ICT Berufsbildung
Formation professionnelle
Formazione professionale

Frontend Development Module

TASKS	POINTS
Image Editor	35 POINTS

This module contains HTML/CSS (Task 1) and JavaScript parts (Task 2). Those parts can be done independently and in any order, meaning you don't have to the the HTML/CSS task first to be able to do the JavaScript task.

Task 1 - Image Editor Design

The goal of the task is to implement the design of an *image editor*. The HTML of the user interface is already provided and cannot be changed. Your goal is to implement the design in the file `work/frontend/src/ui.css`.

To be more efficient, it is recommended to reuse your code as much as possible, however, you are not technically required to do so.

General Information

In this part, you will prove your CSS skills. Therefore, you will modify one CSS file named `ui.css` which is located in the folder `work/frontend/src/`.

You can only change or create new files under `work/frontend/src/`. All other files will be reverted for the review process.

Requirements

Somehow, your boss got rid of the css file of the image editor. Don't ask... Now you have to implement the design of the image editor, because the dev that previously worked on the project is on vacation. To help you out a little bit, on her flight to the beach, she scribbled the spacings in pixels on screenshots of the app and sent them to you.

There are two different views of the image editor:

- The intro view (desktop | mobile) allows the user to upload an image.
- The app view (desktop | mobile) allows the user to select filters to be applied to the image. It also shows a before and after view of the image.

The mobile view should be responsive up to 768px width. Starting at 768px width, the desktop view should be used.

Since this is a proof of concept, only black and white images are supported.

Fonts

Should be applied to `<body>`

Family	Inter, sans-serif
Size	16px
Color	#000535
Weight	300

Special Typography Styles

h1	size: 40px, weight: 500, letter-spacing: 0.04em
intro p	size: 20px
image-wrapper p	size: 16px, uppercase, letter-spacing: 0.04em
image-wrapper p (first letter)	size: 1.4em

Additional requirements

1. `.filters` and `.images` should be full width of `#app` on mobile and half the width of `#app` on desktop (minus spacing).
2. All `li` elements in `.filters-list` should have a 2px solid #dddddd border.

3. The first and the last `li` elements in `.filters-list` should have 8px border radius.
4. `.add-filter`, as well as any other button or text input field, should have a `#f5f5f5` background on hover and `#ffffff` if it is not being hovered.
5. `.filter svg` should have the color #d70000 and #a70000 on hover.
6. `#intro p` should not exceed a width of 480px on desktop

You can use the image `lib/img/mountain.png` to test your app. It will also be used when marking.

Working Directory

The working directory is located in `work/frontend/src/`.

You may only change the files and add all the CSS styles within that folder.

Also make sure to check the header of each file for notes on what you may or may not change.

For reviewing your work, unit tests are used. So be sure that you are executing the unit tests at some point to verify your progress.

Please note that these behaviors are considered as cheating and will give you 0 points for the whole frontend task:

- It is forbidden to change the content of the test files.

Expected Result

You have implemented all the styles in `work/frontend/src/ui.css` according to the design, and all tests in the test suite `ui.cy.js` are passing.

Running the Tests

All tests are located in `work/frontend/cypress/integration`. You can look at them to see what is expected to happen but you are not allowed to change them in any way.

You may run the tests in two ways:

Interactively - `npm start`

- Starts an **interactive test runner** that let's you select which tests to run and gives you extensive feedback on the tests' results.
- Wait for cypress to start (it might take a bit of time initially)
- Select the testsuite for the module you want to test
- A chrome window will open and the tests will run

- Click the "Run all tests"-Button or reload the page to rerun the test
- Select or hover over a test or assertion in the left sidebar to see the website at that specific point in time

In your Terminal - `npm test`

- Runs all the tests in your terminal, gives you feedback on their results and let's you know **how many points you have earned**.
- `npm test -- --spec "cypress/e2e/ui.cy.js"` allows you to only run the specified test file.

Task 2 - Image Editor Functionality

The goal of the task is to implement the functionality of the *image editor*. Some of the functionality the previous dev already implemented, but you have to finish it. The HTML of the user interface is already provided and cannot be changed. Your goal is to implement the functionality in the file `work/frontend/src/task.js` .

To be more efficient, it is recommended to reuse your code as much as possible, however, you are not technically required to do so.

General Information

In this part, you will prove your JavaScript programming skills. Therefore, you will modify one JavaScript file named `task.js` which is located in the folder `work/frontend/src/`.

> You can only change or create new files under `work/frontend/src/`. All other files will be reverted for the review process.

Requirements

The dev that had previously worked on the project is on vacation and did not get to finish the functionality of the image editor. He left you the empty functions that still need to be implemented in the file `work/frontend/src/task.js`. Make sure to implement them according to their documentation right above the function.

