

**WSUK – Test Project  
17. Web Technologies**

**Training Pack Samples**

1 - SPEED

Contents

[Introduction 2](#_Toc159885304)

[Part A: Easy 3](#_Toc159885305)

[A1: Yellow Square (Easy) 3](#_Toc159885306)

[A2: Jumper Ball (Easy) 3](#_Toc159885307)

[A3: Loading Screen (Easy) 3](#_Toc159885308)

[A4: Smiley (Easy) 3](#_Toc159885309)

[A5: 3](#_Toc159885310)

[Part B: Medium 4](#_Toc159885311)

[B1: Simple Image Filter (Medium) 4](#_Toc159885312)

[B2: Transition (Medium) 4](#_Toc159885313)

[B3 RGB Slider (Medium) 5](#_Toc159885314)

[B4 Image blur linear slider (Medium) 5](#_Toc159885315)

[B5 6](#_Toc159885316)

[B6 Triangle (Easy) 6](#_Toc159885317)

[B7 Grayscale image (Easy) 6](#_Toc159885318)

[B8 Jumper Ball (Easy) 6](#_Toc159885319)

[B9: HTML/CSS – Contact Form (Normal) 6](#_Toc159885320)

[B10: Border Radius & Box Shadow (Easy) 6](#_Toc159885321)

[B11: Race track (Normal) 6](#_Toc159885322)

[B12: Video player interface (Layout) 6](#_Toc159885323)

[B13: Gradient animation (Easy) 6](#_Toc159885324)

[B14: Modern layout (Easy) 7](#_Toc159885325)

[Part C: Difficult 7](#_Toc159885326)

[C1: Linear gradient (Difficult) 7](#_Toc159885327)

[C2: Digital Timer (Difficult) 8](#_Toc159885328)

[C3: Drawing Canvas (Normal) 8](#_Toc159885329)

[C4: RGB Slider (Normal) 8](#_Toc159885330)

[C5: JS – Calculation (Easy) 9](#_Toc159885331)

[C6: JS - ASCII-Art (Difficult) 9](#_Toc159885332)

[C7: HTML/CSS, JS – Clock (Difficult) 9](#_Toc159885333)

[C8: HTML/CSS, JS - Tag-Control (Normal) 9](#_Toc159885334)

[C9: List of expenses: (Easy) 9](#_Toc159885335)

[C10: Simple Image Filter (Difficult) 9](#_Toc159885336)

[C11: Canvas Animation (Easy) 10](#_Toc159885337)

[C12: JS Arithmetic (Difficult) 11](#_Toc159885338)

[C13: JS Draggable (Normal) 11](#_Toc159885339)

[C14: Keyboard (Difficult) 11](#_Toc159885340)

[C15: Parallax (Normal) 11](#_Toc159885341)

[C16: Smiley (Normal) 12](#_Toc159885342)

[C17: Transition (Normal) 12](#_Toc159885343)

[C18: Animation (Normal) 12](#_Toc159885344)

[C19: Chart from data (Very Difficult) 12](#_Toc159885345)

[C20: Video Gallery Page (Medium) 13](#_Toc159885346)

[C21: Image blur linear slider (Normal) 15](#_Toc159885347)

[C22: Circles (Normal) 15](#_Toc159885348)

[C23: Square animation (Easy) 15](#_Toc159885349)

[C24: Linear gradient (Normal) 15](#_Toc159885350)

[C25: Fortune wheel (Difficult) 16](#_Toc159885351)

[Part D: Back-end Development (D1 – D10) 17](#_Toc159885352)

[D1: PHP Calendar (Difficult) 17](#_Toc159885353)

[D2: PHP Array (Easy) 17](#_Toc159885354)

[D3: PHP Verification captcha (Difficult) 17](#_Toc159885355)

[D4: Watermark (Easy) 17](#_Toc159885356)

[D5: Predominant colours (Easy) 17](#_Toc159885357)

[D6: Image Cropper (Normal) 17](#_Toc159885358)

[D7: A Pagination Code (Normal) 18](#_Toc159885359)

[D8: XML2JSON Converter (Difficult) 18](#_Toc159885360)

[D9: Remove duplicates (Easy) 19](#_Toc159885361)

[D10: Answer checker (Easy) 19](#_Toc159885362)

[D11: Tic-Tac-Toe (Difficult) 19](#_Toc159885363)

[D12: RGB to HEX (Medium) 19](#_Toc159885364)

# Introduction

In this module, you have to complete these 15 mini tasks. These mini tasks are divided in 3 parts as easy, medium and hard. As a time, guidance for the speed tasks, you should aim to solve the requirements as follow:

* Easy =< 5 min
* Medium <> 5 – 15 min
* Difficult <> 15 – 30 min

Create a new folder for each test number as it is provide in the test project.

# **Part A: Easy**

# **A1: Yellow Square (Easy)**

Make an animation that starts with a big yellow square, and this square turns in red circle, the animation needs to be cantered on the view and the animation needs to loop. A sample is provided as *sample.mp4*. (*CSS only – no JS is allowed*).

# **A2: Jumper Ball (Easy)**

Make a loop animation of a jumper ball respecting all states from an animation (*CSS only – no JS is allowed*).

# **A3: Loading Screen (Easy)**

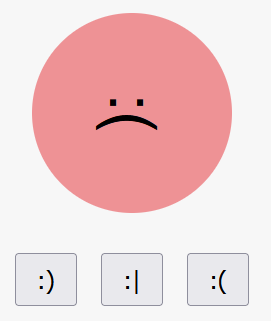
Create an animated loader using only HTML and CSS same as sample.mp4. The loader animation should be looped. 4 different colours (#19A68C, #F63D3A, #FDA543, #193B48) should be used. (*CSS only – no JS is allowed*).

