Week 1- day 1

**HTML5 Introduction**

**Done**

**Table Of Contents**

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#What_you_will_learn)
  + [Useful resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Useful_resources)
    - [What is HTML?](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#What_is_HTML?)
    - [HTML Versions](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#HTML_Versions)
    - [Choosing Your HTML Editor](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Choosing_Your_HTML_Editor)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Feedback)

Last updated : March, 31st 2021

**What You Will Learn**

* The Basics of HTML

**Useful Resources**

* [Difference between HTML and HTML5](https://www.geeksforgeeks.org/difference-between-html-and-html5/).
* [HTML5 | Semantics](https://www.geeksforgeeks.org/html5-semantics/).

**What Is HTML?**

HTML, an acronym for **HyperText Markup Language**, is a computer language for creating websites and web applications. Consisting mainly of series of codes usually written in a text file and saved as .html. Code written in the HTML language translates into a beautiful, well-formatted text or a combination of text and media when viewed through a browser.

British physicist Tim Berners-Lee first developed HTML in 1990. It had gone through so many evolutions since then that the most recent version can achieve far more than was imagined possible when Tim first invented the language.

**HTML Versions**

First, a quick rundown of all the HTML versions since HTML was invented.

* HTML 1.0: This was the barebones version of HTML and the first release of the language.
* HTML 2.0: This version was introduced in 1995. It gradually evolved, allowing extra capabilities, including form-based file upload, tables, client-side image maps, and internationalization.
* HTML 3.2: In an attempt to ensure the development of standards for the World Wide Web, in 1994 Tim Berners-Lee founded the World Wide Web Consortium (W3C). By 1997, they published HTML 3.2.
* HTML 4.0: Later in 1997, the W3C released HTML 4.0 — a version that adopted many browser-specific element types and attributes.
* HTML 4.0 was later reissued with minor edits in 1998.
* HTML 4.01: In December 1999, HTML 4.01 was released.
* XHTML: The specifications were introduced in 2000, and it was recommended to be used as the joint-standard with HTML 4.01. It incorporated XML to ensure code is properly written and to ensure interoperability between programming languages.
* HTML5: The W3C published HTML5 as a recommendation in October 2014 and later released HTML 5.1 in November 2016.

**Choosing Your HTML Editor**

If you are thinking of creating web pages in HTML, you will need an HTML editor. There are several benefits to using an HTML editor.

A good HTML editor will keep your code clean and organized. It will also detect when you open a new tag and automatically close it to avoid you having a buggy code and reduce how much typing you have to do. Today, most HTML editors allow you to preview your web page to see how it will look in a web browser using their WYSIWYG feature.

There are many free and paid HTML editors; below are some of the top options you can choose from:

* Sublime Text
* HTML-Kit
* CoffeeCup
* KompoZer
* Komodo Edit
* Notepad++
* Bluefish
* CodeLobster

## HTML5

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#What_you_will_learn)
  + [Useful resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Useful_resources)
    - [HTML files structure](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#HTML_files_structure)
    - [Basic Building Blocks of HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Basic_Building_Blocks_of_HTML)
      * [Introduction to Tags](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Tags)
      * [Introduction to Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Elements)
      * [Introduction to Attributes](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Attributes)
      * [Creating Headings in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_Headings_in_HTML)
* [h1 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h1_heading)
  + [h2 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h2_heading)
    - [h3 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h3_heading)
      * [h4 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h4_heading)
        + [h5 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h5_heading)

[h6 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h6_heading)

* + - * [Creating paragraphs](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_paragraphs)
      * [Formatting Text in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Formatting_Text_in_HTML)
      * [Ordered and Unordered Lists](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Ordered_and_Unordered_Lists)
      * [Nested lists](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Nested_lists)
      * [Hyperlinking](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Hyperlinking)
      * [Using Images](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_Images)
      * [Creating Tables](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_Tables)
      * [Quotations in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Quotations_in_HTML)
      * [Using Comments in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_Comments_in_HTML)
      * [Using iFrames in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_iFrames_in_HTML)
      * [HTML Forms](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#HTML_Forms)
      * [Defining a form](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Defining_a_form)
        + [The input element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#The_input_element)
        + [The label element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#The_label_element)
    - [Conclusion](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Conclusion)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Feedback)

Last updated : March, 31st 2021

## What You Will Learn

* The Basics of HTML
* Create your first Web page using HTML
* Understand the logic behind a webpage
* HTML tags and attributes

## Useful Resources

* [HTML Periodic Table of Tags](https://websitesetup.org/html5-periodical-table/).
* [W3Schools/Tags – List of all HTML tags ordered alphabetically](https://www.w3schools.com/tags/).
* [Mozilla/HTML Attributed – List of all HTML attributes ordered alphabetically](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes).
* [HTML Cheat Sheet](https://websitesetup.org/html5-cheat-sheet/).
* [Another HTML Cheat Sheet](https://htmlcheatsheet.com/).
* [HTML Color codes/Color Names – List of 140 HTML color names including HEX code, RGB value and HSL value](https://htmlcolorcodes.com/color-names/).

### HTML Files Structure

A HTML file is a text file with the .html extension.  
The code in the file needs to start with <!DOCTYPE html>.

An HTML element is defined by a start tag and an end tag, with the content inserted between:  
<element-type>My element value</element-type>

Notice how a tag is closed with the **/**  
Some tags don’t need to be closed, they’re called **orphan tags**

* All the HTML code is inside an HTML element (<html>).
* The HTML code is separated into two big parts, the <head> and the <body>.  
  The body is the visible part of the HTML code, while the head is here to add some config to the file.

Before content is added, most HTML files basically look like this:

<!DOCTYPE html>

<html>

<head>

<title>Title of the document</title>

</head>

<body>

The content of the document......

</body>

</html>

* Every HTML file begins with the opening tag <html> and ends with the closing tag </html>.
* The HTML file’s first line should declare the type of document so that the browser knows what HTML version you use. You see that before the HTML code begins, it starts with <!DOCTYPE html> .
* The section in the <head> tag usually contains information about the document such as its title, meta tags, and where to locate its CSS file – content that is not directly visible on the browser page.
  + Title - Example : <title>This is our page title</title>
  + This is our page title will be displayed in the browser tab. It’s also indexed as the title for the page when the search engine bots crawl your website.
  + Meta element: often used to specify the information, search engines can describe the content in their listings. This includes the description, keywords, author information, etc. The <meta> element also specifies the character set the HTML document uses.
* The section between the tag <body></body> is where the HTML file’s main content is located. It’s what the viewer will see in the browser. This includes everything from the first paragraph to hyperlinks, formatting, multimedia, and everything else.

### Basic Building Blocks Of HTML

When coding in HTML, it is essential to understand these building blocks. They include tags, attributes, and elements.

#### Introduction To Tags

In essence, **tags** separate normal text from HTML code.  
Tags are practically the building block of HTML – you can’t do HTML without tags! If you’re stuck on which tag to use, be sure to check out the Useful Resources above.

##### Opening And Closing Tags

Almost every open tag must be closed. Keep in mind that there are exceptions.

An example of a tag that does not have to be closed is an empty tag, such as the line break: <br>.

Tags are contained in less than (“<”) and greater than (“>”) angle bracket. Closing tags have a trailing slash that before the name of the tag being closed.

Example of an open tag: <b>. Example of a closed tag </b>.

When it comes to HTML, tags make the difference between whether your document is displayed as ordinary text or ‘transformed text’ . This ‘transformed text’ is a code that can display a series of things (hyperlinks, images, media, or other methods of formatting).

##### Bold & Italic Text

Let’s take a look at the word “He is a boy” as an example: \* In ordinary text format you get: He is a boy. \* In bold text format you get: **He is a boy**

To achieve what we have in the bold format, you have to use the <b></b> tag : <b>He is a boy</b>.

This sentence could also come out italicized.  
This is achieved using the <i></i> tag : <i>He is a boy</i>.

#### Introduction To Elements

In HTML, an “element” consists of the opening and closing tag and the content between the tags.

Example : <b>He is a boy</b>.  
The text “He is a boy” is surrounded by an open and closed tag. Everything, including the opening tag, the content, and the close tag is an **element**.

An element could be in a basic form or a complex form. Why? Because anything in between an opening tag and a closing tag is an element. It means that we can have elements within an element.

In our current example, “He is a boy” (<b>He is a boy</b>) is an element.

#### Introduction To Attributes

While HTML documents use tags for everything, we sometimes want to communicate additional information inside an element. In this case, we use an **attribute**. The attribute is used to define an element’s characteristics and it’s used inside the opening tag of the element. Attributes are made up of a name and a value.

Note that the value of an attribute is placed inside a quotation mark using the format

<tag attribute="value">Your Text</tag>

#### Example

<p align="center">He is a boy</p>  
In this example, we instruct that “He is a boy” is aligned in the center of the document.

#### Creating Headings In HTML

Headings play a significant role in the success of a website. Firstly, they make it easy for readers to scan the content of your pages. Secondly, and perhaps more importantly, they communicate your web pages’ structure to search engines and, therefore, often impact how your content is ranked in search engine results.

That said, it is essential to avoid using heading tags to make your text big or bold. Other tags can be used to make your text big (which we’ll get to later in this section). Instead, heading tags should be used solely for structure.

There are six heading tags in HTML: <h1> to <h6>, with the <h1> tag indicating the most important heading and the <h6> tag indicating the least important heading.

* HTML Headings Example:

Headings are used for titles and big texts

# H1 Heading

## H2 Heading

### H3 Heading

#### H4 Heading

##### H5 Heading

###### H6 Heading

#### Creating Paragraphs

Paragraphs can be created with the <p> tag.

#### Example

<p>This is your first paragraph.</p>

<p>This is your second paragraph, and you will be creating many more paragraphs.</p>

Keep in mind that writing in HTML is very different from writing in pure text. Therefore, if you break up text inside HTML without starting a new paragraph, it won’t matter when the browser displays the text. Instead, you want to use a line break, represented by the <br> tag.

#### Example

<p>This is a new paragraph.

And I want to use a number of new lines.

So I’m breaking it up.</p>

This **will not** come out as the following:

This is a new paragraph.  
And I want to use a number of new lines.  
So I’m breaking it up.

Instead, it will come out as this:

This is a new paragraph. And I want to use a number of new lines. So I’m breaking it up.

So, how do you break texts into new lines so that it would show like this:

This is a new paragraph.  
And I want to use some new lines.  
So I’m breaking it up.

By using line breaks.

<p>This is a new paragraph.<br>And I want to use several new lines.<br>So I’m breaking it up.</p>

If you want to create a horizontal line, you can use the <hr> tag.

<p>This is a new paragraph.<hr>And I want to create a horizontal line</p>

It will display like this:

This is a new paragraph.

And I want to create a horizontal line

#### Formatting Text In HTML

The next step is to format your text in HTML. This is where you can indicate whether you want your text to come out bold, italicized, underlined, subscripted, superscripted, etc. This is a basic guide, so this section won’t be the be-all-end-all for formatting. Instead, we will only include some basic formatting tags. The process for using other forms of formatting is simple – just find the tag you want and go ahead:

Using bold: <b>He is a boy</b> comes out as **He is a boy**

Using italics: <i>He is a boy</i> comes out as *He is a boy*

Underlining text: <u>He is a boy</u> comes out as He is a boy.

It’s worth noting that the <u> tag was deprecated in HTML 4.01 and was redefined to represent stylistically different text in HTML5. As a result, it is recommended to use CSS to indicate text that should be underlined. Since this article is a basic guide, we’re keeping it simple.

Using subscript: <sub>He is a boy</sub> comes out as He is a boy

Using superscript: <sup>He is a boy</sup> comes out as He is a boy

For other tags that can be used to format, take a look at **Useful Resources** at the beginning of the page.

##### Exercise 1

Create a structured HTML file as explained above.

Write about yourselves : your name, age, birthdate and why you decided to learn to code. Style this small text with paragraph, headings, italic and bold characters.

#### Ordered And Unordered Lists

Sooner or later, you will have to create lists. Lists could be ordered (i.e., numbered) or unordered (i.e., unnumbered).

Here is an example of an ordered list:

1. Item 1
2. Item 2
3. Item 3

Here is how to create it:

<ol>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ol>

Here is an example of an unordered list:

* Item 1
* Item 2
* Item 3

Here is how to create it:

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

If it is not already obvious. Here’s a breakdown:

The <li> tag is used to indicate each item on the list.  
When making a list, you can use the <ol> tag to indicate an ordered list (“o” = ordered and “l” = list) or the <ul> tag to indicate an unordered list (“u” = unordered and “l” = list).

#### Nested Lists

We can also have nested lists or a list within a list.

#### Example

* Item 1
  + Item 1 nested
  + Item 2 nested
  + Item 3 nested
* Item 2
* Item 3

This can be created with:

<ul>

<li>Item 1

<ul>

<li>Item 1 nested</li>

<li>Item 2 nested</li>

<li>Item 3 nested</li>

</ul>

</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

As you can see, you open another listing tag – ordered (<ol>) or unordered (<ul>) – depending on what you want before closing the item you want to be nested.

#### Hyperlinking

The web is one massively interconnected network of pages. If you create a website – whether internally or externally, or both – you will have to link to other pages. A link to an internal page on your website or an external page on the web is called a hyperlink. While people usually link a text, any HTML element – an image, for example – can be linked.

The <a> tag is used to define links in HTML, while the “href” attribute is used to specify a link’s destination.  
The link is then put into a quote after the “equal to” sign before the tag is closed.

#### Example

To create a hyperlink,

this:

<a href="https://websitesetup.org">YOUR LINK TEXT HERE</a>

will come out as: [YOUR LINK TEXT HERE](https://websitesetup.org/).

Now, assume you are linking to a local file in which you have all your HTML pages in the same folder. In this case, you don’t have to include a website URL. Just add the file path.

For example, if linking to a file titled about-page.html, your hyperlink becomes

<a href="about-page.html">YOUR LINK TEXT HERE</a>

which links to the about-page.html file.

If the HTML file you want to link to is local but different from the main folder, you specify the file path.  
For example, if the file is in the “files” folder under the main document where your current document is, your hyperlink goes something like this

<a href="files/about-page.html">YOUR LINK TEXT HERE</a>

You might want to specify how you want the link to be opened e.g., in a new window/tab. You will use the target attribute for that.

#### Example

To specify that the link should be opened in a new tab.

<a href="about-page.html" target="\_blank">YOUR LINK TEXT HERE</a>

To specify that the link should be opened in the same tab.

<a href="about-page.html" target="\_self">YOUR LINK TEXT HERE</a>

Other attributes can be used to communicate different ways a link should be opened:

* \_blank – Open in a new tab.
* \_self – Open in the same tab.
* \_parent – Open in the parent frame
* \_top – Open in the full body of the window
* framename – Open in a named frame

You can also have an image point to a link.

#### Example

<a href="https://websitesetup.org">

<img src="heisaboy.jpg" alt="He is a boy">

</a>

This tells the browser to display an image pulled from the image file “heisaboy.jpg” and have it linked to https://websitesetup.org.

The “alt” attribute represents what should be shown if the image cannot be displayed (due to browser settings or other stuff that prevents images from showing).

#### Using Images

Images are added to an HTML document with the <img> tag. The <img> tag is empty, so it doesn’t need to be closed.

#### Example

<img src="heisaboy.jpg" alt="He is a boy">

This is a basic example of telling the browser to display an image titled “heisaboy.jpg” pulled from the same directory as the HTML document.

The “alt” attribute tells the browser to show a text (known as an “alternative text”) called “He is a boy” if, for some reason the browser or internet settings prevents the browser from displaying images. If you want to pull an image in another directory or on an external site, the full address/path needs to be specified.

Going beyond the basics, we could also use other attributes to customize the image better. For example, the “style” attribute can be used to specify the width, the height, or both.

#### Example

<img src="IMAGE PATH" alt="YOUR ALTERNATIVE TEXT" style="width:Xpx;height:Ypx;">

(the values “Xpx” and “Ypx” should be replaced with the actual values in “px” (e.g. “20px”) you want for your image.)

#### Exercise 2

Work on the HTML file you previoulsy created

Make ordered and unordered lists of the things you love to do, to eat, etc ... Don't forget to style it a bit

Create also links to your social media profile

**Bonus:** Use image and icons to make these links more userfriendly

#### Creating Tables

As you get deeper into HTML, you will want to learn various ways to present information in a more organized way. One such way is through the use of tables.

Tables are created with the <table> tag.

* Each header in a table is specified with the <th> (“table header”) tag
* while each row is specified with the <tr> (“table row”) tag.
* The table data is then indicated with the <td> tag.

#### Example

<table>

<tr>

<th>Table Header 1</th>

<th>Table Header 2</th>

<th>Table Header 3</th>

</tr>

<tr>

<td>Input 1 under header 1</td>

<td>Input 1 under header 2</td>

<td>Input 1 under header 3</td>

</tr>

<tr>

<td>Input 2 under header 1</td>

<td>Input 2 under header 2</td>

<td>Input 2 under header 3</td>

</tr>

<tr>

<td>Input 3 under header 1</td>

<td>Input 3 under header 2</td>

<td>Input 3 under header 3</td>

</tr>

</table>

This gives us something like this:

| **Table Header 1** | **Table Header 2** | **Table Header 3** |
| --- | --- | --- |
| Input 1 under header 1 | Input 1 under header 2 | Input 1 under header 3 |
| Input 2 under header 1 | Input 2 under header 2 | Input 2 under header 3 |
| Input 3 under header 1 | Input 2 under header 2 | Input 3 under header 3 |

Here, the table is styled with borders, set horizontal or vertical alignment for the table’s content, introduce dividers, padding, etc. However, you will need to learn some CSS to do stuff like this. We’re keeping this tutorial basic and won’t get into that in this article.

#### Quotations In HTML

There are different types of quotations, and other elements represent these quotations.

For a basic quotation, here’s an example:

<p>This is a sample. And <q>Here is our quote</q></p>

This comes out as:

This is a sample. And Here is our quote

When you use the <q> tag, quotation marks are automatically added to what is enclosed within the tag.

You can also use blockquotes, which is done with the <blockquote> tag.

#### Example

<p>

<blockquote>

This is a sample.

And Here is our blockquote. In this example, we try to demonstrate how to format text to indicate a blockquote in HTML.

This is different from ordinary quotes, in that the actual "quote" symbol may or may not be added depending on

CSS styling, but the text is highlighted.

</blockquote>

</p>

It will look like this:

*This is a sample. And Here is our blockquote. In this example, we try to demonstrate how to format text to indicate a blockquote in HTML. This is different from ordinary quotes, in that the actual "quote" symbol may or may not be added depending on CSS styling, but the text is highlighted*

#### Using Comments In HTML

When doing any form of coding, it is vital to learn how to include comments in your code. Comments are used to make your code more organized. You can include a reminder to yourself or a note to others to make things easier.

Initially, it might not seem important, but when you start writing hundreds or thousands of lines of code, and when things seem to start to get complicated, comments will come in handy.

Comments won’t be shown to the viewer by the browser. Remember that they can be seen in the source code.

Comments can also be used anywhere in the code. They won’t change the function of your code in any way.

You can include a comment by opening a bracket, including a double hyphen, adding your comment, including another double hyphen, and then closing your bracket.

#### Example

<!-- This is a comment used to indicate information I want to remember or that I want others to take note of when looking at this code in the future -->

It is also worth noting that, especially when debugging, you can comment out lines of code. This way, the code remains in the document but does not function since you have commented it.

#### Example

<!-- <b>He is a boy</b> -->

While “He is a boy” is supposed to come out as bold, we have commented it, and the browser would ignore it. In this scenario, as far as the browser is concerned, we have nothing.

This could also be used this way:

<b>He <!-- is a --> boy</b>

which shows **He boy**.

This is because we commented out “is a”. We’re just trying to show you that practically anything, within or outside an element, can be commented out. You can include a note in your comment for reference – it doesn’t change the nature of what is displayed.

#### Exercise 3

Work on the HTML file you previously created

Create a table with 3 columns and 4 rows. The first row should be the headers (ie. the title of each column), The first column should display 3 projects that you would like to create here at Developers Institute, the second column should display your 3 best skills in life, and the third column should display your 3 hobbies.

**Bonus:** Try to style the table with CSS (Hint: Check out "Inline CSS Table Borders" on Google)

Display a quote that represents you the best

#### Using IFrames In HTML

In an HTML document, an iFrame (“inline frame”) can be used to embed a file in the current document. In essence, you can use it to display another website or content from another web page inside the current one.

#### Example

<iframe src="https://websitesetup.org"></iframe>

With the above code, I tell the browser to display the content of https://websitesetup.org in a frame directly inside this page.

Iframes can be used to display a website, video, images, or any other form of content.

The code for this is simple:

<iframe src="URL"></iframe>

(simply replace “URL” with the link to the website/content you want to be displayed.)

You can further customize to specify specific values, e.g., your iframe’s height and width.

#### Example

<iframe src="URL" height="350px" width="400px"></iframe>

We tell the browser to display the iframe using a height of 350px and a width of 400px.

#### HTML Forms

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

You can see HTML forms on (almost) every website, as the “sign-in” section.

Here is a simple form example:

<form>

<label>Username:</label><input type="text">

<label>Password:</label><input type="password">

<input type="submit" value="Submit">

</form>

#### Defining A Form

To define a form, use the <form></form> element.

##### The Input Element

This is the most commonly used element within HTML forms.

The <input> tag allows you to specify various user input fields, depending on the type attribute. An input element can be of type:

* text field
* password field
* checkbox
* radio button
* submit button
* reset button
* file select box

And a lot more!

To change the type of a field, pass the type="<field\_type>" as attribute in the <input> element.

You can specify the size of the field by passing it in the attributes.

##### The Label Element

This element is created to define the label of a <input> element.

### Conclusion

In conclusion, HTML is more complicated than what you have in this guide. Do not give up hope – if you can grasp the basic concepts in this guide, you are up to a good start. It’s like knowing the ABC’s of a language, and everything else becomes much easier.

If you went through this tutorial intending to create a stunning website, we’re sorry to disappoint – it just doesn’t work that way. This is a basic HTML tutorial. It’s here to help you develop a foundation you can build upon. You can be sure that it will take significantly more effort – and a lot of development time, often using a combination of languages – to create an amazing website.

