

DESIGN DAN PEMROGRAMAN WEB – WEEK 6 (JQUERY & BOOTSTRAP)

Name : Khen Muhammad Cahyo

NIM : 244107023002

Class : TI 2I

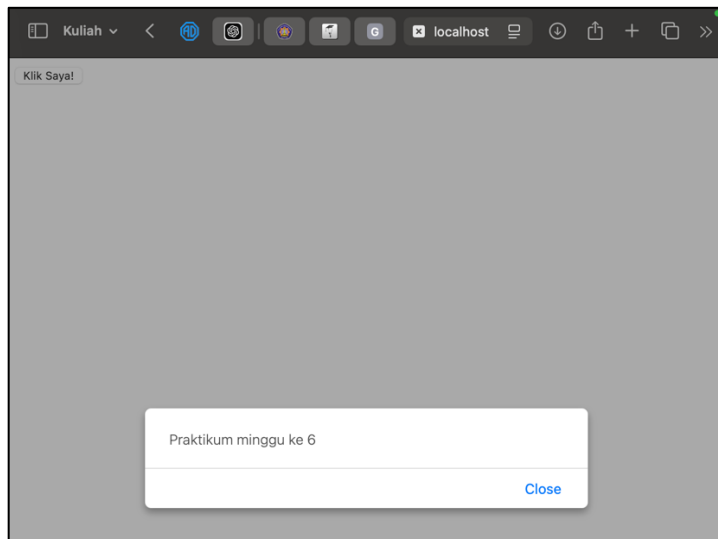
Presence Number : 14

Here are my Github link for all codes in this jobsheet.

<https://github.com/KhenCahyo13/MyCampus-Jobsheet/tree/main/DDPW/week-6>

Question 1

Here are the results:

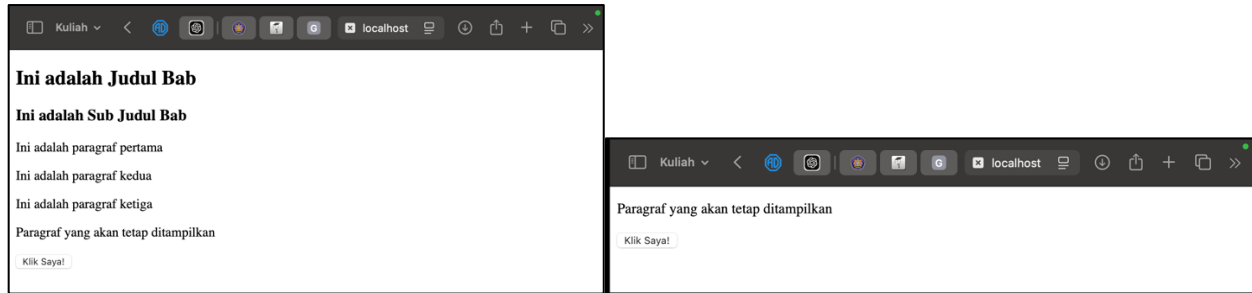


Here are my observations:

The code uses jQuery to create an interactive button on a webpage. It begins by including the jQuery library, specifically version 3.7.1, and defines a script that waits for the DOM to be fully loaded using `$(document).ready()`. Inside this function, an event listener is attached to a button with the ID `button1`, such that when the button is clicked, an alert box will display the message "Praktikum minggu ke 6". The button is created using an `<input>` element with the type set to "button", and its value (label) is "Klik Saya!".

Question 2

Here are the results:



Here are my observations:

The code, utilizing jQuery, hides various elements on the webpage when a button is clicked. After including the jQuery 3.7.1 library, it defines a script where, upon the document being fully loaded (`$(document).ready()`), a click event listener is attached to the `<button>` element. Once the button is clicked, it hides several elements: the `<h2>` header, any elements with the class `.subjudul`, the first paragraph with the ID `#paragraf`, the second paragraph that has both the ID `#paragraf` and class `.dua`, and paragraphs inside a `<div>` with the class `.paragraf`. A separate paragraph that doesn't match these selectors remains visible on the page.

Question 3

Here are my explanation:

- `$("button")`: Selects all `<button>` elements.
- `$("h2")`: Selects all `<h2>` elements.
- `$(".subjudul")`: Selects elements with the class `subjudul`.
- `$("#paragraf")`: Selects the element with the ID `paragraf`.
- `$("#paragraf.dua")`: Selects the element with both the ID `paragraf` and class `dua`.
- `$("div p.paragraf")`: Selects `<p>` elements inside a `<div>` with the class `paragraf`.

