Programming in HTML5 with JavaScript and CSS3

Overview

This course offers an introduction to HTML5, CSS3 and JavaScript. It reviews the fundamentals of each technology and guides the students to design and develop their own user interfaces and make them interactable to capture user input.

Requirements

- For the following exercises you need basic understanding of the browser tools and a preferred text editor installed.
- You'll need to download the following repository.

Module 1: HTML, DIVS & Background-color

HTML is a markup language that defines the structure of a web document and it's composed of tags. All web documents must contain at least a basic structure built by the following tags:

<html>	Defines the root of an HTML document
<head>	Contains metadata for the document
<body>	Defines the document's content

This tags will help us build an html document nesting them as follows:

```
<html>
    <head>
    </head>
    <!--This is a comment -->
    <body>
    </body>
</html>
```

You will encounter two different kinds of tags, there are some with an opening and a closing tag as the ones presented in the example before and there are other self enclosing tags such as an img tag looking something like:

```
<img /> <!--Notice the forward slash at the end of the tag -->
```

When working with an html document you will be adding several different kinds of tags depending on the content or the needs of your project and you will be nesting these elements in an ordered way that will ensure the readability of the document not only for you but for the Web Browsers.

Let' begin with the most common elements inside our head tag:

```
<head>
    <title>Defines the title of the page being displayed</title>
    <meta />Define meta data of the page
    <link  /> <!--Define a favicon or link an external document -->
     <style> /* You can create internal style sheets inside these tags */
</style>
    <script>
        // Define a workspace for JavaScript Code
    </script> <!-- You may also find this tag being used to link to an external
.js file -->
</head>
```

Elements inside the head won't be visible in our browser, they help us to define the identity and the tools needed for our page to properly run.

Then we have our most common elements inside our body tag:

```
<body>
    <h1>Defines a title element, you can use from h1 to h6</h1>
    <p>Defines a paragraph</p>
    <img /> <!-- Defines an image -->
    <a>Defines an anchor or a link and everything nested inside it becomes
the link</a>
    <ol>Defines an ordered list, inside it should nest some li items.</ol>
    <ul> <!-- Defines an unordered list -->
        <li>Defines an item inside a list</li>
</ul>
</body>
```

There are two main structure tags that will help us to organize our content:

```
<div>
    Div is useful to create divisions or different sections in our documents.
</div>
<span>
    Span is useful to create inline divisions, mostly used inside paragraphs
to style text.
</span>
```

Inside each tag you can set properties using a key/value format:

```
<img src="path-to-image" alt="Text shown on the browser in case no file is
found" />
```

Some properties are specific for certain tags and some other properties can be used with all the tags. A very important one is the style property which will allow us to create some inline style in our elements:

```
<p style="font-size: 24px; color: black;">This text is black and it has a font
size of 24px</p>
```

Inside the style tag or the style property we can use CSS which is the language used to style the HTML documents by setting rules. HTML elements have by default some CSS rules depending on the browser we're working on so it would be useful to set some rules to all elements to ensure consistency of our web documents on all the web browsers.

This is how we set rules in CSS:

```
/* We type the tag we want to style and inside the brackets we can choose different properties of that element and modify its values creating our own custom css rules. */
p {
    /* This is a comment in CSS */
    background-color: white;
    color: black;
    display: block;
    position: static;
}
/* You can apply the same rules to different tags at the same time by adding a comma */
h1, h2, h3 {
    font-size: 24px;
}
```

It is a good practice to keep the CSS properties in an alphabetical order.

You can also set the CLASS and ID properties to the tags to identify them easier or to set more comprehensive CSS rules.

The CLASS property can be reused in different tags but if you use the ID property, make sure you use that ID in one and only one element per document.

