|  |  |
| --- | --- |
| **Lab Exercise 1:** | **Slide 28** |

**Instructions**: There is no right or wrong specifically for this exercise, but follow the design procedure: a) define requirements, b) paper sketch, c) lo-fi, d) hi-fi. Use Figma for the lo-fi and hi-fi designs, and don’t make it overly complicated.

**Goal**: To become familiar with using the Figma design tool to create lo-fi and hi-fi designs.

**Submission**: Make the Figma link public and viewable and just create a file (called figma, exercise1 or similar) that contains a link to your Figma design. Commit this file into your repository for Module 4.

**Extension**: Try following the instructions in this design exercise: <https://www.figma.com/community/file/1128352595404858022>

**Resources & Extra Learning**:

* <https://help.figma.com/hc/en-us/sections/4405269443991-Figma-for-beginners-4-parts>
* <https://www.figma.com/resource-library/design-basics/>

|  |  |
| --- | --- |
| **Lab Exercise 2:** | **Slide 36** |

**Instructions**: Make sure you do all three parts of the lab from the slide!

1. Ensure the addCard function can have content passed to the card dynamically via function parameters.
2. Modify the code to call addCard repeatedly (in a loop), so the cards are automatically generated based on data from an array. This ensures that you can make as many cards as needed to display all the data in the array. Use the code provided on the right of the slide as the data.
3. Create a second HTML page using the same technique but a different template, and populate your web page dynamically, using the provided code on the slide of Van Gogh and his portfolio.

**Goal**: To learn how to separate the data (model) from the presentation (view) using JS to dynamically populate HTML templates.

**Submission**: Create a new folder for **Exercise 2** in your **Module 4** folder/repo. Ideally create separate, logically named HTML, CSS and JS files containing your answers to all 3 parts of the lab, including comments.

**Extension**: Do the suggested extension on the slide for part 3, to include an array of artists all with multiple artworks in their portfolio. Improve the CSS to display all of the card templates more attractively, and consider extending the artist data to include extra information such as age, location, bio.

**Resources & Extra Learning**:

* <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>
* <https://www.w3schools.com/js/js_htmldom.asp>
* <https://www.johnpapa.net/render-html-2/>

|  |  |
| --- | --- |
| **Lab Exercise 3:** | **Slide 39** |

**Instructions**: Follow the instructions on the slide to create a basic calculator supporting the 4 main mathematical operations, making sure to follow the software development process on the right of the slide. Make sure to do your requirements definition and design first, and include these in your submission.

*Tip: Think about which buttons you will need and how these will be arranged on the page. Don’t overcomplicate it, and don’t use the eval function.*

If in your research and tackling this project you come across calculation libraries - don’t utilise this to do the leg work of your calculation for you. There are security issues and you don’t want to ‘cheat’ and let something do the logical parts of your application for you.

**Goal**: To practise following core aspects of the software development lifecycle process, from definition to design to development.

**Submission**: Create a separate Github repository just for this exercise. Clone it locally and try to split your development process into features which can be committed into branches, then merged into the main repository.

**Extension**: Try to make your calculator look and work more like a real one by improving the functionality and design. Try to include some unit tests as well.

**Resources & Extra Learning**:

* <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events>
* <https://www.freecodecamp.org/news/dom-events-and-javascript-event-listeners/>
* <https://www.geeksforgeeks.org/javascript-addeventlistener-with-examples/>
* <https://www.makeuseof.com/javascript-event-listeners-ways-to-apply/>

|  |  |
| --- | --- |
| **Lab Exercise 4:** | **Slide 53** |

**Instructions**: Follow the instructions on the slide to utilise bootstrap elements to create a responsive design. Use the Device Toolbar in the browser Dev Inspector to resize your window and test the layout on different sized screens. Use a random image/s from the web to fill in the card data.

**Goal**: To practise using the Bootstrap library to build responsive HTML and CSS using Grid and Card components.

**Submission**: Create a new folder for **Exercise 4** in your **Module 4** folder/repo. Create a logically named HTML file containing your answer to the exercise, which uses the built-in Bootstrap CSS classes.