Screen shots:



# **A4: Smiley (Easy)**

Create a big smiley face on the centre of the page and it is 200px width and height. Below the smiley, there are 3 buttons with texts: “**:)**”, “:|”, and” :(“. By clicking the buttons, the smiley face changes to that state with a 0.5s animation. By default, the smiley is in “:|” state. The states have these designs:



# **A5:**

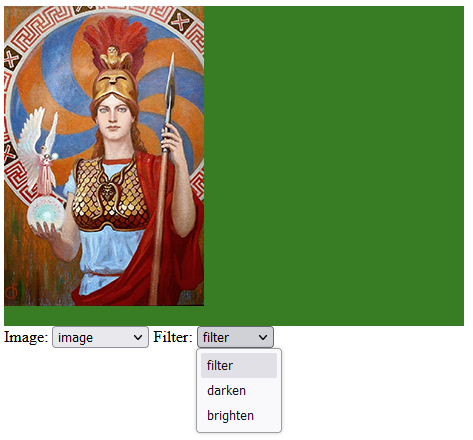
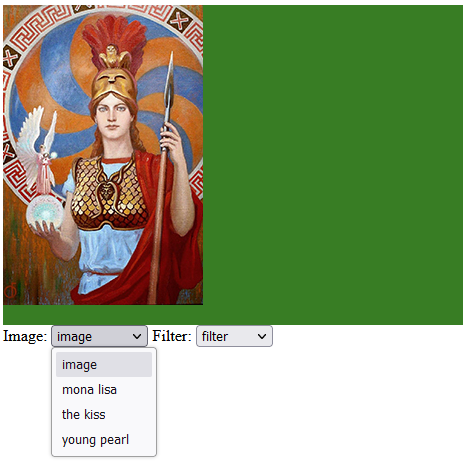
# **Part B: Medium**

# **B1: Simple Image Filter (Medium)**

For this project you must create a function that allows you to add a filter in a certain image using canvas. This speed Project is divided into two parts.

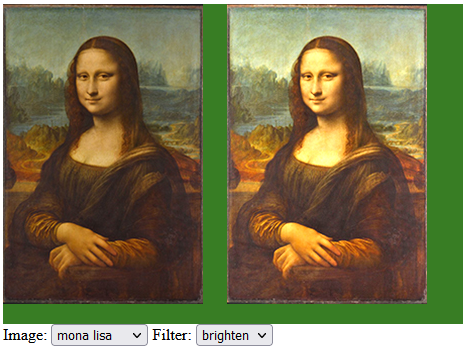
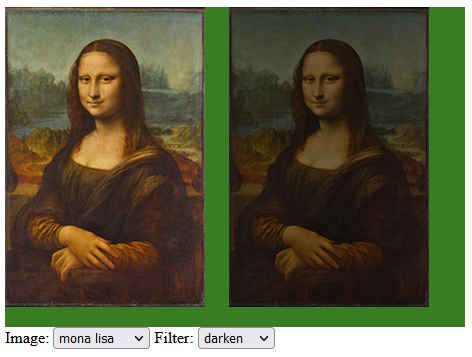
**Part 1 -** Develop the following design:

* The size of the canvas should be 460 pixels wide and 320 pixels high.
* Create a drop-down list, with the image options: athena.jpg, the-kiss.jpg, young-pearl.jpg and monalisa.jpg. There must be a first option called image.
* Load the images on the canvas depending on the option chosen in the drop-down list.
* The image of athena.jpg is loaded by default.



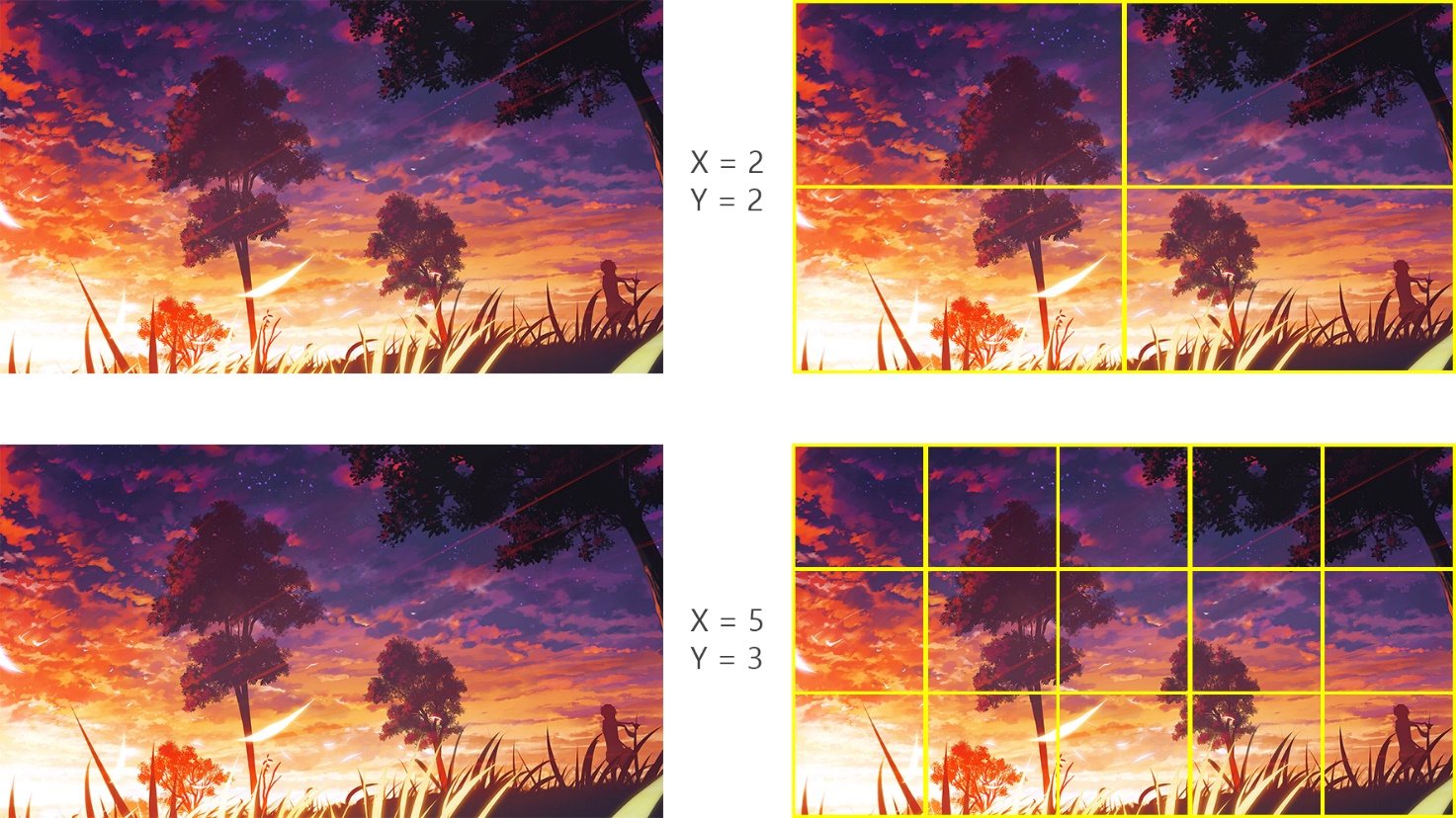
**Part 2**

* Create a drop-down list, called filter with the options: Darken and Lighten. There must be a first option called filter.
* When choosing a specific image in the dropdown list and applying a filter, the result should appear on the right side of the canvas according to the chosen option.