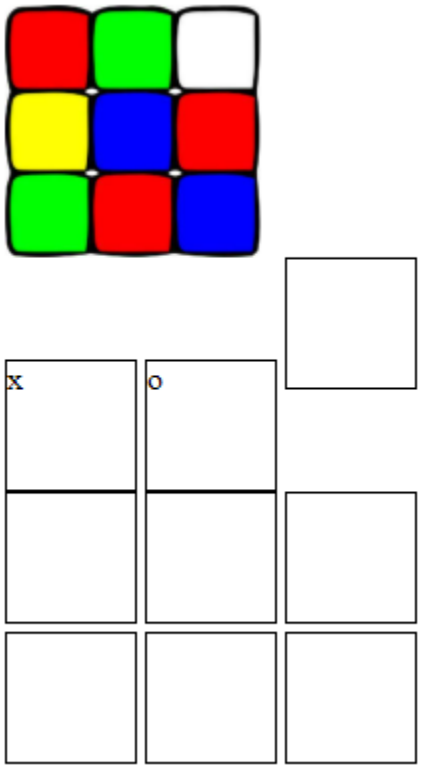
```
<h1 id="title-1" class="title">Title</h1>
<h2 id="title-2" class="title">Title</h2>
<h3 id="title-3" class="title">Title</h3>
```

You can then style these elements in CSS by selecting them properly:

```
.title {
    Background-color: black;
    color: white;
}
#title-1 {
    font-size: 24px;
    font-weight: bold;
}
#title-2 {
    font-size: 18px;
    font-weight: bold;
}
#title-3 {
    font-size: 14px;
    font-weight: normal;
}
```

In the rules we set that all the titles will have a black background and a white colored text, also we specified different font-sizes and weights for the different titles.

Lab: Basic HTML Elements



Scenario

You're creating a website to display different configurations of a Rubik cube's face. You decide to create these configurations by yourself using only the basic structure tags and some styling properties.

Objectives

- Draw a grid of 3x3 by using div and span items (or any other approach).
- Color the elements accordingly to the image provided.
- Copy that grid and generate a play field for a tic tac toe match.
- Choose a different color for your crosses and circles.
- Center the crosses and circles.

Hints

1. Check on the display property of the elements on the browser console.
2. Check on the margin and padding properties of the elements.

Lab Setup

Estimated time: 15 minutes

Module 2: Introduction to position property

Elements in a web document are being stacked following the html structure you're building. The elements will behave differently depending on their display value:

```
display: block;
display: inline;
```

There are more values we can set to the display property but these are the main two.

The block display property will render an element as a box, it will automatically take an entire row of space and everything written after this element will automatically appear below it on the web document.

The inline display property renders an adjustable element that's going to occupy space as needed and you can have several inline elements in the same line.

You can test this by yourself by creating a div and a span element, writing text inside them and then coloring them:

```
<html>
    <head>
        <style>
            div {
                background-color: red;
            }
            span {
                background-color: green;
            }
        </style>
    </head>
    <body>
        <div>
            Some text
        </div>
        <span>
            Some text
        </span>
    </body>
</html>
```

By default these elements obey a static positioning, which is a value we can alter to position or render the elements on different areas in our web document.

```
div {
    position: static; /* The default behaviour */
    position: relative; /* Enables to render an element relative to its static position */
    position: absolute; /* Enables to position an element inside the parent's scope */
    position: fixed; /* Enables to position an element relative to the viewport even when scrolling */
     position: sticky; /* It changes automatically from relative to fixed positioning depending on a condition */
}
```

Once you have chosen a position value, you can modify precisely the location of the element by using the next properties:

```
div {
    bottom: 0;
    left: 0;
    right: 0;
    top: 0;
}
```

You can use different units, it can be in px, percentages, the width and height of the viewport, etc.

Other ways to modify an element’s position is by taking advantage of the next properties:

```
div {
    float: left;
    float: right;

    margin: 0;

    padding: 0;

    text-align: center;

}
```

Also a very common practice is to nest an absolute positioned element inside a relative one:

```
<body>
    <div style="position: relative;">
        <div style="position: absolute;">

        </div>
    </div>
</body>
```

This often ends up being an awesome way to display elements exactly where we want them.