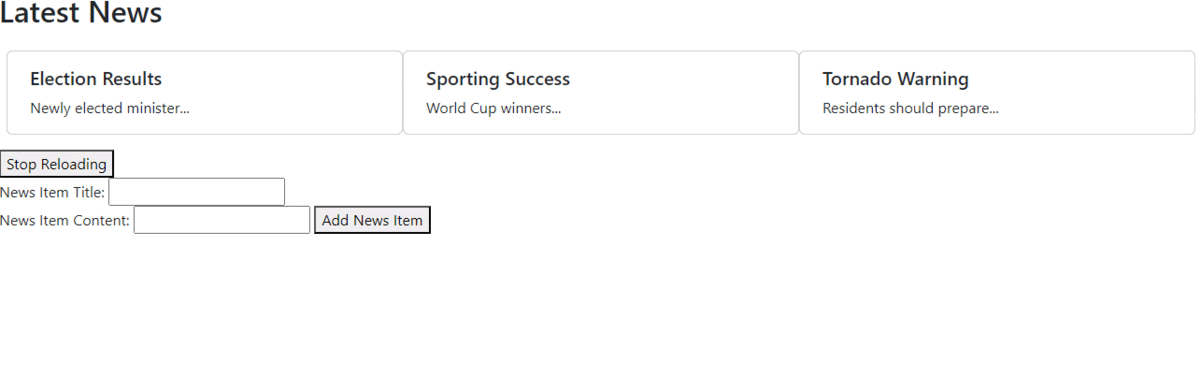
**Extension**: Try using the templating technique from Exercise 2 to create a template and populate it dynamically with JS to create the Bootstrap cards grid.

**Resources & Extra Learning**:

* <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
* <https://getbootstrap.com/docs/5.3/layout/grid/>
* <https://getbootstrap.com/docs/5.3/examples/grid/>
* <https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp>
* <https://www.geeksforgeeks.org/bootstrap-grid-system/>

|  |  |
| --- | --- |
| **Lab Exercise 5:** | **Slide 58-59** |

**Instructions**: Follow the instructions on the slides to dynamically populate templates with sample news article data that may change over time. Do both parts of the lab. Example of how parts 1 & 2 come together -

****

Frames in order:

1. When you first open the page
2. When you add content (newsItem to your news array)
3. Interval function kicks in and displays new data

Part 2 of this lab requires creating a simple form allowing a new news article to be added to the array, which will be picked up by the interval in part 1 and only then displayed on the screen. The code on slides 56-57 may help to get started.

**Goal**: To practise using data which changes over time and is unknown in advance, to dynamically populate templates.

**Submission**: Create a new folder for **Exercise 5** in your **Module 4** folder/repo. Create logically named HTML, CSS and JS files containing your answer to the exercise,

**Extension**: Try including a button that will stop the interval from reloading the news items, and another one to restart it. Try improving the look of the news articles with CSS or using Bootstrap cards in a grid.

**Resources & Extra Learning**:

* <https://developer.mozilla.org/en-US/docs/Web/API/Window/setInterval>
* <https://www.w3schools.com/jsref/event_preventdefault.asp>
* <https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault>

|  |  |
| --- | --- |
| **Lab Exercise 6:** | **Slide 69** |

**Instructions**: Follow the instructions on the slide to create a page displaying cards loaded from the JSON Placeholder API. Fetch the data and use templating techniques from previous labs to dynamically populate it into the page using Bootstrap grid/card components. Set the default limit to 10.

**Goal**: To practise using the Bootstrap library to build dynamic, responsive HTML and CSS using Grid and Card components populated with externally fetched data.

**Submission**: Create a new folder for **Exercise 6** in your **Module 4** folder/repo. Create logically named HTML and JS files containing your answer to the exercise, which uses the built-in Bootstrap CSS classes.