## HTML5 Semantic Elements

**Done**

### Table Of Contents

* + - [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#What_you_will_learn)
    - [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Useful_Resources)
    - [HTML Block and Inline Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#HTML_Block_and_Inline_Elements)
      * [1. Block-level elements:](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#1._Block-level_elements:)
        + [Section and Article](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Section_and_Article)
      * [2. Inline elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#2._Inline_elements)
    - [3. HTML5 Semantic Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#3._HTML5_Semantic_Elements)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Feedback)

Last updated : March, 31st 2021

### What You Will Learn

* HTML semantic elements
* HTML block and inline elements

### Useful Resources

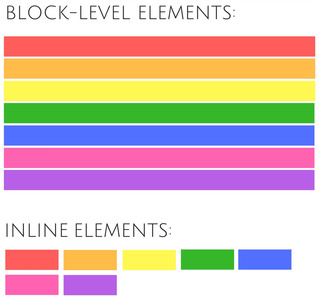
* [HTML Block and Inline Elements](https://www.w3schools.com/html/html_blocks.asp).
* [7 alternatives to the div tag](https://medium.com/@zac_heisey/7-alternatives-to-the-div-html-tag-7c888c7b5036).

HTML5 is the fifth version of the HTML scripting language. It supports a lot of new things that older versions of HTML do not.

For Example:  
In HTML5 there is something new called the Semantic Elements. Semantic elements have meaningful names which tell about the type of content: header, footer, navbar, … etc.

HTML5 introduces many semantic elements as mentioned below, making the code easier to write and understand for the developer and instructing the browser on how to treat them.

### HTML Block And Inline Elements



#### 1. Block-Level Elements:

Block-level element always starts on a new line and takes up the full width, from left to right. A block-level element can take up one line or multiple lines and has a line break before and after the element.

Example of block-level elements

* <div> : often used as a container for other HTML elements and to separate them for the rest
* <header>
* <nav>
* <article>
* <section>

I'm a div and I take the full width of a page

##### Section And Article

Both these elements are used for sectioning content. To decide which one of these you should choose, take note of the following:

1. An article is intended to be independently distributable or reusable.
2. A section is a thematic grouping of content.

#### 2. Inline Elements

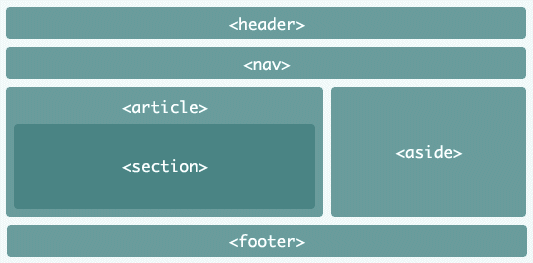
An inline element is usually used within other HTML elements. It does not cause a line break (and does not take up the full width of a page, only the space bounded by its opening and closing tag.

Example of inline elements:

* <span> : often used as a container for some text.
* <a>
* <img>

Hey, I'm a red span in a blue div

### 3. HTML5 Semantic Elements



## CSS Basics

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Useful_Resources)
    - [CSS Basics](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#CSS_Basics)
      * [1. What is CSS](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#1._What_is_CSS)
      * [2. CSS Syntax](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#2._CSS_Syntax)
      * [3. CSS Stylesheets](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#3._CSS_Stylesheets)
      * [4. CSS Simple Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#4._CSS_Simple_Selectors)
        + [Element Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Element_Selector)
        + [HTML id and class](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#HTML_id_and_class)

[Id Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Id_Selector)

[Class Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Class_Selector)

* + - * + [Universal Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Universal_Selector)
        + [Grouping Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Grouping_Selector)
      * [5. CSS Combinators Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#5._CSS_Combinators_Selectors)
        + [Descendant Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Descendant_Selector)
        + [Child Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Child_Selector)
        + [Sibling Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Sibling_Selector)
        + [Adjacent Sibling Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Adjacent_Sibling_Selector)
      * [6. CSS Properties](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#6._CSS_Properties)
        + [Colors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Colors)
        + [Backgrounds](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Backgrounds)
        + [Borders](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Borders)
        + [Margins](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Margins)
        + [Padding](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Padding)
        + [Elements dimensions](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Elements_dimensions)
        + [Text styling](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Text_styling)
        + [List styling](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#List_styling)
        + [Elements display](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Elements_display)

Last updated : April, 2nd 2021

## What You Will Learn

* Starting CSS from scratch
* The concept of Colors and Backgrounds in CSS.
* Fonts and Text
* Widths and Heights in CSS
* Margin, Padding
* Floating and Display.
* Links and Lists.
* Classes IDs

## Useful Resources

* [Basic Introduction to CSS](https://medium.com/the-andela-way/basic-introduction-to-css-81f6041b92d0)
* [35+ HTML & CSS Resources for Beginners](https://medium.com/level-up-web/30-html-css-resources-for-beginners-4e4d0af4b44b)
* [Learn Layout](https://learnlayout.com/display.html)

### CSS Basics

#### 1. What Is CSS

CSS stands for **C**ascading **S**tyle **S**heets. While HTML defines the structure of a web page. CSS describes how elements are displayed on the screen. HTML represents content, and CSS represents the appearance of the content.

#### 2. CSS Syntax

A CSS rule-set consists of a selector and some declarations.

The selector points to the HTML element you want to style. It can be a html tag, class or id.

Declarations are defined in a declaration block, surrounded by curly brackets {}, each declaration includes a CSS property name and value in this format name: value, and ends with a semicolon ;.

You can also add comments, surrounded by /\* and \*/.

For example:

h1 { /\* Selector \*/

color: blue; /\* Declaration 1 \*/

font-size: 12px; /\* Declaration 2 \*/

}

#### 3. CSS Stylesheets

We can write CSS in three different places.

First, it can be written in any HTML element,

For example :

<h1 style="color:blue; font-size:12px;">Hello World</h1>

A more readable way is to write CSS in a **stylesheet**; a **stylesheet** is a text file with the .css extension. The browser will read this text file, and the webpage will apply the style.

You need to specify the path of your **stylesheet** in your HTML code if you want it to be executed:

<html>

<head>

<link rel="stylesheet" href="/path/to/stylesheet.css"></link>

</head>

<body>

<h1>Hello world</h1>

</body>

</html>

CSS can also be written directly on the HTML page, but that method is deprecated.

#### 4. CSS Simple Selectors

CSS selectors are used to “find” (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) : Select elements based on a specific relationship between them
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) : Select elements based on a certain state
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) : Select and style a part of an element
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp): Select elements based on an attribute or attribute value)

##### Element Selector

Element selector selects HTML elements based on the element name (tag).

For example, this style will be applied on every <p> tag:

p {

color: red;

}

##### HTML Id And Class

HTML elements can have a class attribute, used to define similar styles for elements with the same class name. Elements can also have a unique id attribute.

###### Id Selector

The **id selector** uses the id attribute of an HTML element. To use it, write a hash **#** character, followed by the element’s id.

For example, to style this paragraph <p id="welcome">Hello World</p>, use:

#welcome{

color: red;

}

###### Class Selector

While the **class selector** uses the class attribute of the elements, to use it, write a period **.** followed by the class of the element.

For example, to style this image <img class="netfliximg" src="netflix-pic"/>, use:

.netfliximg{

width: 200px;

height: 200px;

}

##### Universal Selector

**Universal selector** is also available in CSS (and in a lot of computer languages), **\*** means “Everything.”

This style will be applied on every HTML elements of the page:

\* {

color: blue;

}

##### Grouping Selector

You can style more than one element at the time; just separate each selector with a comma **,**:

h1, h2, p {

color: red;

}

#### 5. CSS Combinators Selectors

##### Descendant Selector

The descendant selector matches **all descendants** of a specified element (meaning they are nested in it).

The following example selects all <p> elements inside <div> elements:

div p {

background-color: yellow;

}

##### Child Selector

The child selector selects all elements that are the **direct descendant** of a specified element.

The following example selects all <p> elements that are children of a <div> element:

div > p {

background-color: yellow;

}

##### Sibling Selector

The general **sibling selector** selects all elements that are siblings of a specified element.

Sibling means “that have the same parent”.

The following example selects all <p> elements that are siblings of <div> elements:

div ~ p {

background-color: yellow;

}

##### Adjacent Sibling Selector

The **adjacent sibling selector** selects all elements that are the adjacent siblings of a specified element.

“Adjacent” means “immediately following”.

The following example selects all <p> elements that are placed immediately after <div> elements:

div + p {

background-color: yellow;

}

##### Exercise 1

Create a stuctured HTML file

Copy this paragraph and style it with Inline CSS first and then with a CSS stylesheet.

*Hello Everyone, Congratulation on your first day of coding at DevelopersInstitute*

What happens if you style this paragraph both with Inline CSS and with a CSS stylesheet ?

#### 6. CSS Properties

##### Colors

In CSS, colors can be Hexadecimal Color Codes (#ffffff), rgb (rgb(255,255,255)), or color names (black).

##### Backgrounds

You can use CSS to add a background to your sections; it can be a background-color or a background-image.

background-image needs a URL to work.

body{

background-image: url("my\_img.png")

}

##### Borders

The CSS border properties allow you to specify the border-style, border-width, and border-color of an element’s border.

p {

border-style: solid;

border-width: 5px;

border-color: green;

}

##### Margins

The CSS margin properties are used to create space **around elements**, **outside** of any defined borders.

You can specify the width of margin for every side, but you can also set the margin for each element’s side.

p {

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

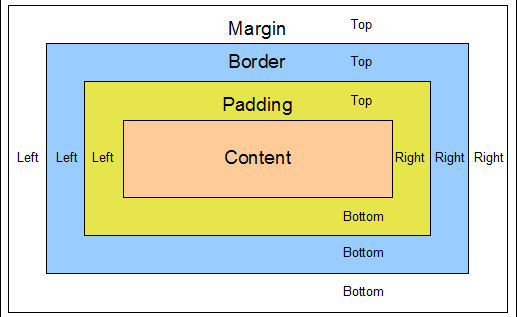
}

##### Padding

The CSS padding properties are used to generate space **around an element’s content**, **inside** of any defined borders.

It works the same way as margin.

See the [box model](https://www.w3schools.com/css/css_boxmodel.asp) to understand padding and margin



##### Elements Dimensions

The height and width properties are used to set an element’s height and width.

##### Text Styling

You can specify the color of the text, but also the alignment (with text-align), the text-decoration, the text-transformation, the text-indent and [much more](https://www.w3schools.com/css/css_text.asp)

p {

color: blue;

text-align: justify;

text-indent: 50px;

text-decoration: underline;

text-transform: uppercase;

}

##### List Styling

You can modify the icon in front of each list elements, with list-style-type to use a built in icon, or with list-style-image to add an image of your own.

##### Exercise 2

On the same HTML file

Recreate the green square of the Exercise 1, using margin, padding and borders

Try different types of borders, and shadows (Check it on Google)

##### Elements Display

You can modify the way elements are displayed.

The display property is controlling the layout of an element.

* Block-level: display: block; will display the element as a block, meaning stretch it out to the left and right as far as possible, and the next element will be placed below it.
* Inline: display: inline; elements don’t start on a new line and only take up as much width as necessary.
* Inline block: display: inline-block allows to set a width and height on the element and respect the padding, but doesn’t add a line break after the element.
* None: display: none won’t display the element. **Careful** : It will not hide it (meaning it won’t keep a blank place for the element). To hide an element, use visibility: hidden

See  
[This Course on CSS Layout](https://www.w3schools.com/css/css_display_visibility.asp)  
[Example with squares](https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1)  
[Example within texts](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_display)

##### Exercise 3

On the same HTML file

Recreate this flag with the **display property**



You can look at these links to help you :

* [css-display-property](https://www.geeksforgeeks.org/css-display-property/)
* [Understanding CSS Display: None, Block, Inline and Inline-Block](https://medium.com/better-programming/understanding-css-display-none-block-inline-and-inline-block-63f6510df93)

##### Exercise 5

On the same HTML file

Copy this code and do the following steps :

<div>

<ul>

<li>What is Lorem Ipsum?</li>

<li>Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, </li>

<li>when an unknown printer took a galley of type and scrambled it to make a type specimen book. </li>

</ul>

</div>

<div>

<p>It has survived not only five centuries, but also the leap into electronic typesetting,</p>

<ol>

<li> remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets </li>

</ol>

<p> containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>

</div>

1.Change the second li of the 1st div to blue

2.Make all the li of the first div in uppercase and in bold

3.Change the 1st paragraph of the second div so its background color will be pink and its color white

4.Make the 2nd paragraph of the second div with a background color light blue, with padding and margin

**Bonus:** Try to find a few ways to achieve each of the steps

## CSS Advanced

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Useful_Resources)
    - [Elements position](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Elements_position)
    - [Dealing with overflow](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Dealing_with_overflow)
    - [CSS Pseudo Class Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#CSS_Pseudo_Class_Selectors)
      * [hover](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#hover)
      * [children](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#children)
      * [not](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#not)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Feedback)

Last updated : April, 2nd 2021

## What You Will Learn

* Positions
* Overflow
* Pseudo Class Selector

## Useful Resources

* [Learn Layout](https://learnlayout.com/display.html)

### Elements Position

The position property specifies the type of positioning method used for an element (static on the page, relative to his parent, etc..)

It can be:

* static: Default value. Elements render in order, as they appear in the document flow
* relative: The element is positioned relative to its normal position; you can add left, right, top or bottom arguments to add a gap by the element.
* fixed: Always stays in the same place even if the page is scrolled, left, right, top and bottom are used to position the element
* absolute: The element is positioned relative to its first positioned (not static) ancestor element
* sticky: The element is positioned based on the user’s scroll position

[More on CSS Position](https://www.w3schools.com/css/css_positioning.asp)  
[CSS Position Tricks](https://css-tricks.com/almanac/properties/p/position/)

#### Exercise 1

On the same HTML file (as the one in the CSS lesson)

Recreate this image with the **position** property

The elements are : a parent box (grey), a blue box and an orange box



### Dealing With Overflow

The overflow property specifies whether to clip the content or add scrollbars when an element’s content is too big to fit in the specified area.

It can be:

* visible - Default. The overflow is not clipped. The content renders outside the element’s box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

[Look at the demonstration Here](https://www.w3schools.com/cssref/playit.asp?filename=playcss_overflow)

### CSS Pseudo Class Selectors

A pseudo-class is used to style an element when he is in a special state (for example, when the user hovers over it)

The syntax is : element:pseudo-class {...}

#### Hover

hover means “mouse is over it”, for example, to change the color of a <div> when the mouse goes over it:

div:hover {

background-color: blue;

}

#### Children

You can target elements only if they are the nth child of their parent. For example, to style <p> tags that are the first child of another element, use:

p:first-child{

color: blue;

}

The same thing works with last-child or nth-child(n)

#### Not

The not pseudo-class allows you to exclude elements; for example, select all the divs that are not of the ignore class.

div:not(.ignore){

background-color: yellow;

}

For more, see [this](https://www.w3schools.com/css/css_pseudo_classes.asp).

## Basic HTML5 Template For Any Project

**Done**

### Table Of Contents

* + - [DOCTYPE](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#DOCTYPE)
    - [HTML element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#HTML_element)
    - [HEAD element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#HEAD_element)
    - [Fair Competition](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Fair_Competition)
      * [Next is history](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Next_is_history)
      * [Next step](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Next_step)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Feedback)

Last updated : April, 2nd 2021

As you learn HTML5 and add new technologies to your toolbox, you may want to build a boilerplate for yourself to start all HTML5-based projects. We encourage this, and you can also consider using one of the online resources, which gives you an essential starting point for HTML5.

In this article, we will see how to get started. Let’s start with a simple, basic HTML5 page:

<!doctype html>

<html>

<head>

<meta charset = "utf-8">

<title> The HTML5 Herald </ title>

<meta name = "description" content = "The HTML5 Herald">

<meta name = "author" content = "SitePoint">

<link rel = "stylesheet" href = "css/styles.css?v = 1.0">

</head>

<body>

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

</body>

</html>

With the basic template in place, let’s examine some of the markup’s important parts and how they differ from the way HTML was written before HTML5.

### DOCTYPE

First, we have a “document type declaration” or doctype. This is just one way to tell the browser (or any other parser) what type of document is being viewed. For HTML files, it represents a specific version and style of HTML.

The doctype should always be the first item at the top of any HTML file. Many years ago, the doctype declaration was ugly and hard to remember mess. For XHTML 1.0 Strict:

<! DOCTYPE html PUBLIC "-// W3C // DTD XHTML 1.0 Strict // EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

HTML4 conversion:

<! DOCTYPE HTML PUBLIC "-// W3C // DTD HTML 4.01 Transitional // EN"

"http://www.w3.org/TR/html4/loose.dtd">

Although the long string of text at the top of the document didn’t hurt us (apart from forcing visitors to our website to download some extra bytes), HTML5 has eliminated the obscure things. What you need now is:

<! doctype html>

Simple and fair. Doctype can be written in uppercase, lowercase, or mixed case. You will notice that the “5” is missing from the statement. Although the current iteration of web markup is called “HTML5”, it is really just an evolution of the previous HTML standard-future specifications will only be an evolution of what we have today.

Since browsers are often required to support all existing content on the web, there is no need to rely on document types to tell them which features a given document should support. In other words, doctype alone will not make your page compatible with HTML5. It all depends on the browser.

You can use one of these two old document types with new HTML5 elements on the page, and the page will look the same as when using the new doctype.

### HTML Element

The following HTML document is an HTML element, which has not changed significantly in HTML5. In our example, we set lang the attribute’s value to en, which specifies that the document is in English. In the based xhtmlsyntax, xmlns attributes need to be included. In HTML5, this is no longer necessary, and even lang attributes are not necessary for the validation or correct operation of the document.

This is what we have so far, including closed </html> tags:

<! doctype html>

<html lang = "en">

</ html>

### HEAD Element

The next part of the page is the <head> section. head The first line defines the document’s character encoding. This is another element that has been simplified since XHTML and HTML4 and is an optional feature, but recommended. In the past, you could write:

<meta http-equiv = "Content-Type" content = "text / html; charset = utf-8">

HTML5 improves this by minimizing character encoding tags:

<meta charset = "utf-8">

In almost all cases, UTF-8 is the value you will use in the document. A complete explanation of character encoding is beyond this article’s scope, and you may not be interested in it. Nonetheless, if you want to dig deeper, you can read topics about [W3C](https://html.spec.whatwg.org/multipage/infrastructure.html#encoding-terminology) or  
[WHATWG](https://html.spec.whatwg.org/multipage/infrastructure.html#encoding-terminology) .

**Note**: To ensure that all browsers can read character encoding correctly, the encoding declaration must be included within the first 512 characters of the document. It should also appear before any content-based elements (such as the <title> element that follows it in our sample site).

We still have a lot to write about this topic, but we want to keep you awake-so we won’t tell you those details! Now, we are content to accept this simplified statement and then move on to the next part of the document:

<title> The HTML5 Herald </ title>

<meta name = "description" content = "The HTML5 Herald">

<meta name = "author" content = "SitePoint">

<link rel = "stylesheet" href = "css/styles.css? v = 1.0">

In these lines, HTML5 is almost indistinguishable from the previous syntax. The page title’s declaration (the only required element in the header) is the same as before. The meta tags we include are just some optional examples to indicate these tags’ position; you can put as many valid meta elements here.

This markup block’s key part is the style sheet, which is contained using the idiomatic link element. Except href and rel, link does not require other attributes. The type attribute (common in older versions of HTML) is not required and is not required to indicate the style sheet’s content type.

### Fair Competition

When HTML5 was introduced, it included many new elements, such as articles and sections. You may think that this is the main problem with the support of unrecognized elements in older browsers, but you’re mistaken. This is because most browsers don’t actually care what tags you use.

If you have an HTML document with a recipe tag (or even a ziggy tag), and your CSS attaches some styles to the element, almost every browser will treat it as a completely regular operation; nothing Apply your style without complaints.

Of course, such hypothetical documents will not be verifiable, and there may be accessibility issues. Still, it will render correctly in almost all browsers-except for older versions of Internet Explorer (IE). Before version 9, IE prevented unrecognized elements from receiving styles.

These mysterious elements are considered “unknown elements” by the rendering engine, so you cannot change their appearance or behavior. This includes the elements we imagine and any elements that were not defined when the browser version was developed. This means new HTML5 elements.

The good news is that the usage of IE has dropped. The global usage of IE11 has dropped to about 2.7% (as of 2018), and the previous version has almost disappeared from the map.

However, if you need to support ancient browsers, you can still use the trusted HTML5 Shiv, a straightforward JavaScript initially developed by John Resig. Inspired by Sjoerd Visscher’s idea, it enables new HTML5 elements to be styled in older versions of IE.

However, in reality, it is not needed now. All modern browsers and even the latest older versions support HTML5 elements. The one exception is that some browsers do not recognize the newer significant elements. However, for these browsers, you can still use this element as long as you add the appropriate style (such as setting it as a block element).

#### Next Is History

Looking at the rest of the start template, we have common body elements with their end tags and closing html tags. We also reference a JavaScript file in the script element.

Much like the link tag discussed earlier, the <script> tag does not need to declare a type attribute. If you have ever written XHTML, you may remember that your script tags look like this:

<script src = "js/scripts.js" type = "text/javascript"></script>

Since JavaScript is the only true scripting language used on the Web, and all browsers assume that you are using JavaScript, even if you do not explicitly state the fact, the type attribute is unnecessary in HTML5 documents:

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

We place the script element at the bottom of the page to comply with best practices for embedding JavaScript. This has to do with page load speed; when the browser encounters a script, it will pause the download and render the rest of the page while parsing it.

This will cause the page to load much slower when including large scripts at the top of the page before loading anything. That’s why most scripts should be placed at the very bottom of the page so that they will only be parsed after the rest of the page is loaded.

However, in some cases (such as using HTML5 shiv), the script may need to be placed at the head of the document because you want to take effect before the browser starts rendering the page.

#### Next Step

One way to take HTML5 to the next level is to try the HTML5 Boilerplate. This regularly updated resource provides a convenient starting point for your project, with all the latest best practices established by hundreds of the best programmers in the world.

Even if you want to go through the code and see how certain elements are used these days, such as the various meta-elements in the document header, it’s worth downloading and checking out.

Another way to take your website or web application development to the next level is to try a modern framework that is widely used today.

The above is the detailed content of the basic HTML5 template suitable for any project.

## What Is The Use Of Asterisk (\*) Select

**Done**

# What Is The Use Of Asterisk (\*) Selector In CSS ?

By **Shaishta Pandea** - Developers Institute Alumni

### Overview

The **asterisk (\*)**, also known as the **CSS** universal selector, is used to select all items in an HTML file. **CSS selectors** are used to select the sections of your web page you wish to style.

### Example Of CSS Universal Selectors

#### Example 1

In the first result, all of the text is in black font color with a purple background, as our code overwrites the default background color for all HTML elements.

