FREE TO CODE

Improving Digital and Coding Skills for Inmates

Training Content (Intellectual Output 2)

2018-1-RO01-KA204-049298

Co-funded by the Erasmus+ Programme of the European Union

Co-funded by the Erasmus+ Programme of the European Union

Project Number 2018-1-RO01-KA204-049298

Programme
FREE TO CODE - *Improving Digital and Coding Skills for Inmates*

Partners
European Strategies Consulting (Romania)
APROXIMAR – Cooperativa de Solidariedade Social, Crl (Portugal)
Asturia vzw (Belgium)
Avaca Technologies Consulting, Informatics AE (Greece)
BeCode (Belgium)
Dlearn – European Digital Learning Network (Italy)
Penitenciarul Bucaresti-Jilava (Romania)

Design
Aproximar, Cooperativa de Solidariedade Social

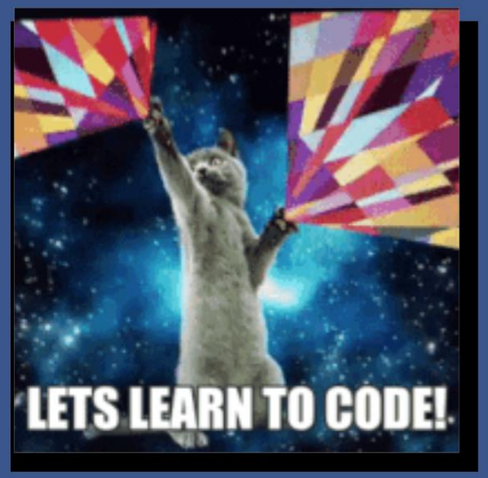
Publication date
February, 2020

www.free2code-initiative.eu

/// Index

a } General Introduction

General Intro to the Course



Hello World!

Since the computers and the Internet appeared, they have changed deeply how we play, how we communicate, how we work, how we live. Because of these deep changes in our Society, there are many jobs available for people who have learned to code and solve problems by coding websites and web applications.

Perhaps you think you need a University degree in Mathematics and nuclear engineering to be able to code? Not at all! Coding is easy once you "get it" : even kids can learn programming... Because it actually is a lot of fun! If you ask programmers what they do every day, they will say things like: "it's like playing with Lego Bricks" or "It's like cooking, writing recipes...".

What you can expect

The objective of this training is to allow you to have a try at programming and logical thinking, to see if you too find it fun and something you would like to pursue once you get out of jail.

This course will have you learn the basics of website making and programming using the 3 main languages of the web: HTML, CSS, and JavaScript. By the end of the training, you will be more than ready to join a coding bootcamp that will help you reach a level in which you can get a real job as a programmer!

Learn by doing

The best way to learn coding is just like cooking... By doing it! This is why the course is organized as a series of exercises that you need to succeed in order to move on to the next step. Don't worry about being fast, it's not like school! The system remembers your progress so you don't have to start over from one session to the next.

... It doesn't work! I suck at this!!

No, you don't. Coding is about trying until it works. So we all start by failing! Get used to it, failure is part of the learning process. Just keep trying, you will make it!

A final word from the team

Ready ? Let's go!

b (Module 0

Internet and Web Info

Background section

@ *The internet*

_A network of networks

Before the invention of the internet, many universities and institutions already had some computer infrastructure, consisting of computers connected to each other as local networks. Using these computers, one could therefore publish content and communicate within its own institution, but not globally.

To write this content, people could use a very simple but powerful innovation: hypertext. Hypertext simply means text that can contain "hyperlinks", which is a fancy way to describe a link pointing to texts located in other parts of the network.

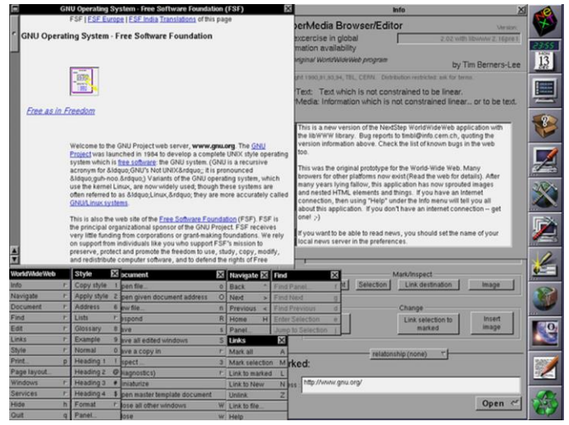
But again, given the fact that networks were only accessible locally, the texts could only be linked and accessed locally. So, if a researcher in one university wanted to consult a document from another university, she would have to take her car and drive over there, in order to log into the University's local network and finally access the document. Not very practical, is it?

A British scientist named Tim Berners-Lee, who worked at CERN, invented in 1989 a software solution that allowed people to publish content and make it accessible to anyone having the address, across all networks. That's why the Internet is often called: the network of networks. And thus, the first web site was born! You can still visit it at this address: http://info.cern.ch

Source : https://en.wikipedia.org/wiki/Tim_Berners-Lee

_A browser to surf on the web

Now, to visit this first website (and the many others which followed) one needed another piece of software. Tim Berners-Lee thus created the so-called "WorldWideWeb browser", the first Web browser. This first version was very basic, only allowing to show text and hyperlinks.



Screen capture of the first browser developed by Sir Tim Berners-Lee - Image author: Tim Berners-Lee (public domain) - Source: https://en.wikipedia.org/wiki/WorldWideWeb

_The Internet, today

Fast forward to 2021: modern web browsers can display not only text and links, but also images, videos, and run full-fledged applications!

Therefore, anyone can now not only consume but also create new content: the internet has become the main platform for most of our professional but also private lives. Communicating with friends and families, collaborating with colleagues, managing your bank account or planning your holidays... Everything is now happening online, thanks to the Internet!

_The Internet is not the Web

The Internet is the technical infrastructure that makes the Web possible. It is a network made of millions of computers spread around the world, connected together through many different types of connections (phone lines, fiber cables, ethernet cables, WiFi, Bluetooth…) and routers that make sure information transfers reach their destinations.

So, whereas the Internet is the infrastructure, the Web is a service built on top of the infrastructure. A service enabled by specific software called... web servers! When you visit a website using an internet browser, the browser acts as a "client", requesting the relevant content to the "web server".

As mentioned on MDN[1] (Mozilla Developer Network - one of the best resources to learn web development):

- “ Clients are the typical web user's internet-connected devices (for example, your computer connected to your Wi-Fi, or your phone connected to your mobile network) and web-accessing software available on those devices (usually a web browser like Firefox or Chrome).
- Servers are computers that store web pages, sites, or apps. When a client device wants to access a webpage, a copy of the webpage is downloaded from the server onto the client machine to be displayed in the user's web browser. “

Therefore, when you type or click on a web address, what happens is your browser triggers a request to the relevant web server, located at that address. The server receives the URL as a request and returns the relevant content to the "client" (your browser).

[1] Source: How the Web works? - MDN - https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

_HyperText Markup Language

We already mentioned the Hypertext as a great innovation. Let's have a closer look at the whole coding language that enables it: HTML (Hypertext Markup Language). HTML is very easy to learn... So, let's start with that!

c >Module 1

HTML

Introduction to HTML

@ *Why use HTML*

Computers and by extension web browsers are inherently dumb. We can't expect a computer to read a text document an make a distinction between titles, paragraphs, links, lists, etc.

Imagine now that you are a computer. Take a look at this document; you most likely won't understand anything.

Just like we can't give **meaning** to the data they get from the web. Therefore we use a **markup language** to add additional meaning to the text. On the web, the markup language we use is HTML or **hypertext markup language**.

By adding some <tags> to the document that describe the content, we have a better understanding of the meaning and structure of the text. The area of linguistics that is concerned with the meaning of text is called semantics. When writing HTML it is important

to use the correct HTML tags for your content. This is called writing semantic HTML, and its importance cannot be understated.

_*What is HTML?*

HTML is a coding language used to markup content. So, we say it is a *markup language*. Mark up means *that*, thanks to HTML, you can define the structure and the meaning of your content. According to MDN, ““HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to control its semantic and appearance. The enclosing tags can make a word or image hyperlink”[2] to a different place on the page or on the internet or point to a video file (which the browser can then render inside a video player), and so on. Let's take a simple example:

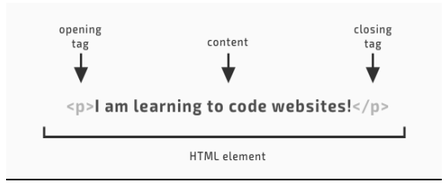
I am learning to code websites!

If we wanted the line to stand by itself, we would characterize it as a Paragraph, and thus wrap it as a Paragraph element:

** I am learning to code websites! **

_*HTML element (also called “HTML tags”)*

Let's look into this paragraph element a bit more closely. What is it made of?



As you can see, the main parts of our element are:

1. The opening tag: This includes the name element (here: a P tag) wrapped in opening and closing angle brackets. This tag states where the element starts — in this case where the paragraph begins.

2. The closing tag: This is similar to the opening tag, except that it includes a forward slash (/) before the element name. The closing tag states where the element ends. Forgetting the closing tag can lead to errors and strange results.

3. The content: This is the content of the element. Here it is just text, but it can be anything: items from a list, an image, a video or audio file, a section within the document….

4. The element: The opening tag, the closing tag and the content together form the element.

_*Nesting elements*

Just like Russian dolls, HTML elements can be nested in other HTML elements.

[2] HTML Basics - https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Let's continue with the HTML element from the previous example.
I would like to stress the word code. To do this I will use the element and wrap it around the word I want to emphasize.

** I am learning to code websites! **

In this example the element is **nested** in the element.

Important: *When nesting html elements, make sure to close the nested element before closing the parent element*

HTML 5

@ First HTML document

_File Extension

Your computer knows what application to launch to open a certain file by checking the extension of the file. Some examples are:

- Documents ending in **.docx** will be opened by **Microsoft Word.**
- Documents ending in **.pdf** will be opened with a **pdf reader.**
- A file ending in **.mp3** will be opened with a **music player.**

When you write HTML code, you want your code to be read by the **browser.** Your computer will automatically open all files ending in **.html** in the browser. The browser will then render the HTML code inside the document to a human-readable webpage.
If you want to view the pure HTML code, you will have to use a different type of application to open these documents: **a code editor**.

_Code editors

You could use the default text editor that is installed on your computer to write your HTML files. On Windows for example there is Notepad. That would work perfectly fine as long as you don't write any errors in your HTML code and save it as a .html file.

There are also applications created specifically to write code. These applications have built-in functionalities such as code highlighting, indentation of your code, autocomplete, etc. which help a lot with writing code efficiently.

