Real Time Group

Web Foundations

Web Foundations

Web Basics

What is Internet

- Internet is really a network of computing resources.
- Think of it as a spider net, where each line is a computing device connected to it and is capable of sending and receiving some resources.
- Moreover a router is responsible of giving the computing piece the ability to send resources to other routers which will also push them to the connected devices in its spot.

Web Basics

Internet major uses

Email – A fast, easy, and inexpensive way to communicate with other Internet users around the world.

Telnet – Allows a user to log into a remote computer as though it were a local system.

FTP – Allows a user to transfer virtually every kind of file that can be stored on a computer from one Internet-connected computer to another.

World Wide Web (WWW) – A hypertext interface to Internet information resources.

Web Basics

What is WWW

WWW stands for World Wide Web. A technical definition of WWW is as follows:
All pieces of computing that are capable and are using the HTTP protocol to transfer resources between them.

In simple words, WWW is a way of exchanging information between different computers which are connected to the internet. This connection makes them a collection of interactive multimedia resources.

Web Basics

What is URL

URL stands for Uniform Resource Locator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files).

Web Basics

A URL format looks like the following:
protocol://hostname/other_information

The protocol is the main part here, describing how the data will be transferred.
The most common protocol is HTTP.

Hostname is the name of a specific computer hosting resources we are trying to reach. Most used with a DNS to have a memorable name instead of an IP address.

Last part is a path inside the host where specific resources are sitting. This part may vary depending on the server configurations.

Web Basics

What is IP

- An IP address, short for Internet Protocol address, is an identifying number for a piece of network hardware.
- Having an IP address allows a device to communicate with other devices over an IP-based network like the internet.

- Most IP addresses look like this:
  151.101.65.121 (IPv4)
- Other IP addresses you might come across could look more like this:
  2001:4860:4860::8844 (IPv6)

Web Basics

What is IP

- IP addresses are made of 4 bytes, each between 0 to 255.
  i.e. : 255.255.255.255 or in more realistic way 192.168.1.28
- IP address is given to any PC that is connected to a router and is used to communicate over the internet.

- There is a limited amount of addresses we can use in IPv4 .
- This is the reason why IPv6 has been implemented
  (but not widely used just yet).

HTTP & HTTPS

What is HTTP

- HTTP stands for Hypertext Transfer Protocol.
- This is the protocol being used to transfer hypertext documents that makes the World Wide Web possible.

What is HTTPS

- HTTPS is the secured version of HTTP.
- This protocol allows us to keep some sensitive information hidden.
- This is the used protocol for sites with sensitive information.
- Its considered a good practice to use it with every website.

Web Basics

What is Website

- Website is a collection of HTML pages hosted by some specific host.
- By reaching the hosts address, it provides the same resources to every incoming user.

- Each page on a website has its own role and job to do.
- Each page is created to show specific information and to maintain order and convenience.

Web Servers

What is Web Server

- A server is a computer hosting a website or a web application.
- This computer has to be connected to the internet all the time to provide correct functionality.
- A web server is the place where everything really happens. When some user enters our website, it actually reaches the application running on the server.
- The server makes some needed calculations and if everything is correct it sends the user the requested HTML content.
- Think of it like - behind the scenes of some show. Everything that we see on a website, is first located on a server, calculated and then sent to the website.

Web Servers

What is Web Server

- Common usage of servers are storing data in a databases.
  We can use servers to add,edit or remove data from the database.

- Below are some other usages for servers:
  # Storing files
  # performing some tasks on a file and returning it back
  # delivering given data to other locations

- Common platforms of servers might be: php, nodeJS or python.

Web Servers

What is REST

- Representational State Transfer is stateless and is not affected by architecture of any side.
- Meaning we can make changes to the server or the client side without affecting the correct data transfer.
- There are 4 common usages of this communication:

  # GET - requests some data from the server

  # POST - Send some bulk data to the server

  # PUT - Sends single data to the server

  # DELETE - Sends a request to remove data from the server

Web Servers

What is REST

Rest meant to make our work easier, by using same tech to work and communicate our data across all platforms.

Another reference to this technique called CRUD - Create, Read, Update and Delete. It is specifically made to be easily read and used.

When working with data, We use CRUD to manipulate our data and achieve the result we want.

Web Servers

REST & CRUD

- In a REST environment, CRUD often corresponds to the HTTP methods POST, GET, PUT, and DELETE, respectively. These are the fundamental elements of a persistent storage system.

- REST - CRUD
  - GET = Read
  - POST = Create
  - UPDATE = Put
  - DELETE = Delete

- Keeping these rules, we can eliminate common mistakes, save time and help others easily understand our intentions.

What is JSON

- JavaScript Object Notation is a syntax for storing and exchanging data.
- When exchanging data between a browser and a server, the data can only be text.
- Using JSON is very convenient since converting it from text to a JavaScript object is simple.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Web Basics

What is JSON

- JSON looks exactly like JS object to keep the transformation simple:

Example:

```
{
  “Name”: “John”,
  “Age”: 30,
  “City”: “Jerusalem”
}
```

Web Browsers

What is Web Browser

- Web Browser is a software installed on our PC.
- When we want to use the internet and surf websites, we must use a browser like Chrome or Firefox.

- A browser is developed to request and receive resources from the internet, translate and show them to us - the users.
- Every browser has its own team of developers, but they all must use the guidance of a global internet organization which writes the rules of our internet today.

Web Servers

What is SMTP Server

SMTP stands for Simple Mail Transfer Protocol Server. This server takes care of delivering emails from one server to another server. When you send an email to an email address, it is delivered to its recipient by a SMTP Server.

While it is a different protocol, it is not that different and made only for a purpose of delivering mails from one point to another.

Web Basics

What is ISP

- ISP stands for Internet Service Provider. They are the companies whom provide you the service in terms of internet connection(and Bandwidth).

- These companies will take care of the Hardware (i.e. antennas and cables) and software (i.e. special drivers if needed) in order to connect to the internet.
- After joining one of such companies, you will obtain your own IP address to start surfing the web.

Question: is your given IP STATIC ?

Web Basics

What is HTML

- HTML stands for Hyper Text Markup Language.
- This is the language in which we write web pages for any Website.
- It is an old and agreed language to provide information along the internet.

- It is the only internet language, which is managed by global organizations to keep it safe, ease to use and related to our time.
- HTML 5 is the latest version used, includes improvements in comparison to the previous release.

Web Basics

What is Hyperlink

- A hyperlink or simply a link is an item on a website, that is used to navigate us around the pages of a website or websites.
- A general rule is to use a simple text links for that purpose but any element like button or image will do the trick.

Web Basics

What is DNS

DNS stands for Domain Name System. When an address is entered in your browser, it will be directed to some specific website. As we know after joining some ISP, we now have an IP address of our specific PC.
But using an IP is really hard to remember when we have so many different websites around.

DNS is a service, that allows us to give a text name to an IP address so it will be more likely to us to remember.

The result of a DNS is usually just called a domain which is used to reach our server.

Web Basics

What is a Domain

Domain is DNS name for our website host IP. Usually we buy it from some provider with a yearly payment. The best practice is to use our brand name so it will be easy to find us. The ending part is used to infrom the users what kind of service is running on our website.

Examples may be:
.com : a general website.
.info : an informative website.
.tv : tv providers website
.gov : a government website

Web Basics

What is a request and response

Requests and Response is the way HTTP protocol works. When a client trying to receive resources from the server it sends a request. The server then calculates what it should do with the request and send back a response.

The request itself contains:
The host who sent the request
The address it is trying to reach
Headers with additional information about the request
Optional data to send to the server

We will cover Headers in the security section of the course.

Web Basics

What is a request and response

Response is the answer of the server to the given request. It contains further headers which will affect the connection and its data.
Usually there are headers for security reasons to block unwanted behavior and to understand what types of data are in use.

Header Example:

```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Sun, 25 Feb 2018 16:32:55 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: keep-alive
Location: https://www.webnots.com/
Cache-Control: max-age=3600
Expires: Sun, 25 Feb 2018 17:32:55 GMT
```

Web Basics

What is a request and response

Response is the answer of the server to the given request. It contains further headers which will affect the connection and its data.
Usually there are headers for security reasons to block unwanted behavior and to understand what types of data are in use.

Header Example:

```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Sun, 25 Feb 2018 16:32:55 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: keep-alive
Location: https://www.webnots.com/
Cache-Control: max-age=3600
Expires: Sun, 25 Feb 2018 17:32:55 GMT
```

Web Basics

What is W3C

W3C stands for World Wide Web Consortium which is an international consortium of companies involved with the Internet and the Web.

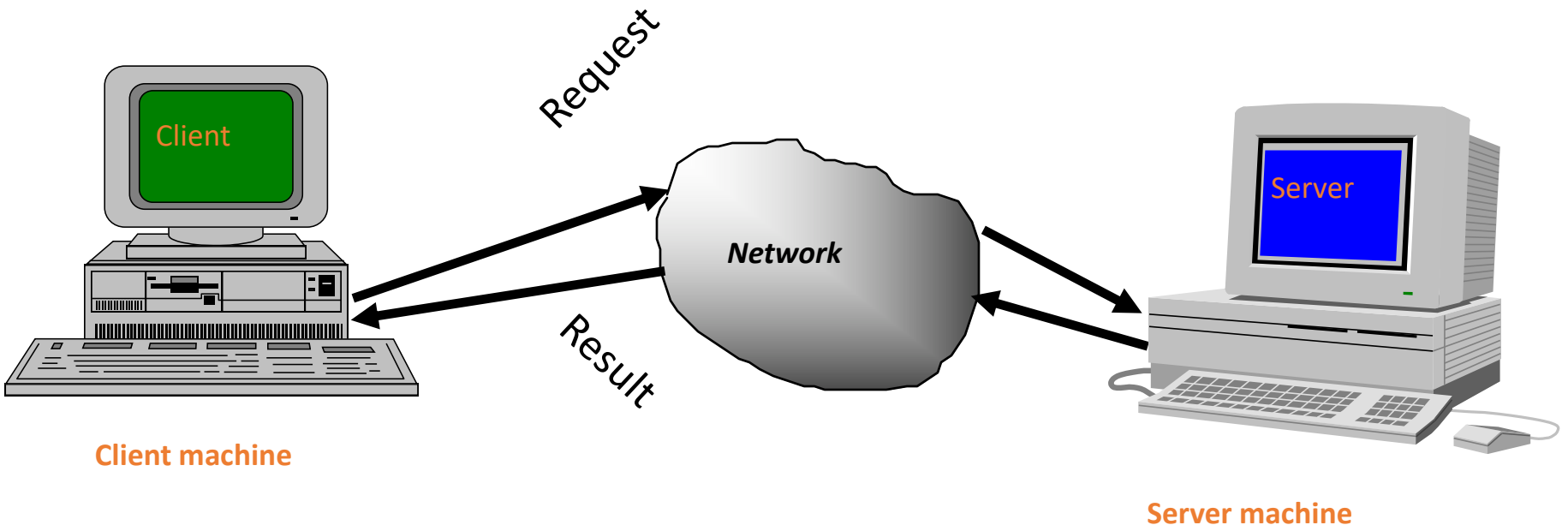
The W3C was founded in 1994 by Tim Berners-Lee, the original architect of the World Wide Web. The organization's purpose is to develop open standards so that the Web evolves in a single direction rather than being splintered among competing factions. The W3C is the chief standards body for HTTP and HTML.