**HTML - index.html**

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div id="content"/>

<h1>Example of CSS.</h2>

<h2>Example of grouping of CSS universal selectors.</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a urna elit. Integer malesuada tempus enim nec rhoncus.</p>

</body>

</html>

**CSS - style.css**

\* {

color: black;

/\* color of all the elements should be black \*/

background: purple;

/\* purple background is set for all the elements \*/

}

**Output**



#### Example 2

In the second result, all of the text is the same size, as our code overwrites the default font-size for all HTML elements.

**HTML - index.html**

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div id="content"/>

<h1>Example of CSS.</h2>

<h2>Example of grouping of CSS universal selectors.</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a urna elit. Integer malesuada tempus enim nec rhoncus.</p>

</body>

</html>

**CSS - style.css**

\* {

color: black;

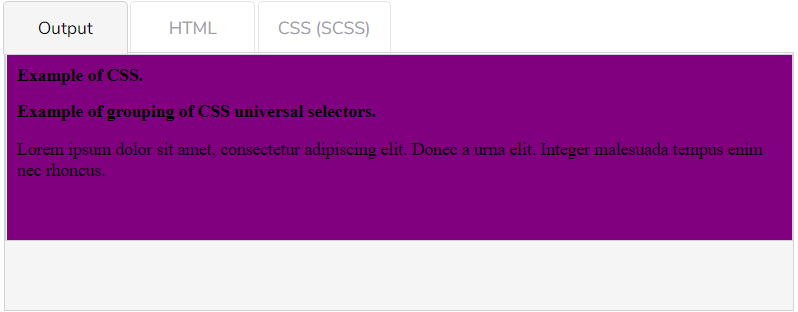
/\* color of all the elements should be black \*/

background: purple;

/\* purple background is set for all the elements \*/

font-size:14px;

}



#### Example 3

Let’s say that you have a document that contains the following HTML elements:

* An H1
* Paragraphs
* A table
* Pre-formatted text
* An unordered list

Then, use the declaration below to make sure all elements are purple.

**CSS - style.css**

\* {color: purple;}

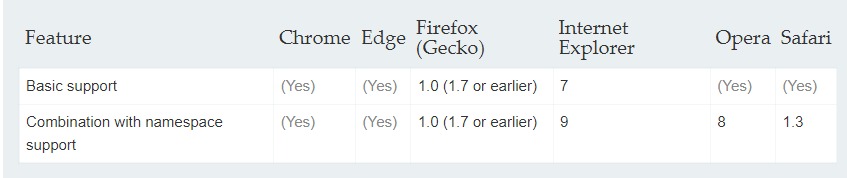
Use the previous declaration instead of the code below to simplify your code.

**CSS - style.css**

BODY, H1, P, TABLE, TR, TD, TH, PRE, UL, LI {color: purple;}

### Browser Compatibility

Browser Compatibility is the manner in which a web page looks in different web browsers. Different browsers read the website code differently. In other words, Chrome will render a website differently than Firefox or Edge will. Thus, it is important to know in which browser, the code written by a programmer is acceptable.



The table above shows that the CSS universal selector is supported by Chrome, Edge, Firefox, Internet Explorer, Opera and Safari.

## What Does The “+” Plus Sign CSS Select

**Done**

# What Does The “+” Plus Sign CSS Selector Mean?

By **Oceanne Lai** - Developers Institute Alumni

### Introduction

CSS has many selectors, and the + sign is one of them. A **selector** is a pattern of elements that directs the browser to which HTML elements should have the selected CSS property.

The + **sign** is best known as an adjacent sibling combinator, which means it separates two selectors, then combines the second selector with other elements which are placed immediately after the chosen selector. The + sign selector is used when the user wants to have the same styling for different elements.

#### Syntax

**CSS - style.css**

element1 + element2{

style code

}

The browsers that are supported by the + selector are listed below:

* Apple Safari
* Google Chrome
* Internet Explorer 7.0
* Opera
* Firefox
* Edge

#### Example

In this example code, we will apply different styles to the tags.

**HTML - index.html**

<html>

<head>

</head>

<style type="text/css">

h2 + p {

color: lightblue;

text-align: center;

font-size:20px;

background-color: yellow;

}

p{

color: palevioletred;

font-weight: bold;

}

body{

text-align: center

}

</style>

<body>

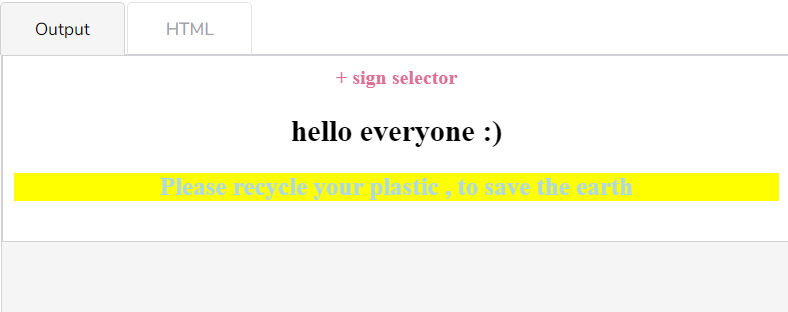
<p> + sign selector </p>

<h2> hello everyone :) </h2>

<p> Please recycle your plastic , to save the earth</p>

</body>

</html>



* As you can see in the example above, h2+p and p don’t have the same style, as the first <p> is placed before the <h2>.
* So, the style doesn’t apply to the first <p>.
* The + sign essentially applies a style to all the tags that come after the first selector.

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1898#breadcrumbs)

## How To Apply Style To Parent

**Done**

# How To Apply Style To Parent If It Has A Child With CSS

By **Varshana Gopal** - Developers Institute Alumni

### Introduction

In this shot, we will learn how to apply style to a parent class that has a child class with CSS.

It’s easy to apply style to a child element, but if you want to apply style to a parent class that already has child elements, you can use the CSS selector **child combinators (>)**, which are placed between two CSS selectors. For example, div > p selects all <p> elements where the parent is a <div> element.

#### Syntax

**CSS - style.css**

selector1 > selector2{

style code

}

Selector 1 is the parent element and selector 2 is the child element.

**HTML - index.html**

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title></title>

<style type="text/css">

.div1{

background-color: blue;

}

.div1 > .div2{

background-color: red;

width: 50%;

}

p{

background-color: yellow;

width: 50%;

}

</style>

</head>

<body>

<div class="div1">

An HTML example:

<div class="div2">

Good Morning

<p>How was your day?</p>

Mine was great!

</div>

Good-Bye!

</div>

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

</body>

</html>

**JAVASCRIPT - script.js**

document.addEventListener("DOMContentLoaded", function () {

pTag = document.querySelector("div");

newVal = document.createElement("p");

newVal.innerHTML = 'Hello, Hello';

pTag.appendChild(newVal);

});

// Async Tests Example

/\*

window.onModulesLoaded = new Promise( function( resolve, reject ) {

setTimeout(function() {

pTag = document.querySelector("div");

pTag.innerHTML = '';

newVal = document.createElement("p");

newVal.innerHTML = 'Hello World';

pTag.appendChild(newVal);

resolve();

}, 100)

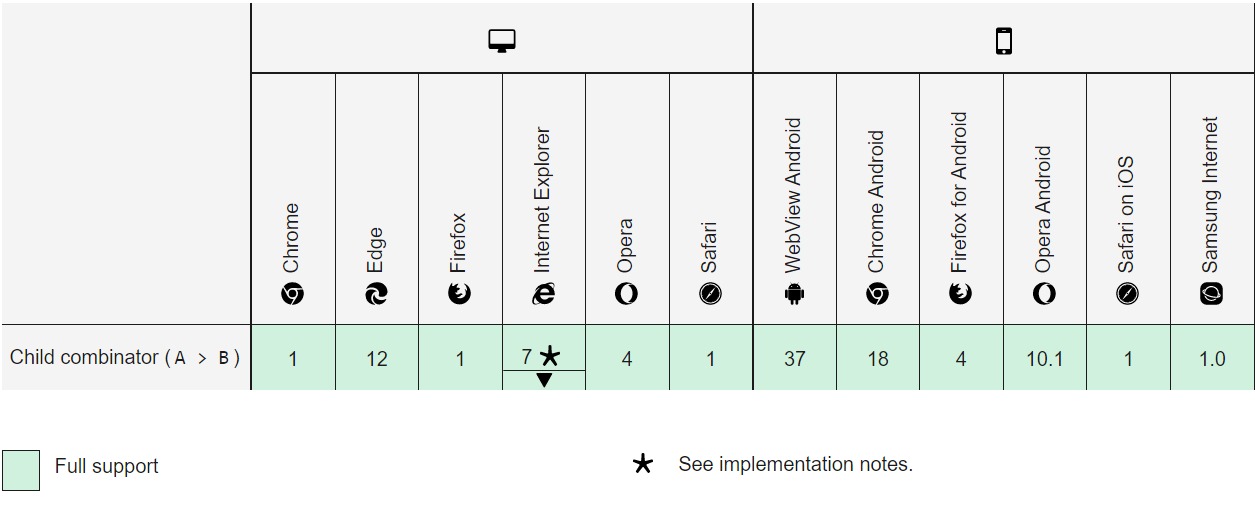
});

\*/



In the example above, .div2 is the child element of .div1. The style can also be applied in a CSS file which will then have to be linked to the HTML file. However, this applies to immediate children. For nested children, we’ll have to use the descendant combinator.

#### Browser Compatibility



[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1899#breadcrumbs)

### The Command Line

Before understanding the Command Line’s goal (Or Command Prompt, or Terminal), we need to talk about GUI.

### What Is A GUI?

GUI or graphical user interface is a system of interactive visual components for computer software.

A GUI uses windows, icons, and menus to carry out commands, such as opening, deleting, and moving files. Although a GUI operating system is navigated using a mouse and a keyboard.

For example, if you want to open a GUI system program, you would move the mouse pointer to the program’s icon and double-click it.

### What Is A Command-Line?

A window, usually called the **command line** or **command-line interface**, is a text-based application for viewing, handling, and manipulating files on your computer.

Using a command line, you can perform almost all the same tasks with a GUI. However, many tasks can be completed quicker and can be easier to automate and do remotely.

### Cheatsheets

#### For MAC Users : Follow This [Tutorial](https://www.makeuseof.com/tag/mac-terminal-commands-cheat-sheet/)

#### For WINDOWS Users : Follow This [Tutorial](http://www.cs.columbia.edu/~sedwards/classes/2017/1102-spring/Command%20Prompt%20Cheatsheet.pdf) And This [Tutorial](https://enexdi.sciencesconf.org/data/pages/windows_vs_mac_commands_1.pdf)

#### For LINUX Users : Follow This [Tutorial](https://cheatography.com/davechild/cheat-sheets/linux-command-line/)

### Some More Useful Commands:

* Create a new empty file
* Windows
* C:\Users\John> copy NUL test.txt
* OR
* C:\Users\John> type NUL > test.txt
* Launching an editor from the Command Line

**Visual Studio code**

Windows & Linux

Open the whole folder in VS.

C:\Users\John> code .

Open a file in VS

C:\Users\John> code hello.html

**For Mac Users**

* According to the documentation, “Users on macOS must first run a command (Shell Command: Install ‘code’ command in PATH) to add VS Code executable to the PATH environment variable.”
* Therefore, follow this [tutorial](https://code.visualstudio.com/docs/setup/mac)

**Sublime Text**

Windows, Mac & Linux  
Follow this [tutorial](https://scotch.io/tutorials/open-sublime-text-from-the-command-line-using-subl-exe-windows#:~:text=A%20nice%20shortcut%20to%20do,folder%20of%20your%20Sublime%20installation)

* Open the file explorer
* Windows:
* C:\Users\John>start .
* Mac & Linux:
* C:\Users\John>open .
* Open a specific folder
* Windows:
* C:\Users\John>cd Documents\Game
* C:\Users\John\Documents\Game> start .
* Mac & Linux:
* C:\Users\John>cd Documents\Game
* C:\Users\John\Documents\Game> open .

Go several directories up.

C:\Users\John\Documents\Games\Mario>cd ../..

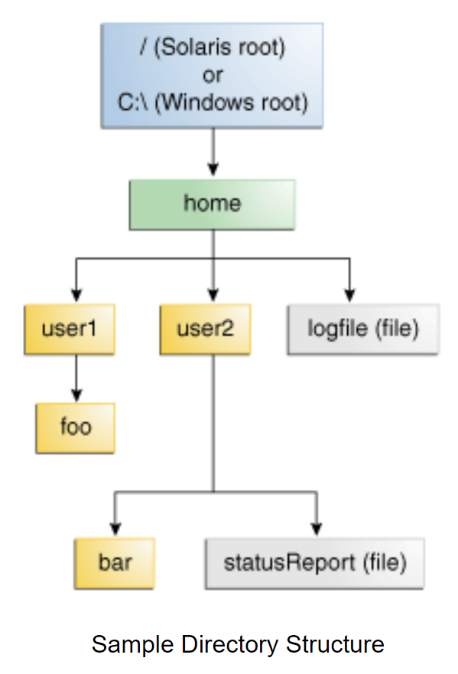
You get to C:\Users\John\Documents>

C:\Users\John\Documents\Games\Mario>cd ../../..

You get to: C:\Users\John>

### Absolute And Relative Paths

#### What Is A Path?



Its path identifies a file through the file system. For example, the status report file in the previous figure is described by the following notation.

/home/user2/statusReport

### Absolute Path

Absolute paths contain a complete URL, including a protocol, the website’s domain name, and possibly a specific file, subfolder, or page name. For example https://www.facebook.com/

<a href="https://www.facebook.com/">My Facebook</a>

Note: While your browser may let you omit the protocol, https://, you should always include the scheme (e.g. http:// or https://) when coding absolute links to make sure they work for all visitors.

### Relative Path

The relative path only includes the name of a specific file or page relative to the current path. For example: cat.html

<a href="/public/cat.html">My Cat</a>

In this example, you’re telling the browser to look in the current folder (**public**) containing the file you want the user taken to (**cat.html**).

Of course, if you just pasted /public/cat.html into a browser’s search bar, it wouldn’t take you where you wanted to go.

Therefore, the link’s path is **relative to the current document being displayed by the browser**.

### HTML Relative Paths

**Some theory first**

A relative file path points to a file relative to the current page.

/images/picture.jpg: the file path points to a file in the images folder located at the root of the current web.

images/picture.jpg: the file path points to a file in the images folder located in the current folder.

../images/picture.jpg: the file path points to a file in the images folder located in the folder one level above the current folder.

/ = Root directory

. = This location

.. = Up a directory

./ = Current directory

../ = Parent of current directory

../../ = Two directories backwards

**First Example**

Let’s understand it with an example; here is the tree of the folder.

vacations

├───blog

│ vacation.html

│

└───images

beach.jpg

The directory vacations has 2 subfolders. Each subfolder has a file. We want to link the image file in the HTML file.

**vacation.html**

<img src="../images/beach.jpg">

**Second Example**

vacations

└───blog

│ vacation.html

│

└───images

beach.jpg

Now the directory images is a subfolder of the blog directory.

**vacation.html**

<img src="images/beach.jpg">

OR

<img src="./images/beach.jpg">

**Third Example**

vacations

│ beach.jpg

│

└───blog

│ vacation.html

The image is now in the parent directory.

<img src="../beach.jpg">

**Fourth Example**

vacations

├───blog

│ beach.jpg

│ vacation.html

**vacation.html**

<img src="./beach.jpg">

or

<img src="beach.jpg">

## Exercise 1

Using only the Command Line to create the structure of the main folder:

Create a new folder on your computer called movies. Inside, add an image icon that will represent your website.

Inside the main folder, create 2 folders: movie\_website and movie\_images. Add some images in the movie\_images folder (images of films created by the same director).

Create an HTML page in the movie\_website folder called movies\_list. It displays the list of movies and their images. In the navbar, display the image icon of your website. At the end of the body, create an anchor, which links to the HTMl file we will create below

Create another HTML file called director\_details, that will display all the details of the director you chose. Create an anchor, which links to his/her Wikipedia.

In this exercise, add comments to your files everytime you use a relative or absolute path. And explain why this path is relevant

## Git & Github - Register & Install

**Done**

### Table Of Contents

* + [Useful resources](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/819#Useful_resources)
    - [Some information to start](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/819#Some_information_to_start)
    - [Git tutorial](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/819#Git_tutorial)
      * [Registration and Configuration](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/819#Registration_and_Configuration)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/819#Feedback)

Last updated : April, 3rd

## Useful Resources

* [Git and GitHub learning resources](https://help.github.com/en/articles/git-and-github-learning-resources)
* [Git cheat sheet](https://developers.institute/wp-content/uploads/2019/10/git-cheat-sheet-education.pdf)
* [Student Generated Top Resources](https://github.com/devtlv/Ressource_JavaScript_Bootcamp). It is a list of great resources as you go through the course and also as you progress through your career as a web developer. You can visit it if you need extra information and contribute your favorite resources to improving it. I hope this helps you along your journey with a little “extra”!
* [Beginner’s Tutorial on GitHub’s website](https://guides.github.com/activities/hello-world/)
* [Great video introduction to the usages of GitHub](https://www.youtube.com/watch?v=w3jLJU7DT5E)
* [GitHub Documentation and cheat sheets](https://git-scm.com/docs/git-checkout)
* [How to install Git on Linux, Mac, or Windows](https://www.linode.com/docs/development/version-control/how-to-install-git-on-linux-mac-and-windows/)
* [The Udemy videos by Andrei (already embedded on di-learning) are great at explaining branches](https://www.di-learning.com/courses/42/sections/186/chapters/999)

### Some Information To Start

**What the heck is GitHub, and why are developers so excited about it**?

You may have heard that **GitHub is a code sharing and publishing service**, or that it’s a social networking site for programmers. Both statements are true, but neither explains why GitHub is special.

At the heart of **GitHub is Git**, an open-source project started by Linux creator Linus Torvalds.

Matthew McCullough, a GitHub trainer, explains that Git, like other **version control systems, manages and stores revisions of projects**. Although it’s **mostly used for code**, McCullough says Git could be used to manage any other type of file, such as Word documents or Final Cut projects. Think of it as a filing system for every draft of a document.

Your familiarity with GitHub and ability to use Git is crucial and required in every coding project you will work on during your college years and beyond.

After completing this task, you would be finally ready to contribute to Open Source, so take your time and go through it slowly and ask for help wherever you get stuck.

Before starting the tutorial, **read this**[**article**](https://opensource.com/resources/what-is-git)**on Git** to better understand what Git is.

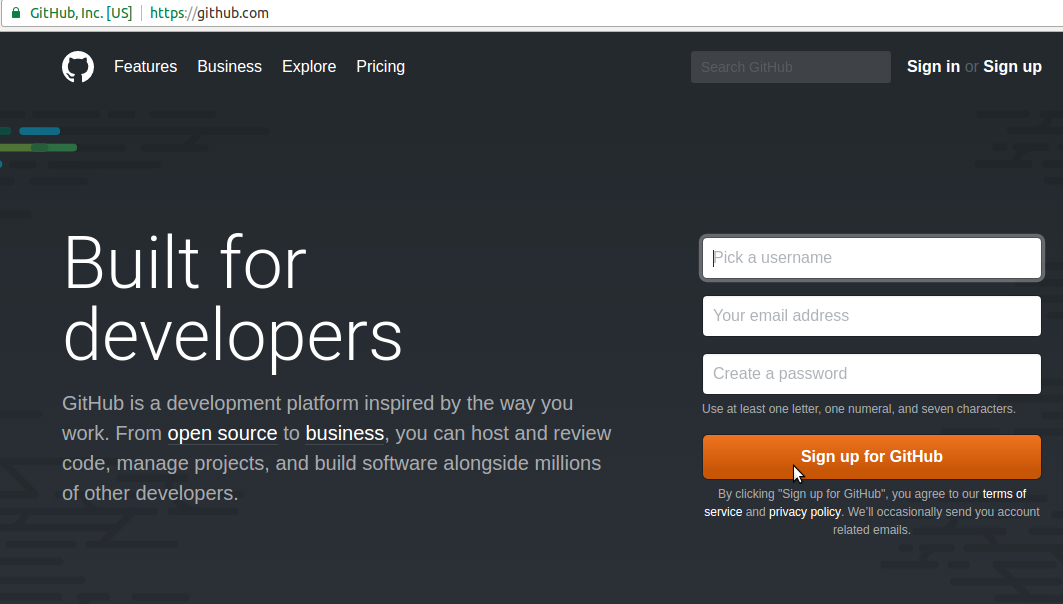
### Git Tutorial

#### Registration And Configuration

##### Step 1: Registration

First and foremost, each of you will need a Github account. It is fairly easy to register, and similar to most of the other sites, it offers login using your Gmail or other Social Network ID’s.

###### [Register Here](https://www.github.com/join)



Before continuing with the rest of the tutorial it is highly recommended that you go through the [**Hello World Tutorial**](https://guides.github.com/activities/hello-world/)offered by GitHub to get familiar with the Github workflow.

##### Step 2 : Install Git

You have to install Git client in your system.

For most MacOS and Linux systems Git is already pre-installed. To check it, go into your Command Line (or Terminal) and enter the command

git

If you don’t have it installed already, it will prompt you to install it.

* [Download GIT for OSX](https://git-scm.com/download/mac)
* [Download GIT for Windows](https://gitforwindows.org/)
* [Download GIT for Linux](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)

##### Step 3 : Configuring Git

Now that we’ve installed git on our computer, we will need to add some quick configurations. Many options can be fiddled with, but we will set up the most important ones: our username and email. Open a terminal and run these commands:

git config --global user.name "github username"

git config --global user.email "email you signup with in github"

Every action we do in Git will now have a stamp with our name and address on it. This way, users always know who did what, and everything is way more organized.

