**Internet Programming**

**Laboratory Manual**

CSE-222

Semester 4th

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hamdard UniversityList of Experiments

|  |  |  |
| --- | --- | --- |
| **Lab No** | **Topics** | **Remarks** |
| 1 | Introduction to Internet Programming |  |
| 2 | Review of HTML |  |
| 3 | Introduction to Bootstrap |  |
| 4 | Bootstrap 5 Grids, Text/Typography, Colors |  |
| 5 | Bootstrap Tables and Images |  |
| 6 | Bootstrap Forms |  |
| 7 | Introduction to jQuery |  |
| 8 | jQuery Event Methods |  |
| 9 | jQuery Effects (Part 1) |  |
| 10 | jQuery Effects (Part 2) |  |
| 11 | jQuery HTML – Get & Set |  |
| 12 | jQuery HTML – Add Elements & Remove Elements |  |
| 13 | jQuery Traversing (Part 1) |  |
| 14 | jQuery Traversing (Part 2) |  |

# Marks Evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment No.** | **Marks** | | | |
| **Class**  **Participation**  **(0.3)** | **Experiment**  **Performance**  **(0.5)** | **Experiment**  **Reporting**  **(0.2)** | **Total**  **(1)** | |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| Total |  | | | |

**Instructor’s signature**

# General Laboratory Procedure:

While there is no specific document to be submitted at the beginning of the Lab –unless your instructor advises you otherwise-, you are expected to read the experiment fully before you come to the laboratory? Interestingly, you can even try parts of the experiment at home. Here is a list of programs that will equip you with a virtual lab at your home:

**Troubleshooting**

Things will not always go as expected; this is the nature of the learning process. While conducting the Experiment **think before you do anything.** If you do so you will avoid wasting time going down dead-end streets. Be logical and systematic. First, look for obvious errors that are easy to fix. Is your measuring device correctly set and connected? Are you looking at the proper scale? Is the power supply set for the correct voltage? Is the signal generator correctly set and connected? How are the variables in the code set? Is there a syntax error? And so on. Next, check for obvious misconnections or broken connections, at least in simple circuits.

As you work through your circuit, use your Lab Manual record tests and changes that you make as you go along; don't rely on your memory for what you have tried. Identify some test points in the system at which you know what the signal should be and work your way backwards from the output through the test points until you find a good signal.

**Neatness**

When you have finished for the day, return all modules to their proper storage bins, return all test leads and probes to their storage racks, return all equipment to its correct location, and clean up the lab station. If appropriate switch off the unneeded equipment. Save your files in the Computer and on any USB device for your records because you might not get the same PC System again for the next experiment. Also email your file contents to your email address as a backup.

# Laboratory Safety

Always pay attention to what you are doing and you’re surrounding during the experiments and notify the Instructor for any unlikely event or mishap and leave the Laboratory with the permission of Instructor immediately.

All students must read and understand the information in this document with regard to laboratory safety and emergency procedures prior to the first laboratory session.

**Your personal laboratory safety depends mostly on YOU**. Effort has been made to address situations that may pose a hazard in the lab but the information and instructions provided cannot be considered all-inclusive.

Students must adhere to written and verbal safety instructions throughout the academic term. Since additional instructions may be given at the beginning of laboratory sessions, it is important that all students arrive at each session on time. With good judgment, the chance of an accident in this course is very small. Nevertheless, research and teaching workplaces (labs, shops, etc.) are full of potential hazards that can cause serious injury and or damage to the equipment. Working alone and unsupervised in laboratories is forbidden if you are working with hazardous substances or equipment. With prior approval, at least two people should be present so that one can shut down equipment and call for help in the event of an emergency. Safety training and/or information should be provided by a faculty member, teaching assistant, lab safety contact, or staff member at the beginning of a new assignment or when a new hazard is introduced into the workplace.

**Emergency Response**

1. It is your responsibility to read safety and fire alarm posters and follow the instructions during an emergency
2. Know the location of the fire extinguisher, eye wash, and safety shower in your lab and know how to use them.
3. Notify your instructor immediately after any injury, fire or explosion, or spill.
4. Know the building evacuation procedures.

**Common Sense**

Good common sense is needed for safety in a laboratory. It is expected that each student will work in a responsible manner and exercise good judgment and common sense. If at any time you are not sure how to handle a particular situation, ask your Teaching Assistant or Instructor for advice.**DO NOT TOUCH ANYTHING WITH WHICH YOU ARE NOT COMPLETELY FAMILIAR**!!! It is always better to ask questions than to risk harm to yourself or damage to the equipment.

**Personal and General laboratory safety**

1. Never eat, drink, or smoke while working in the laboratory.
2. Read labels carefully.
3. Do not use any equipment unless you are trained and approved as a user by your supervisor.
4. Wear safety glasses or face shields when working with hazardous materials and/or equipment.
5. Wear gloves when using any hazardous or toxic agent.
6. Clothing: When handling dangerous substances, wear gloves, laboratory coats, and safety shield or glasses. Shorts and sandals should not be worn in the lab at any time. Shoes are required when working in the machine shops.
7. If you have long hair or loose clothes, make sure it is tied back or confined.
8. Keep the work area clear of all materials except those needed for your work. Coats should be hung in the hall or placed in a locker. Extra books, purses, etc. should be kept away from equipment that requires air flow or ventilation to prevent overheating.
9. Disposal - Students are responsible for the proper disposal of used material if any in appropriate containers.
10. Equipment Failure - If a piece of equipment fails while being used, report it immediately to your lab assistant or tutor. Never try to fix the problem yourself because you could harm yourself and others.
11. If leaving a lab unattended, turn off all ignition sources and lock the doors.
12. Never pipette anything by mouth.
13. Clean up your work area before leaving.
14. Wash hands before leaving the lab and before eating.
15. Unauthorized person(s) shall not be allowed in a laboratory for any reason

**Electrical safety**

1. Obtain permission before operating any high voltage equipment.
2. Maintain an unobstructed access to all electrical panels.
3. Wiring or other electrical modifications must be referred to the Electronics Shop or the Building Coordinator.
4. Avoid using extension cords whenever possible. If you must use one, obtain a heavy- duty one that is electrically grounded, with its own fuse, and install it safely. Extension cords should not go under doors, across aisles, be hung from the ceiling, or plugged into other extension cords.
5. Never, ever modify, attach or otherwise change any high voltage equipment.
6. Always make sure all capacitors are discharged (using a grounded cable with an insulating handle) before touching high voltage leads or the "inside" of any equipment even after it has been turned off. Capacitors can hold charge for many hours after the equipment has been turned off.
7. When you are adjusting any high voltage equipment or a laser which is powered with a high voltage supply, USE ONLY ONE HAND. Your other hand is best placed in a pocket or behind your back. This procedure eliminates the possibility of an accident where high voltage current flows up one arm, through your chest, and down the other arm.
8. Discard damaged cords, cords that become hot, or cords with exposed wiring.
9. Before equipment is energized ensure, (1) circuit connections and layout have been checked by a Teaching Assistant (TA) and (2) all colleagues in your group give their assent.
10. Know the correct handling, storage and disposal procedures for batteries, cells, capacitors, inductors and other high energy-storage devices.
11. Experiments left unattended should be isolated from the power supplies. If for a special reason, it must be left on, a barrier and a warning notice are required.
12. Equipment found to be faulty in any way should be reported to the Lab Engineer immediately and taken out of service until inspected and declared safe.
13. Voltages above 50 V RMS AC and 120 V DC are always dangerous. Extra precautions should be considered as voltage levels are increased.
14. Never make any changes to circuits or mechanical layout without first isolating the circuit by switching off and removing connections to power supplies.
15. Know what you must do in an emergency.
16. Emergency Power Off: Every lab is equipped with and Emergency Power Off System.
17. Only authorized personnel are permitted to reset power once the Emergency Power Off system has been engaged.

**Electrical Emergency Response**

The following instructions provide guidelines for handling two types of electrical emergencies:

1. When someone suffers serious electrical shock, he or she may be knocked unconscious. If the victim is still in contact with the electrical current, immediately turn off the electrical power source. If you cannot disconnect the power source, depress the Emergency Power Off switch.
2. Do not touch a victim that is still in contact with a live power source; you could be electrocuted.
3. Have someone call for emergency medical assistance immediately. Administer first-aid, as appropriate.
4. If an electrical fire occurs, try to disconnect the electrical power source, if possible. If the fire is small and you are not in immediate danger; and you have been properly trained in fighting fires, use the correct type of fire extinguisher to extinguish the fire. When in doubt, push in the Emergency Power Off button.
5. NEVER use water to extinguish an electrical fire.

**Mechanical safety**

1. When using compressed air, use only approved nozzles and never direct the air towards any person.
2. Guards on machinery must be in place during operation.
3. Exercise care when working with or near hydraulically- or pneumatically-driven equipment. Sudden or unexpected motion can inflict serious injury.

**Additional Safety Guidelines**

1. Never do unauthorized experiments.
2. Never work alone in laboratory.
3. Keep your lab space clean and organized.
4. Do not leave an on-going experiment unattended.
5. Always inform your instructor if you break a thermometer. Do not clean mercury yourself!!
6. Never taste anything. Never pipette by mouth; use a bulb.
7. Never use open flames in laboratory unless instructed by TA.
8. Check your glassware for cracks and chips each time you use it. Cracks could cause the glassware to fail during use and cause serious injury to you or lab mates.
9. Maintain unobstructed access to all exits, fire extinguishers, electrical panels, emergency showers, and eye washes.
10. Do not use corridors for storage or work areas.
11. Do not store heavy items above table height. Any overhead storage of supplies on top of cabinets should be limited to lightweight items only. Also, remember that a 36" diameter area around all fire sprinkler heads must be kept clear at all times.
12. Areas containing lasers, biohazards, radioisotopes, and carcinogens should be posted accordingly. However, do not post areas unnecessarily and be sure that the labels are removed when the hazards are no longer present.
13. Be careful when lifting heavy objects. Only shop staff may operate forklifts or cranes.
14. Clean your lab bench and equipment, and lock the door before you leave the laboratory.

**Clothing**

1. Dress properly during a laboratory activity.
2. Long hair, dangling jewelry, and loose or baggy

Clothing are a hazard in the laboratory.

1. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured.
2. Shoes must completely cover the foot.
3. No sandals allowed on lab days.
4. A lab coat or smock should be worn during laboratory experiments.

**Accidents and Injuries**

1. Do not panic.
2. Report any accident (spill, breakage, etc.) or injury (cut, burn, etc.) to the teacher immediately, no matter how trivial it seems.
3. If you or your lab partner is hurt, immediately (and loudly) yell out the teacher's name to get the teacher's attention.

**General Warning Signs**



# LAB # 01

# Introduction to Internet Programming

**Objectives:**

* Be able to learn theoretical concept related to Internet programming

**Theory:**

### Basic Ideas of the Web

The World Wide Web (Web) is a hypermedia system. It has largely achieved the goal of Tim Berners-Lee, its British inventor, of a universal information space. Thanks to the global reach of the Internet, there is potentially universal access to an enormous volume of documents over the Internet. (Of course, in many developing countries, access is poor, which raises issues of disenfranchisement and disempowerment.) Many organizations make publicly available collections of hypermedia documents as part of either their marketing program, customer service or global operations. Computer suppliers, for example, now publish very detailed specifications of their products via the Web.

Web servers and clients may be located at any part of the world and connected to each other by telecommunication links. If the Web is in some sense a digital library, it is one with no single geographical location. When it comes to commerce, distance begins to lose importance. As long as a supplier can provide goods or services where they are required, the location of the vendor and the consumer will not matter. (This gives rise to issues about jurisdiction for taxes, consumer laws, legality of product, etc.) This absence of distance is supported by the ease with which Web documents may be located world-wide; the mechanism is straightforward thanks to the way the location of such 'resources' are identified by a Uniform Resource Location (URL). The URL format unambiguously specifies locations of 'documents' on the Web. This location mechanism allows the actual implementation of geography-independent feature of the Web.

Generally speaking, there is no central authority controlling the Web, although fully qualified domain names are subject to controlled allocation, and Internet Service Providers may be subject to the laws of the countries in which they operate. Furthermore, the World-Wide Web Consortium (W3C), headed by Tim Berners-Lee at the Massachusetts Institute of Technology, is influencing — and to a large degree controlling — how technologies are deployed on the Web. The W3C specifies HTML and XML, but others bodies, such as the European Computer Manufacturers Association (ECMA), have standardized other Web technologies, such as what we mostly call JavaScript. JavaScript is a programming language originally developed by Netscape.

Anyone with the appropriate knowledge, and with access to server space, can create a Web document. These Web documents can make reference to any other document. Moreover, a user does not require specific, proprietary software on their computer platform to access the Web, with many Web browsers being free software. While browsers can access and display the information on the Web, not all of them can supply the user with the interactive portions of the Web pages. For example, if Java applets are prohibited or a browser does not support JavaScript, interactivity with the Web document will be limited; some information may even be missing if that information required the presence of these interactive components.

The implications are easy to predict. With different browsers supporting different features, and with the navigation difficulties associated with hypertext's mesh / graph connections, chaos might ensue. However, even the most inexperienced users currently cope and the Web, and it is becoming both a universal world of information, and a universal place for doing business.

Dynamic pages can respond interactively to user input. It is possible to have portions of a hypertext document be produced by a programed as the document is requested. In this way, Web pages are increasingly being used as a front end to databases.

This allows the user to fill out a query and send it off for processing by the hypertext document. The server queries the database using the user's information and returns the output as HTML. To allow data to be sent in such a way to and from Web servers, a standard called the Common Gateway Interface (CGI) has been created. The difference between dynamic Web pages and non-dynamic (so called 'static') Web pages is transparent to the browser and user.