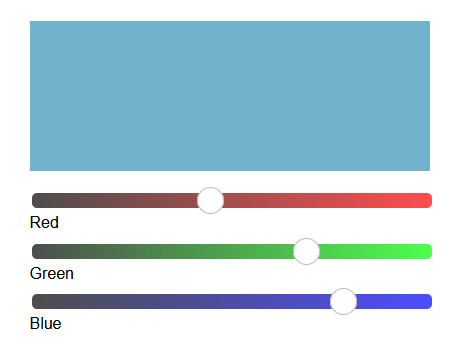
# **B2: Transition (Medium)**

You are given a picture that you need to cut into N cards and each card should disappear animated. X is 5 and Y is 3. Refer to the sample provided (b2\_media.mp4). You can use either CSS or JS.



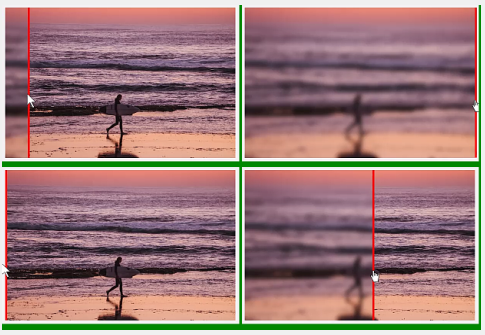
# **B3 RGB Slider (Medium)**

Please implement an RGB slider as following. There are 3 sliders for adjusting values of red, green and blue. Refer to the sample provided (b3\_media.mp4)



# **B4** **Image blur linear slider (Medium)**

Using JavaScript create a linear slider that will blur an imagine progressively left and right. The required effect is presented in the b4\_media.mp4. The slider should start as default from 50% of the imagine.



# **B5**

# **B6 Triangle (Easy)**

Make a blue equilateral triangle on the centre of the view, and animate this to transform a yellow square. the animation needs to be fluid and loop infinite;

# **B7 Grayscale image (Easy)**

Turn the image on grayscale, and when hover the mouse, the image needs to be colour with a transition;

# **B8 Jumper Ball (Easy)**

Make a loop animation of a jumper ball respecting all states from an animation (only CSS);

# **B9: HTML/CSS – Contact Form (Normal)**

An HTML and CSS implementation of a not yet finished form are given.

Modify the CSS and HTML so that the form looks as depicted in the screenshot in the media files. Its responsive behaviour should follow the screencast in the media files.

*The use of JavaScript is NOT allowed.*

# **B10: Border Radius & Box Shadow (Easy)**

Create a page that contains a 400px rectangle in the middle.

Inside the rectangle there is a settings panel with inputs, where you can setup the rectangle's Border radius, and box shadow properties. The rectangle’s CSS updates on input changes.

# **B11: Race track (Normal)**

By using HTML and CSS, create a race track animation in the form of “8” (vertically or horizontally), along which the car will endlessly animated.

# **B12: Video player interface (Layout)**

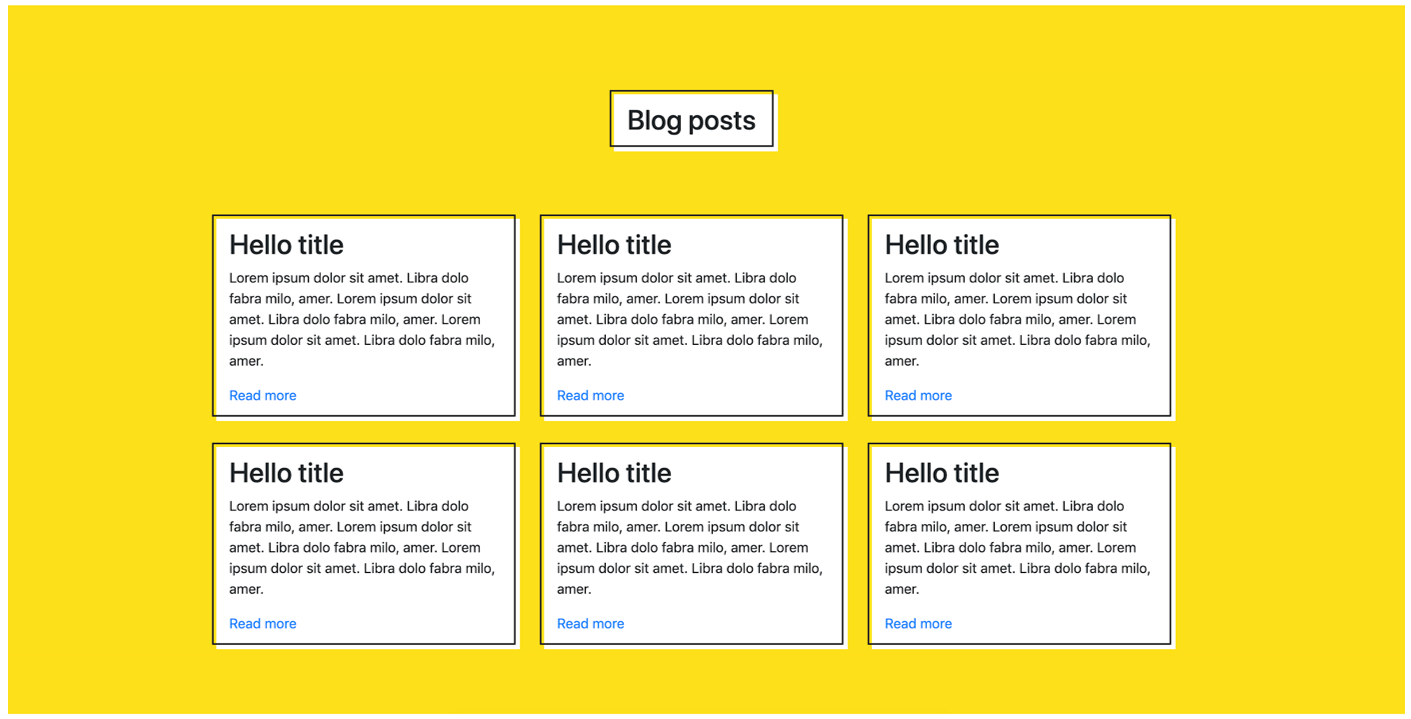
Create video player interface: search form, browse screen, action buttons (play, mute, and volume), progress slider, playlist (one item from playlist must be active), information about the video (title, description, number of views), comments (not less than 5) (and 1 attached comment).