**IMPORTANT** : to check if your username and email were updated, write those commands:

git config user.name

git config user.email

## Git & GitHub - Steps

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#What_you_will_learn)
    - [I. Some important commands](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#I._Some_important_commands)
      * [1. Create a repository on Github](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#1._Create_a_repository_on_Github)
      * [2. Creating a new repository - git init](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#2._Creating_a_new_repository_-_git_init)
      * [3. Change the default branch](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#3._Change_the_default_branch)
      * [4. Checking the status - git status](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#4._Checking_the_status_-_git_status)
      * [5. Connecting to a remote repository - git remote add](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#5._Connecting_to_a_remote_repository_-_git_remote_add)
      * [6. Staging - git add](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#6._Staging_-_git_add)
      * [7. Commiting - git commit](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#7._Commiting_-_git_commit)
      * [8. Uploading to a server - git push](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#8._Uploading_to_a_server_-_git_push)
    - [II. Some important commands when working in a group](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#II._Some_important_commands_when_working_in_a_group)
      * [1. Cloning a repository - git clone](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#1._Cloning_a_repository_-_git_clone)
      * [2. Getting changes from a server - git pull](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#2._Getting_changes_from_a_server_-_git_pull)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/38#Feedback)

Last updated : April, 3rd

## What You Will Learn

* Practicing using terminal
* How to create a GitHub account and set up a repository
* Make commits
* Learn commands to move files and directories to and from the GitHub servers
* Introduce changes with pull requests

### I. Some Important Commands

#### 1. Create A Repository On Github

To create a new repository, you have to go on Github and click on the “New Repository” button.  
Add also a ReadMe file to your project.

#### 2. Creating A New Repository - Git Init

Git stores its files and history directly as a folder in your project. To set up a new repository, we need to open a terminal, navigate our project directory and run git init. This will enable Git for this particular folder and create a hidden .git directory where Git will store the repository history and configuration.

Create a folder, go to the terminal and write these commands:

cd <path-to-your-folder>

git init

The command line should respond with something along the lines of:

Initialized empty Git repository in <path-to-your-folder/.git/>

This means that our repo has been successfully created but is still empty.

**IMPORTANT** : This command is only written **once**: when you create a new local folder.  
This command needs to be written in the most general local folder :

For example:

- GeneralFolder --> HERE YOU HAVE TO USE THE COMMAND git init

-- Subfolder

-- file

#### 3. Change The Default Branch

You need to change the default branch to **main**, therefore enter this command:

git branch -M main

#### 4. Checking The Status - Git Status

Git status is another must-know command that returns information about the current state of the repository: is everything up to date, what’s new, what’s changed, and so on. Running git status in our newly created repo should return the following:

git status

On branch main

Initial commit

You should all the time check the status of your repo.

#### 5. Connecting To A Remote Repository - Git Remote Add

First, create an authentication token on Github, tutorial [here](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token) and save it.

To upload something to a remote repo, we first have to establish a connection with it.

To link our local repository with the one on GitHub, we execute the following line in the terminal:

git remote add origin https://token@github.com/Githubusername/repoName.git

For example:

git remote add origin https://1234@github.com/JohnD/projectGame.git

A project may have many remote repositories at the same time. To be able to tell them apart, we give them different names. Traditionally the main remote repository in git is called origin.

**IMPORTANT** : This command is only written **once** : when you create a new local folder.

#### 6. Staging - Git Add

Git has the concept of a “staging area”. You can think of this as a blank canvas, which holds the changes you would like to commit. It starts empty, but you can add files to it (or even single lines and parts of files) with the git add command, and finally commit everything (create a snapshot) with git commit.

After you created a new file in your local repository, go to the path of the repository on the terminal, then write this command :

cd <path-to-your-folder>

git add <name-of-file>

If we want to add everything in the directory, we can use:

git add .

Rechecking the status should return a different response from before.

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached ..." to unstage)

new file: <name-of-file>

Our file is ready to be committed. The status message also tells us what has changed about the files in the staging area - in this case, it’s new file, but it can be modified or deleted, depending on what has happened to a file since the last git add.

#### 7. Commiting - Git Commit

A commit represents the state of our repository at a given point in time. It’s like a snapshot, which we can go back to and see how things were when we took it.

To create a new commit, we need to have at least one change added to the staging area (we just did that with git add) and run the following:

git commit -m "<name-of-the-commit>"

This will create a new commit with all the changes from the staging area. The -m "<name-of-the-commit>" part is a custom user-written description that summarizes the changes done in that commit. It is considered good practice to commit often and always write meaningful commit messages.

#### 8. Uploading To A Server - Git Push

Now it’s time to transfer our local commits to the server. This process is called a **push**, and is done every time we want to update the remote repository.

The Git command to do this is git push and takes two parameters - the name of the remote repo (we called ours origin) and the branch to push to (main is the default branch for every repo).

$ git push origin main

Counting objects: 3, done.

Writing objects: 100% (3/3), 212 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://1234@github.com/JohnD/projectGame.git

\* [new branch] main -> main

Depending on the service you’re using, you will need to authenticate yourself to push to go through.

## Exercise 1

1. Create a new repository on Github called **git\_tutorial\_yourname**

2. Create a new local folder called **git\_tutorial\_yourname**

3. Use the commands git init, git status, git branch -M main and git remote add origin ... as explained above

4. Create a new file in you local folder : **hello.txt** and add this sentence inside *Hey Everyone, is working on Git and Github*

5. Check again the status of the repo . You should see a returned message that states that **hello.txt** is untracked. This means that the file is new and Git doesn't know yet if it should keep track of the changes happening to that file or just ignore it. To acknowledge the new file, we need to stage it.

6. Use the commands git add . and git commit -m "your message" as explained above

7. Check again the status of the repo

8. Use the commands git push origin main as explained above. If everything was done correctly, when you go in your web browser to the remote repository created earlier, **hello.txt** should be available there.

### II. Some Important Commands When Working In A Group

#### 1. Cloning A Repository - Git Clone

At this point, people can see and browse through your remote repository on Github. They can download it locally and have a fully working copy of your project with the git clone command:

git clone <path-of-your-remote-repository>

A new local repository is automatically created, with the Github version configured as a remote.

#### 2. Getting Changes From A Server - Git Pull

If you make updates to your repository, people can download your changes with a single command - **pull**:

git pull origin main

From https://github.com/tutorialzine/awesome-project

\* branch main -> FETCH\_HEAD

Already up-to-date.

### Exercise 2

1. Work now in pairs

2. The person 1 should clone the person 2 repository (lets call this repository RepoB). Than you should both check the status of your local RepoB

3. The person 2 should add the person 1 as a Collaborator. To do so in his Github Repository, the person 2 goes to Settings/Manage Access/Add Collaborator

4. In his local RepoB, the person 2 will change the sentence of **hello.txt** by *Hey Everyone, are working together on Git and Github*. Than add, commit and push

5. The person 1 should look at his local RepoB. Has the sentence in **hello.txt** changed ?

6. In order to see the changes of person 2, person 1 should use the command git pull origin main inside his local RepoB. Has the sentence in **hello.txt** changed ?

## CSS Advanced

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#Useful_Resources)
    - [Elements position](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#Elements_position)
    - [Dealing with overflow](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#Dealing_with_overflow)
    - [CSS Pseudo Class Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#CSS_Pseudo_Class_Selectors)
      * [hover](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#hover)
      * [children](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#children)
      * [not](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#not)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/1140#Feedback)

Last updated : April, 2nd 2021

## What You Will Learn

* Positions
* Overflow
* Pseudo Class Selector

## Useful Resources

* [Learn Layout](https://learnlayout.com/display.html)

### Elements Position

The position property specifies the type of positioning method used for an element (static on the page, relative to his parent, etc..)

It can be:

* static: Default value. Elements render in order, as they appear in the document flow
* relative: The element is positioned relative to its normal position; you can add left, right, top or bottom arguments to add a gap by the element.
* fixed: Always stays in the same place even if the page is scrolled, left, right, top and bottom are used to position the element
* absolute: The element is positioned relative to its first positioned (not static) ancestor element
* sticky: The element is positioned based on the user’s scroll position

[More on CSS Position](https://www.w3schools.com/css/css_positioning.asp)  
[CSS Position Tricks](https://css-tricks.com/almanac/properties/p/position/)

#### Exercise 1

On the same HTML file (as the one in the CSS lesson)

Recreate this image with the **position** property

The elements are : a parent box (grey), a blue box and an orange box



### Dealing With Overflow

The overflow property specifies whether to clip the content or add scrollbars when an element’s content is too big to fit in the specified area.

It can be:

* visible - Default. The overflow is not clipped. The content renders outside the element’s box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

[Look at the demonstration Here](https://www.w3schools.com/cssref/playit.asp?filename=playcss_overflow)

### CSS Pseudo Class Selectors

A pseudo-class is used to style an element when he is in a special state (for example, when the user hovers over it)

The syntax is : element:pseudo-class {...}

#### Hover

hover means “mouse is over it”, for example, to change the color of a <div> when the mouse goes over it:

div:hover {

background-color: blue;

}

#### Children

You can target elements only if they are the nth child of their parent. For example, to style <p> tags that are the first child of another element, use:

p:first-child{

color: blue;

}

The same thing works with last-child or nth-child(n)

#### Not

The not pseudo-class allows you to exclude elements; for example, select all the divs that are not of the ignore class.

div:not(.ignore){

background-color: yellow;

}

For more, see [this](https://www.w3schools.com/css/css_pseudo_classes.asp).

## Media Queries

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#Useful_Resources)
    - [Media queries](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#Media_queries)
    - [CheatSheet](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#CheatSheet)
    - [Example on Media Queries in CSS](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#Example_on_Media_Queries_in_CSS)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/15/chapter/88#Feedback)

Last updated : April, 2nd

## What You Will Learn

* Media Queries : what are they ? how do we use them ?

## Useful Resources

* [Tutorial on Media Queries](https://www.sitepoint.com/creating-media-queries-for-responsive-web-designs/)
* [CSS3 Media Queries Template](https://gist.github.com/marcobarbosa/798569)

### Media Queries

Media queries are a feature of CSS that helps us build responsive websites. The key to designing and building responsive layouts is having your content respond to the device’s size (or “medium”).  
Your CSS can query the “media” (plural of “medium”) to find out their sizes and proportions.

When defining a media query, we use the @media keyword followed by conditional statements that will trigger the media query to apply or not.

Then you have to specify the media type and the size of the device, for example:

| **Device “viewport”** | **Explanation** |
| --- | --- |
| max-width | CSS will not apply on a screen width wider than specified. |
| min-width | CSS will not apply on a screen width narrower than specified. |
| max-height | CSS will not apply on a screen height taller than specified. |
| min-height | CSS will not apply on a screen height shorter than specified |

@media screen and (max-width: 600px) {

/\* apply for screens where the maximum width is 600px

}

@media screen and (max-width: 600px) {

h1 {

color: blue

}

}

You can add several size conditions that specify the width range.  
Here, the CSS modifications are implemented when the screen size is between 1200 and 1600px.

@media screen

and (min-width: 1200px)

and (max-width: 1600px) {

}

### CheatSheet

/\*------------------------------------------

Responsive Grid Media Queries - 1280, 1024, 768, 480

1280-1024 - desktop (default grid)

1024-768 - tablet landscape

768-480 - tablet

480-less - phone landscape & smaller

--------------------------------------------\*/

@media all and (min-width: 1024px) and (max-width: 1280px) { }

@media all and (min-width: 768px) and (max-width: 1024px) { }

@media all and (min-width: 480px) and (max-width: 768px) { }

@media all and (max-width: 480px) { }

### Example On Media Queries In CSS

**HTML file**

<!DOCTYPE html>

<html>

<head>

<title></title>

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

</head>

<body>

<h1>Media Queries</h1>

<div id="div1"></div>

<div id="div2"></div>

<div id="div3"></div>

<div id="div4"></div>

</body>

</html>

**CSS File**

div {

height: 100px;

}

#div1 {

background-color: red;

}

#div2 {

background-color: blue;

}

#div3 {

background-color: green;

}

#div4 {

background-color: purple;

}

//The logical keywords "not" or "only" can be used optionally to include or exclude specific media types or screen sizes

@media only screen and (min-width: 600px) {

div {

width: 100%;

}

}

@media only screen and (min-width: 900px) {

div {

width: 50%;

float: left;

}

}

@media only screen and (min-width: 1200px) {

div {

width: 25%;

float: left;

}

}

## CSS Flexbox

**Done**

### Table Of Contents

* + [What you will learn:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#What_you_will_learn:)
    - [Useful Resources:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Useful_Resources:)
      * [Games to practice flexbox:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Games_to_practice_flexbox:)
    - [What is Flexbox?](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#What_is_Flexbox?)
      * [Flexbox structure](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Flexbox_structure)
      * [Follow this demonstration as you read the course](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Follow_this_demonstration_as_you_read_the_course)
    - [Flexbox properties on the Parent Container](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Flexbox_properties_on_the_Parent_Container)
      * [1. display: flex](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#1._display:_flex)
      * [2. Flex Direction](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#2._Flex_Direction)
      * [3. Justify Content](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#3._Justify_Content)
      * [4. Align Items](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#4._Align_Items)
      * [5. Flex wrap](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#5._Flex_wrap)
      * [6. Align Content](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#6._Align_Content)
    - [Flexbox properties on the Flex Item](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Flexbox_properties_on_the_Flex_Item)
      * [1. Align Self](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#1._Align_Self)
      * [2. Order](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#2._Order)
      * [3. Flex](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#3._Flex)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/62#Feedback)

Last updated : April, 3rd

## What You Will Learn:

* Intro to concepts of responsive web design
* Flexbox

### Useful Resources:

* [Google Fonts](https://fonts.google.com/)
* [Free images](http://unsplash.com/)
* [Icons from Font Awesome:](https://fontawesome.com/icons?d=gallery)
* [Written Tutorial Flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)
* [Grid and Flexbox](https://medium.com/better-programming/build-a-responsive-modern-dashboard-layout-with-css-grid-and-flexbox-bd343776a97e)
* [Flexbox Cheatsheet](https://yoksel.github.io/flex-cheatsheet/)
* [Another Flexbox CheatSheet](https://darekkay.com/dev/flexbox-cheatsheet.html)

#### Games To Practice Flexbox:

* [flexbox froggy](http://flexboxfroggy.com/)
* [flexbox defense](http://www.flexboxdefense.com/)

### What Is Flexbox?

The “Flexbox” layout mode offers an alternative to float for defining the overall appearance of a web page. Whereas floats only let us horizontally position our boxes, flexbox gives us complete control over the alignment, direction, order, and size of our boxes.



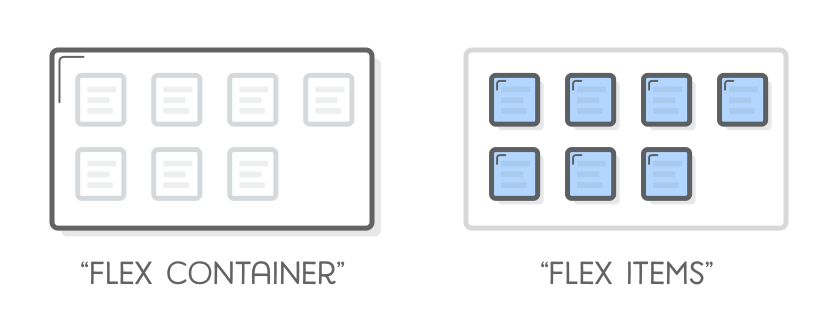
Flexbox is used to lay out the web pages, reserving floats for when you need text to flow around a box.



#### Flexbox Structure

Flexbox uses two types of boxes :

* “flex containers” : to group a bunch of flex items and define how they’re positioned
* “flex items”



#### Follow This Demonstration As You Read The Course

[**Flexbox Codepen Here**](https://codepen.io/avnermaman/pen/xxGWQNK?editors=1101)

### Flexbox Properties On The Parent Container

#### 1. Display: Flex

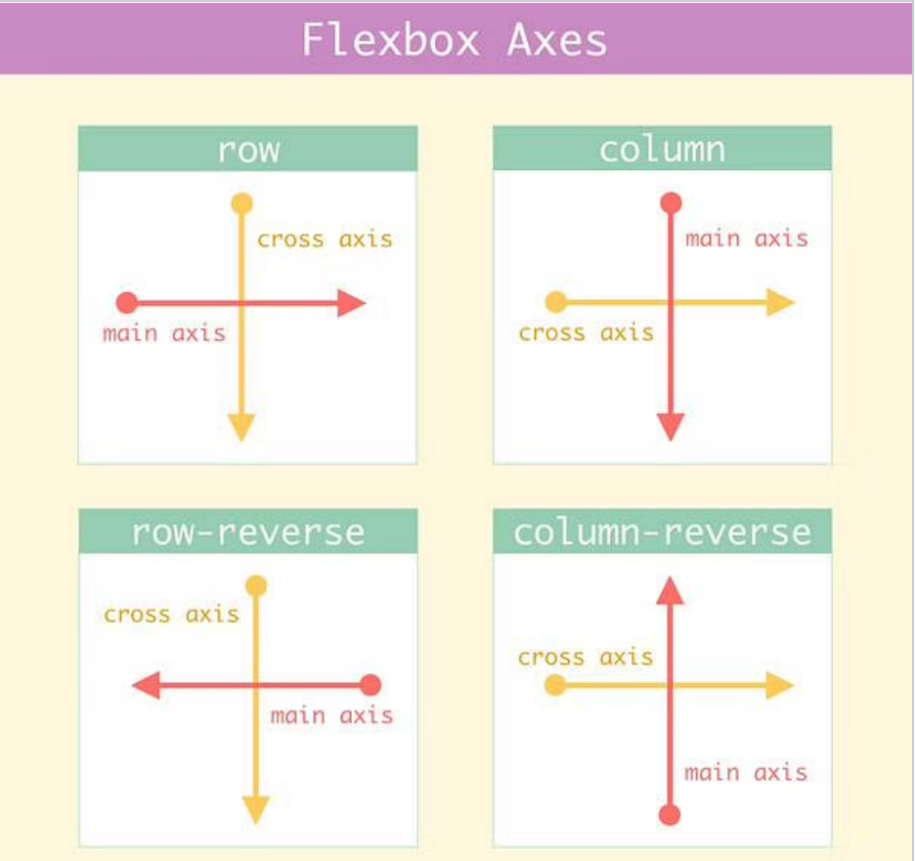
To do a Flexbox layout, you have to start by styling the parent container with display: flex

#### 2. Flex Direction

flex-direction allows you to control how flex items are placed in the flex container.

**It establishes the main-axis, thus defining the direction flex items are placed in the flex container.**

* row (default)
* row-reverse
* column
* column-reverse



#### 3. Justify Content

justify-content is a property to align items in the container **along the main axis**.

It defines how the browser distributes space between and around content items along the main-axis of a flex container.

* If the flex-direction is row, then the **main-axis** is x –> therefore, justify-content will align the items horizontally
* If the flex-direction is column, then the **main-axis** is y –> therefore, justify-content will align the items vertically

The values taken by the property justify-content are:

* flex-start (default)
* flex-end
* center
* space-around
* space-between

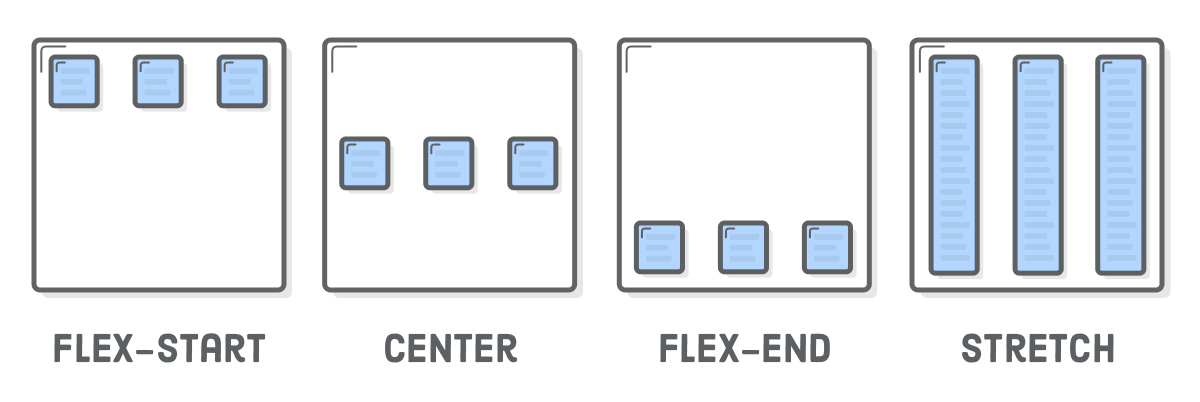
#### 4. Align Items

align-items is a property to align items in the container **along the cross axis**.

* If the flex-direction is row, then the **cross-axis** is y –> therefore, align-items will align the items vertically
* If the flex-direction is column, then the **cross-axis** is x –> therefore, align-items will align the items horizontally

The values taken by the property align-items are:

* stretch (default)
* baseline
* center
* flex-start
* flex-end

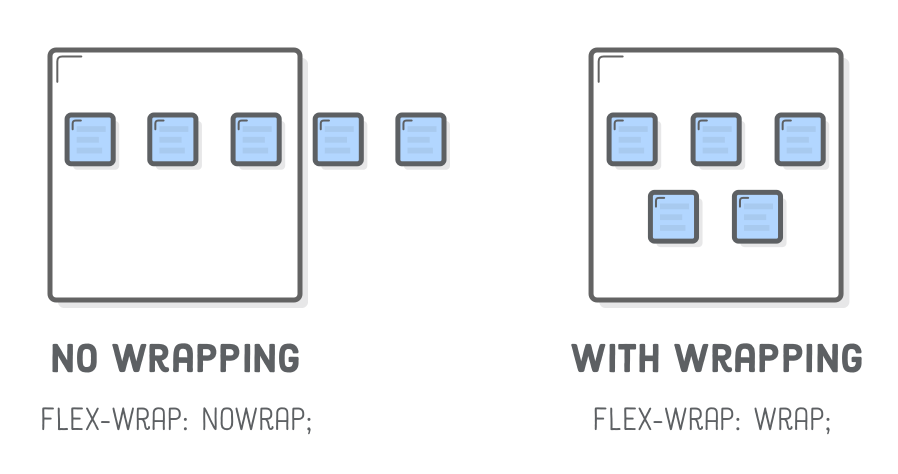


#### 5. Flex Wrap

flex-wrap is a property to fit all elements into one row.

It only applies if the flex items’ combined width is greater than its container’s width.

* nowrap (default)
* wrap
* wrap-reverse

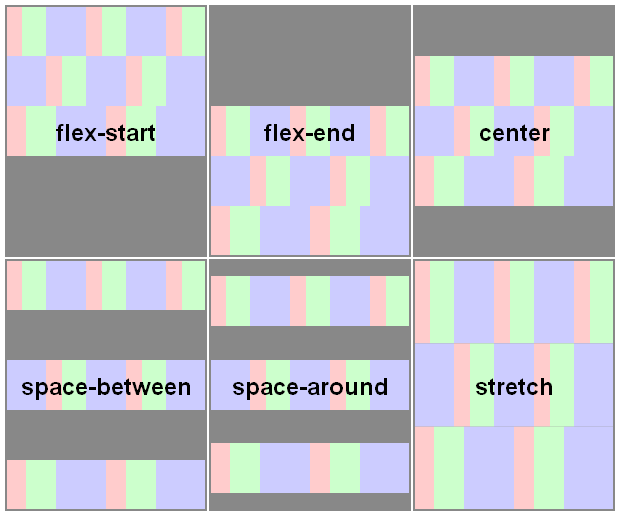


#### 6. Align Content

align-content is a property for aligning items with multiple lines. It is for aligning on the cross axis and will not affect one line’s content.

