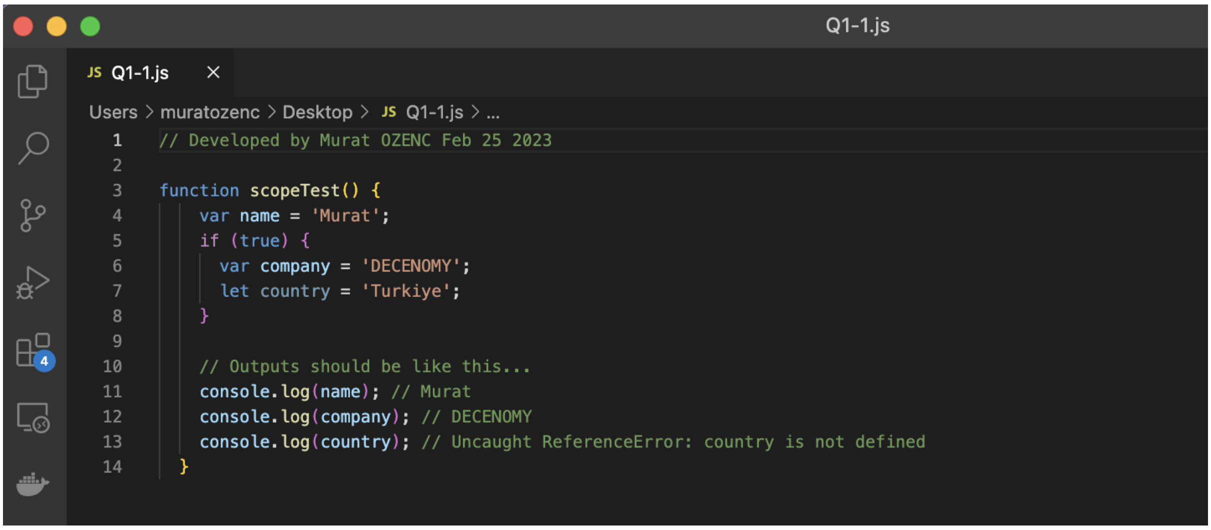
DECENOMY Skill Test Questions & Answers by Murat ÖZENÇ

Q1 : Explain the difference between "let" and "var".

A1 : In the first example below, you can clearly see that there is one ‘let’ and two ‘var’ defined in the function, one of ‘var’s in curly brackets. In the end we can also see that when we call them separately, the variable defined by using ‘let’ is undefined, because the scope of ‘var’ is the all function and the scope of ‘let’ is the curly brackets.

In the second example below, you can see that you can make assignments multiple times for ‘var’ but not for ‘let’.

To put it in short, ‘var’ has a function scope, which means that a variable declared with ‘var’ is accessible throughout the entire function in which it is declared. On the other hand, ‘let’ has a block scope, which means that a variable declared with ‘let’ is accessible only within the block in which it is declared (in example 1, {} that enclose the variable declaration).

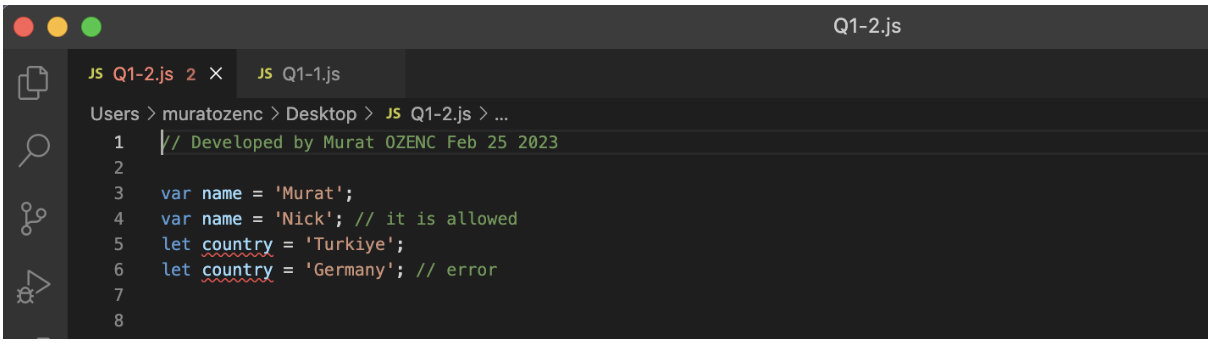


```javascript
// Developed by Murat OZENC Feb 25 2023

function scopeTest() {
    var name = 'Murat';
    if (true) {
      var company = 'DECENOMY';
      let country = 'Turkiye';
    }

    // Outputs should be like this...
    console.log(name); // Murat
    console.log(company); // DECENOMY
    console.log(country); // Uncaught ReferenceError: country is not defined
  }
```

