- **Week 01**

Students should be able to

1. Explain what HTML is and how it works in a browser
2. Explain the terms of element, tag, and attribute
3. Explain the purpose of the title and meta elements
4. Explain character encoding, use UTF-8 for emojis
5. Insert favicons on a webpage
6. Use headings, paragraphs, lists, links, anchor tags, and images
7. Use GIT for (at least) cloning repositories, and doing some workflow (add, commit, pull, push) from a command line or GitHub Desktop.

Task: Create a personal webpage from scratch using newly learned skills.

Task: Create an account on GitHub. Watch/read some tutorials to gain a basic understanding of Git and GitHub.

- **Week 02**

Students should be able to

1. Use tables to show tabular content
2. Use tables for layout and formatting
3. Connect different web pages using anchor tags
4. Use frames to load web pages inside a page
5. Use form elements and identify different types of form input types
6. Publish a static website on github.io

Task: Update the personal webpage using newly learned skills. It should have multiple pages (Home, Education, and Contact Me pages). On the Contact Me page, change the form element so that content is encoded as plain text upon submission of the form. Modify your profile picture to be a circular image (you may use online circle-crop tools or installed software, e.g., Photoshop)

Task: Learn about HTML 5 and its benefits over previous versions. What is a semantic web, and how do semantic elements help better interpret the content of the webpage?

Task: On your personal website, add another page, which will be a questionnaire. Ask many questions to a user and collect answers. Submit the collected values to yourself via email. (Note: Use as many different form input elements as you can.)

Task: Try publishing your static HTML website to GitHub pages. You should try it before the session.

- **Week 03**

Students should be able to

1. explain the difference and use cases of inline, internal, and external CSS
2. identify simple selectors (element name, class, id) and pseudo-classes (e.g., :hover) and use them
- Extra: combinator selectors, pseudo-element selectors, and attribute selectors
1. use color and background-color properties
2. use custom fonts (e.g., get a custom font from, for instance, Google Fonts and apply it to the webpage)
- use font-family, font-weight, and font-size properties
1. use border-radius to have rounded images
2. use background images for pages
3. style borders of elements
4. use margins to better organize elements on a webpage
5. change the default list style
6. distinguish between div and span elements
7. identify the box model and use margin, border, and padding as necessary

Task: Update the personal website using CSS.

- all the web pages across your website should have the same background color and font style. Make the body of all pages 90% of the browser's and center it.
- make your horizontal rule shorter and horizontally in the middle of the page. Make it thicker, and change the style to dotted or dashed. See how.
- make your profile picture circular (using CSS) with a width and height equal to 200px.
- add necessary changes so that when the mouse is over the profile picture, it will grow a bit (say, width and height equal to 220px)
- read about the CSS styling of a table and apply it to the Education page.
- read about the CSS styling of a list element and apply it to your lists across the website.
- read about the CSS styling of a form and apply it to the Contact Me page.

- **Week 04**

Students should be able to

1. use the display property to change the default settings of elements and control the layout
2. display: inline vs display: block
3. display: none vs visibility: hidden
4. display: inline-block
5. use position property to control the positioning method type of elements
6. static, relative, fixed, absolute

7. bonus: sticky
8. use the float and clear properties to control how elements should float
9. use the flexbox layout module to create responsive web pages without float or positioning.
10. bonus: use grid layout as an alternative to flexbox
11. use media queries to apply different style rules based on certain conditions, such as media type or screen size.

Task: Update the personal website using newly learned properties of CSS:

1. Profile image and bio should look the same, but without using tables.
2. Create a navigation bar to hold all the links among pages and make it stick to the top even when you scroll.
3. When the mouse pointer hovers over a link, its style (background color and color) should change.
4. The link to the active web page needs to be styled differently, and hovering on it should not change its style.
5. Apply the styling to all the pages of your website.
6. Create a photo gallery with at least three or four pictures of you or pictures from one of your projects.
7. Use your imagination to make your website look even nicer.
8. Employ some flexbox properties and see if it looks the same way with less effort or not.