Lab: Positioning HTML Elements

Scenario

You have a bunch of sheeps and wolfs hanging around in your terrain. You need to separate them by using a wall or else the sheeps will get eaten.

Objectives

- Move the wolfs to one side of the viewport.
- Move the sheeps to the other side of the viewport.
- Position the wall between the wolfs and the sheeps.

Lab Setup

Estimated time: 15min

Module 3: HTML5 tags

HTML5 adds new tags to HTML and to start working in an HTML5 document you need to specify at the beginning of your document the next tag:

```
<!DOCTYPE html>
```

HTML5 adds new useful elements to add media content optimizing the best resource for the target device and also to better structure your documents.

Before HTML5 a common practice to structure a website was as follows:

```
<body>
    <div class="header">
        <div class="nav">

        </div>
    </div>

    <div class="main">

    </div>

    <div class="footer">

    </div>
</body>
```

The main elements added in HTML5 are:

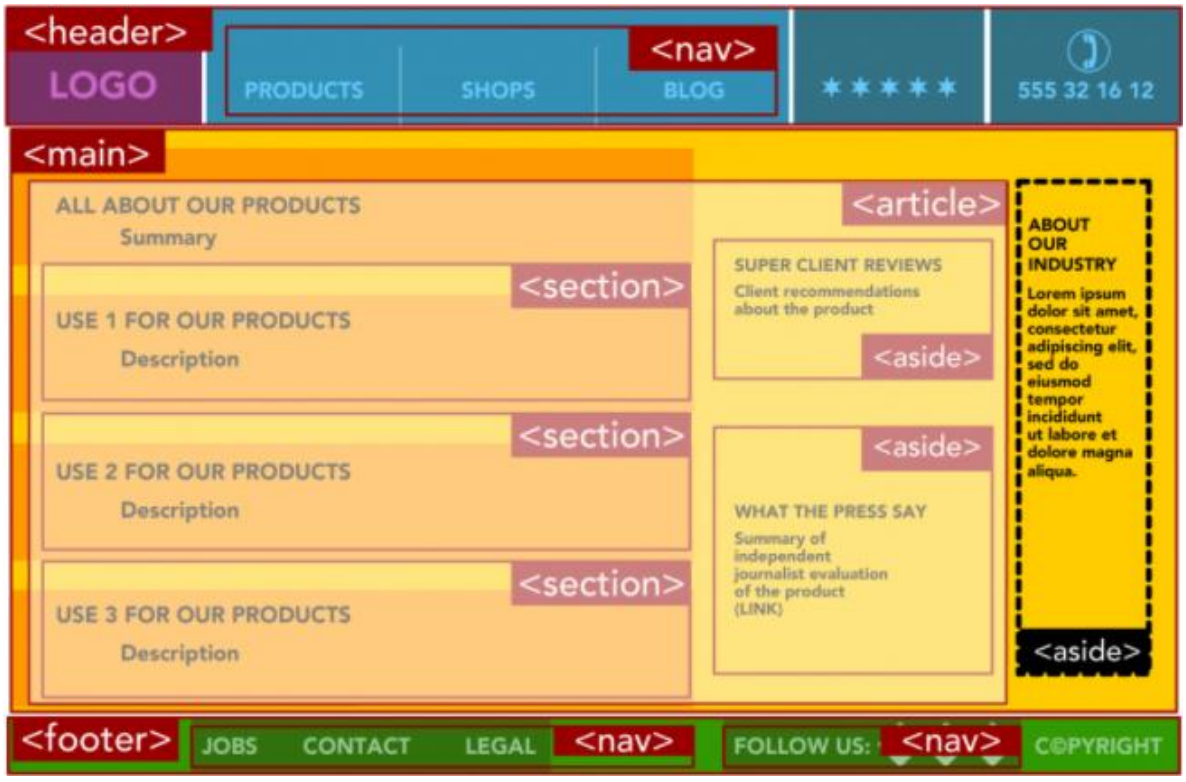
```
<header></header>

<nav></nav>

<main></main>

<footer></footer>
```

```
<article></article>

<section></section>

<aside></aside>
```

We can considere them as specialized DIV elements as they will fulfill a task telling the browser what kind of contents are found within the element.