First Example Screenshot for Q1 (You can find the file as attached as Q1-1.js)



```javascript
// Developed by Murat OZENC Feb 25 2023

var name = 'Murat';
var name = 'Nick'; // it is allowed
let country = 'Turkiye';
let country = 'Germany'; // error
```

Second Example Screenshot for Q1 (You can find the file as attached as Q1-2.js)

DECENOMY Skill Test Questions & Answers by Murat ÖZENÇ

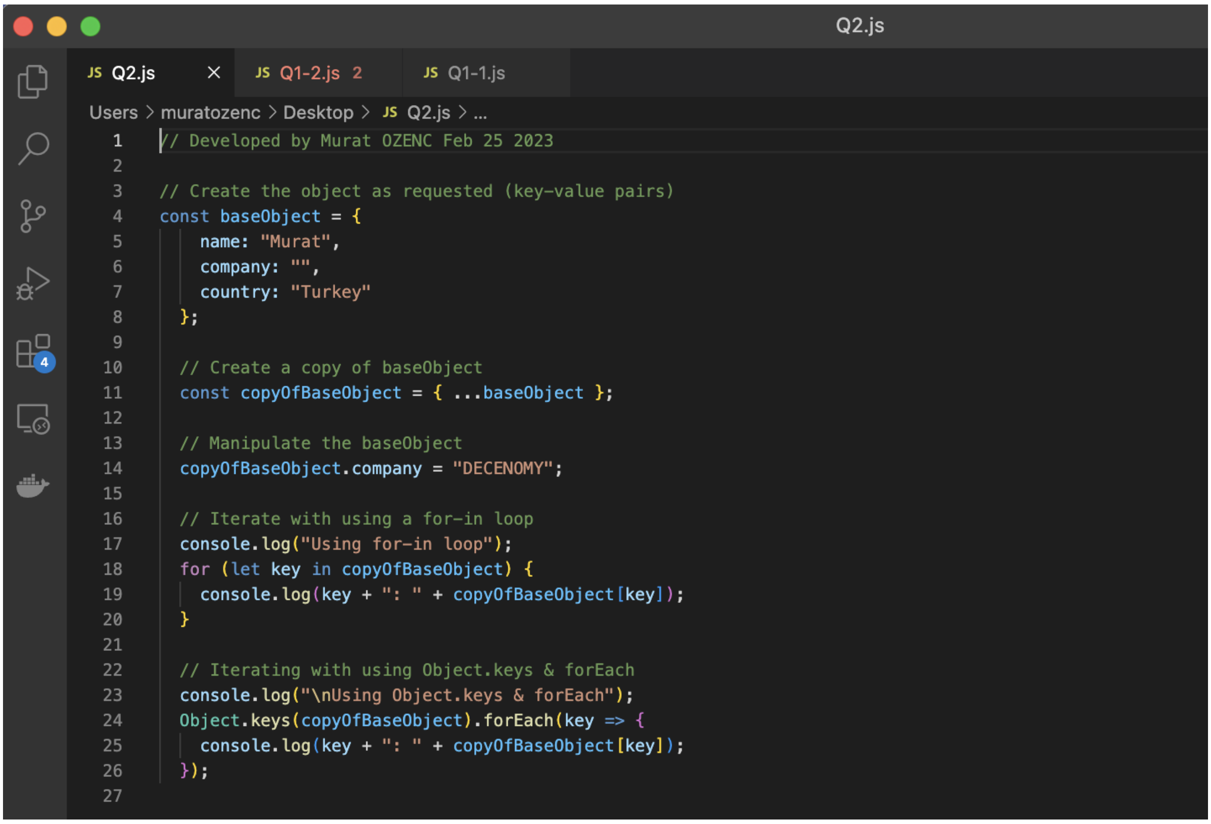
Q2 : Using modern vanilla JavaScript only, create an Object with some textual key-value properties, create a copy of that object, modify some of its properties and then iterate through this object while outputting the key-value names.