# **B13: Gradient animation (Easy)**

By using HTML and CSS, create a gradient shift animation on the page. The gradient may move into any direction.

# **B14: Modern layout (Easy)**

Your task is to create the following layout, the posts’ width is ⅓ of the containers on desktop screen size (>=1440px) and full width below this.



# **Part C: Difficult**

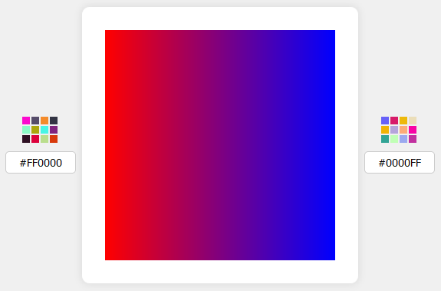
# **C1: Linear gradient (Difficult)**

Create the JavaScript code for the following task:

On the centre of the screen there is a 300 px sized square. By default, it has a red to blue linear gradient background. On the left and right of the square there are 1-1 input fields where you can change the starting, and ending colour of the linear gradient, by writing in the HEX value of the colour.

Over the inputs there are 12 randomized colour buttons (10px\*10px) on each side. If you click one of the randomized colours, its colour is inserted in the input. Whenever any input is changed, and is a valid hex, that side’s linear gradient colour changes on the square.

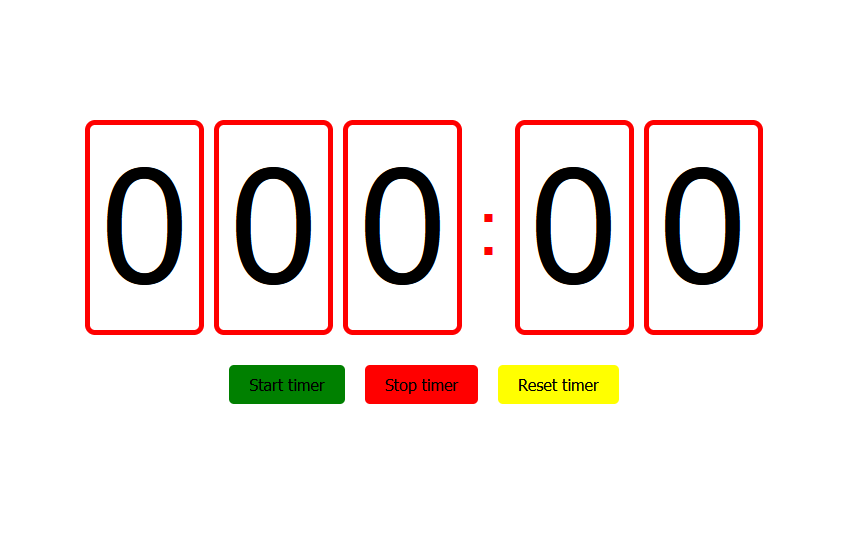
If you manually input any valid HEX value the colour should change to the given value.



See c1\_media.mp4

# **C2: Digital Timer (Difficult)**

Time format is: 3 digits of seconds passed, and cent seconds (*a unit of time equal to 0.01 seconds*) following it. Its max value is: “**999:99**” seconds. **Start timer** button starts, **Stop timer** button stops, **Reset timer** button stops and resets to 0 the timer.



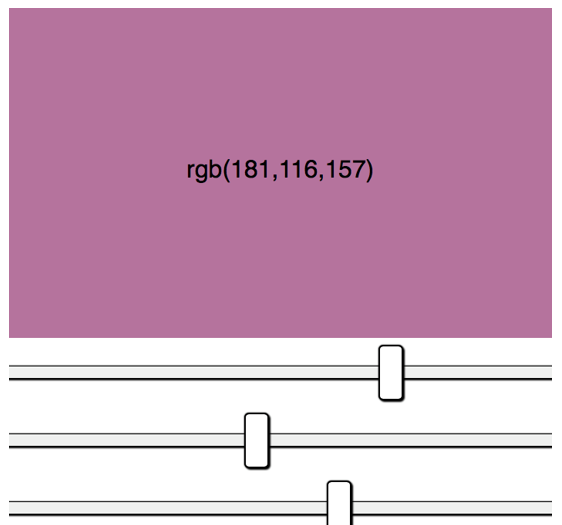
# **C3: Drawing Canvas (Normal)**

Please try to use canvas to build a drawing pad with the following features:

* Basic drawing with mouse down and move
* 3 colours switching

# **C4: RGB Slider (Normal)**

Please implement an RGB slider as following. There are 3 sliders for adjusting values of red, green and blue.



# **C5: JS – Calculation (Easy)**

Implement the function that calculate the sum of the squares of the odd numbers in a given array.

# **C6: JS - ASCII-Art (Difficult)**

You should implement the two methods for transforming the ASCII-input (an image made of standard characters) as described in file “ascii-art-editor.js” (rotate and mirror).

You can check the functionality in index.html – it will execute tests (call your functions with different input and test the output) and show its results.

# **C7: HTML/CSS, JS – Clock (Difficult)**

You have to implement a clock showing the current local system time.

