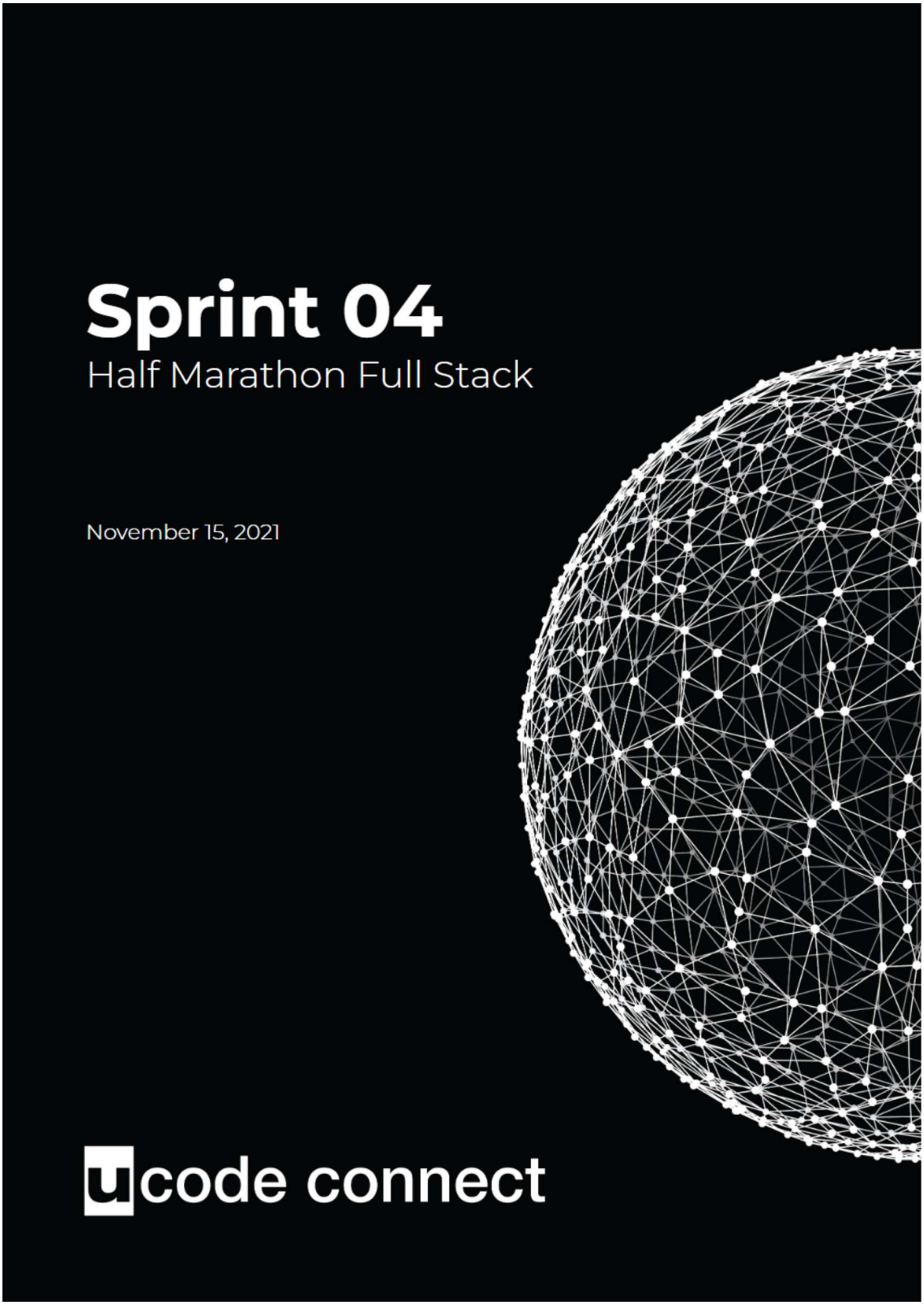

Sprint 04
Half Marathon Full Stack
November 15, 2021
ucode connect

Contents

ucode connect

Engage

DESCRIPTION

You have now worked with both HTML and JavaScript. Even though you know how to include your JS scripts into HTML files, you haven't yet done any actual interaction between the two. It's time to find out how to affect the elements of your HTML web page using JS.

In this Sprint you will be working with the Document Object Model (DOM). It's basically a programming interface for HTML and XML documents. By using DOM, it becomes possible to connect to, modify, and control web pages using programming languages.

You will learn how DOM works, and how to use JS to manipulate you web page.

BIG IDEA

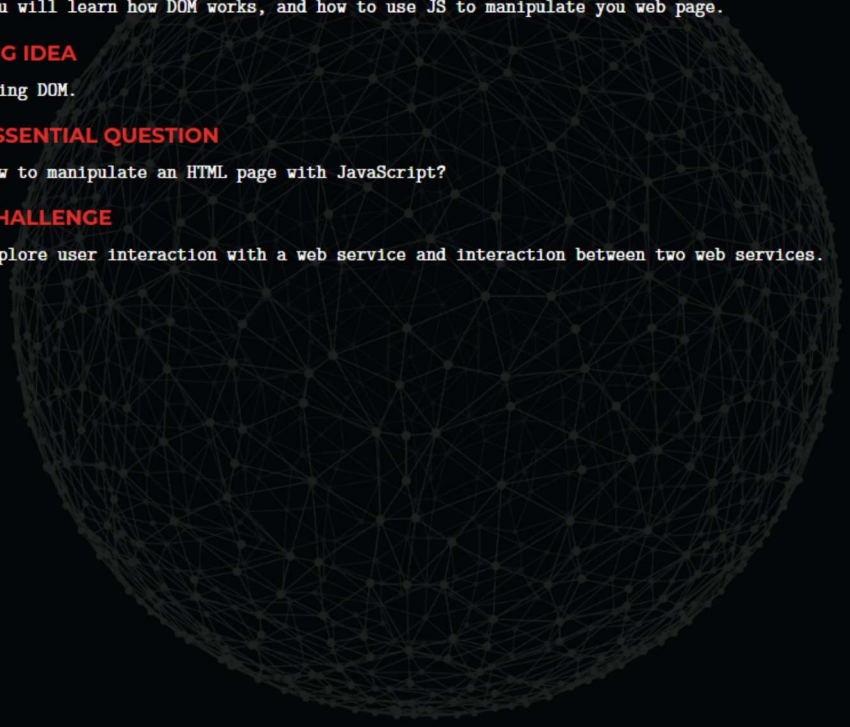
Using DOM.