It is also possible to embed programs inside HTML. When the browser loads such a page, the code is immediately executed. This mechanism supports remote transactions for the commercial aspect of the Web

#### Hypertext Transfer Protocol (HTTP)

HTTP is a network protocol used to retrieve documents from a variety of machines in a minimum of time. It was invented by Tim Berners-Lee to support a project in developing a distributed hypertext system. Distributed hypertext requires the retrieval of documents from many different machines. File Transfer Protocol (FTP), which predates the Web, would be too slow for this purpose as it is a connection-oriented protocol that requires a permanent connection to a server, thus requiring a connection-maintenance overhead when accessing different machines.

Therefore, to support browsing, HTTP has the following characteristics:

* connection-less: a connection is established only for the period of transfer, and the connection need not be maintained after thereafter;
* stateless: the server has no 'history' of client visits (although the implementation of cookies overcomes this);
* comprehensive addressing: diverse files on any HTTP server world-wide can be referenced via URLs
* diverse data: using extensible MIME-types (see later), HTTP servers can supply information of every possible data types;
* rapid: allows request-response cycles of less than 100 milliseconds

HTTP is not mandatory for distributed hypertext; there are other techniques and protocols that can be used to access or transfer information. However, like TCP/IP and HTML it is ubiquitous, and so enables investment to develop e-commerce.

#### Fields of Application

The Web began as a tool to share knowledge and has successfully evolved into a general communications mechanism. With the support of transactions and synchronous communications, the Web has application in many different fields.

A primary use is the dissemination of knowledge, which takes many forms. For example, chat rooms and bulletin boards are integral to interactive discussion of all kinds of subjects. Frequently Asked Questions (FAQs), published on Web sites, and offer answers to users' questions on how to do certain kinds of tasks. The variety of information that can be pulled out of the Web is wide-ranging.

Education includes a variation of the dissemination of knowledge. Open- or distance-learning programs spearhead this aspect of the Web. Basically, any kind of demonstration on how to carry out certain tasks can be considered education. For example, a user can learn how to create a Web page from the numerous websites publishing such instructions.

With the possibilities of online trading, business transactions are carried out on the Web. The user supplies their order and credit card details so as to buy products advertised on the Web. The Selling module would cover this subject area in depth.

#### Network Protocols

A network protocol is a standard way of regulating data transmission between computers. Just as diplomats adhere to protocols — rules of behavior — when in foreign lands, network communications do the same. They have to obey agreed rules if they are to communicate and 'get on with each other'. After many years of both public and private research and development, two network protocols are now dominant: TCP (Transaction Control Protocol) and IP (Internet Protocol), together known as TCP/IP. (These were actually unlikely protocols to be so widely accepted, as faster, standardized protocols had been agreed upon, but none had the same robustness and extensibility as TCP/IP.)

Very often protocols were implemented without any formal acceptance and, because they worked most of the time, they became standards by default. Although TCP/IP is an accepted, de facto standard, work on Internet protocols continue in order to improve communication quality and support the continued growth of the Internet. There is no dictating authority for the Internet. Without a controlling authority, interim proposals about protocol changes are made by groups of interested individuals and then opened up for discussion. Documents containing the various proposed standards are published as Requests for Comment documents (RFCs). You may see references to a specific RFC as the best description of a protocol!

#### Uniform Resource Locator (URL)

An URL is needed to locate any resources on the Web. It is an address format that specifies how and where to find a document. The general format is as follows, where the various items in italics must be substituted with part of a real URL, or omitted altogether.

http://machine\_name:port/path/file\_name.file\_extension

**machine\_name** is either an IP address, for example 137.234.33.89, or a Fully Qualified Domain Name (also known as a DNS name, because Domain Name Servers map between Domain Names and IP addresses), for example, [**www.apple.com**](http://www.apple.com/)

**port** is the TCP port to connect to; this is an entry point to software on the server; an optional part of a URL

**path** is a relative file path from the server's document root; the server will start looking for a file in a specific directory and paths are relative to this

**file\_name** is the name of the file to be browsed, e.g. welcome

**file\_extension** is one of a number of suffixes which, by convention and operating system setup, indicate the type of data contained within the file, e.g. htm,html, txt.

#### The Client-server Computing Model

When you are surfing the Web, you are using a Web browser. When you go to a website for documents, the site delivers them using software called the Web server. The browser is considered to be a client in the relationship with the server as it is requesting information services from the server. This is just one particular example of the client-server model of computing.

##### A Definition and some History

The client-server model has been defined as:

A software partitioning paradigm in which a distributed system is split between one or more server tasks which accept requests, according to some protocol, from (distributed) client tasks, asking for information or action. There may be either one centralized server or several distributed ones. This model allows clients and servers to be placed independently on nodes in a network.

Client-server computing is mainly about the client computer possessing its own computing power. In the days of mainframes, all the processing power took place on central computers. The client 'terminals' were little more than a television that could send and receive characters. When microprocessors became available, it was possible to make the terminals more powerful so that they could handle some of the processing. Over time this has meant that mainframes have been replaced by smaller server machines and terminals have been replaced by more powerful client workstations.

The client-server model provides a good division of processing power, since the server primarily provides information to the client which is responsible for interpreting and displaying it. This means that servers do not have to be powerful machines, allowing more people to become service providers.

A more important characteristic is that because the client-server model provides for significant processing power at the (remote) client end, the operator of the client system has considerable autonomous power in contributing to the enterprise of which he or she is a part. This means that local decisions can be made, possibly faster than if they were made remotely, and action taken.

You may hear client-server computing being talked about as a modern computing 'paradigm'. Other than being part of a sales pitch, this is likely to mean that the model has made a significant impact on, and change to, the way we design and use computer systems. In particular, it is the current model for distributed business systems, and fits nicely into the emerging Web.

##### Functionality

In the context of the Web, users run client programmes (i.e. Web browsers) which provide the following functionality:

* They allow the user to send a request for information to the server.
* They format the request so that the server can understand it.
* They format the response from the server in a way that the user can read it.

Server programs carry out the following:

* They receive a request from a client and process the request.
* They respond by sending the requested information back to the client.

#### Information and Processing on the Web

Information is passed from the server to the browser. This information may be in the form of HTML documents, GIF files, Excel spreadsheets, movies — just about any digital content.

Information can also be passed from the browser to the server. When you click on a hyperlink you are sending information to the server, and when you fill in an online form, you are usually sending information to the server.

In addition to passing information backwards and forwards, some processing can also be done in the browser. For instance, you might have a simple Web page that calculates the overall cost of a loan once the initial value of the loan, the interest rate and the length of the loan have been entered.

But where does the processing take place? Does the server process the information and generate the result, or is it the client that processes the information? If the client does the processing, then this is a client-side application; if it is the server, it is a server-side application.

In the loan example above, the client has the information (the principle, rate and time). It could send this information to the server to process the information, generate the result and send it back to the client. Alternatively, the server could send a program to the client that will carry out the processing. In the latter case, since the client has all the information and program is pretty small, it is probably better to run the application on the client side.

Of course, there is also a problem of who has the information. If the server has a database, and the client wants to query it, then there are two possibilities. The server could send the database and the querying program to the client to process it or the server could process it and simply send the result. In this case, it would probably be better to do the processing on the server side.

## Conclusion

In this lab you have got enough idea about what is internet programming and why it is used.

## Tasks:

Q1. Write down your ideas about the possible benefits of hypertext using the following headings. If you like, go on-line to discuss these with colleagues before writing them down.

* Ease of insertion of new information
* Pointers to external materials
* Browsing

Q2. Write down your ideas about the possible drawbacks of hypertext using the following headings. If you like, go on-line to discuss these with colleagues before writing them down.

* Navigation Difficulties
* No Main Catalogues
* Network Overload
* Link Fossilisation

Q3. The client-server model applies to a lot of things outside of computers. Imagine going to a bank to withdraw some money? Who is the client and who is the server? Clearly, you are the client and the bank is the server.

One of the advantages of the client server model is that one server can handle many clients. The teller in the bank (server) handles many customers (clients). Also, you can use lots of different servers to get the service you need. (That is there are a lot of tellers, and for that matter, bank branches and cash machines.)

For any website, say the [Hamdard University Website](http://www.cs.uct.ac.za/), think about the following questions and write down your answers:

1. Are there multiple clients?
2. Who are these clients?
3. Are there multiple servers?
4. Why would there be multiple servers?

Q4. On the East Med. Trading Co. website, they would like to display to the user the number of pages that he or she has visited at that site. Think about the following questions and make a note of your answers.

1. What data is needed?
2. Where is the data stored?
3. Should this be a client or a server side application

**Learning outcomes:**

# Lab # 02

# Review of HTML

**Objectives:**

In this lab you will review the basics of HTML.

**Theory:**

##### **Standard Procedure for Creating and View an HTML document?**

* Use a text editor such as Notepad to write the document.
* Save the file as filename.html on a PC. This is called the Document Source.
* Open the file that you have saved in any browser Off-Line.
* Your HTML page should now appear just like any other Web page in browser.
* You may now switch back and forth between the Source and the HTML

**What is HTML?**

* + A series of tags that are integrated into a text document.
  + A series of tags that are integrated into a text document.
  + These look like: <code>formatted text</code> o <code> begins the formatting tag. o

</code> ends the formatting tag.

* + These tags are then read by a Browser, which translates the tags into the formatting that they represent

##### **What are Tags?**

* + HTML tags are used to mark-up HTML elements.
  + HTML tags are surrounded by the two characters < and >.
  + The surrounding characters are called angle brackets .
  + HTML tags normally come in pairs like <b> and </b>.
  + The first tag in a pair is the start tag; the second tag is the end tag.
  + The text between the start and end tags is the element content.
  + HTML tags are not case sensitive; <b> means the same as <B>.

##### **Structure Tags In HTML HTML Tag**

**<HTML></HTML>**

These tags begin and end an HTML document.

##### **HEAD Tag**

<HEAD></HEAD>

These tags are in the beginning of the document. Important information is stored in- between these tags including: title, meta-data, styles, and programming scripts

##### **TITLE Tag**

<TITLE></TITLE>

These tags are in-between the HEAD tags and contain the text that appears in the title of the Web page.

##### **BODY Tag**

<BODY></BODY>

As you may have guessed, the BODY tags contain all the text in the body of the document.

##### **Block Level tags**

**HTML Headings –**

Headings are defined with the<h1> to<h6> tags. Where <h1> -Defines the largest headings.

<h6> -Defines the smallest headings.

##### **HTML Paragraphs -**

Paragraphs are defined with the<p> tag.

##### **HTML Line Breaks -**

Use the <br/> tag if you want a line break (a new line) without starting a new paragraph.

##### **Horizontal Rule**

The <hr> element is used for horizontal rules that act as dividers between sections

##### **HTML Text Formatting Tags**

|  |  |
| --- | --- |
| <b> | Defines bold Text |
| <big> | Defines big text. |
| <em> | Defines emphasized text |
| <i> | Defines Italic text |
| <small> | Defines italic text. |
| <strong> | Defines Strong text |
| <sub> | Defines subscripted text |
| <sup> | Defines superscripted text |
| <ins> | Defines inserted text |
| <del> | Defines deleted text |

**HTML Tables:**

* Tables are defined with the <table> tag.
* A table is divided into rows (with the <tr> tag),
* Each row is divided into data cells (with the <td> tag).
* td stands for "table data," and holds the content of a data cell.
* A <td> tag can contain text, links, images, lists, forms, other tables, etc.

##### **HTML Tables and the Border Attribute**

If you do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show.

##### **To Create Table Header**

<html>

<body>

<h4>Table headers:</h4>

<table border="1">

<tr>

<th>Name</th>

<th>Telephone</th>

<th>Telephone</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

</body>

</html>

**Output**



##### **How to add a caption to a table.**

**<Caption>Monthly savings</caption>**

**Table Cells then spans more than one row/column**

For Example-

<html>

<body>

<h4>Cell that spans two columns:</h4>

<table border="1">

<tr>

<th>Name</th>

<th colspan="2">Telephone</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

<h4>Cell that spans two rows:</h4>

<table border="1">

<tr> <th>First Name:</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">Telephone:</th>

<td>555 77 854</td>

</tr>

<tr>

<td>555 77 855

</td>

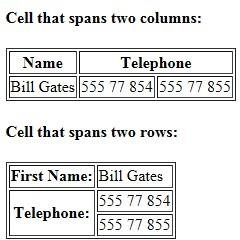
</tr>

</table>

</body>

</html>

##### **Output-**



#### The Anchor Tag and the href Attribute

* An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc. The syntax of creating an anchor:
* <a href="url">Text to be displayed</a>
* The <a> tag is used to create an anchor to link from, the href attribute is used to tell the address of the document or page we are linking to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

#### The Target Attribute

* With the target attribute, you can define **where** the linked document will be opened. By default, the link will open in the current window.
* The code below will open the document in a new browser window.

<a href=[http://www.](http://www/) google.com/ target="\_blank">Visit google!</a>

#### Email Links

* To create an email link, you will use mailto: plus your email address. Here is a link to ACC's Help Desk:

<a href="[mailto:helpdesk@abc.com"](mailto:helpdesk@abc.com)>Email Help Desk</a>

* To add a subject for the email message, you would add ?subject= after the email address. For example:

<a href="[mailto:helpdesk@abc.com?subject=Email](mailto:helpdesk@abc.com) Assistance">Email Help Desk</a>

## Conclusion

This lab gives the complete review and important tags used in HTML.

## Tasks:

Q1. Use different basic tags of HTML for creating webpage of Hamdard University that display information.

Q2. With the help of given information about the Table and Table tags design a time table in tabular format.

Q1. With the help of given procedure and information about the Links write HTML code for creating Hyperlinks in html page.

**Note: Attach a snap shot of every above mentioned task.**

**Learning outcomes:**

# Lab # 03

**Introduction to Bootstrap**

Objective:

* To start learning about Bootstrap.

**Theory:**

* Bootstrap is a free front-end framework for faster and easier web development
* Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
* Bootstrap also gives you the ability to easily create responsive designs

**What is Responsive Web Design?**  
Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

**What You Can Do with Bootstrap**

There are lot more things you can do with Bootstrap.