The background should look as depicted in the media file “full-clock-fancy.png” containing clock border and ticks for hour and minute. Also, the clock hands should look as depicted in the image. Seconds clock hand is updated every second, minutes clock hand is updated every minute, hour clock hand is updated every minute.

The image “full-clock-fancy.png “can be used as you wish.

# **C8: HTML/CSS, JS - Tag-Control (Normal)**

You have to implement a Tag-UI-Control as shown in the animated GIF in the media-files.

Using the control, it should be possible to select tags from taglist.json, add new ones or remove selected tags. Typing should filter the available tags and – if none available – add the text as new tag.

# **C9: List of expenses: (Easy)**

1. Create a form with two fields called: **Spending** **name** and **Cost** and an **add** button.
2. Using JavaScript, perform the following functions:
3. Get the values of the fields.
4. Validate the entry of the data, if the fields are empty, print the message "There was an error", otherwise print "Correct".
5. Remove the expanded message and the data entered in the form after 3 seconds.
6. The data after being entered, must be shown in an ordered list using the HTML tags.

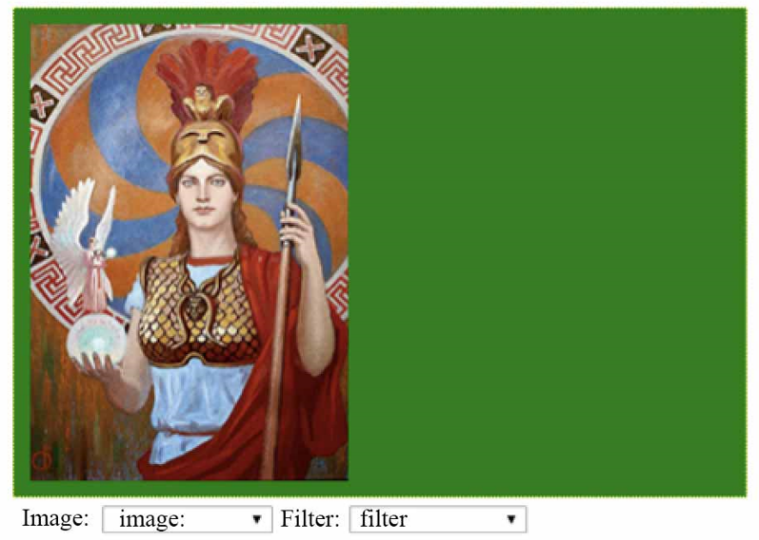
\* No resources are provided.

# **C10: Simple Image Filter (Difficult)**

For this project you must create a function that allows you to add a filter in a certain image using canvas. This speed Project is divided into two parts.

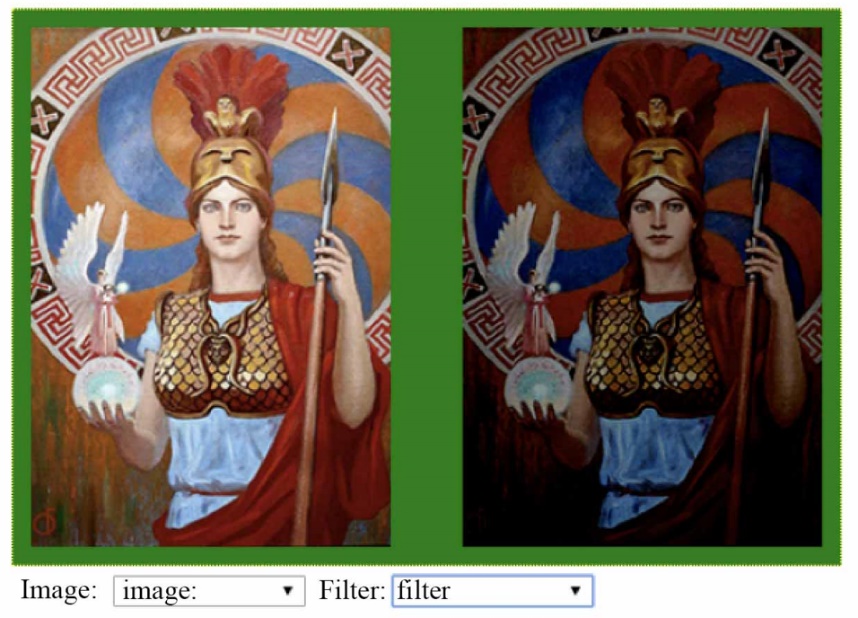
**Part 1 -** Develop the following design:

* The size of the canvas should be 460 pixels wide and 320 pixels high.
* Create a drop-down list, with the image options: Athena.jpg, theKiss.jpg, young-pearl.jpg and mona-lisa.jpg. There must be a first option called image.
* Load the images on the canvas depending on the option chosen in the drop-down list.
* The image of athena.jpg is loaded by default.



**Part 2**

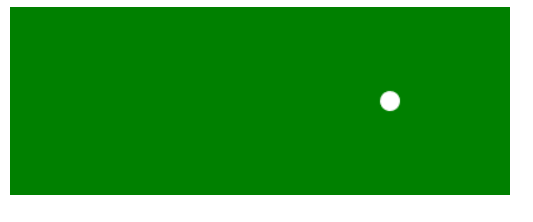
* Create a drop-down list, called filter with the options: Darken and Lighten. There must be a first option called filter.
* When choosing a specific image in the dropdown list and applying a filter, the result should appear on the right side of the canvas according to the chosen option.



# **C11: Canvas Animation (Easy)**

Draw a circular figure in a container of 320px height and 400px width using canvas. When the browser window reloads, the circular object must move from left to right continuously.





# **C12: JS Arithmetic (Difficult)**

Please create an arithmetic function, that when you enter a string of expression, it returns the calculated result. This function should include the following operators, listed by precedence:

* + 1. Parentheses: **()**
    2. Minus sign: **-**
    3. Exponentiation: **^**
    4. Multiplication, division, modulus: **\* / %**
    5. Addition, subtraction: **+ -**

***Example 1:***

The function input: **1+2\*3**

The function output: **7**

***Example 2:***

The function input: **3^2%5**

The function output: 4

***Example 3:***

The function input: **(-1-2)^3/9**

The function output: **-3**

# **C13: JS Draggable (Normal)**

Please implement Draggable function by using provided script.js, to make the parameter element draggable. Please see the provided video named video.mp4 to find draggable effect.