@ File Structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Most Simple HTML page in the world</title>
  </head>
  <body>
    <h1>The Most Simple HTML page in the world</h1>
    <p>It might just be this page.</p>
  </body>
</html>
```

Example of a simple html document

Let's examine the above example of an HTML document.

- **<!DOCTYPE html>** The first line of a html document starts with the doctype declaration. This line informs the browser what version of html you are using.
- **<html>...</html>** All your html code will be placed inside the html element. The html element consists of an opening tag: <html>, and a closing tag: </html>.
- **<head>...</head><body>...</bod>** Inside the html element we have a **head** and a **body** element. The *head* element contains information about the document. It will not be visible to the human consulting your web page. The *body* element contains the actual content that will be visible on the website.
- **<title>...</title>** The title element is part of the head. As the name suggests, it contains the title for the html page.
- **<h1>...</h1>** The h1 element is placed in the body of the document. It is one of the 6 section headers in html. (h1-h6). "h" stands for "headline".
- ** ... ** The p element or paragraph element is used to display text.

Semantics

@ Why use semantics

Semantics happens when we attach a specific signification to a sign. Consider the image below.

We have learned to attach a meaning to this visual : an arrow pointing towards a direction.

Yet a dog for instance, or a cat, would not understand that from looking at the sign below.

In fact, it is hard for us humans to look at this sign and not infer its semantics because our brains love semantics!

We use semantics all the time, everywhere around. This helps us navigate our lives in cities for example. We learn conventions (such as the traffic signs) and rely on them to tell us what lies ahead. We see, we know.

In HTML, when we tell that a chunk of text is a Paragraph, the browser then "knows" how to render it in a specific way. For instance, a screen reader software can read the paragraph in a specific tone. This is particularly useful with images for instance. Properly adding syntax to an image tag can make or break the blind person's understanding of the content.

@ Semantic HTML

The HTML5 specification includes several semantic elements to help organize documents. Semantic elements are specifically designed to communicate meaning to the browser, developer, reader, and any other technologies interpreting the document (e.g. voice assistants, screen readers, search engines...).

_Section elements in HTML5

Section element are very useful to group semantically together, according to MDM[3], there are 4 different elements that you can use to identify sections of your document:

[3] Using HTML sections and outlines by Mozilla Contributors, licensed under CC-BY-SA 2.5. Source: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML_sections_and_outlines%23section_elements_in_html5

- "HTML Navigational Element (**<nav>**) useful for navigation links, very frequent on websites. You can have several menus on the same page, but you can never nest them, which means put a second **<nav>** element inside a **<nav>** element.
- HTML Article Element (**<article>**) defines a piece of content that can stand on its own.
- HTML Section Element (**<section>**) defines a section of a document to indicate a related grouping of semantic meaning.
- HTML Aside Element (**<aside>**) defines a section that, though related to the main element, doesn't belong to it, like an explanation box or an advertisement, or a "related products" list, for example. It should not be inside the main one (typically an **<article>**)".

_Other Semantic HTML elements used in Sectioning

The same Mozilla Contributor advances the following definitions for the elements used in Sectioning[4]:

- "HTML Body Element (**<body>**) defines all the content of a document. It contains all the content and HTML tags.
- HTML Header Element (**<header>**) defines a page which typically contains the logo, title, and navigation. The header can also be used in other semantic elements such as **<article>** or **<section>** — or section header, containing perhaps the section's heading, author name, etc. **<article>**, **<section>**,**<aside>**, and **<nav>** can have their own <header>. Despite its name, it is not necessarily positioned at the beginning of the page or section.
- HTML Footer Element (**<footer>**) defines a page footer which typically contains the copyright, legal notices and sometimes some links". This footer can be also defined as a section footer and can contain the section's publication date, license information, etc. **<article>**, **<section>**, **<aside>**, and **<nav>** can have their own **<footer>**. Despite its name, it is not necessarily positioned at the end of the page or section[5].

_Example layout

The example below is a layout for the body of blog page. There is a <header> with an <h1> element and a <nav> element which contains the navigation links. The <header> is on the same level as the <section> and <footer> elements. The main <section> element has a <h2> title, 2 <articles> and an <aside>.

```
<body>
    <header>
      <nav>
        <ul>
          <li><a href="#">link</a></li>
          <li><a href="#">link</a></li>
          <li><a href="#">link</a></li>
        </ul>
      </nav>
      <h1>
        Page Title
      </h1>
    </header>
    <section>
      <h2>
        My Blog Posts
      </h2>
      <article>
        <header>
          <p>
            Article Title
          </p>
        </header>
        <p>
          content
        </p>
      </article>
      <article>
        <header>
          <p>
            Article Title
          </p>
        </header>
        <p>
          content
        </p>
      </article>
      <aside>
        <p>
          Author info
        </p>
      </aside>
    </section>

    <footer>
      Copyright Info
    </footer>
</body>
```

[4] Ibid _11

[5] Information adapted from: https://learnabouthtml5.blogspot.com/2016/12/using-html-sections-and-outlines.html

d / Module 2

CSS

Introduction to CSS

In our previous journey, we have discovered the power of HTML. As you have seen, HTML was created to describe the content of a web page, like:

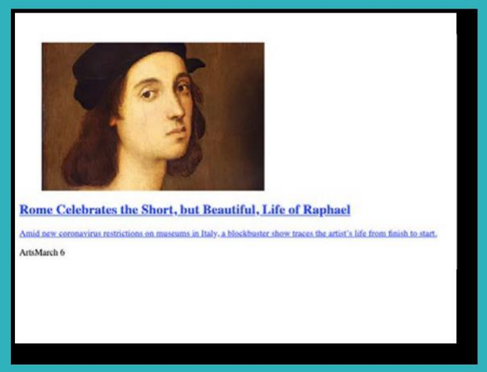
<h1>This is a heading</h1>

** This is a paragraph. **

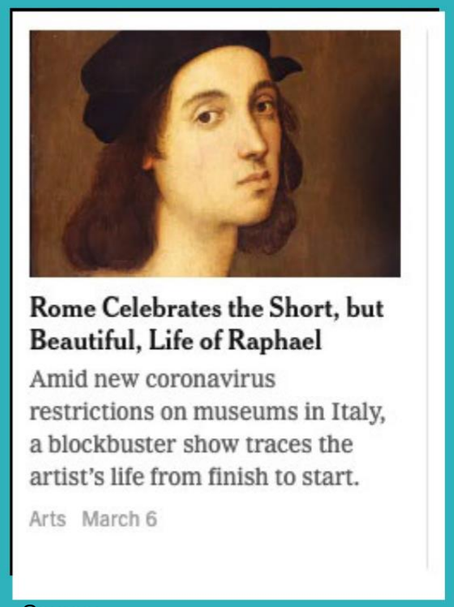
But you might have noticed that there is no color at all, it just looks like a plain word document. Which is boring to look at.

That's because on top of their HTML, these pages are using another language, called CSS, which job is to make the html look visually better.

In other words, CSS allows us to turn this:



Into that:



Source:
https://www.nytimes.com/2020/03/06/arts/raphael-rome-coronavirus.html

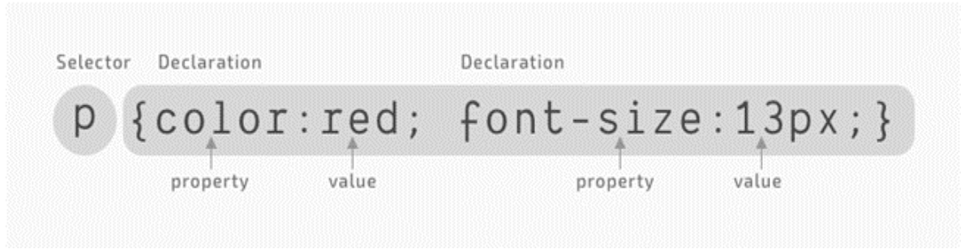
@ What is CSS?

- CSS Stands for Cascading Style Sheets
- CSS describes how HTML Elements are to be rendered on screen.
- It can control the layout of multiple pages at once.

In short, CSS is what makes our web pages look good and presentable. It's a must-have skill for any web developer out there.

@ The CSS Syntax

Look at this schematic, which sums all how to write CSS in a way that the browser understands it.



As you can see, there are fancy words here. Don't panic, there won't be much more :-)

"Selector" indicates which element(s) of your HTML file should receive the instructions. It basically points to the HTML element(s) you want to style. In this example, all tags on the html page will receive the CSS properties assigned to the pcss selector.

The **declaration** block contains one or more styling declarations, separated by semicolon character (;).

Each declaration includes a CSS property name and a value, separated by a colon (this character **:**).
A CSS declaration always ends with a semicolon (**;**), and declaration blocks are surrounded by curly braces.

Example:

p { color: red; font-size:14px; }

@ Getting started

There are three ways of implementing CSS in our web-pages.

_1. Inline CSS

Firstly, we can include CSS directly in our HTML elements. To achieve to this result, we make use of the style attribute, and write CSS code inside its value (the part in between the quotes). Example:

```
<h1 style="color: blue"> Hello world! </h1>
```

As you see, the css instructs the browser to use the color blue to render the h1tag. Easy, isn't it?
And you can combine several properties. For example, let's make it blue and bold:

```
<h1 style="color:blue;font-weight:bold">Hello world!

</h1>
```

Easy again! As you can see, you can add up properties using a semicolon (the character **;**) to separate them so the browser does not get confused.

We could add many more properties inside of this method but it can get really messy in our HTML file so it's not really recommended. The next two methods are much cleaner...

_2. The <style> block

Another way to include CSS is by using the **<style>** tag inside of our head section of our HTML page.

```
<head>

  <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>CSS is awesome! - BeCode</title>

    <style> h1{       color:blue; }

    </style>

    </head>
```

We just found a solution to not mix our HTML with our CSS, but our styling is still inside of our HTML file.

Maybe, there is a better way to include our CSS.? Well, yes there is!

_3. External CSS

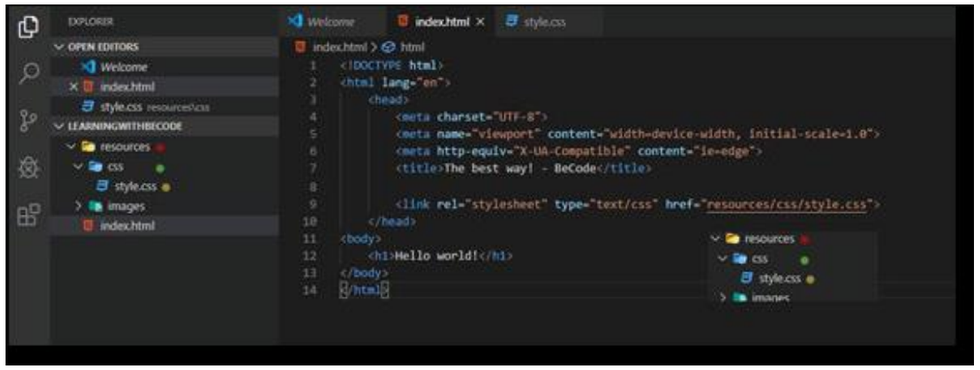
Like the name gives it away, we will have some external CSS file(s), which we will import inside the <head> of our HTML page.

```
<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge"> <title>The best way! -
BeCode</title>

<link rel="stylesheet" type="text/css" href="resources/css/style. css">

</head>
```

As you can see, we use a **<link>** tag this time to make a connection with our CSS file. This link tag will need a few attributes to work, the **rel=" stylesheet"** specifies the relationship between the HTML and CSS file, the browser knows now that we are trying to link a Stylesheet (CSS file). The **type=" text/CSS"** will tell the browser what kind of resource we are linking. It's not an obligation to use this, but we recommend using it to avoid any problems in the future. Last but not least, the **href="resources/css/style.css"** is our path that the link will use to find the document.

Having an external CSS file is the most recommended way to do, because it "separates concerns": the HTML file is for content, the CSS file is for decoration!

Here is an example of our folder structure:



Inside of our CSS file we have written the following:

h1{color:blue;}

This will give the same output as our result in example 1, the benefit of this is that our CSS is separated from our HTML and we can import this CSS file in multiple pages at once!

@ Working with colors

Colors are a big part of how things look. And we, humans, love color!

There are many millions of colors available in Nature.... Which was quite a challenge to transfer in the digital world of computers. First, there were only a few colors available, using predefined color names (like "red", "blue", "beige", "chocolate")...

_Named colors

The easiest way to start using colors is…by their names! They are easy to remember for us humans, especially for English speakers, since they are in English. There are currently 140 color keywords, including primary and secondary colors (such as red, blue, or orange), various gray (from **black** to **white**, including colors like **darkgray** and **lightgrey**), and a variety of other blended colors including **AliceBlue**, **DarkOrchid,** and **rebeccapurple**. The full list is available on W3Schools: https://www.w3schools.com/colors/colors_names.asp

It's nice to know they exist, but you will feel fastly limited by having only 140 colors.. So let's rather move on to the next way to express color values...

_The RGB system

Quickly computers became more powerful and able to manipulate millions of different colors, using a mix of the fundamental 3 colours of the screen: Red, Green and Blue, which is known as the "RGB system".

p {color: rgb (255, 0, 0) ; }

This says "I want the maximum of Red (maximum is 255), no green (0), and no blue (0).. Leading to a full bright red color.

This is exactly the same as

p{color: red;}

So if, for example, you really want that specific shade of blue that describes the Scottish sky in Spring, then you need to find its correct translation in the RGB system.

p{color: rgb(0, 182, 255);}

_Transparency!

You can also use a fourth value, to set the "alpha", which means "transparency" (or "opacity" if you prefer). Its value goes from 0 (totally transparent, the tag would be invisible) to 1 (fully opaque). Instead of RGB we use RGBA to add the transparent layer to our colour.

So let's say you want an orange square with 60% transparency, you would do this:

div { width: 100px; height: 100px; background-color: rgba(255, 221, 0,0.6); }

_The Hexadecimal system

For your information, there is yet another way to express the colour values, using the Hexadecimal system. In that system, red for example is expressed as **#FF0000, blackis #000000 and white: #FFFFFF**.

Hexadecimal system functions from values going from 0 to 9 and continues to A up to F for a total of 16 values. By using 6 Hexadecimal values, you are able to express 256 millions of colours.

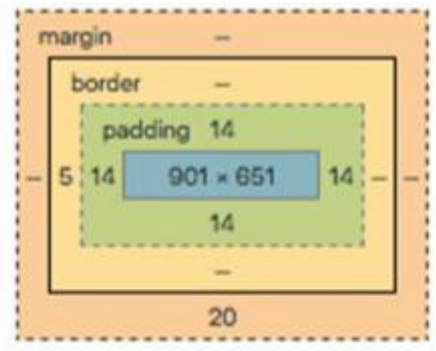
The first 2 digits describe the value of Red, the next 2 the values of Green, and the last two the values of Blue.

Just know that it exists and that you will be able to use it if you want. We will not really dig into that in this training.

@ Playing with borders

By default, without styling, each tag is rendered as a rectangle which background and borders are transparent. It does not have to stay that way!

Here is a visual representation of that rectangle, called the "box model".



This image represents how you can play with border, margin, padding to style any HTML tag!

Okay to explain this a little deeper, let's get our hands dirty!

Create an HTML file and copy these lines in our body:

```
<div class="box1"> <div class="box2"> </div> </div> <div class="box3">
</div>
```

Next create an CSS file and copy the following lines inside of this:

```
.box1{ width:200px;
    height:200px;
    border-top:1px solid red;
    border-right: 1px solid black;
    border-bottom: 2px dotted green;
    border-left: 2px dashed green;
    padding:100px;
    padding-right:50px;
    background-color: yellow;

    /*--We will cover this later--*/
    display:inline-block;
    /*----------------------------*/}

.box2{ width:200px;
    height:200px;
    background-color:red;

    /*--We will cover this later--*/
    display:inline-block; /*----------------------------*/}

.box3{
    width:100px;
    height:100px;
    background-color:green;
    margin-left:200px;
    /*--We will cover this later--*/
    display:inline-block;
    /*----------------------------*/}
```

That sure won't look exactly pretty, but that's not the concern yet.

As you can see, you can specify each border of the rectangle using 3 parameters: the thickness of the line (here, in pixels), the line type (solid, dashed, dotted), and its colour.

Now while you are at it, try to figure out the difference between padding and margin. Play with it's values, we will discuss this in group later on.

Borders can be used to turn rectangles into a square!

As you learn CSS, you will see that CSS is full of hacks and tricks. One really useful one is that you can turn an image like this:

into that:

© image: The Duffer Brothers. Source: https://www.strangerthings.fr

Here is the one property that makes it possible:

```
border-radius:50%;
```

You'll get a chance to experiment with it in the exercises....

@ Comments in CSS

Comments[6] are used to explain the code, to help and inform you (or a colleague) when editing the source code later. Comments are ignored by browsers.

A CSS comment starts with /* and ends with */:

/* This is a single-line comment */

p { color: red; }

You can add comments wherever you want in the code:

p { color: red;

/* Set text color to red */}

Comments can also span multiple lines:

/* This isa multi-linecomment */

p { color: red; }

@ Congratulations!

You just discovered the magic of CSS. Now let's put our knowledge to the test by doing a few exercises!

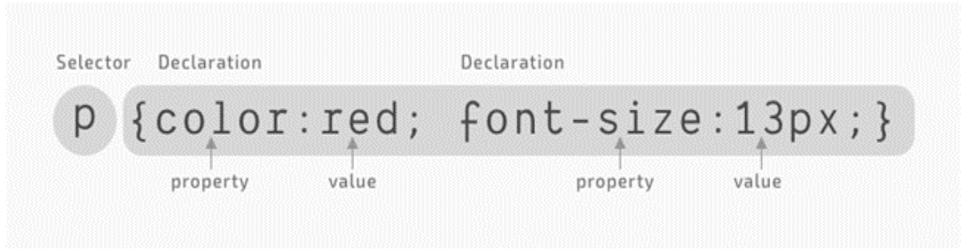
Selectors

@ CSS Selectors

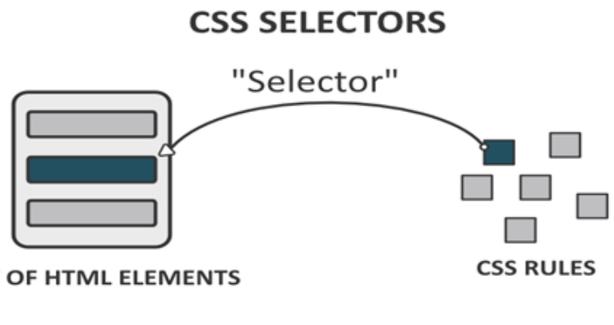
Welcome to a very important part of css, the selectors.

Do you remember this visual from the introduction?

[6] This section uses information adapted from https://www.w3schools.com/css/css_syntax.asp



A selector is the part of a CSS rule set that selects the content you want to style. Selectors are what allows the browser to know which html element(s) should be painted with which CSS properties.



For example I got this piece of HTML:

** **

** I am on my way to become a web-developer! **

** **

If I want to have a blue background on the first div, I could do this in CSS:

div{height:100px;

width:100px;

background-color:blue;}

However, if you do this, you will notice that both div elements will have a blue background.

That's why there are more selectors than just the div selector! For example, we have written class 'becode', why not use it in our css file?

We can select class elements like this:

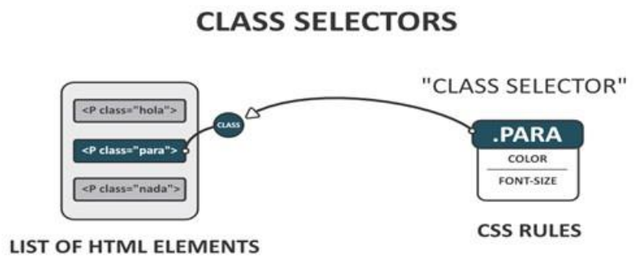
.becode{height:100px;

width:100px;

background-color:blue;}

Now only the background of our first div will be changed, just like we wanted!

What you did here, my friend, was using a class selector.



The Class Selector is indicated by a leading period (the .character) immediately followed by the class name you want to use. It can be anything, as long as it is the same as in the HTML file.

So, if in the html, the class name is "burger", the CSS class selector will be .... .burger. If the html class name is "Ferrari", the CSS class selector becomes.... .Ferrari. Got it?

There are many ways to select html elements using the CSS selectors. Too many for this course. Know that the class selectors are the most used and will allow you to select everything you need, especially if you also know how to use...

_Pseudo-class selectors

Most elements like H1, P ... are not meant to be interactive. But some are, like links which in HTML are called Anchors and are coded using the Atag.

Examine this CSS, styling all anchors on your page so that they look like the IKEA logo.

a.ikea {text-decoration:none; /* remove underline */

background-color: blue; /* paint the background blue */ color:yellow; /* paint the text yellow */ font-weight:bold; /* make the characters bold */ text-transform: uppercase; /* turn characters in UPPERCASE */ padding:5px 10px; /* add some padding */ font-size:23px; /* make the text bigger */}

So now, in your html, you can add the class "ikea" to all links that you want to look like they were bought at IKEA :-)

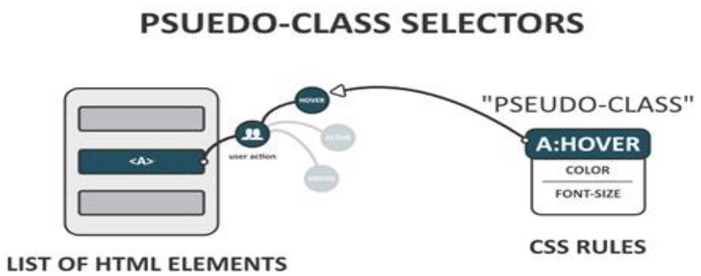
** I looooooove design, especially furniture from Aeki. It's cheap and stylish... Okay, if you actually manage to build them!! **

Result:

I looooooove design, especially furniture from AEKI. It's cheap and stylish... Okay, if you actually manage to build them!!

Cool, but if you go with your mouse on top of it, nothing happens. Sad, for an interactive element!

That's where Pseudo-class enters the game...



CSS “pseudo-classes” make it possible for you to control the look of intermediary/temporary states of an element. For instance, an <a href> element can be in a number of different states (a link can be hovered on with the mouse, “visited” if it points to a file that was previously visited), and you can use pseudo-classes to style them individually. They are built into the browser so you need to know them in order to use them.

_Basic link styles

Oliver James (InternetingIsHard.com) summarizes[7] the syntax as such.
“Pseudo-classes begin with a colon followed by the name of the desired class. The most common link pseudo-classes are as follows:

- **:link –** A link the user has never visited.
- **:visited –** A link the user has visited before.
- **:hover –** A link with the user’s mouse over it.
- **:active –** A link that’s being pressed down by a mouse (or finger).”

That's about it for the CSS Selectors. You have enough information to do the exercises! Don't hesitate to go back here if you cannot manage the exercises - you cannot remember all this. Practice will help you remember ;-)

Basic properties

We know how to call our elements in css, we have used some basic properties, but there are way many more properties available to you, Picasso! In this section you will learn to use a few more interesting properties through short exercises.

[7] source: https://www.internetingishard.com/html-and-css/css-selectors/#pseudo-classes-for-links

Positioning

The big part of your job is to place elements on the screen, and control how they should adapt to the adjacent content as well as to the screen size (mobile phone or Desktop computer). In a word: positioning

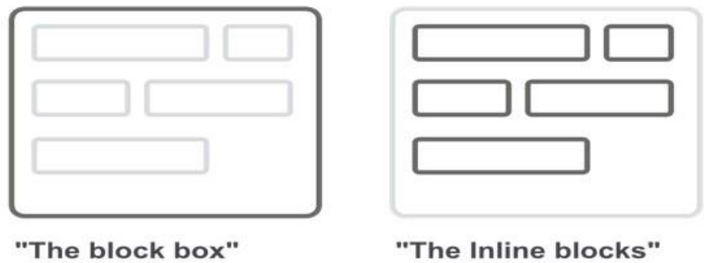
CSS offers several ways to control the positioning of the elements on the screen. Let's start by giving you an understanding of how the element itself is laid out.

@ Boxes and blocks

You can imagine each HTML element as a box, we have 2 different versions of these so called “boxes”.

- A block box
- A Inline box

Have a look at this visual explanation of the difference between a block and an inline element.



Each HTML element has a default type of box, you will see that not all elements share the same box type in the end! For example, , <h1>, <p>, <ul>,… are block elements whilst <img>, <a>,…are inline elements.
These are so-called “boxes” are important for the flow of our webpage.

_Block elements

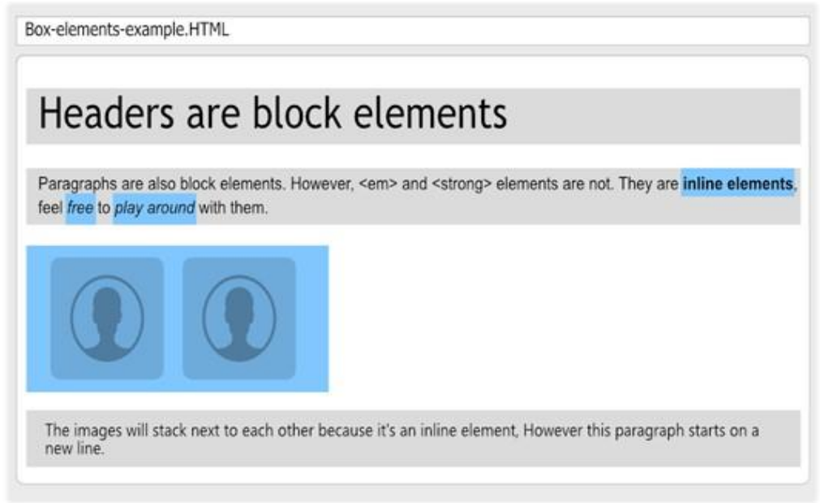
A block element usually takes the whole width of the parent container. As a result, it will force the next block element onto a new line, each block element will appear below the other.

The height of a block element will always be decided by its content - if the text doesn't fit on the screen, its default behavior is to collapse the text onto a new line.

_Inline elements

Inline elements behavior is much different than Block elements, they are used to style items inside block elements. Unlike the Block element, which takes the full parent width, the Inline element width is always based on its content.

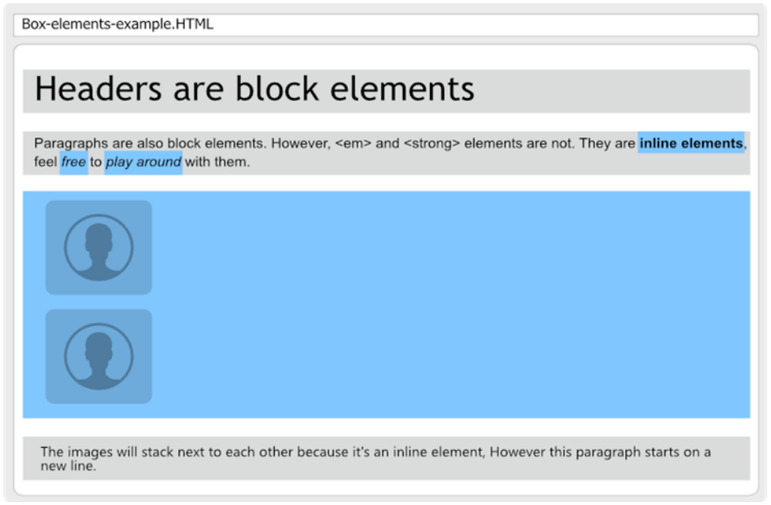
Take a look at the visual below - You will see that the grey parts (block elements) take the full width of the body. Whilst the blue parts (inline elements) only take the width of its content.



_Changing box type behaviour

In CSS we are fully in control of how our elements should display, this means we can change our elements default behavior as well! Maybe you want your images to stack below each other, that can be easily done by using the display attribute.