Question 4

Here are the results:

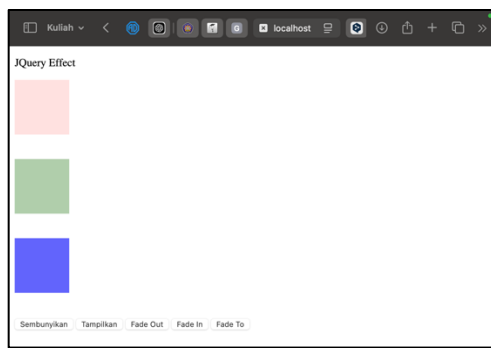


Here are my observations:

The code uses jQuery to apply various interactive styling effects to a paragraph with the ID `paragraf` when certain events occur. When the paragraph is clicked, its text color changes to white. On mouseover, its background changes to silver, and when the mouse leaves, the background changes to blue. If the paragraph is double-clicked, a black border of 3 pixels solid appears around it. All these event listeners are set to run once the document is fully loaded using `$(document).ready()`.

Question 5

Here are the results:

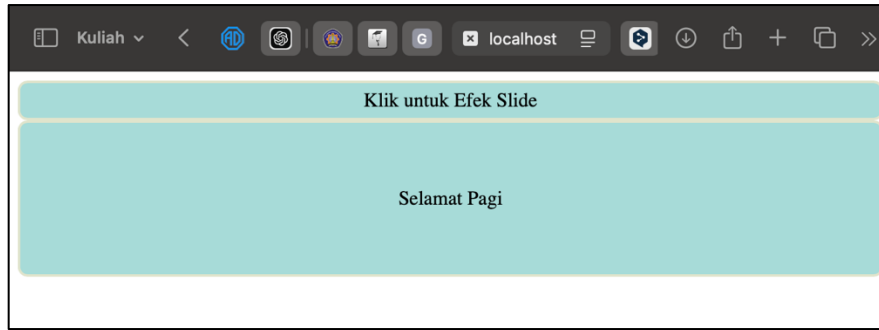


Here are my observations:

The jQuery code adds interactive effects to three `<div>` elements (`#div1`, `#div2`, `#div3`) using buttons. Upon loading, clicking the button with class `tombol1` hides all `<div>` elements, while `tombol2` shows them. The button `tombol3` fades out the `<div>` elements at different speeds: `#div1` instantly, `#div2` slowly, and `#div3` over 3 seconds. The button `tombol4` is defined twice; it first fades in the `<div>` elements and then reduces their opacity to 0.15, 0.4, and 0.7, respectively, with a slow effect.

Question 6

Here are the results:

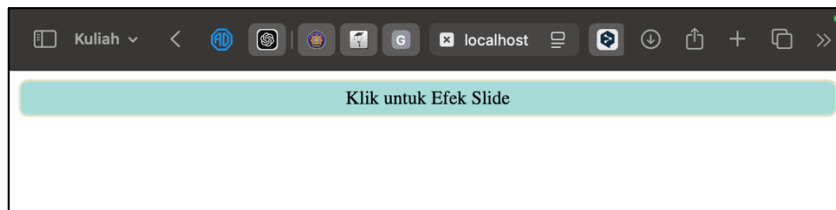


Here are my observations:

The jQuery code creates an interactive effect where clicking on the <div> element with the ID flip triggers a sliding action to hide another <div> with the ID kotak2. When the page is fully loaded, the \$(document).ready() function ensures that the click event is attached to the #flip element, causing #kotak2, which contains the text "Selamat Pagi," to slide up slowly when clicked. This creates a smooth visual transition, enhancing the user experience.

Question 7

Here are the results:

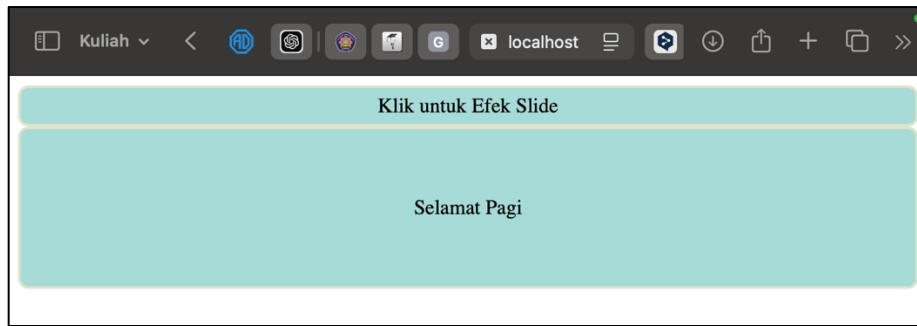


Here are my observations:

The jQuery code implements an interactive feature that reveals a hidden <div> element when a user clicks on another <div> labeled flip. Upon the document's readiness, the \$(document).ready() function attaches a click event handler to the #flip element. When clicked, the #kotak2 element, which contains the text "Selamat Pagi" and is initially set to display: none;, smoothly slides down into view over a slow duration using the slideDown method. This creates a visually appealing effect that enhances user engagement on the webpage.