Understanding servers:

A simple definition of Client Server is

“ server software accepts requests for data from client software and returns the results to the client”

a client, a server, and network



Functional Roles

- Data storage
- Application Host
- Processing data
- Data management
- Website Host

Categories of Servers

- File Server
- Data Server
- Computer Server
- Database Server
- Communication Server
- Video Server

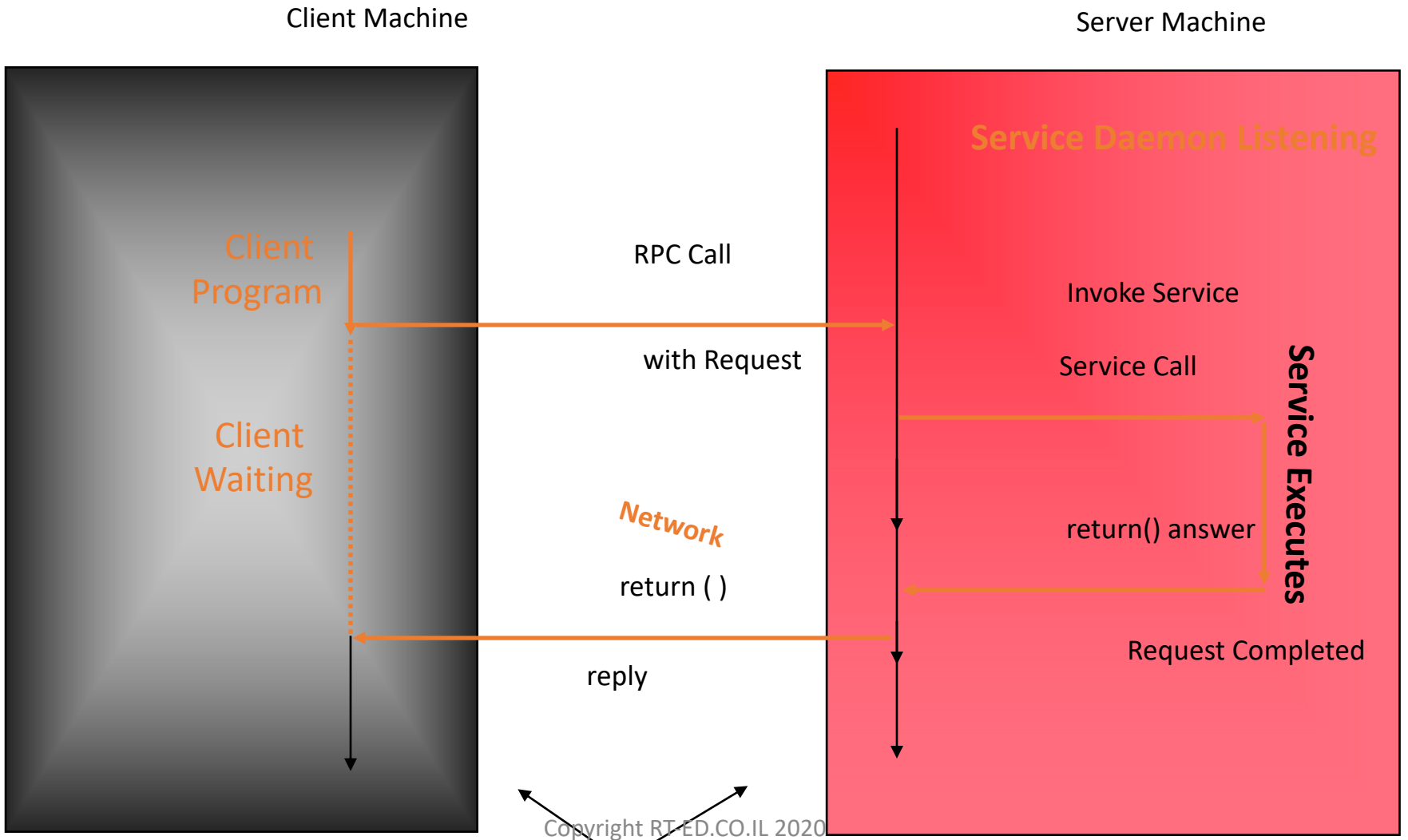
34

File Server

- File Servers manage a work group's application and data files, so that they may be shared by the group.
- Very I/O oriented
- Pull large amount of data off the storage subsystem and pass the data over the network
- Requires many slots for network connections and a large-capacity, fast hard disk subsystem.

35

Flow Control in a Sychronous RPC



Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

36

Compute Server

- Performs Application logic processing
- Compute Servers requires
  - processors with high performance capabilities
  - large amounts of memory
  - relatively low disk subsystems
- By separating data from the computation processing, the compute server's processing capabilities can be optimized

37

Database Server

- Most typical use of technology in client-server
- Accepts requests for data, retrieves the data from its database

  (or requests data from another node)and passes the results back.
- Compute server with data server provides the same functionality.
- The server requirement depends on the size of database, speed with which the database must be updated, number of users and type of network used.

38

Why are Servers Faster ?

- More Ram and CPU cores
- Different architecture in RAM allocation
- Different architecture in CPU usage
- Have predefine tasks , and design for it.

A Little about Linux

Introduction :

- most of servers in our days functions with LINUX operating system.
- In this section we will learn the basic of Linux filesystem handling.
- We will travel alongside the file system and create files and directories,
- Copy files, remove files, view the content of files, connect to remote server and copy file to/from my computer and more…

Everything is a file

Almost everything in Unix is a file!

-	Regular files
d	Directories (Directories are just files listing a set of files)
l	Symbolic links (Files referring to the name of another file)
c , b	Devices and peripherals (Read and write from devices as with regular files)
p	Pipes (Used to cascade programs => cat *.log | grep error)
s	Sockets (Inter process communication)

42

File names

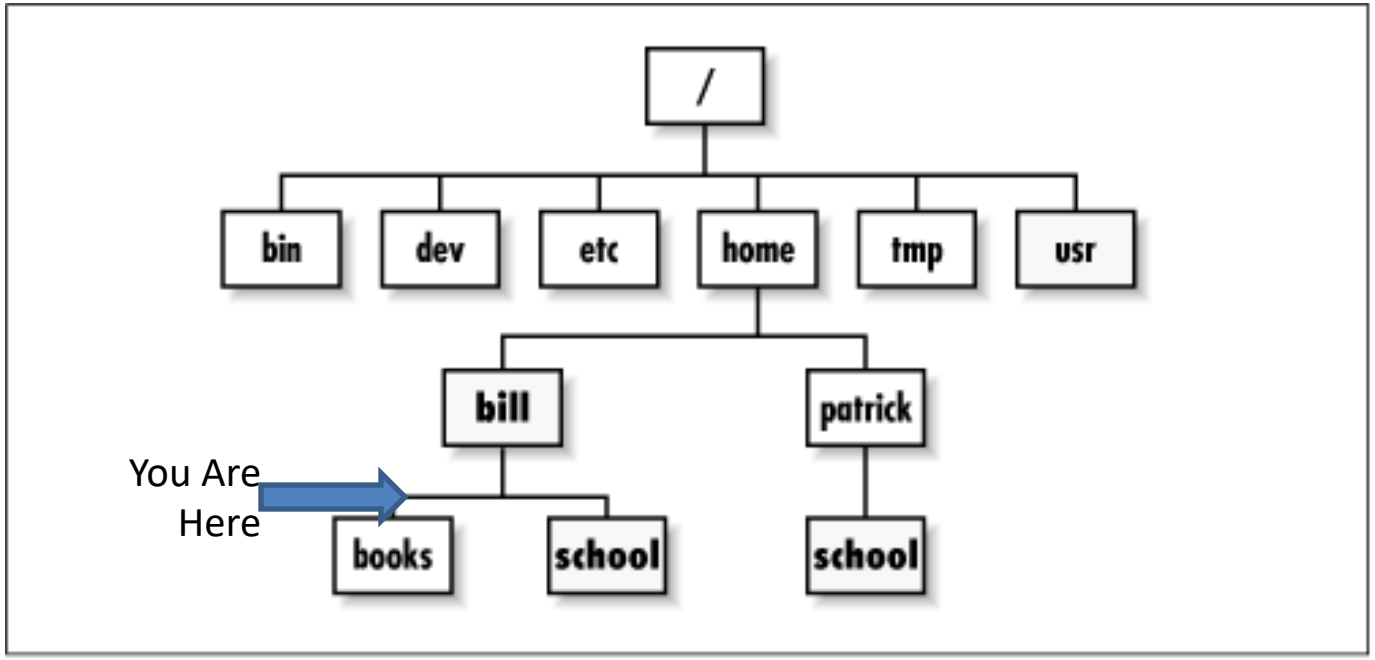
File name features since the beginning of Unix

- Case sensitive
- No obvious length limit
- Can contain any character (including whitespace, except /).
- File name extensions not needed and not interpreted by linux. Just used for user convenience. Applications may require an extension.

File name examples: `README` `.bashrc` `Windows` `Buglist` `index.html` `index.htm` `index.html.old`

File name with special characters (e.g spaces or uncommon symbols) can be referred with a '\' before the symbol or have a quotation (").

File paths (2)



Lets illustrate Absolute path and Relative path.

-You wish to refer to a file named XYZ.txt in the school directory that is found in patrick folder
-You wish to refer to a file named LinuxAdmin.zip in the books folder under bills
-You wish to refer to the folder school in patrick folder

44

File paths

path is a sequence of nested directories with a file or directory at the end separated by the / character.

• Relative path: Relative to the current directory. Example: ./documents/fun/microsoft_jokes.html

• Absolute path: Starting from the Root directory.
Example: /home/bill/bugs/crash9402031614568

• / : root directory: Start of absolute paths for all files on the system (even for files on removable devices or network shared).

Command line interpreters

Shells: tools to execute user commands.

Called “shells” because they hide the details on the underlying operating system under the shell's surface.

Commands are input in a text terminal, either a window in a graphical environment or a text-only console.

Results are also displayed on the terminal. No graphics are needed at all.

Shells can be scripted: provide all the resources to write complex programs (variable, conditionals, iterations...)

46

File name pattern substitutions

Better introduced by examples!

ls *txt The shell first replaces *txt by all the file and directory names ending by txt (including .txt), except those starting with "." , and then executes the ls command line.

ls d.* Lists all the files and directories starting with . d

ls -d tells ls not to display the contents of directories.

cat ?.log

Displays all the files which names start by 1 character and end by .log

find . -name "*.pdf "

Search for a file that ends by .pdf

Special directories

./ - The current directory. Useful for commands taking a directory argument. Also sometimes useful to run commands in the current directory (see later), ./readme.txt and readme.txt are equivalent.

../ - The parent (enclosing) directory. Always belongs to the . Directory (see ls a).

Only reference to the parent directory.

Typical usage: cd ..

~/ - Shells just substitute it by the home directory of the current user. Cannot be used in most programs, as it is not a real directory.

for example ~sydney/ is substituted by shells by the home directory of the sydney user.

48

Copyright RT-ED.CO.IL 2020

The cd and pwd commands

cd <dir> - Changes the current directory to <dir>.

cd .. - Go up to parent directory.

cd - - Gets back to the previous current directory.

cd / - Goes to the root directory

pwd - Displays the current directory ("working directory").

Special Note:

If you want more options to each command, or you don't know, ask Linux! itself.

Use one of the following:

```
[command] --help
info [command]
man [command]
help [built-in shell command]    # for built-in commands only (see man bash)
```

The cp command

cp <source_file> <target_file> Copies the source file to the target.

cp file1 file2 file3 ... dir Copies the files to the target directory (last argument).

cp -i
interactive, Asks for user confirmation if the target file already exists

cp –r <source_dir> <target_dir> (recursive) Copies the whole directory.

50

mv and rm commands

mv <old_name> <new_name> (move)

Moves \ renames the given file or directory.

mv –i (interactive)

If the new file already exits, asks for user confirm

rm file1 file2 file3 ... (remove)

Removes the given files.

rm -i (interactive)

Always ask for user confirm.

rm –r dir1 dir2 dir3 (recursive)

Removes the given directories with all their contents.

Creating and removing directories

mkdir dir1 dir2 dir3 ... (make dir)

Creates directories with the given names.

rmdir dir1 dir2 dir3 ... (remove dir)

Removes the given directories

Safety: only works when directories are empty.

Alternative: rm -r (doesn't care if they are empty directories or not).

SSH - Secure SHell

• Used to access a remote computer on most Linux/Unix systems
(write shell commands just as if you were sitting at the workstations) just like telnet.

• Telnet method poses a danger in that everything that you send or receive over that session is visible in plain text so anyone can easily "sniff" the connection.

• Not only does it encrypt the session, it also provides better authentication facilities, as well as features like secure file transfer, X session forwarding, port forwarding and more .

Syntax:

```
ssh  user_name@computer_name
ssh  user_name@computer_ip_address
```

Windows:

use PuTTY for SSH

SCP - Secure Copy

- scp - remote file copy program, copies files between hosts on a network.
- uses ssh for data transfer authentication and provides the same security as SSH does.
- Unlike rcp (another network copy utility), scp will ask for passwords for authentication..
- Copies between two remote hosts are permitted.

Syntax:

```
scp  -r user_name@computer_name:/home/*  /home
```

Windows :

```
pscp.exe  bennyc@192.168.42.38:/home/minicom.log   c:
```


Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

Real Time Group

HTML

Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

55

Html 5

Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

Course Overview

HTML is a markup language for designing and writing content to load on our browsers. Furthermore it is the official and most distributed markup language around the world which makes it a skeleton of most websites and web applications.

H - T - M - L

Hyper - Text - Markup - Language

Hyper: The opposite of linear, meaning you can do whatever you want by interacting with links - actions doesn't have any order set.

Text: The content of the pages.

Markup: Is responsible for arranging and designing the text inside **HTML tags**.

Language: HTML is written with code-words and has its own syntax like any other programming language.

Course Overview

- Built to work on any system and environment.
- Very forgiving little errors.
- Allowed for use without any purchases or copyright to the creators.
- Allowing to create link to other documents and data.
- Supports media, allowing to display video, images and sound.
- Marked as the official internet language and supported by W3C.