```
Img {
        display:block;
}
```

@ The CSS Box Model

There is one thing that the block and inline elements have in common - and that is the CSS Box Model. What is that I hear you say, another box? Yep, well not really - it's still the same box we were talking about earlier. Let’s take a close look at what is included in this so-called "box".

While it may sound and look complicated, it's actually easy to understand. This box model is used to calculate how much space there is between/inside each element or if needed, to add a border around the element.

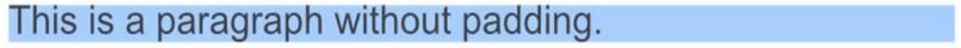
This is perhaps a reason why people sometimes say - HTML is like building with lego bricks.

Let's take a look to the box model visually

1. Content: This is what matters to most the developer that writes HTML, this is the only part that has a semantic value. The next parts are purely visual for the box model.
2. Padding: This will decide how much space there is between the content and its border.
3. Border: A-line (by default transparent) between the padding and margin.
4. Margin: The space between this element and its surrounding elements.

_Padding

We've already seen the content part, so let's move one level up to padding. This property will decide how much padding there will be for the selected element:
Here we have an example:




```
p {      padding: 10px;}
```

As you can see the blue background behind the p element has increased. That is because we told our CSS to increase the padding by 10px! Now you see it's doing 10px for all sides, but sometimes we want to style one side only or have different values everywhere.
We can easily do this by doing the following:

```
p {
padding-top: 20px;

padding-bottom: 20px;

padding-left: 10px;

padding-right: 10px;
 }
```

You can see the slight difference of the top and bottom padding compared to the left side.
Padding works on block and inline elements.

_Borders

Now the next property of the box model is the Border. By default, most elements have a transparent border. If you would like to assign a border to your element, you would have to follow this CSS syntax: Specify the width of the border, then which type of border it should have, and last its colour.
Let's add a border to our previous padding example!

```
p {
  border: 4px solid #5d6063;
}
```

Just like padding, there are -top, -bottom, -left, and -right variants for the border property:

Example:

```
p {
        border: 1px solid #5D6063;
}
```

This would only draw 1 border on the bottom of the element.
Borders works on block and inline elements.

_Margins

We have reached our last Box Model property, margins. Margins are the transparent spaces around an element, so don't confuse it with padding. Padding takes care of spacing inside an element where margin pushes away other elements around the selected element.

```
p {
        padding: 20px 10px 20px 10px;
         margin-bottom: 20px;
}
```

.

And just like padding and borders, you can target one or more sides to make changes on. Margins work only on block elements.

_Changing widths or heights

Up until now, we let the browser decide how much width our elements should be, but sometimes you want to set the width yourself. Maybe an image is way too big, or there is text that should only cover 20% of your screen. For this, we can use the width and height properties within CSS. This will change the default width of your box's content.
Let’s give our paragraph a width of 200px.

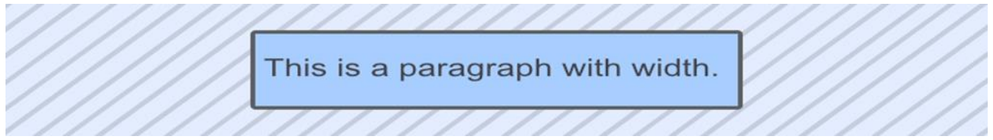
```
button {
  width: 200px;
}
```

Instead of taking the full browser width, our paragraph is now 200 pixels, but it hugs the left side of our page...
Width/height works only on block (or inline-blocks) elements.

_Centering with auto-margins

To avoid our element sticking to the left, we can use the auto-margins value inside our margin property. This will automatically make the space around your block element and divide it equally on the left and right sides.
We can center our paragraph as follows:

```
p {
  margin: 20px auto;
}
```



This will only work when a width has been set.

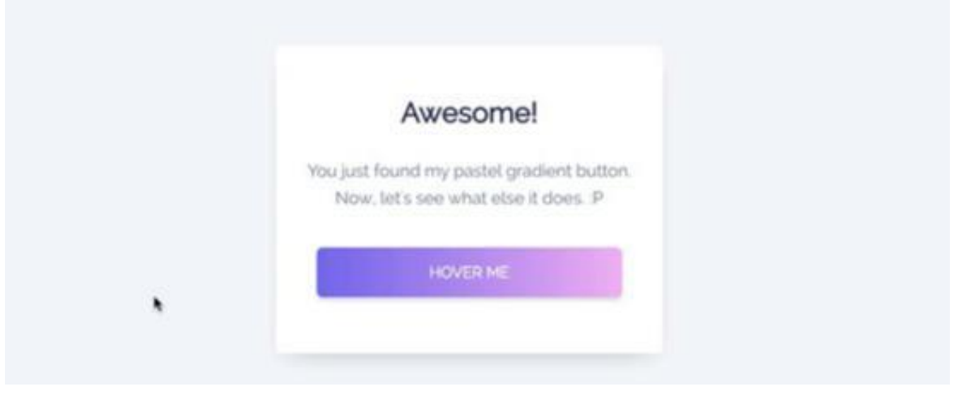
@ Summary

- Everything inside a HTML page is a box.
- Boxes can be inline or block elements.
- Those boxes have content, padding, borders, and margins.
- Once you know the Box Model, it`s easier to master layouts of web pages.

The CSS properties we just covered in this chapter might have appeared simple — and they are. Yet they are very powerful: have a look at the websites you visit through the lenses of the CSS box model, and you’ll see that they are used everywhere.
All right, let's go ahead and continue learning by doing a few positioning challenges using 3 techniques: Display, Position and Flexbox!

CSS Animations & Transitions

We are almost through the CSS part, well done! But there is still something that we have not covered yet. What if you want to move elements around? Or what if you want to animate your beautiful button just like the example below:



Well, all of this is possible with Transitions and Animations!

@ Transitions

Without transitions all changes on a CSS element will happen instantly. Let's imagine we use a :hover pseudo-class. Normally the hover effects will happen instantly, as soon as the mouse hovers over the element. But when we add a transition, we create a smooth transition between the two states of the element.
You can use transitions with widths, heights, transforms...

_How to use a transition in CSS?

Have a look at the following CSS snippet:

```
button {
  color: red;
  transition: color 0.5s ease-in-out 1s;
  /*transition: (property) (duration) (transition-timing-function) (delay)*/
}
button:hover{
  color:blue;
}
```

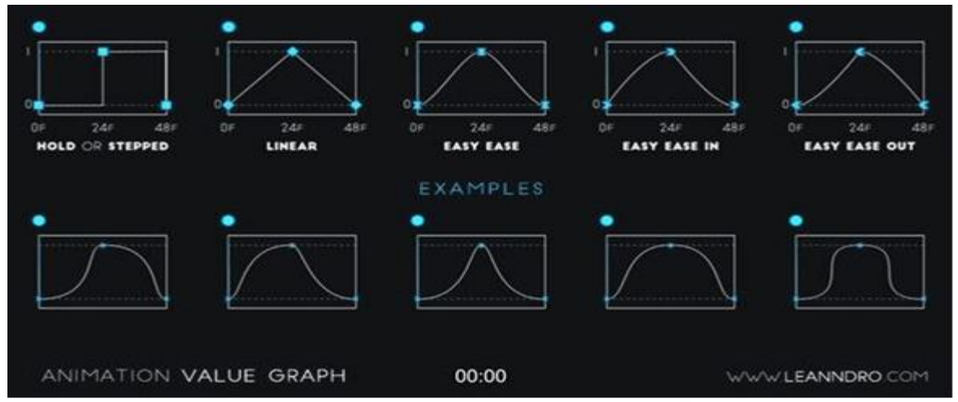
Have a look at the following CSS snippet:
If we would use this CSS it would result in a button that has a font color of red, but when we hover it, it will wait 1 second, after that, it smoothly changes the color red to blue over a period of 0.5 seconds.
Let's go over the different values

- Property: What do you want to animate? Width, height, color,...
- Duration: How long should the transition last? 0.5 seconds, 1 second, ...
- transition-timing-function: What is the speed of the transition (see below for more info).
- Delay: How much delay should there be when the transition triggers

You can use transitions without using all properties, if you don't assign any values, it will use it's default value.
These are the default values:
transition: all 0 ease 0;
As you can see, in order for your transition to work you can leave all values empty except from the duration value, otherwise there will be no smooth transition.
Note: The all value will animate all different properties if they change. While this can be easy to animate everything quickly, it can lead to strange and unwanted behaviors.

_Transition timing

Sometimes animations feel boring and lifeless, if you want to make changes to that, you most likely want to look into the transition-timing-function.
This will decide the "rhythm" of your animation. Does it start slow or fast ? Does it end slowly ? Not only is this better for the eye, it makes an animation feel more natural.
Take a look at this animation value sheet:



Leanndro www.leanndro.com - Animation Value Graph - Timing
Source: http://spungella.blogspot.com/2016/03/animation-value-graph-timing.html

- The top row shows you the curve and animation behind each value.
- The bottom one shows animations of when we would manipulate this curve.

Amazing how we can change the feeling by changing a curve, right?

Feel free to experiment with other easing functions (search for them in the Documentation - remember searching is part of the job of a professional developer! ;-))

@ Animations

You would think an animation is the same as a transition: they both change the property values. Whilst the transition can change over only two states, an animation can transition over as many states as you want.
In order to use CSS animations you have to use "keyframes", which contains an element's property and of course the value to what it should change to. You can use as many keyframes as you want, to make really smooth and long animations.

_How do we use @keyframes?

First of all, we have to design the keyframes animation, this can be achieved by simply creating a **@keyframes** rule, next give it a name so we can call it later in our elements.
The next step is to define from and where to our animation should go. You will see it the following example,

- That we start from a font-size of 50px and the color red.
- And that we go to a font-size of 100px and the color blue.