Question 8

Here are the results:

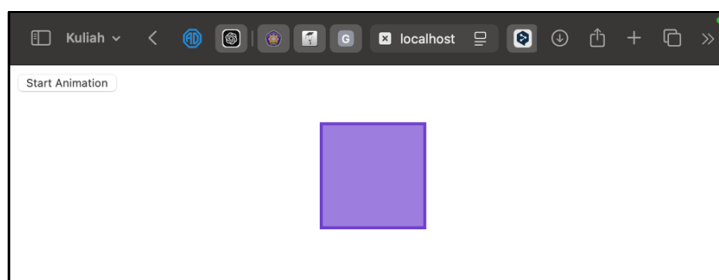


Here are my observations:

The jQuery code adds an interactive slide effect to a webpage where clicking on the <div> element with the ID fliptoggles the visibility of another <div> labeled kotak2. When the document is fully loaded, the \$(document).ready() function binds a click event to the #flip element. Upon clicking, the #kotak2 element, which contains the text "Selamat Pagi," either slides up to hide or slides down to reveal itself, depending on its current state, using the slideToggle method with a slow animation. This creates a smooth, engaging user experience.

Question 9

Here are the results:

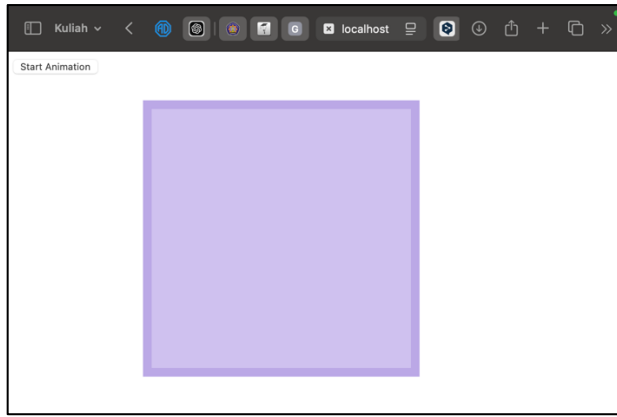


Here are my observations:

The jQuery code enables an animation effect on a <div> element with the class box when a button is clicked. Once the document is fully loaded, the \$(document).ready() function binds a click event to the button. When the button is pressed, the animate method is called on the <div>, moving it horizontally to the right by changing its left CSS property to 300 pixels. This creates a smooth animation, enhancing the visual appeal and interactivity of the webpage.

Question 10

Here are the results:

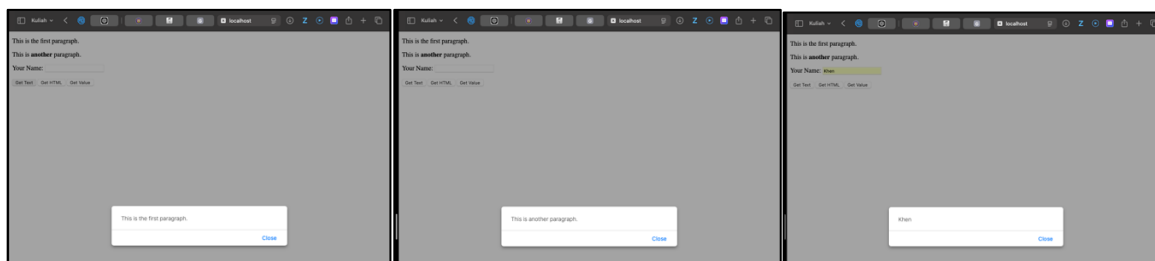


Here are my observations:

The jQuery code creates a series of sequential animations on a `<div>` element with the class `box` when a button is clicked. Upon loading the document, the `$(document).ready()` function attaches a click event handler to the button. When the button is pressed, it triggers a chain of animations: first, the box's width is increased to 300px, followed by an increase in height to 300px, a left margin shift of 150px, a border width adjustment to 10px, and finally, a reduction in opacity to 0.5. Each animation smoothly transitions to the next, providing a dynamic visual effect that enhances user interaction on the webpage.