ESSENTIAL QUESTION

How to manipulate an HTML page with JavaScript?

CHALLENGE

Explore user interaction with a web service and interaction between two web services.



ucode connect

Sprint 04 | Half Marathon Full Stack **> 2**

Investigate

GUIDING QUESTIONS

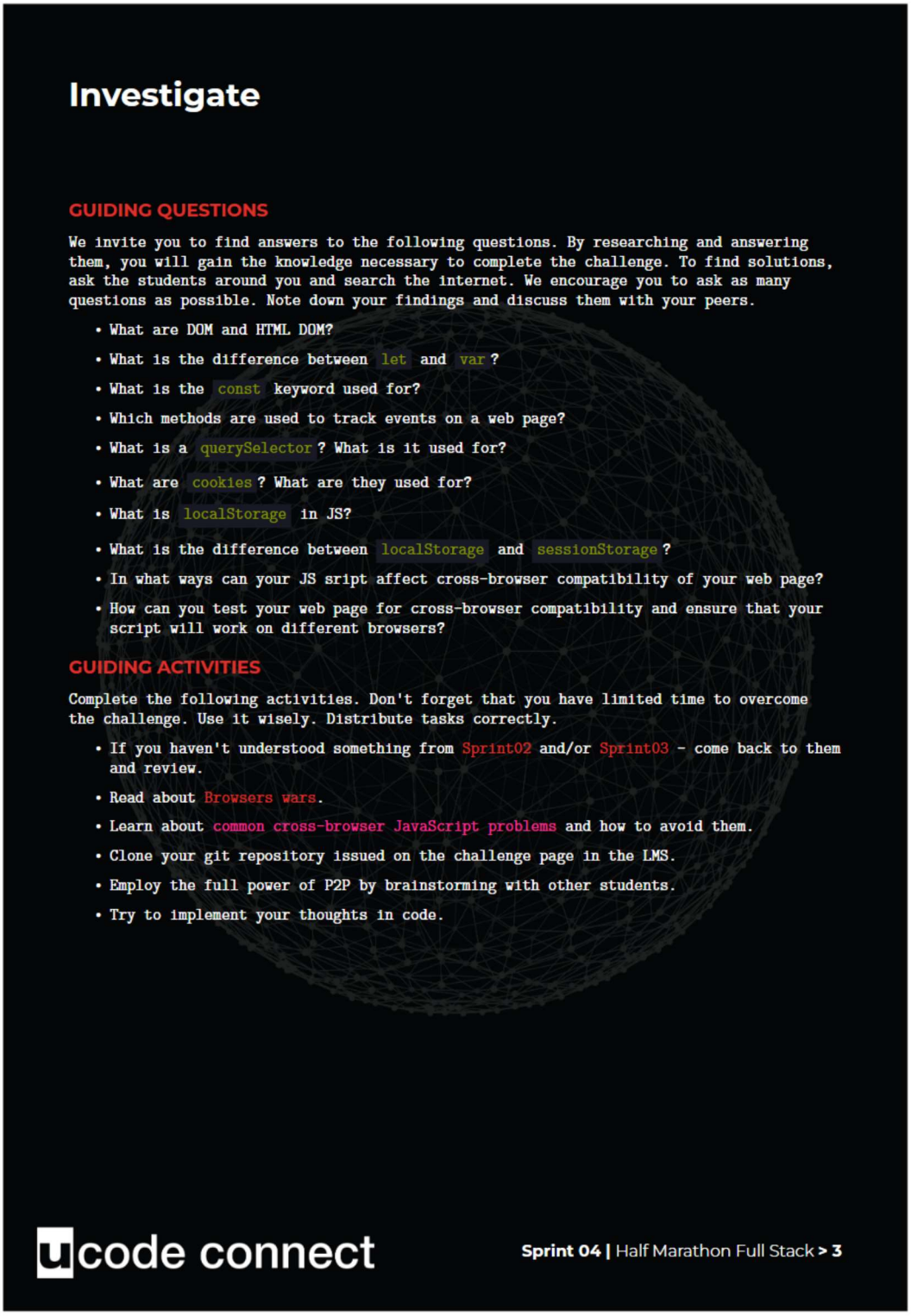
We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find solutions, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What are DOM and HTML DOM?
- What is the difference between `let` and `var`?
- What is the `const` keyword used for?
- Which methods are used to track events on a web page?
- What is a `querySelector`? What is it used for?
- What are `cookies`? What are they used for?
- What is `localStorage` in JS?
- What is the difference between `localStorage` and `sessionStorage`?
- In what ways can your JS sript affect cross-browser compatibility of your web page?
- How can you test your web page for cross-browser compatibility and ensure that your script will work on different browsers?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

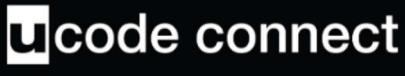
- If you haven't understood something from Sprint02 and/or Sprint03 - come back to them and review.
- Read about Browsers wars.
- Learn about common cross-browser JavaScript problems and how to avoid them.
- Clone your git repository issued on the challenge page in the LMS.
- Employ the full power of P2P by brainstorming with other students.
- Try to implement your thoughts in code.



ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- All tasks are divided into Act Basic and Act Advanced. This means that the complexity of the tasks increases gradually. Try to complete all tasks to get maximum points and more knowledge.
- Analyze all information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.
- Submit only the specified files in the required directory and nothing else. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- The web page in the browser must open through `index.html`.
- The scripts must be written outside the HTML file - in a separate JS file (script.js).
- You can always use the `Console` panel to test and catch errors.
- Complete tasks according to the rules specified in the following style guides:
  - HTML and CSS: Google HTML/CSS Style Guide. As per section 3.1.7 Optional Tags, it doesn't apply. Do not omit optional tags, such as `<head>` or `<body>`
  - JavaScript:
    * JavaScript Style Guide and Coding Conventions
    * JavaScript Best Practices
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Your work may also be graded by your mentor. So, be ready for that.
- Also, the challenge will pass automatic evaluation which is called `Oracle`.
- If you have any questions or don't understand something, ask other students or just Google it.



Act Basic: Task 00

NAME

Secret lab

DIRECTORY

`t00_secret_lab/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

DOM

DESCRIPTION

Create a function that changes the content and style of a web page. Use the HTML and CSS files available in the SYNOPSIS.