Course Subjects

1. File structure and how to read it
2. Tags and how to work with them
3. Text
4. Titles
5. Links
6. Lists
7. Tables
8. Images
9. Video
10. Audio
11. File structure to groups
12. Inline styling
13. Browser dev tools

File structure and how to read it

The file is built by a group of tags.
Every line of the file has to be related to a tag.
Some tags has to be closed by its closing tag.
Example:

```
<html>
  <head>
    <title>My Special Title</title>
  </head>
  <body>
    <!-- Some more content here -->
    <script src="script.js"></script>
  </body>
</html>
```

File structure and how to read it

Every HTML file contain HTML tag.
HTML tag is divided to two groups - Head & Body.
Head: group will contain page settings, addons and outer links.
Body: group will contain the content of the page which will be showed to the visitors.

```
<html>
  <head>
    <title>My Special Title</title>
  </head>
  <body>
    <!-- Some more content here -->
    <script src="script.js"></script>
  </body>
</html>
```

Tags Table

Description	Tag
Declares the Web page to be written in HTML	<html> … </html>
Declares the head of the file configuring the page	<head> … </head>
Sets the title of the page	<title> … </title>
Sets page configuration option	<meta> … </meta>
Imports outsource file into the page	<link> … </link>
Declares the body containing the page content	<body> … </body>
Sets a Header type text element	<h1-6> … </h1-6>
Sets the selected text to **bold**	...
Sets the selected text to *italic*	…
Declares a paragraph to contain simple text input	...
Sets a new line between two lines	
Sets a horizontal line between two lines	<hr>

Tags Table

Description	Tag
Declares un-ordered list of elements	…
Declares ordered list of elements	…
Declares a list item	…
Declares a table	<table> … </table>
Sets a row of a table	<tr> … </tr>
Sets a column of a row	<td> … </td>
Places an image on the page	<img src=””>
Sets the selected text to a link	 …
Declares an area of work	…

Working with Tags

HTML file will be written in English - left to right.

Every line of code has to be written inside tags for HTML to read it.

Tag names defining the content of the tag for the browsers to know how to work with it and what to expect.

Tags contain the name of the command - <command> and a closing tag </command>

```
<p>
T        'Some' text' have more options to it and its content
         <a href="location">I'm a link</a>
</p>
```

Pay attention to the use of a tag inside a tag.

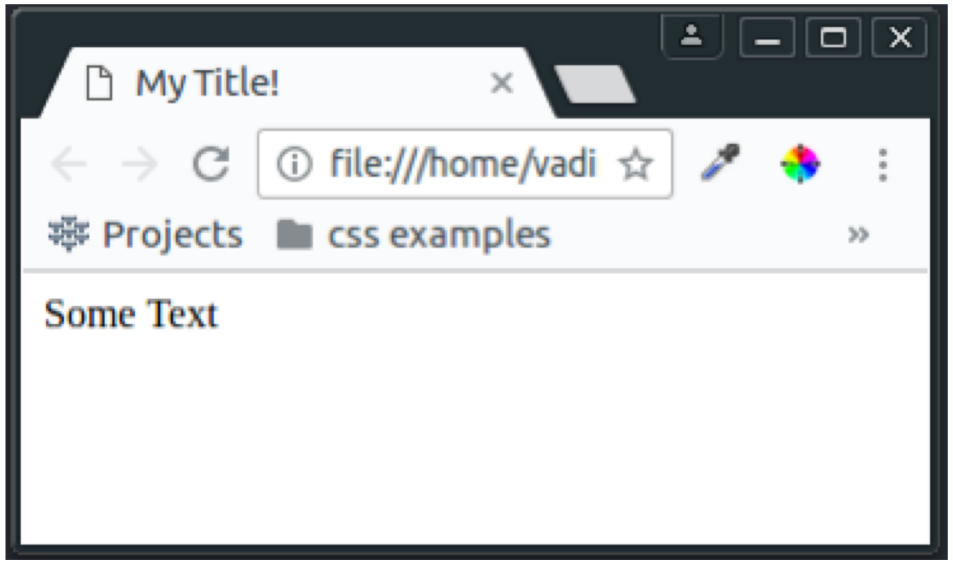
Don't forget to close the main tag after adding a content to it.

Working with Text and Titles

To add a text to the page you should just start typing inside most of the tags.

If the tag knows how to show a text it will. Otherwise it will ignore it.

```
<body>
        Some text
</body>
```

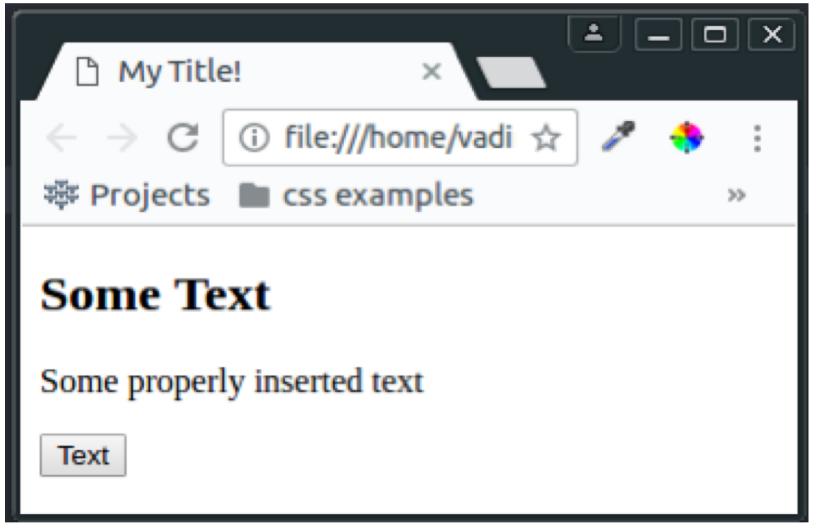


Even though it is possible to add text inside many tags, it is a good practice to write text inside tags representing a text.

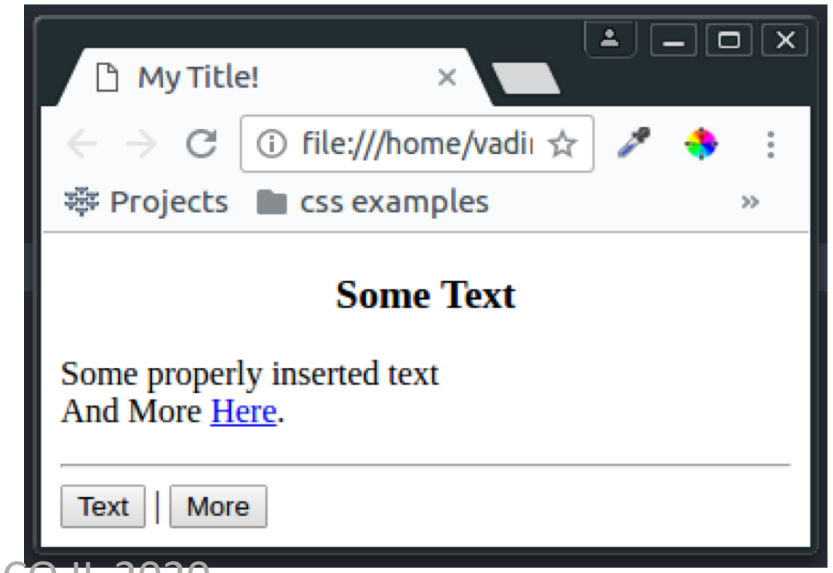
Working with Text and Titles

In the example you can see the use of a text as a title, a paragraph and a button.

```
<body>
        <h2>Some Text</h2>
        <p>Some proper text</p>
        <button>Text</button>
</body>
```




```
<body>
        <h2>Some Text</h2>
        <p>Some proper text</p>
        <button>Text</button>
</body>
```



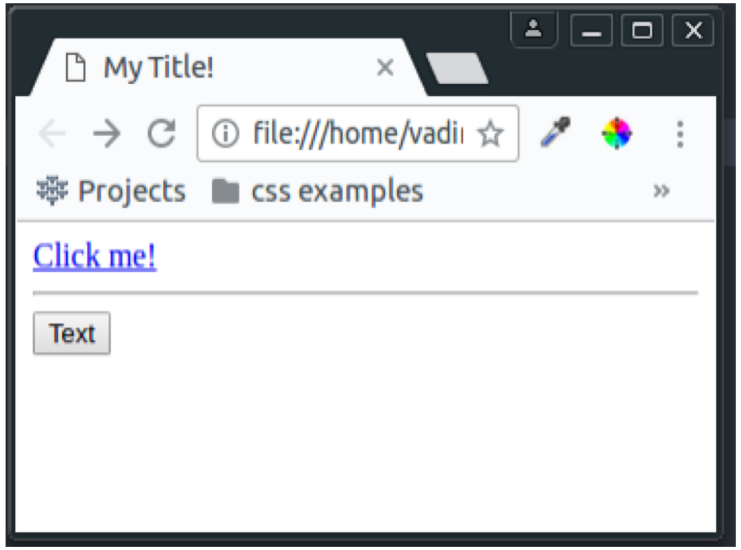
Working with links

Link is a pointer to other address and is used as a route to the address.

Usually links is used as a buttons which help us move around the page or to reach outer address.

Link is used as shown:

```
<a href="https://google.co.il"
target="_blank">Click Me</a>
<hr>
<button>Text</button>
```



Other than names, tags contain more options called properties.

In the example above we seen two new properties:

href="" - storing the new address for the link.

target="_blank" - sets to open the new address in a new page.

Working with links

Button is used as links that points to something too.
The difference is that buttons usually will handle some page event and interactions other that point to other address.
The difference is really not mandatory and we will learn soon that a button can also point to a new address as well as a link can trigger some event.

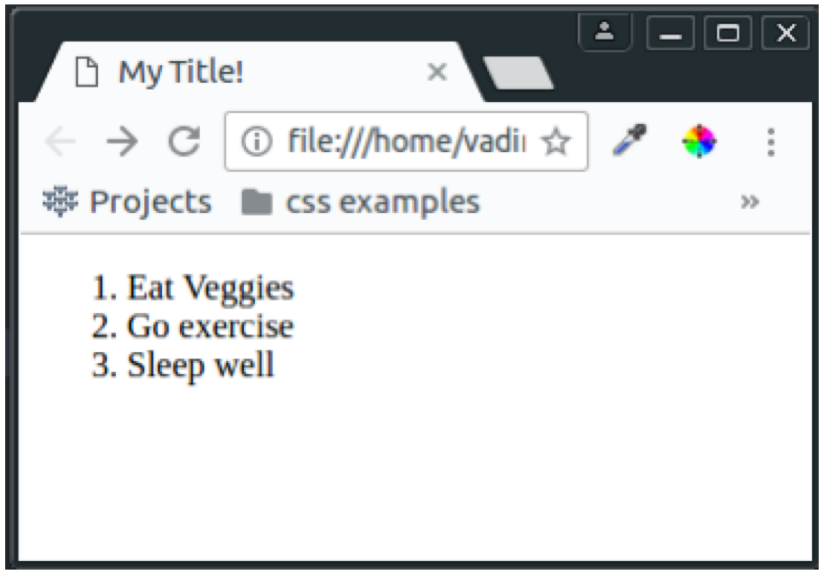
The importance here is to keep our code organized and to use the tags as intended by the creators.

Working with lists

Before adding a list to our page, we should decide it style.

We can choose two different styles, ordered list

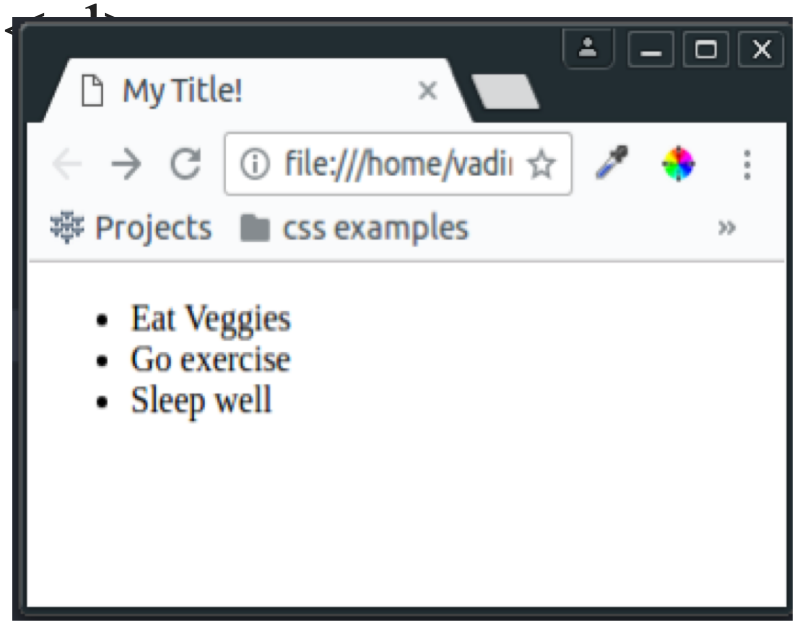
```
<ol>
	<li>Eat Veggies</li>
	<li>Go exercise</li>
	<li>Sleep well</li>
</ol>
```



Working with lists

Or un-ordered list where items has no order needed.

```
<ul>
      <li>Eat Veggies</li>
      <li>Go exercise</li>
      <li>Sleep well</li>
</ul>
```



Items of a list can contain any content like text, image or link.

List is widely used in HTML because of its nature to keep similar items in a group.

For example, you will see list used even for creating a menu.