- **Week 05**

Students should be able to

- write consistent JavaScript codes to solve problems
- use variables, comments, and operators
- use conditional statements (if and switch)
- use Chrome developer tools to practice JavaScript codes
- use arrays and strings
- use string interpolation and template literals
- construct repetitive blocks using while, for, for..in, and for..of statements
- use Math functions
- generate pseudorandom numbers
- use date and time values

Task: Create an interactive program in JavaScript to

1. Get the full name and course grade of the user.
2. Greet the user.
3. Let the user know if she/he passes or fails the course by alerting a message.

4. In case the user passes the course alert the letter grade.

Task: Given an array of integers as input

1. Find the sum, average, min, and max of the elements.
2. Find the most repeated and least repeated element of the array.

[Bonus] Task: Game of Craps

1. One of the casino games.
2. The rules are as follows (simplified for practice purposes)
3. Player rolls two dice.
4. If the come-out roll is 7 or 11, the bet wins.
5. If the come-out roll is 2,3 or 12, the bet loses (known as "crapping out").
6. If the roll is any other value, it establishes a point.
7. If, with a point established, that point is rolled again before a 7, the bet wins.
8. If, with a point established, a 7 is rolled before the point is rolled again ("seven out"), the bet loses.
9. Our task is to let the computer play a Game of craps and visualize the progression, finally displaying whether it wins or loses.
10. How about if we ask the computer to play N number of rounds and print the results of each?

[Bonus] **Task:** The Sieve of Eratosthenes

1. A prime integer is any integer greater than 1 that can be divided evenly only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers.
2. It works as follows:
3. Create an array with all elements initialized to 1 (true). Array elements with prime indices will remain 1. All other array elements will eventually be set to zero.
4. Starting with array index 2 (index 1 is not prime), every time an array element is found whose value is 1, loop through the remaining indices of the array and set to zero every element whose index is a multiple of the index for the element with value 1.
5. For array index 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (indices 4,6, 8, 10, and so on.).
6. For array index 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (indices 6, 9, 12, 15, and so on.).

Task: Exploded string

1. You are given a string (i.e., "Baku")
2. Define a method to return an *'exploded'* string (i.e., "BBaBakBaku").

Task: Sorting strings

1. Given a string. Can you sort its letters in alphabetical order?
2. Define a method that takes a string and returns the "sorted" string.

Task: Anagrams: ref

1. Two strings are called anagrams if one can be obtained by rearranging the letters of another.
2. Define a method that takes two strings as input parameters, returns true in case they are anagrams, and false otherwise.

Task: Slices

1. Given a word. Print out all the possible slices of that word. (Consider slicing as dividing a word into two parts)
2. Example:
3. In: SITE
4. Out: (underscore can be replaced by space or tab)

_SITE

S_ITE

SI_TE

SIT_E

- **Week 06-Part1 (Functions)**

Students should be able to

- identify function declarations and function expressions to define custom functions
- use ES6 arrow functions
- providing default values for input parameters

- use some common array methods, e.g.,forEach(), map(), filter(), etc.

Task: Fibonacci elements in the given range. Fibonacci sequence: 0 1 1 2 3 5 8 ….

1. Define a function to take two indexes as input arguments and return an array containing Fibonacci elements from the range. E.g., fib(3,6) would return an array [2, 3, 5, 8]
2. Get two integers from the user
3. Print the Fibonacci elements from this range using the defined function.