Question 11

Here are the results:

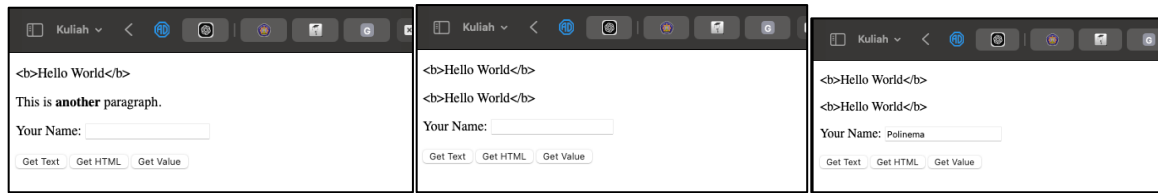


Here are my observations:

The jQuery code retrieves text, HTML, or input values when specific buttons are clicked. On page load, event handlers are set for three buttons: the first button shows an alert with the plain text from `#test1`, the second shows HTML content from `#test2`, and the third displays the value from the input field `#test3`. This creates basic interactivity with the webpage content.

Question 12

Here are my results:

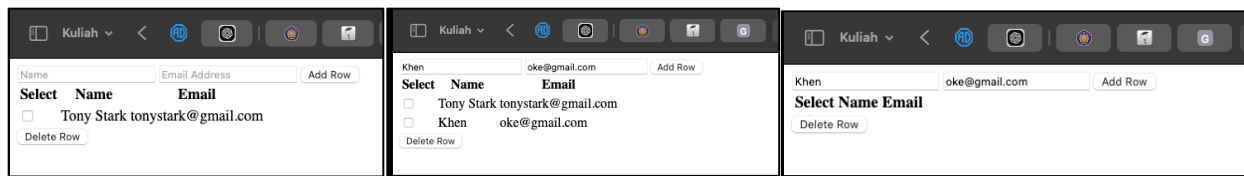


Here are my observations:

The jQuery code updates the content of specified elements when buttons are clicked. Upon loading, click event handlers are set for three buttons: clicking #btn1 changes the text inside the paragraph with the ID test1 to display the string Hello World (as plain text, without formatting). Clicking #btn2 does the same for the paragraph #test2, replacing its content with the same text. Clicking #btn3 changes the value of the input field #test3 to "Polinema." This provides simple dynamic updates to the page content.

Question 13

Here are my results:

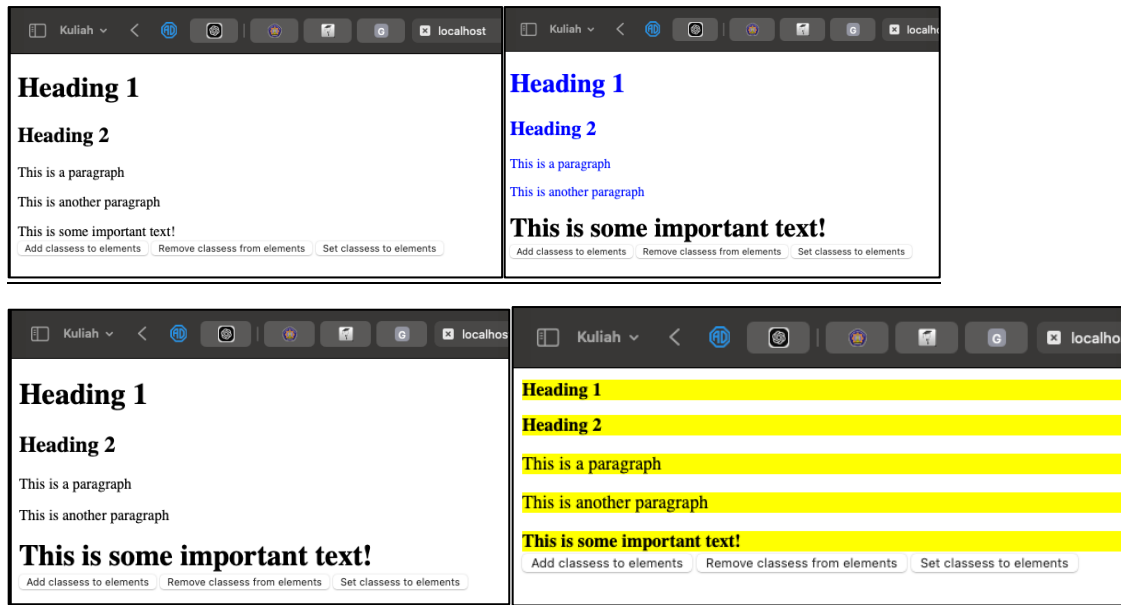


Here are my observations:

The jQuery script dynamically adds and removes rows from an HTML table. When the "Add Row" button is clicked, it retrieves the values entered in the "Name" and "Email" fields and appends a new row with this data to the table, along with a checkbox. The "Delete Row" button removes rows where the checkbox is selected. It checks each checkbox in the table, and if any are marked, it removes the corresponding row. The code provides a basic interface for adding and deleting table rows dynamically.