Bonus: use more than one iteration approach and explain benefits of one over the other.

A2 : Two methods have been coded as you can see below. Both of them have their own advantages.

The advantage of Object.keys & forEach is more easy to understand and simple if you are looping only with keys.

The advantage of for-in is that you can loop all of the properties, including inherited ones. Also you can iterate all iterables, including objects, maps and arrays.



```
// Developed by Murat OZENC Feb 25 2023

// Create the object as requested (key-value pairs)
const baseObject = {
  name: "Murat",
  company: "",
  country: "Turkey"
 };

 // Create a copy of baseObject
 const copyOfBaseObject = { ...baseObject };

 // Manipulate the baseObject
 copyOfBaseObject.company = "DECENOMY";

 // Iterate with using a for-in loop
 console.log("Using for-in loop");
 for (let key in copyOfBaseObject) {
  console.log(key + ": " + copyOfBaseObject[key]);
 }

 // Iterating with using Object.keys & forEach
 console.log("\nUsing Object.keys & forEach");
 Object.keys(copyOfBaseObject).forEach(key => {
  console.log(key + ": " + copyOfBaseObject[key]);
 });
```

Screenshot (You can find the file as attached as Q2.js)

——------------------——-------------------------——--------------------------——-------------------------——----------

Q3 : Explain why some JavaScripts are inserted in the `<head>` section, and others at the end of the `<body>` section at the page. Explain upsides and downsides of each approach.

Bonus: explain what "async" HTML attribute do, and when it is safe to use it.

A3 : JavaScript codes placed in the '<head>' are loaded before the page is rendered. In this way, the page can run smoothly as the variables and functions required for the page to run smoothly are loaded first. The disadvantage of this approach is that pages with excessive JavaScript code can slow down the page and create a bad user experience, as it may take time for all the code to load.

JavaScript codes placed in the '<body>' section are executed after the page content is loaded. In this way, JavaScript codes run after the user sees the page content. The disadvantage of this approach is that the user may experience delay due to having to wait for the JavaScript codes required for the page to work properly.

So, there are two ways to resolve this dilemma. First one is placing the JavaScripts which are required for proper page rendering to the '<head>' section and placing the rest to the end of the '<body>' tag. The second one is using the 'async' HTML attribute, which is a better way.