Working with Tables

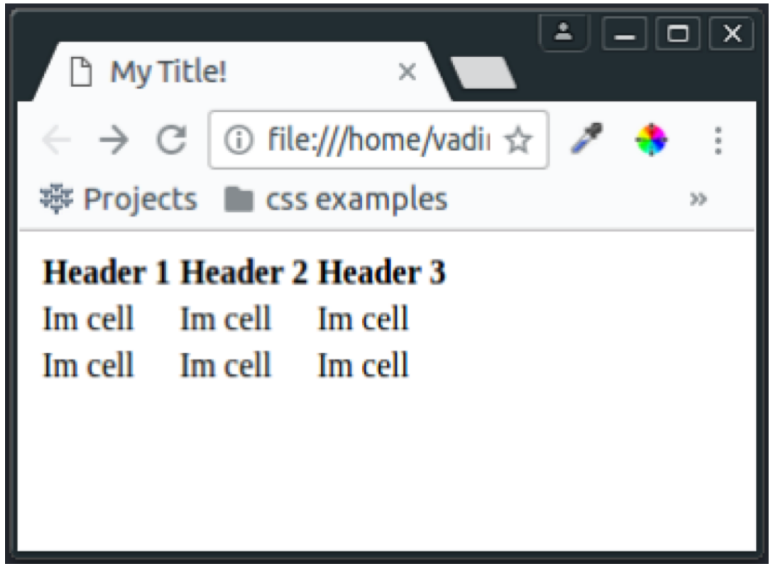
Tables are a very popular tag as well.
Its advantages is order of the content which is easily controlled to deliver your idea to the visitors.

Tables are divided to two groups. The top part called Head, and it contains the headers. The bottom part called body and contains the columns of content.
The body must be structured as rows, which will contain columns as a number of headers.

Table code Example: <table>

Working with Tables

```
<table>
        <thead>
                <tr>
                        <th>Header 1</th>
                        <th>Header 2</th>
          <tr>
        </thead>
        <tbody>
                <tr>
                        <td>im cell</td>
                        <td>im cell</td>
          </tr>
        </tbody>
</table>
```

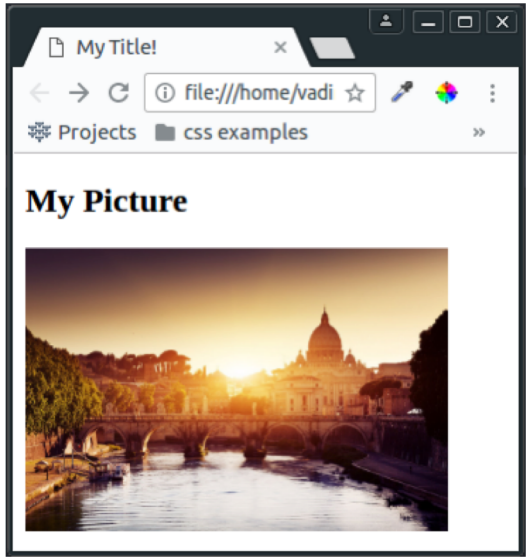


Working with images

Showing an image on a page requires a local file on your pc or a link to an image.

Image code Example <im src=””>

```
<h2>My Picture</h2>
<img src="image.png" width="300" height="200">
```



src=”” - Source - location of the image.

alt=”” - caption of the image - text to be shown when hovering over the image.

width=”” - the width of the image.

Height=”” - the height of the image.

Working with images

Images playing an important role of the website and it is important to know how to position it on the page.

Images can be used with many different tags.

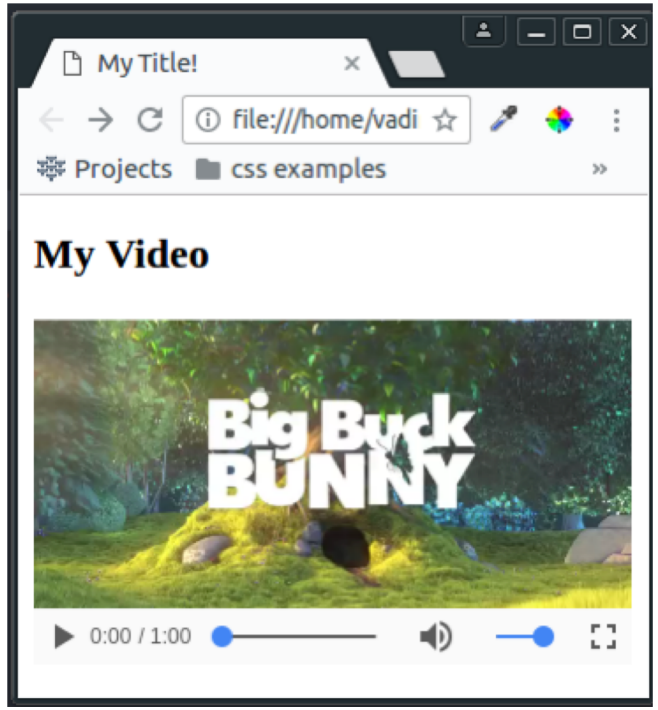
Working with video

Adding a video to the page has several tasks. First we will provide a space on a page where the image will be located. Then we will provide a temporary image while the video is loading. At last we will add the source of the image location.
We can also provide some text if the users browser cant load the video.

Video as an image can be placed in many different tags.
Defining the width and height of the video is very important!

Working with video

```
<h2>My Video</h2>
<video>
      <source src="video.mp4" type="video/mp4">
      Your browser does not support video tag.
</video>
```

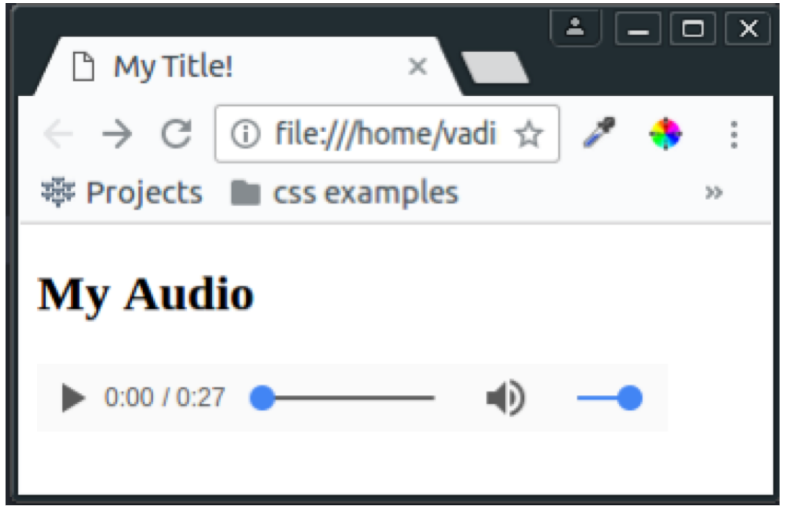


<video> - setting space for the video.
width="" - width of the video space.
height="" - height of the video space.
<source> - setting up the video tag.
src="" - Source - location of the video file.
Text - Text for older browsers.

Working with audio

Audio files is inserted to the page exactly like the video files.

```
<h2>My Audio</h2>
<audio>
      <source src="audio.mp3" type="audio/mp3">
      Your browser does not support audio tag.
</audio>
```



Video notes working the same here.

File structuring to groups

HTML file on one side can be very simple and easy to read but on the other hand in bigger projects may become very long and without clear order can confuse, produce errors and not work as expected.

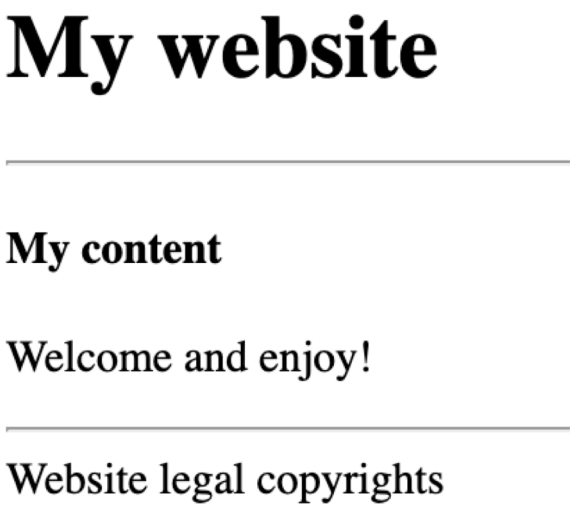
There is many ways to organize your pages and each developer chooses his favorite or according to the project needs.

There is some famous libraries that help us to divide the content correctly by controlling the content in a grid.

File structuring to groups

The basic division coming from html look likes as the following:

```
<header>
   <h1>My website</h1>
   <hr>
</header>
<main>
     <h4>My content</h4>
        <p>Welcome and enjoy!</p>
</main>
<footer>
   <hr>
   Website legal copyrights
</footer>
```



File structuring to groups

A semantic element clearly describes its meaning to both the browser and the developer.
We can see three new semantic tags here that divide our content:
<header> , <main> , <footer>.
There is no difference in between them but their names for clarity.

The most widely used non-semantic tag is the . Remember it - it is now your best bud.
Usually you will want to wrap your content in groups to better control the content displayed on your page allowing you to control and play with the inside content without hurting the main structure of the page.

File structuring to groups

<header> - Tag that specifies a header for a document or section.
It's should be used as a container for introductory content.

<main> - Tag that specifies a main for a document or section.
It's should contain unique content of the document and not any content that is repeated across documents such as sidebars, navigation link etc.

<footer> - Tag that specifies a footer for a document or section.
Element should contain information about its containing element.

 - Tag that defines a division or a section in an HTML document.
Usually used as a container of other elements.
You can use div instead of <header>,<main>,<footer> but not recommend

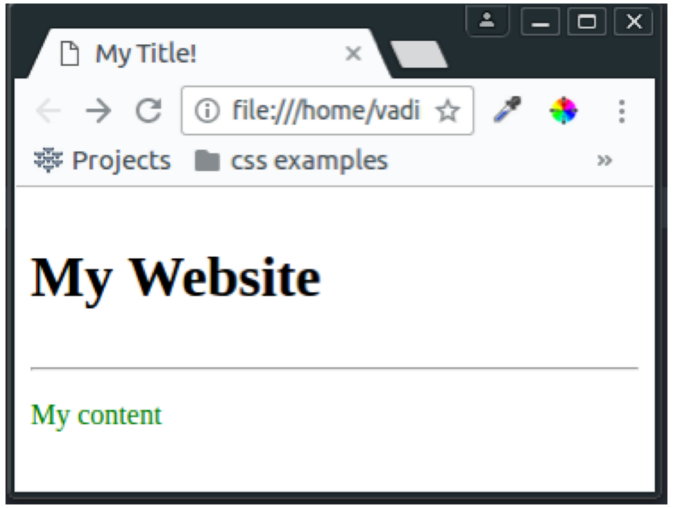
Inline styling

Html file is the skeleton of your page and contains its content and markup.

Designing the page coming next, and there is two ways you can use.

1. Inline style - individual styling for the specified element.

```
<header>
        <h3>My Website</h3>
        <hr>
        <p style=""color: green;"">My content</p>
</header>
```



2. Out-source style file which we add to our html page to control our styling.

We will learn that soon.

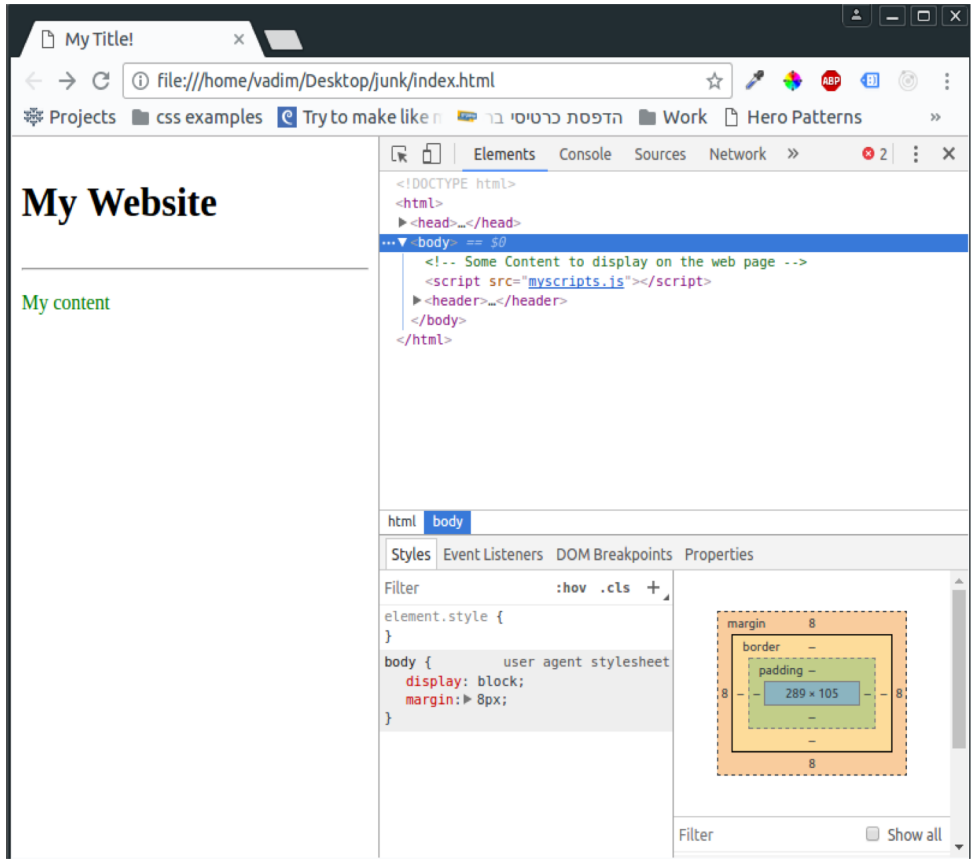
With the help of CSS styling, we can style, organize and change how the content looks.