As you can see, on click of the button, the function `transformation()` is being called. Create this function inside your `script.js` file.

The function switches between two states of the web page.
State 1:

- displays `Bruce Banner` with `60px` font size and `2px` letter spacing
- has a `#ffb300` background color

State 2:

- displays `Hulk` with `130px` font size and `6px` letter spacing
- has a `#70964b` background color

It must be possible to keep endlessly switching between the two states.

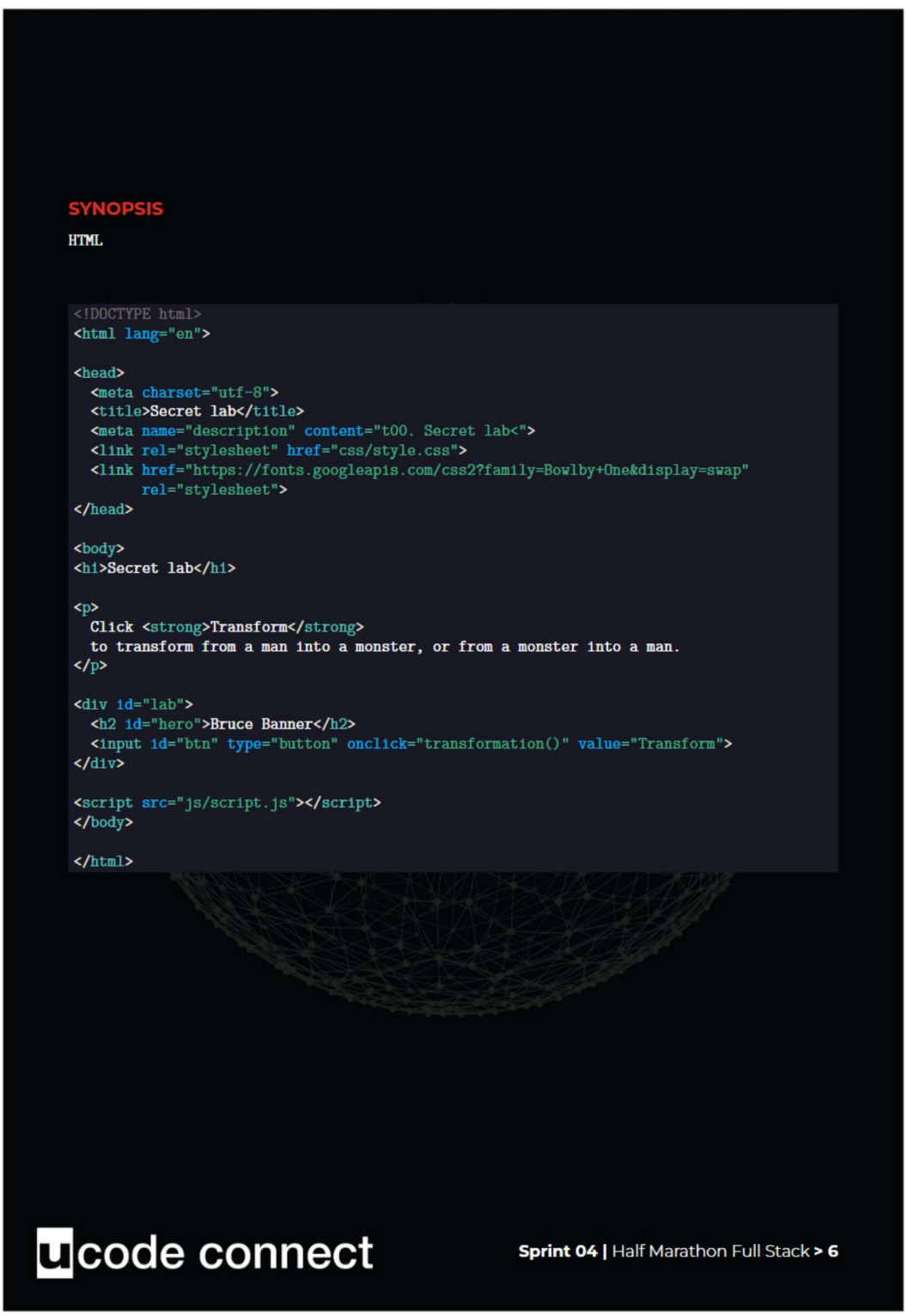
ucode connect

Sprint 04 | Half Marathon Full Stack **> 5**

SYNOPSIS

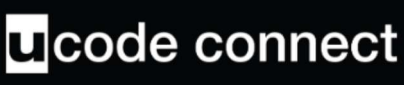
HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Secret lab</title>
  <meta name="description" content="t00. Secret lab<">
  <link rel="stylesheet" href="css/style.css">
  <link href="https://fonts.googleapis.com/css2?family=Bowlby+One&display=swap"
        rel="stylesheet">
</head>

<body>
<h1>Secret lab</h1>

<p>
  Click <strong>Transform</strong>
  to transform from a man into a monster, or from a monster into a man.
</p>

<div id="lab">
  <h2 id="hero">Bruce Banner</h2>
  <input id="btn" type="button" onclick="transformation()" value="Transform">
</div>