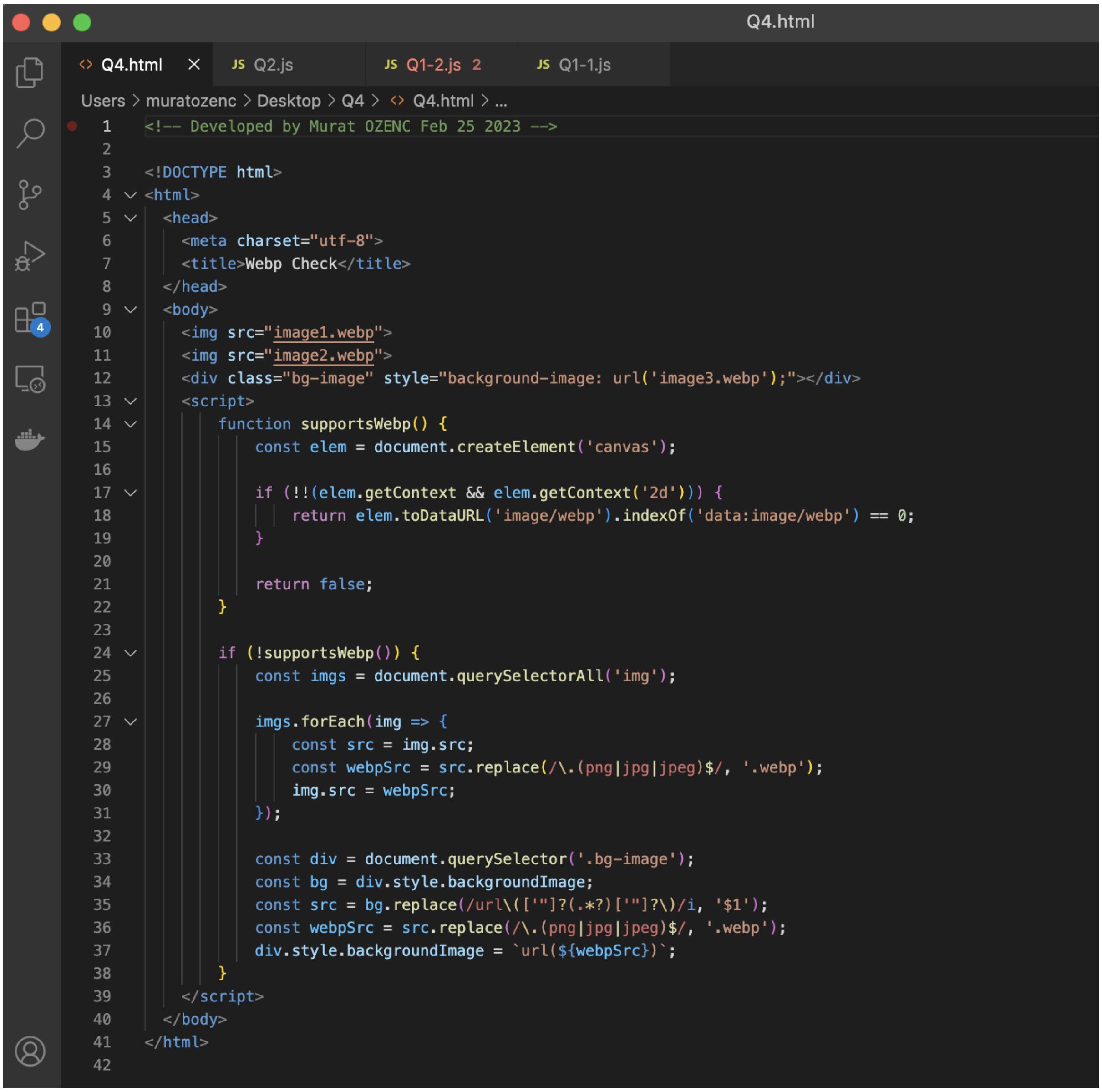
The 'Async' attribute allows JavaScript to load asynchronously. When this attribute is used, the browser causes the rest of the page to be loaded in the background, along with any code that uses this tag. This can result in a faster page load time since the browser does not need to wait for the JavaScript code to be loaded before rendering the page.

Using the 'async' attribute is very useful as long as we fully know the dependencies of all javascript files, otherwise it may not be safe for the system to function properly.

DECENOMY Skill Test Questions & Answers by Murat ÖZENÇ

Q4 : Get three random images (some JPEGs and some PNGs - as you please), compress them and save them in WebP format. Then create a simple HTML page with two of those images used as `<img...>`, and one as a `<div>` element with a background image. Then - without using the "srcset" attribute - write a simple vanilla JavaScript that will detect if a browser running your page supports WebP format, and if not - automatically replace all the images with their originals.