All these elements can be reused in the same document except for the main tag, and they don't exclude the use of the DIV and SPAN tags.

Lab: Basic HTML Elements



Scenario

You can choose between 3 different scenarios:

1. You want to build a personal blog
2. You wish to create a new social network
3. You desire to design a new streaming platform

After you selected one scenario, look up for some already existing websites based on your choice and try to design your own structure for your project considering what elements you would need to display.

Objectives

- Choose 1 scenario
- Do some benchmarking by comparing the competition with your project
- Create a web structure by using HTML5 tags, position them and color them with CSS

Lab Setup

Estimated time: 30 minutes

Module 4: Forms

Forms are useful to capture user input and data. We will use special tags to create graphical user interface elements depending on the data we want to capture. Forms was heavily improved since HTML5 adding more input types.

A basic structure of a form would look like this:

```
<html>
    <head>
        <title>Forms</title>
    </head>
    <body>
        <h1>Forms</h1>

        <form action="index.html" method="get">
            Name: <input type="text" name="name">
            <input type="submit">
        </form>
    </body>
</html>
```

You'll notice the form element which contains all the elements to capture user data. We used an action property to specify to whom the data is going to be submitted, in this case the index.html specifies we're going to send the data to the same document assuming we're working on that file. The method property specificies how the data is going to be sent, get sends the data via the url and post will send the values in a not visible way for the user.

The input tags are the elements used to capture data and we have several different types of input types depending on what we need to capture, these types by default will enable some input validation on the browser. The name property will help us to identify and retrieve the data sent by the user.

Some input types are:

```
<input type="text">
<input type="email">
<input type="password">
<input type="number">
<input type="tel">
<input type="checkbox">
<input type="radio">
<input type="color">
<input type="button">
<input type="date">
<input type="file">
<input type="reset">
<input type="submit">
```

There exist some other elements that can contribute to improve and organize a form:

```
    <body>
        <h1>Forms</h1>

        <form action="." method="post">
            <fieldset>
                <legend>User</legend>
                      <label  for="name">Name:  <input  type="text"  id="name"
name="name"></label>
            </fieldset>

            <fieldset>
                <legend>HTML5</legend>
                      <label  for="name">Name:  <input  type="text"  id="name"
name="name"></label>
            </fieldset>

            <input type="submit">
        </form>
    </body>
```

Lab: Basic HTML Elements

Scenario

You built your project from module 3. You want to create a form to allow users to sign up to your website.

Objectives

- Define what user data you need to ask from your users
- Build a form that fulfills your requirements
- Style the form and test the by default input validation offered by the browser

Lab Setup

Estimated time: 30 minutes

Module 5: Advanced selectors in css

In module 1 and 2 we wrote simple CSS rules to style our HTML elements. However we can go a step ahead and use advanced selectors to improve our stylesheets and achieve better results.

Advanced selectors will allow us to simplify code and also to obtain interesting results, the easiest way to target an element is by using the CLASS or the ID property, however sometimes maybe we don't want to target a specific CLASS, maybe we want to style a CLASS that's inside another element. We can achieve that by nesting in CSS:

```
/* We target to style a strong element that is inside a p element that is
inside a div element */
div p strong {
    background-color: black;
    font-weight: 900;
}
```

In the previous css rule not all the strong elements will be styled, they need to comply with the written conditions, that is being inside a p element that is inside a div element.