<script src="js/script.js"></script>
</body>

</html>
```



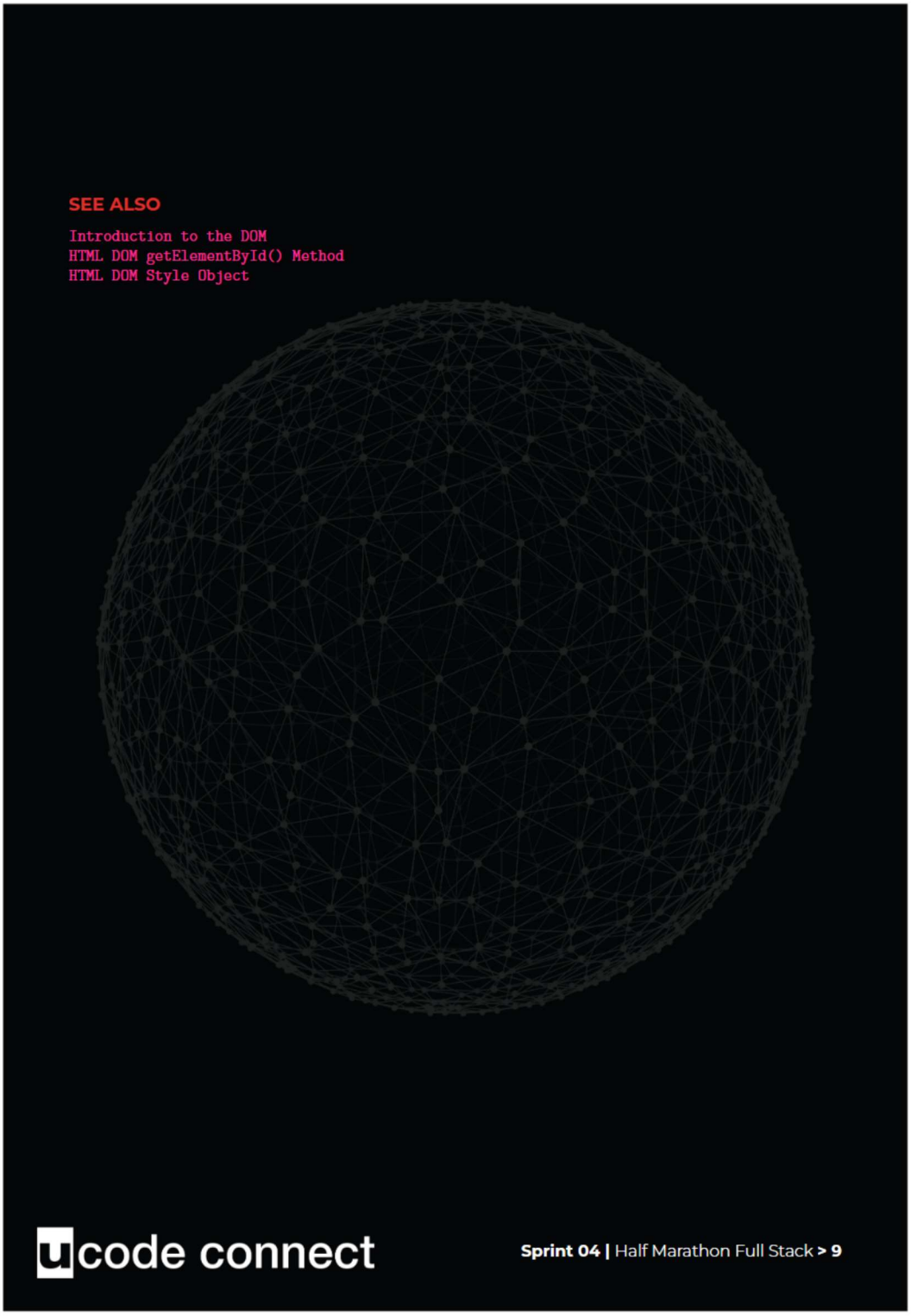
CSS

```css
#lab {
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 10px;
  border-radius: 8%;
  width: 500px;
  height: 500px;
  margin: auto;
  flex-direction: column;
  background: #ffb300;
  border: 6px solid #253324;
  text-align: center;
}

#btn {
  display: inline-block;
  padding: 7px 25px;
  cursor: pointer;
  box-shadow: 3px 4px 0 1px #8a2a21;
  background: #c62d1f linear-gradient(to bottom, #c62d1f 5%, #f24437 100%);
  border-radius: 18px;
  border: 3px solid #d02718;
  color: #fff;
  text-align: center;
  font: bold 25px Arial;
  text-decoration: none;
  text-shadow: 0 1px 0 #810e05;
}

#btn:hover {
  background: #f24437 linear-gradient(to bottom, #f24437 5%, #c62d1f 100%);
}

#btn:active {
  position: relative;
  top: 1px;
}

#hero {
  font: 60px 'Bowlby One', cursive;
  letter-spacing: 2px;
}
```



Sprint 04 | Half Marathon Full Stack > **7**


EXAMPLE
Default view of the page
Secret lab
Bruce Banner
Transform
After clicking "Transform"
Secret lab
Hulk
Transform
After clicking it again
Secret lab
Bruce Banner
Transform
ucode connect
Sprint 04 | Half Marathon Full Stack > 8


SEE ALSO
Introduction to the DOM
HTML DOM getElementById() Method
HTML DOM Style Object
ucode connect
Sprint 04 | Half Marathon Full Stack > 9

Act Basic: Task 01

NAME

Elements

DIRECTORY

`t01_elements/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

`DOM, String.*, Array.*`

DESCRIPTION

Create a JS script that changes the content of a web page. In this task you will need to work with HTML attributes. Use the HTML and CSS files available in the SYNOPSIS.

The web page contains a list of characters. Each character has some attributes. The `class` attribute specifies whether the character is good or evil. And the `data-element` attribute lists the elements that the character can control.

Compare the appearance of the web page with the screenshot in the EXAMPLE. Without editing the HTML and CSS files, use JS only to achieve the same result.

For each `li` element, your script must do the following:

- correct errors in attributes
  - if the `class` attribute is missing, or doesn't equal to `good`, `evil` or `unknown`, make the `class` equal to `unknown`
  - if the `data-element` attribute is missing, make the `data-element` equal to `none`
- append circle `div` elements depending on the attributes
  - for each `data-element` attribute, append a circle
  - each circle must have two class names: `elem`, and the name of the respective `data-element` attribute
  - to each `none` circle, append a `div` element with the class name `line`

Do not hardcode the solution. Your script must work, even if the contents of the list changes (items are added, removed, attributes are changed, etc.).

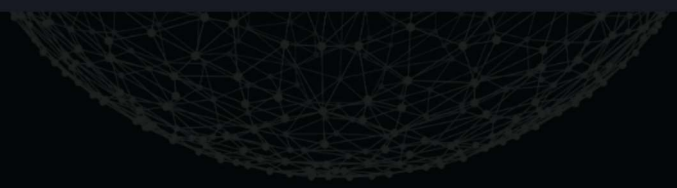
ucode connect

Sprint 04 | Half Marathon Full Stack **> 10**

SYNOPSIS

HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Elements</title>
  <meta name="description" content="t01. Elements<">
  <link rel="stylesheet" href="css/style.css">
  <link href="https://fonts.googleapis.com/css2?family=Bowlby+One&display=swap"
        rel="stylesheet">
</head>

<body>
  <h1>Elements</h1>
  <ul id="characters">
    <li class="good" data-element="air water earth fire">Aang</li>
    <li class="evil" data-element="fire">Zuko</li>
    <li class="good" data-element="water">Katara</li>
    <li class="good">Sokka</li>
    <li class="good" data-element="fire">Iroh</li>
    <li class="evil" data-element="fire">Azula</li>
    <li class="good" data-element="air water earth fire">Korra</li>
    <li>Suki</li>
    <li class="good" data-element="earth">Toph Beifong</li>
  </ul>
  <script src="js/script.js"></script>
</body>

</html>
```