Styling can affect colors, sizes, borders, background and more.

We will learn that soon.

Browser dev tools

Html language passed a long way to become what it is now.
Along with it browser creators worked very hard as well to achieve the best result available and developed tools to help us work with HTML in convenience.
One such tool is very important and called dev tools which is located in most modern browsers which can be reached with right click on the page and choosing inspect or by pressing Ctrl + Shift + i.



Copyright RT-ED.CO.IL 2020

Browser dev tools

On the left side you can see your content, on the right side there is many tools available for us to use.

- Elements tab will contains all the elements of your page where you can edit them and their style to see immediate results.
- Console tab will print all the logs of your website if there any errors or logs you asked to print there. You can also use javascript from here to interact with the page.
- Source tab will contain all the files downloaded to the browser and edit them as well.
- Network tab will show you all the traffic of your website like downloads and its time.

As a web developer it is very important to know how to use those tools to achieve best results and debug your pages.

Extra tags

DOCTYPE

The <!DOCTYPE html> declaration is used to inform a website visitor's browser that the document being rendered is an HTML document. While not actually an HTML element itself, every HTML document should being with a DOCTYPE declaration to be compliant with HTML standards.

The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.

The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

<!--...--> Comment

The comment tag is used to insert comments in the source code. Comments are not displayed in the browsers.

You can use comments to explain your code, which can help you when you edit the source code at a later date. This is especially useful if you have a lot of code.

```
<!--This is a comment. Comments are not displayed in the browser-->

<p>This is a paragraph.</p>
```

Map & Area

The <map> element is used in conjunction with one or more <area> elements to define hyperlinked regions of an image map.

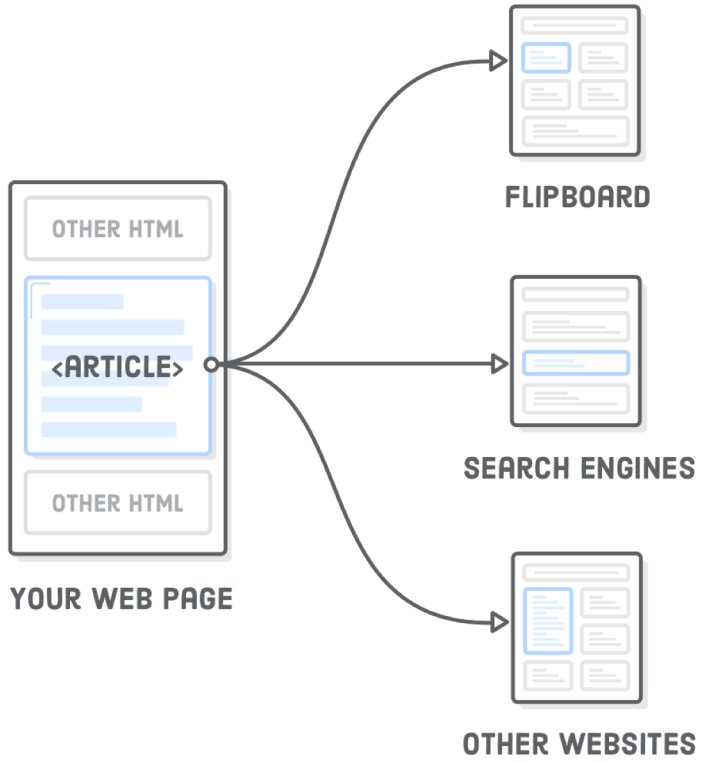
The <area> element is used as a child of a <map> element to define clickable a region on an image map. Different regions of an image map can be hyperlinked to different locations by nesting multiple <area> elements in a single <map> element.

Note: The usemap attribute in the <img> tag is associated with the <map> element's name attribute, and creates a relationship between the image and the map.

Article

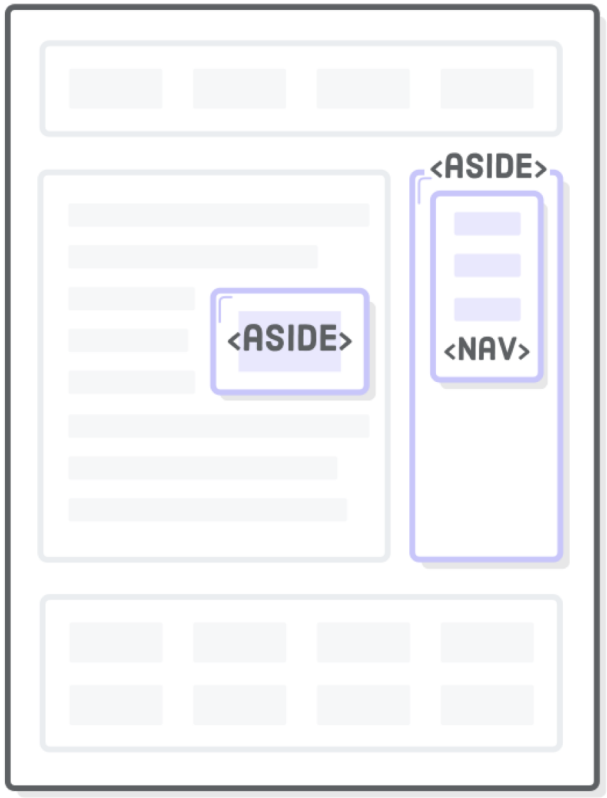
The HTML <article> element represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication).

Examples include: a forum post, a magazine or newspaper article, or a blog entry.



Aside

The <aside> element is used to identify content that is related to the primary content of the webpage, but does not constitute the primary content of the page. Author information, related links, related content, and advertisements are examples of content that may be found in an aside element.



B

The HTML Bring Attention To element (<b>) is used to draw the reader's attention to the element's contents, which are not otherwise granted special importance. This was formerly known as the Boldface element, and most browsers still draw the text in boldface. However, you should not use <b> for styling text; instead, you should use the CSS font-weight property to create boldface text, or the element to indicate that text is of special importance.

Base

The HTML <base> tag is used to specify a base URI, or URL, for relative links.

For example, you can set the base URL once at the top of your page in header section, then all subsequent relative links will use that URL as a starting point.

<base href="Url" target="target"> (target Specifies the default target for all hyperlinks and forms in the page)

Target options:

_blank

_parent

_self

_top

Blockquote

What does <blockquote> HTML Tag do?

The <blockquote> element defines a block of text that is a direct quotation. The <quote> element should be used when a quotation is presented inline with the surrounding text, but when the quotation is presented as a separate paragraph, <blockquote> is the appropriate element to use to identify the quotation.

Button

The HTML <button> element represents a clickable button, which can be used in forms or anywhere in a document that needs simple, standard button functionality. By default, HTML buttons are typically presented in a style similar to that of the host platform the user agent is running on, but you can change the appearance of the button using CSS.

<button type="button" onclick="alert('Hello world!')">Click Me!</button>

Canvas

HTML5 has brought some exciting new advantages to the HTML coding world. Canvas allows you to render graphics powered by Javascript. So throw away that flash code and dive into Canvas. Here you will find the best tutorials and resources to learn Canvas and other HTML5 aspects.

```
<canvas id="myCanvas"></canvas>


<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0, 0, 80, 80);
</script>
```

Caption

The <caption> element is used to add a caption to an HTML table. A <caption> must appear in an HTML document as the first descendant of a parent <table>, but it may be positioned visually at the bottom of the table with CSS.

Datalist

The HTML <datalist> element contains a set of <option> elements that represent the values available for other controls.

```
<form action="/action_page.php" method="get">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit">
</form>
```

Details

The HTML Details Element (<details>) creates a disclosure widget in which information is visible only when the widget is toggled into an "open" state. A summary or label can be provided using the <summary> element.

A disclosure widget is typically presented onscreen using a small triangle which rotates (or twists) to indicate open/closed status, with a label next to the triangle. If the first child of the <details> element is a <summary>, the contents of the <summary> element are used as the label for the disclosure widget.

Form

The HTML <form> element represents a document section that contains interactive controls for submitting information to a web server.
<form action="page url" target="target" method="GET/POST"></form>

action

The URI of a program that processes the form information. This value can be overridden by a formaction attribute on a <button> or <input> element.

method

The HTTP method that the browser uses to submit the form. Possible values are:

post: Corresponds to the HTTP POST method ; form data are included in the body of the form and sent to the server.

get: Corresponds to the HTTP GET method; form data are appended to the action attribute URI with a '?' as separator, and the resulting URI is sent to the server. Use this method when the form has no side-effects and contains only ASCII characters.

dialog: Use when the form is inside a <dialog> element to close the dialog when submitted.

Form

target

A name or keyword indicating where to display the response that is received after submitting the form. In HTML 4, this is the name/keyword for a frame. In HTML5, it is a name/keyword for a browsing context (for example, tab, window, or inline frame). The following keywords have special meanings:

_self: Load the response into the same HTML 4 frame (or HTML5 browsing context) as the current one. This value is the default if the attribute is not specified.

_blank: Load the response into a new unnamed HTML 4 window or HTML5 browsing context.

_parent: Load the response into the HTML 4 frameset parent of the current frame, or HTML5 parent browsing context of the current one. If there is no parent, this option behaves the same way as _self.

Form

_top: HTML 4: Load the response into the full original window, and cancel all other frames. HTML5: Load the response into the top-level browsing context (i.e., the browsing context that is an ancestor of the current one, and has no parent). If there is no parent, this option behaves the same way as _self.

iframename: The response is displayed in a named <iframe>.

HTML5: This value can be overridden by a formtarget attribute on a <button> or <input> element.

Iframe

The HTML Inline Frame element (<iframe>) represents a nested browsing context, embedding another HTML page into the current one.

<iframe src="https://www.google.com"></iframe>

Img

The <img> tag defines an image in an HTML page.

It’s has two required attributes: src and alt.

```
<img src="img.png" alt="img">
```

Input

Type Submit create button that send the form to the server.
Type checkbox create check box.
<input> elements of type radio are generally used in radio groups—collections of radio buttons describing a set of related options. Only one radio button in a given group can be selected at the same time. Radio buttons are typically rendered as small circles, which are filled or highlighted when selected.

Input text

<input> elements of type text create basic single-line text fields.

The maximum number of characters the input should accept	maxlength
The minimum number of characters long the input can be and still be considered valid	minlength
A regular expression the input's contents must match in order to be valid	pattern
An exemplar value to display in the input field whenever it is empty	placeholder
A Boolean attribute indicating whether or not the contents of the input should be read-only	readonly
A number indicating how many characters wide the input field should be	size
Controls whether or not to enable spell checking for the input field, or if the default spell checking configuration should be used	spellcheck
Name the tag to control it with Javascript	Id, name

Label

Tag that defines a label for elements such as <button>, <input>, <select> etc.

This element does not render as anything special for the user.

Meta

The <meta> tag is a tag in the <head> section, that contain metadata about the document.

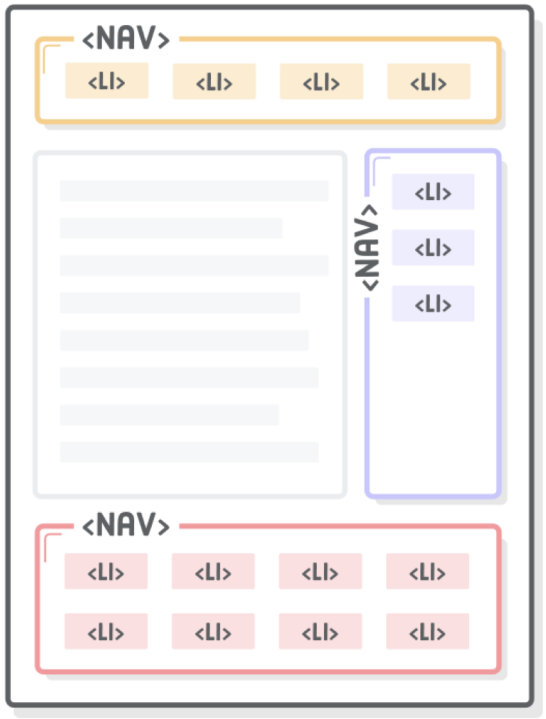
The metadata will not be displayed on the page.

Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

Nav

The <nav> tag is a semantic element that defines a set of navigation links.

The <nav> element is intended only for major block of navigation links.



Option and Select

The <select> element is used to create a drop-down list.

In the <select> tag we have <options> tags that define the available options in the list

```
<select>
        <option value="option1">option1</option>
          <option value="option2">option2</option>
 </select>
```

Progress

The <progress> tag represents the progress of a task.