* You can easily create responsive websites.
* You can quickly create multi-column layout with pre-defined classes.
* You can quickly create different types of form layouts.
* You can quickly create different variation of navigation bar.
* You can easily create components like accordions, modals, etc. without writing any JS code.
* You can easily create dynamic tabs to manage large amount of content.
* You can easily create tooltips and popovers to show hint text.
* You can easily create carousel or image slider to showcase your content.
* You can quickly create different types of alert boxes.

And much more…

**Advantages of Using Bootstrap**

If you have had some experience with any front-end framework, you might be wondering what makes Bootstrap so special. Here are some advantages why one should opt for Bootstrap framework:

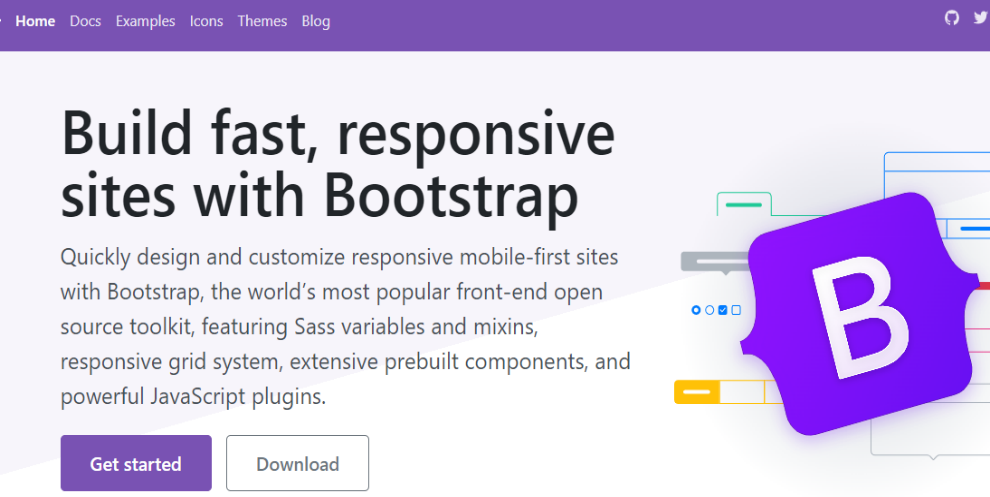
* **Save lots of time** — You can save lots of time and efforts using the Bootstrap predefined design templates and classes and concentrate on other development work.
* **Responsive features** — Using Bootstrap you can easily create responsive websites that appear more appropriately on different devices and screen resolutions without any change in markup.
* **Consistent design** — All Bootstrap components share the same design templates and styles through a central library, so the design and layout of your web pages will be consistent.
* **Easy to use** — Bootstrap is very easy to use. Anybody with the basic working knowledge of HTML, CSS and JavaScript can start development with Bootstrap.
* **Compatible with browsers** — Bootstrap is created with modern web browsers in mind and it is compatible with all modern browsers such as Chrome, Firefox, Safari, Internet Explorer, etc.
* **Open Source** — and the best part is, it is completely free to download and use.

# Getting Start Bootstrap:

**Way 1: CDN via jsDelivr**

* 1. Go to Address Bar of Google Chrome.
  2. Type: <https://getbootstrap.com/>.

You will see the screen like this:



* 1. Click on Download Button.
  2. You will see many options in Download page.
  3. Make a folder BootStrap at desktop or on any other folder.
  4. Open Notepad and write below html code.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>**

**12 Years**

**</title>**

**<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">**

**</head>**

**<body>**

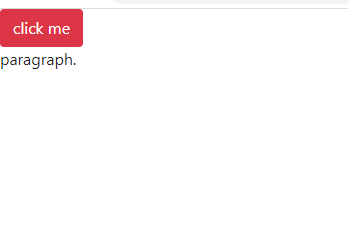
**<a href=" " class ="btn btn-danger"> click me </a>**

**<p>paragraph.</p>**

**</body>**

**</html>**

**Output:**



* 1. Save the notepad as Bootstrap.html
  2. If you will see the button like below then it is confirm that a CDN access via jsDelivr.

**Way 2: Compiled CSS and JS**

Download ready-to-use compiled code for **Bootstrap v5.1.3** to easily drop into your project, which includes:

* Compiled and minified CSS bundles
* Compiled and minified JavaScript plugins

Download from the following link from:

<https://getbootstrap.com/docs/5.1/getting-started/download/>

See the section below and download it:

## Compiled CSS and JS

Copy the CSS and JS folder to BootStrap folder where you place your following html code:

**HTML Code:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>**

**12 Years**

**</title>**

**<link rel="stylesheet" href="css/bootstrap.css">**

**</head>**

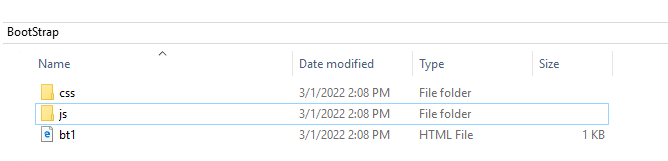
**<body>**

**<a href=" " class ="btn btn-danger"> click me </a>**

**<p>paragraph.</p>**

**</body>**

**</html>**



## Create First Web Page with Bootstrap

**1. Add the HTML5 doctype**

Bootstrap uses HTML elements and CSS properties that require the HTML5 doctype.

Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set:

<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="utf-8">  
  </head>  
</html>

**2. Bootstrap 5 is mobile-first**

Bootstrap 5 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.

To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element:

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

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

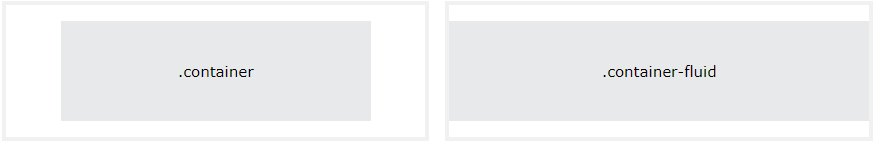
The initial-scale=1 part sets the initial zoom level when the page is first loaded by the browser.

**3. Containers**

Bootstrap also requires a containing element to wrap site contents.

There are two container classes to choose from:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport



# Bootstrap 5 Containers

Bootstrap requires a containing element to wrap site contents.

Containers are used to pad the content inside of them, and there are two container classes available:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport
3. **Fixed Container**

Use the **.container** class to create a responsive, fixed-width container.

Note that its width **(max-width)** will change on different screen sizes:

|  | **Extra small <576px** | **Small ≥576px** | **Medium ≥768px** | **Large ≥992px** | **Extra Large ≥1200px** | **XXL ≥1400px** |
| --- | --- | --- | --- | --- | --- | --- |
| max-width | 100% | 540px | 720px | 960px | 1140px | 1320px |

Write the code below and resize the browser window to see that the container width will change at different breakpoints:

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Bootstrap Example</title>**

**<meta charset="utf-8">**

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

**<link rel="stylesheet" href="css/bootstrap.css">**

**<script src="js/bootstrap.bundle.min.js"></script>**

**</head>**

**<body>**

**<div class="container">**

**<h1>My First Bootstrap Page</h1>**

**<p>This part is inside a .container class.</p>**

**<p>The .container class provides a responsive fixed width container.</p>**

**<p>Resize the browser window to see that the container width will change at different breakpoints.</p>**

**</div>**

**</body>**

**</html>**

## Fluid Container

Use the **.container-fluid** class to create a full width container that will always span the entire width of the screen **(width is always 100%)**:

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Bootstrap Example</title>**

**<meta charset="utf-8">**

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

**<link rel="stylesheet" href="css/bootstrap.css">**

**<script src="js/bootstrap.bundle.min.js"></script>**

**</head>**

**<body>**

**<div class="container-fluid">**

**<h1>My First Bootstrap Page</h1>**

**<p>This part is inside a .container class.</p>**

**<p>This part is inside a .container-fluid class.</p>**

**<p>The .container-fluid class provides a full width container, spanning the entire width of the viewport.</p>**

**</div>**

**</div>**

**</body>**

**</html>**

## Container Padding

By default, containers have left and right padding, with no top or bottom padding. Therefore, we often use **spacing utilities**, such as extra padding and margins to make them look even better. For example, .pt-5 means "add a large **top padding**":

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Bootstrap Example</title>**

**<meta charset="utf-8">**

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

**<link rel="stylesheet" href="css/bootstrap.css">**

**<script src="js/bootstrap.bundle.min.js"></script>**

**</head>**

**<body>**

**<div class="container pt-5">**

**<h1>My First Bootstrap Page</h1>**

**<p>This part is inside a .container class.</p>**

**<p>This part is inside a .container-fluid class.</p>**

**<p>The .container-fluid class provides a full width container, spanning the entire width of the viewport.</p>**

**</div>**

**</div>**

**</body>**

**</html>**

## Container Border and Color

Other utilities, such as borders and colors, are also often used together with containers:

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Bootstrap Example</title>**

**<meta charset="utf-8">**

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

**<link rel="stylesheet" href="css/bootstrap.css">**

**<script src="js/bootstrap.bundle.min.js"></script>**

**</head>**

**<body>**

**<div class="container p-5 my-5 border">**

**<h1>My First Bootstrap Page</h1>**

**<p>This container has a border and some extra padding and margins.</p>**

**</div>**

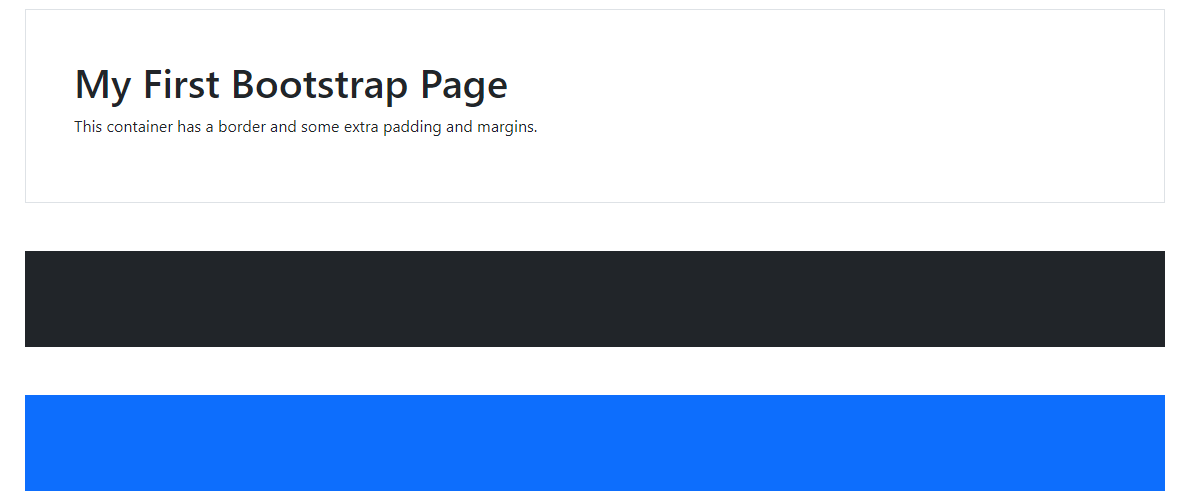
**<div class="container p-5 my-5 bg-dark text-white"></div>**

**<div class="container p-5 my-5 bg-primary text-white"></div>**

**</body>**

**</html>**

**Output:**

****

**Responsive Containers**

You can also use the **.container-sm|md|lg|xl** classes to determine when the container should be responsive.

The **max-width** of the container will change on different screen sizes/viewports:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Extra small <576px** | **Small ≥576px** | **Medium ≥768px** | **Large ≥992px** | **Extra large ≥1200px** | **XXL ≥1400px** |
| .container-sm | 100% | 540px | 720px | 960px | 1140px | 1320px |
| .container-md | 100% | 100% | 720px | 960px | 1140px | 1320px |
| .container-lg | 100% | 100% | 100% | 960px | 1140px | 1320px |
| .container-xl | 100% | 100% | 100% | 100% | 1140px | 1320px |
| .container-xxl | 100% | 100% | 100% | 100% | 100% | 1320px |

**Coding:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Bootstrap Example</title>**

**<meta charset="utf-8">**

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

**<link rel="stylesheet" href="css/bootstrap.css">**

**<script src="js/bootstrap.bundle.min.js"></script>**

**</head>**

**<body>**

**<div class="container pt-3">**

**<h1>Responsive Containers</h1>**

**<p>Resize the browser window to see the effect.</p>**

**</div>**

**<div class="container-sm border">.container-sm</div>**

**<div class="container-md mt-3 border">.container-md</div>**

**<div class="container-lg mt-3 border">.container-lg</div>**

**<div class="container-xl mt-3 border">.container-xl</div>**

**<div class="container-xxl mt-3 border">.container-xxl</div>**

**</body>**

**</html>**

**Output:**

****

## Conclusion

This lab gives the Introduction to Bootstrap and it demonstrate Container.

## Tasks:

Q1. **Explain why Bootstrap is preferred for website development.**

**Q2.** Bootstrap includes several predefined button styles, each serving its own semantic purpose, one of them you studies in class is "btn btn-danger". Below is the list of some Button Styles that used with a few extras thrown in for more control. Write the code with the following list of buttons and also demonstrate the difference between these buttons.

* **btn btn-primary**
* **btn btn-secondary**
* **btn btn-success**
* **btn btn-danger**
* **btn btn-warning**
* **btn btn-info**
* **btn btn-light**
* **btn btn-dark**
* **btn btn-link**

**Q3. Write the code to generate following output:**

Bootstrap container example

This container has 1.5rem padding, white text color and dark background.

Bootstrap container example

This container has 1rem padding, white text color and blue background.

Bootstrap container example

This container has 1rem padding, white text color, green background and text in the center.