ucode connect

Sprint 04 | Half Marathon Full Stack > 11

CSS

```css
body {
  background-color: #333;
  text-align: center;
  max-width: 800px;
  margin: 0 auto;
}

h1 {color: white;}

li {
  list-style: none;
  display: inline-block;
  background-color: lightgrey;
  padding: 15px;
  margin: 5px;
  border-radius: 10%;
  text-align: center;
  border: none;
  width: 120px;
}

.good {border: 3px solid #59a440;}
.evil {border: 3px solid #ba2d29;}
.unknown {border: 3px solid #764cae;}

.elem {
  display: inline-block;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  border: 2px solid white;
  margin: 3px;
}

.air {background-color: #5a8de1;}
.water {background-color: #0f1b8b;}
.earth {background-color: #496b2e;}
.fire {background-color: #9f000e;}
.none {background-color: lightgrey;}

.line {
  position: relative;
  width: 20px;
  height: 2px;
  background-color: white;
  border-radius: 0;
  top: 9px;
```

```
  transform: rotate(-225deg);
}
```

EXAMPLE

SEE ALSO

Element.attributes
HTML | DOM appendChild() Method

ucode connect

Sprint 04 | Half Marathon Full Stack > **13**

Act Basic: Task 02

NAME

Image slider

DIRECTORY

`t02_image_slider/`

SUBMIT

`index.html, css/style.css, js/script.js, assets/images/*`

ALLOWED

`DOM, String.*, Array.*, Object.*, setInterval(), clearInterval()`

DESCRIPTION

Create a web page with an image slider. Follow these requirements:

- one image is shown at a time
- the "left" and "right" buttons let the user to see the previous/next image
- the images slide every 3 seconds by default until user click any button to slide the image
- there are at least 3 images (store them in `assets/images/*`.)

The design of your web page must look like the image in the EXAMPLE. Use CSS to achieve the result.

EXAMPLE

ucode connect

Sprint 04 | Half Marathon Full Stack **> 14**

Act Basic: Task 03

NAME

Weather forecast

DIRECTORY

`t03_weather_forecast/`

SUBMIT

`index.html, css/style.css, js/script.js, assets/images/*`

ALLOWED

DOM, Object.*, json, Date.*, String.*, Class.*, Array.*

BEFORE YOU BEGIN

In order to succefully complete this task you need to get aquantied with a few topics. Start with API. What is it? Which web services have it? Consider watching this video as an example of API usage.

Then, understand a concept of JS promises.
And, in your mind, try to "connect" promises with fetch. Because no serious work with API in JS could be done without it.

And of course, do not hesitate to read MDN Web Docs.

DESCRIPTION

Create a weather forecast web page.

In order to get real weather information, find an API for weather forecasts.
There are many to choose from.

Your forecast can be for any city/cities.

The web page must visualize weather forecast for the next several days with information about the temperature (in Celsius), and a visual representation of the weather state (whether it will be sunny, cloudy, raining, etc.).

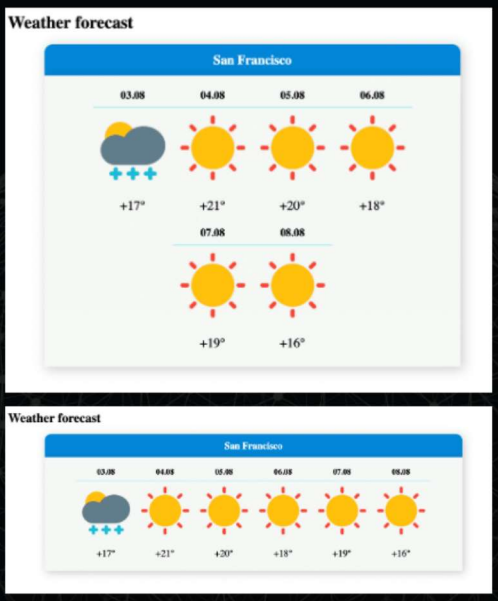
The days of the forecast must not be hardcoded. They must depend on the current date. For example, if today is 15th of March, the forecast will display information from 15th onwards.

See the EXAMPLE images for a reference.

ucode connect

Sprint 04 | Half Marathon Full Stack **> 15**

EXAMPLE



SEE ALSO

Good summarizing article
For those who want to understand asynchronous programming better...

ucode connect

Sprint 04 | Half Marathon Full Stack **> 16**

Act Basic: Task 04

NAME

Notes with cookies

DIRECTORY

`t04_notes_with_cookies/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED FUNCTIONS

`alert(), confirm(), DOM, String.*, Array.*, Date.*, Object.*, JSON.*`

DESCRIPTION

Create a note-tracking web page that contains:

- text area field `Text input`
- button `Add to cookies`
- button `Clear cookies`
- output field `Notes archive`

Look at the EXAMPLE. The design is up to you.

Behavior of the web page:

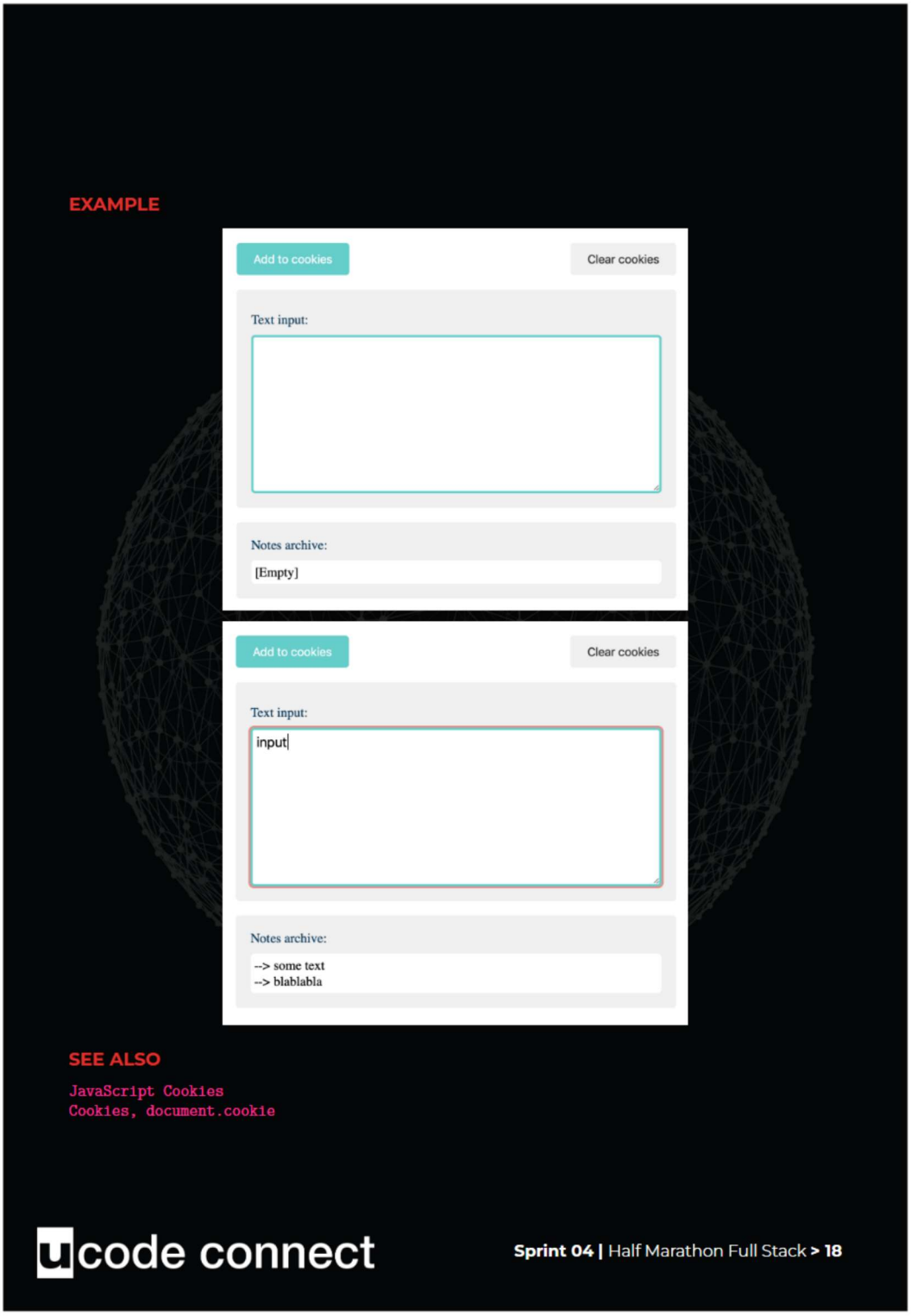
- if the button `Add to cookies` is pressed - the text is added to the `Notes archive`
- if the text input area is empty - an alert box appears with a message `It's empty. Try to input something in "Text input".`
- if the button `Clear cookies` is pressed - the confirm box appears with a message `Are you sure?`, and if the answer is positive - the output field is cleared
- `[Empty]` is displayed in the output field if the web page is launched for the first time, or the date storage has expired

In this task, you must use cookies for implementation. Set the expiry date to 30 days. After refreshing the page, the `Notes archive` stays the same (doesn't clear).

ucode connect

Sprint 04 | Half Marathon Full Stack **> 17**

EXAMPLE



SEE ALSO

JavaScript Cookies
Cookies, document.cookie

ucode connect

Sprint 04 | Half Marathon Full Stack **> 18**

Act Advanced: Task 05

NAME

Drag'n'Drop Stones

DIRECTORY

`t05_dragndrop_stones/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

`Object.*, Class.*, String.*, Array.*, Function.*`

LEGEND

With all the six stones, I could simply snap my fingers, and they would all cease to exist. I call that... mercy.

DESCRIPTION

Create a web page with six blocks of different color, as shown in the EXAMPLE.
Implement the following features:

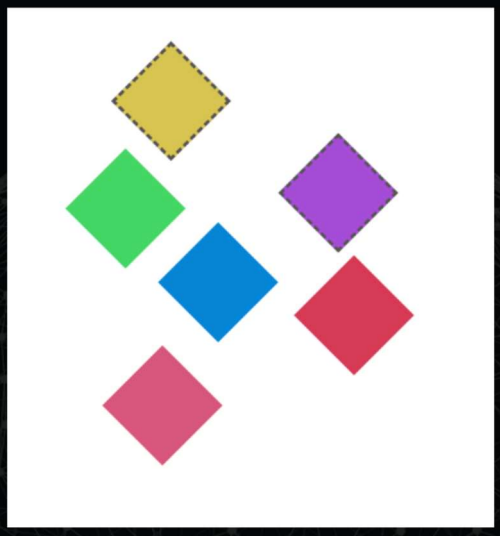
- a block can have the Drag'n'Drop functionality switched on/off by clicking it
- a block with Drag'n'Drop off has a visible border and cannot be moved
- a block with Drag'n'Drop on can be moved around using the mouse

Use events to implement this web page.

ucode connect

Sprint 04 | Half Marathon Full Stack **> 19**

EXAMPLE



SEE ALSO

JavaScript Events

ucode connect

Sprint 04 | Half Marathon Full Stack > 20

Act Advanced: Task 06

NAME

Sort table

DIRECTORY

`t06_sort_table/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

`DOM, String.*, Array.*, Object.*`

DESCRIPTION

Create a web page with a table using JS. Follow these requirements:

- a table must have three columns and ten rows
- the first column must be filled with Marvel superheroes
- the second must be filled with the hero's strength
- the second must be filled with the hero's age
- the web page must have click events for the headers of the table:
  - sort the table according to the selected column by rearranging the rows. Sorting can be performed in ascending and descending order
  - show a service message `Sorting by [parameter], order: __`. Instead of "__" must be "ASC" or "DESC"

The design of your web page must look like the image in the EXAMPLE. Use CSS to achieve the result.