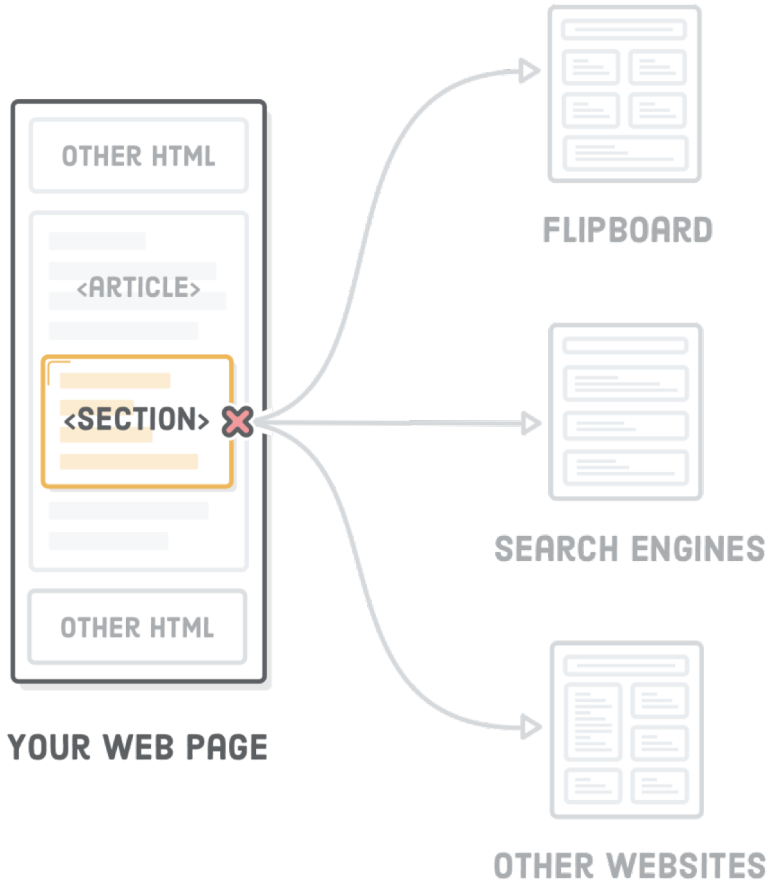
It needs value and max attributes.

```
<progress value="22" max="100"></progress>
```

Section

The HTML <section> element define a section in a document.

Section can be chapters, headers, footers etc.



Svg

The <svg> tag defines a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Textarea

The HTML <textarea> element represents a multi-line plain-text editing control, useful when you want to allow users to enter a sizeable amount of free-form text, for example a comment on a review or feedback form.

Title

The <title> tag is a tag in the <head> section that required in all HTML documents and it defines the title of the document.

It isn’t shown in the document itself but in the browser tab, bookmark etc.

U

The HTML Unarticulated Annotation Element (<u>) represents a span of inline text which should be rendered in a way that indicates that it has a non-textual annotation. This is rendered by default as a simple solid underline, but may be altered using CSS.

Hope you enjoy

Real Time Group

CSS

CSS3 & Bootstrap

Copyright RT-ED.CO.IL 2020

Copyright RT-ED.CO.IL 2020

Course Overview

CSS is the next technology you should learn after HTML.
CSS is a cascading driven meaning it will run from top to bottom and may override values if they match same element.
It's use is to style and layout your HTML documents.
For example to change font size/color, arrange the content, add animations and other decorative features.

C - S - S

Cascading - Style - Sheets

Cascading: Reads the code from top to bottom and apply last known value.
Style-Sheet: List of properties and values to be applied.

Course Overview

- Runs on any system and environment
- Forgiving nature of errors
- Works alongside and only with HTML
- Allowing to import external media to use on the page

Course Subjects

1. How to read a CSS File
2. Under the hood
3. Common css property values
4. Advanced selectors
5. Pseudo classes
6. Transitions
7. Animations
8. Bootstrap introduction
9. Common properties
10. Grid system
11. Working with Bootstrap

CSS file and selection types

CSS File is very simple stylesheet in form of - property = value.

Properties may be general like tag names, or more specific classes and ids set to elements by HTML.

You should read the file from top to bottom, and make sure you are not overriding same element value. Overriding same property values is resource wasting approach and is not recommended.

```
Body {
Background: gray;
}
.greenButton {
Color: blue;
}
#myButton {
Font-size: 18px;
}
```

CSS file and selection types

In the example above we see three different selectors:

Body - is a name of a tag, meaning it will apply the value on any element with this tag name anywhere on the page.

.greenButton - is a class name placed on an element by HTML, meaning it will apply the value on any element with this class name.

#myButton - is an id placed on an element by HTML, meaning it will apply the value on that specific element only.

Rule - Assign an id on one element only. Id is made to specify a very specific element on your page.

Rule - Class names are more general than ids, you can set them on many elements making them kind of a family to set values to all at once.

Rule - general tag name working exactly like class names, but with less of importance. We will learn about it soon.

Under the hood

CSS has some rules we must know to properly work with it.
Working without understanding those rules may affect your work with errors or unwanted behavior and may be hard to track.

Cascading is the operation method of CSS. What it says it will read the css file from top to bottom and may override values if set on same element.
For example if i will set:

```
.greenButton {
      Color: blue;
}
.greenButton {
      Color: green;
}
```

Under the hood

CSS will read the file from top and will set the color of .greenButton element to blue.
Next it will read the second property and will set the color of same element to green.
Meaning the color of the element on the page will be green.
In general we want to prevent such operations to reduce the size of our css file and to increase load time of our pages. Moreover such actions may lead to unwanted behavior and may be ready hard to track.

```
.greenButton {
        Color: blue;
}
.greenButton {
        Color: green;
}
```

Under the hood

Calculating Specificity is very important and many may not know about it.

The idea behind it is to prioritize different selector to give them more weight than others.
The rule is the following:

General tag name: 0 - 0 - 1
Class Name: 0 - 1 - 0
Id: 1 - 0 - 0

Example: <button class=”btn”></button>
Meaning That if we will set color value on a tag name to be red and then set color value to be green on a class name on the same element - it will be green because classes has more weight than tag names.

Under the hood

Calculation:

We have two paragraphs in the example above, let's calculate them:

html:

```
<div>
  <p>some text</p>
  <p class="special">some text</p>
</div>
```

The first property has value of: 0 - 0 - 2
Each tag name gives us 0 - 0 - 1.

The second property has value of: 0 - 1 - 2
Two tag names each 0 - 0 - 1
and one class 0 - 1 - 0

css:

```
Div P {
    Color: red;
}
Div p.special {
    Color: green;
}
```

Under the hood

Both are affecting the the .special paragraph but the second styling has more weight and will be the one in charge of that element style.

That way we can control which element will get what value, and differ between elements.

One good example may be a menu of a website, where we have multiple links to different pages, and only one will have an .active class which will have different style to show as the active page right now.

Under the hood

Three dimensions:

CSS has three dimensions layout meaning each element is set by horizontal, vertical and z-axis and are formatted one on top the other.

Z-axis positions are relevant when elements overlap visually - When two element positioned on overlapping horizontal and vertical position, z-axis will decide which will be on top of the other.

For Example:

```
<body>
  <div class=”top”></div>
  <div class=”bottom”></div>
</body>
```

```
.top {
    Background: red;
}
.bottom {
    Background: green;
}
```

Under the hood

Result:

Simple positioning: two element placed one after the other.

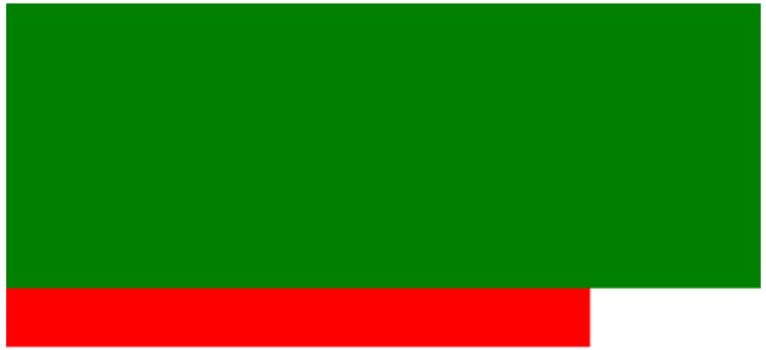
This is the regular case with html element - they will be placed from top to bottom each one after the other.

We may make them to be aligned horizontally but still they won't overlap each other.

Under the hood

Now we will make them overlap one on top of the other:

```
.top {
Position: absolute;
Background: red;
}
.bottom {
Position: absolute;
Background: green;
}
```



Under the hood

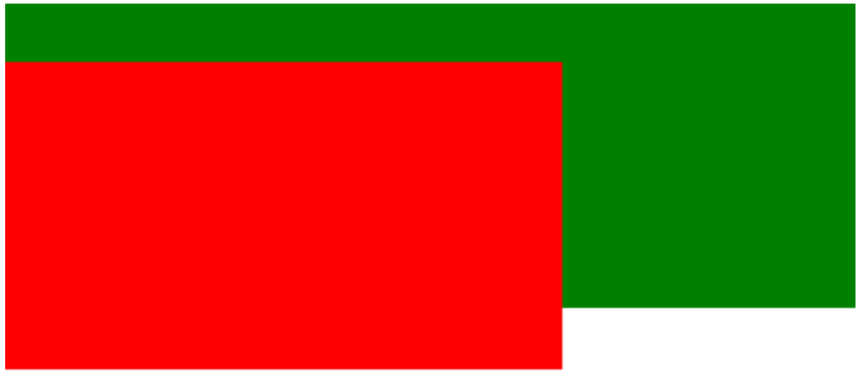
We will talk about positions soon, but for now look at how the elements overlap.
Adding absolute position make the element to overlap one on top of the other.
Without z-index the bottom (green) element is placed on top (red) element because it was rendered after the red element.

We have placed both elements on overlapping horizontal and vertical positions, but we didn’t change their z-axis position leaving them both on same layer.

Under the hood

Let's change that:

```
.top {
      Position: absolute;
      Background: red;
      Z-index: 2;
}
.bottom {
      Position: absolute;
      Background: green;
}
```



Under the hood

Adding z-index to the red element, we now placed it in higher level of z-axis layer which brings it closer to the observer meaning it will be seen before other elements.

Now you should know that whenever you need to work with overlapping element, you should place them on the z-axis accordingly to your design plan for the elements to show right.

These three rules are very IMPORTANT and you should check yourself over the first time that you are working by the rules and not the other way around.
These rules are a mandatory part of every project you will develop next.

Common properties

CSS has many different properties, so many that some of us may not even use all of them.
Lets list the properties that is widely used by most developers.

Property	Explanation
Color - general name, RGBA, HSLA, Hexadecimal	Sets a color of the specified element text content. Example: color: red / #fff / rgb(0,0,0)
Length - absolute & relative	Sets the width and height of the specified element. Absolute length is set with simple numbered size like: pixels (30px) Relative length is set by setting the requested amount relatively to other elements. (50%)
Background - color, gradient & image	Sets the background of an element. Background: green / #000 Background: linear-gradient(red, yellow) Background: url('http://….')

Common properties

Property	Explanation
Border & Outline	Sets the border/outline of an element. Border: solid/double/dashed 1px red outline: solid/double/dashed 1px red
position	Sets the position of an element Position: static/relative/absolute/fixed
Font - size & family	Sets the size, color and family of text elements. Font-size: 30px Font-family: Arial

Advanced selectors

CSS is offering some more advanced selectors to uniquely select the requested elements. Instead of selecting only by tags, classes and ids we can target specific elements with attributes. This way we can choose element by their values and types for example.

Selector usage:

[attr] { color: red }

[attr=value] { color: red}

Advanced selectors

```
<input type="text">
<input type="email">
<input value="name">
<input placeholder="your name">
<input data-special="my data">
```


```
[type] {
    Background: red;
}
[type="text"] {
    Background: green;
}
[value="name"] {
    Background: orange;
}
[placeholder] {
    Background: purple;
}
```

Advanced selectors

As you can see, we can target elements by their attribute name or attribute values.
Good Example may be when you create a form, and you want to show the user that he didn’t insert a correct value to some input field - you can target this element by wrong value and color it with red.

Pseudo classes

CSS is offering some more advanced selectors to give us full control of the page.
Pseudo classes control elements by specific state of the element.
For example when a visitor will mouse over your element, the document will see it and by the css pseudo class will know what you want it to do.
Using pseudo classes can seriously increase your control over the document and allow you full freedom of imagination.

Pseudo classes

We are targeting the first input element by its placeholder text and changing its color to orange. Now every text input element with placeholder will have orange text.
The second target is an element which has a mouse over it. When the page see's we are hovering over the element , it will change its background to gray.
Remember, you can target elements by any attribute and their value.
With the usage of pseudo classes we can achieve way better user experience and step further in an ultimate website.

Lets try an example:

```
[type="text"]::placeholder {
    Color: orange;
}
Input:hover {
    Background: gray;
}
```

Transitions

Transitions are our way to animate changes on the fly with CSS.
When we want that some change will occur on the page, we can also make it look much nicer.
Although it makes the changes look beautiful, it has a very important role as well - the visitor of our page will be able to notice the change. This way we can know that we are not losing our visitors, and that they are fully understand what is happening on our pages.