```
/* The animation code */
from {
   font-size: 50px;
   color: red;
 }
 to {
   font-size: 100px;
   color: blue;
 }
}

/* The element to apply the animation to */
p {
 color: red;
 animation: test 4s infinite alternate ease-in-out;
```


This would result something like this:

As you can see, we use keywords **"from" (0%)** and **"to" (100%)** - which is only 2 states. But as I have told you earlier, it's possible to animate many more states.
This can be achieved by using percentages. Take a look at the following example:

```
/* The animation code */
@keyframes test {
 0% {
   font-size: 0px;
   color: red;
 }
 25% {
   font-size: 100px;
   color: pink;
 }
 50% {
   font-size: 30px;
   color: green;
 }
 75% {
   font-size: 120px;
   color: yellow;
 }
 100% {
   font-size: 0px;
   color: blue;
 }
}

/* The element to apply the animation to */
p {
 color: red;
 animation: test 4s infinite alternate ease-in-out;
}/
```

You can see the result at this link:
https://github.com/becodeorg/free2code/blob/master/2.CSS/4.CSS-ANIMATIONS/exercise-09/exercise.md
As with transitions, you will have to assign a duration value, otherwise no animation will be played.

Conclusion

Animations are pretty cool and fun, aren't they? I could tell you much more about transitions and animation, but that is not the point here! Let's dive into the documentation and get your hands dirty! Feel free to experiment and break things!

Source: https://tenor.com/view/monsters-inc-typing-group-chat-is-lit-gif-11492598

References (as they appear in the text)

1. Portrait of Tim Berners-Lee - © Paul Clarke, CC BY-SA 4.0, via Wikimedia Commons Source: https://en.wikipedia.org/wiki/Tim_Berners-Lee
2. Screen capture of the first browser developed by Sir Tim Berners-Lee - Image author: Tim Berners-Lee (public domain) - Source: https://en.wikipedia.org/wiki/WorldWideWeb
3. How does the Internet work? by Mozilla Contributors licensed under CC-BY-SA 2.5. Source: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_does_the_Internet_work
4. HTML Basics by Mozilla Contributors and is licensed under CC-BY-SA 2.5. Source: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
5. Using HTML sections and outlines by Mozilla Contributors, licensed under CC-BY-SA 2.5. Source: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML_sections_and_outlines%23section_elements_in_html5
6. https://learnabouthtml5.blogspot.com/2016/12/using-html-sections-and-outlines.html
7. Rome celebrates short but beautiful file of Rafael. Source: https://www.nytimes.com/2020/03/06/arts/raphael-rome-coronavirus.html
8. Full list of colors available on W3Schools: https://www.w3schools.com/colors/colors_names.asp
9. The Duffer Brothers. Source: https://www.strangerthings.fr
10. Comments on CSS, adapted from: https://www.w3schools.com/css/css_syntax.asp
11. Oliver James on Pseudo-classes. Source: https://www.internetingishard.com/html-and-css/css-selectors/#pseudo-classes-for-links
12. Animation Value Graph - Timing - Leanndro www.leanndro.com - Source: http://spungella.blogspot.com/2016/03/animation-value-graph-timing.html
13. Monster Inc Typing GIF. Source: https://tenor.com/view/monsters-inc-typing-group-chat-is-lit-gif-11492598

Annexes

Session Plan – Module 0

Training: Free to Code – Improving Digital and Coding Skills for Inmates		
Module: 0 – General introduction		*Trainer:*
Session nr. : 0 *(half of a session)*	Duration: 1h	Date:
General objectives:	The learners (prisoners from 18 to 99 y.o.) will be able to modify an existing webpage and create basic web pages and using HTML and CSS with a code editor. They will also acquire basic on how to describe a solution to simple web development problems in logical steps and understand what each frontend technology is best used for (html, css and Javascript).	
Learning outcomes:	• Frontend coding fundamentals: Fix webpage and create a basic webpage using HTML and CSS • Describe a solution to simple problems in logical steps • Basic terminology of frontend web development technologies • Be able to use the Coding interface provided • Algorithmic and problem solving	
Methods:	Expository, interrogative, and active. Computer-based self-learning based on active learning techniques, such as project-based learning, quizzes and learning by doing, under the guidance of a technical monitor – blended learning.	
Content:	• Background • Internet • Web	

Time	Activities:	Resources and materials:
10m	Icebreaking activity Wool ball activity – put the trainees and trainers in circle, and ask them to introduce themselves (name, age, where they are from) and then pass the ball to whoever they wish The purpose of this activity is in the end to say that they have started a network of contacts at this time and that they must keep with each other in order to help each other now and in the future - the training aims to be self-learning and the trainees must help each other to succeed. This activity also serves as the starting point of the next brainstorming – analogy between the network that was created with the wool ball and technology to begin the brainstorming questions (see below)	• Ball of wool and space in the room to create a circle with all participants
10m	Brainstorming regarding Technology to open the discussion and to introduce the topic Questions suggestions: •*"What do you see when thinking in Technology?"* •*"Makes our lives easier?"* •*"What is an website? And an app?"* "*Where does coding stand in the technology world?"* *Other questions you might find relevant* Presentation of the main topics covered in the learning part of the module 0.	• Flipchart and pens or • Word document • Etc.
40m	1. Background a. The Internet – 2. b. How does it work, network of networks and the web Conclusion, discussion, and questions *(to be continued this 1st session in the next table)*	• Learning platform
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) Trainers must gather participants' feedback as well as take their own notes regarding the session.	
Activity (Time):	Lessons 1-3 (1h)	

Session Plan – Module 1

Training: Free to Code – Improving Digital and Coding Skills for Inmates		
Module: 1 – HTML		*Trainer:*
Session nr. : 0 to 2 *(2 and half sessions)*	Duration: 8h	Date:
General objectives:	The learners (prisoners from 18 to 99 y.o.) will be able to modify an existing webpage and create basic web pages and using HTML and CSS with a code editor. They will also acquire basic on how to describe a solution to simple web development problems in logical steps and understand what each frontend technology is best used for (html, css and Javascript).	
Learning outcomes:	• Frontend coding fundamentals: Fix webpage and create a basic webpage using HTML and CSS • Describe a solution to simple problems in logical steps • Basic terminology of frontend web development technologies • Be able to use the Coding interface provided • Algorithmic and problem solving	
Methods:	Expository, interrogative, and active. Computer-based self-learning based on active learning techniques, such as project-based learning, quizzes and learning by doing, under the guidance of a technical monitor – blended learning.	
Content:	• Intro • HTML5 • Semantics	

Time	Activities:	Resources and materials:
1h	**Session 0 (continuation)** The training of the 1st module will be divided in two parts - a presentation part, and exercises-part. Presentation of the 3 main topics covered in the learning part of module 1: 1. Intro a. Why and what is HTML? 2. HTML5 a. First HTML document 3. Semantics a. Why use semantics, semantic HTML	• Platform – learning part
1h	After the presentation trainees must do the exercises designated for this 1st training module. The exercises should be carried out by the trainees, but always supported by the trainers, and with continuous access to the learning part of the platform so that they can review content that helps them perform the exercises. *Exercises:* • Exercise 1 – hello world (15 minutes) • Exercise 2 – Titles (15 minutes) • Exercise 3 – Paragraphs (15 minutes) • Exercise 4 – Bold Cursive (15 minutes) *After 3 hours close the first session with conclusion, discussion, and questions*	• Platform – learning part
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises and Final Challenge) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	

Time	Activities:	Resources and materials:
3h	**Session 1** Review of the learning contents discussed before and keep working in the exercises. The exercises should be carried out by the trainees, but always supported by the trainers, and with continuous access to the learning part of the platform so that they can review content that helps them perform the exercises. Act accordingly for the following sessions of all the modules. *Exercises:* • Exercise 5 – Lists (30 minutes) • Exercise 6 – Relative Links (30 minutes) • Exercise 7 – Images (30 minutes) • Exercise 8 – The letter 30 minutes) • Exercise 9 – Subjects of study (30 minutes) • Exercise 10- Exotic dance moves (30 minutes) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises and Final Challenge) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	
3h	**Session 2** *Exercises:* • Exercise 11 – Recipe formatting (1h) • Exercise 12 – Chinese farmer (2h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises and Final Challenge) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	
Activity (Time): *Lessons 1-3 (1h)*	Exercise 1 – hello world (15 minutes) Exercise 2 – Titles (15 minutes) Exercise 3 – Paragraphs (15 minutes) Exercise 4 – Bold Cursive (15 minutes) Exercise 5 – Lists (30 minutes) Exercise 6 – Relative Links (30 minutes) Exercise 7 – Images (30 minutes) Exercise 8 – The letter (30minutes) Exercise 9 – Subjects of study (30 minutes) Exercise 10 – Exotic dance moves (30 minutes) Exercise 11 – Recipe formatting (1h) Exercise 12 - Chinese farmer (2)	

Session Plan – Module 2

Training: Free to Code – Improving Digital and Coding Skills for Inmates		
Module: 2 – CSS		*Trainer:*
Session nr. : 3 to 18	Duration: 43h	Date:
General objectives:	The learners (prisoners from 18 to 99 y.o.) will be able to modify an existing webpage and create basic web pages and using HTML and CSS with a code editor. They will also acquire basic on how to describe a solution to simple web development problems in logical steps and understand what each frontend technology is best used for (html, css and Javascript).	
Learning outcomes:	• Frontend coding fundamentals: Fix webpage and create a basic webpage using HTML and CSS • Describe a solution to simple problems in logical steps • Basic terminology of frontend web development technologies • Be able to use the Coding interface provided • Algorithmic and problem solving	
Methods:	Expository, interrogative, and active. Computer-based self-learning based on active learning techniques, such as project-based learning, quizzes and learning by doing, under the a technical monitor – blended learning.	
Content:	• Introduction to CSS • Selectors • Basic Properties • Positioning • CSS Animations • Final Challenge	

Time	Activities:	Resources and materials:
2h	**Session 3** The training will be divided in two parts - a presentation part, and exercises-part. Presentation of the topics covered in the learning part of the Platform: 1. Introduction to CSS a. What is, syntax, getting started, working with colours, borders, comments 2. HTML5 a. CSS selectors, class selectors, pseudo-class selectors 3. Basic properties 4. Positioning a. The box model, padding, borders, margins, dimensions 5. CSS animations a. Transitions, animations	• Platform – learning part
1h	**Session 4** *Exercises:* • Exercise 1 – Add some inline CSS (30 minutes) • Exercise 2 – Plan your vacations with style (30 minutes) *After 3 hours close the first session with conclusion, discussion, and questions*	• Platform – Training Part
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 5** *Exercises:* • Exercise 3 – Canary Islands (30 minutes) • Exercise 4 – Classes to differentiate elements and style them differently (30 minutes) • Exercise 5 – Borders (30 minutes) • Exercise 6 – Class Selector (1h) • Exercise 7 – Pseudo-class selector (30 minutes) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	

Time	Activities:	Resources and materials:
3h	**Session 6** *Exercises:* Exercise 8 – Style this page (30 minutes) Exercise 9 – Paragraphs (30 minutes) • Exercise 10 – CSS Diner (30 minutes) • Exercise 11 – Basic properties (30 minutes) • Exercise 12 – Display properties exercise 1 (1h) *After 3 hours close the session with conclusion, discussion, and Questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 7** *Exercises:* • Exercise 13 – Display properties exercise 2 (1h) • Exercise 14 – Position properties exercise 1 (2h) *After 3 hours close the session with conclusion, discussion, and Questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 8** *Exercises:* • Exercise 15 – Position properties exercise 2 (1h) • Exercise 16 – Position properties exercise 3 (1h) • Exercise 17 – Flex-box 1 (1h) *After 3 hours close the session with conclusion, discussion, and Questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	

Time	Activities:	Resources and materials:
3h	**Session 9** *Exercises:* • Exercise 18 – Flexing card (2h) • Exercise 19 – Flexing page (1 of 3h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 10** *Exercises:* • Exercise 19 – Flexing page (2 of 3h) • Exercise 20 – CSS integration series (1 of 4h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 11** *Exercises:* • Exercise 20 – CSS integration series (3 of 4h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	

Time	Activities:	Resources and materials:
3h	**Session 12** *Exercises:* • Exercise 21 – CSS Animations (1h) • Exercise 22 – Small CSS animations series (2h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session.	
3h	**Session 13** *Exercises:* • Exercise 23- Small CSS animations series 1 (2h) • Exercise 24 – Small CSS animations series 2 (1 of 3h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session..	
3h	**Session 14** *Exercises:* • Exercise 24 – Small CSS animations series 2 (2 of 3h) • Exercise 25 – Small CSS animations series Bonus (1 of 4h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the “Satisfaction form” of “How to setup your pilot” support document. Trainers must gather participants’ feedback as well as take their own notes regarding the session..	

Time	Activities:	Resources and materials:
3h	**Session 15** *Exercises:* • Exercise 25 – Small CSS animations series Bonus (3 of 4h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	
3h	**Session 16** *Exercises:* • Exercise 26 – Final Challenge (3 of 9h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	
3h	**Session 17** *Exercises:* • Exercise 26 – Final Challenge (3 of 9h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session.	

Time	Activities:	Resources and materials:
3h	**Session 18** *Exercises:* • Exercise 26 – Final Challenge (3 of 9h) *After 3 hours close the session with conclusion, discussion, and questions*	• Platform – learning and training parts
Assessment:	Formal assessments on the learning outcome (quizzes, Platform exercises and Final Challenge) – if there is any drop out, they must fill the "Satisfaction form" of "How to setup your pilot" support document. Trainers must gather participants' feedback as well as take their own notes regarding the session. To close the training, trainers must fill "Observation & feedback template" and trainees "Satisfaction form" available in the "How to setup up your pilot" support document	
Activity (Time): *Lessons 1-5 (1h)*	Exercise 1 – Add some inline CSS (30 minutes) Exercise 2 – Plan your vacations with style (30 minutes) Exercise 3 – Canary Islands (30 minutes) Exercise 4 – Classes to differentiate elements and style them differently (30 minutes) Exercise 5 – Borders (30 minutes) Exercise 6 – Class Selector (1h) Exercise 7 – Pseudo-class selector (30 minutes) Exercise 8 – Style this page (30 minutes) Exercise 9 – Paragraph (30 minutes) Exercise 10 – CSS Diner (30 minutes) Exercise 11 – Basic properties (30 minutes) Exercise 12– Display properties exercise 1 (1h) Exercise 13 – Display properties exercise 2 (1h) Exercise 14 – Position properties exercise 1 (2h) Exercise 15 – Position properties exercise 2 (1h) Exercise 16 – Position properties exercise 3 (1h) Exercise 17 – Flex-box 1 (1h) Exercise 18 – Flexing card (2h) Exercise 19 – Flexing page (3h) Exercise 20 – CSS integration series (4h) Exercise 21 – CSS Animations (1h) Exercise 22 – Small CSS animations series (2h) Exercise 23 - Small CSS animations series 1 (2h) Exercise 24 – Small CSS animations series 2 (3h) Exercise 25 – Small CSS animations series Bonus (4h) Exercise 26 – Final Challenge (9h)	

Exercises_Module 0 and Module 1

1.	**Hello world**	Take a look at the index.html file. At the moment the file is empty. Add all the necessary tags for the base layout of an html document: a doctype, an <html> element and a <head> and <body> element. Be careful though, you can't just place these elements wherever you like. They have a function and should be placed in a specific order. Read the documentation if you're not sure how to use these html elements. Finally add a <p> element with the text 'hello world' to your document. The text should be displayed in the browser.
2.	**Titles**	Time to start exploring more html elements. Look at the index.html file and the result in the browser. There should be nothing displayed. Add a <h1> element with the text 'title from an h1 tag' and look at the result. There is also an <h2> element, repeat the same process for this tag as well. Continue doing this for all <h#> elements.
3.	**Paragraphs**	Add 6 paragraph elements to the index.html file, one for each title tag. You can write something yourself or copy some placeholder text from here: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quod saepe architecto laborum omnis reiciendis unde et voluptatibus repudiandae eaque quibusdam doloribus officia odit voluptatum minima ab, reprehenderit cupiditate. Consequuntur, sapiente."
4.	**Bold Cursive**	Right now you should have a html document with 6 titles and a paragraph for each title. We can add some extra meaning to the text by adding more html elements. Make use of the and tags to emphasize one or more words in your text. You will notice that the browser adds some styling to the text wrapped in an or tag. Make sure to read the documentation for these html elements.
5.	**Lists**	Open the index.html file and compare the html code to the output of the document. Identify the 3 different types of lists: ordered, unordered and definition lists. Make sure to check the documentation of these html tags! In the <body> element, add one list of each type with your own content underneath the example lists.
6.	**Relative links**	The true goal of the web was to have a big network of files connected to each other by hyperlinks, that's why we use the HyperText Markup Language (html). We can achieve this by using the <a> tag, or anchor tag, to link to a different file. For the <a> tag to know where to link to, we will have to provide it with some extra information. We do this by using the 'href' attribute. An attribute is always added in the opening tag of an element. To link to a different page on your web domain, you use relative links. In that case the href attribute holds the path to the document you're linking to. For example: Link to a page The above html element will link to the html document in the pages folder named pagename.html. Between the opening and closing tag you can provide the text for the hyperlink. In the index.html file, add a relative link to the about page in the pages directory, as well a target, so when we click the link it opens a new tab.
7.	**Images**	Up until now we have only added different forms of text to your html document. Titles, paragraphs, lists, ... Most websites nowadays are a little more visually pleasing than the black on white textpage we have right now. One way to brighten up your website is by adding images. To do this we use the <img> html element. Unlike the previous tags we have used, the <img> tag does not have an opening and closing tag. It consists of one self-closing tag, written as follows: <img/> Inside this tag we can add different attributes. The following attributes can be added to an img tag: 'src', 'width', 'height'. 'src' attribute: the src, or source, attribute tells the image tag what picture it should display. 'width' attribute: sets the width of the image. 'height' attribute: sets the height of the image. Try adding the 'funny-goose-sitting-Karen-Arnold.jpg' to this img tag, also fill in the other blank attributes. Image copyright information: Karen Arnold - public domain. Source: https://www.publicdomainpictures.net/nl/view-image.php?image=34335&picture=funny-goose-zittend

Convert the following letter to valid HTML by making use of semantic HTML elements. (Source: Marking up a letter by Mozilla contributors, licensed under CC-BY-SA 2.5[8])

Block/structural semantics:	***Inline semantics[9]:***	***The head of the document:***
● You should structure the overall document with an appropriate and valid structure including doctype, and <html>, <head> and <body> elements. ● The letter in general should be marked up with a structure of paragraphs and headings, with the exception of the below points. There is one top level heading (the "Re:" line) and three second level headings. ● The semester start dates, study subjects and exotic dances should be marked up using an appropriate list type. ● The two addresses should be put inside <address> elements. Each line of the address should sit on a new line, but not be in a new paragraph.	● The names of the sender and receiver (and "Tel" and "Email") should be marked up with strong importance. ● The four dates in the document should be given appropriate elements containing machine-readable dates. ● The five acronyms/abbreviations in the main text of the letter should be marked up to provide expansions of each acronym/abbreviation. ● The six sub/superscripts should be marked up appropriately — in the chemical formulae, and the numbers 103 and 104 (they should be 10 to the power or 3 and 4, respectively). ● Try to mark up at least two appropriate words in the text with strong importance/emphasis. ● There are two places where a hyperlink should be added; add appropriate links with titles. For the location that the links point to, just use http://example.com. ● The university motto quote and citation should be marked up with appropriate elements.	● The character set of the document should be specified as utf-8 using an appropriate meta tag. ● The author of the letter should be specified in an appropriate meta tag.

Exercise 8 The letter[10]

Dr. Eleanor Gaye Awesome Science faculty University of Awesome Bobtown, CA 99999, USA Tel: 123-456-7890 Email: no_reply@example.com
20 January 2016
Miss Eileen Dover 4321 Cliff Top Edge Dover, CT9 XXX UK

Re: Eileen Dover university application
Dear Eileen,
Thank you for your recent application to join us at the University of Awesome's science faculty to study as part of your PhD next year. I will answer your questions one by one, in the following sections.
Starting dates
We are happy to accommodate you starting your study with us at any time, however it would suit us better if you could start at the beginning of a semester; the start dates for each one are as follows:
First semester: 9 September 2016 Second semester: 15 January 2017 Third semester: 2 May 2017
Please let me know if this is ok, and if so which start date you would prefer.
You can find more information about important university dates on our website.

Exercise 9 Subjects of study

At the Awesome Science Faculty, we have a pretty open-minded research facility — as long as the subjects fall somewhere in the realm of science and technology. You seem like an intelligent, dedicated researcher, and just the kind of person we'd like to have on our team. Saying that, of the ideas you submitted we were most intrigued by are as follows, in order of priority:
Turning H2O into wine, and the health benefits of Resveratrol (C14H12O3.) Measuring the effect on performance of funk bass players at temperatures exceeding 30°C (86°F), when the audience size exponentially increases (effect of 3 × 103 increasing to 3 × 104.) HTML and CSS constructs for representing musical scores.
So please can you provide more information on each of these subjects, including how long you'd expect the research to take, required staff and other resources, and anything else you think we'd need to know? Thanks.

Exercise 10 Exotic dance moves[11]

Yes, you are right! As part of my postdoctoral work, I did study exotic tribal dances. To answer your question, my favourite dances are as follows, with definitions:
Polynesian chicken dance A little known but very influential dance dating back as far as 300 BC, a whole village would dance around in a circle like chickens, to encourage their livestock to be "fruitful". Icelandic brownian shuffle Before the Icelanders developed fire as a means of getting warm, they used to practice this dance, which involved huddling close together in a circle on the floor, and shuffling their bodies around in imperceptibly tiny, very rapid movements. One of my fellow students used to say that he thought this dance inspired modern styles such as Twerking. Arctic robot dance An interesting example of historic

[8] Information adapted from: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Marking_up_a_letter
[9] This section was adapted from Marking up letter – W3CSchool, available at: https://www.w3cschool.cn/webstart/webstart-marking-up-a-letter.html
[10] The source or the letter is here: https://mdn.github.io/learning-area/html/introduction-to-html/marking-up-a-letter-finished/
[11] The reply was adapted from two sources: https://discourse.mozilla.org/t/marking-up-a-letter-assessment/24676 and https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/marking-up-a-letter-start/letter-text.txt

	misinformation, English explorers in the 1960s believed to have discovered a new dance style characterized by "robotic", stilted movements, being practiced by inhabitants of Northern Alaska and Canada. Later on however it was discovered that they were just moving like this because they were really cold. For more of my research, see my exotic dance research page. Yours sincerely, Dr Eleanor Gaye University of Awesome motto: "Be awesome to each other." -- The memoirs of Bill S Preston, Esq
Exercise 11 - Recipe formatting ***Source: Marking up a letter by Mozilla contributors, licensed under CC-BY-SA 2.5*** ***Instruction:*** ***Add the missing HTML elements to the recipe in the index.html file***	**Quick hummus recipe**[12] This recipe makes quick, tasty hummus, with no messing. It has been adapted from a number of different recipes that I have read over the years. Hummus is a delicious thick paste used heavily in Greek and Middle Eastern dishes. It is very tasty with salad, grilled meats and pitta breads. Ingredients 1 can (400g) of chick peas (garbanzo beans) 175g of tahini 6 sundried tomatoes Half a red pepper A pinch of cayenne pepper 1 clove of garlic A dash of olive oil **Instructions** Remove the skin from the garlic, and chop coarsely Remove all the seeds and stalk from the pepper, and chop coarsely Add all the ingredients into a food processor Process all the ingredients into a paste. If you want a coarse "chunky" hummus, process it for a short time If you want a smooth hummus, process it for a longer time For a different flavour, you could try blending in a small measure of lemon and coriander, chili pepper, lime and chipotle, harissa and mint, or spinach and feta cheese. Experiment and see what works for you. Storage Refrigerate the finished hummus in a sealed container. You should be able to use it for about a week after you've made it. If it starts to become fizzy, you should definitely discard it. Hummus is suitable for freezing; you should thaw it and use it within a couple of months.