SYNOPSIS

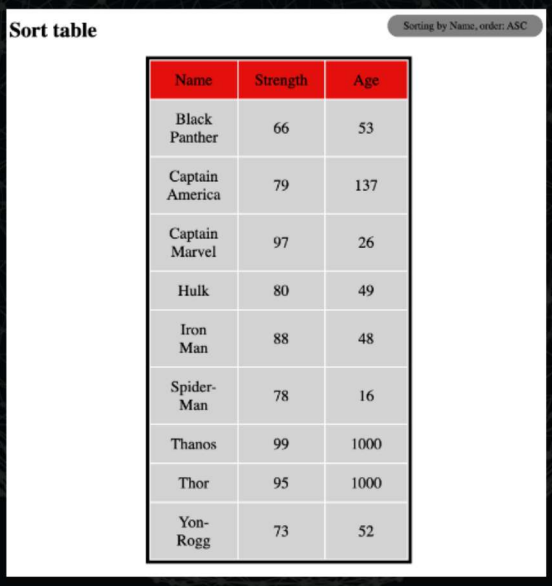
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
  <title>Sort table</title>
  <meta name="description" content="t06. Sort table">

  <link rel="stylesheet" href="css/style.css">
  <script src="js/script.js" defer></script>
</head>
```

ucode connect

Sprint 04 | Half Marathon Full Stack > **21**

```
<body>
  <main>
    <h1>Sort table</h1>

    <div id="placeholder">Placeholder with <b>HTML</b></div>
    <div id="notification">Click on cell to sort the column</div>
  </main>
</body>

</html>
```

EXAMPLE



ucode connect

Sprint 04 | Half Marathon Full Stack **> 22**

Act Advanced: Task 07

NAME

Marvelous list

DIRECTORY

`t07_marvelous_list/`

SUBMIT

`index.html, css/style.css, js/script.js, assets/images/*`

ALLOWED

`DOM, Object.*, String.*, Array.*, Set.*`

DESCRIPTION

Create an HTML page with a list of at least three films like shown in the EXAMPLE. The idea is to be able to click different titles of the list and see more information on the selected one.

The page contains the following elements:

- a list of clickable titles of the films
- a block of information about a particular film from the list with the film's
  - title
  - poster image (save it to `assets/images/` directory)
  - the date of production
  - information
  - main actors

Add a `click` event to the document.
When the user clicks on a film's title in the list - the web page displays information about the selected title.

ucode connect

Sprint 04 | Half Marathon Full Stack **> 23**

EXAMPLE

Marvelous list

John Wick

Avengers: Endgame

Inception

Avengers: Endgame

April 22, 2019

Robert Downey Jr. | Chris Evans | Ian McShane | Chris Hemsworth | Scarlett Johansson

After the devastating events of Avengers: Infinity War, the universe is in ruins due to the efforts of the Mad Titan, Thanos. With the help of remaining allies, the Avengers must assemble once more in order to undo Thanos' actions and restore order to the universe once and for all, no matter what consequences may be in store.

ucode connect

Sprint 04 | Half Marathon Full Stack > 24

Act Advanced: Task 08

NAME

Lazy loading

DIRECTORY

`t08_lazy_loading/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

`DOM, Object.*, Class.*, String.*, Array.*, setTimeout()`

DESCRIPTION

In this task, you'll practice lazy loading images.

Create a web page with 20 images positioned vertically, one after another.
Use the template given in the SYNOPSIS to add the images in your 'index.html' file. In this template

- `src="temp.jpg"` is the placeholder image (what you would see instead of the image before it loads)
- `data-src="img.jpg"` is the real image that you want on your web page (replace "img.jpg" with a link or path to your image)

When the page is loaded - unavailable images must not be displayed (display the placeholder images instead).

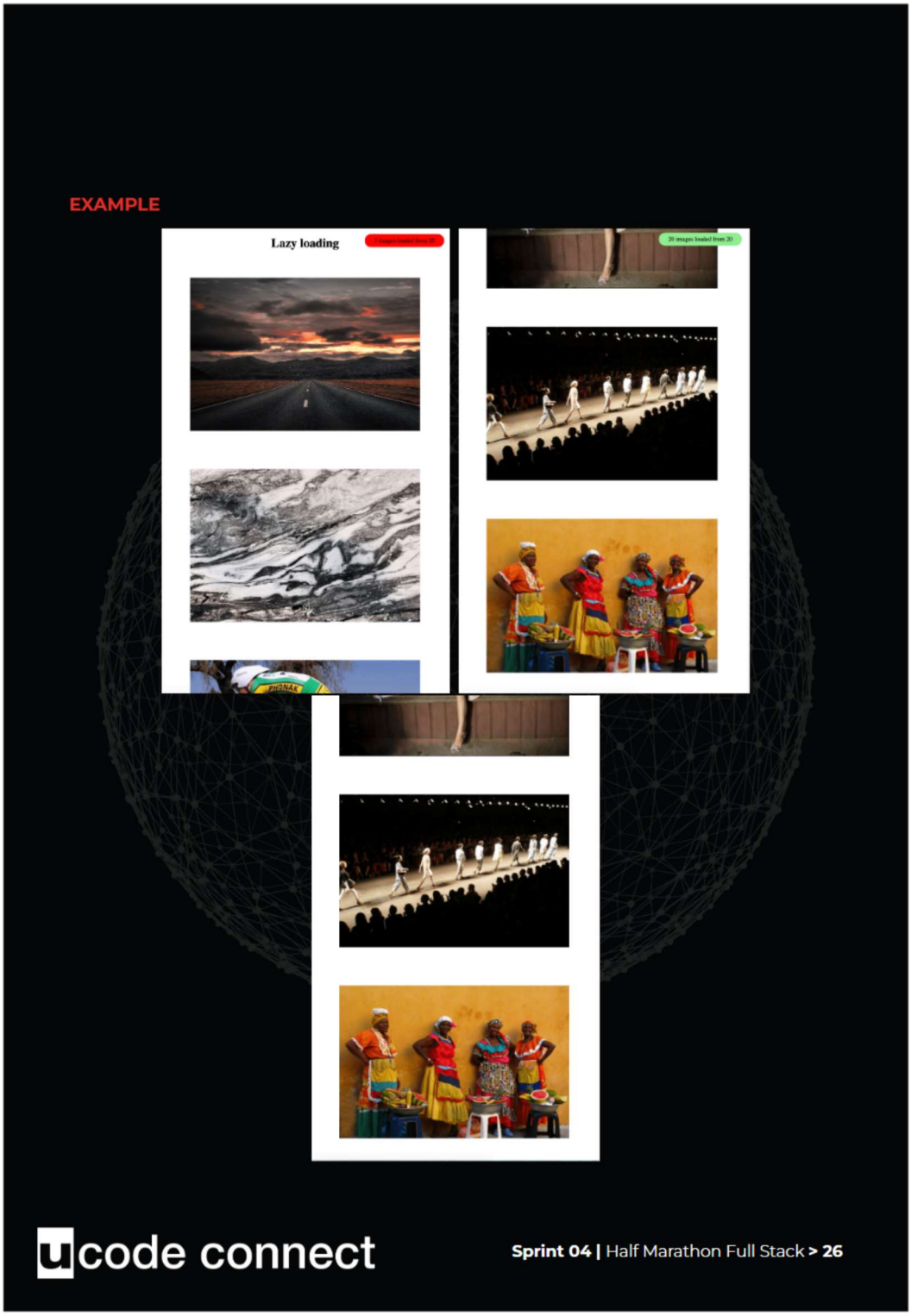
Implement the lazy loading effect - load the real image only when it's visible in your window.
Also, implement a message in the right top corner of the screen. The message must display the number of images that were loaded in real-time.
When all images were loaded, the message becomes green and disappears after 3 seconds.
You can either use image files (and put them into `assets/images/*`, or use links to images online).

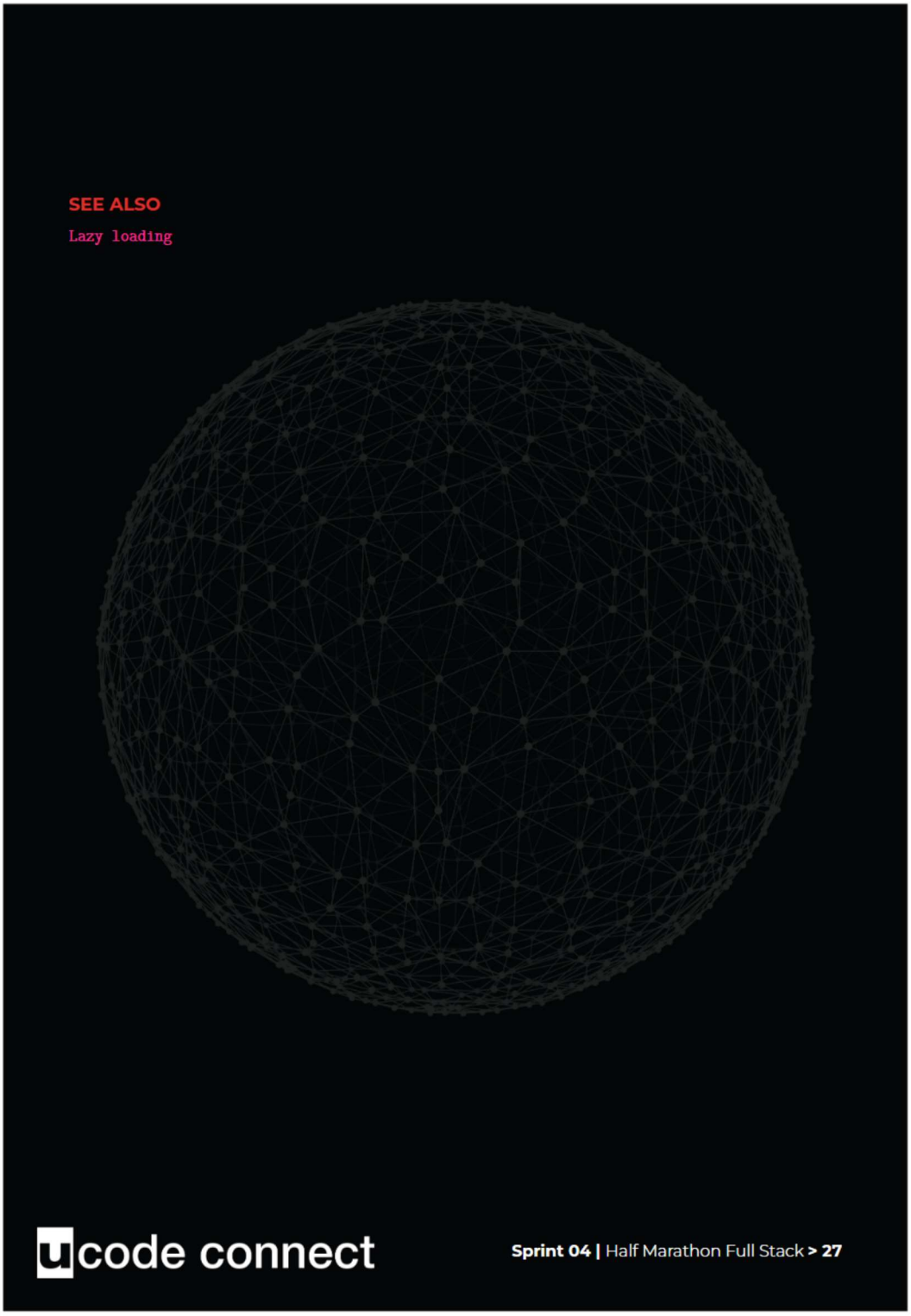
SYNOPSIS