For example, we will use the last input with mouse over pseudo effect but this time we will add a transition so it will become noticeable.

Without transitions:

Without transitions:

Transitions

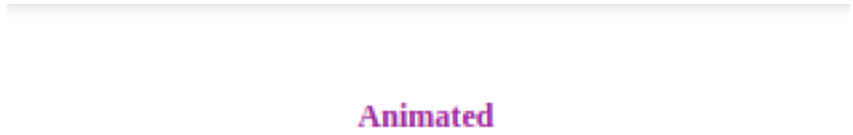
As you can see the background change is slower and therefore noticeable.
In this case i have used very dark color and probably you would not miss the change without the transition. In real project, you may find way more usages of mouse or other events, and the changes may occur on some other part of the page and may be harder to see.

The use of transitions is not mandatory, but as I stated before, it has a very important role in the world of websites.
Developers who respect the user experience will gain more traffic and positive reviews.

Animations

Alongside the transitions, CSS offering us full animations. Animations is more of a groups of transitions that start right after the other. This way we can create animation like results. Creating animations tho has some more advanced techniques and requires some deeper learning.

To achieve that result we had to create three different transitions as follows:



Let's start by Example:

```
@keyframes jump {
    From {
            Font-size: 12px;
     }
     50% {
            Font-size: 48px;
     }
     100% {
            Font-size: 12px;
     }
}
```

Animations

Each group defines the state of the element at a relative point of time in our animation.
Starting from small text size - growing at the middle of the animation to bigger text and shrinking back to mall text at the end.
To add this animation to an element we should provide some more information:

```
.centered h1 {
  Color: purple;
  Animation-name: jump;
  Animation-delay: 1s;
  Animation-duration: 2s;
  Animation-fill-mode: forwards;
 Animation-iteration-count:
infinite;
}
```

Animations

name: is the name we defined when creating the keyframes.
delay: time before the animation will start.
Duration: time it will take to the animation to complete.
Fill-mode: defines the value outside the animation. You can choose if to keep the last value of the animation or the first one.
Iteration-count: how many times the animation will run.

Building the animation keyframes may be straightforward for simples animations, but can become harder with a complex animation.

Animation format is as follows:
From: the initiate state of the element before the animation.
%: state of the element at the given percentage.
To: the last state of the element after the animation.

Flex

Most importantly, the flexbox layout is direction-agnostic as opposed to the regular layouts (block which is vertically-based and inline which is horizontally-based). While those work well for pages, they lack flexibility (no pun intended) to support large or complex applications (especially when it comes to orientation changing, resizing, stretching, shrinking, etc.).

Note: Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

Flex

The Flexbox Layout (Flexible Box) module (a W3C Candidate Recommendation as of October 2017) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word "flex").

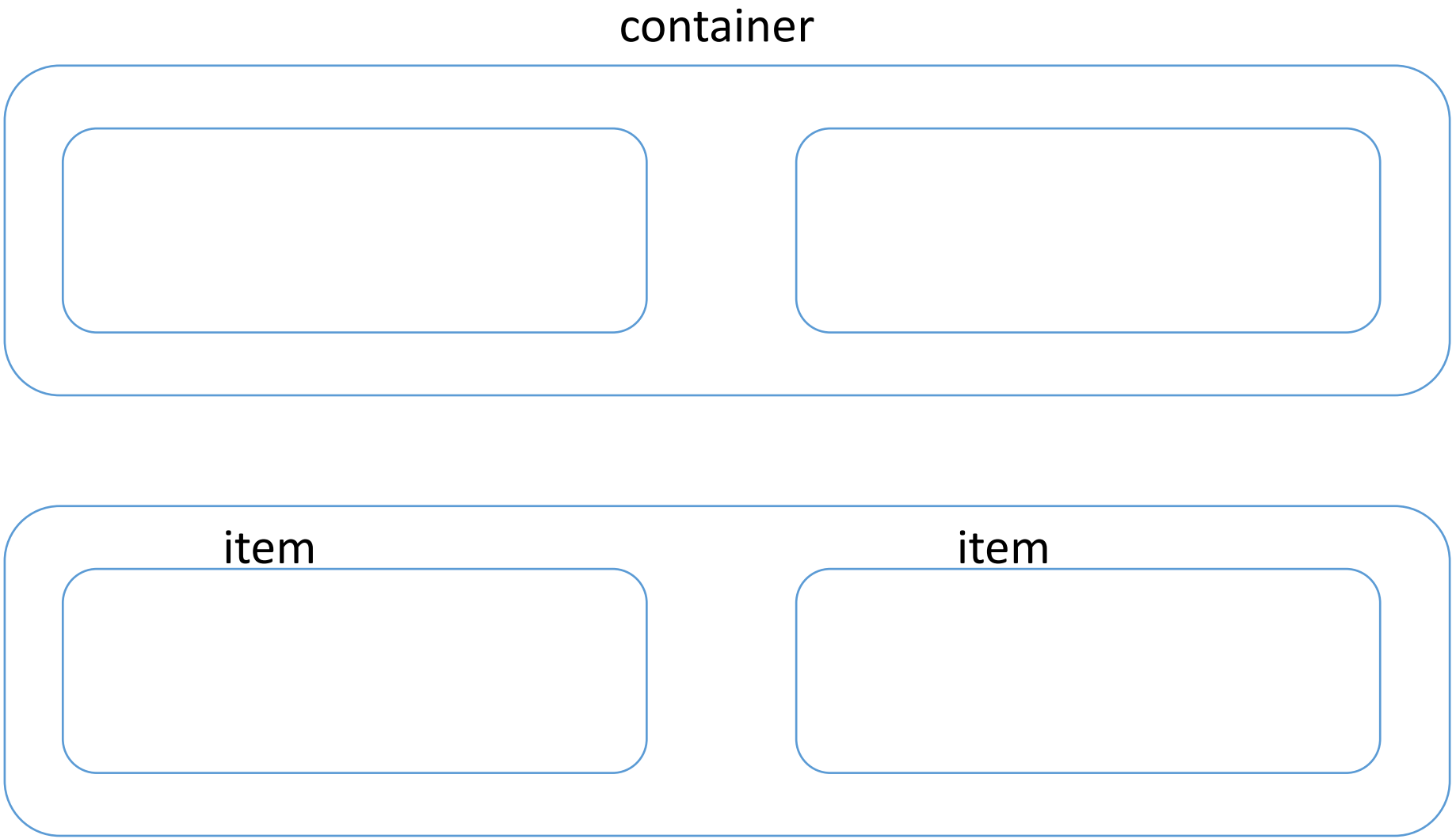
The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

Flex

The Flexbox Layout (Flexible Box) module (a W3C Candidate Recommendation as of October 2017) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word "flex").

The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

Flex



Flex

display

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

```
.container {
  display: flex; /* or inline-flex */
}
```

Note that CSS columns have no effect on a flex container.

Flex

order

By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in the flex container.

```
.item {
order: <integer>; /* default is 0 */
}
```

Flex

flex-direction

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

- row (default): left to right in ltr; right to left in rtl
- row-reverse: right to left in ltr; left to right in rtl
- column: same as row but top to bottom
- column-reverse: same as row-reverse but bottom to top

Flex

flex-grow

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, the remaining space would take up twice as much space as the others (or it will try to, at least).

```
.item {
  flex-grow: <number>; /* default 0 */
}
```

Negative numbers are invalid.

Flex

flex-wrap

two rows of boxes, the first wrapping down onto the second

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

```
.container{
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

nowrap (default): all flex items will be on one line
wrap: flex items will wrap onto multiple lines, from top to bottom.
wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

Flex

flex-shrink

This defines the ability for a flex item to shrink if necessary.

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

Negative numbers are invalid.

Flex

flex-flow (Applies to: parent flex container element)

This is a shorthand flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. Default is row nowrap.

flex-flow: <‘flex-direction’> || <‘flex-wrap’>

Flex

flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The auto keyword means "look at my width or height property" (which was temporarily done by the main-size keyword until deprecated). The content keyword means "size it based on the item's content" - this keyword isn't well supported yet, so it's hard to test and harder to know what its brethren max-content, min-content, and fit-content do.

```
.item {
  flex-basis: <length> | auto; /* default auto */
}
```

If set to 0, the extra space around content isn't factored in. If set to auto, the extra space is distributed based on its flex-grow value.

Flex

flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The auto keyword means "look at my width or height property" (which was temporarily done by the main-size keyword until deprecated). The content keyword means "size it based on the item's content" - this keyword isn't well supported yet, so it's hard to test and harder to know what its brethren max-content, min-content, and fit-content do.

```
.item {
  flex-basis: <length> | auto; /* default auto */
}
```

If set to 0, the extra space around content isn't factored in. If set to auto, the extra space is distributed based on its flex-grow value.

Flex

justify-content

This defines the alignment along the main axis. It helps distribute extra free space left over when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

```
.container {
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;
}
```

Flex

justify-content

- flex-start (default): items are packed toward the start line
- flex-end: items are packed toward to end line
- center: items are centered along the line
- space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
- space-around: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
- space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

Flex

Flex-start

Flex-end

center

Flex

Space-between

Space-around

Space-evenly

Flex

flex

This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. Default is 0 1 auto.

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

It is recommended that you use this shorthand property rather than set the individual properties. The short hand sets the other values intelligently.

Flex

align-self

This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

Please see the align-items explanation to understand the available values.

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

Note that float, clear and vertical-align have no effect on a flex item.

Flex

align-items

This defines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

```
.container {
  align-items: flex-start | flex-end | center | baseline | stretch;
}
```

- flex-start: cross-start margin edge of the items is placed on the cross-start line
- flex-end: cross-end margin edge of the items is placed on the cross-end line
- center: items are centered in the cross-axis
- baseline: items are aligned such as their baselines align
- stretch (default): stretch to fill the container (still respect min-width/max-width)

Flex

align-content

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

Note: this property has no effect when there is only one line of flex items.

```
.container {
  align-content: flex-start | flex-end | center | space-between | space-around |
stretch;
}
```

Flex

align-content

- flex-start: lines packed to the start of the container
- flex-end: lines packed to the end of the container
- center: lines packed to the center of the container
- space-between: lines evenly distributed; the first line is at the start of the container while the last one is at the end
- space-around: lines evenly distributed with equal space around each line
- stretch (default): lines stretch to take up the remaining space

Bootstrap

Bootstrap introduction

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery. *from getbootstrap.com*

As stated by bootstrap, you must include jQuery in your project if you want to use their toolkit.

BS is very famous and one of the best toolkits available right now, and is widely used. If you in need to achieve faster results, have less css skills or you want to keep your css clean and stable you should consider using this toolkit.

You can download or use their cdn from https://getbootstrap.com

Bootstrap introduction

To import, add a link at the head of the page like so:

```
<html>
  <head>
    <title>My Website</title>
    <link rel=”stylesheet” href=”./bootstrap.min.css”>
    <script src=”./jquery.min.js”></script>
    <script src=”./bootstrap.min.js”></script>
  </head>
</html>
```

Common properties

CSS is a toolkit meaning it provides us a set of tools to use. Lets see some most used properties which they call components.

Containers are the most basic layout element in Bootstrap and are required when using our default grid system. Choose from a responsive, fixed-width container (meaning its max-width changes at each breakpoint) or fluid-width (meaning it’s 100% wide all the time).

Common properties

Buttons nicely designed buttons to use for document events.

Usage:

```
<a class="btn btn-primary">Im a link</a>
<a class="btn btn-danger">Im a link</a>
```

Result: BS vs regular link

im a link im a link

im a link im a link

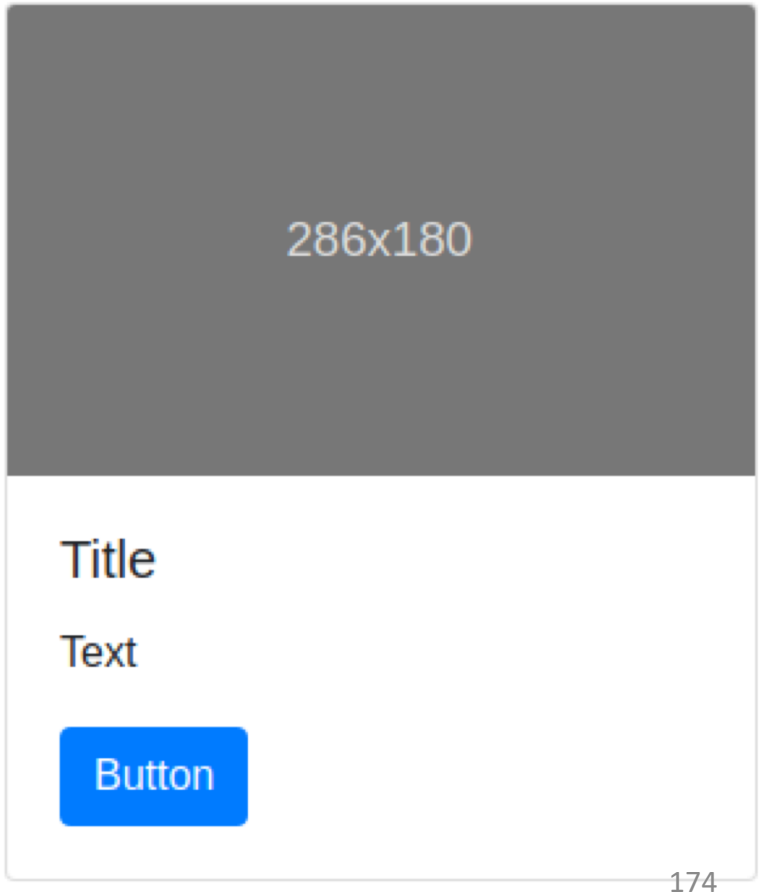
Common properties

Card is a grouped content of title, image, text and a button.

Usage:

```
<div class="card">
  <img class="card-img-top" src="...">
  <div class="card-body">
    <h5 class="card-title">Title</h5>
    <p class="card-text">Text</p>
    <a class="btn btn-primary"
href="#">Button</a>
  </div>
</div>
```

Result:



Common properties

List is used to group items together in ordered or un-ordered style.

