Expliquer ce code et à quoi ça sert :

```
<!doctype html> <html lang="en"> <head> <meta charset="utf-8"> <meta http-equiv="X-
UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-
width, initial-scale=1"> <title>MD Jubayer Ahmmed Riyad</title> <link rel="stylesheet"
href="css/bootstrap.min.css"> <link rel="stylesheet" href="css/styles.css"> <link
href='https://fonts.googleapis.com/css?family=Oxygen:400,300,700' rel='stylesheet'
type='text/css'> <link href='https://fonts.googleapis.com/css?family=Lora' rel='stylesheet'
type='text/css'> </head> <body> <header> <nav id="header-nav" class="navbar navbar-
default"> <div class="container"> <div class="navbar-header"> <a href="index.html"
class="pull-left visible-md visible-lg"> <div id="logo-img" alt="Logo image"></div> </a>
<div class="navbar-brand"> <a href="index.html"><h1>MD Jubayer Ahmmed
Riyad</h1></a> <p> <img src="images/star-k-logo.png" alt="Kosher certification">
<span>Kosher Certified</span> </p> </div> <button id="navbarToggle" type="button"
class="navbar-toggle collapsed" data-toggle="collapse" data-target="#collapsable-nav" aria-
expanded="false"> <span class="sr-only">Toggle navigation</span> <span class="icon-
bar"></span> <span class="icon-bar"></span> <span class="icon-bar"></span> </button>
</div> <div id="collapsable-nav" class="collapse navbar-collapse"> <ul id="nav-list"
class="nav navbar-nav navbar-right"> <li id="navHomeButton" class="visible-xs active"> <a
href="index.html"> <span class="glyphicon glyphicon-home"></span> Home</a> </li> <li
id="navMenuButton"> <a href="#" onclick="$dc.loadMenuCategories();"> <span
class="glyphicon glyphicon-cutlery"></span><br class="hidden-xs"> Menu</a> </li> <li> <a
href="#"> <span class="glyphicon glyphicon-info-sign"></span><br class="hidden-xs">
About</a> </li> <li> <a href="#"> <span class="glyphicon glyphicon-
certificate"></span><br class="hidden-xs"> Awards</a> </li> <li id="phone" class="hidden-
xs"> <a href="tel:017-4074-5325"> <span>017-4074-5325</span></a><div>* We
Deliver</div> </li> </ul><!-- #nav-list --> </div><!-- .collapse .navbar-collapse -->
</div><!-- .container --> </nav><!-- #header-nav --> </header> <div id="call-btn"
class="visible-xs"> <a class="btn" href="tel:017-4074-5325"> <span class="glyphicon
glyphicon-earphone"></span> 017-4074-5325 </a> </div> <div id="xs-deliver" class="text-
center visible-xs">* We Deliver</div> <div id="main-content" class="container"></div>
<footer class="panel-footer"> <div class="container"> <div class="row"> <section
id="hours" class="col-sm-4"> <span>Hours:</span><br> Sun-Thurs: 11:15am -
10:00pm<br> Fri: 11:15am - 2:30pm<br> Saturday Closed <hr class="visible-xs">
</section> <section id="address" class="col-sm-4"> <span>Address:</span><br> Dogair
West para<br> Sarulia, Demra 1361 <p>* Delivery area within 3-4 miles, with minimum
order of $20 plus $3 charge for all deliveries.</p> <hr class="visible-xs"> </section> <section
id="testimonials" class="col-sm-4"> <p>"The best Chinese restaurant I've been to! And that's
saying a lot, since I've been to many!"</p> <p>"Amazing food! Great service! Couldn't ask
for more! I'll be back again and again!"</p> </section> </div> <div class="text-
center">&copy; Copyright MD Jubayer Ahmmed Riyad 2021</div> </div> </footer> <!--
jQuery (Bootstrap JS plugins depend on it) --> <script src="js/jquery-2.1.4.min.js"></script>
<script src="js/bootstrap.min.js"></script> <script src="js/ajax-utils.js"></script> <script
src="js/script.js"></script> </body> </html>
```

Ceci est un code HTML pour une page web. Il comporte une en-tête, un menu de navigation, un corps de contenu et un pied de page.

L'en-tête définit la barre de navigation supérieure qui comporte un logo, un nom de marque, un bouton de menu déroulant pour les catégories de menu et des liens vers différentes sections du site web.

Le corps de contenu est un conteneur où le contenu principal de la page sera affiché.

Le pied de page comporte des informations sur les heures d'ouverture, l'adresse et d'autres informations de contact.

Il y a également des feuilles de style associées qui sont chargées à partir des fichiers CSS pour donner un aspect plus agréable et cohérent à la page. Les styles sont définis dans les fichiers `bootstrap.min.css` et `styles.css`.