Question 14

Here are the results:

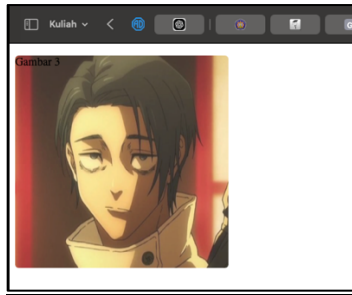


Here are my observations:

The jQuery script manipulates CSS classes and styles on various HTML elements. When the "Add classes to elements" button is clicked, it adds the class `blue` to all `<h1>`, `<h2>`, and `<p>` tags, and the class `important` to the `<div>`. The "Remove classes from elements" button removes the `blue` class from the `<h1>`, `<h2>`, and `<p>` tags. The "Set classes to elements" button applies inline CSS styles to `<h1>`, `<h2>`, `<p>`, and `<div>`, setting the background color to yellow and font size to 100%.

Question 15

Here are the results:

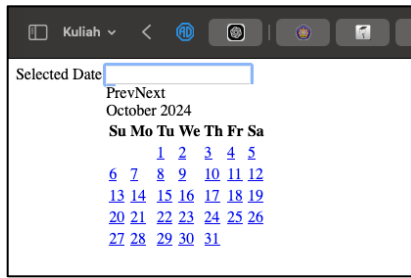


Here are my observations:

The jQuery code creates an image slider where both the images and their corresponding titles are hidden initially and then displayed one by one in a cycle. The `showNextImage()` function increments a counter `i` to show the next image and title by fading them in, displaying them for 1100 milliseconds, and then fading them out. The images and titles are placed in the `#slider` div, and once the counter reaches 3 (the number of images), it resets to 0 to repeat the cycle. The images and titles alternate every 3 seconds using `setInterval()`.

Question 16

Here are the results:

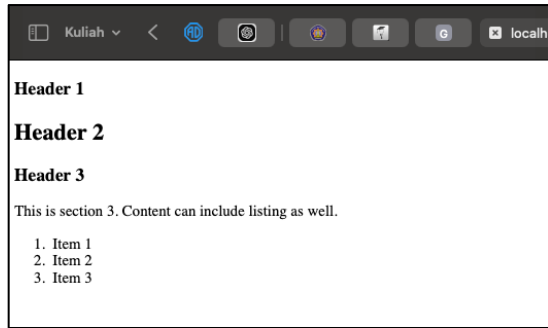


Here are my observations:

The jQuery code initializes a date picker on an input field. When the document is ready, the `$("#date_ex").datepicker()` function is called, which attaches a jQuery UI date picker to the input field with the ID `date_ex`. This allows users to select a date from a pop-up calendar when they click on the input field. The date picker will be displayed within the `<div>` containing the text "Selected Date" and the input box.

Question 17

Here are the results:



Here are my observations:

The jQuery code creates an accordion interface on a div element with the ID JQuery_accordion. When the document is ready, the `$("#JQuery_accordion").accordion()` function is triggered, which converts the specified content into collapsible sections with clickable headers. The accordion contains three headers: "Header 1" followed by a paragraph, "Header 2" followed by text and an image, and "Header 3" followed by a paragraph and an ordered list. Users can click on the headers to expand or collapse the content within each section.

Question 18

Here are the results:

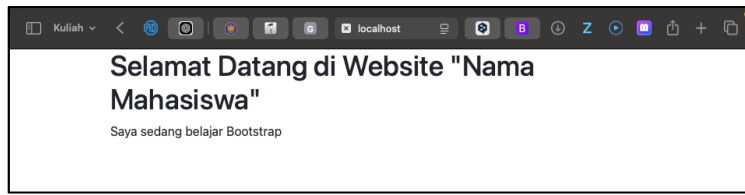


Here are my observations:

The jQuery code dynamically loads external content into a div with the ID box when a button is clicked. The script first ensures the document is fully loaded using `$(document).ready()`. Upon clicking the button, the `$("#box").load("/week-6/test-content.html")` function fetches and injects the content from the specified HTML file (`/week-6/test-content.html`) into the div. Initially, the div displays a message prompting the user to click the button to load the external content.