# **C14: Keyboard (Difficult)**

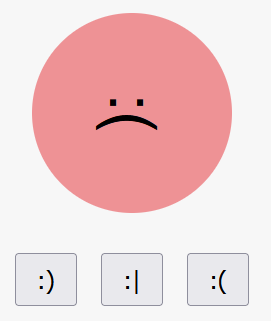
Create a text area where the text will be typed and under it a virtual keyboard on which keystrokes will be highlighted, including holding shift and stuff. Also, with the keys on the virtual keyboard, you can type text by clicking on them. Shift is turned on and off by clicking on it, while the remaining keys enter their character in the text area.

# **C15: Parallax (Normal)**

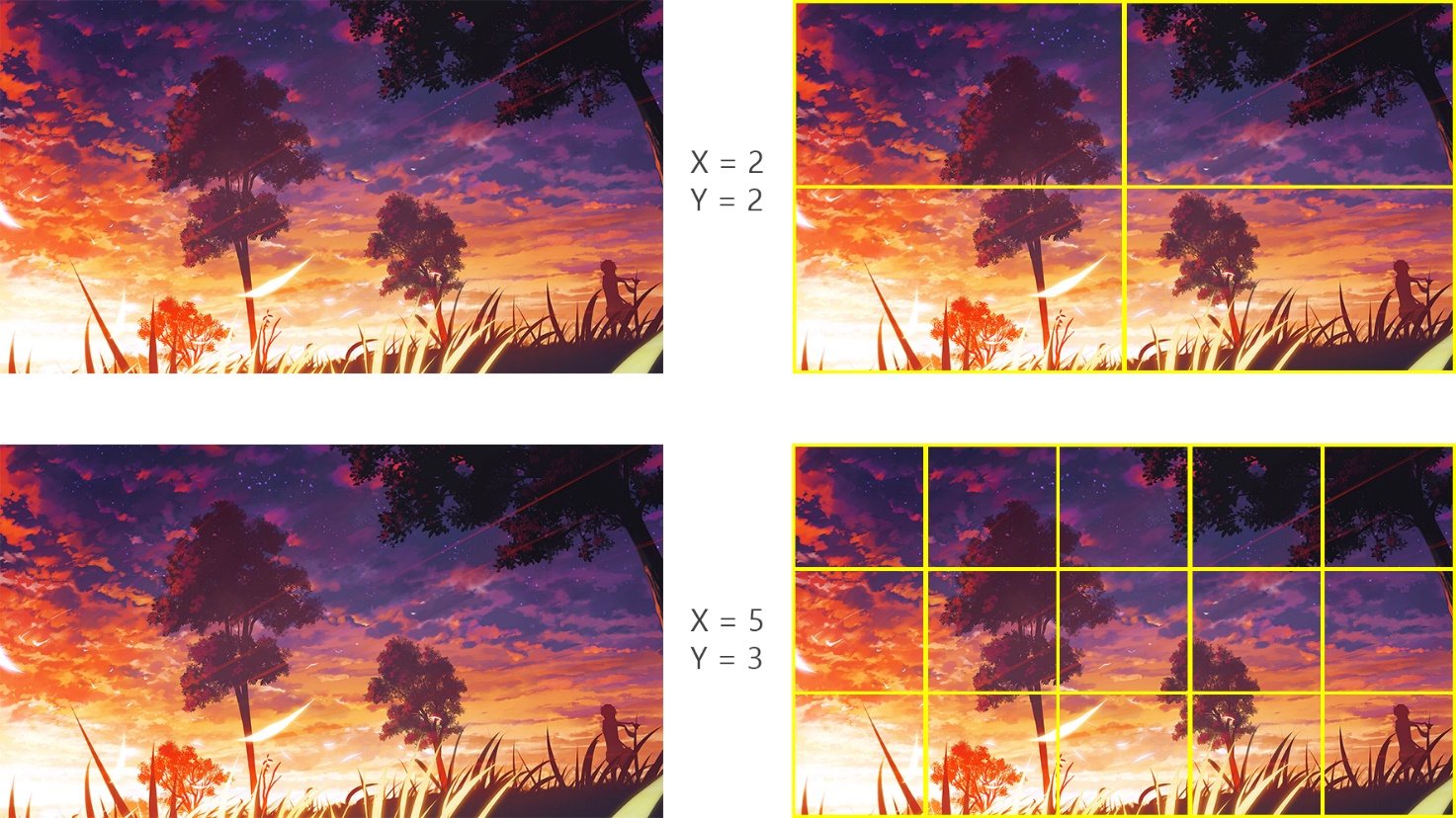
Adjust the background parallax depending on the position of the mouse cursor (the closer the objects are to the user, the faster they move).

# **C16: Smiley (Normal)**

Create a big smiley face on the centre of the page. It is 400px wide, and rectangular. Below the smiley, there are 3 buttons with texts: “**:)**”, “:|”, and” :(“. By clicking the buttons, the smiley face changes to that state with a 0.5s animation. By default, the smiley is in “:|” state. The states have these designs:



# **C17: Transition (Normal)**

You are given a picture that you need to cut into N cards and each card should disappear animated. X is 5 and Y is 3.

# **C18: Animation (Normal)**

Create the same animation that is presented in the video. A Hello world wording appears with line animation.



# **C19: Chart from data (Very Difficult)**

You should create a Chart Drawing with provided data(death-rate-by-source-from-air-pollution.csv). There are Death rates from air pollution of all countries from 1990 to 2017 in the data. Chart should calculate all data of all countries.

In the page, there should be a Chart which shows rates of 3 data (Ozone pollution, Household pollution from solid fuels, Outdoor particulate matter) from 1990 to 2017. See the examples below, but you can create your own style.

For this test, you can only use **HTML,** **CSS** and **JavaScript** (including jQuery or JS Framework).