En gros, ce code HTML constitue la structure de base d'une page web et peut être personnalisé pour inclure du contenu spécifique pour le site en question.

et celui la { "name": "project-html-portfolio", "version": "1.0.0", "description": "", "private": true, "scripts": { "start": "browser-sync start --server ./src --files ./src", "test": "mocha --compilers js:babel-register test/*.spec.js", "deploy:github-pages": "git subtree push --prefix src origin gh-pages" }, "author": "Sergio Cruz <sergio@codeschool.com>", "license": "MIT", "devDependencies": { "babel-preset-es2015": "^6.18.0", "babel-register": "^6.18.0", "browser-sync": "^2.14.0", "chai": "^3.5.0", "jsdom": "^9.4.1", "mocha": "^3.0.1" }, "engines": { "node": ">=4.6", "npm": ">=2.15" }, "babel": { "presets": ["es2015"] } }

The file you shared is a `package.json` file for a project called "project-html-portfolio". This file is used to manage the dependencies and scripts for a Node.js project.

The `scripts` section lists the commands that can be run for this project using `npm run`. For example, running `npm run start` will start the development server using the `browser-sync` tool. Running `npm run test` will run the test suite using `mocha` and `chai`. Finally, `npm run deploy:github-pages` will push the contents of the `src` directory to the `gh-pages` branch on GitHub, effectively deploying the project to GitHub Pages.

The `devDependencies` section lists the tools and libraries required for development and testing, such as `mocha`, `chai`, `browser-sync`, and others. These dependencies will only be installed when you run `npm install` in the project directory.

The `engines` section lists the required versions of Node.js and npm. The project requires a version of Node.js greater than or equal to 4.6, and a version of npm greater than or equal to 2.15.

The `babel` section specifies the configuration for Babel, which is a JavaScript compiler. The `es2015` preset is specified, meaning that the project uses ECMAScript 2015 (ES6) syntax, which will be transpiled to ES5 syntax for older browsers.

et celui la # Coursera-HTML-CSS-and-Javascript-for-Web-Developers This repository contains all of the source code used in the course called HTML, CSS and Javascript for Web Developers in Coursera. # Assignments : * Module-1 - Quiz * Module-2 - Coding * Module-3 - Coding * Module-4 - Coding * Module-5 - Coding # Outputs : * [Module-2](https://jubayerriyad.github.io/Coursera-HTML-CSS-and-JavaScript-for-Web-Developers/Assignments/module-2/index.html) * [Module-3](https://jubayerriyad.github.io/Coursera-HTML-CSS-and-JavaScript-for-Web-Developers/Assignments/module-3/index.html) * [Module-4](https://jubayerriyad.github.io/Coursera-HTML-CSS-and-JavaScript-for-Web-Developers/Assignments/module-4/index.html) * [Module-5](https://jubayerriyad.github.io/Coursera-HTML-CSS-and-JavaScript-for-Web-Developers/Assignments/module-5/index.html)

This repository contains the source code used in the course HTML, CSS and Javascript for Web Developers on Coursera. The course is divided into 5 modules, each including a quiz and coding assignments. The coding assignments for the modules 2, 3, 4, and 5 are available to view at the URLs provided in the Outputs section of the repository.

et celui la : // ******************************* // START HERE IF YOU WANT AN EASIER STARTING POINT FOR THIS ASSIGNMENT // ******************************* // // Module 4 Assignment Instructions. // // The idea of this assignment is to take an existing array of names // and then output either Hello 'Name' or Good Bye 'Name' to the console. // The program should say "Hello" to any name except names that start with a "J" // or "j", otherwise, the program should say "Good Bye". So, the final output // on the console should look like this: /* Hello Yaakov Good Bye John Good Bye Jen Good Bye Jason Hello Paul Hello Frank Hello Larry Hello Paula Hello Laura Good Bye Jim WARNING!!! WARNING!!! The code does NOT currently work! It is YOUR job to make it work as described in the requirements and the steps in order to complete this assignment. WARNING!!! WARNING!!! */ (function () { var names = ["Yaakov", "John", "Jen", "Jason", "Paul", "Frank", "Larry", "Paula", "Laura", "Jim"]; for (var i = 0; i < names.length; i++) { var firstLetter = names[i].charAt(0).toLowerCase(); if (firstLetter === 'j') { byeSpeaker(names[i]); } else { helloSpeaker(names[i]); } } })();

This is a JavaScript code for a program that outputs either "Hello 'Name'" or "Good Bye 'Name'" to the console based on the first letter of the name. The names are stored in an array called "names".

The program uses a for loop to iterate through the names in the array. For each name, it gets the first letter of the name, converts it to lowercase, and checks whether it is equal to 'j'. If the first letter is 'j', the program calls the "byeSpeaker()" function and passes the name as an argument. If the first letter is not 'j', the program calls the "helloSpeaker()" function and passes the name as an argument.