Question 19

Here are the results:

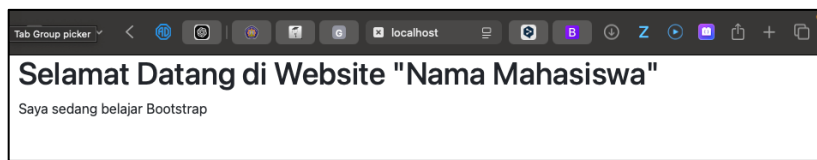


Here are my observations:

The HTML code creates a simple webpage with a container class wrapping a heading (h1) and a paragraph (p). The container class is typically used with Bootstrap for responsive layouts. When the browser is resized to a smaller width, the content inside the container adjusts automatically, remaining centered with margins on both sides, depending on Bootstrap's responsive grid system. Since I cannot resize the browser or take screenshots, you can visualize this by reducing your browser window size to see how the container class adapts to different screen sizes.

Question 20

Here are the results:

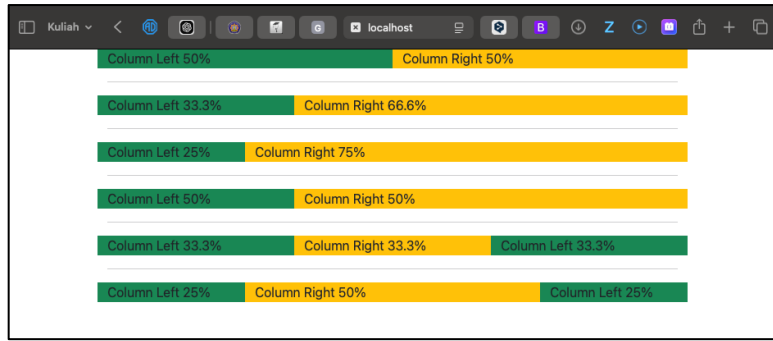


Here my observations:

The code creates a webpage with a container-fluid class that holds a heading and a paragraph, allowing for full-width responsiveness. When resizing the browser to a smaller size, the content remains flush against the edges, in contrast to the standard container class, which would center the content with fixed margins. This difference affects how the content is presented on various screen sizes, with container-fluid offering a more expansive layout option.

Question 21

Here are the results:

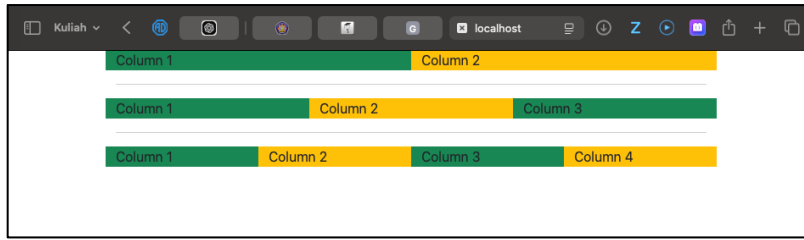


Here my observations:

The provided code utilizes Bootstrap's grid system to create a responsive layout with multiple rows and columns, each displaying different background colors to distinguish between left and right sections. The first row consists of two equal columns (50% width each), while the second row features a three-column layout with a 33.3% and 66.6% distribution. The subsequent rows showcase varying column widths, such as a 25% and 75% split, and an equal distribution among three columns, each occupying 33.3%. The use of the `bg-success` and `bg-warning` classes visually differentiates the columns, enhancing the layout's clarity and usability. Additionally, the `<hr>` elements between the rows provide visual separation, making the overall structure more organized and easier to navigate. This layout adapts responsively to different screen sizes, maintaining its integrity and functionality across devices.

Question 22

Here are the results:

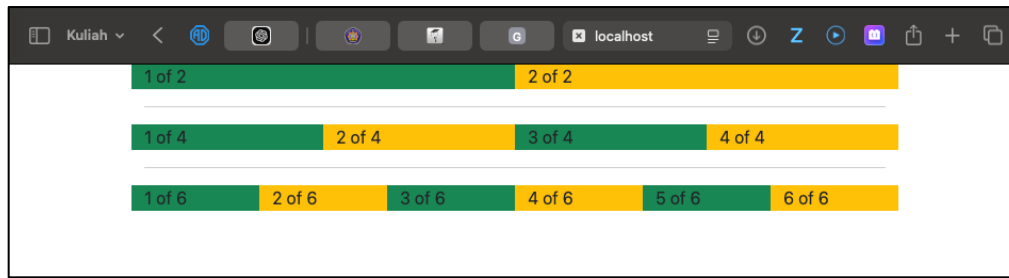


Here are my observations:

The provided code creates a responsive layout using Bootstrap's grid system, featuring three rows, each containing columns of varying numbers and widths. The first row has two equal columns, styled with the classes `bg-success` (green) and `bg-warning` (yellow), representing "Column 1" and "Column 2," respectively. The second row expands to three equal columns, with the first and third columns in green and the middle one in yellow, labeled as "Column 1," "Column 2," and "Column 3." The third row contains four columns, where the first and third are green, and the second and fourth are yellow, representing "Column 1," "Column 2," "Column 3," and "Column 4." The use of `<hr>` tags between the rows provides a visual separation, enhancing the layout's clarity. Overall, this structure showcases Bootstrap's ability to create flexible and responsive grid layouts easily, adapting to various screen sizes while maintaining an organized appearance.