A screenshot of a cell phone

Description automatically generated

# **C20: Video Gallery Page (Medium)**

Create Video Gallery page with provided videos. The first video width is full width of page and the others are smaller than the first one like thumbnails. All videos must have controllers with functions below

* + Play
  + Pause
  + Full screen / Normal screen(When screen is full)
  + A timeline to move to the desired moment of video
  + Mute / Volume(When volume is muted)

You can use Font awesome icons for the controller.

* + Play : <i class="fas fa-play"></i>
  + Pause : <i class="fas fa-pause"></i>
  + Full screen : <i class="fas fa-expand-arrows-alt"></i>
  + Normal screen : <i class="fas fa-compress-arrows-alt"></i>
  + Volume : <i class="fas fa-volume-up"></i>
  + Mute : <i class="fas fa-volume-mute"></i>

You must create controllers using HTML, CSS, JavaScript (including jQuery and JS Frameworks), not use original controls which video element provide.

Before a video played, the controller and the play button (centre of screen) should be shown. Since video starts play, the controller should be hide until mouse entered in the video area. When the mouse pointer entered while a video is played, the controller should be show and the controller should be hide when mouse out. When a video is not played the controller should be shown.

Video file name should be shown on the top-left of video screen.

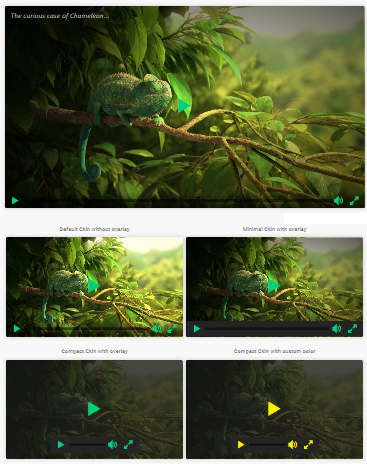
**You can create your own design to improve user interface.**

*\*Screenshots (just an example):*

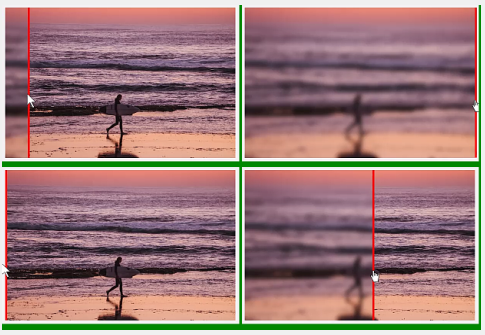
<First video and controller>



<Full page screen shop>



# **C21: Image blur linear slider (Normal)**

Using JavaScript create a linear slider that will blur an imagine progressively left and right. The required effect is presented in the media/blur\_slider\_sample.mp4. The slider should start as default from 50% of the imagine. 

# **C22: Circles (Normal)**

Create a page, that contains 5 circles. Each circle contains one of the following texts: ***Home***, ***About***, ***Contact***, ***Map,*** ***Privacy***, the texts are cantered both horizontally, and vertically. Position of the circles are random on the page, but no overlap allowed. The whole area of the circle must be visible. The page has a 20px padding. Size of the circles is random between 300-400px on each load too.

The colour of the circles are: #1F2041, #4B3F72, #417B5A, #D0CEBA, #E9D2C0.

# **C23: Square animation (Easy)**

Using the provided HTML & CSS create the JavaScript code that will animate the square box from top-left corner to bottom-right corner every time you click the Animate button.

See c23\_media.mp4

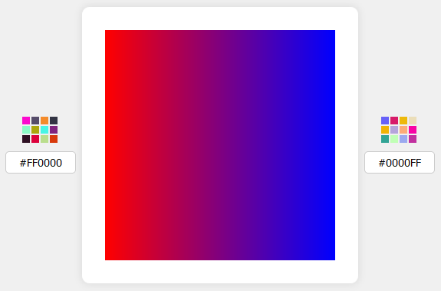
# **C24: Linear gradient (Normal)**

Create the JavaScript code for the following task:

On the centre of the screen there is a 300 px sized square. By default, it has a red to blue linear gradient background. On the left and right of the square there are 1-1 input fields where you can change the starting, and ending colour of the linear gradient, by writing in the HEX value of the colour.

Over the inputs there are 12 randomized colour buttons (10px\*10px) on each side. If you click one of the randomized colours, its colour is inserted in the input. Whenever any input is changed, and is a valid hex, that side’s linear gradient colour changes on the square.

If you manually input any valid HEX value the colour should change to the given value.



See c24\_media.mp4

# **C25: Fortune wheel (Difficult)**

You are given two versions of a fortune wheel that will spin when the **Spin** button is clicked and will stop after approximatively 5-6 seconds. See the c25\_media.mp4

Version A – The wheel is draw using JavaScript. You need to add the spinning function when the button is clicked.

Version B – The wheel is draw using CSS. You need to add the spinning function when the button is clicked.

Do not modify any HTML or CSS file just the script.js file.

# **Part D: Back-end Development (D1 – D10)**

# **D1: PHP Calendar (Difficult)**

* You can create ***index.php*** using index.html
* When you open the ***index.php***, in the top area of calendar, current month and current year should be shown and today's date should be highlighted like index.html
* If you click the left arrow button the previous month calendar should be shown.
* If you click the right arrow button the next month calendar should be shown.



# **D2: PHP Array (Easy)**

Given the code with some PHP arrays in media files, please implement a PHP function that compare the given two arrays and return a new array that contains the common elements from both given arrays.

# **D3: PHP Verification captcha (Difficult)**

Please use PHP to produce verification code image (captcha), which must comply the following rules (please see the provided images):

1. It randomly generates four characters, consisting of English Alphabet (A-Z) or digit numbers (0-9).
2. The four characters must be slightly rotated.
3. The four characters must not be located at the same row.
4. The image contains at least 3 random lines.
5. At least 3 mixed noise points.

# **D4: Watermark (Easy)**

Put a watermark (.png file) on the image provided. The watermark should be in the lower right corner.