Task: Random numbers in a range

1. Define a function that will take two numbers as a range (a and b, both inclusive) and the number of elements (n) to be generated. It will return n random numbers as an array. E.g., rand(10,100, 5) would return an array containing 5 random numbers.
2. Create another function that will take an array of random numbers and count each number. E.g., count([3,5,2,6,3,4,5]) would return {3 => 2, 5 => 2, 2 => 1, 6 => 1, 4 => 1}
3. Take from the user three numbers, a and b for range and n for the number of elements, and print out the list with the number of each element: E.g.,
4. 3 is repeated 2 times
5. 5 is repeated 2 times
6. 2 is repeated 1 time
7. 6 is repeated 1 time
8. 4 is repeated 1 time

Task: “NoRmAlIzE” a full name

1. Define a function called proper, which will take a string and convert its first char to uppercase and the rest to lowercase. E.g., proper(“ab”) would return “Ab”.
2. Define a function called normalizeWord which will take a word, chunk it down into two-character pieces, and apply the proper function to each piece. E.g., normalizeWord(“abcdefg”) would produce “AbCdEfG”.
3. Define a function called normalizeSentence which will take a sentence and split it by space. Then it will apply normalizeWord function to each word in the sentence. E.g., normalizeSentence(“abcd efghijklm”) would return “AbCd EfGh IjKlM”.
4. Prompt the user for his/her full name and “Normalize” it using the normalizeSentence function.

[Bonus] **Task**: Create a "Wordle" game alternative

1. Take a random word from a list of words (you may decide on the word list)
2. Get from the user a word
3. Let the user know if the entered word is a correct guess or not.
4. If yes, the player wins.
5. Otherwise, let the user know that he has made fewer attempts. (Say, initially it was 5 attempts; every wrong guess will decrease it)

Task: Expression evaluator (advanced)

1. Given an expression in the form of "operand1 operator operand2". Evaluate the expression based on any given binary operator.
2. You may use a map of (operator name, operator function) to keep a record of all the available operators in the program.
3. Define another function where given (operator, operand1, and operand2) as input arguments, evaluates the result of the expression if it is supported, otherwise returns a result of a default operation (say, sum operation).

CLASSWORK

Task: Given a list of favorite fruits. Print out the same list, normalizing each favorite item.

Sample: IN: apple, banana, pomegranate OUT: Apple, Banana, Pomegranate

Task: Given a list of names. Print out only those starting with lowercase letters.

Sample: IN: ["Nuraddin", "sama", "Aliya", "seymur"] OUT: ["sama", "seymur"]

Task: Given an array of numbers. In one iteration find the min and max of the array.
Challenge: Use a Reducer.

- **Week 06-Part2 (Objects and classes)**

Students should be able to

- work with objects
- use function constructors

- add/remove properties and methods to an existing object.
- add/remove properties and methods to a function constructor using object prototypes.
- Iterating over object properties and their values: for…in and for…of
- Work with classes.
- use extends keyword to achieve class inheritance.
- use static keyword to define class methods.
- work with set and map data structures in JS

Task: Create a function constructor Person:

1. In the constructor get firstname,lastname and date_of_birth and assign the object properties.
2. Define a fullName() function which will return the Person's full name.
3. Define an age() function which will return the age of the Person object (i.e., current year – the year the person was born)
4. Define toString() function which will return a string: FullName is x years old!

Task: Given an array of book objects (https://gist.github.com/nanotaboada/6396437)

1. Write a JS program that will define a class Book.
2. Create an array of Books that holds the book info from the above link.
3. Ask from user which based on which field, and in which order they want to sort the books:
4. The user might enter isbn ASC or titleDESC.
5. If there is no such property, then show an error message, otherwise log the list of books in sorted order.
6. Note: use any sorting algorithm you find convenient, but make sure you implement the algorithm yourself.

Task: Study JSON format (https://www.json.org/json-en.html) and practice JSON.stringify(obj) method.

- **Week 07**

Students should be able to

- understand and traverse through the DOM tree.
- use getElement…() methods as well as querySelector() or querySelectorAll() methods to access HTML elements, their attributes and styles and modify them.
- add/remove a class to the classList of an element (add(), remove(), and toggle() methods)

-
- understand what HTML event attributes and DOM events are and how to react to them
- using HTML attributes (on***event*** property in JS) to react DOM events
- registering event listeners to react to DOM events
- adding multiple event listeners to the same element
- removing a registered event listener
- explain event bubbling vs event capturing
-
- create new DOM elements and change their properties.
- append/insert new elements/nodes in the DOM tree.
- remove existing nodes from the DOM tree.
- explain the difference between DOMCollections and Node Lists (e.g., use children and childNodesproperties or getElementsByTagName(), getElementsByClassName(), and querySelectorAll()functions to experience the difference)

CLASSWORK

Task: Imagine an HTML document with many heading elements. Change the style of the heading elements to blue or red depending on their index: even indexes should be colored in blue, and the rest in red.