It only applies if there is more than one row of items.

* stretch (default)
* center
* flex-start
* flex-end
* space-around
* space-between



### Flexbox Properties On The Flex Item

#### 1. Align Self

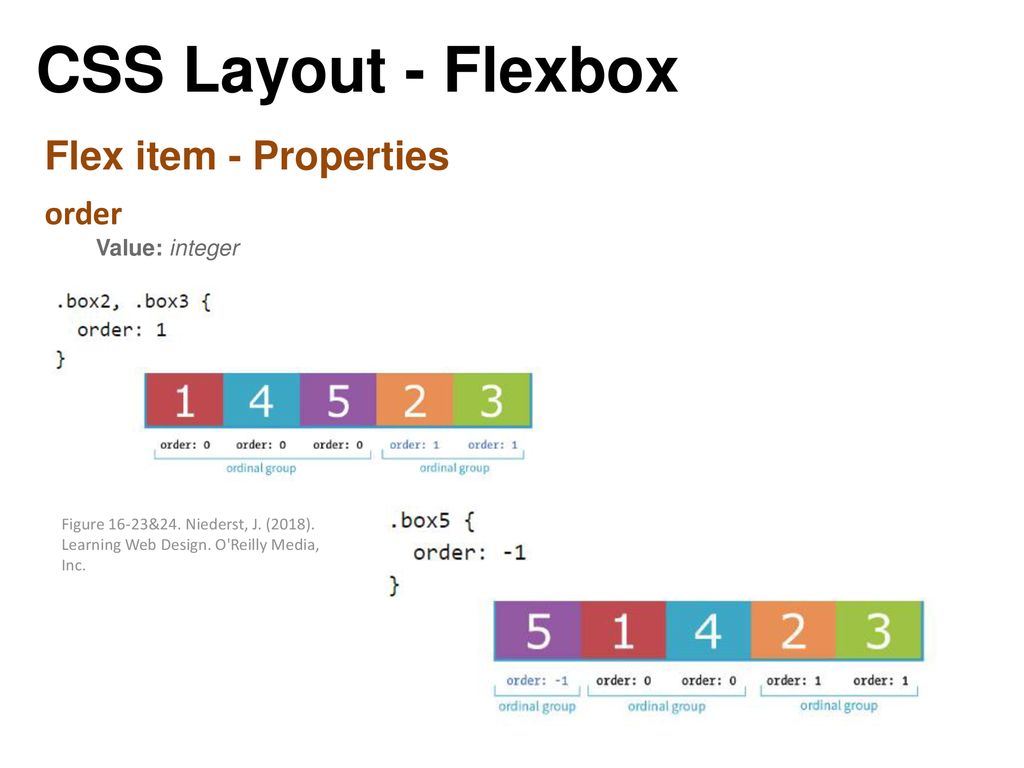
align-self is a property for aligning an item individually. It overrides the align-items value from its container.

* stretch (default)
* center
* flex-start
* flex-end
* space-around
* space-between

#### 2. Order

order property to a flex item defines its order in the container without affecting surrounding items.

Its default value is 0, and increasing or decreasing it from there moves the item to the right or left, respectively.



#### 3. Flex

flex is a property that defines the width of individual items in a flex container.

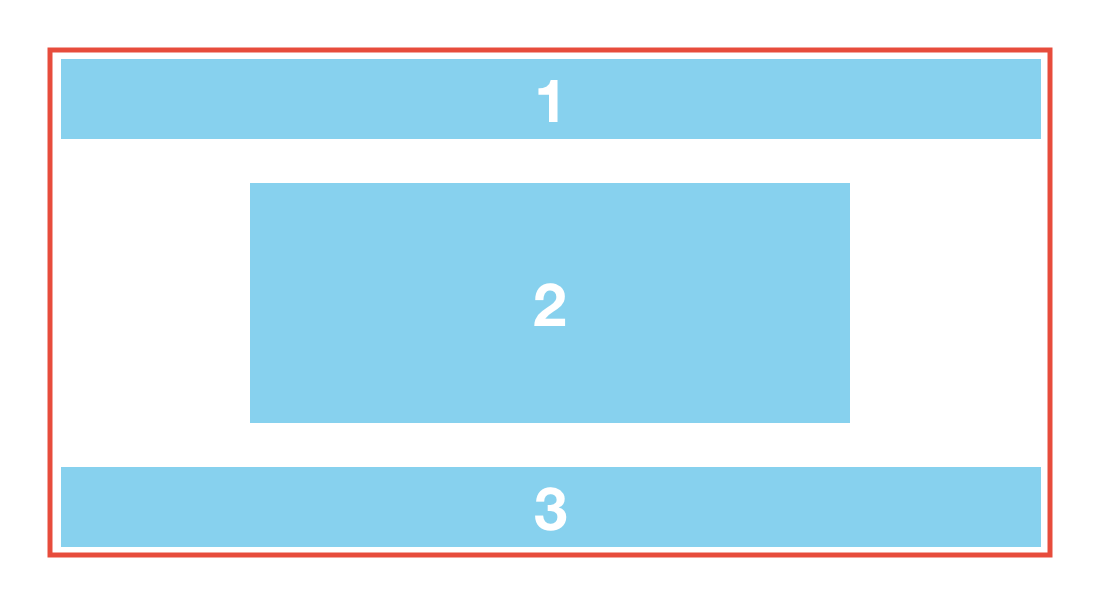
* It allows them to have **flexible widths**.
* It tells the flex container how to distribute extra space to each item. For example, an item with a flex value of 2 will grow twice as fast as items with the default value of 1.

[Video on the property flex](https://www.youtube.com/watch?v=DzAacFWSx0c)

[Video on the properties flex-grow, flex-shrink, flex-basis](https://www.youtube.com/watch?v=CFgeJq4l1YM)

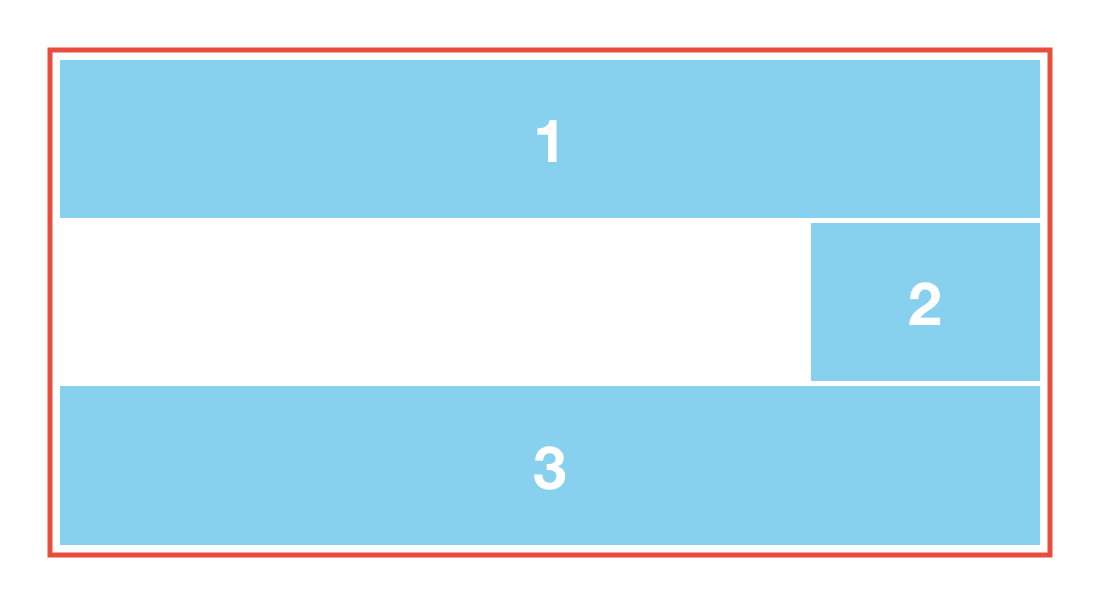
## Exercise 1 : Flexbox

Recreate this structure with Flexbox



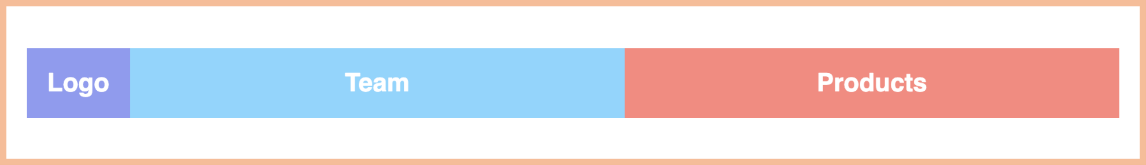
## Exercise 2 : Flexbox

Recreate this structure with Flexbox



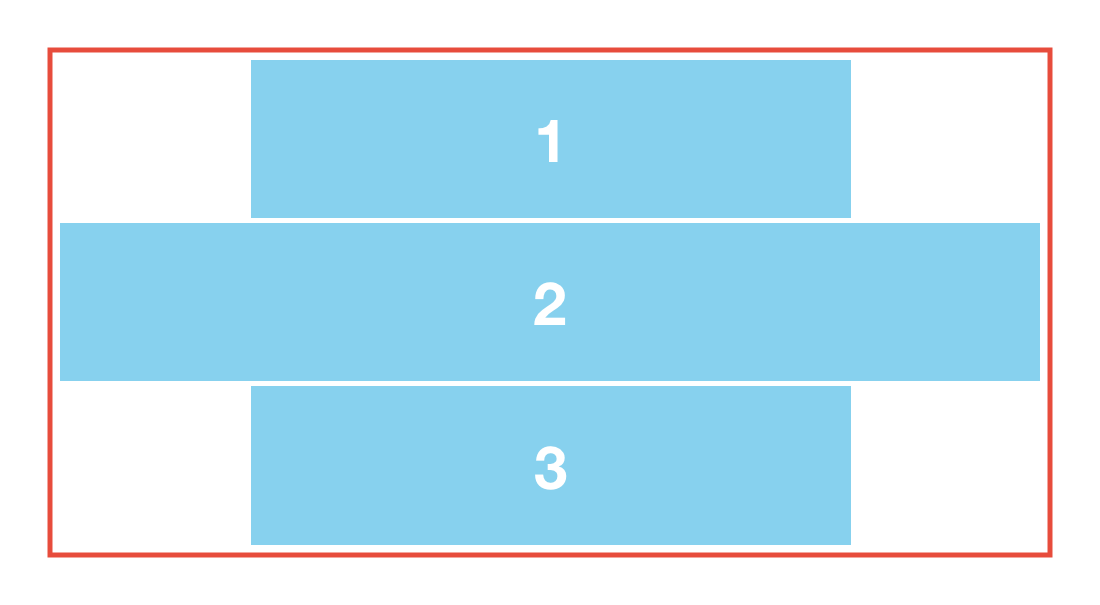
## Exercise 3 : Flexbox

Recreate this structure with Flexbox



## Exercise 4 : Flexbox

Recreate this structure with Flexbox



## CSS Grid

**Not Completed**

### Table Of Contents

* + [What you will learn:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#What_you_will_learn:)
    - * [Useful Resources:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Useful_Resources:)
      * [Games to practice grid:](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Games_to_practice_grid:)
    - [Advanced CSS](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Advanced_CSS)
    - [What is Grid?](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#What_is_Grid?)
    - [Follow this demonstration as you read the course](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Follow_this_demonstration_as_you_read_the_course)
    - [Grid properties on the Parent Container](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Grid_properties_on_the_Parent_Container)
      * [1. Grid-template-rows](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#1._Grid-template-rows)
      * [2. Grid-template-columns](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#2._Grid-template-columns)
      * [3. minmax](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#3._minmax)
      * [4. repeat](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#4._repeat)
      * [5. Grid gap](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#5._Grid_gap)
    - [Grid properties on the grid Item](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Grid_properties_on_the_grid_Item)
      * [1. Grid row, Grid column](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#1._Grid_row,_Grid_column)
    - [Grid Area](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Grid_Area)
    - [Grid Alignment](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Grid_Alignment)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#Feedback)

Last updated : April, 3rd

## What You Will Learn:

* Intro to concepts of responsive web design
* CSS grids

#### Useful Resources:

* [Google Fonts](https://fonts.google.com/)
* [Free images](http://unsplash.com/)
* [Icons from Font Awesome:](https://fontawesome.com/icons?d=gallery)
* [Tutorial on Media Queries](https://www.sitepoint.com/creating-media-queries-for-responsive-web-designs/)
* [Written Tutorial Grid](https://css-tricks.com/snippets/css/complete-guide-grid/)
* [Grid for beginners](https://medium.com/@elad/css-grid-for-beginners-ee649080529b)
* [Grid](http://grid.malven.co/)
* [Free Video](https://cssgrid.io/)
* [PDF File about CSS GRID: Good Explanation](https://github.com/devtlv/studentsGitHub/files/5843771/css-grid.pdf)

#### Games To Practice Grid:

* [Grid Garden](https://cssgridgarden.com/)

### Advanced CSS

Advanced CSS can add much more functionality and versatility to a webpage. There are various powerful tools to help make your website layout dynamic, giving you infinite options for how you wish to display your content. Mastery of these tools can remove the heavy lifting of playing around with sizes to fit everything correctly. We learned how to make a website responsive so that it will show up differently on mobile devices.

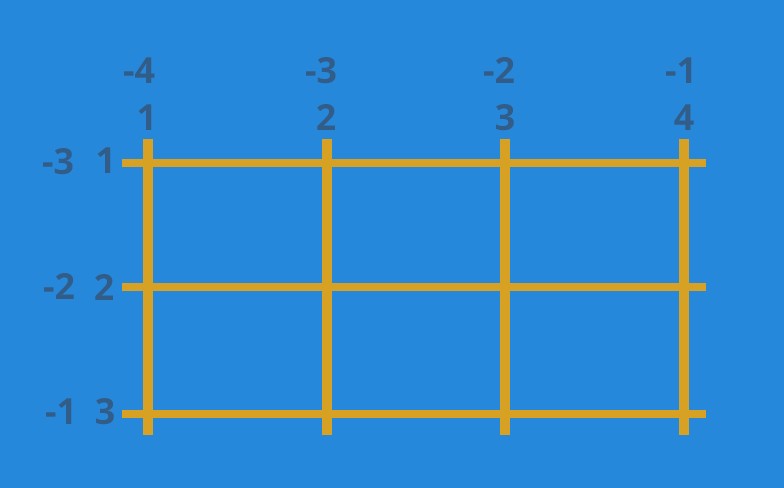
### What Is Grid?

CSS Grid is a two-dimensional layout system created specifically to tackle grid-based user interfaces on the web. Grids are fundamental to the design of layouts.

CSS Grid introduces a series of properties that allow us to create grid structures and control the placement and sizing of grid items using CSS

The grid uses the same pattern of Flexbox with a parent container and its child elements. In Grid there is the **grid container** and its child elements the **grid items**

Gridlines are the horizontal and vertical lines that form the basis of the grid structure. Gridlines also have negative indices, which allow us to reference grid lines starting from the grid’s end.



### Follow This Demonstration As You Read The Course

[**Grid Codepen Here**](https://codepen.io/avnermaman/pen/abOKoPV?editors=1101)

### Grid Properties On The Parent Container

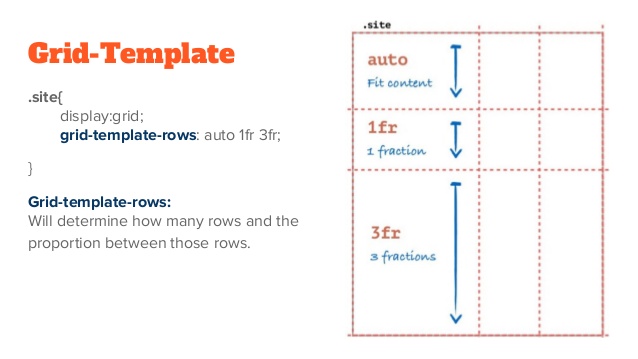
To do a grid layout you have to start by styling the parent container with display:grid

#### 1. Grid-Template-Rows

grid-template-rows: Xpx Ypx

A row track is created for each value specified for grid-template-rows. Track size values can be any non-negative length value (px, %, em, etc.)

Items 1 and 2 have fixed heights of Xpx and Ypx.



#### 2. Grid-Template-Columns

grid-template-columns: Xpx Ypx.

#### 3. Minmax

grid-template-rows: minmax(100px, auto);

grid-template-columns: minmax(auto, 50%) 1fr 3em;

The minmax() function accepts two arguments: the first is the minimum size of the block and the second is the maximum size. The values can also be auto, which allows the track to grow/stretch based on the content’s size.

Here, the first-row track is set to have a minimum height of 100px, but its maximum size of auto will allow the row track to grow it the content is taller than 100px.

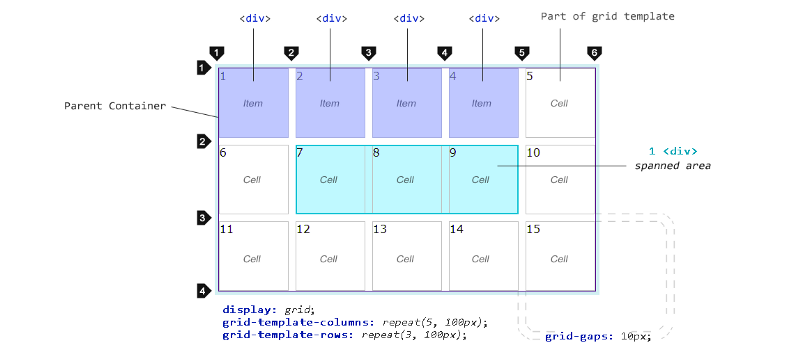
The first column track has a minimum size of auto, but its maximum size of 50% will prevent it from getting no wider than 50% of the grid container width.

#### 4. Repeat

grid-template-rows: repeat(4, 100px);

grid-template-columns: repeat(3, 1fr);

The repeat() notation accepts two arguments: the first represents the number of times the defined tracks should repeat, and the second is the block definition.



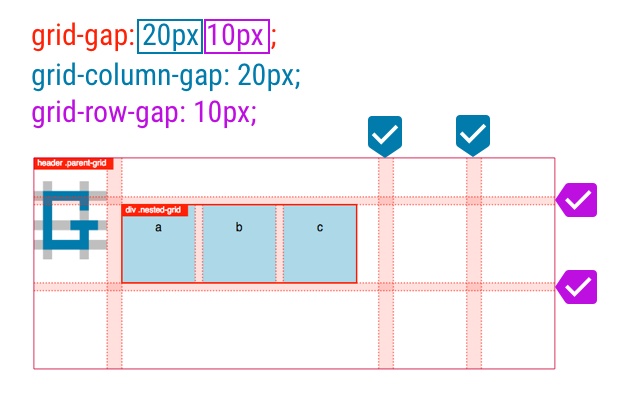
grid-template-columns: 30px repeat(3, 1fr) 30px

repeat() can also be used within block listings.

In this example, the first and last column tracks have widths of 30px, and the 3 column tracks in between, created by repeat(), have widths of 1fr each.

#### 5. Grid Gap

The grid-column-gap and grid-row-gap properties create spaces between columns and rows.



### Grid Properties On The Grid Item

#### 1. Grid Row, Grid Column

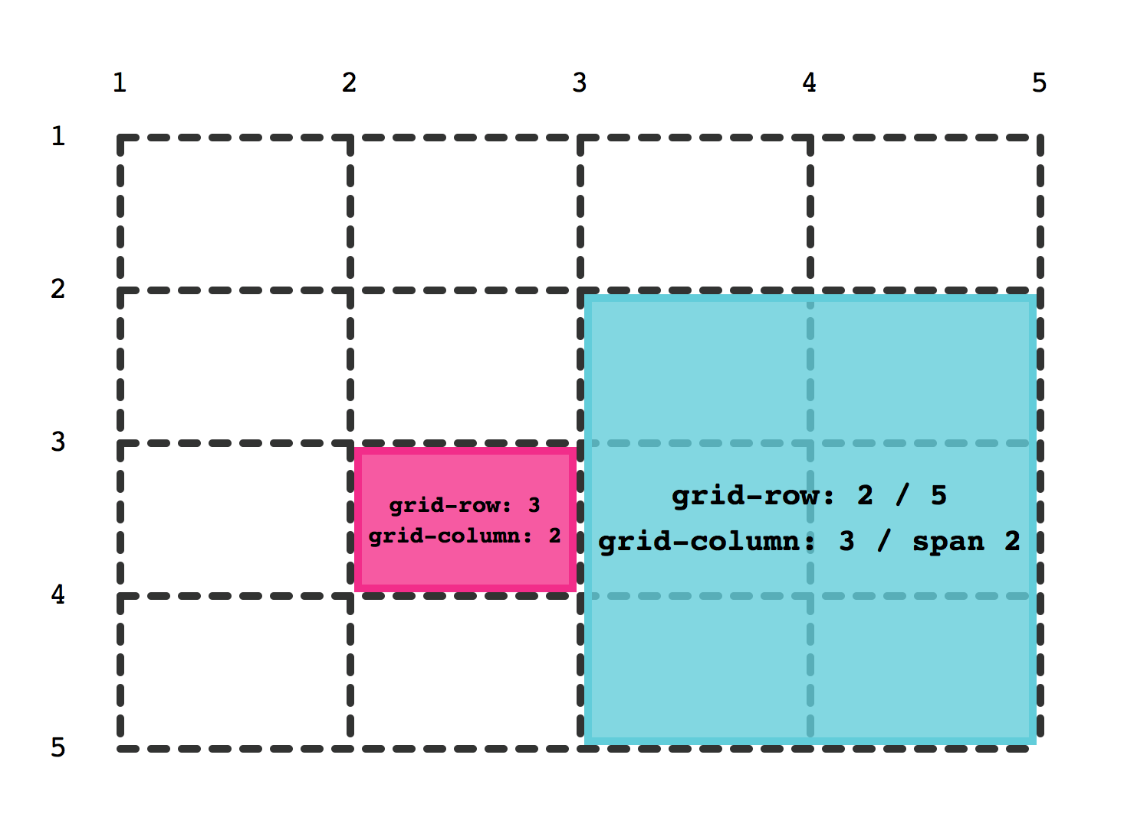
grid-row: 2 / span 3;

grid-column: span 2;

The keyword span, followed by the # of columns or rows to span, can also be used.