**Learning outcomes:**

# Lab # 04

# Bootstrap 5 Grids, Text/Typography, Colors

**Objectives:**

In this lab you will learn about Bootstrap's grid system that is built with flexbox and allows up to 12 columns across the page.

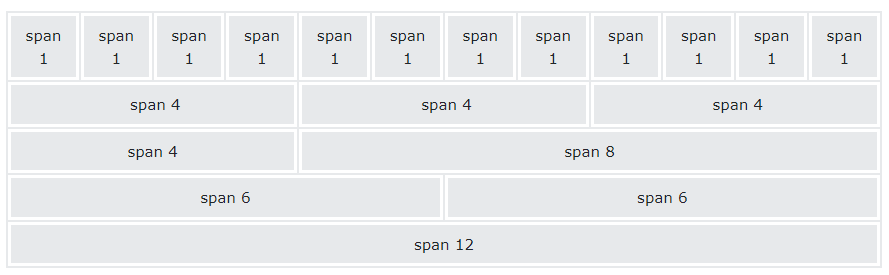
**Theory:**

## Bootstrap 5 Grid System

**A grid system** in graphic design **uses a two-dimensional framework to align and lay out design elements**. Breaking down a single design space into a grid can help position individual components in ways that can catch the eye, create a user flow and make information and visuals more appealing and accessible to audiences.

Bootstrap's grid system is built with flexbox(**a one-dimensional layout method for arranging items in rows or columns**) and allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:



The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).

## Grid Classes

The Bootstrap 5 grid system has six classes:

* .col- (extra small devices - screen width less than 576px)
* .col-sm- (small devices - screen width equal to or greater than 576px)
* .col-md- (medium devices - screen width equal to or greater than 768px)
* .col-lg- (large devices - screen width equal to or greater than 992px)
* .col-xl- (xlarge devices - screen width equal to or greater than 1200px)
* .col-xxl- (xxlarge devices - screen width equal to or greater than 1400px)

The classes above can be combined to create more dynamic and flexible layouts.

**Tip:** Each class scales up, so if you want to set the same widths for sm and md, you only need to specify sm.

## Basic Structure of a Bootstrap 5 Grid

The following is a basic structure of a Bootstrap 5 grid:

<!-- Control the column width, and how they should appear on different devices -->  
<div class="row">  
  <div class="col-\*-\*"></div>  
  <div class="col-\*-\*"></div>  
</div>  
<div class="row">  
  <div class="col-\*-\*"></div>  
  <div class="col-\*-\*"></div>  
  <div class="col-\*-\*"></div>  
</div>  
  
<!-- Or let Bootstrap automatically handle the layout -->  
<div class="row">  
  <div class="col"></div>  
  <div class="col"></div>  
  <div class="col"></div>  
</div>

Second example: instead of adding a number to each col, let bootstrap handle the layout, to create equal width columns: two "col" elements = 50% width to each col, while three cols = 33.33% width to each col. Four cols = 25% width, etc. You can also use .col-sm|md|lg|xl|xxl to make the columns responsive.

Below we have collected some examples of basic Bootstrap 5 grid layouts.

## Three Equal Columns



The following example shows how to create three equal-width columns, on all devices and screen widths:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<h1>Three equal width columns</h1>

<p>Note: Try to add a new div with class="col" inside the row class - this will create four equal-width columns.</p>

<div class="row">

<div class="col p-3 bg-primary text-white">.col</div>

<div class="col p-3 bg-dark text-white">.col</div>

<div class="col p-3 bg-primary text-white">.col</div>

</div>

</div>

</body>

</html>

## Responsive Columns



The following example shows how to create four equal-width columns starting at tablets and scaling to extra large desktops. **On mobile phones or screens that are less than 576px wide, the columns will automatically stack on top of each other**:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<h1>Responsive Columns</h1>

<p>Resize the browser window to see the effect.</p>

<p>The columns will automatically stack on top of each other when the screen is less than 576px wide.</p>

<div class="row">

<div class="col-sm-3 p-3 bg-primary text-white">.col</div>

<div class="col-sm-3 p-3 bg-dark text-white">.col</div>

<div class="col-sm-3 p-3 bg-primary text-white">.col</div>

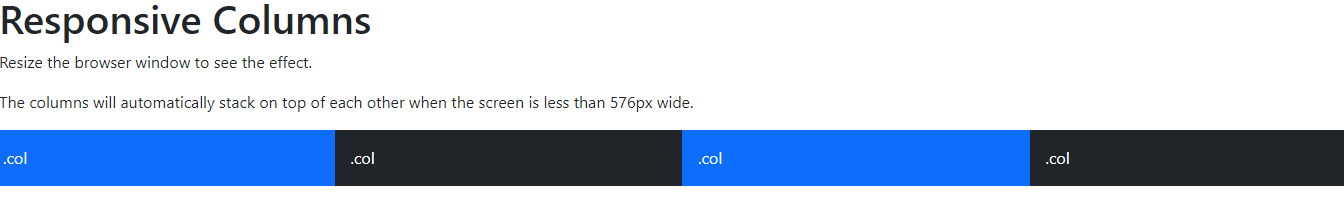
<div class="col-sm-3 p-3 bg-dark text-white">.col</div>

</div>

</body>

</html>

**Output:**



## Two Unequal Responsive Columns

.col-sm-4

.col-sm-8

The following example shows how to get two various-width columns starting at tablets and scaling to large extra desktops:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<h1>Two Unequal Responsive Columns</h1>

<p>Resize the browser window to see the effect.</p>

<p>The columns will automatically stack on top of each other when the screen is less than 576px wide.</p>

<div class="row">

<div class="col-sm-4 p-3 bg-primary text-white">.col</div>

<div class="col-sm-8 p-3 bg-dark text-white">.col</div>

</div>

</body>

</html>

# Bootstrap 5 Text/Typography

## Bootstrap 5 Default Settings

Bootstrap 5 uses a default font-size of 1rem (16px by default), and its line-height is 1.5.

In addition, all <p> elements have margin-top: 0 and margin-bottom: 1rem (16px by default).

**<h1> - <h6>**

Bootstrap 5 styles HTML headings (<h1> to <h6>)  with a bolder font-weight and a responsive font-size.

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<p>The font-size of each Bootstrap heading depends on the screen size. Try to resize the browser window to see the effect.</p>

<h1>h1 Bootstrap heading</h1>

<h2>h2 Bootstrap heading</h2>

<h3>h3 Bootstrap heading</h3>

<h4>h4 Bootstrap heading</h4>

<h5>h5 Bootstrap heading</h5>

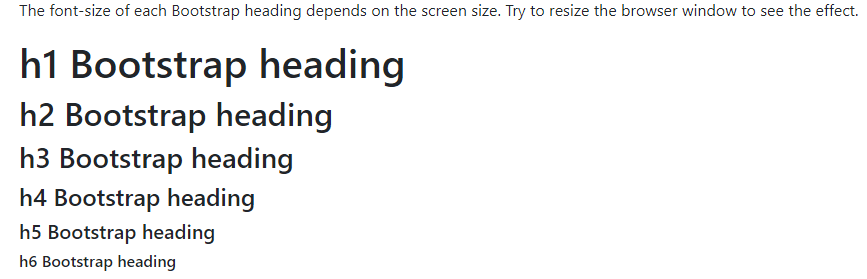
<h6>h6 Bootstrap heading</h6>

</div>

</body>

</html>

**Output:**



You can also use .h1 to .h6 classes on other elements to make them behave as headings if you want:

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<p class="h1">h1 Bootstrap heading</p>

<p class="h2">h2 Bootstrap heading</p>

<p class="h3">h3 Bootstrap heading</p>

<p class="h4">h4 Bootstrap heading</p>

<p class="h5">h5 Bootstrap heading</p>

<p class="h6">h6 Bootstrap heading</p>

</div>

</body>

</html>

**Output:**

****

## Display Headings

Display headings are used to stand out more than normal headings (larger font-size and lighter font-weight), and there are six classes to choose from: .display-1 to .display-6:

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h1>Display Headings</h1>

<p>Display headings are used to stand out more than normal headings (larger font-size and lighter font-weight):</p>

<h1 class="display-1">Display 1</h1>

<h1 class="display-2">Display 2</h1>

<h1 class="display-3">Display 3</h1>

<h1 class="display-4">Display 4</h1>

<h1 class="display-5">Display 5</h1>

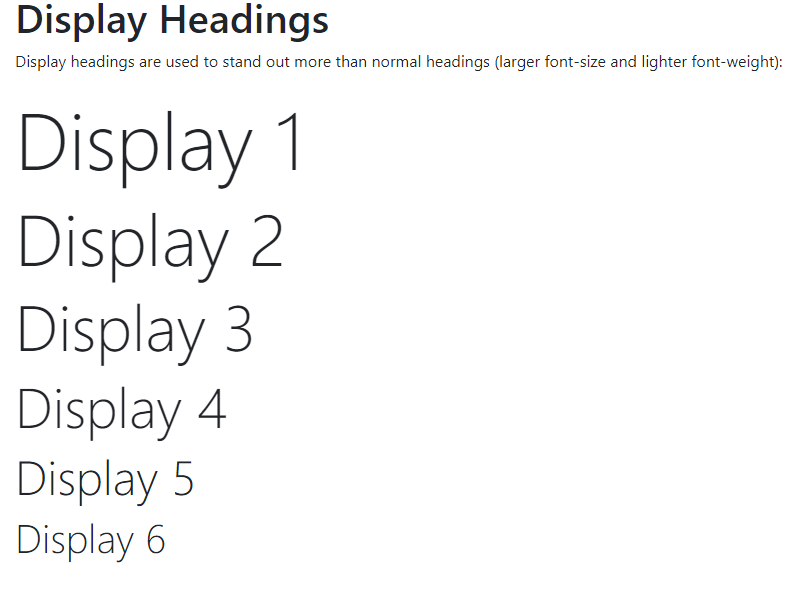
<h1 class="display-6">Display 6</h1>

</div>

</body>

</html>

**Output:**

****

## <small>

In Bootstrap 5 the HTML <small> element (and the .small class) is used to create a smaller, secondary text in any heading:

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h1>Smaller, Secondary Text</h1>

<p>The small element (and the .small class) is used to create a smaller, secondary text in any heading:</p>

<h1>h1 heading <small>secondary text</small></h1>

<h2>h2 heading <small>secondary text</small></h2>

<h3>h3 heading <small>secondary text</small></h3>

<h4>h4 heading <small>secondary text</small></h4>

<h5>h5 heading <small>secondary text</small></h5>

<h6>h6 heading <small>secondary text</small></h6>

</div>

</body>

</html>

**Output:**

## 

## <mark>

Bootstrap 5 will style <mark> and .mark with a yellow background color and some padding:

## <abbr>

Bootstrap 5 will style the HTML <abbr> element with a dotted border bottom and a cursor with question mark on hover:

## <blockquote>

Add the .blockquote class to a <blockquote> when quoting blocks of content from another source. And when naming a source, like "from WWF's website", use the .blockquote-footer class:

## <dl>

Bootstrap 5 will style the HTML <dl> element.

## <code>

Bootstrap 5 will style the HTML <code>.

## <pre>

Bootstrap 5 will style the HTML <pre> element in the following way:

## Complete Program using <mark>, <abbr>, <blockquote>, <dl>, <code> and <pre> elements:

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h1>Highlight Text</h1>

<p>Use the mark element (or the .mark class) to <mark>highlight</mark> text.</p>

<h1>Abbreviations</h1>

<p>The abbr element is used to mark up an abbreviation or acronym:</p>

<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>

<h1>Blockquotes</h1>

<p>The blockquote element is used to present content from another source:</p>

<blockquote class="blockquote">

<p>For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.</p>

<footer class="blockquote-footer">From WWF's website</footer>

</blockquote>

<h1>Description Lists</h1>

<p>The dl element indicates a description list:</p>

<dl>

<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

<h1>Code Snippets</h1>

<p>Inline snippets of code should be embedded in the code element:</p>

<p>The following HTML elements: <code>span</code>, <code>section</code>, and <code>div</code> defines a section in a document.</p>

<h1>Keyboard Inputs</h1>

<p>To indicate input that is typically entered via the keyboard, use the kbd element:</p>

<p>Use <kbd>ctrl + p</kbd> to open the Print dialog box.</p>

<h1>Multiple Code Lines</h1>

<p>For multiple lines of code, use the pre element:</p>

<pre>

Text in a pre element

is displayed in a fixed-width

font, and it preserves

both spaces and

line breaks.

</pre>

</div>

</body>

</html>

## More Typography Classes

The Bootstrap 5 classes below can be added to style HTML elements further:

|  |  |
| --- | --- |
| **Class** | **Description** |
| .lead | Makes a paragraph stand out |
| .text-start | Indicates left-aligned text |
| .text-break | Prevents long text from breaking layout |
| .text-center | Indicates center-aligned text |
| .text-decoration-none | Removes the underline from a link |
| .text-end | Indicates right-aligned text |
| .text-nowrap | Indicates no wrap text |
| .text-lowercase | Indicates lowercased text |
| .text-uppercase | Indicates uppercased text |
| .text-capitalize | Indicates capitalized text |
| .initialism | Displays the text inside an <abbr> element in a slightly smaller font size |
| .list-unstyled | Removes the default list-style and left margin on list items (works on both <ul> and <ol>). This class only applies to immediate children list items (to remove the default list-style from any nested lists, apply this class to any nested lists as well) |
| .list-inline | Places all list items on a single line (used together with .list-inline-item on each <li> elements) |

## Text Colors

Bootstrap 5 has some contextual classes that can be used to provide "meaning through colors".