Using the next HTML snippet, try to identify in which cases this css rule will apply:

```
<html>
    <head>

    </head>
    <body>
        <p>This is an <strong>interesting</strong> phrase</p>
        <div>
            <p>This is a <strong>pretty</strong> phrase.</p>
        </div>

        <div>
            <div>
                <p>This is a <strong>prettier</strong> phrase.</p>
            </div>
        </div>
        <div>
            <div>
                <span>
                    <p>This is a <strong>cute</strong> phrase.</p>
                </span>
            </div>
        </div>
    </body>
</html>
```

Now let's add some advanced selectors to the formula:

```
* {
    background-color: black;
    font-weight: 900;
}
div > p strong {
    background-color: black;
    font-weight: 900;
}
```

The asterisk is selecting all the elements in the document, it can be very useful to get rid of some by default styles set by the browser.

The greater than sign is forcing the nesting order of the target, in order to style the strong element the p element must exactly be the direct child of the div element.

In the next scenario something similar occurs:

```
<html>
    <head>
        <style>
            div ~ a {
                text-decoration: none;
            }
        </style>
    </head>
    <body>
        <div>
            <a href="#">Test this</a>
        </div>

        <a href="#">And also test this</a>
    </body>
</html>
```

The circumflex accent specifies that the target must be preceded by a DIV.

In the next example you must check on the browser console to understand what’s going on:

```
<html>

    <head>
        <style>
            .one.two {
                background-color: purple;
            }
            .one .two {
                background-color: cyan;
            }
        </style>
    </head>
    <body>
        <div class="one">
            <div class="two">
                First phrase
            </div>
        </div>

        <div class="one two">
            <div class="two">
                Second phrase
            </div>
        </div>
    </body>
</html>
```

Both CSS rules are being applied, so both DIV elements are getting colored, however since one div is structured inside the other and since they're both the same width and height it seems like if only one rule applied. Can you notice the difference between the two CSS rules?

On the next example we can use some special properties to precisely select a specific element.

```
<html>
    <head>
        <style>
            div:nth-child(1),
            div:nth-child(3),
            div:nth-child(5) {
                background-color: gray;
            }
        </style>
    </head>
    <body>
        <div>
            <p>1</p>
        </div>
        <div>
            <p>2</p>
        </div>
        <div>
            <p>3</p>
        </div>
        <div>
            <p>4</p>
        </div>
        <div>
            <p>5</p>
        </div>
    </body>
</html>
```

Try modifying the CSS rule by:

```
div:nth-child(odd) {
    background-color: gray;
}
```

Tip: You should really check on the nth-of-type property seems is very similar to nth-child with a little twist.

You can guess what the next css rule does:

```
div:first-child,
div:last-child {
    background-color: gray;
}
```

Also for links, there are some special properties, check on the next example:

```
<html>
    <head>
        <style>
            a:link {
                color: blue;
            }
            a:visited {
                color: purple;
            }
            a:hover {
                color: red;
            }
            a:active {
                color: black;
            }

        </style>
    </head>
    <body>
        <a href="#1">First link</a>
        <a href="#2">Second link</a>
        <a href="#3">Third link</a>
        <a href="#4">Fourth link</a>
    </body>
</html>
```

Notice how we're adding some # in the target url of the link element, this is called an anchor and it helps us to send the visitor to a specific part of a document using the ID property. If a section of the document contains an ID equals to 1 the first link will send the user directly to it.

As an extra for some advanced CSS, the display property offers the possibility to convert an element to a flex element. A flex element will become a smart structure capable of calculate distances between its child elements and therefore help us to align them depending on our needs.

The display property also allows to create a grid container and this will help us to create and specify the behavior and look of columns and rows of different elements. The grid value of the display property is basically a replacement for the table tag as these are hard to maintain and very difficult to convert them to a friendly version for different devices.

The different display property's values, overall the flex and the grid values are going to be important assets in the construction of Responsive Webpages.