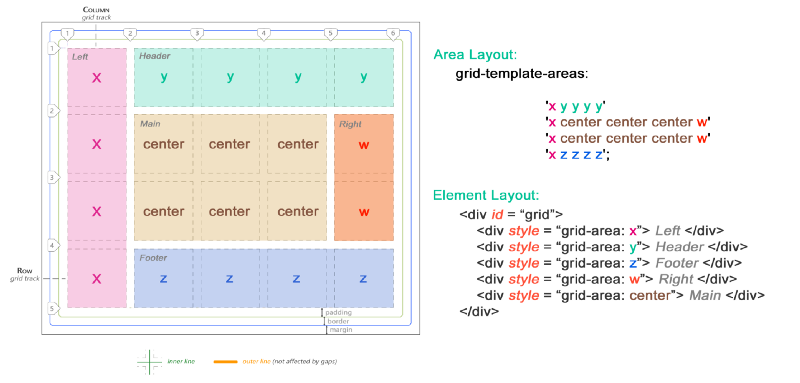
This code means :

* Height : the blocks starts at the 2nd row and take a height of 3 rows
* Width : the blocks starts at the 1st column and and take a width of 2 columns



### Grid Area

#### [Everything Is Explained Here](https://alligator.io/css/css-grid-layout-grid-areas/)



### Grid Alignment

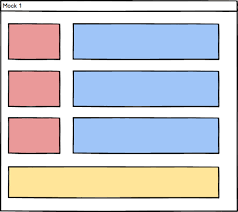
There are 6 properties :

* justify-items : on the parent container, move every block along the **row axis**
* align-items : on the parent container, move every block along the **column axis**
* justify-content : on the parent container, move the entire grid along the **row axis**
* align-content: on the parent container, move the entire grid along the **column axis**
* justify-self : on the child container, move the specific block along the **row axis**
* align-self: on the child container, move the specific block along the **column axis**

#### [Everything Is Explained Here](https://alligator.io/css/align-justify/)

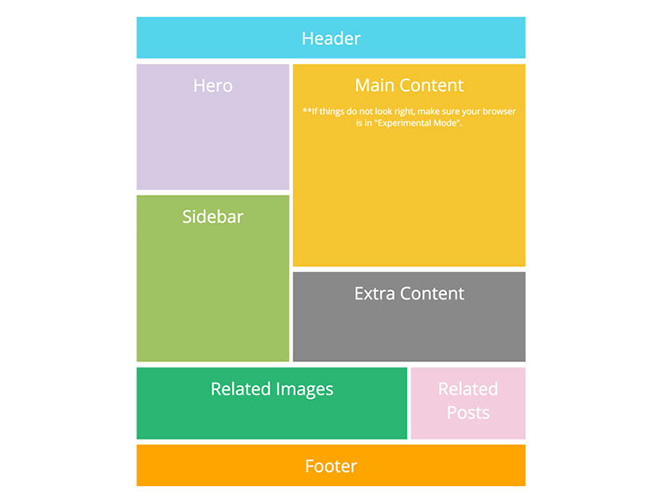
## Exercise 1 : Grid

Recreate this structure with Grid



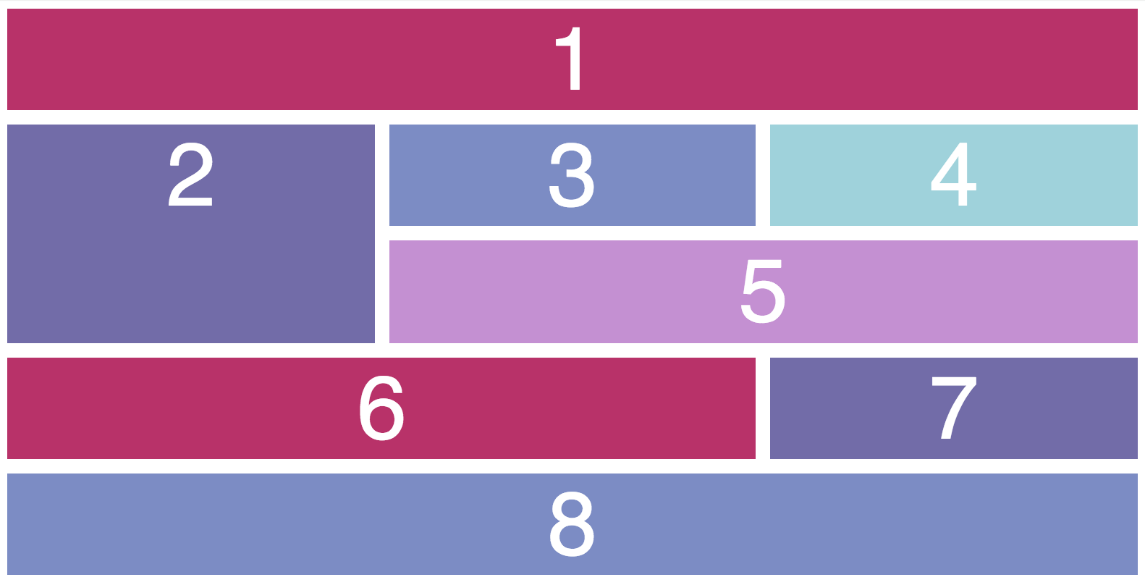
## Exercise 2 : Grid

Recreate this structure with Grid



## Exercise 3 : Grid

Recreate this structure with Grid



[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/9/section/17/chapter/63#breadcrumbs)

**How To Add A Color Picker In HTML**

**Done**

**Color Picker**

**Color Pickers** in HTML allow the user to select and input a color.

Once submitted, the browser converts the input color into a string that represents it.

**Usage**

To add a color picker in an HTML page, use an <input> tag with type = 'color'.

The initial value can be set using the value property. This value needs to be set in hexadecimal because colors are represented as six-digit hexadecimal values following a hashtag (#). The first two digits represent the red component, followed by two for the green component, and then the last two for the blue component. These components represent a full RGB color. Following this format, #ff0000 is red, #00ff00 is green, #0000ff is blue, and ##ffffff is white. These three colors can be mixed to get more colors. For instance, #bdaa55 gives us a brownish shade.

*The default color will be black or #000000 if no initial value is entered.*

<html>

<head>

</head>

<body>

<label for="colorpicker">Color Picker:</label>

<input type="color" id="colorpicker" value="#0000ff">

</body>

</html>

This example demonstrates the usage of the <input type = 'color'> tag. The initial value is set as #0000ff, i.e., blue.

## Bootstrap Basics

**Not Completed**

### Table Of Contents

* + [What you will Learn:](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#What_you_will_Learn:)
    - [Useful Resources:](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#Useful_Resources:)
    - [Bootstrap](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#Bootstrap)
      * [Setup](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#Setup)
      * [How bootstrap work](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#How_bootstrap_work)
      * [Bootstrap grid](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#Bootstrap_grid)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/873#Feedback)

Last updated : April, 3rd

## What You Will Learn:

* What Bootstrap is
* How to use Bootstrap to make your site easier to create and to make responsive
* Understand how to include Bootstrap’s CSS and JavaScript properly and begin customizing

### Useful Resources:

Bootstrap Templates:

* [Templates Link 1](https://mdbootstrap.com/freebies/)
* [Templates Link 2](http://mashup-template.com/templates.html)
* [Templates Link 3](https://startbootstrap.com/template-categories/all/)
* [Creative Tim Templates](https://www.creative-tim.com/bootstrap-themes/ui-kit?direction=asc&sort=price)

### Bootstrap

Bootstrap is a front-end framework that you can download a lot of work that has already been done for you. This can be very helpful, especially for making responsive websites. It has plenty of pre-built features like navbars, forms, buttons, etc., that you can pick and choose to add to your website by just copying and pasting the code, then customizing the font, etc. as you wish. Bootstrap is one of the most popular front-end frameworks for designing websites because it’s easy for anyone with basic HTML/CSS and JavaScript knowledge to get started. Bootstrap’s mobile-first and responsive design makes sure it’s functional across devices of all sizes and it’s compatible across all modern browsers.

(Be careful when running JS scripts online. You don’t know what’s in the code you’re porting from someone else, and it may have backdoors or other harmful code.)

#### Setup

Bootstrap consists mainly of style sheets and script. As such, they can be loaded in your HTML like any other style sheet.

**Using a CDN**

CDN stands for Content Delivery Network.

A CDN server holds resources such as images, videos, audio clips, and CSS and JavaScript files.

With a CDN, you don’t need to download the CSS/Javascript framework to add to your code; just use the remote resources.

For example, to get Bootstrap into your page, use the <link rel="stylesheet"> familiar syntax, but in the href, insert the CDN address.

You can copy the Starter Template of Bootstrap (You will get all the necessary links from there)

[**Starter Template of Bootstrap**](https://getbootstrap.com/docs/5.0/getting-started/introduction/#starter-template)

#### How Bootstrap Work

Bootstrap gives you a lot of pre-made CSS classes for almost all the HTML elements.

For example, the alert bootstrap class will style your element in an alert style.

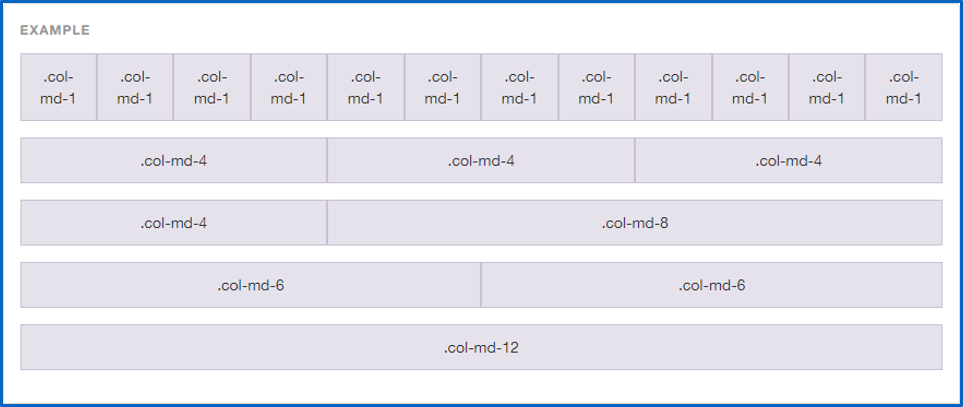
You can add additional styles (see bootstrap documentation for more info) by adding another class.

For example, alert alert-primary will style your element in a primary alert style.

#### Bootstrap Grid

One of the most powerful features of Bootstrap is its grid system.

A bootstrap grid system overview:

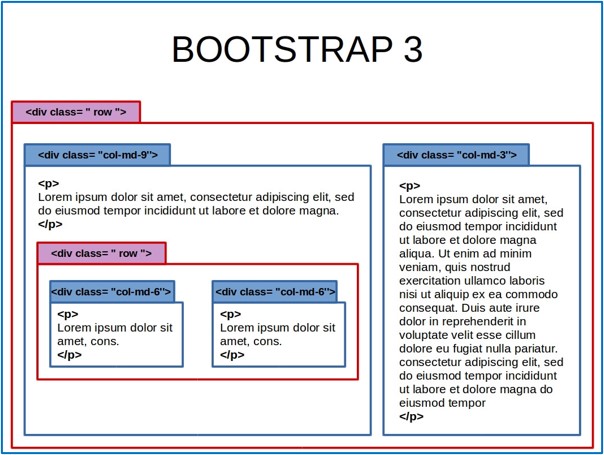


[**Bootstrap Responsive Layout 1 - CodePen Here**](https://codepen.io/avnermaman/pen/GRJxKBK)

The grid system is built with flexbox and allows you to organize your HTML file with rows and columns.

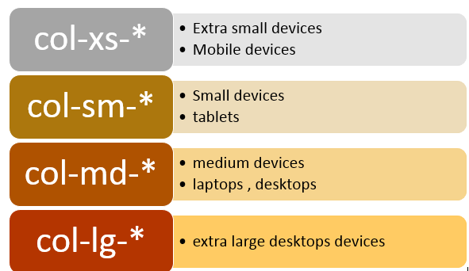
How it works:

* First, your grid is defined under a container (meaning an element with the container class)
* Then, you need to define each row in an element with row as class; each row contains 12 columns



* Inside each row you can define some columns in this way class="col-<responsiveness>-<size>". The size is the number of columns this element will take, and the responsiveness parameter defines which screen size will display the column. Here are the existing responsiveness modes:
  + Nothing: (Extra small devices - < 576px)
  + sm: (small devices - >576px)
  + md: (medium devices - > 768px)
  + lg: (large devices - > 992px)
  + xl: (xlarge devices - > 1200px)

For example, col-md-5 is an element that will take 5 columns of the row but will be displayed only if the screen width is greater than 768px.



[**Bootstrap Responsive Layout - Codepen Here**](https://codepen.io/avnermaman/pen/BaNrBPo)

[**Bootstrap Responsive Layout-Offset - Codepen Here**](https://codepen.io/avnermaman/pen/wvamwxM)

[**More on Bootstrap Layout**](https://getbootstrap.com/docs/5.0/layout/grid/)

## Exercise 1

Create a stuctured html file with Boostrap CDN.

!! Make sure to put a border in each of the block so you can see the way they work !!

1. Create a structure with a container, 2 rows and 3 columns in each row. What should be the width of each column so that the 3 columns will take the whole width of the row ?

2. Create a structure with a container, 2 rows and 2 columns in each row. Use *offset* to make a space between the 2 columns

3. Create a structure with 2 rows and 2 columns in each row. What happens if you don't have any container ?

4. Create a structure with a container, 2 columns. What happens if you don't have any row ?

5. Create a structure with a container, 1 row and 3 columns in the row. Make it responsive so depending of the width of the screen, the columns will be displayed differently. Write a code for 4 different type of screen.

## Bootstrap Components

**Not Completed**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#What_you_will_learn)
    - [Bootstrap alerts](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_alerts)
    - [Bootstrap icons](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_icons)
    - [Bootstrap forms](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_forms)
    - [Bootstrap buttons](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_buttons)
    - [Bootstrap dropdowns](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_dropdowns)
    - [Bootstrap collapses](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_collapses)
    - [Bootstrap navbar](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_navbar)
    - [Bootstrap carousels](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_carousels)
    - [Bootstrap modal box](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Bootstrap_modal_box)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/48#Feedback)

Last updated : April, 3rd

## What You Will Learn

\*What are Boostrap Components - and how to use them

### Bootstrap Alerts

Bootstrap provides a lot of CSS classes to display alerts (or any message to the user).

First, add the alert class to the element, and then decorate it with some other classes, like:

* alert-success
* alert-info
* alert-warning
* alert-danger

You can also add links in your alert to style them, add the alert-link class to their attributes.

Closing the alert can be done by adding a link or a button with the class="close" and the data-dismiss="alert" attributes. It would help if you also put alert-dismissible in the alert classes.

You can add animations to those actions. For example, an alert with the fade and the in class will fade in and out.

### Bootstrap Icons

Bootstrap provides a big set of icons, called glyphicons, to add a glyphicon, create a <span> element with the glyphicon class, and then add the icon’s class you want, like glyphicon-search or gliphycon-user.

For example:

<p>Alarm icon: <i class="bi bi-alarm"></i></p>

<p>Asterix icon: <i class="bi bi-asterisk"></i></p>

<p>Book icon: <i class="bi bi-book"></i></p>

Check the whole list [here](https://icons.getbootstrap.com/).

Look at this [**video about Boostrap Icons**](https://www.youtube.com/watch?v=DPnJldwv22o)

### Bootstrap Forms

Bootstrap provides three types of form layouts:

* Vertical, the default one
* Horizontal
* Inline

Check out the [**Documentation about Bootstrap Forms**](https://getbootstrap.com/docs/5.0/forms/layout/#forms)

### Bootstrap Buttons

Bootstrap also provides some button styles, add the btn class to your button (it works on <a>, <button> and <input> tag).

A list of the bootstrap buttons:

* btn-link
* btn-primary
* btn-success
* btn-info
* btn-warning
* btn-danger

You can add the classes btn-lg, btn-md, btn-sm, btn-xs to size your button, and the disabled to make it unclickable.

You can also make a group of buttons by putting all the buttons in a div with the btn-group or btn-group-vertical classes.

#### Exercise 2

On the same HTML file as above

1. Create a few alerts. Look on Bootstrap documentation to make 3 different alerts : blue, green red

2. Create a few buttons. Look on Bootstrap documentation to make 3 different buttons : blue, green red

3. Change the width of the alerts and of the buttons

4. Make the 3 buttons next to each other horizontally

5. Create a form that has 4 inputs : name, surname, phone number and email. The button of the form should be an icon of your choice

6. Make the form responsive depending on the width of the screen

### Bootstrap Dropdowns

**Don’t forget to add the script tag at the end of the body to create the dropdown**

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

Look at this [**article on Boostrap5 Dropdowns**](https://javascript.plainenglish.io/bootstrap-5-dropdowns-f7f85146d126)

Look at this [**tutorial on Boostrap5 Dropdowns**](https://www.geeksforgeeks.org/bootstrap-5-dropdowns/)

A dropdown menu is a toggleable menu that displays a predefined list of values.

To create a dropdown with bootstrap, first create a div with class="dropdown".

Then a toggle button with a class of dropdown-toggle and the data-toggle="dropdown".

Finally, add the predefined list in a <ul> element, add the dropdown-menu class to it to bind it.

Now you can add all your toggleable elements to the list in <li> elements.

You can add a <li><hr class="dropdown-divider"></li> to create a thin horizontal border in your dropdown menu.

<div class="dropdown">

<button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">

Dropdown button

</button>

<ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">

<li><a class="dropdown-item" href="#">Action</a></li>

<li><a class="dropdown-item" href="#">Another action</a></li>

<li><hr class="dropdown-divider"></li>

<li><a class="dropdown-item" href="#">Something else here</a></li>

</ul>

</div>

### Bootstrap Collapses

**Don’t forget to add the script tag at the end of the body to create the collapse**

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

A collapsible element is an element that can be hidden and showed with a button.

To do so, just add the class="collapse" attribute to an element, and data-bs-toggle="collapse" data-bs-target="<target\_id>" to the button that controls the collapse.

For example:

<button class="btn btn-primary" type="button" data-bs-toggle="collapse" data-bs-target="#collapseExample" aria-expanded="false" aria-controls="collapseExample">

Button with data-bs-target

</button>

<div class="collapse" id="collapseExample">

<div class="card card-body">

Some placeholder content for the collapse component. This panel is hidden by default but revealed when the user activates the relevant trigger.

</div>

</div>

### Bootstrap Navbar

**Look at**[**these navbar examples**](https://getbootstrap.com/docs/5.0/components/navbar/)

To create a navbar, add the navbar class to your <nav> or <div> navbar element.

**Look at**[**these examples to position your navbar**](https://getbootstrap.com/docs/5.0/components/navbar/#placement)

Navbar direct children (generally <ul> elements) should have nav and navbar-nav classes.

For example:

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="collapse navbar-collapse" id="navbarText">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="#">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Features</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Pricing</a>

</li>

</ul>

</div>

</nav>

You can also put this navbar into a container or a fluid-container

Finally, if your navbar is too big, you can add the collapse navbar-collapse classes to a div of the navbar and control it with a button:

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

#### Exercise 3

On the same HTML file as above

1. Create a navbar with 2 links, a form, and a dropdownlist

2. Make the navbar fixed to the top

3. Add an icon on the left on the navbar

### Bootstrap Carousels

Carousels are another remarkable feature of bootstrap.

**Don’t forget to add the script tag at the end of the body to create the carousel**

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

**See some examples :**[**Try it here**](https://mdbootstrap.com/docs/standard/components/carousel/)

Carousels are another awesome feature of bootstrap.

To build a carousel:

* First, add an outer div, with an id of your choice, the carousel and the slide classes, and the data-bs-ride="carousel" attribute. Add all the carousel content in this div.
* The slides are in a specific div with class carousel-inner, each slide should be in another div with class carousel-item, it can be text or images. You need to add active class to one of the slide, to set the carousel on this slide by default.  
  The indicators are some little dots at the bottom, which indicates how many slides there are and which one the user is currently viewing. Specify them in an ordered list with class carousel-indicators. Each indicator should have a data-bs-target attribute (the id of the carousel div) and a data-bs-slide-to attribute that specifies which slide to go to when clicking on the dot.
* The left and right controls are buttons that allows the user to slide back and forth, just add them in <a> elements, set left/right and carousel-control as classes, data-bs-slide="prev"/data-bs-slide="next" as attribute, and the id of the carousel div as value for the href attribute.

[**Carousel Example Codepen Here**](https://codepen.io/avnermaman/pen/ZEGqJjr?editors=1100)

The code :

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

<title>Hello, world!</title>

</head>

<body>

<!-- Carousel -->

<div id="carouselExampleIndicators" class="carousel slide demo" data-bs-ride="carousel">

<div class="carousel-indicators">

<button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>

<button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide 2"></button>

<button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2" aria-label="Slide 3"></button>

</div>

<div class="carousel-inner">

<div class="carousel-item active">

<img src="https://www.history.com/.image/ar\_16:9%2Cc\_fill%2Ccs\_srgb%2Cfl\_progressive%2Cg\_faces:center%2Cq\_auto:good%2Cw\_768/MTYyMzAxNjY4MjA4MTU4NjIx/topic-los-angeles-gettyimages-943489042-feature.jpg" class="d-block w-100" alt="Los Angeles">

<div class="carousel-caption d-none d-md-block">

<h3>Los Angeles</h3>

<p>LA is always so much fun!</p>

</div>

</div>

<div class="carousel-item">

<img src="https://krcrtv.com/resources/media/e6198348-bf1d-4a97-91d8-c4b33fa817eb-large16x9\_1280x960\_80806B00ILGKJ.jpg?1542740203755" class="d-block w-100" alt="Chicago">

<div class="carousel-caption d-none d-md-block">

<h3>Chicago</h3>

<p>Thank you, Chicago!</p>

</div>

</div>

<div class="carousel-item">

<img src="https://cdn.wallpapersafari.com/78/18/l1RySF.jpg" class="d-block w-100" alt="New York">

<div class="carousel-caption d-none d-md-block">

<h3>New York</h3>

<p>We love the Big Apple!</p>

</div>

</div>

</div>

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Previous</span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Next</span>

</button>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

</body>

</html>

### Exercise 4

On the same HTML file as above

1. Create a carousel with your 3 best pics. Make it pretty !

### Bootstrap Modal Box

**Don’t forget to add the script tag at the end of the body to create the modal**

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

The modal box is a little popup window displayed on the top of the current page; it’s perfect for some explanation on something.