Exercise 12 - Chinese farmer ● *Translate the html page into semantic html, using the right html tags : h1, h2, blockquote, q, img, img, hr, figure and caption, table, th, tr, td, ul or ol andli.* ● *No div or span: they do not provide any semantics.* ● *Find, for each of these tags, the origin of their name (that's how we remember them). If in doubt, look for the answer on the documentation.* ● *Add two or three links of your choice in the html page via the tag a.* ● *Is there a part that could be considered as a header? If so, group it in a header tag.* ● *And a footer? If so, group this content together in a footer tag* ● *Put all instances of the words "Maybe" in an em or strong tag.* ● *Add the attribute Alt to the images. What is the purpose of this attribute?* ● *Add a "good" or "bad" class to the tags surrounding the words "Good" and "Bad".* ● *Find the link attribute to indicate the page to which the link should lead, and add it.* ● *Make sure that when you click on the links, the page opens in a new browser tab.* ● *Find the attribute to display a small text box when*	**The Story of the Chinese Farmer** Is it a good thing? Is that a bad thing? I don't know. May 11, 2016, Alexandre Gorius In nature, Good & Evil do not exist. The Universe itself is neither good nor bad, but a potential in permanent development. In life, there are 4 types of people. - people who think the Internet is good, - people who think the Internet is evil, - people who think the Internet is cats, - people who don't think about it. [Image: images 4 students/youngBlogger.jpeg] Up in space, turning around, looking at the earth's face. From that rock's perspective on Mars or that gaz opinion from Jupiter, nothing cares about your problems. Up there, it's just not serious, and there's no such thing as an issue[13]. Down here, the only reality is the one inside our brain. Nothing exists if the brain doesn't think about it. Nothing exists either if our senses don't transmit the message. That means that if we mis sense, our reality changes. That's possible because our perception is limited : you can't see at 360 degrees for example. If we miss translate the message, our reality changes. If we think about it in a way, our reality changes depending on that way. Therefore, if we can change reality that much, everything's fake. They're proof of that. Fall in love with someone and you'll see that person as divine. Pull yourself out of that grip and the divine falls from clouds to solid ground, revealing him or herself differently. But the person is the same, you just changed your mind. Everything works like that: it's not just that you can make your world, it's that you passively do it anyway. [Image : images 4 students/shineyRock.jpeg caption: "what does he think about it?"] The story [Image : images 4 students/chineseman.jpeg caption: "The Old Chinese Farmer"] Once upon a time there was a Chinese farmer[14] whose horse ran away. That evening, all of his neighbors came around to commiserate. They said, "We are so sorry to hear your horse has run away. This is most unfortunate." The farmer said, "Is it good? Is it bad? I don't know." The next day the horse came back

[12] The recipe was taken from: https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/html-text-formatting/text-start.html
[13] Section adapted from: https://byrslf.co/theres-no-good-nor-bad-85a62f371abc

hovering over links, like this:

image
Done?
Well done! You just completed the first chapter! But you probably noticed that we still have not made something appealing for the eye. We will cover that in our next chapter: CSS.

bringing seven wild horses with it, and in the evening everybody came back and said, “Oh, isn’t that lucky. What a great turn of events. You now have eight horses!” The farmer again said, “Is it good? Is it bad? I don't know.”
The following day his son tried to break one of the horses, and while riding it, he was thrown and broke his leg. The neighbors then said, “Oh dear, that’s too bad,” and the farmer responded, “Is it good? Is it bad? I don't know.” The next day the conscription officers came around to conscript people into the army, and they rejected his son because he had a broken leg. Again all the neighbors came around and said, “Isn’t that great!” Again, he said, “Is it good? Is it bad? I don't know.”
The whole process of nature is an integrated process of immense complexity, and it’s really impossible to tell whether anything that happens in it is good or bad — because you never know what will be the consequence of the misfortune; or, you never know what will be the consequences of good fortune.

Alan Watts

Event	Good or Bad ?
Lose a horse	Maybe
Win more horses	Maybe
Son injury	Maybe
Avoid war	Maybe

Table representing the farmer's philosophy.

Conclusion

Good and Bad are just one perspective. Bananas are good for monkeys, but monkeys are not good for bananas. Let's not make a big deal out of it...

---

[14] The Story of the Chinese Farmer, by Alan Watts, taken from here: https://www.craftdeology.com/the-story-of-the-chinese-farmer-by-alan-watts/ and from here: https://wellsbaum.blog/alan-watts-story-of-the-chinese-farmer/

Module 2 Exercises

Exercise 1 Add some inline CSS	**Instruction** Here is some HTML content. Use the style attribute to : ● make the body have a background color of light gray (use this value: #DDDDDD) ● make the paragraph text in blue ● make the strong tags look like they were highlighted, by adding a background color in yellow. ● Underline the heading ● Add a margin bottom of 10 pixels to the List Items (li tags) ● Style the Ordered List (ol tag) so that its content is in red. <body> The 7 steps to planning your vacation Everyone needs a break once in a while! How about trekking on mountains, or perhaps you're more the "beach" type ? Here is a memo on how to make sure your holidays are a success! - Step 1: Decide where to go. - Step 2: Pick a time to travel. - Step 3: Take time off work. - Step 4: Find affordable flights. - Step 5: Find a great place to stay. - Step 6: Call your credit card companies. - Step 7: Budget consciously for your trip. </body>
Exercise 2 Plan your vacations with style!	**Instruction** Redo the exercise above by removing all inline style and put all style properties in a <style> tag. Remember: should it go above the content, or below the content ? If you're not sure, try both and see what works…
Exercise 3. Canary Islands	Welcome to the Canary Islands [15] Scented pine forests, haunting volcanoes, lunar-like landscapes, secret sandy coves, miles of Sahara-style dunes, beach-hugging resorts – the beautiful, unique Canary Islands wear many tantalising hats. Otherworldly Landscapes Marvel at the pine-forested peaks of Gran Canaria's mountainous interior, the tumbling waterfalls of La Palma or the subtropical greenery of La Gomera's Parque Nacional de Garajonay. Then contrast all this lushness with the extraordinary bare flatlands flanking Tenerife's El Teide, the surreal party of colours glittering across Lanzarote's lava fields, the gentle flower-filled hillsides of El Hierro, and Fuerteventura's endless cacti-sprinkled plains. The Canary Islands' near-perfect temperatures mean that, year-round, you can soak up fantastical, varied landscapes otherwise only found by crossing continents. The Great Outdoors [16] It's this very diversity that makes outdoor pursuits such an easily accessible and key pleasure of the Canaries. Hike the many footpaths criss-crossing the islands, from meandering coastal trails to challenging mountain treks to tranquil forest walks; go diving or snorkelling in blissfully warm waters inhabited by more than 350 species of fish (and the odd shipwreck); or pump up the adrenaline by riding the wind and the waves – kitesurfing, windsurfing, surfing and paragliding are all big here. Then slow things down with horse rides, boat trips, kayaking and paddle-boarding jaunts or beachfront yoga.

[15] Story taken from here: https://www.lonelyplanet.com/canary-islands and here: https://www.globeguides.co/destinations/YXZU3UGA4VJO936AZ7YH and here: https://euresorts.co.uk/destinations/canaries/

[16] Use classes to differentiate elements and style them differently

	Art & Architecture Contrary to many expectations, the Canary Islands are immensely rich in both original art and architecture – sometimes you just need to know where to look. The spectacular surrealist canvases of world-acclaimed painter Óscar Domínguez grace his Tenerife homeland; the enormous abstract sculptures of Martín Chirino are impossible to miss on Gran Canaria; and César Manrique's inspired 'interventions' pop up all over Lanzarote (and beyond). Everywhere, seek out the emblematic wooden balconies, leafy internal patios and cheerily painted facades that typify vernacular Canarian architecture, and pop into charming palm-shaded churches, many of which date back several centuries.
Exercise 4 Use classes to differentiate elements and style them differently	***Instruction*** Let's make the last exercise look better, using more advanced techniques. Use the CSS Cheat Sheet and the Documentation to find the proper syntax. The wording is made to give a hint at the correct property name). 1. Style the body tag so that it has a padding of 30px and a light grey background color (use these values: Red: 220, Green: 220, Blue: 220) 2. Style all Paragraphs so that their font is 16px, with a line height of 120% , with a dark (but not black) color (use these values: Red: 25, Green: 25, Blue: 25) 3. Add a class to the paragraph tag underneath the H1 tag, with a value of "introduction" (the syntax is like this: and in CSS, you can mention that it is a class by adding a period in front of it, like this .classname (If you are stuck, make sure to read about "Selectors" in the Documentation or in the lessons) 4. Style that "introduction" paragraph by setting its font size at 140% . Change the line height to 200% and give it an "italic" font style.
Exercise 5 Borders	**Instruction** Look at this stylish double underline below : the blue thick line spreads the word's length, while the grey line spreads over the entire block. Try to mimick by using CSS on the H2 and the P tags! Tip 1: You will need to use the display property on the h2 element. Check the documentation to find out which value you will need. (Standard value is block which is why the line covers the whole page.) Tip 2: Find a way to set the font family to 'sans-serif'. Tip 3: Color: #2A99FB. About Us Look at this stylish double underline above: the blue thick line spreads the word's length, while the grey line spreads over the entire block. Try to mimick by using CSS on the H2 and the P tags!

Exercise 6: Class selector	**Instructions** Add the necessary classes in the html below. Then style each of the brand names in the html so that they look as close as possible as their brand logo. Note that you can also add more HTML tags if you want to (especially for FedEx, you will need to!). In such case, use tags, they are useful to color a part of a text Here is the result you should try to reach (or as close as possible). Brands 1. Toyota 2. Toyota 3. FedEx Express
Exercise 7: Pseudo-class selector	**Instructions** Add the necessary style so that when the mouse hover on the word below, it changes color. ● by default: text is yellow ● when hovering on it: text is red (state: :hover) ● when clicking on it: text is blue (state: :active) ● after clicking on it: text is green (state: :visited) Here is an illustration of the result to achieve: Links States Link
Exercise 8: Style this page[17]	**Instructions** Style the HTML elements according to the following instructions. DO NOT ALTER THE EXISTING HTML TO DO THIS. WRITE ONLY CSS! ● Give the <body> element a background of #bdc3c7 ● Make the <h1> element #9b59b6 ● Make all <h2> elements orange ● Make all elements blue (pick your own hexadecimal blue) ● Change the background on every paragraph to be yellow ● Make all inputs have a 3px red border ● Give everything with the class 'hello' a white background ● Give the element with id 'special' a 2px solid blue border(pick your own rgb blue) ● Make all the <p>'s that are nested inside of divs 25px font(font-size: 25px) ● Make only inputs with type 'text' have a gray background ● Give both <p>'s inside the 3rd a pink background ● Give the 2nd <p> inside the 3rd a 5px white border ● Make the in the 3rd element white and 20px font(font-size:20px) ● Make all "checked" checkboxes have a left margin of 50px(margin-left: 50px) ● Make the <label> elements all UPPERCASE without changing the HTML ● Make the first letter of the element with id 'special' green and 100px font size(font-size: 100) ● Make the <h1> element's color change to blue when hovered over ● Make the <a> element's that have been visited gray

[17] Exercise adapted from: https://gist.github.com/weav797/78888a9b10db101eef1e87d1f18f80bc

Exercise 9 PARAGRAPH NOT INSIDE A DIV

I am a paragraph with a class
I am a paragraph with an ID

I am an awesome h2
Roof party yr hella synth, Wes Anderson narwhal four dollar toast before they sold out retro lo-fi. Austin iPhone pop-up farm-to-table, PBR&B McSweeney's ennui messenger bag distillery before they sold out Portland wolf fanny pack YOLO. Locavore slow-carb trust fund farm-to-table. Pinterest gastropub lo-fi, McSweeney's trust fund VHS shabby chic ugh Austin twee. Messenger bag banjo lumbersexual, whatever 3 wolf moon normcore. Pug pack 3 wolf moon, typewriter organic chia mustache scenester seitan shabby chic Blue Bottle salvia ugh iPhone.Pack Williamsburg direct trade, cold-pressed disrupt flannel listicle health goth asymmetrical freegan mixtape street art pour-over whatever.
Things I need to do
- Walk Dog
- Feed Dog
- Wash Dog

I am another awesome h2
Cardigan Tumblr mlkshk, tilde 3 wolf moon Portland. Heirloom health goth taxidermy blog lo-fi selfies, post-ironic master cleanse fingerstache normcore. Kickstarter plaid twee, bespoke single-origin coffee sustainable lo-fi vinyl Pinterest pork belly cronut skateboard 3 wolf moon. Normcore single-origin coffee salvia, bespoke Austin swag Godard before they sold out kogi disrupt locavore. Shoreditch Vice, artisan American Apparel master cleanse yr salvia vegan. Bespoke letterpress heirloom kale chips deep v four loko. Lomo sustainable put a bird on it trust fund post-ironic
I'm the second paragraph inside this div!

PARAGRAPH NOT INSIDE A DIV
A less awesome h2
Roof party yr hella synth, Wes Anderson narwhal four dollar toast before they sold out retro lo-fi. Austin iPhone pop-up farm-to-table, PBR&B McSweeney's ennui messenger bag distillery before they sold out Portland wolf pack YOLO. Locavore slow-carb trust fund farm-to-table. Pinterest gastropub lo-fi, McSweeney's trust fund VHS shabby chic ugh Austin twee. Messenger bag banjo lumbersexual, whatever 3 wolf moon normcore. Pug pack 3 wolf moon, typewriter organic chia moustache scenester seitan chic Blue Bottle salvia ugh iPhone. Pack Williamsburg direct trade, cold-pressed disrupt flannel listicle health goth asymmetrical freegan mixtape street art pour-over whatever
One last paragraph here!
I am a link to facebook I am another link to facebook