The classes for text colors are: .text-muted, .text-primary, .text-success, .text-info, .text-warning, .text-danger, .text-secondary, .text-white, .text-dark, .text-body (default body color/often black) and .text-light:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<h2>Contextual Colors</h2>

<p>Use the contextual classes to provide "meaning through colors":</p>

<p class="text-muted">This text is muted.</p>

<p class="text-primary">This text is important.</p>

<p class="text-success">This text indicates success.</p>

<p class="text-info">This text represents some information.</p>

<p class="text-warning">This text represents a warning.</p>

<p class="text-danger">This text represents danger.</p>

<p class="text-secondary">Secondary text.</p>

<p class="text-dark">This text is dark grey.</p>

<p class="text-body">Default body color (often black).</p>

<p class="text-light">This text is light grey (on white background).</p>

<p class="text-white">This text is white (on white background).</p>

</body>

</html>

## Background Colors

The classes for background colors are: .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light.

Note that background colors do not set the text color, so in some cases you'll want to use them together with a .text-\* color class.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<h2>Contextual Backgrounds</h2>

<p>Use the contextual background classes to provide "meaning through colors".</p>

<p>Note that you can also add a .text-\* class if you want a different text color:</p>

<p class="bg-primary text-white">This text is important.</p>

<p class="bg-success text-white">This text indicates success.</p>

<p class="bg-info text-white">This text represents some information.</p>

<p class="bg-warning text-white">This text represents a warning.</p>

<p class="bg-danger text-white">This text represents danger.</p>

<p class="bg-secondary text-white">Secondary background color.</p>

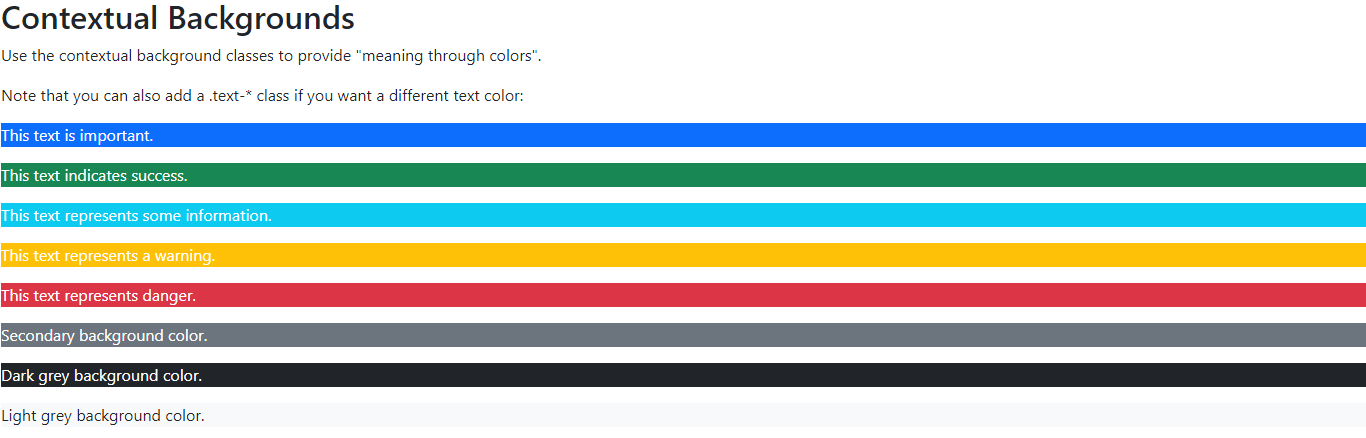
<p class="bg-dark text-white">Dark grey background color.</p>

<p class="bg-light text-dark">Light grey background color.</p>

</body>

</html>

**Output:**

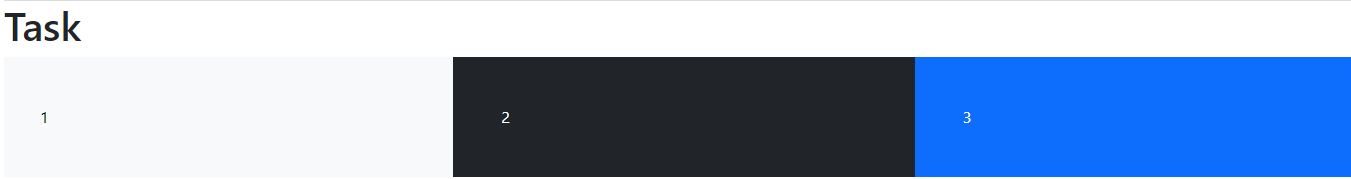


**Conclusion:**

# In this lab we learn about the Bootstrap 5 Grids, Text/Typography, Colors

## Tasks:

Q1. Write the code to generate following output:



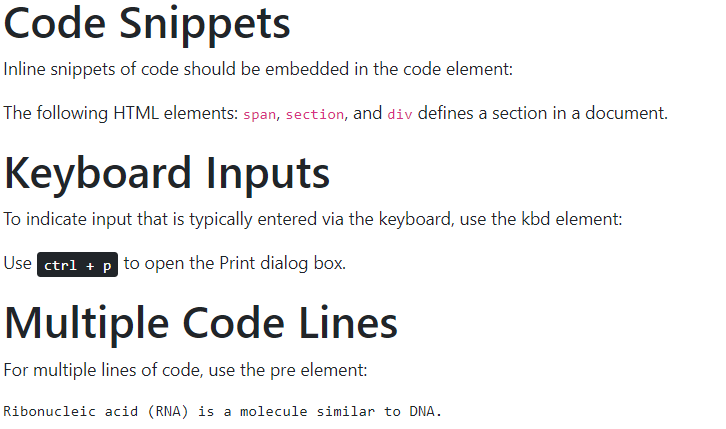
Q2. Write a program using following bootstrap functions:

* list-inline
* .list-unstyled
* .text-decoration-none
* .text-end
* .text-nowrap

Q3. Write the code to generate following output:

Output

****

****

**USE Following Text for Output:**

## Section 1

**Engineers**

Engineers

Computer Engineers

Electronic Engineers

Aerospace Engineering

Chemical Engineering

# Abbreviations

The DNA organic chemical of complex molecular structure

# Blockquotes

The blockquote element is used to present content from another source:

DNA is made of nucleotides.

From abc

# Code Snippets

Inline snippets of code should be embedded in the code element:

The following HTML elements: span, section, and div defines a section in a document.

# Keyboard Inputs

To indicate input that is typically entered via the keyboard, use the kbd element:

Use ctrl + p to open the Print dialog box.

# Multiple Code Lines

For multiple lines of code, use the pre element:

Ribonucleic acid (RNA) is a molecule similar to DNA.

**Learning outcomes:**

# Lab # 05

# Bootstrap Tables and Images

Objectives:

* To learn the basic structure of .

Theory:

## Basic Table

A basic Bootstrap 5 table has a light padding and horizontal dividers.

The .table class adds basic styling to a table:

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Basic Table</h2>

<p>The .table class adds basic styling (light padding and horizontal dividers) to a table:</p>

<table class="table">

<thead>

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Email</th>

</tr>

</thead>

<tbody>

<tr>

<td>John</td>

<td>Doe</td>

<td>john@example.com</td>

</tr>

<tr>

<td>Mary</td>

<td>Moe</td>

<td>mary@example.com</td>

</tr>

<tr>

<td>July</td>

<td>Dooley</td>

<td>july@example.com</td>

</tr>

</tbody>

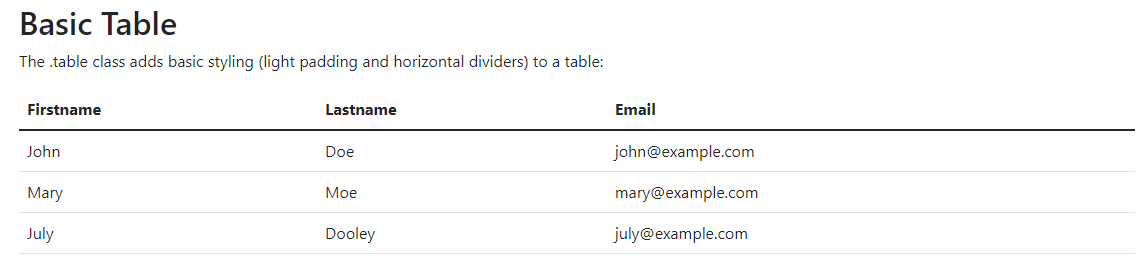
</table>

</div>

</body>

</html>

**Output:**



## Striped Rows

The .table-striped class adds zebra-stripes to a table:

<table class="table table-striped">

## Bordered Table

The .table-bordered class adds borders on all sides of the table and cells:

<table class="table table-bordered">

## Hover Rows

The .table-hover class adds a hover effect (grey background color) on table rows:

<table class="table table-hover">

## Black/Dark Table

The .table-dark class adds a black background to the table:

<table class="table table-dark">

## Borderless Table

The .table-borderless class removes borders from the table:

<table class="table table-borderless">

## Contextual Classes

Contextual classes can be used to color the whole table (<table>),  the table rows (<tr>) or table cells (<td>).

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Contextual Classes</h2>

<p>Contextual classes can be used to color the table, table rows or table cells. The classes that can be used are: .table-primary, .table-success, .table-info, .table-warning, .table-danger, .table-active, .table-secondary, .table-light and .table-dark:</p>

<table class="table">

<thead>

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Email</th>

</tr>

</thead>

<tbody>

<tr>

<td>Default</td>

<td>Defaultson</td>

<td>def@somemail.com</td>

</tr>

<tr class="table-primary">

<td>Primary</td>

<td>Joe</td>

<td>joe@example.com</td>

</tr>

<tr class="table-success">

<td>Success</td>

<td>Doe</td>

<td>john@example.com</td>

</tr>

<tr class="table-danger">

<td>Danger</td>

<td>Moe</td>

<td>mary@example.com</td>

</tr>

<tr class="table-info">

<td>Info</td>

<td>Dooley</td>

<td>july@example.com</td>

</tr>

<tr class="table-warning">

<td>Warning</td>

<td>Refs</td>

<td>bo@example.com</td>

</tr>

<tr class="table-active">

<td>Active</td>

<td>Activeson</td>

<td>act@example.com</td>

</tr>

<tr class="table-secondary">

<td>Secondary</td>

<td>Secondson</td>

<td>sec@example.com</td>

</tr>

<tr class="table-light">

<td>Light</td>

<td>Angie</td>

<td>angie@example.com</td>

</tr>

<tr class="table-dark">

<td>Dark</td>

<td>Bo</td>

<td>bo@example.com</td>

</tr>

</tbody>

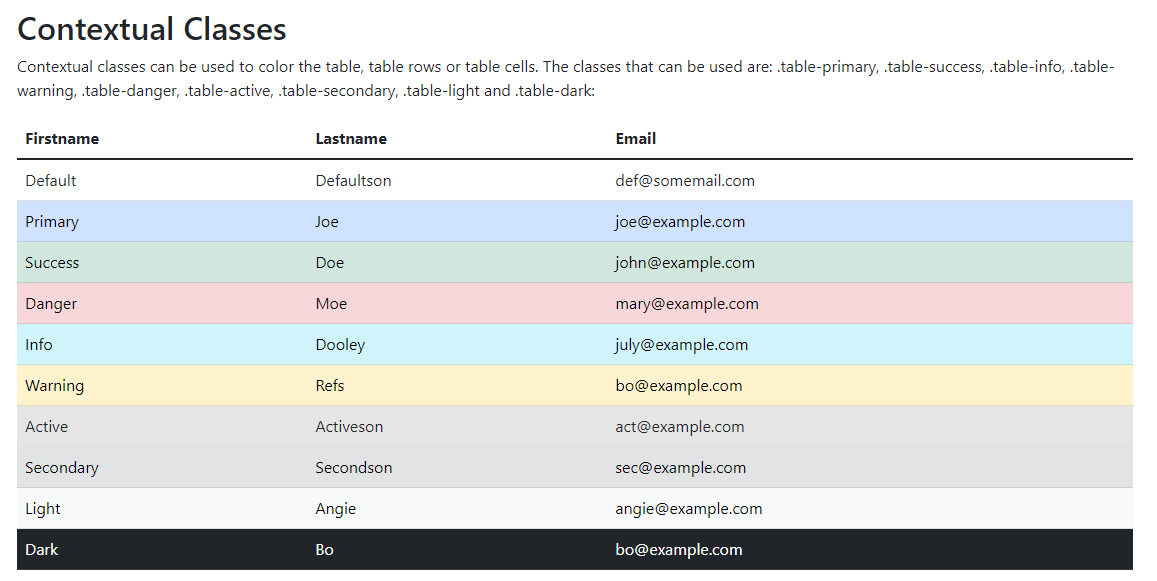
</table>

</div>

</body>

</html>

**Output:**



The contextual classes that can be used are:

| **Class** | **Description** |
| --- | --- |
| .table-primary | Blue: Indicates an important action |
| .table-success | Green: Indicates a successful or positive action |
| .table-danger | Red: Indicates a dangerous or potentially negative action |
| .table-info | Light blue: Indicates a neutral informative change or action |
| .table-warning | Orange: Indicates a warning that might need attention |
| .table-active | Grey: Applies the hover color to the table row or table cell |
| .table-secondary | Grey: Indicates a slightly less important action |
| .table-light | Light grey table or table row background |
| .table-dark | Dark grey table or table row background |

## Responsive Tables

The .table-responsive class adds a scrollbar to the table when needed (when it is too big horizontally):

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="table-responsive">

<table class="table table-bordered">

<thead>

<tr>

<th>#</th>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

<th>City</th>

<th>Country</th>

<th>Sex</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

<th>Example</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>Anna</td>

<td>Pitt</td>

<td>35</td>

<td>New York</td>

<td>USA</td>

<td>Female</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

<td>Yes</td>

</tr>

</tbody>

</table>

</div>

</div>

</body>

</html>

# Bootstrap 5 Images

## Image Shapes



The .rounded class adds rounded corners to an image:

The .rounded-circle class shapes the image to a circle:

The .img-thumbnail class shapes the image to a thumbnail (bordered):

## Aligning Images

Float an image to the left with the .float-start class or to the right with .float-end:

<img src="image.jpg" class="float-start" alt="Cinque Terre" width="200" height="200">

<img src="image.jpg" class="float-end" alt="Cinque Terre" width="200" height="200">

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<img src="image.jpg" class="rounded" alt="Cinque Terre" width="250" height="250">

<img src="image.jpg" class="rounded-circle" alt="Cinque Terre" width="200" height="200">

<img src="image.jpg" class="img-thumbnail" alt="Cinque Terre" width="200" height="200">

<br>

<br>

<br>

<img src="image.jpg" class="float-start" alt="Cinque Terre" width="200" height="200">

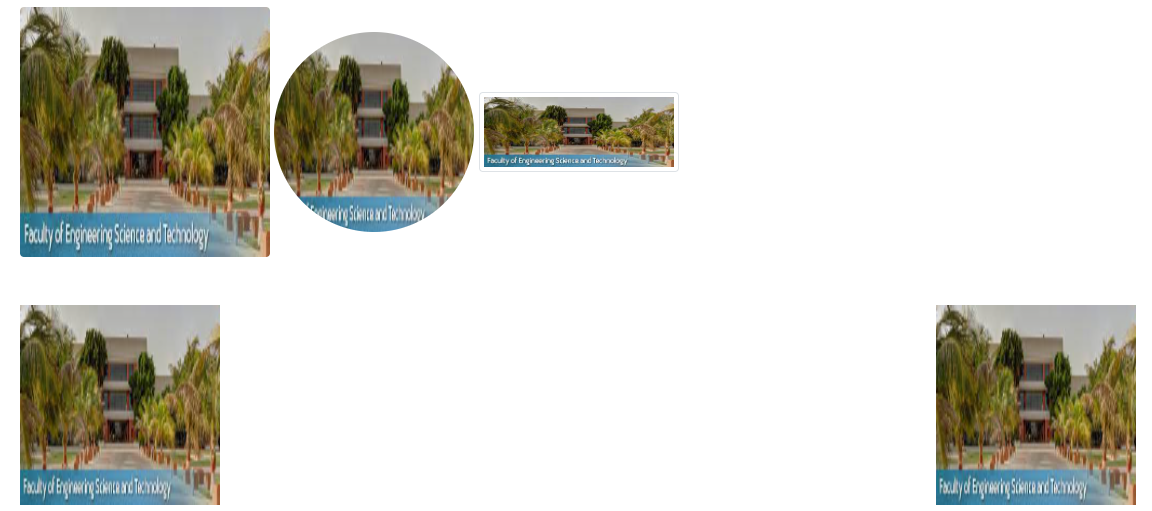
<img src="image.jpg" class="float-end" alt="Cinque Terre" width="200" height="200">

</div>

</body>

</html>

**Output:**



## Responsive Images

Images come in all sizes. So do screens. Responsive images automatically adjust to fit the size of the screen.

Create responsive images by adding an .img-fluid class to the <img> tag. The image will then scale nicely to the parent element.

The .img-fluid class applies max-width: 100%; and height: auto; to the image:

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<img class="img-fluid" src="image.jpg" alt="New York" width="1100" height="500">

</div>

</body>

</html>

## Tasks:

Q1. Write the code to print your time table with the Combination of .table-dark, table-hover and .table-striped to create a dark, striped table.

Hint:

<table class="table table-dark table-hover … ">

Q2. Write the code to place an image in your web page with following image properties:

The corners of image is rounded and shape of image is circle. The position of image is to the right of the screen. The image should also be responsive.

**Learning outcomes:**

# Lab # 06

# Bootstrap Forms

Objectives:

* .

Theory:

## Stacked Form

All textual <input> and  <textarea> elements with class .form-control get proper form styling:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<form action="/action\_page.php">

<div class="mb-3 mt-3">

<label for="email" class="form-label">Email:</label>

<input type="email" class="form-control" id="email" placeholder="Enter email" name="email">

</div>

<div class="mb-3">

<label for="pwd" class="form-label">Password:</label>

<input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd">

</div>

<div class="form-check mb-3">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" name="remember"> Remember me

</label>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</body>

</html>

Also note that we add a .form-label class to each label element to ensure correct padding.

Checkboxes have different markup. They are wrapped around a container element with .form-check, and labels have a class of .form-check-label, while checkboxes and radio buttons use .form-check-input.

## Text area

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Textarea</h2>

<p>Use the .form-control class to style textareas as well:</p>

<form action="/action\_page.php">

<div class="mb-3 mt-3">

<label for="comment">Comments:</label>

<textarea class="form-control" rows="5" id="comment" name="text"></textarea>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

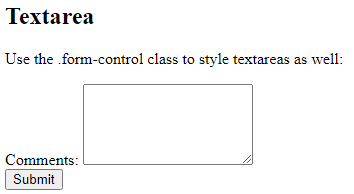
</form>

</div>

</body>

</html>

## Output:



## Form Row/Grid (Inline Forms)

If you want your form elements to appear side by side, use .row and .col:

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Inline Forms</h2>

<p>If you want your form elements to appear side by side, use .row and .col:</p>

<form>

<div class="row">

<div class="col">

<input type="text" class="form-control" placeholder="Enter email" name="email">

</div>

<div class="col">

<input type="password" class="form-control" placeholder="Enter password" name="pswd">

</div>

</div>

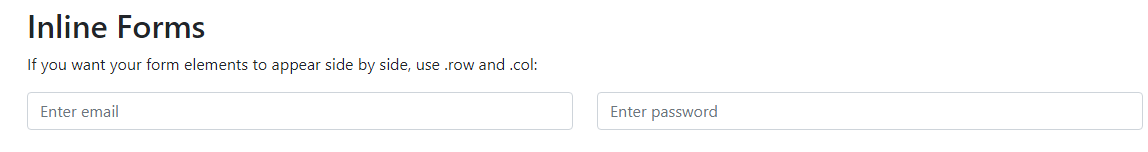
</form>

</div>

</body>

</html>

**Output:**



## Form Control Size

You can change the size of .form-control inputs with .form-control-lg or .form-control-sm:

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Inline Forms</h2>

<p>If you want your form elements to appear side by side, use .row and .col:</p>

<form>

<input type="text" class="form-control form-control-lg" placeholder="Large input">

<input type="text" class="form-control" placeholder="Normal input">

<input type="text" class="form-control form-control-sm" placeholder="Small input">

</form>

</div>

</body>

</html>

**Output:**



## Disabled and Readonly

Use the disabled and/or readonly attributes to disable the input field:

<input type="text" class="form-control" placeholder="Normal input">  
<input type="text" class="form-control" placeholder="Disabled input" disabled>  
<input type="text" class="form-control" placeholder="Readonly input" readonly>

## Plain text Inputs

Use the .form-control-plaintext class to style an input field without borders, but keep proper marigins and padding:

<input type="text" class="form-control-plaintext" placeholder="Plaintext input">

<input type="text" class="form-control" placeholder="Normal input">

## Color Picker

<input type="color" class="form-control form-control-color" value="#CCCCCC">

# Bootstrap 5 Select

Select menus are used if you want to allow the user to pick from multiple options.

To style a select menu in Bootstrap 5, add the .form-select class to the <select> element:

<div class="container mt-3">

<form>

<select class="form-select">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

</select>

</form>

</div>

## Data Lists

Bootstrap will also style data lists, which is a list of pre-defined options for an <input> element:

**Code:**

Choose your browser from the list:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<label for="browser" class="form-label">Choose your browser from the list:</label>

<input class="form-control" list="browsers" name="browser" id="browser">

<datalist id="browsers">

<option value="Edge">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

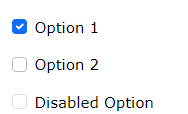
</body>

</html>

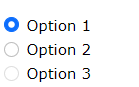
# Bootstrap 5 Checkboxes and Radio buttons::

# Checkboxes, Radio buttons, Toggle Switches, ,Range

Checkboxes are used if you want the user to select any number of options from a list of preset options.



Radio buttons are used if you want to limit the user to just one selection from a list of preset options.



If you want your checkbox to be styled as a toggle switch, use the .form-switch class together with the .form-check container:



Code:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h2>Stacked form</h2>

<form action="/action\_page.php">

<div class="mb-3 mt-3">

<label for="email">Email:</label>

<input type="email" class="form-control" id="email" placeholder="Enter email" name="email">

</div>

<div class="mb-3">

<label for="pwd">Password:</label>

<input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd">

</div>

<div class="form-check mb-3">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" name="remember"> Remember me

</label>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>

<br> <br> <br>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio1" name="optradio" value="option1" checked>Option 1

<label class="form-check-label" for="radio1"></label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio2" name="optradio" value="option2">Option 2

<label class="form-check-label" for="radio2"></label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" disabled>Option 3

<label class="form-check-label"></label>

</div>

<br> <br> <br>

<div class="form-check form-switch">

<input class="form-check-input" type="checkbox" id="mySwitch" name="darkmode" value="yes" checked>

<label class="form-check-label" for="mySwitch">Dark Mode</label>

</div>

</body>

</html>

# Bootstrap 5 Form Validation

## Form Validation

You can use different validation classes to provide valuable feedback to users. Add either .was-validated or .needs-validation to the <form> element, depending on whether you want to provide validation feedback before or after submitting the form. The input fields will have a green (valid) or red (invalid) border to indicate what's missing in the form. You can also add a .valid-feedback or .invalid-feedback message to tell the user explicitly what's missing, or needs to be done before submitting the form.

**Top of Form**

**Coding:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.css">

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container mt-3">

<h3>Form Validation</h3>

<p>Try to submit the form.</p>

<form action="/action\_page.php" class="was-validated">

<div class="mb-3 mt-3">

<label for="uname" class="form-label">Username:</label>

<input type="text" class="form-control" id="uname" placeholder="Enter username" name="uname" required>

<div class="valid-feedback">Valid.</div>

<div class="invalid-feedback">Please fill out this field.</div>

</div>

<div class="mb-3">

<label for="pwd" class="form-label">Password:</label>

<input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd" required>

<div class="valid-feedback">Valid.</div>

<div class="invalid-feedback">Please fill out this field.</div>

</div>

<div class="form-check mb-3">

<input class="form-check-input" type="checkbox" id="myCheck" name="remember" required>

<label class="form-check-label" for="myCheck">I agree on blabla.</label>

<div class="valid-feedback">Valid.</div>

<div class="invalid-feedback">Check this checkbox to continue.</div>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

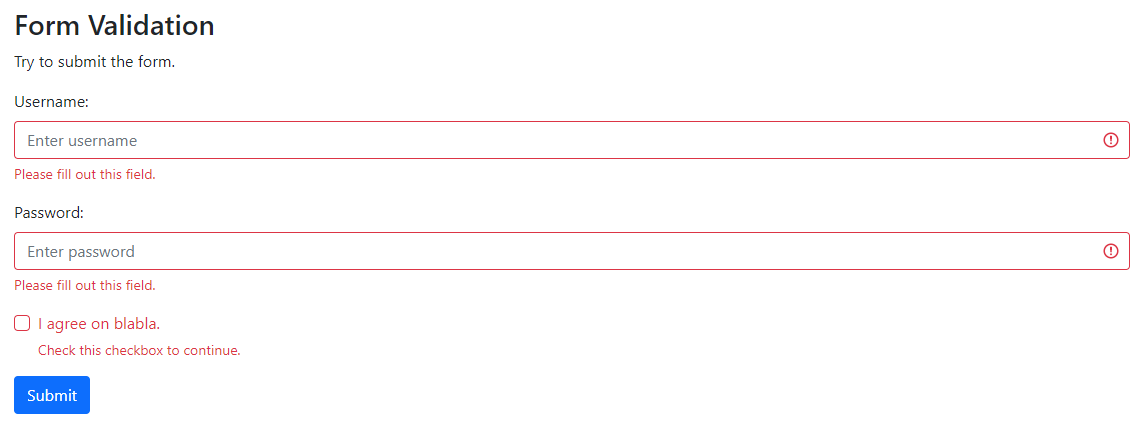
</form>

</div>

</body>

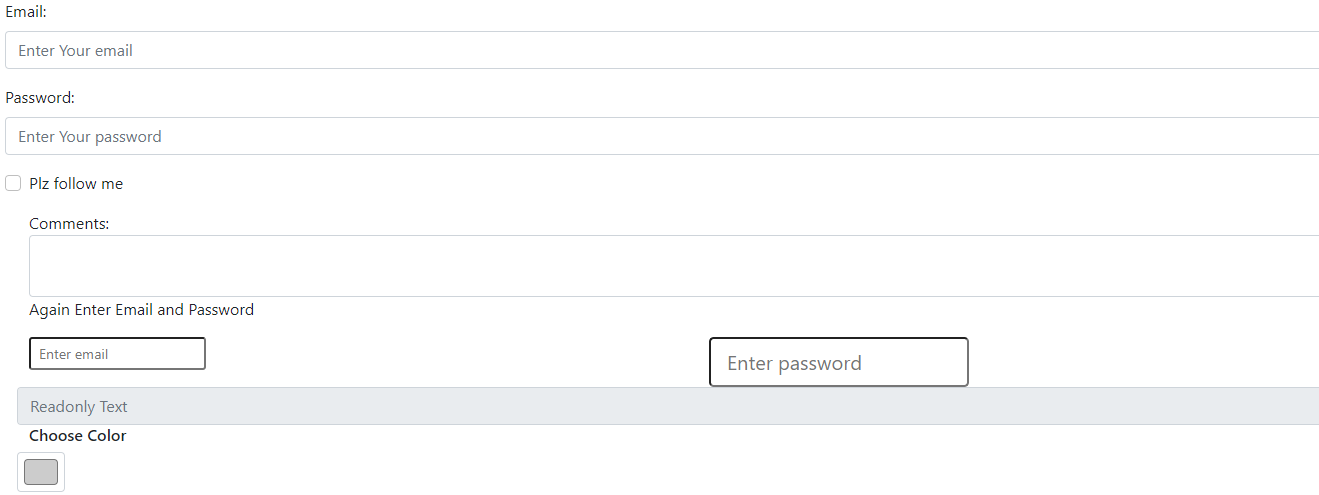
</html>

Output:

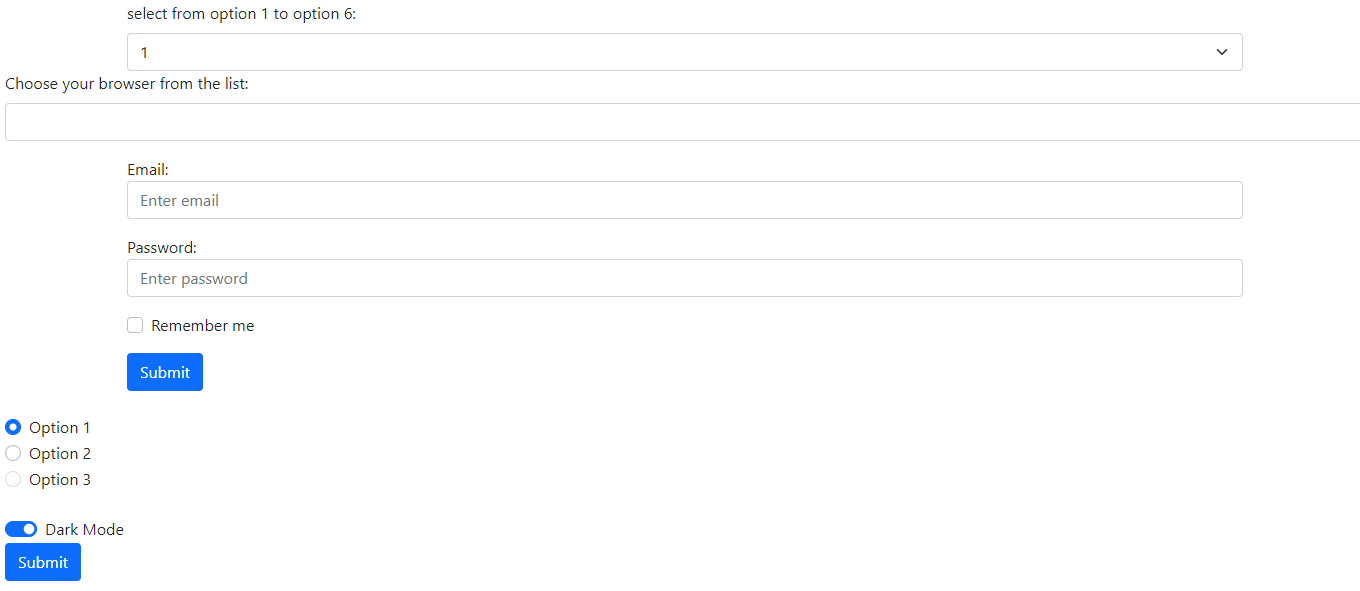


## Tasks:

Q1. Write the code to print the following screen:



Q2. Write the code to print the following screen:



**Data for Q2:**

Options:

* 1
* 2
* 3
* 4

Browser Datalist Options:

* Edge
* Firefox
* Chrome
* Opera
* Safari

**Learning outcomes:**

# Lab # 07

# Introduction to jQuery

Objectives:

To familiar with the environment of Jquery.

Theory:

The purpose of jQuery is to make it much easier to use JavaScript on your website.

## What You Should Already Know

Before you start studying jQuery, you should have a basic knowledge of:

* HTML
* CSS
* JavaScript

## What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

* HTML/DOM manipulation
* CSS manipulation
* HTML event methods
* Effects and animations
* AJAX
* Utilities

## Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

* Google
* Microsoft
* IBM
* Netflix

**Will jQuery work in all browsers?**  
The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers.

# jQuery Get Started

## Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

* Download the jQuery library from jQuery.com
* Include jQuery from a CDN, like Google

## Downloading jQuery

There are two versions of jQuery available for downloading:

* Production version - this is for your live website because it has been minified and compressed
* Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from [jQuery.com](http://jquery.com/download/).

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

<head>  
<script src="jquery-3.6.0.min.js"></script>  
</head>

**Tip:** Place the downloaded file in the same directory as the pages where you wish to use it.

# jQuery Syntax

With jQuery you select (query) HTML elements and perform "actions" on them.

## jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

* A $ sign to define/access jQuery
* A (*selector*) to "query (or find)" HTML elements
* A jQuery *action*() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

## The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

$(document).ready(function(){  
  
  *// jQuery methods go here...*  
  
});

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

* Trying to hide an element that is not created yet
* Trying to get the size of an image that is not loaded yet

**Tip:** The jQuery team has also created an even shorter method for the document ready event:

$(function(){  
  
  *// jQuery methods go here...*  
  
});

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

# jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library.

Query selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing [CSS Selectors](https://www.w3schools.com/cssref/css_selectors.asp), and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: $().

## The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

$("p")

**Example**

When a user clicks on a button, all <p> elements will be hidden:

Code:

<!DOCTYPE html>

<html>

<head>

<script>

$(document).ready(function(){

$("button").click(function(){

$("p").hide();

});

});

</script>

</head>

<body>

<h2>This is a heading</h2>

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

<p>This is another paragraph.</p>

<button>Click me to hide paragraphs</button>

</body>

</html>

## The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

$("#test")

**Example**

When a user clicks on a button, the element with id="test" will be hidden:

Code:

<!DOCTYPE html>

<html>

<head>

<script src="jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#test").hide();

});

});

</script>

</head>

<body>

<h2>This is a heading</h2>

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

<p id="test">This is another paragraph.</p>

<button>Click me</button>

</body>

</html>

## The .class Selector

The jQuery .class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

$(".test")

**Example**

When a user clicks on a button, the elements with class="test" will be hidden:

### Example

$(document).ready(function(){  
  $("button").click(function(){  
    $(".test").hide();  
  });  
});

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$(".test").hide();

});

});

</script>

</head>

<body>

<h2 class="test">This is a heading</h2>

<p class="test">This is a paragraph.</p>

<p>This is another paragraph.</p>

<button>Click me</button>

</body>

</html>

## Functions In a Separate File

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file.

When we demonstrate jQuery in this tutorial, the functions are added directly into the <head> section. However, sometimes it is preferable to place them in a separate file, like this (use the src attribute to refer to the .js file):

Example:

<head>  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>  
<script src="my\_jquery\_functions.js"></script>  
</head>

## Tasks:

Q1. Write a function in JS that hides the class elements of HTML paragraph.

Q2. Write a function in JS that hides the class elements of HTML heading.

Q3. Write a program in JS in which when a user clicks on a button, the element with an id will be hidden:

**Learning outcomes:**

# Lab # 08

# jQuery Event Methods

Objectives:

To familiar with the Jquery event methods.

Theory:

jQuery is tailor-made to respond to events in an HTML page.

## What are Events?

All the different visitors' actions that a web page can respond to are called events.

An event represents the precise moment when something happens.

Examples:

* moving a mouse over an element
* selecting a radio button
* clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

Here are some common DOM events:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mouse Events** | **Keyboard Events** | **Form Events** | **Document/Window Events** |
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave |  | blur | unload |

## jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

$("p").click();

The next step is to define what should happen when the event fires. You must pass a function to the event:

$("p").click(function(){  
  // action goes here!!  
});

## Commonly Used jQuery Event Methods

**$(document).ready()**

The $(document).ready() method allows us to execute a function when the document is fully loaded.

**click()**

The click() method attaches an event handler function to an HTML element.

The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("p").click(function(){

$(this).hide();

});

});

</script>

</head>

<body>

<p>If you click on me, I will disappear.</p>

<p>Click me away!</p>

<p>Click me too!</p>

</body>

</html>

**dblclick()**

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("p").dblclick(function(){

$(this).hide();

});

});

</script>

</head>

<body>

<p>If you double-click on me, I will disappear.</p>

<p>Click me away!</p>

<p>Click me too!</p>

</body>

</html>

**mouseenter()**

The mouseenter() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer enters the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#p1").mouseenter(function(){

alert("You entered p1!");

});

});

</script>

</head>

<body>

<p id="p1">Enter this paragraph.</p>

</body>

</html>

**mouseleave()**

The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#p1").mouseleave(function(){

alert("Bye! You now leave p1!");

});

});

</script>

</head>

<body>

<p id="p1">This is a paragraph.</p>

</body>

</html>

**mousedown()**

The mousedown() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#p1").mousedown(function(){

alert("Mouse down over p1!");

});

});

</script>

</head>

<body>

<p id="p1">This is a paragraph.</p>

</body>

</html>

**mouseup()**

The mouseup() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#p1").mouseup(function(){

alert("Mouse up over p1!");

});

});

</script>

</head>

<body>

<p id="p1">This is a paragraph.</p>

</body>

</html>

**hover()**

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#p1").hover(function(){

alert("You entered p1!");

},

function(){

alert("Bye! You now leave p1!");

});

});

</script>

</head>

<body>

<p id="p1">This is a paragraph.</p>

</body>

</html>

**focus()**

The focus() method attaches an event handler function to an HTML form field.

The function is executed when the form field gets focus:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("input").focus(function(){

$(this).css("background-color", "yellow");

});

$("input").blur(function(){

$(this).css("background-color", "green");

});

});

</script>

</head>

<body>

Name: <input type="text" name="fullname"><br>

Email: <input type="text" name="email">

</body>

</html>

**blur()**

The blur() method attaches an event handler function to an HTML form field.

The function is executed when the form field loses focus:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("input").focus(function(){

$(this).css("background-color", "yellow");

});

$("input").blur(function(){

$(this).css("background-color", "green");

});

});

</script>

</head>

<body>

Name: <input type="text" name="fullname"><br>

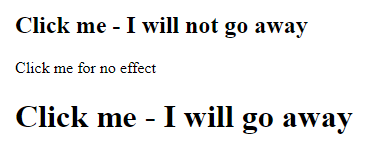
Email: <input type="text" name="email">

</body>

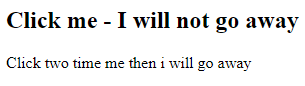
</html>

## Tasks:

Q1. Write the code with output given below. When you click on heading “Click me - I will go away” then this heading must disapper.

****

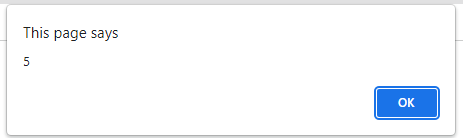
Q2. Write the code with output given below. When you double click on paragraph “Click two time me then i will go away” then this must disapper.



Q3. Write the program that display following paragraph with id. When you click on the paragraph then alert box will appear and display sum of 2 and 3.

****

On click 2+3 alert box will appear as shown below:



**Learning outcomes:**

# Lab # 09

# jQuery Effects (Part 1)

Objectives:

To familiar with the Jquery event effects.

Theory:

# jQuery Effects - Hide and Show:

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#hide").click(function(){

$("p").hide();

});

$("#show").click(function(){

$("p").show();

});

});

</script>

</head>

<body>

<p>If you click on the "Hide" button, I will disappear.</p>

<button id="hide">Hide</button>

<button id="show">Show</button>

</body>

</html>

**Speed parameter of the hiding/showing**

**Syntax:**

$(*selector*).hide(*speed,callback*);  
  
$(*selector*).show(*speed,callback*);

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("p").hide(2000);

});

});

</script>

</head>

<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>

<p>This is another small paragraph.</p>

</body>

</html>

## jQuery toggle()

You can also toggle between hiding and showing an element with the toggle() method. Shown elements are hidden and hidden elements are shown:

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("p").toggle();

});

});

</script>

</head>

<body>

<button>Toggle between hiding and showing the paragraphs</button>

<p>This is a paragraph with little content.</p>

<p>This is another small paragraph.</p>

</body>

</html>

**jQuery Effects - Fading**

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

* fadeIn()
* fadeOut()
* fadeToggle()
* fadeTo()

## jQuery fadeIn() Method

The jQuery fadeIn() method is used to fade in a hidden element.

**Syntax:**

$(*selector*).fadeIn(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeIn() method with different parameters:

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").fadeIn();

$("#div2").fadeIn("slow");

$("#div3").fadeIn(3000);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeIn() with different parameters.</p>

<button>Click to fade in boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>

</body>

</html>

## jQuery fadeOut() Method

The jQuery fadeOut() method is used to fade out a visible element.

**Syntax:**

$(*selector*).fadeOut(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeOut() method with different parameters:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").fadeOut();

$("#div2").fadeOut("slow");

$("#div3").fadeOut(3000);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeOut() with different parameters.</p>

<button>Click to fade out boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>

## jQuery fadeToggle() Method

The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

If the elements are faded out, fadeToggle() will fade them in.

If the elements are faded in, fadeToggle() will fade them out.

**Syntax:**

$(*selector*).fadeToggle(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeToggle() method with different parameters:

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").fadeToggle();

$("#div2").fadeToggle("slow");

$("#div3").fadeToggle(3000);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeToggle() with different speed parameters.</p>

<button>Click to fade in/out boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div>

<br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div>

<br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>

## jQuery fadeTo() Method

The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).

**Syntax:**

$(*selector*).fadeTo(*speed,opacity,callback*);

The required speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).

The optional callback parameter is a function to be executed after the function completes.

The following example demonstrates the fadeTo() method with different parameters:

**Code:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").fadeTo("slow", 0.15);

$("#div2").fadeTo("slow", 0.4);

$("#div3").fadeTo("slow", 0.7);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeTo() with different parameters.</p>

<button>Click to fade boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

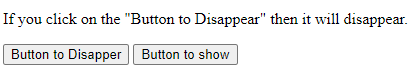
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>

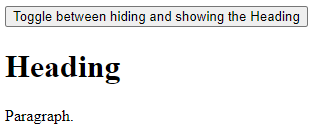
## Tasks:

Q1. Write program in jQuery that generates output as shown in the picture below:



When user clicks on Button to Disappear then the paragraph “If you click on the "Button to Disappear" will disappear with speed 2000. “If you click on the "Button to show" then the paragraph show with speed 1000.