Task: Create a simple counter, where three buttons are used (increment, decrement, reset) and a display of the counter value.

Task: Create a random image picker from Lorem Picsum given different widths and heights. The page contains input fields for width and height, a button to submit, and a place to display the fetched image.

Task: Update any existing project (e.g., Class Work – Week 07)

1. When the page is loaded, h1#titleelement should show “Good Morning”, “Good day”, “Good evening” or “Good night” depending on the time of the day.
2. The element h4#date should show the current date.
3. Create a directory images/ and put several images in the same folder. Every time the page is loaded one of these images should be randomly selected and shown in the img.

4. Make sure each li with an even index is italicized and each with an odd index is boldfaced.

Task: Create a simple Rock, Paper, Scissors game.

1. There are two players (for now each is a computer).
2. Create two div elements where you will show the move of each player.
3. When the page is loaded, the div on the left is going to show the move of the first player (chosen randomly out of Rock, Paper, or Scissor) and the div on the right will do the same.
4. Use the fonts below to make the game interesting.
5. If the first player wins, show a message "Player 1 wins", if the second one wins, show a message "Player 2 wins", otherwise show a message "A Draw".
6. See if you can also show a smiling face on the side of the winning player.

https://fontawesome.com/icons/hand-back-fist?s=solid&f=classic

https://fontawesome.com/icons/hand?s=thin&f=classic

https://fontawesome.com/icons/hand-scissors?s=light&f=classic

Task: Update the Rock, Paper, Scissors game so that a user can play with the computer.

1. Users can select to start the round by clicking on a button saying, "Start nth round".
2. Here n is a counter starting from 1. E.g., start 1st round or start 5th round, etc.
3. Every time the user starts a new round, it is updated.
4. Then the user is prompted to select one of the moves: rock, paper, or scissors. As soon as the user makes a choice by clicking on one of them, computer's choice is shown as well.
5. The result of the current round is shown as both players (the user and the computer) make a choice.
6. The overall score (accumulated score of all the rounds played) should always be shown at the top/bottom.

Task: Update the existing project (i.e., Personal Website).

1. Make sure each li with an even index is italicized and each with an odd index is boldfaced for both ol and ul elements.
2. When the mouse is over a list item its font size should slightly be increased.
3. The user should be able to add a new drink to the list of drinks.

4. The user should be able to add a new item under the action plan as well.
5. For both lists, the user should be able to remove the item from the lists:
6. For drinks, the item should be removed from the list when the item is clicked.
7. For the action plan, it should not be removed entirely, but instead, the click will strike through the list item as if it is checked. If the already checked list item is clicked again it will be unchecked meaning the strikethrough will be undone.
8. Create a third list (My favorite books)
9. The user should be able to add new books to the list as in the previous two lists.
10. When the left mouse button is double-clicked, a popup menu will be shown right next to the selected list item. the mouse button is double-clicked, and a popup menu will be shown right next to the selected list item.
11. There are two options: remove and strikethrough.
12. Once any of them is chosen, the pop-up menu is dismissed, and the action is taken according to the user's choice.

Task: Create an input text to let the user enter the name.

1. While the user types in the name, there is a paragraph or span element that shows the live update of the name entered in the input field.
2. Bonus: can you make sure that the user cannot enter any lowercase letter, i.e., any character the user enters should be converted to uppercase and shown not only in the paragraph or span but also in the input field itself?

Task: Update the gallery from week 4.

1. On your personal website, update the image gallery page.
2. The gallery is showing one photo at a time. There are Prev and Next buttons, and when clicked, they make the current photo change accordingly.
3. In the first picture, the Prev button is disabled but still shown.
4. In the last picture, the Next button is disabled but still shown.