```
<img src="temp.jpg" width="200" height="200" data-src="img.jpg">
```

ucode connect

Sprint 04 | Half Marathon Full Stack **> 25**


EXAMPLE
Lazy loading
ucode connect
Sprint 04 | Half Marathon Full Stack > 26


SEE ALSO
Lazy loading
ucode connect
Sprint 04 | Half Marathon Full Stack > 27

Act Advanced: Task 09

NAME

Notes with local storage

DIRECTORY

`t09_notes_with_local_storage/`

SUBMIT

`index.html, css/style.css, js/script.js`

ALLOWED

`alert(), confirm(), String.*, Array.*, Date.*, Object.*, RegExp.*, DOM, localStorage.*, JSON.*`

DESCRIPTION

This task is very similar to the Task 04 Notes with cookies, but with one key difference, your web page will "remember" things using localStorage instead of cookies.
In this task, you must:

- change data storage from cookies to localStorage
- rename the buttons `Add to storage` and `Clear storage`.
  Of course, their behavior must be the same as the Task 04 Notes with cookies
- add date display as in the EXAMPLE

ucode connect

Sprint 04 | Half Marathon Full Stack **> 28**

EXAMPLE

SEE ALSO

LocalStorage
The complete guide to using localStorage

ucode connect

Sprint 04 | Half Marathon Full Stack **> 29**

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. It allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the task and better understand your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.