Name
Password
PARAGRAPH NOT INSIDE A DIV

Exercise 10 - CSS Diner

Source: css diner, licensed under Mozilla Public License 2.0[18].
Welcome to CSS Diner!
CSS Diner is a fun little "game" that teaches you the basics of CSS selectors. We know that understanding the way CSS selectors work can be confusing at first, but once you spend a bit more time on the topic, it will become straightforward!
In this game, you will see a fancy little table that will serve you a plate with or without food. Your job is to select the requested items shown in a short paragraph on the top of your screen.
Use the editor and the HTML viewer to find the correct CSS selector.
Can you reach level 32?
Good luck!

[18] The game is available here: https://flukeout.github.io/

Exercise 11 - Basic Properties

It is important to take your time with these exercises. Try to understand what is happening. Why is this color changing? Why is the size different? Etc..
Learning the basics
We all love basic things! - Don't overthink it.😉
Give me a red color and align me to the center
Give me a blue background
Give me a bigger margin to the left.

Give me a font size of 10px, Give me a font size of 20px.
Don't change me! But give the 'lorem' a line height of 10px,letter space of 4px
Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloribus illo tempora, dolores iure rem quos neque harum omnis unde. Nam pariatur quidem nostrum nisi harum voluptatem tempore impedit fugiat minus? Esse quo debitis nam commodi architecto praesentium, ullam laborum quasi odit. Quis recusandae et magni possimus mollitia at. Distinctio, esse? Consectetur quam enim consequatur delectus quod in est repellendus a. Tempora, labore autem? Ipsum, exercitationem officiis velit iste quasi facilis id deleniti. Officiis temporibus est repellat numquam omnis quam quia non sint! Esse velit dolore impedit deleniti reiciendis unde hic. Commodi iure eius sunt minima quod repellendus sed magni explicabo architecto sit a temporibus, assumenda debitis accusantium omnis non soluta natus laboriosam. Optio ad assumenda sunt, dolores consequatur perspiciatis nihil.

Give me a underline
Remove my bulletpoint
Make me uppercase only
Give me a border of 2px, not solid but dotted.
Give me a border of 4px to the right and solid.
Make this text, bold. Give it a color of white. Give it a big font size. And add the image in this project to the background. Give it a width and height of 200px.
Give this button a background color, give 5px padding to the top 10px to the bottom and 15px to the left and right.
After this, give the button round corners, except from the bottom right corner.
After this, when we hover over the button, the background color should change.
Button
Make a circle shape around the image. Tip: border-radius, and overflow. Give it a width and height of 150px. When we hover over it, the image scales. But the circle shouldn't get bigger.
Remove the underline,bullet points, give each li a different background color. give each li a different (random) margin values on hover change the font-sizes. All font colors should be white.
Home
Store
About
Contact

Display properties

Exercise 12_ Display properties exercise1

Before we start
Before we start with these exercises, I want you to take a good look in the documentation, and search for the term display. Take a good read to see what the differences are and what it exactly does.
Please take your time with this! It is important to know the basics about this.
Instructions
Find a way to display the content in the editor like this:
image
NOTE: Do not change the HTML tags or content.
Only work with your styling. Keep in mind what we just learned about.
Calendar
Reminder set for 20:00.

Exercise 13__ Display properties exercise 2

Instructions
Find a way to display the content in the editor like this:
image
NOTE: Do not change the HTML tags or content.
Only work with your styling. Keep in mind what we just learned about.

This span should have a width of 200px. This
list
should follow up in one line & should not use any width, but we are not allowed to remove it either…

Position properties

Exercise 14_Position properties exercise 1

Before we start
Before we start with these exercises, I want you to take a good look in the documentation, and search for the term position. Take a good read to see what the differences are and what it exactly does.
Please take your time with this! It is important to know the basics about this.
On the job
As a refreshment of everything we have currently seen so far. I want you to re-create this simple navbar.
image
After this, find a way to let the navbar stick to the top of the screen.
Please scroll down!
The navbar should still be visible down here!

Exercise 15_ Position properties exercise 2

Now re-create this styling as good as possible:
image
Now make it so that if we scroll down, each case collapses onto each other. (Leave 20px space between the top and the cases.)

Exercise 16_ Position properties exercise 3

Now re-create this styling as good as possible:
image
Don't forget to use display! (Sometimes even on parents...)
top left
top right
bottom left
bottom right

Flex-box

Exercise 17_Flex-box 1

So far we have not touched flex-box, and there is a good reason for it. It makes our lives as developers much easier, so easy that you would forget about floats,inline-blocks,etc..
I strongly advice to take a look at the documentation to get a brief understanding about the power of the dar- flex-box.
You have to make a exact copy of this by just using flex-box:
Learning to flex
It's just a little different than this kind of flex -- Please make an exact copy of the example image!
1. Center the divs
2. There must be space between
3. Spaces should be about the same.
4. Boxes start from the start
5. Boxes start from the end
6. Boxes should be spread around in height
7. make the order random
8. Reverse the order.

Exercise 18_Flexing card

For this exercise we will not use any set-up code. We gonna refresh your HTML skills! What are we going to make?
A card that could be used on a application for a pizza delivery service!
image
Colors:
Green: #444444
Blue: #FFFFFF
Tip: always start with the content: so focus on the html first, without worrying to much about CSS. When your HTML is good, the move on to CSS. This way, you will only add classes when required by the CSS, which is a good practice.

Exercise 19 Flexing page

For this exercise we will not use any set-up code. We gonna refresh your HTML skills! What are we going to make?
A landing page about Tim Berners-Lee!
Does his name not ring a bell? Don't worry, You will get to know him during this exercise.
Refreshing
Recreate the following structure in HTML:
image
Flex those containers!
Great work! Now style this page into this:
image
Try to use flex-box as much as possible!
Note: You will not find the correct font, try to use something that gets "close enough".
Tip: always start with the content: so focus on the html first, without worrying to much about CSS. When your HTML is good, the move on to CSS. This way, you will only add classes when required by the CSS, which is a good practice.

Exercise 20 CSS Integration Series ,

Instructions
You work in a Web Design agency as Frontend Integrator. The UI Designer provides you with images representing a high-fidelity mockup of the website's interface components.
Your job is to reproduce in html and CSS each of these components. Each UI component has been validated by your customer. It's therefore really important that the end-result looks as close as possible to the mockups.
Look at this image:
image
Reproduce this interface as close as possible using html and CSS. You have to reach the closest "pixel-perfect" version you can. Ideally, we should not see any difference between the image mockup and your result, without using any image. Images are only accepted for illustrations, when there is no CSS alternative.
Color:
Tip: always start with the content: so focus on the html first, without worrying to much about CSS. When your HTML is good, the move on to CSS. This way, you will only add classes when required by the CSS, which is a good practice.
The CSS has to be in a <style> tag, no inline CSS.

Exercise 21 CSS Animations

The exercise
In this exercise, you will build a complex animation, step by step.
As often with CSS, this is an exercise in precision and accuracy (and a zest of dementia, because... CSS).
To give you a reliable visual assistant, you will thus use a technique that could be called "the image background guide Technique".
Tip: the image background guide Technique: for each step of the exercise, use the images as your body's background image (with a slight opacity perhaps?) to help you style precisely.
Of course, when done, remove the background image for a cleaner look.

- Create the stage: add a div with the id "stage", style it so that it appears exactly like this image, and in the horizontal middle of your index page.

images

- Add a div with an id "hero".
- Style the #hero so it looks exactly like this:

images
Now, construct the animation step by step.
Step 1
Move the hero to the right, one time.
image
Step 2
Move the hero to the right, in 5 seconds, indefinitely (loop animation).
image
Step 3
Move the hero to the right, then to the left, in 2 seconds, indefinitely (loop animation).
image
Step 4
image
Step 5-7
You got it ? Ok, now follow these instructions. Think "pixel-perfect".
image
Done ?
Bravo !
Image

Exercise 22 Small CSS animations series

In these series of exercises you have to replicate a series of elements.
Make an exact copy of the examples and write your own HTML and CSS!
Before you start
Please copy and paste this code inside your editor:
Code
Animated component:

Code for arrow
Or
CSS CODE

Exercise 23 Small CSS animations series 1

In these series of exercises you have to replicate a series of elements.
Make an exact copy of the examples and write your own HTML and CSS!
SPECIAL
This exercise is a little different than the others, Try to make a little page and style it to your taste. Include this text somewhere on the page (be creative!).
Lonely commute
Join Australia-based photographer Mark Forbes as he wanders the streets of Tokyo with a camera & heart full of curiosity documenting the Lonely Commute.
Now imagine a small gallery of images where you can hover over and get the animated result at the bottom for each of them.
Animated component:

Exercise 24 Small CSS animations series 2

In these series of exercises you have to replicate a series of elements.
Make an exact copy of the examples and write your own HTML and CSS!
SPECIAL
You will need this text[19]:

Myths about Scrolling and Long Content Pages on the Desktop

Myth #1: Users don’t scroll long pages

Users do scroll when the content is relevant, organized properly, and formatted for ease of scanning. In fact, people prefer scrolling the page for content over pagination when the topics within that page answer the right questions. The standard scroll wheel on a mouse, arrow keys, and track pads have made scrolling much easier than acquiring click targets.

Myth #2: Customers don’t read information at the bottom of the page
---
Our eyetracking research show that while users spend 80% of their attention on information above the page fold, they allocate 20% of it to content below the fold. Reluctance to scroll is a behavior of the past. While you should still be mindful of people’s limited attention span on websites and prioritize content wisely, you shouldn’t fear long formats. People will see the bottom if you give them good reason to go there.

Myth #3: People avoid pages with a lot of content

People have the ability to handle vast amounts of information, when presented properly. In our Writing for the Web courses, we emphasize the requirement for writing well, and more importantly, writing for web-based reading. Reading and scanning patterns are different between web-based and print-based content. While online users typically scan for information, it does not mean they want less information. Websites should not be information light. The same information needs to be written, structured, and presented differently.

Accordions Are Not Always the Answer for Complex Content on Desktops

[19] Text taken from : https://www.nngroup.com/articles/accordions-complex-content/ and here: https://www.linkedin.com/pulse/20140807120829-35361745-above-the-fold-and-other-myths-about-scrolling-websites/

Exercise 25 Small CSS animations series BONUS

In these series of exercises you have to replicate a series of elements.
Make an exact copy of the examples and write your own HTML and CSS!
SPECIAL
For this one we are going to create a series of checkbox that will have this animation when getting checked:
image
Animated component:
The arrow:
Step one:
image
Step two:
image
Step three:
image
Step four:
image
The checkbox:
Step one:
image
Step two:
image
Step three:
image
Now finish it!
Image

Exercise 26 _Final Challenge

This is your final assessment. If you can solve it, you will receive a certificate that testify on your ability to start a full training on Frontend Web Development and become a professional web developer (frontend).
Instructions
The company owner approaches you with a design for the Banana Foundation. He asks you to build a website that looks EXACTLY like it. to buildCompany. Reproduce as accurately as possible the following layout.
You will find images necessary to build the interface in the project folder.
Remember:

1. Plan your work
2. Divide the plan into smaller steps
3. Start with the content (the HTML) without worrying about the look
4. Debug (fix problems and repeat from step 1).

Color codes & font family :

- yellow: rgb(242, 201, 76)
- blue: rgb(131, 175, 237)
- font family: Roboto

Final ChallengePHRASES

Login
UserName
Password
Sign in
Please fill out your e-mail.
The e-mail address you entered is invalid.
Please enter your password.
The password you entered is wrong.
Home
Courses
My Profile
Review
Logout
Sitemap
Contact us
Modules Index
Lessons
Read More
General
Courses Main Page
Start Training
Start Examination
Previous
Next
Finish
Part
Restore
Save
Submit
Active
Continue
Dates
Status
Questions
Pass
Mark
Start
End
In progress
Total
Answered
Correct
Show details
Get Certifcate
Question
Not Marked
Marked

Co-funded by the
Erasmus+ Programme
of the European Union

Project Number 2018-1-RO01-KA204-049298