Task: Update the Game of Craps from week 5

1. When the page loads it shows a button to start the round.
2. Simulate two 6-sided dice that is rolled on the page.
3. According to the outcome
4. inform the user s/he wins by displaying a proper message
5. anytime, in case the user wins or loses, display an appropriate message
6. let s/he decide weather to continue playing or not by pressing the new button "Continue?"

7. in case of continue, the dice are rolled again according to the outcome the play continues
8. Note: you may find images from the web or icons from the fontawesome of a 6-sided die

- **Week 08**

Students should be able to

- add jQuery to the website using either a downloaded library or included from a CDN.
- use different jQuery selectors.
- handle events using jQuery functions.
- use .on() function to register multiple handlers to different events
- practice setTimeout() and setInterval() functions
- use jQuery to access and manipulate CSS style and HTML attributes of an element
- create, add, and remove HTML elements into the DOM tree using jQuery
- using .on() function will help with registering listeners for dynamically created elements as well.
- e.g., Try the following:$(parent).on(eventType, selector, callback)
- add, remove, and toggle classes of an element using jQuery
- use AJAX to send GET/POST requests, receive responses, and process the data
- start a JSON server to fake theREST API without writing any backend code
- use Vanilla JS or jQuery to send/receive HTTP requests/responses to the fake JSON server
- use alternatives of AJAX, such as Fetch API or Axios HTTP client.

Task: Using jQuery to change the visibility of an element.

1. Create two paragraph elements with two different classes followed by a button with the text “Click me!”.
2. When the button is clicked for the first time the first paragraph should be hidden (not display:none)
3. When the button is clicked the next time the first paragraph is visible again.
4. Every time the button is clicked, the paragraph keeps changing its visibility.
5. When the button is clicked once the text should be changed to “Click again!”.

Task: Complete the previous week’s tasks using jQuery this time.

Task: Create a ToDo list app using jQuery.

1. Design from scratch or find a lovely design file (e.g., a Figma file) and work on it)
2. Once the design is complete go ahead and use jQuery to develop the dynamic parts.
3. The user should be able to add, remove or cross-out the items.

Task: Timer

1. Create a timer to set a time and start the countdown.
2. The user sets the time and starts the counter.
3. Counter works until it reaches ZERO.
4. Users can reset it anytime.

- **Week 10**

Students should be able to

- create a React application incodesandbox.io and render some simple web pages
- setting up a React application using CDN for React and Babel
- explain and use JSX
- create new modules, use ES6 import and export statements
- create function and class components to build a static web page
- explain the difference between function and class components
- use CSS to design components

Task: Try to create React app to rewrite the Personal Web Site you built in the first half of the semester.

Task: In the React application we created during the class change the background color depending on the time of the year (i.e., seasons) and change the title (Welcoming message).

1. You should consider Morning, Afternoon and Evening.

Task: Update the wm1-react-starter-project-components project (your own copy)

1. In the header, add a logo to the left of the title (React Project – Components)
2. The logo should be aligned to left and the title should be aligned to right (you may use React logo or ADA logo or even any logo you like)
3. Change the background color of the header slightly
4. Add some shadow to the header (you may use the box-shadow property)
5. Apply a similar styling to the footer as well.
6. Add some more information about each car component.

Task: Quotes app

1. Following the wm1-react-starter-project-components project create a new react app from scratch
2. You may add new header and footer sections as you feel appropriate.
3. In the main section you will need to render some boxes with quotes:
4. Each quote instance will have quote_text, category (or tag), and the author.
5. Try to render several boxes even though they hold the same information.
6. Style these as you see fit.
7. Next time, you will learn how to dynamically set the values of the component instance using props.

- **Week 11**

Students should be able to

- use props to create different instances of components dynamically
- use the map() function to map some data to a list of components
- get a JSON array (exported by another module) and map it into components
- Bonus: practicemap(), filter(), reduce() functions in JS.
- Refer to the documentation for JSArray
- create a React application using an a node environment using npx command
- run a locally created React application
- deploy locally developed React application to GitHub pages.

