From <https://www.coursera.org/learn/bootstrap-4/supplement/BTUdi/css-preprocessors-additional-resources>

Bootstrap Documentation

- Modals
- Modal Methods

From <https://www.coursera.org/learn/bootstrap-4/supplement/773Fm/assignment-4-additional-resources>

Building and Deployment: NPM Scripts: Additional Resources

PDFs of Presentations

3-Building-Deployment.pdf

PDF File

4-NPM-Scripts.pdf

PDF File

Additional Resources

- Why npm Scripts?
- How to Use npm as a Build Tool
- The Command Line for Web Design

NPM Modules

- onchange
- parallelshell
- rimraf
- copyfiles
- imagemin-cli
- usemin-cli
- cssmin
- uglifyjs
- htmlmin

From <https://www.coursera.org/learn/bootstrap-4/supplement/YF9i0/building-and-deployment-npm-scripts-additional-resources>

Building and Deployment: Task Runners: Additional Resources

PDFs of Presentations

5-Task-Runners.pdf

PDF File

Grunt Resources

- Grunt
- Writing an Awesome Build Script with Grunt
- Clean Grunt
- File Globbing
- The Command Line for Web Design: Automation With Grunt

Grunt Plugins

- grunt-contrib-jshint
- jshint-stylish
- grunt-contrib-copy
- grunt-contrib-clean
- grunt-usemin
- grunt-contrib-concat
- grunt-contrib-cssmin
- grunt-contrib-htmlmin
- grunt-contrib-uglify
- grunt-filerev

Gulp Resources

- Gulp
- An Introduction to Gulp.js
- Getting started with gulp
- Building with Gulp
- The Command Line for Web Design: Automation with Gulp

Gulp Plugins

- gulp
- gulp-sass
- browser-sync
- del
- gulp-imagemin
- gulp-uglify
- gulp-usemin
- gulp-rev
- gulp-clean-css
- gulp-flatmap
- gulp-htmlmin

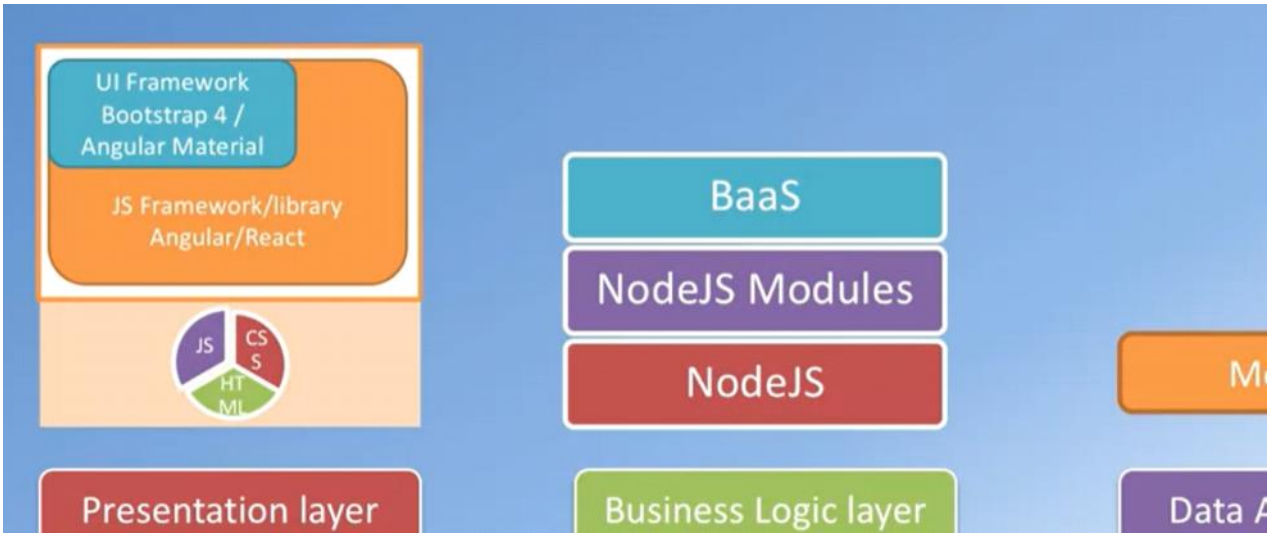
Tasks

- Minification
- UglifyJS
- JSHint

General Resources

- Node, Grunt, Bower and Yeoman - A Modern web dev's Toolkit
- The Advantages of Using Task Runners
- Gulp vs Grunt. Why one? Why the Other?
- Why we should stop using Grunt & Gulp
- Why I Left Gulp and Grunt for npm Scripts

From <https://www.coursera.org/learn/bootstrap-4/supplement/j1vxe/building-and-deployment-task-runners-additional-resources>



Front-End Web UI Frameworks and Tools: Bootstrap 4: Conclusions

Front-End Web UI Frameworks: Bootstrap 4: Conclusions

PDFs of Presentations

7-Conclusion.pdf
PDF File

From <https://www.coursera.org/learn/bootstrap-4/supplement/Of0kz/front-end-web-ui-frameworks-and-tools-bootstrap-4-conclusions>

Honors

Tuesday, September 13, 2022 6:03 PM

Ideation: Objectives and Outcomes

The first step in your journey towards the implementation of the Capstone project begins with an idea. In this module you will develop the idea for your project, the set of expected features, survey the market to look at similar ideas to enable you to differentiate your project from others, while at the same time drawing inspiration from them. You are required to submit a formal ideation report following the structure given in the template. This will enable your peers to provide you feedback and suggestions for your project.