**Extension**: Try using both the fetch function and the Axios library to source the data. Try adding a simple form that allows the user to choose how many posts from JSON Placeholder to retrieve and display.

**Resources & Extra Learning**:

* <https://jsonplaceholder.typicode.com/>
* <https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>
* <https://axios-http.com/docs/intro>

|  |  |
| --- | --- |
| **Lab Exercise 7:** | **Slide 75** |

**Instructions**: Follow the instructions on the slide to create a simple CSS animation.

**Goal**: To practise using CSS animations to add movement and interest to front-end designs.

**Submission**: Create a new folder for **Exercise 7** in your **Module 4** folder/repo. Create logically named HTML and CSS files containing your answer to the exercise.

**Extension**: Follow the tasks at <https://css-animations.io/> to understand more about animations.

**Resources & Extra Learning**:

* <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animations/Using_CSS_animations>
* <https://themeisle.com/blog/css-animations-tutorial/#gref>
* <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_transitions/Using_CSS_transitions>
* <https://cssanimation.rocks/transition-vs-animation/>

|  |  |
| --- | --- |
| **Lab Exercise 8:** | **Slide 88** |

**Instructions**: Follow the instructions on the slide to implement the Moment.js library to perform some common tasks. For question 2 (years, months & days between birthdate and current date), don’t fret too much if you don’t get a hyper accurate representation. Make sure you follow the Moment.js documentation to know the kinds of functions available and make best use of them.

**Goal**: To practise using useful JS Date libraries to perform more complex date-based tasks.

**Submission**: Create a new folder for **Exercise 8** in your **Module 4** folder/repo. Create logically named HTML and JS files containing your answer to the exercise.

**Extension**: Try re-doing the exercise using the Date-FNS library - include the library via:  
 <script src="http://cdn.date-fns.org/v2.0.0-alpha0/date\_fns.js"></script>

**Resources & Extra Learning**:

* <https://momentjs.com/docs/>
* <https://date-fns.org/>

|  |  |
| --- | --- |
| **Lab Exercise 9:** | **Slide 94** |

**Instructions**: Follow the instructions on the slide to display a JS chart based on data sourced from the Fake Store API. Use the starter code on the slide, and replace the static data in the options with data fetched from the Fake Store API (you will need to process it a little to get the right format).

**Goal**: To practise using common JS charting libraries to display a chart based on dynamically sourced data.

**Submission**: Create a new folder for **Exercise 9** in your **Module 4** folder/repo. Create logically named HTML and JS files containing your answer to the exercise.

**Extension**: Try different types of charts including line and pie charts. Try using the same data to make another chart of products in different price brackets (eg. $1-$100, $100-$250, $250-$500, $500-$1000).

**Resources & Extra Learning**:

* <https://fakestoreapi.com/docs>
* <https://echarts.apache.org/handbook/en/how-to/data/dynamic-data>

|  |  |
| --- | --- |
| **Lab Exercise 10:** | **Slide 95** |

**Instructions**: Follow the instructions on the slide to create a mini ecommerce website using the data from the Fake Store API. Do at least the first 3 points to fetch the Fake Store product data, use templating to display it via bootstrap cards, and include a drop-down to allow users to filter the products by category.

You may wish to use this API and exercise as a starting point for your mini-project #1. If so, try to include all the optional extras and some ideas of your own, but if you choose a different mini-project topic don’t spend too long on this task but put the time into the mini-project instead.

**Goal**: To practise integrating front-end concepts and techniques into a realistic ecommerce scenario.

**Submission**: Create a new folder for **Exercise 10** in your **Module 4** folder/repo. Create logically named HTML, CSS and JS files containing your answer to the exercise.

**Extension**: Try to style your ecommerce store more professionally, and try the optional tasks in the exercise.

**Resources & Extra Learning**:

* <https://fakestoreapi.com/docs>
* <https://getbootstrap.com/docs/5.3/examples/dropdowns/>
* <https://getbootstrap.com/docs/5.3/components/card/>