Task: Create a similar app to Google Keep App

1. In codesandbox, create a React App from scratch
2. In this app, you will create a notes.js module which will export an array of notes
3. Each note has a title, content, created date, and optionally an image
4. In your array create at least five notes where some have images while others do not
5. Create a Note component that can display the information about a note.
6. Design the Note component as you wish (you may refer to the Google Keep App)
7. Render the list of Notes in your main App
8. Which has a Header, Footer, and the Main sections

Task: Create the Google Keep App analogue locally

1. Build and run the application on the local machine
2. Deploy the application to the GitHub pages.

- **Week 12**

Students should be able to

- practice ES6 expression-destructuring assignment
- apply destructuring to arrays and objects
- effectively destructure arrays and objects passed as input arguments
- serve assets (e.g., image files) from ./src folder and use them in the application by importing
- conditionally render different segments of the page using traditional if…else statement, ternary operator (inline if), and logical && operator
- add events to components and elements, handle events, and pass arguments to event handlers using an arrow function
- use the event object to access the event object being triggered, the element or component that triggered it (i.e., event.target), etc.
- use the ES6 spread operator to expand arrays and objects
- use React hooks (useState) to create a stateful functional components
- set the state of a functional component using setter function
- changing multiple states and complex states
- change the state of a React component providing a callback function

Task: Extend the Note Keep App (Google Keep Analogue)

1. We mentioned that the image is optional. So, what if it is missing?
2. Conditionally render the image so that if the image is missing, it does not render an img element at all.
3. Apply destructuring assignment to objects that you pass around the components
4. Add an event listener to the Note component so that when clicked, it alerts a message with the title and the creation date of the note.

Task: Extend the *wm1-react-starter-project-destructuring* project

1. apply different colors to the h3 element depending on which button is clicked

- **Week 13**

Students should

- create a React app (ToDo List) and review all topics covered so far
- explain the difference between controlled and uncontrolled form elements
- changing the state of the parent component from the child components
- passing callback functions to child components through props
- passing data from child components to the parent component(s)

Task: Update ToDo List

1. when clicking on a ToDo Item change its style (i.e., put a line through the item, change its color, or mark it as completed)

Task: Update ToDo List

1. when clicking on a ToDo Item, remove it from the tree (list of all the items)

Task: Update ToDo List

1. move the input part (text and button) out to another component. You may decide on the name of the component and use an instance of this component in the App
2. make sure you achieve the same functionality using this style as well
3. assure that the user cannot add an empty string as a new ToDo Item.

Task: Update Google Keep App Analogue

1. In the Keep App you have built locally, try to achieve the same functionality.
2. Users can remove or cross out a note from the list of all.
3.

- **Week 14**.

Students should

- use createContext and useContext to pass down properties to all the children components
- manage states using useReducer

Task: In the react-starter-project-managing-context-and-reducers project, apply the theme changes to all the subcomponents.

Task: In the react-starter-project-managing-context-and-reducers project, update the TodoItem component so that there is complete and remove features. Consider passing *dispatch* function as a callback to the child component(s)

Week 15

Students should

- use useEffect hooks to handle (side)effects in a React app function components
- use fetch API to fetch data from external resources in a React app
- handle fetch errors
- use React Router library to add routing to the applications
- use BrowserRouter, Routes, Route, Link, and NavLink components
- route URLs with URL params
- use useParam() Hook to get the parameters passed in the URL
- use nested routes
- catch all the pages that are not found

Task: Update the Products app developed in class

1. add an input field via which users can enter the product id to search for
2. when submitted, it should load the product with the id
3. try to search for product with the id = 1000
4. see if you can handle this error and properly display an error message instead of attempting to render a product

Task: Update Google Keep App Analogue

1. This time you are supposed to start a JSON server that serves an array of notes
2. Fetch the data returned from the server and populate the Note component instances

Task: Add Routes to your final project (Assignment 3)