# **D5: Predominant colours (Easy)**

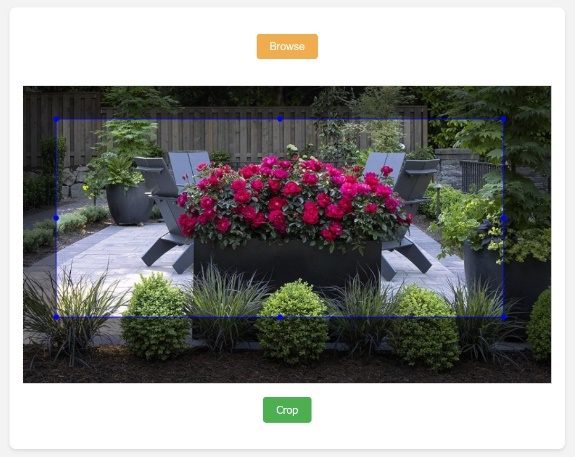
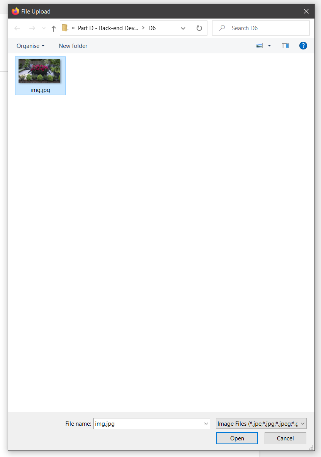
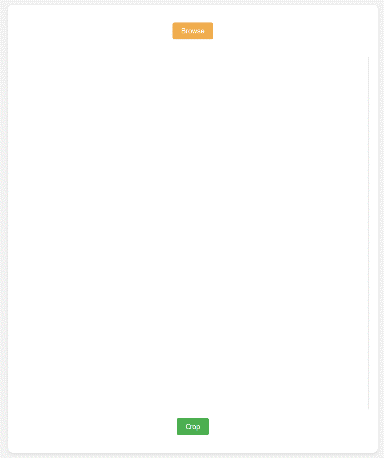
Identify the 3 predominant colours in the provided image and show them.

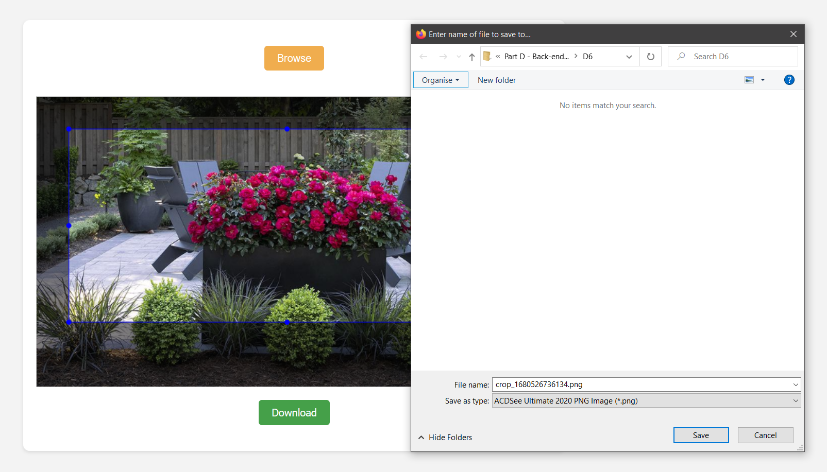
# **D6: Image Cropper (Normal)**

Create an application which can upload an image and crop to save and download.

* Upload an image on the server
* Show the uploaded image on the screen.
* Click [crop] button to crop the selected rectangle area and cropped image saved with prefix “crop\_”.
* After crop complete the [crop] button should be changed to [download] button.
* When click [download] button, user can download cropped image.

*\*See below screenshots.*





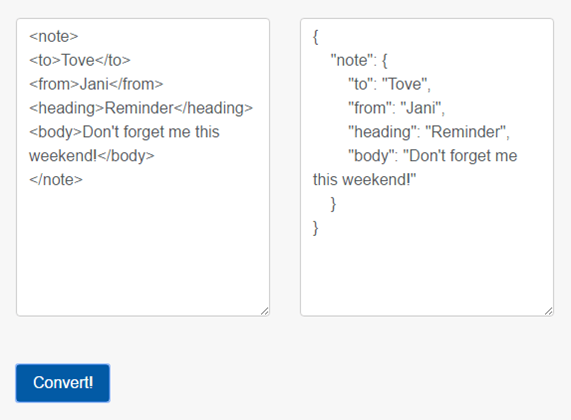
# **D7: A Pagination Code (Normal)**

* After reading data.json, the data in the current page must be shown..
* 5 data must be shown in one page.
* 5-page links to move to another page must be shown in one page
* Create buttons to move 5 pages at once, previous and next.
* Page link of current page must be located in the centre
* Take a look at the provided screen shots and data.json.
* You can only use HTML & CSS & PHP, but for reading json data and the pagination code must use only PHP.

***\*You can use bootstrap. JavaScript is not allowed.***

# **D8: XML2JSON Converter (Difficult)**

Create a simple xml to json convertor using pure PHP(See the example.xml).

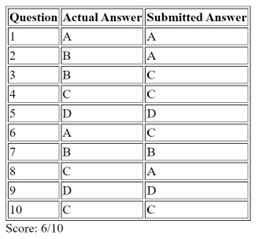


# **D9: Remove duplicates (Easy)**

Given an array, remove the duplicate values, and show the new array;

# **D10: Answer checker (Easy)**

You are provided with two CSV files. One file contains the actual answers, and the other file contains the submitted answers. You are expected to develop a web page, displaying a table showing the question number, actual answer and submitted answer. At the bottom of the table, display the scores for this submission. Eg, if there are 8 correct answers and there are 10 questions. Display the score as “8/10”.



# **D11: Tic-Tac-Toe (Difficult)**

Create a Tic-Tac-Toe game, according to the following requirements:

* When you click a cell, your piece “X” is placed in the cell, and the robot who operating the “O” will place a piece randomly.
* If you click the reset button, the game will be initialized, the win prompt will be hidden.
* The win prompt will be hidden at initial status.
* If you refresh the page, game status should not be initialized.
* All data should be stored in the backend.
* The page data should be taken from backend asynchronously.

Calendar

Description automatically generated

# **D12: RGB to HEX (Medium)**

Given three integers between 0 and 255, corresponding to the red, green, and blue channel values of a colour, return the hex string for that colour.

hexcolor(255, 99, 71) => "#FF6347"