Q2. Write program in jQuery that generates output as shown in the picture below



When user clicks on Button “Toggle between hiding and showing the Heading” then the Heading show and hide.

Q3. Write program in jQuery that uses the combination of following methods:

**fadeIn()**

**fadeOut()**

**fadeToggle()**

**fadeTo()**

**Learning outcomes:**

# Lab # 10

# jQuery Effects (Part 2)

Objectives:

To familiar with the Jquery event effects.

Theory:

# jQuery Effects - Sliding

The jQuery slide methods slide elements up and down.

With jQuery you can create a sliding effect on elements.

jQuery has the following slide methods:

* slideDown()
* slideUp()
* slideToggle()

## jQuery slideDown() Method

The jQuery slideDown() method is used to slide down an element.

**Syntax:**

$(*selector*).slideDown(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideDown() method:

## Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#flip").click(function(){

$("#panel").slideDown("slow");

});

});

</script>

<style>

#panel, #flip {

padding: 5px;

text-align: center;

background-color: pink;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<div id="flip">Click to slide down panel</div>

<div id="panel">Hello world!</div>

</body>

</html>

**Coding:**

## <!DOCTYPE html>

## <html>

## <head>

## <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

## <script>

## $(document).ready(function(){

## $("#flip").click(function(){

## $("#panel").slideUp("slow");

## });

## });

## </script>

## <style>

## #panel, #flip {

## padding: 5px;

## text-align: center;

## background-color: #e5eecc;

## border: solid 1px #c3c3c3;

## }

## #panel {

## padding: 50px;

## }

## </style>

## </head>

## <body>

## 

## <div id="flip">Click to slide up panel</div>

## <div id="panel">Hello world!</div>

## </body>

## </html>

## jQuery slideToggle() Method

The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

If the elements have been slid down, slideToggle() will slide them up.

If the elements have been slid up, slideToggle() will slide them down.

$(*selector*).slideToggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideToggle() method:

## Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#flip").click(function(){

$("#panel").slideToggle("slow");

});

});

</script>

<style>

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<div id="flip">Click to slide the panel down or up</div>

<div id="panel">Hello world!</div>

</body>

</html>

## jQuery Animations - The animate() Method

The jQuery animate() method is used to create custom animations.

**Syntax:**

$(*selector*).animate({*params*}*,speed,callback*);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the animation completes.

The following example demonstrates a simple use of the animate() method; it moves a <div> element to the right, until it has reached a left property of 250px:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("div").animate({left: '250px'});

});

});

</script>

</head>

<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>

</html>

## Tasks:

Q1. Write program in jQuery that uses Slide Up and Slide down.

Q2. Write a program in jQuery that uses the animate method.

**Learning outcomes:**

# Lab # 11

**jQuery HTML – Get & Set**

Objectives:

To familiar with the Jquery **jQuery HTML – Get & Set**.

Theory:

# jQuery - Get Content and Attributes

jQuery contains powerful methods for changing and manipulating HTML elements and attributes.

## jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM.

jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

**DOM = Document Object Model**  
  
The DOM defines a standard for accessing HTML and XML documents:  
  
*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

## Get Content - text(), html(), and val()

hree simple, but useful, jQuery methods for DOM manipulation are:

* text() - Sets or returns the text content of selected elements
* html() - Sets or returns the content of selected elements (including HTML markup)
* val() - Sets or returns the value of form fields

The following example demonstrates how to get content with the jQuery text() and html() methods:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn1").click(function(){

alert("Text: " + $("#test").text());

});

$("#btn2").click(function(){

alert("HTML: " + $("#test").html());

});

});

</script>

</head>

<body>

<p id="test">This is some <b>bold</b> text in a paragraph.</p>

<button id="btn1">Show Text</button>

<button id="btn2">Show HTML</button>

</body>

</html>

The following example demonstrates how to get the value of an input field with the jQuery val() method:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

alert("Value: " + $("#test").val());

});

});

</script>

</head>

<body>

<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>

<button>Show Value</button>

</body>

</html>

## Get Attributes - attr()

The jQuery attr() method is used to get attribute values.

The following example demonstrates how to get the value of the href attribute in a link:

Coding:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

alert($("#w3s").attr("href"));

});

});

</script>

</head>

<body>

<p><a href="https://www.w3schools.com" id="w3s">W3Schools.com</a></p>

<button>Show href Value</button>

</body>

</html>

# jQuery - Set Content and Attributes

The following example demonstrates how to set content with the jQuery text(), html(), and val() methods:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn1").click(function(){

$("#test1").text("Hello world!");

});

$("#btn2").click(function(){

$("#test2").html("<b>Hello world!</b>");

});

$("#btn3").click(function(){

$("#test3").val("Dolly Duck");

});

});

</script>

</head>

<body>

<p id="test1">This is a paragraph.</p>

<p id="test2">This is another paragraph.</p>

<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p>

<button id="btn1">Set Text</button>

<button id="btn2">Set HTML</button>

<button id="btn3">Set Value</button>

</body>

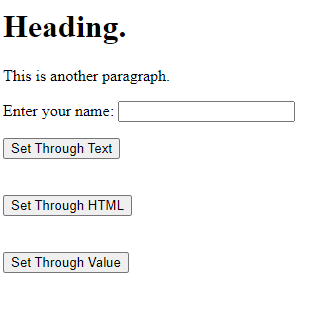
</html>

## Tasks:

Q1. Write a program that uses the combination of following three jQuery methods for DOM manipulation:

* + text()
  + html()
  + val()

.Q2. Write a program that produces output as shown in screenshot below. When your click on button **Set Through Text** then it change Heading. with .text( ) , When your click on button **Set Through HTML** then it change the paragraph with .text( ). and When your click on button **Set Through Value** then it change the paragraph with .text( ).



**Learning outcomes:**

# Lab # 12

**jQuery HTML – Add Elements & Remove Elements**

Objectives:

To familiar with the jQuery HTML – Add Elements & Remove Elements

Theory:

# jQuery - Add Elements:

With jQuery, it is easy to add new elements/content.

## Add New HTML Content

We will look at two jQuery methods that are used to add new content:

* append() - Inserts content at the end of the selected elements
* prepend() - Inserts content at the beginning of the selected elements

## jQuery append() Method

The jQuery append() method inserts content AT THE END of the selected HTML elements.

## jQuery prepend() Method

The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

**Coding:**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn1").click(function(){

$("p").append(" <b>Text Added</b>.");

});

$("#btn2").click(function(){

$("ol").append("<li>Nimco</li>");

});

$("#btn3").click(function(){

$("p").prepend("<b>Prepended text</b>. ");

});

$("#btn4").click(function(){

$("ol").prepend("<li>Prepended item</li>");

});

});

</script>

</head>

<body>

<p>Paragraph 1</p>

<p>Paragraph 2</p>

<p>Paragraph 3</p>

<ol>

<li>Tea</li>

<ul>

<li> <Green Tea</li>

<li>Black Tea </li>

</ul>

<li>Biscuit</li>

<li>Chips</li>

<li>Chocolate</li>

<li>Jelly</li>

</ol>

<button id="btn1">This is for Append to paragraph text</button> <br> <br> <br>

<button id="btn2">This is for Append to List Item </button> <br> <br> <br>

<button id="btn3">Prepend text</button> <br> <br> <br>

<button id="btn4">Prepend list item</button>

</body>

</html>

# jQuery - Remove Elements

## Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

* remove() - Removes the selected element (and its child elements)
* empty() - Removes the child elements from the selected element

## Query remove() Method

The jQuery remove() method removes the selected element(s) and its child elements.

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").remove();

});

});

</script>

</head>

<body>

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

This is some text in the div.

<p>This is a paragraph in the div.</p>

<p>This is another paragraph in the div.</p>

</div>

<br>

<button>Remove div element</button>

</body>

</html>

## jQuery empty() Method

The jQuery empty() method removes the child elements of the selected element(s).

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#div1").empty();

});

});

</script>

</head>

<body>

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

This is some text in the div.

<p>This is a paragraph in the div.</p>

<p>This is another paragraph in the div.</p>

</div>

<br>

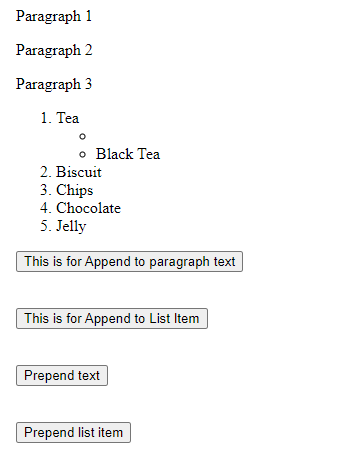
<button>Empty the div element</button>

</body>

</html>

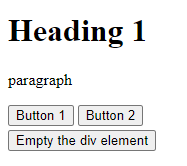
## Tasks:

Q1. Write a program that displays output as shown in screenshot below. When you click on button **“Prepended text”** then text **“Prepended text**.**”** added before Paragraphs. When you click on button **“Prepend list item”** then text **“**Prepended item**”** added before List items. When you click on button **“This is for Append to paragraph text”** then text **“Prepended text**.**”** added after Paragraphs. When you click on button **“This is for Append to List Item”** then text **“Nimco”** added after List items.

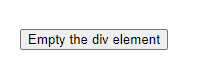


Q2. Write a program that displays output as shown in Screenshot 1 When you click on any button in screenshot then produces output as shown in Screenshot 2.

**Screenshot 1:**



**Screenshot 2:**



**Learning outcomes:**

# Lab # 13

# jQuery Traversing (Part 1)

Objectives:

# To understand the jQuery Traversing and jQuery Traversing – Ancestors.

.

Theory:

## What is Traversing?

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates an HTML page as a tree (DOM tree). With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM tree.

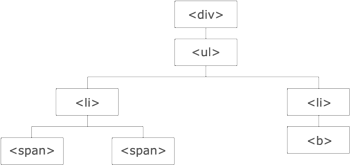


Illustration explained:

* The <div> element is the**parent** of <ul>, and an **ancestor** of everything inside of it
* The <ul> element is the **parent** of both <li> elements, and a **child** of <div>
* The left <li> element is the **parent** of <span>, **child** of <ul> and a **descendant** of <div>
* The <span> element is a **child** of the left <li> and a **descendant** of <ul> and <div>
* The two <li> elements are **siblings** (they share the same parent)
* The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of <div>
* The <b> element is a **child** of the right <li> and a **descendant** of <ul> and <div>

An ancestor is a parent, grandparent, great-grandparent, and so on. A descendant is a child, grandchild, great-grandchild, and so on. Siblings share the same parent.

## Traversing the DOM

jQuery provides a variety of methods that allow us to traverse the DOM.

The largest category of traversal methods is tree-traversal.

# jQuery Traversing - Ancestors

With jQuery you can traverse up the DOM tree to find ancestors of an element.

An ancestor is a parent, grandparent, great-grandparent, and so on.

## Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

* parent()
* parents()
* parentsUntil()

## jQuery parent() Method

The parent() method returns the direct parent element of the selected element.

This method only traverse a single level up the DOM tree.

The following example returns the direct parent element of each <span> elements:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("span").parent().css({"color": "red", "border": "2px solid red"});

});

</script>

</head>

<body>

<div class="ancestors">

<div style="width:500px;">div (great-grandparent)

<ul>ul (grandparent)

<li>li (direct parent)

<span>span</span>

</li>

</ul>

</div>

<div style="width:500px;">div (grandparent)

<p>p (direct parent)

<span>span</span>

</p>

</div>

</div>

</body>

</html>

## jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

The following example returns all ancestors of all <span> elements:

!DOCTYPE html>

<html>

<head>

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("span").parents().css({"color": "red", "border": "2px solid red"});

});

</script>

</head>

<body class="ancestors">body (great-great-grandparent)

<div style="width:500px;">div (great-grandparent)

<ul>ul (grandparent)

<li>li (direct parent)

<span>span</span>

</li>

</ul>

</div>

</body>

<!-- The outer red border, before the body element, is the html element (also an ancestor) -->

</html>

## Tasks:

## Q1. Understand the code and write your own scenario with programs that uses jQuery parent() Method and jQuery parent()s Method.

**Learning outcomes:**

# Lab # 14

# jQuery Traversing (Part 2)

Objectives:

# To understand the jQuery Traversing and jQuery Traversing Descendants.

Theory:

With jQuery you can traverse down the DOM tree to find descendants of an element.

A descendant is a child, grandchild, great-grandchild, and so on.

## Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

* children()
* find()

## jQuery children() Method

The children() method returns all direct children of the selected element.

This method only traverses a single level down the DOM tree.

The following example returns all elements that are direct children of each <div> elements:

**Coding:**

<!DOCTYPE html>

<html>

<head>

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("div").children().css({"color": "red", "border": "2px solid red"});

});

</script>

</head>

<body>

<div class="descendants" style="width:500px;">div (current element)

<p>p (child)

<span>span (grandchild)</span>

</p>

<p>p (child)

<span>span (grandchild)</span>

</p>

</div>

</body>

</html>

You can also use an optional parameter to filter the search for children.

The following example returns all <p> elements with the class name "first", that are direct children of <div>:

<!DOCTYPE html>

<html>

<head>

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("div").children("p.first").css({"color": "red", "border": "2px solid red"});

});

</script>

</head>

<body>

<div class="descendants" style="width:500px;">div (current element)

<p class="first">p (child)

<span>span (grandchild)</span>

</p>

<p class="second">p (child)

<span>span (grandchild)</span>

</p>

</div>

</body>

</html>

## Tasks:

## Q1. Write the program in jQuery that uses jQuery children () Method and returns all direct children of the selected element such as heading, paragraph, image … etc

**Learning outcomes:**