To build it:

* Create a modal box. This is the div that contains the modal window; create a div with the modal class and an id of your choice. You can add an animation class like fade. Inside this div, create another div with the class modal-dialog (this is to set the modal’s proper width and margin). You can add a size to this modal-dialog, by adding the modal-sm or modal-lg class.  
  **Insert the modal content**: Inside the previous div (the modal-dialog one), you need to create the modal window’s content. Add a div with the modal-content class, inside this div; you can create a modal-header, a modal-body, and a modal-footer div.
* Create a trigger. A button/link that display this modal box, to do so, add the data-bs-toggle="modal" and the data-bs-target="<modal\_div\_id>" in the button attributes.

For example:

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal">

Launch demo modal

</button>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Modal title</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

Modal Body

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary">Save changes</button>

</div>

</div>

</div>

</div>

## Github Branches

**Not Completed**

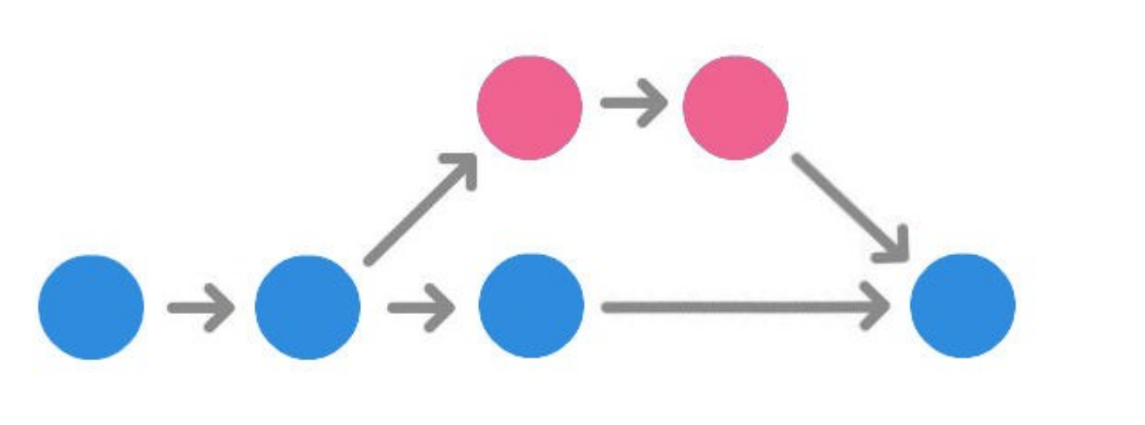
### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#What_you_will_learn)
    - [I. Some important commands when doing a big project](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#I._Some_important_commands_when_doing_a_big_project)
      * [Branches](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#Branches)
        + [1. Creating new branches - git branch](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#1._Creating_new_branches_-_git_branch)
        + [3. Merging branches - git merge](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#3._Merging_branches_-_git_merge)
    - [II. Some important commands when doing a big project in a group](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#II._Some_important_commands_when_doing_a_big_project__in_a_group)
      * [1. Merging branches with pull request](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#1._Merging_branches_with_pull_request)
      * [Steps to merge a branch into main](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#Steps_to_merge_a_branch_into_main)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/16/chapter/817#Feedback)

## What You Will Learn

* Understand the role of branches and how to use them

### I. Some Important Commands When Doing A Big Project



#### Branches

When developing a new feature, it is considered a good practice to work on a copy of the original project, called a branch. Branches have their own history and isolate their changes from one another, until you decide to merge them back together. This is done for a couple of reasons:

* An already working, stable version of the code won’t be broken.
* Many features can be safely developed at once by different people.
* Developers can work on their own branch, without the risk of their codebase changing due to someone else’s work.
* When unsure what’s best, multiple versions of the same feature can be developed on separate branches and then compared.

##### 1. Creating New Branches - Git Branch

The default branch of every repository is called **main**. To create additional branches use the git branch command:

git branch <name-of-the-branch>

This just creates the new branch, which at this point is exactly the same as our main.

##### 2. Switching Branches - Git Checkout

Now, when we run git branch, we will see there are two options available:

git branch

<name-of-the-branch>

\* main

Main is the current branch and is marked with an asterisk. However, we want to work on our new amazing features, so we need to switch to the other branch. This is done with the git checkout command, expecting one parameter - the branch to switch to.

git checkout <name-of-the-branch>

##### 3. Merging Branches - Git Merge

Our <name-of-the-branch> is going to be just another text file called <new-file>. We will create it, add it, and commit it.

git add <new-file>

git commit -m "New feature complete."

The new feature is complete, we can go back to the main branch.

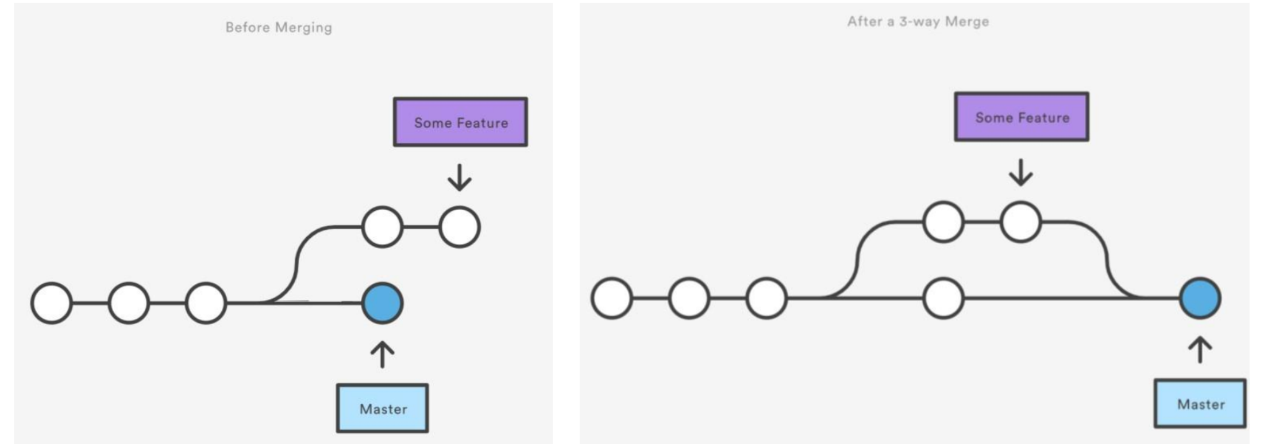
git checkout main

Now, if we open our project in the file browser, we’ll notice that <new-file> has disappeared. That’s because we are back in the main branch, and here <new-file> was never created. To bring it in, we need to git merge the two branches together, applying the changes done in <name-of-the-branch> to the main version of the project.

git merge <name-of-the-branch>

The main branch is now up to date. The <name-of-the-branch> branch is no longer needed and can be removed.

git branch -d <name-of-the-branch>



##### Exercise 1

1. Work now alone

2. In the initial repo (Exercise1), and from the main branch create a new branch called *weather*. Checkout to this new branch and add this sentence to *hello.txt*: *Hey it's sunny today!*. Than add, commit and push

3. Checkout to your main branch, has something changed in *hello.txt* ?

4. Checkout again to the *weather* branch and create a new file called *talking\_about\_weather.txt* and add this sentence *Hey what do you think about the weather today?*. Than add, commit and push

5. In order to see all the changes made in the new branch, inside the main branch, checkout to the main branch and merge inside the *weather* branch.

### II. Some Important Commands When Doing A Big Project In A Group

#### 1. Merging Branches With Pull Request

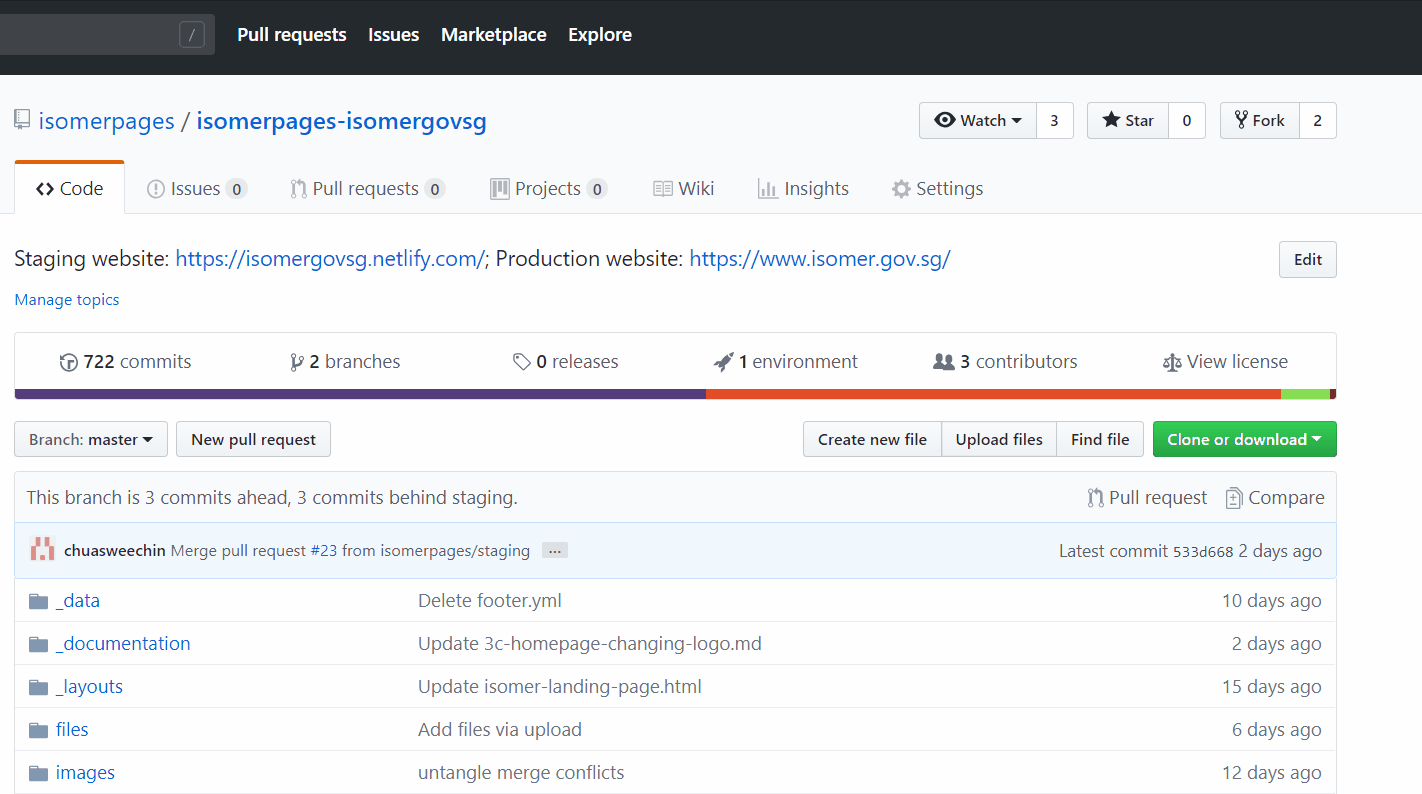
Pull requests are a mechanism for a developer to notify team members that they have completed a feature. Once their feature branch is ready, the developer files a pull request via their Github account. This lets everybody involved know that they need to review the code and merge it into the main branch.

When you file a pull request, all you’re doing is requesting that another developer (e.g., the project maintainer) pulls a branch from your repository into their repository

Using pull requests with each of these workflows is slightly different, but the general process is as follows:

1. A developer creates the feature in a dedicated branch in their local repo.
2. The developer pushes the branch to a public Github repository.
3. The developer files a pull request via Github.
4. The rest of the team reviews the code, discusses it, and alters it.
5. The project maintainer merges the feature into the official repository and closes the pull request.
6. [Details on pull request](https://opensource.com/article/19/7/create-pull-request-github)
7. [Details on pull request](https://www.digitalocean.com/community/tutorials/how-to-create-a-pull-request-on-github)

#### Steps To Merge A Branch Into Main



Set the base branch to the **main** because we want to merge the new branch into the main branch.

Then select the branch which will be merged into the main branch.

It will compare the two branches. If there are no conflicting lines between the two codes, it will show a positive message “Able to merge”. Then click on the button “Create pull request”

After a Pull Request is made, the participant of the team and the owner of the repository will be able to see the changes.  
When everything is well, and the team or you, are satisfied, you can merge the two branches by clicking on “Merge pull request”

Next step: Making sure that your local code is updated from the changes made to the main branch.

##### Exercise 2

1. Work now in pairs (the same as before)

2. You will both work on the RepoB. From the main branch,the person 1 will create a new branch in the Repob called *hobbies*, add a new file *hobbies.txt* with the sentence *Hey I love swimming!*. Than add, commit and push.

3. On Github, click on the button "Compare and pull request", and merge the new branch with the main branch. Check before if they are not merging issues.

## IndexOf() Method In JavaScript

**Done**

In JavaScript, the indexOf() method takes a string and a substring and returns the starting index of the first occurrence of the given substring in that string.

### Syntax

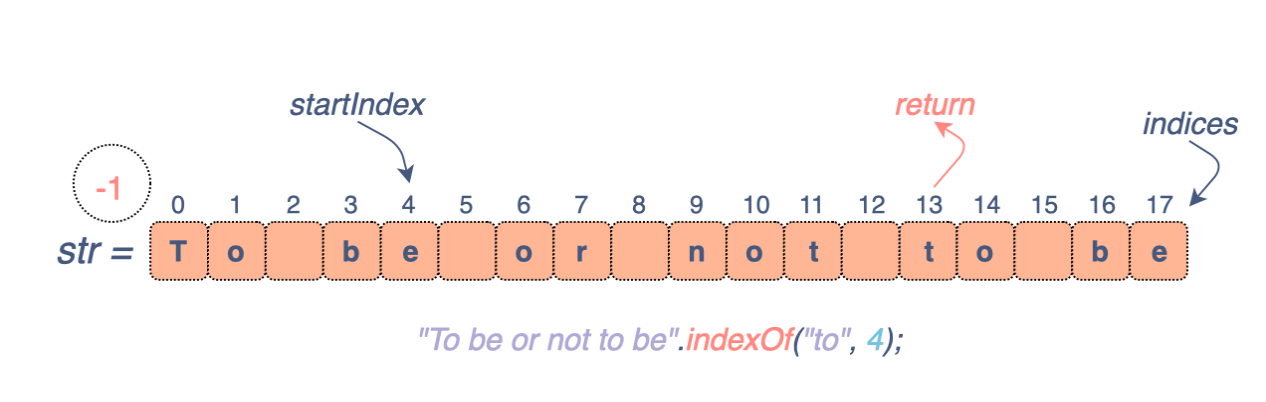
str.indexOf(substr, startIndex);

**str**: String in which the substring substr is searched.

**substr**: Substring whose position is searched in given string str.

**startIndex**: Starting index for the search. If no startIndex is provided, the search starts at the 0th index.

**Return Value**: Returns the start index of the first occurrence of the substr in str or returns -1 (if no occurrence found).



### Examples

As shown in **Line 1 and 5**, the indexOf() method is case-sensitive. The same fact is justified by **Line 6**, where on searching “B”, -1 is returned since only the lowercase forms of the alphabet exist in the given string.

console.log('To be or not to be'.indexOf('To'));

console.log('To be or not to be'.indexOf(' '));

console.log('To be or not to be'.indexOf('o', 2));

console.log('To be or not to be'.indexOf('be', 4));

console.log('To be or not to be'.indexOf('to'));

console.log('To be or not to be'.indexOf('B'));

console.log('To be or not to be'.indexOf('', 9));

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/10/section/33/chapter/1773#breadcrumbs)

**String Split() Method In JavaScript**

**Done**

The split() method in JavaScript enables us to break down a string into an array of (smaller) strings. Let’s jump directly into the code.

**Example**

string = "A-quick-brown-fox" //string to be split

array\_of\_strings = string.split("-") //The split() function

console.log(string) //note that the split function doesn't change the original string

console.log(array\_of\_strings)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

The break character can be any available character. The codes below show the exact same task using space (” “) and empty (””) characters.

Space

Using space character will divide the entire string into individual words

string = "A quick brown fox" //string to be split

array\_of\_strings = string.split(" ") //The split() function

console.log(string) //note that the split function doesn't change the original string

console.log(array\_of\_strings)

Empty

Using empty character will divide the entire string into individual characters

string = "A quick brown fox" //string to be split

array\_of\_strings = string.split("") //The split() function

console.log(string) //note that the split function doesn't change the original string

console.log(array\_of\_strings)

### What Is Javascript?

JavaScript is a **scripting language** used to create and control dynamic website content, for example:  
- autocomplete text suggestions  
- interactive forms

You can type Javascript directly into an HTML document, and web browsers will understand them.  
Using JavaScript doesn’t require downloading any additional programs or compilers.

### Important To Know

* Be careful; JavaScript and Java are two different languages
* JavaScript accepts both double “ and single quotes '
* JavaScript is case-sensitive: A ≠ a
* Only put sensitive data in the server side - not in the client side (it is not safe)
* Insert a comment: // at the beginning of a line OR /\* entire code \*/

### How To Integrate JS Code

There are two possibilities: you can either write your JS code directly in your HTML file or write it in a new JS file that you link to your HTML file (it’s an external JS file like we have been doing with CSS files).

#### Inside Your HTML File

Insert the tag

<script> write your JS code here </script>

Or At the end of your HTML file, just before the closing tag </body>.

Placing scripts at the bottom of the element improves the display speed because script compilation slows down the display.

<html>

<head>

</head>

<body>

...html code…

<script>

...JS code..

</script>

</body>

</html>

#### Create A New JS File: This Solution Is Better.

Just as you create a CSS file for your project, you create a JS file where you write your code.  
Save your file inside a JS folder within your project folder: Project > JS > script.js

Then do not forget to link this file to your HTML file! Place this line at the end of your HTML file, just before the closing tag </body>.

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

The benefits of using a separate JavaScript file :

* Cleaner code
* Easier maintenance
* Pages load faster
* Can run the same JavaScript on several pages on a web site

### The Inspect Element And The Console.Log Command

To open the console on your webpage, go on your webpage, and **Right-click > Inspect.**

If you write your JS code in a file, write console.log(content to display); to display it in the JavaScript console.

You can also write JS code directly in the console. It is helpful to test code, and the result will not appear on your website.

### Javascript Variables

### Variables

In a programming language, variables are used to **store data values**.  
JavaScript uses the letkeyword to declare variables.  
An equal sign is used to **assign values to variables**.

let x = 3

#### Naming

All JavaScript variables must be identified with **unique names**.  
These unique names are called **identifiers**.

The general rules for constructing names for variables (unique identifiers) are:

* Names can contain letters, digits, underscores, and dollar signs.You cannot use any other characters, including spaces, symbols, and punctuation marks.
* Names must begin with a letter
* Names can also begin with $ and \_
* Names are case sensitive (y and Y are different variables)
* You cannot use one of JavaScript’s reserved words as a variable name. All programming languages have a supply of words that are used internally by the language and that can’t be used for variable names because doing so would cause confusion
  + [JavaScript’s keywords](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#keywords)
  + [JavaScript’s reserved words](https://flaviocopes.com/javascript-reserved-words/)
* Make your names descriptive.

When the name contains multiple words, **camelCase** is commonly used. That is: words go one after another, each word except first starting with a capital letter: myVeryLongName.

Examples of valid names:

let userName;

let test123;

Examples of incorrect variable names:

let 1a; // cannot start with a digit

let my-name; // hyphens '-' aren't allowed in the name

#### Keywords

The let keyword tells the browser to create variables:

let x, y;

x = 5 + 6;

y = x \* 10;

#### Comments

Code after double slashes // or between /\* and \*/is treated as a comment.  
Comments are ignored, and will not be executed:

let x = 5; // I will be executed

// let x = 6; I will NOT be executed

#### Undefined

Undefined is when a variable is declared but not assigned to anything.  
let a;  
You have created the variable a, but it has no assignment and is undefined.

### Javascript Primitive Data Types

#### String

This is the text that we put inside quotes: "Hello world"

Strings may have single quotes ('), double quotes ("), or backticks (`) which are read equally, but they must have the exact, consistent quotes at beginning and end.

To use an apostrophe within a string, either uses double quotes around the string or escape the apostrophe with a backslash immediately before it. Like so: 'Jimmy\'s Car'

When you add two strings (or even a string and a number), JS will perform string concatenation and return the two strings together:

let a = "Hello";

let b = "World";

let c = a+b;

console.log(c);

The console would show: "HelloWorld".

Anything inside the quotes is considered a string, even numbers: “43” (string) ≠ 43 (number).

##### Long Literal Strings

Rather than having lines that go on endlessly, you can specifically break the string into multiple lines in  
the source code without affecting the actual string contents.

**The + operator**

let longString = "This is a very long string which needs " +

"to wrap across multiple lines because " +

"otherwise, my code is unreadable.";

**The backslash character \**

At the end of each line to indicate that the string will continue on the next line.  
Make sure there is no space or any other character after the backslash.

let longString = "This is a very long string which needs \

to wrap across multiple lines because \

otherwise my code is unreadable.";

##### String Properties

* Built-in length property:

let lengthTxt = longString.length

##### String Methods

* The indexOf() method: returns the **index of (the position of) the first occurrence** of a specified text in a string:  
  Return -1 if the text is not found

let str = "Hello Everyone, please say hello to the class";

let posUppercaseHello = str.indexOf("Hello"); // case sensitive

let posLowercaseHello = str.indexOf("hello");

console.log(posUppercaseHello) // 0

console.log(posLowercaseHello) // 27

let secondstr = "hello Everyone, please say hello to the class";

let secondStrPosHello = secondstr.indexOf("hello");

console.log(secondStrPosHello) // 0

* The substring(startIndex, endIndex) methods:  
  Returns the characters in a string that begins at “startIndex” and ends at the “endIndex”.  
  “endIndex” is optional, and if omitted, up to the end of the string is assumed.  
  “endIndex” is not included.

let str = "Hello Everyone, please say hello to the class ";

str.substring(0,4) //returns "Hell" --> the index 4 isn't included

str.substring(2,4) //returns "ll"

str.substring(15) //returns "please say hello to the class"

* The toLowerCase() method: Returns the string with all of its characters converted to lowercase.

let myStr = "Hello Everyone, Please say hello to the class ";

myStr.toLowerCase() //returns "hello everyone, please say hello to the class "

##### More String Methods

* replace()
* concat()
* toUpperCase()
* trim()
* charAt()

## Exercise 1

Create a structured HTML file linked to a JS file

1. Create these variables and give them values:

* addressNumber
* addressStreet
* country

2. Write a variable named *globalAddress*, and concatenate inside, the variables:

* addressNumber
* addressStreet
* country

In order to create a sentence

3. Display *globalAddress* Example: *globalAddress* should display « I live in BenYehuda 5, in Israel »

#### Number

It can be either whole numbers (4), negative numbers (-3), or decimal numbers (9.3).  
You can do mathematical operations in the console of JS: + - / \*

console.log(5+3)

// It will display "8"

If you try to do math with object types that are not valid numbers you will get an NaN (not a number) error.