Lab: Basic HTML Elements

Scenario

You were asked by your company to give them some ideas to animate the company's logo when interacting with it, also the company wants to improve the image of some of their web

displayed components, one of them is a static clock and another are some statistic tables. Also they would like to get rid of the default look of their links, they think the underline of the links look awful.

Objectives

- Animate the logo by selecting it and then using some advanced CSS properties.
- Create a clock and add it a hand. If you manage to make the hand spin it’s a bonus.
- Create a table of elements and make it so that the table header is one color, and then the odd rows share another color and the even rows stay completely transparent.
- Add some links to send to the different sections of the document and style them differently.

Lab Setup

Estimated time: 20 minutes

Module 6: Dynamic elements using JavaScript (DOM)

We covered HTML to structure, organize and define our web pages. We also styled our elements with the help of CSS but there's another piece yet to be covered.

JavaScript is the actual programming language behind a website or very often in web applications. This tool is responsible for the functionality and interaction with our project and very similarly to a stylesheet, we can use it inline in our tags by making calls to JS functions and passing parameters, we can use it internally in our web document or we can create an external script file and call it from within our web document.

JS works right away in our browser and it's the client's side programming language by default. You can use it directly when you open the browser's console and run some commands.

JS allows to send messages directly to the output of the console or it allows to generate an alert. Both are good options to test and debug our code:

```
console.log("Hello world")

alert("Hello again")
```

We can create and store variables as well as functions:

```
x = 5
y = 10
z = "Hello"

function add(x, y) {
    return x + y
}
```

And one of the most important features, it allows us to interact with the DOM. We talked about how HTML helps us to structure our web project, basically what we're doing is adding nodes to the DOM. The div tag will be a parent node of the p element inside it and this p element becomes a child node of the div tag.

JS is capable of accessing all the nodes in our web document to delete, create or modify content at any time and without the need to refresh the viewport of our webpage.

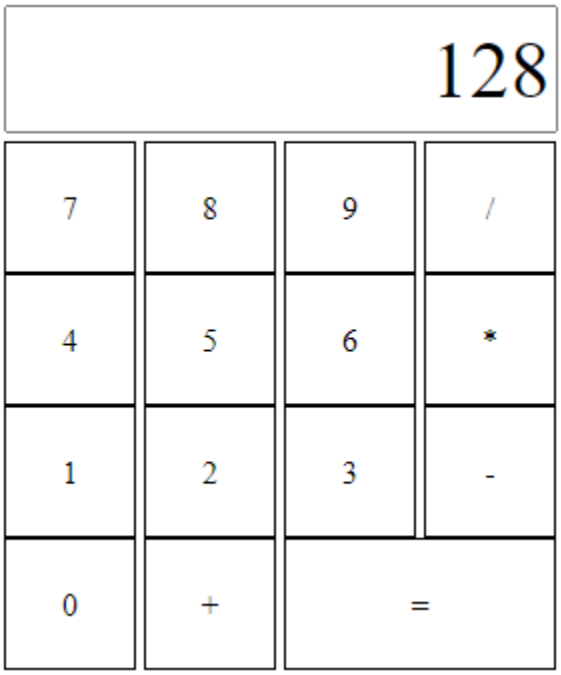
You can create functions that are triggered by user interaction with some element in the web document:

```
function popHello() {
    alert("hello")
}

document.getElementById('div').addEventListener('click', popHello);
```

You can also retrieve values from the elements:

```
var x = document.getElementById("screen").value;
```

Lab: Basic HTML Elements



Scenario

The IT department of the North Star Supermarket decided to implement a calculator on their website to allow visitors to use it without losing focus on the products.

Objectives

- Design and implement a calculator with the basic four operations

Lab Setup

Estimated time: 1 hour

Module 7: Communicate to a remote server

Web development consists of two different stages:

1. Frontend
2. Backend

We have been working the frontend so far, that means, on the client side of the project or what the user can see and interact with. More often than never we also have to take care of what the user don't see but makes everything work as it should, the backend.