A4 : This question has been developed and the screenshot is given below. I think this is the tricky one because during my research I saw that this code block may not work 100% in all browsers, I could not have a chance to test it in all.



```
<!-- Developed by Murat OZENC Feb 25 2023 -->

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Webp Check</title>
  </head>
  <body>
    <img src="image1.webp">
    <img src="image2.webp">
    <div class="bg-image" style="background-image: url('image3.webp');"></div>
    <script>
      function supportsWebp() {
        const elem = document.createElement('canvas');

        if (!!(elem.getContext && elem.getContext('2d'))) {
          return elem.toDataURL('image/webp').indexOf('data:image/webp') == 0;
        }

        return false;
      }

      if (!supportsWebp()) {
        const imgs = document.querySelectorAll('img');

        imgs.forEach(img => {
          const src = img.src;
          const webpSrc = src.replace(/\.(png|jpg|jpeg)$/, '.webp');
          img.src = webpSrc;
        });

        const div = document.querySelector('.bg-image');
        const bg = div.style.backgroundImage;
        const src = bg.replace(/url\(['"]?(.*?)['"]?\)/i, '$1');
        const webpSrc = src.replace(/\.(png|jpg|jpeg)$/, '.webp');
        div.style.backgroundImage = `url(${webpSrc})`;
      }
    </script>
  </body>
</html>

```

Screenshot (You can find the file as attached as Q4.html inside folder called "Q4")

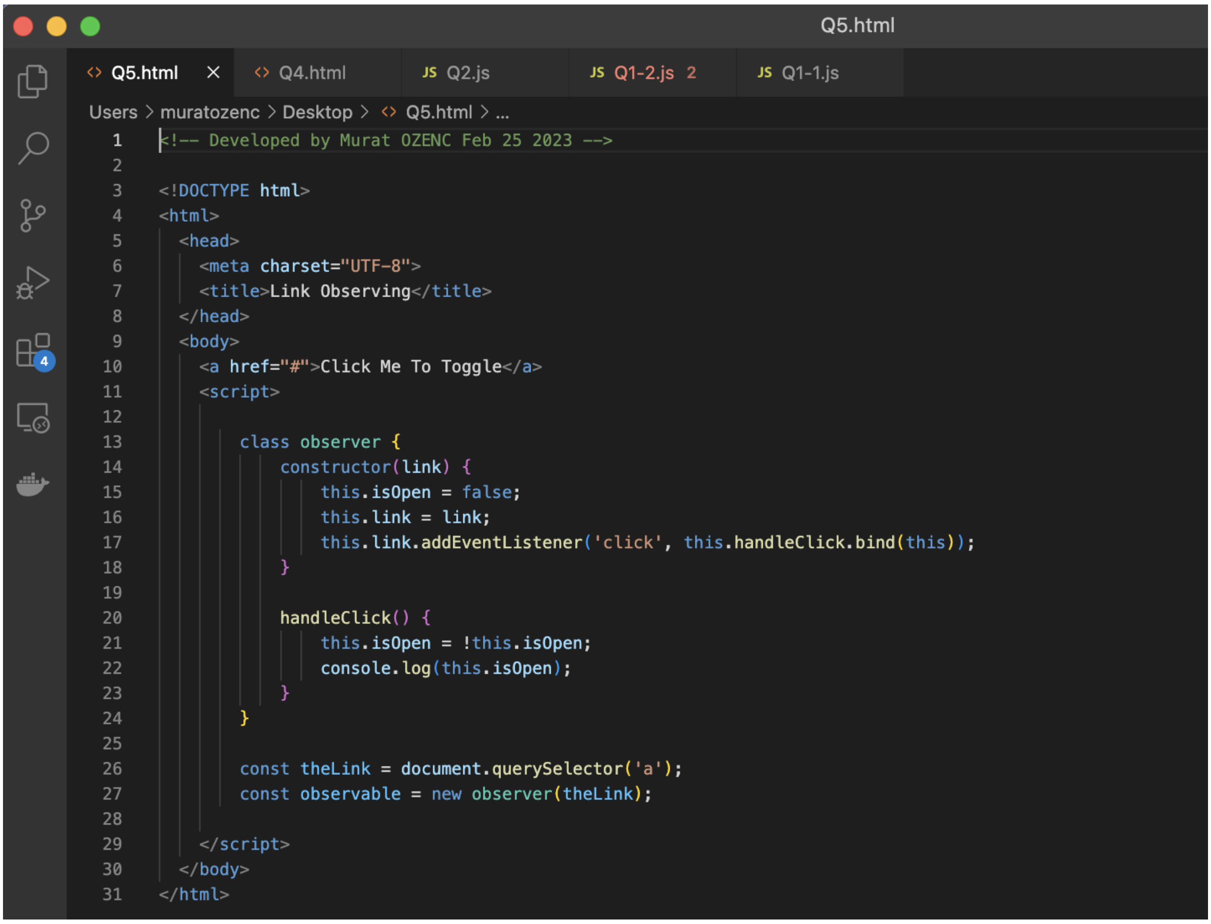
———------------------———------------------———------------------———------------------———----------