This code does not work as is, as the "byeSpeaker()" and "helloSpeaker()" functions are not defined in the code. It is the job of the reader to make it work according to the requirements and steps specified in the assignment.

et celui la : // NOTE! The steps in this file are basically identical to the ones you // performed in the SpeakHello.js file. // STEP 6: Wrap the entire contents of SpeakGoodBye.js inside of an IIFE // See Lecture 52, part 2 // STEP 7: Create an object, called 'byeSpeaker' to which you will attach // the "speak" method and which you will expose to the global context // See Lecture 52, part 1 // var byeSpeaker = // DO NOT attach the speakWord variable to the 'byeSpeaker' object. // STEP 8: Rewrite the 'speak' function such that it is attached to the // byeSpeaker object instead of being a standalone function. // See Lecture 52, part 2 // STEP 9: Expose the 'byeSpeaker' object to the global scope. Name it // 'byeSpeaker' on the global scope as well. // xxxx.xxxx = byeSpeaker; (function(window) { var speakWord = "Good Bye"; var byeSpeaker = function (name) { console.log(speakWord + " " + name); } window.byeSpeaker = byeSpeaker; })(window);

This code is defining an IIFE (Immediately Invoked Function Expression) in JavaScript. The purpose of the IIFE is to create a new scope and protect the variables defined inside it from being accessed globally. The `speakWord` variable is assigned a string value of "Good Bye". The `byeSpeaker` function takes a `name` parameter, and logs the string "Good Bye" concatenated with the `name` parameter to the console. The function is then attached to the global `window` object, so that it can be accessed from other scripts. This is useful for code organization and modularization, as you can define functions that you want to be reusable in other parts of your code in separate files, and then expose them to the global scope by attaching them to the `window` object.

et ceui ci : // STEP 2: Wrap the entire contents of SpeakHello.js inside of an IIFE // See Lecture 52, part 2 // STEP 3: Create an object, called 'helloSpeaker' to which you will attach // the "speak" method and which you will expose to the global context // See Lecture 52, part 1 // var helloSpeaker = // DO NOT attach the speakWord variable to the 'helloSpeaker' object. // STEP 4: Rewrite the 'speak' function such that it is attached to the // helloSpeaker object instead of being a standalone function. // See Lecture 52, part 2 // STEP 5: Expose the 'helloSpeaker' object to the global scope. Name it // 'helloSpeaker' on the global scope as well. // See Lecture 52, part 2 // (Note, Step 6 will be done in the SpeakGoodBye.js file.) // xxxx.xxxx = helloSpeaker; (function(window) { var speakWord = "Hello"; var helloSpeaker = function (name) { console.log(speakWord + " " + name); } window.helloSpeaker = helloSpeaker; })(window);

Ces trois morceaux de code sont des exemples de ce qu'on appelle une "IIFE" (Immediately Invoked Function Expression). L'objectif est d'empaqueter le code pour le rendre "privé" et ne pas polluer l'espace global. Les deux premiers morceaux de code montrent comment le code a été encapsulé dans une fonction qui est immédiatement invoquée. Lors de l'invocation, une fonction anonyme est créée et exécutée immédiatement, et la portée de ses variables est limitée à l'intérieur de cette fonction. Le troisième morceau de code montre comment la fonction utilise "window" comme argument pour exposer certaines de ses variables à l'espace global.

et celui la : .container-fluid{ margin: 0; padding: 0; } .navbar{ border-radius: 0px; background-color: rgba(0,0,0,0.8); } .navbar-brand{ font-size: 35px; color: white; padding-left: 25px; } .navbar-brand:hover{ color: white; } .nav{ margin: 0; padding: 0; } .navbar-nav{ font-size: 25px; text-align: center; margin: 0; padding: 0; } .navbar-toggle{ border: 2px solid white; margin-right: 25px; } .icon-bar{ background-color: white; } .main-title{ margin-bottom: 15px; text-align: center; color: #ff4532; font-size: 50px; font-family: "Comic Sans MS", cursive, sans-serif; font-weight: bold; } li{ box-sizing: border-box; width: 100%; } .menu-item{ padding: 0; margin: 0; width: 100%; background-color: white; border-bottom: 1px solid black; color: black; display: block; } .menu-item:hover{ background-color: rgba(0,0,0,0.9); color: rgba(0,0,0,0.4); } .row{ margin: 20px; } .content-box{ margin: 20px; width: auto; height: auto; color: black; background-color: rgba(0,0,0,0.2); font-family: "Comic Sans MS", cursive, sans-serif; padding: 20px; } .item-name{ text-align: center; font-size: 20px; font-weight: bold; } #top{ }

The above code is a CSS stylesheet for a website. It sets styles for elements such as the container-fluid, navbar, navbar-brand, navbar-nav, navbar-toggle, icon-bar, main-title, li, menu-item, row, content-box, and item-name. The styles include things such as font size, font family, text alignment, background color, color, border-radius, border, padding, margin, and hover effects. The stylesheet uses CSS selectors to apply styles to specific HTML elements, such as applying the "navbar" style to elements with a class of "navbar".