```
<ul class=”list-group”>
<li class=”list-group-item active”>Some text</li>
<li class=”list-group-item”>Some text</li>
<li class=”list-group-item”>Some text</li>
<li class=”list-group-item”>Some text</li>
<li class=”list-group-item”>Some text</li>
</ul>
```

Result:

- some text here
- some text here
- some text here
- some text here
- some text here

Usage:

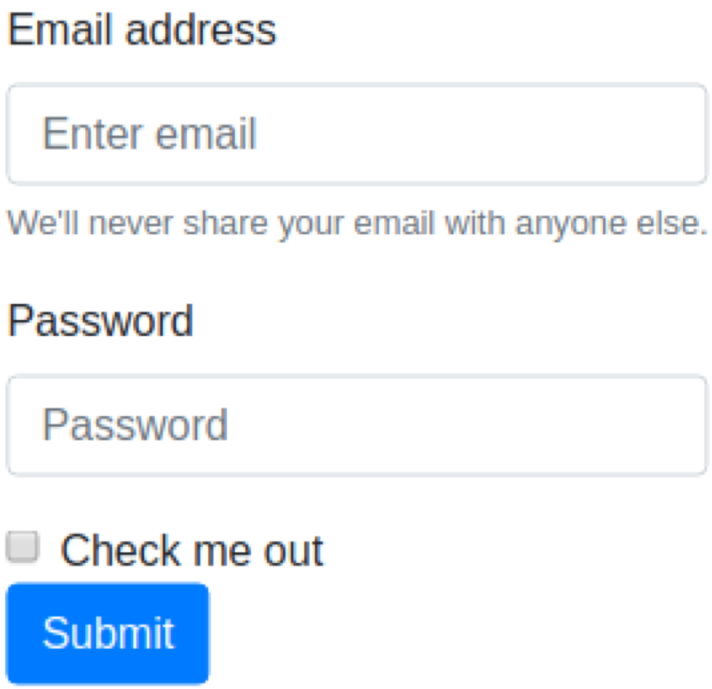
- some text here
- some text here
- some text here
- some text here
- some text here

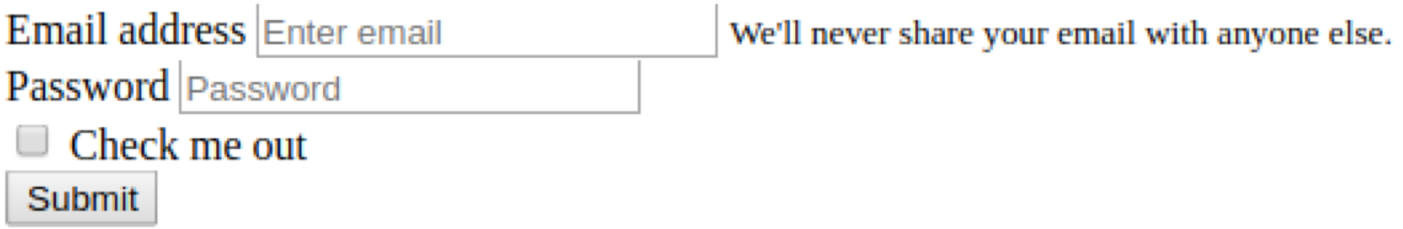
Common properties

Form is widely used and usually requires some styling to meet our requirements.

BS offering some different types of forms pre built.

BS vs regular forms



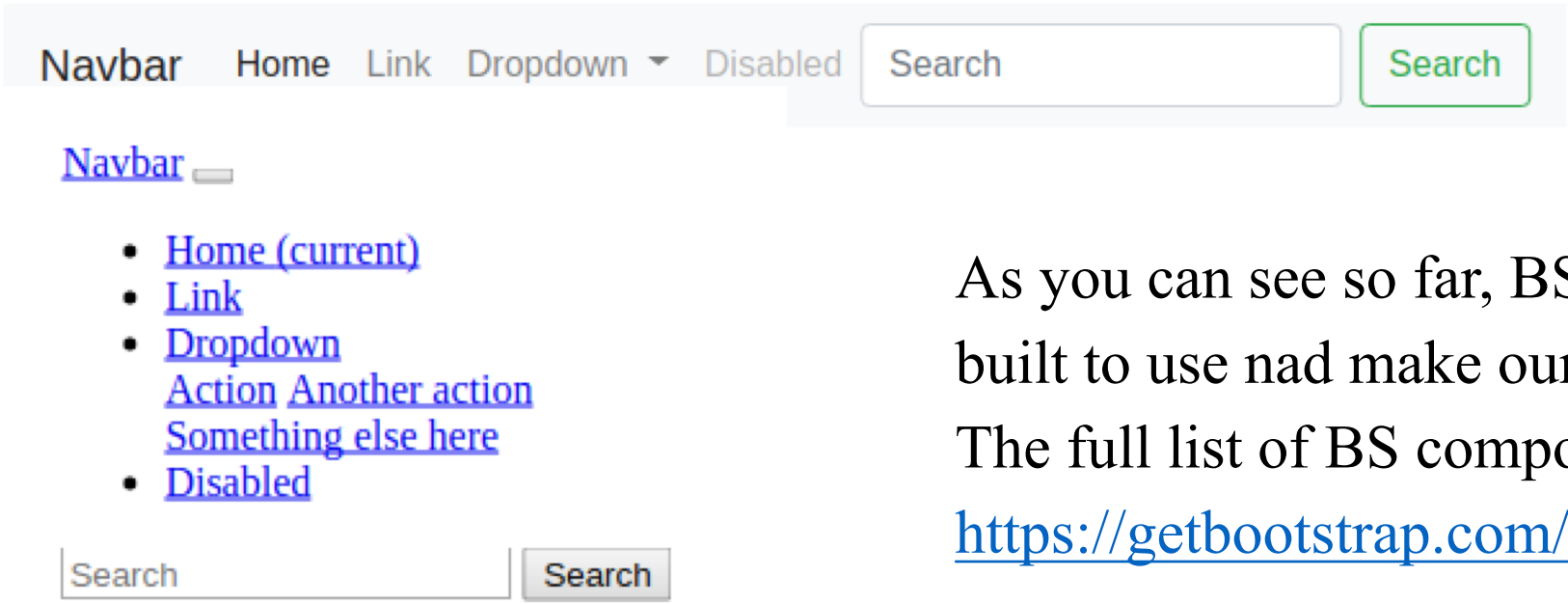


BS offers also inline forms, and some other styling to it.

Common properties

Navigation is usually found on every website and takes a mandatory part of it.
BS offers us some different types like horizontal or vertical or different theming color.

BS vs regular navs:



As you can see so far, BS offers many styles pre built to use nad make our work easier.
The full list of BS components can be found here:
https://getbootstrap.com/docs/4.0/components/

Bootstrap Grid

Grid systems taking a mandatory place in a modern web development to fully control how our element align and show to the visitors.
moreover , now when we all use our smartphones all the type, grid systems must fit any size of a screen which makes it pretty hard task.
BS offers us a well built grid system to meet the requirements.

Usage:

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive.

Bootstrap Grid

BS grid offers up to 12 columns per row.

As BS states, we must start with a container, add a row and then some columns.

One of three columns	One of three columns	One of three columns

```
<div class="container">
  <div class="row">
    <div class="col-4">
      One of three columns
    </div>
    <div class="col-4">
      One of three columns
    </div>
  </div>
</div>
```

Bootstrap Grid

Row: is defining biggest part of the content. A basic layout will have a row defining 100% width of the page, letting the columns divide it to parts.
Column: is defining an inner part of a row, usually you will use at least two columns to divide the page content as you like.

In the example above, we created full width row, and divided it to 3 columns.
We wanted to divide the page to 3 parts, therefore we used col-4 three times,
To take the whole page of 12 columns and to divide them equally.
You of course can divide rows in any size you want from 1 to full 12.

To answer the responsive matter of the grid system, you should state exactly the sizes you want per different sizes of a device.

Bootstrap Grid

There are 5 options BS offers us:

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-

Bootstrap Grid

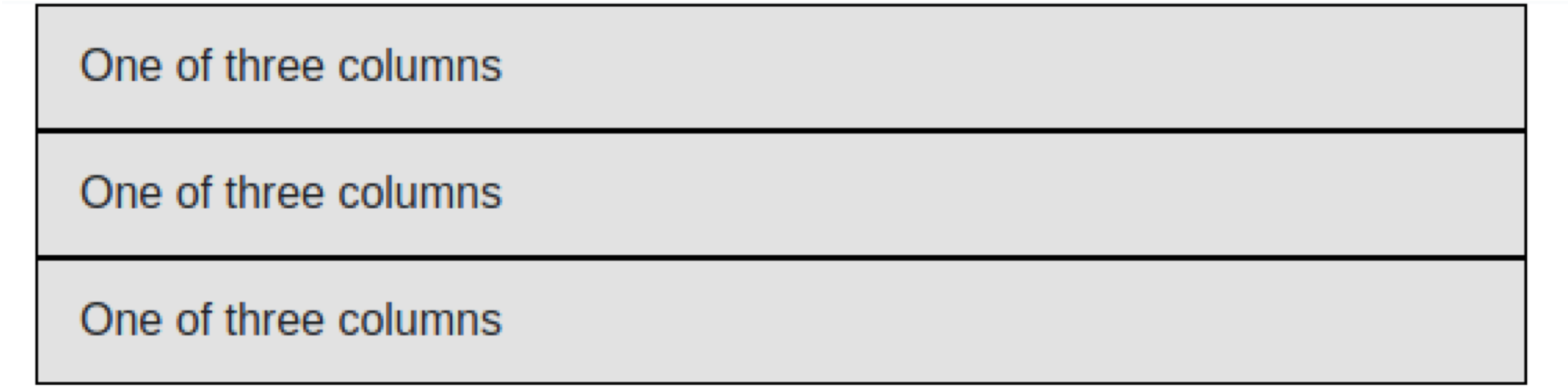
So when we want our last example to be responsive we would add some classes:

```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-6 col-xl-4">
      One of three columns
    </div>
    <div class="col-4 col-md-6 col-xl-4">
      One of three columns
    </div>
  </div>
</div>
```

Now we stated that when the screen is extra small, we want the columns to be full size of the page and take all 12 columns.
On medium devices, we want the columns to take half of the page and take 6 columns.
On extra large devices we want to keep the 4 column division of the page.

Bootstrap Grid

When we resize our browser now to be extra small size, we will see our columns take full width and stack one after another:



Understanding grid system is very important role as we stated before that now the smartphones is the main device to surf the web.
We need to be prepared to divide our page content so it will fit correctly any device size.

Working with Bootstrap

Bootstrap has many tools pre built and you may use only them to complete your page.
But usually we want to expand BS components to be more accurate and curated exactly for our project.

At first we will import the BS library at the top of our page, and then create and import our own css file after.
This file will be used to overwrite - or better to say edit BS code.
This way we can leave the good habits and stability of BS code, but edit its looks and maybe behavior to meet better the requirements of our project.

Working with Bootstrap

How to correctly edit BS code: For example this is the code for a button in BS:

```
.btn {
  Display: inline-block;
  Font-weight: 400;
  Text-align: center;
  White-space: nowrap;
  Vertical-align: middle;
  Border: 1px solid transparent;
}
.btn-primary {
  Color: #fff;
  Background-color: #007bff;
  Border-color: #007bff;
}
```

As you can see there is many styles to it, and we would like to keep them as they make it align good, look nice, work with accessibility and mainly its stable.

Working with Bootstrap

But we would like to change it loo BS button vs our new BS button.

Button

With just editing the color

Button

```
.btn-primary {
  Color: #fff;
  Background-color: #6200ff;
  Border-color: #5900a1;
}
```

Working with Bootstrap

Make sure you use as a selector exactly the same as BS library has.
To find it, just open the file of BS library, better the non minified one and find the selector there.

You can also add a new class to your element and edit it as well.

```
<a href="#" class="btn btn-primary myBtn">Button</a>
```

```
.myBtn {
  Background: #ff274b;
  Color: #000;
  Border-color: #b9122e;
}
```

Button

Hope you like it ^_^