DECENOMY Skill Test Questions & Answers by Murat ÖZENÇ

Q5 : Write a simple HTML file with a single `<a...>` element, then write a JavaScript class which in its constructor instantiates "isOpen" public class variable, observes the "click" event on said element and applies a callback function to it (class, or anonymous - up to you). Then - modify (toggle) the "isOpen" class variable within said callback function and log its value. Then create a new object of your class in your HTML file.

Bonus: Explain how did you preserve "this" reference and why did you do it the way you did.

A5 : Please see the screenshot of the code developed below. By using the bind() method in our event listener, we ensure that this always refers to the instance of our class, even when it's called within the context of the link element. This is important because this can change depending on how a function is called in JavaScript, and we want to make sure we're always referring to the correct instance of the class.



```html
<!-- Developed by Murat OZENC Feb 25 2023 -->

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Link Observing</title>
  </head>
  <body>
    <a href="#">Click Me To Toggle</a>
    <script>

        class observer {
            constructor(link) {
                this.isOpen = false;
                this.link = link;
                this.link.addEventListener('click', this.handleClick.bind(this));
            }

            handleClick() {
                this.isOpen = !this.isOpen;
                console.log(this.isOpen);
            }
        }

        const theLink = document.querySelector('a');
        const observable = new observer(theLink);

    </script>
  </body>
</html>
```