Before you get started on a project, the first step is to develop the idea for the project. In this module you will explore how you develop your idea and come up with possible set of features for your project. At the end of this step you should be able to:

- Clearly express the central idea of your project, and identify the problem being addressed
- Delineate a set of features that you expect your website and app should support
- Identify other projects that might have similar features and would act as exemplars for your project

From <https://www.coursera.org/learn/bootstrap-4/supplement/M69PD/ideation-objectives-and-outcomes>

Ideation Report Template

Project Title

1. Introduction

- A brief introduction to your website idea. State the goals of the project.
- The values / benefits (tangible and intangible) this application can bring to a company/organization/end-user.

2. Expected List of Features

- A brief list of features that you expect your website to support.
- Brief justifications for including these features.

3. Market Survey

- Do a survey of the Web to find about five web sites that might have similar ideas as yours.
- Briefly compare the features of these applications with your application idea.

4. References

- Give references to any material / websites / books etc. relevant to your application idea
- Give the links to the websites relevant to your idea, that you listed in the section above.

From <https://www.coursera.org/learn/bootstrap-4/supplement/X2N0y/ideation-report-template>

Ideation: Additional Resources

General Resources

- Ideation (creative process)

Volunteer your Services

- VolunteerMatch.org
- Free Code Camp

From <https://www.coursera.org/learn/bootstrap-4/supplement/P4Luy/ideation-additional-resources>

UI Design and Prototyping: Objectives and Outcomes

Now that you are more clear about your project idea, it's time to conceive how your project is going to look like for the end-users. This is the time to design the user interface and the flow of your application. User interface design and prototyping helps you to conceptualize the look and feel of your application. This can be achieved in two ways: using wireframe diagrams, or using prototyping tools. We provide links to wireframing and prototyping tools in the additional resources. The focus in this lesson is to be able to visually represent various UI elements to enable designing your application. The aim is to deliver an reasonable representation of the end-user experience with your application At the end of this lesson, you should be able to:

- Construct a wireframe diagram to visually represent the structure of your user interface
- Construct a prototype to enable understanding the flow of your application

From <https://www.coursera.org/learn/bootstrap-4/supplement/2Zoox/ui-design-and-prototyping-objectives-and-outcomes>

UI Design and Prototyping Report Template

Project Title

1. Introduction

- Give a brief introduction to your project and the list of features. Summarize in a few sentences what you proposed in the ideation report.

2. User Interface Design and Prototype

- Give some sample user interface layouts for your application. You can use either wireframe diagrams

or prototyping tools to construct the mock representations of your UI design
- Briefly explain the rationale behind designing your UI and how it is geared towards supporting the list of features for your application.

3. Navigation Structure

- Give a brief overview of the navigation structure for your application.
- Briefly indicate a typical flow of your application in terms of user experience. You can use any way of representing the flow. You can also construct a prototype using one of the prototyping tools to illustrate this.

4. References

- Provide any references relevant to the report.

From <https://www.coursera.org/learn/bootstrap-4/supplement/ughzX/ui-design-and-prototyping-report-template>

UI Design and Prototyping: Additional Resources

Wireframing, Mockups and UI Design

- Wireframe.cc
- Moqups.com
- Axure
- proto.io
- framerjs.com
- The 20 best wireframe tools
- Web Design Inspirations
- Adobe Experience Design
- Free Bootstrap Wireframing Set for PowerPoint

UI Templates

- Bootstrap Expo
- Ionic Showcase

Information Architecture

- A visual vocabulary for describing information architecture and interaction design
- The Elements of User Experience
- The Elements of User Experience: User-Centered Design for the Web and Beyond (2nd Edition) (Voices

From <https://www.coursera.org/learn/bootstrap-4/supplement/UhqjV/ui-design-and-prototyping-additional-resources>

Project Implementation: Objectives and Outcomes

This is the final stretch before you complete your project. By this time you should already have a good scaffolding for your website. As you flesh out your project and race towards completion, it is time to pause and take stock of the current situation. Perhaps a critical look back at the past, with the view of learning from our experience and consolidating this learning into an effective, organized and repeatable process is in order. Upon completion of the project it is important not only to demonstrate the working project, but also summarize the process of reaching the final goal. At the end of this lesson, you will be able to:

- Document the process of starting from an idea and reaching the conclusion of the project, not just the implementation, but also recognizing the process of reaching the end.
- Learn lessons from the process in understanding what worked and what did not, and being able to make intelligent choices in the future based on the experience
- Understand the design and development process through the practice.

From <https://www.coursera.org/learn/bootstrap-4/supplement/Yy5il/project-implementation-objectives-and-outcomes>

Final Report Template

Project Title

1. Introduction

- Briefly describe the salient features of your project.

2. Design and Implementation

Give a detailed system description and design and implementation details.

In particular, this section should contain:

- Details of how you converted from design to the actual realization of your project in terms of implementing the code.
- Any choices that you made, and any modifications that you made to the design, in response to difficulties that you might have encountered while implementing the project.
- A brief discussion of various modules and libraries that you used in implementing your project. In particular highlight the reasons for your choices briefly.
- Include a few screen shots of your website in the report

3. Conclusions

- Briefly state what results you obtained from your project.
- Discuss any features and shortcomings of the project.
- Discuss any choices that you might have made differently, in hindsight after completing the project.

4. References

- Give references to any material / websites / books etc. relevant to your project.

From <https://www.coursera.org/learn/bootstrap-4/supplement/t1YtJ/final-report-template>