Question 23

Here are the results:

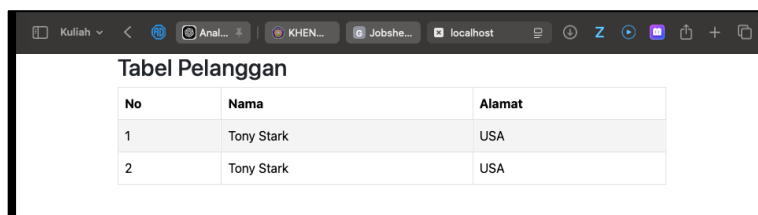


Here are my observations:

The provided code creates a responsive layout using Bootstrap's grid system, consisting of three rows with different column configurations. The first row features two columns labeled "1 of 2" (green) and "2 of 2" (yellow). The second row contains four columns ("1 of 4," "2 of 4," "3 of 4," "4 of 4"), alternating colors for visual contrast. The third row expands to six columns labeled from "1 of 6" to "6 of 6," also alternating colors. Horizontal rules (<hr>) separate the rows, enhancing readability and organization while showcasing Bootstrap's flexibility for adaptive layouts across screen sizes.

Question 24

Here are the results:



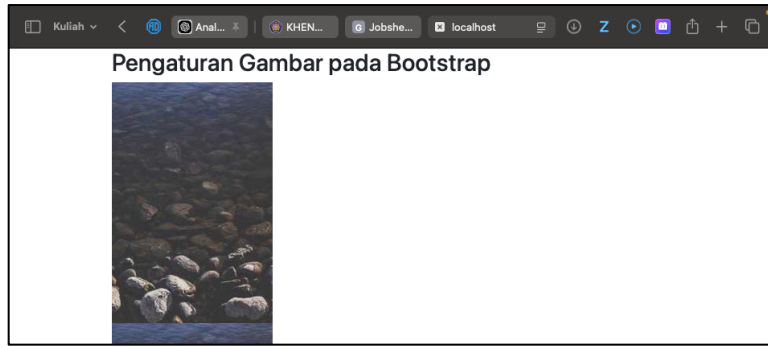
No	Nama	Alamat
1	Tony Stark	USA
2	Tony Stark	USA

Here are my observations:

The code creates a Bootstrap-styled table titled "Tabel Pelanggan," which displays customer information. It includes a header (<thead>) with three columns: "No," "Nama," and "Alamat," followed by a body (<tbody>) containing two rows with customer details, both listing "Tony Stark" from the USA. The table utilizes Bootstrap classes like table, table-bordered, table-striped, and table-hover for enhanced styling and interactivity. There's a minor syntax error in the <thead> due to an extra <tr> tag.

Question 25

Here are the results:

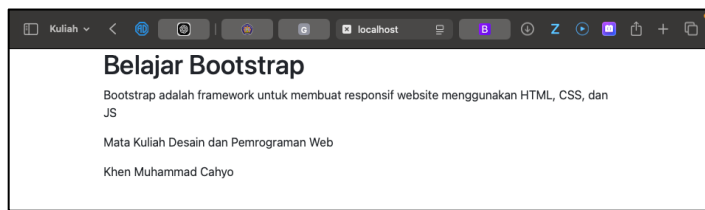


Here are my observations:

The code sets up a Bootstrap container titled "Pengaturan Gambar pada Bootstrap," featuring a responsive image gallery. Inside a row with one column per row (row-cols-1), it displays four images sourced from Picsum, each assigned the class `img-fluid` to ensure they scale appropriately based on the screen size. When the browser window is resized to a smaller width, the images stack vertically, maintaining their original aspect ratio while adapting to the container's width, allowing for a mobile-friendly layout. This responsive behavior enhances usability across various devices.

Question 26

Here are the results:

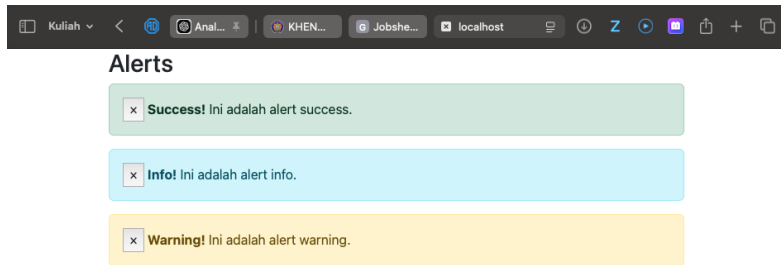


Here are my observations:

The code creates a Bootstrap container featuring a jumbotron that highlights the title "Belajar Bootstrap" and a description of Bootstrap as a framework for responsive web design. Below the jumbotron, two paragraphs provide details about the course, "Mata Kuliah Desain dan Pemrograman Web," and the author's name, "Khen Muhammad Cahyo." The jumbotron enhances visibility and user engagement.