The backend can be very simple or extremely complex and you can use different technologies for it. In the past it was very common to handle the backend with PHP but nowadays it is possible to create great experiences using only JS.

We are free to choose our preferred tool, however, either tool should allow us to communicate to external services to send or retrieve data and update the web project according to the results.

JavaScript offers the advantage that it can modify the content of a web document on the go, thus it's very useful to modify and update the content based on asynchronous events.

We can then request files or information to an external server and transparently update the content of the website on the browser of the visitor without making him wait for a load screen to end.

In case of not having access to the contents of a remote server, you can simulate the same behavior by retrieving the contents of another document in your local machine.

However, one of the most common ways to send and receive data from an external server or service is by using API's.

To work with an API you need an endpoint which is a basically a link that points to a service. Then you can create a request to that endpoint and manipulate the response.

```
// Endpoint
const url = 'https://example.com/api/v1/weather/hawaii';

// The fetch method send a request and can manage a successful or an
unsuccessful response from the server.
fetch(url)
// In case of successful response, it parses the response to json and then
you can add the values contained in that json to any html element to display
it on the browser
    .then(response => response.json())
    .then(response => {

        let element = document.getElementById('some_element')
        element.innerHTML = '<p>' + response.celsius +'</p>'
        console.log(data)
}).catch(err=>console.log(err))
```

Lab: Basic HTML Elements

Scenario

You're creating a pet shop's website and you need to serve contents from an external api.

Objectives

- Retrieve the contents of the api and once it's loaded asynchronously display it to the user.

Lab Setup

Estimated time: 30 minutes

Module 8: Login Pages

Login pages are often simple forms that ask for a username or email and a password. We can easily replicate one with two inputs. The magic starts when we need to authenticate a user, for that we would need to communicate with an external database to check if a user has been stored/registered and compare its password.

We can try to recreate this locally by storing two variables:

```
username = "Hibou365"
password = "12345678"
```

We have set up our little local database, and we can use these values to be compared against what we capture in a form.

First we need to retrieve the values captured in the form:

```
var usernameInput = document.getElementById("username");
var passwordInput = document.getElementById("password");
```

Then we can compare them to the values stored in our database:

```
var usernamesMatch = usernameInput.value === username.value;
var passwordsMatch = passwordInput.value === password.value;
```

You can then set yourself a custom system to display feedback to the user:

```
if (passwordsMatch) {
    // Clear any previous error message.
    document.getElementById("errormessage").value("");
} else {
    // Setting this error message will prevent the submission of the form.
       document.getElementById("errormessage").value("Your  passwords  don't
match. Please type the same password again.");
}
```

Lab: Basic HTML Elements

Scenario

You're creating the login page of the inventory system in your company's intranet. You're asked to ensure that only the administrator can access to the inventory.

Objectives

- Generate some local credentials for the administrator.
- Create a form and code the authentication flow for the admin user.
- Provide feedback for the user when trying to login.

Lab Setup

Estimated time: 30 minutes

{ } CHALLENGE

For the next challenge you're going to use all the tools we've covered so far. Please read the scenario carefully and don't forget you can review the precedent modules at any moment.

Lab: Basic HTML Elements

Scenario

You were hired by a young team of entrepreneurs with an idea for a startup. They tell you they first need a website to publish and visualize some posters, they let you choose if you want to use movie posters, videogame portraits, art displays or others. The young team provides you with the design requirements for the web platform and they ask you to add the possibility to mock up the register and login users functionality. Logged in users can visualize the details of each poster and they can also see their username being displayed on the main navigation.

Objectives

- Review the design of the website and build the structure with HTML.
- Check on the functionality requirements and add every element that's needed.
- Once all the elements are in place code a mockup with local credentials for the login functionality.
- Read the style requirements and comply with every one of them.

Lab Setup

Estimated time: 2 hours