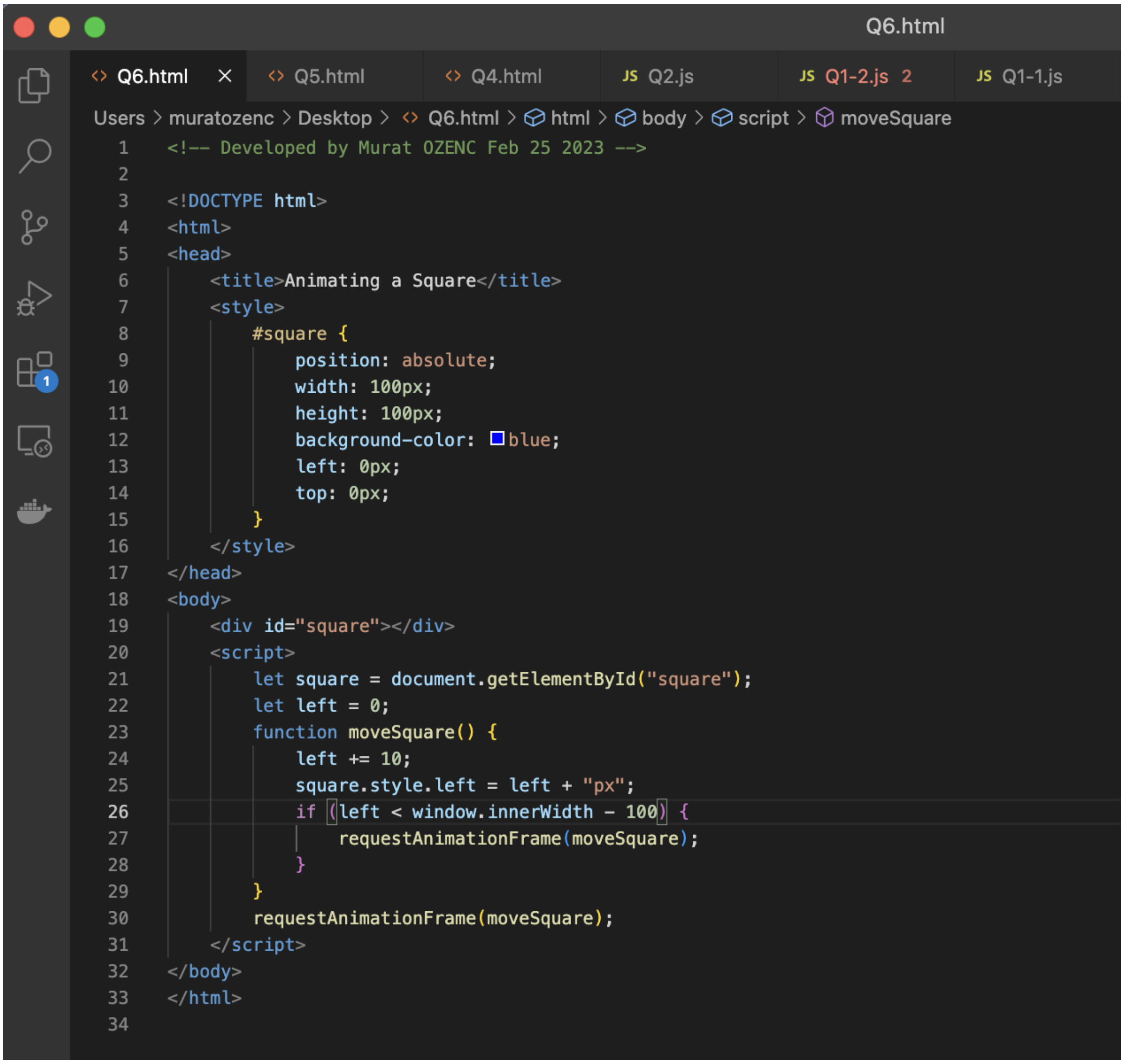
Screenshot (You can find the file as attached as Q5.html)

——-------------------——-------------------——-------------------——-------------------——--------------------——----------

DECENOMY Skill Test Questions & Answers by Murat ÖZENÇ

Q6 : Write a simple HTML file with some random graphical object in it (can be a square, colored `<div>`) and animate its movement from one side of the screen to the other. Use the most performant animation technique available in modern JavaScript and explain what is it.

A6 : You can see the screenshot of the code written below. 'requestAnimationFrame' is the most efficient animation technique in modern JavaScript because the animation is synchronized with the browser's rendering process, which minimizes stuttering. On the other hand, the energy consumption of this method is much better than setInterval and setTimeout. This gives an advantage of mobile web development.


Q6.html
Q6.html
Q5.html
Q4.html
Q2.js
Q1-2.js 2
Q1-1.js
Users > muratozenc > Desktop > Q6.html > html > body > script > moveSquare

```html
<!-- Developed by Murat OZENC Feb 25 2023 -->

<!DOCTYPE html>
<html>
<head>
    <title>Animating a Square</title>
    <style>
        #square {
            position: absolute;
            width: 100px;
            height: 100px;
            background-color: blue;
            left: 0px;
            top: 0px;
        }
    </style>
</head>
<body>
    <div id="square"></div>
    <script>
        let square = document.getElementById("square");
        let left = 0;
        function moveSquare() {
            left += 10;
            square.style.left = left + "px";
            if (left < window.innerWidth - 100) {
                requestAnimationFrame(moveSquare);
            }
        }
        requestAnimationFrame(moveSquare);
    </script>
</body>
</html>

```

Screenshot (You can find the file as attached as Q6.html)