Question 27

Here are the results:

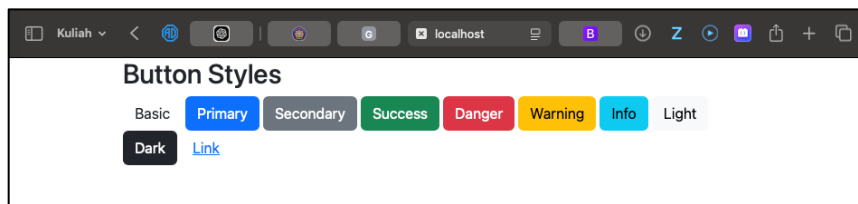


Here are my observations:

The code creates a Bootstrap container that displays a section titled "Alerts," containing three dismissible alert messages. Each alert is styled with different contextual classes: alert-success for a success message, alert-info for informational alerts, and alert-warning for warnings. Each alert includes a close button, allowing users to dismiss them, enhancing user experience by providing feedback on different situations. However, there's a typo in the class name; it should be alert-dismissible instead of alert-dismissible for proper functionality.

Question 28

Here are the results:

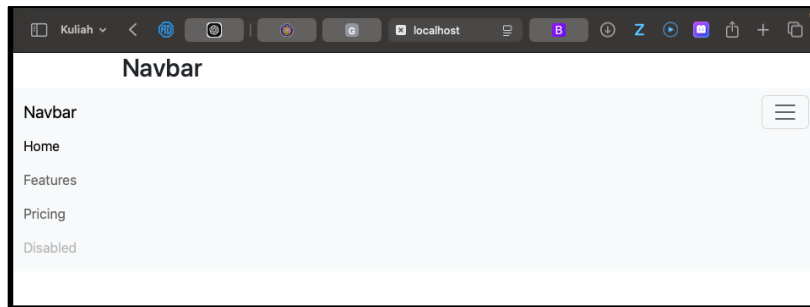


Here are my observations:

The code showcases various button styles using Bootstrap within a container. It features a heading "Button Styles" followed by multiple button elements, each styled with different Bootstrap classes to represent various contextual designs: btn-primary, btn-secondary, btn-success, btn-danger, btn-warning, btn-info, btn-light, btn-dark, and btn-link. The first button uses the basic btn class without any contextual styling. This structure allows for a visually appealing and consistent interface, making it easy for users to understand the purpose of each button through color and style differentiation.

Question 29

Here are the results:

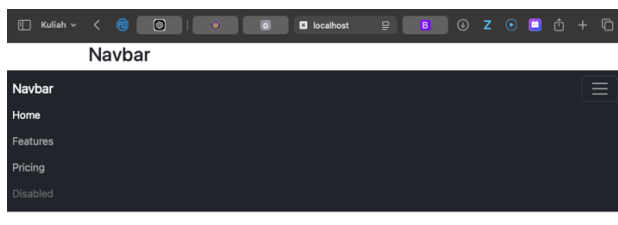


Here are my observations:

The code creates a responsive navigation bar using Bootstrap. It starts with a container displaying the heading "Navbar" and then defines the navigation bar with the navbar class and a background of bg-body-tertiary. Inside the navigation bar, there is a brand link labeled "Navbar," and a button for toggling the navbar on smaller screens, indicated by the navbar-toggler class. The collapsible section contains a list of navigation items, including "Home," "Features," "Pricing," and a disabled item. The use of Bootstrap's classes ensures that the navbar adapts to different screen sizes while providing an accessible and user-friendly navigation experience.

Question 30

Here are the results:



Here are my observations:

In the code, the navigation bar is enhanced with a dark theme by using the classes bg-dark and navbar-dark from Bootstrap. The bg-dark class applies a dark background color to the navigation bar, while the navbar-dark class changes the text color to a lighter shade, ensuring good contrast and readability against the dark background. This setup includes a responsive design that collapses into a toggler button on smaller screens, maintaining a clean and modern appearance. The navbar contains navigation items such as "Home," "Features," "Pricing," and a disabled link, providing users with a clear and accessible menu that adapts seamlessly to various screen sizes.