##### Number Methods

* The isNaN(x)method: return true or false

let op = "me";

isNaN(op); //returns true because op is Not a Number

* The toString() method: returns a number as a string.

let op = 10;

op.toString(); //returns "10"

* The toFixed() method: returns a string, with the number written with a specified number of decimals:

let op = 10.6789

op.toFixed(0); // returns 11

op.toFixed(2); // returns 10.68.

## Exercise 2

1. Store your birth year in a variable.

2. Store a future year in a variable.

3. Calculate your possible ages for that year based on the stored values.

4. Display : "I will be NN in YYYY", substituting the values.

#### Boolean

JavaScript has a Boolean data type. It can only take the values true or false

They are interpreted as a number: 1 or 0

* The Boolean() method: to find out if an expression (or a variable) is true

let op = Boolean(10 > 9) // op returns true

#### Null

The special null value does not belong to any of the types described above.

It forms a separate type of its own which contains only the null value

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/10/section/33/chapter/94#breadcrumbs)

### Array

JavaScript arrays are used to store multiple values in a single variable. : it’s a **list** of values.

If you have many variables, for example, a login page with many users, you can write everything into one variable: an array. Arrays can have many different data types within them, but it’s best to keep consistent data types within one array.

let user1 = "John"

let user2 = "Bill"

let user3 = "Nancy"

Instead of that, which will get out of hand quickly, you can use an array:

let users = ["John", "Bill", "Nancy"]

You place square brackets around the list and separate the items with commas.

Arrays are a particular type of object. The typeof() operator in JavaScript returns “object” for arrays.

Syntax

let arrayName = [item1, item2, ...];

let colors = ["blue", "yellow", "green"];

console.log(typeof(colors)) // object

#### Accessing Elements In An Array

An array is an ordered list; therefore, its elements can be referred to by their index, starting with zero.

You use the index number: starting at **0**

let colors = ["blue", "yellow", "green", 54];

let favorite = colors[0];

let secondFavorite = colors[2];

console.log(favorite) // blue

console.log(secondFavorite) // green

You can create an array within an array:

let sampleArray = [

[1, 2, 5],

[7, 6, 10, 11, 12],

[3]

]

This created an array with a length of 3. If you want to get the number 11, you will access it as sample\_array[1][3].

#### Changing An Array Element

let colors = ["blue", "yellow", "green"];

colors[0] = "pink"

console.log(colors) // ["pink", "yellow", "green"];

#### Array Properties

* The length property: returns the length of an array (the number of elements inside the array).
* The length starts counting at **1** (not at **0** like the index (ie.seen above))

let colors = ["blue", "yellow", "green"];

colors.length // 3

#### Array Methods

* The push() method: adds a new element to the end of an array

let colors = ["blue", "yellow", "green" ];

colors.push("orange")

console.log(colors) // ["blue", "yellow", "green", "orange"];

* The pop() method : removes the last element from an array:

let colors = ["blue", "yellow", "green" ];

colors.pop()

console.log(colors) //["blue", "yellow"];

* The splice() method : adds new items to an array/ remove items

The first parameter defines the position where new elements should be added (spliced in).  
The second parameter defines how many elements should be removed.  
The rest of the parameters define the new elements to be added.  
The splice() method returns an array with the deleted items:

let colors = ["blue", "yellow", "green" ];

colors.splice(1, 1, 45, 23);

console.log(colors) // ["blue", 45, 23, "green"];

* The slice() method: slices a piece of an array and starts a new array.  
  The slice() method creates a new array. **It does not remove any elements from the source array**.  
  The method then selects elements from the start argument and up to (but not including) the end argument.

let colors = ["blue", "yellow", "green", "pink" ];

let favorite = colors.slice(2)

console.log(favorite) // ["green" , "pink"];

console.log(colors) // ["blue", "yellow", "green", "pink" ];

let secondFavorite = colors.slice(0,1)

console.log(secondFavorite) //[ 'blue' ]

console.log(colors)// ["blue", "yellow", "green", "pink" ];

* the toString()method: converts an array to a string of (comma separated) array values.

let colors = ["blue", "yellow", "green" ];

let colorstring = colors.toString()

console.log(colorstring) // blue,yellow,green

console.log(colors) // ["blue", "yellow", "green" ];

#### More Array Methods

* You can join all the elements of an array with join() method. Whatever’s within the parentheses will be placed between each item.
* To remove the first element, use shift()
* unshift({item}) will add a new element to the list’s head. It will become index 0

#### Exercise 3

1. Create a numerically indexed table (ie. an array): *pets*, like this *['cat','dog','fish','rabbit','cow']*

2. Display *dog*

3. Add to the array *pets*, the pet horse. Remove the pet rabbit

4. Display the array length

### User-Interface Functions

The mini popup window with a message is called a modal window. The word “modal” means that the visitor can’t interact with  
the rest of the page, press other buttons, etc., until they have dealt with the window.

#### 1. Alert

This shows a message and **pauses script execution** until the user presses “OK”.

alert(message);

alert("Hello");

#### 2. Prompt

It shows a modal window with a text message, **an input field** for the visitor, and the buttons OK/Cancel.  
The visitor may type something in the prompt input field and press OK. Or they can cancel the input by pressing Cancel or hitting the Esc key.  
The call to prompt returns the text from the input field or null if the input was cancel

prompt(title, [default]) \* **title** : the text to show the visitor. \* **default**: optional second parameter, the initial value for the input field.

let age = prompt('How old are you?', 20);

alert(`You are ${age} years old!`); // You are 20 years old!

#### 3. Confirm

The function confirm shows a modal window with a question and two buttons: OK and Cancel.  
The result is true if OK is pressed and false otherwise.

confirm(question)

let isBoss = confirm("Are you the boss?");

alert(isBoss); // true if OK is pressed

## What You Will Learn

* Comparison

### Comparison

| **Symbol** | **Meaning** |
| --- | --- |
| = | is assignment |
| == | is comparison of value |
| === | is comparison of value and type |

| **Symbol** | **Meaning** |
| --- | --- |
| != | Not equal |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| || | Or |
| && | And |
| ! | Not (if x is true, then x! is false) |

**Some more help on comparisons**[**here**](https://www.w3schools.com/js/js_comparisons.asp)

### Some Examples

When you put an exclamation mark before a true variable, it will change it to false.

let x = true

console.log(x)

>> true

console.log(!x)

>> false

When you put an exclamation mark before a false variable, it will change it to true.

let y = false

console.log(y)

>> false

console.log(!y)

>> true

**The exclamation mark means not**

let test = true

console.log(test)

>> true

console.log(!test) // means not test // means not true // means false

### Working With Numbers

| **Operator** | **Example** | **Same As** |
| --- | --- | --- |
| ++ | x++ | x = x + 1 |
| -- | x-- | x = x - 1 |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/10/section/33/chapter/95#breadcrumbs)

## Switch Statement In JavaScript

**Not Completed**

The switch statement is used to control the flow of a program and is a part of JavaScript’s conditional statements. It allows the execution of a specific code block depending upon the evaluation of an expression.

### The Structure Of Switch

* The switch structure starts with the switch keyword followed by the expression to be evaluated in parentheses.

switch (expression){

}

* Inside the curly braces different cases are defined followed by a value to be strictly matched with the evaluated expression.

switch (expression){

case value1:

/\* implement the statement(s) to be executed when

expression = value1 \*/

break;

case value2:

/\* implement the statement(s) to be executed when

expression = value2 \*/

break;

case value3:

/\* implement the statement(s) to be executed when

expression = value3 \*/

break;

default:

/\* implement the statement(s) to be executed if expression

doesn't match any of the above cases \*/

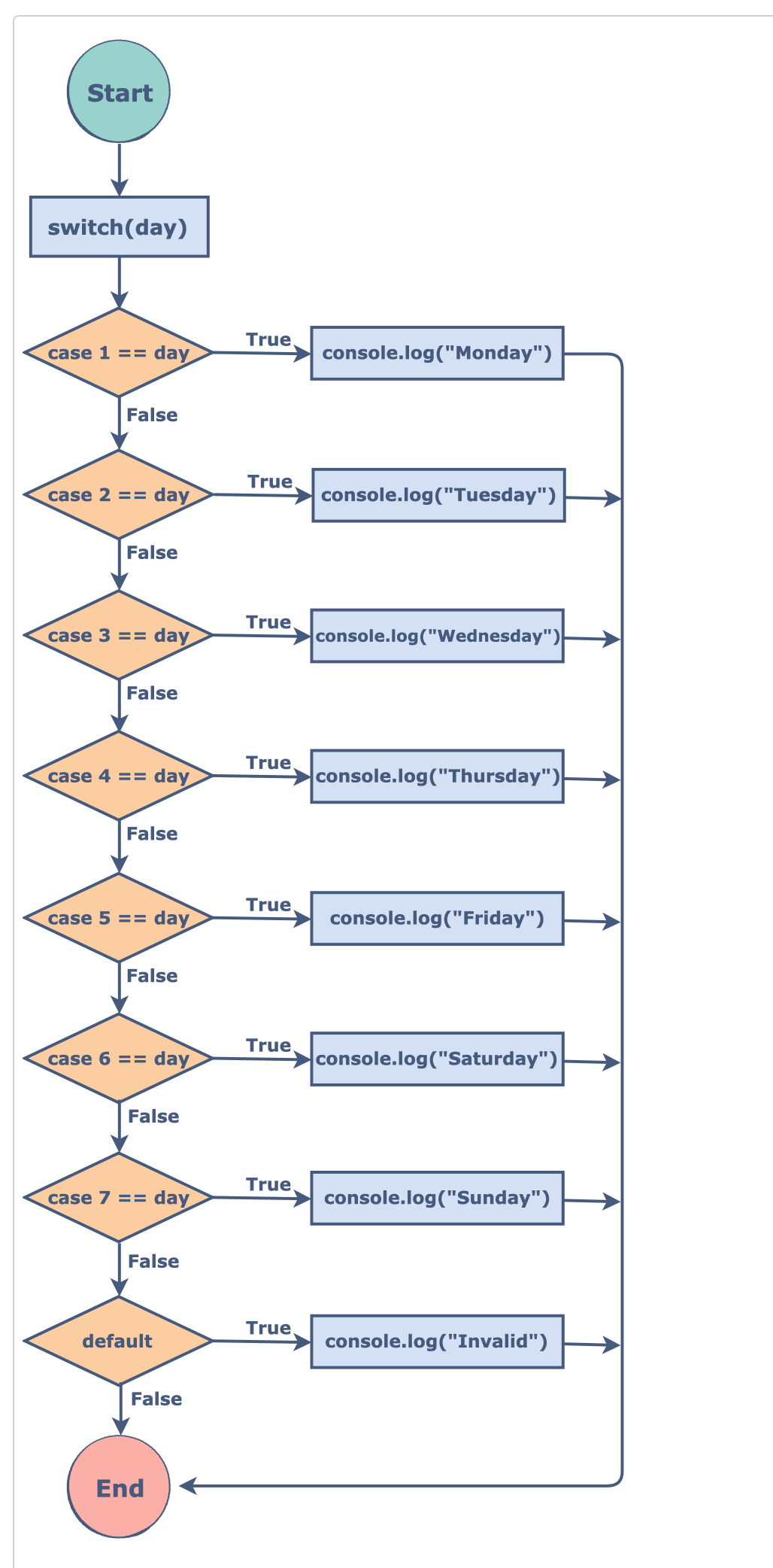
}

* The break statement is used to exit the switch structure after the execution of a case. If it is not used, all the subsequent cases will be executed until the program encounters any break statement or the ending curly brace } of the structure.
* The default: is a special type of case and is executed when none of the cases match the evaluated expression.

### Example

The program will find out the day of the week based on the value of the day variable.

Let’s look at the flow chart of this example for a better understanding:



Let’s look at the implementation of the example:

var day = 2; //change and try with different values

switch(day)

{

case 1: //if day = 1

console.log("Monday");

break;

case 2: //if day = 2

console.log("Tuesday");

break;

case 3: //if day = 3

console.log("Wednesday");

break;

case 4: //if day = 4

console.log("Thursday");

break;

case 5: //if day = 5

console.log("Friday");

case 6: //if day = 6

console.log("Saturday");

case 7: //if day = 7

console.log("Sunday");

break;

default: //if day doesn't match any of above

console.log("Invalid");

}

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/10/section/34/chapter/1771#breadcrumbs)

## What Are JavaScript Short-Circuiting

**Not Completed**

# What Are JavaScript Short-Circuiting Operators?

By **Olga Smirnova** - Developers Institute Alumni

Commonly, logical operators are used to combine two boolean values. But the truth is that we can do a lot more with the AND and OR operators.

In JavaScript, **short-circuiting** is the evaluation of an expression from left to right with || and && operators.

If the condition is met and the rest of the conditions won’t affect the already evaluated result, the expression will short-circuit and return that result (value).

### Short-Circuiting With The Or (||) Operator

The || **operator** will return the first truthy value of all the operands, or simply the last value if all of them are falsy.

**JAVASCRIPT - script.js**

true || false

// returns true

### Code

What do you think the result of this one will be? Run the code to check.

**JAVASCRIPT - script.js**

console.log(3 || 'Orange');

console.log('' || 'Orange');

console.log(true || 0);

console.log(undefined || null);



### Explanation

In the example below we want to check if the object person contains the job key. To do this, we simply console.log the value of person or for person.job we use OR to get the default string of unemployed.

**JAVASCRIPT - script.js**

var person = {

name: 'Jack',

age: 34

}

console.log(person.job || 'unemployed');

// 'unemployed'

Since person.job doesn’t exist, we get undefined. Since undefined is a falsy value, JavaScript will instead go to the other side of the || and accept whatever value is there.

### Short-Circuiting With The And (&&) Operator

The && operator will return false as soon as it gets any falsy value and will return the last true value if all the values are truthy.

**JAVASCRIPT - script.js**

true && false

// returns false

### Code

Now try to predict the result of this code. Run it to check.

**JAVASCRIPT - script.js**

console.log(0 && 'Orange');

console.log('' && 'Orange');

console.log(true && null);

console.log('Apple' && 'Orange');



### Explanation

It’s possible to take advantage of short-circuiting and to shorten, or even avoid, the if statement.

In the example below, we want to check if an object person key age value is higher than 18. And if yes, we state that this person is allowed to drive.

**JAVASCRIPT - script.js**

var person = {

name: 'Jack',

age: 34

}

console.log(person.age > 18 && 'Driving allowed');

// 'Driving allowed'

Since person.age is higher than 18, we get true. In this case, JavaScript has no false value, so the && operator returns the last true value.

To sum it up, for practical applications, we can use the OR operator to set default values, and we can use the AND operator to execute code in the second operand if the first one is true.

[**Back to Top**](http://learn.di-learning.com/courses/collection/6/course/10/section/34/chapter/1901#breadcrumbs)

### Conditional JS

JavaScript supports conditional statements, which are used to perform different actions based on various conditions.

#### If…Else



JavaScript supports the following forms of if..else statement :

* if statement
* if…else statement
* if…else if… statement.

##### 1. If Statement

Syntax

if (expression) {

Statement(s) to be executed if the expression is true

}

let age = 20

if (age > 18) {

console.log("You're a big man")

}

##### 2. If…Else Statement

Syntax

if (expression) {

Statement(s) to be executed if expression is true

} else {

Statement(s) to be executed if the expression is false

}

let age = 20

if (age > 18) {

console.log("We can go to a pub together !!")

} else {

console.log("Sorry...You have to stay at home tonight")

##### 3. If…Else If… Statement.

Syntax

if (expression 1) {

Statement(s) to be executed if expression 1 is true

} else if (expression 2) {

Statement(s) to be executed if expression 2 is true

} else if (expression 3) {

Statement(s) to be executed if expression 3 is true

} else {

Statement(s) to be executed if no expression is true

}

let age = 20

if (age === 18) {

console.log("It's your birthday !!)

} else if (age > 18) {

console.log(We can go to a pub together !!")

} else {

console.log("Sorry...You have to stay at home tonight")

}

##### Exercise 1

Make a keyless car!

This car will only let you drive if you are over 18. Make it do the following:

Using prompt() and alert(), ask a user for their age.

* IF they say they are below 18, respond with: "Sorry, you are too young to drive this car. Powering off
* IF they say they are 18, respond with: "Congratulations on your first year of driving. Enjoy the ride!
* IF they say they are over 18, respond with: "Powering On. Enjoy the ride!"

### Switch Case

Use the switch statement to select one of the many code blocks to be executed.

You can include many different options in a switch statement.  
Include a break statement after each option, so it will not execute all of them.  
In the end, add a default parameter to be executed if no other parameter is met.

Syntax

switch(expression) {

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

}

#### 1. How Does It Work?

* The switch expression is evaluated once.
* The value of the expression is compared with the values of each case.
* if there is a match, the associated block of code is executed.

If multiple cases match a case value, the **first** case is selected.  
Switch cases use **strict comparison (===)**. The values must be of the same type to match.

let fruit = 'Papayas';

switch (fruit) {

case 'Oranges':

console.log('Oranges are $0.59 a pound.');

break;

case 'Mangoes':

case 'Papayas':

console.log('Mangoes and papayas are $2.79 a pound.');

// expected output: "Mangoes and papayas are $2.79 a pound."

break;

default:

console.log('Sorry, we are out of ' + fruit + '.');

}

#### 2. The Break Keyword

When JavaScript reaches a break keyword, it breaks out of the switch block.  
This will stop the execution inside the block.  
It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

#### 3. The Default Keyword

The default keyword specifies the code to run if there is no case match:  
The default case does not have to be the last in a switch block:

let fruit = 'Papayas';

switch (fruit) {

case 'Oranges':

console.log('Oranges are $0.59 a pound.');

break;

default:

console.log('Sorry, we are out of ' + fruit + '.');

break;

case 'Mangoes':

case 'Papayas':

console.log('Mangoes and papayas are $2.79 a pound.');

// expected output: "Mangoes and papayas are $2.79 a pound."

break;

}

##### Exercise 2

Write as comments the steps of the resolution of this piece of code

Guess what will be the result before checking it

let a = 2 + 2;

switch (a) {

case 3:

alert( 'Too small' );

break;

case 4:

alert( 'Exactly!' );

break;

case 5:

alert( 'Too large' );

break;

default:

alert( "I don't know such values" );

}

##### Exercise 3

Write as comments the steps of the resolution of this piece of code

Guess what will be the result before checking it

let a = 2 + 2;

switch (a) {

case 4:

alert('Right!');

break;

case 3: // (\*) grouped two cases

case 5:

alert('Wrong!');

alert("Why don't you take a math class?");

break;

default:

alert('The result is strange. Really.');

}

### Objects

The object type is unique.

All other types are called “primitive” because their values can contain only a single thing (be it a string or a number). In contrast, objects are used to store collections of data and more complex entities.

Objects in JavaScript may be defined as an unordered collection of related data, in the form of  
“key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in an object’s context.

**Example** : A lamp has properties –> height,width, brightness and has methods (an action) –> to shine

Syntax

Empty object :

let objName = {};

let x = {

property: value,

property: value,

method: function(){}

...

};

let person = {

firstName: "John",

lastName: "Doe",

age: 50,

eyeColor: "blue"

};

#### 1. Accessing Object Properties

Syntax

objectName.propertyName

OR

objectName["propertyName"]

let person = {

firstName: "John",

lastName: "Doe",

};

console.log(person.firstName) // John

console.log(person["firstName"]) // John

#### 2. Adding/ Changing Object Properties

let person = {

firstName: "John",

lastName: "Doe",

};

person.firstName = "Sarah"

person.eyeColor= "blue"

console.log(person)

// {

// firstName: "Sarah",

// lastName: "Doe",

// eyeColor: "blue"

// };

#### 3. Deleting Object Properties

let person = {

firstName: "John",

lastName: "Doe",

};

delete person.firstName

console.log(person)

// {

// lastName: "Doe",

// };

##### Exercise 1

Create a stuctured html file linked to a JS file

1. Create an object that has properties "username" and "password". Fill those values in with strings.

2. Create an array which contains the object you have made above and name the array "database".

3. Create an array called "newsfeed" which contains 3 objects with properties "username" and "timeline".

### What Is Debugging?

Debugging is the routine process of locating and removing computer program bugs, errors, or abnormalities, which software programmers methodically handle via debugging tools.

#### How To Debug

#### Mandatory Simple Tutorial To Follow

1. [Javascript Info Tutorial](https://javascript.info/debugging-chrome)

#### Advanced Tutorials

1. [Debugging tools](https://raygun.com/learn/javascript-debugging-tools)
2. [Debugging pratices](https://blog.bitsrc.io/javascript-debugging-best-practices-e28a09b4528a)

**Regular Expressions (Advanced)**

**Not Completed**

### About RegEx:

A regular expression is an object that describes a pattern of characters.  
Expressions are used to perform pattern-matching and “search-and-replace” functions on text.

#### Documents About Regular Expression: PDF To Download

* [Explanation about Regular Expression](https://github.com/devtlv/studentsGitHub/files/5843795/Javascript.Regular.Expressions.pdf)
* [RegExp CheatSheet](https://github.com/devtlv/studentsGitHub/files/5843797/regex-cheat-sheet-1.pdf)

#### Example With RegEx:

let str = "Happy BirthDay";

let patt = /birthday/i;

let result = str.match(patt);

console.log(result); //returns true

if (result){

console.log('Yes')

} else{

console.log('No');

}

#### Example Explained:

/birthday/i is a regular expression.

* birthday is a pattern (to be used in a search).
* i is a modifier (modifies the search to be case-insensitive).

#### RegEx Example 2: Test To Match An Email:

let regex = /^.+@.+\..+$/;

console.log(regex.test('johndoe@gmail.com')); //returns true