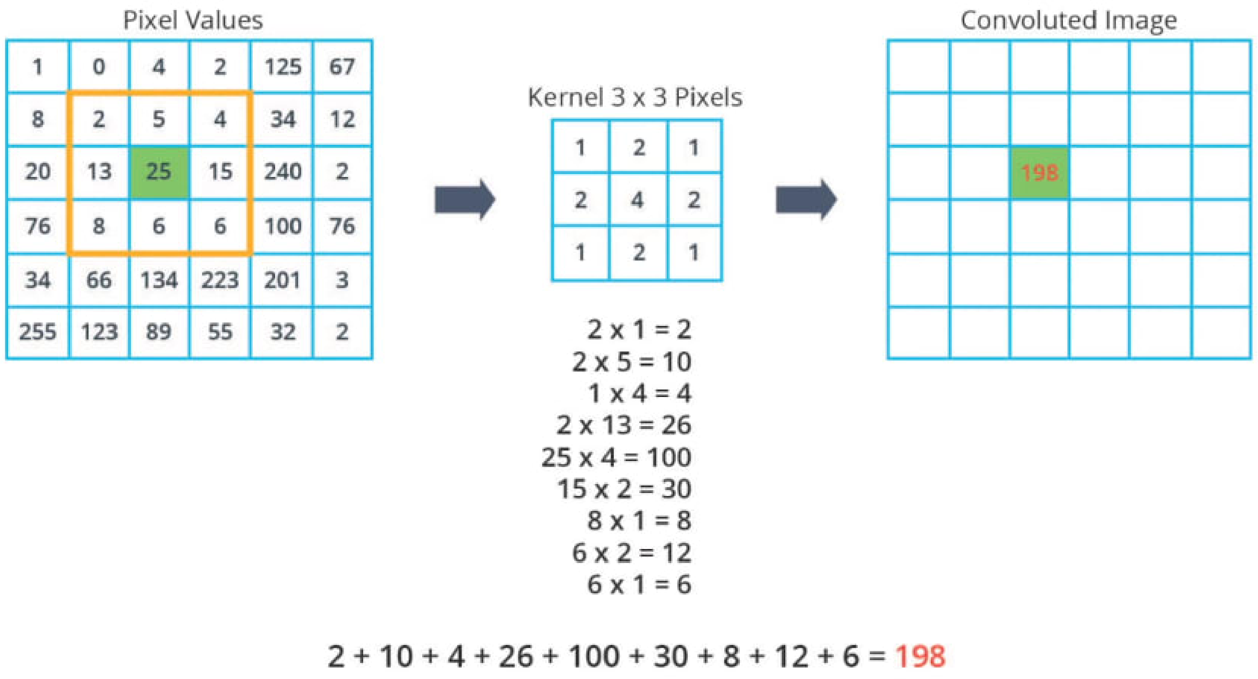
Additional Information for `applyKernel`

The `applyKernel` function takes two arguments:

1. `imageData` - A two-dimensional array of numbers spanning from 0 to 255 representing the image data (where 0 is black and 255 is white).
2. `kernel` - A two-dimensional array of numbers representing the kernel to be applied to the image data.

The `applyKernel` function should return a new two-dimensional array of numbers spanning from 0 to 255 representing the image data after applying the kernel.

"Applying the kernel" means that for each pixel in the image data, the kernel is applied to the pixel and its surrounding pixels and the sum of the products of the kernel and the pixels is calculated and stored in the new image data. The following image should give you and idea of how the kernel is applied to the image. Keep in mind, that a kernel can be of any size.



1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

1	2	1
2	4	2
1	2	1

2 x 1 = 2
2 x 5 = 10
1 x 4 = 4
2 x 13 = 26
25 x 4 = 100
15 x 2 = 30
8 x 1 = 8
6 x 2 = 12
6 x 1 = 6

2 + 10 + 4 + 26 + 100 + 30 + 8 + 12 + 6 = 198

Working Directory

The working directory is located in `work/frontend/src/`.

You may only change the files and add all the JavaScript functionality within that folder.

Also make sure to check the header of each file for notes on what you may or may not change.

You are **not** allowed to install additional NPM packages.

For reviewing your work, unit tests are used. So be sure that you are executing the unit tests at some point to verify your progress.

Please note that these behaviors are considered as cheating and will give you 0 points for the whole frontend task:

- It is forbidden to change the content of the test files.
- It is forbidden to return static results (only make it work for the exact same situations that the one described in the tests).

Expected Result

You have implemented all the methods marked with `@todo` in `work/frontend/src/task.js`, and all tests in the test suite `task.cy.js` are passing.

Running the Tests

All tests are located in `work/frontend/cypress/integration`. You can look at them to see what is expected to happen but you are not allowed to change them in any way.

You may run the tests in two ways:

Interactively - `npm start`

- Starts an **interactive test runner** that let's you select which tests to run and gives you extensive feedback on the tests' results.
- Wait for cypress to start (it might take a bit of time initially)
- Select the testsuite for the module you want to test
- A chrome window will open and the tests will run
- Click the "Run all tests"-Button or reload the page to rerun the test
- Select or hover over a test or assertion in the left sidebar to see the website at that specific point in time

In your Terminal - `npm test`

- Runs all the tests in your terminal, gives you feedback on their results and let's you know **how many points you have earned**.
- `npm test -- --spec "cypress/e2e/task.cy.js"` allows you to only run the specified test file.