Table of Content

1. What is HTML? 2
2. Basic HTML Document Structure 2
3. Text Formatting Tags 2
4. Attributes 2
5. Links and Images 3
6. Lists 3
7. Tables 4
8. Forms 4
9. Semantic Tags 5
10. Non-Semantic Tags 5

1. What is HTML?

- **HTML** stands for **HyperText Markup Language**
- It's used to **structure content** on the web
- HTML uses **tags** to define elements like headings, paragraphs, links, images, etc.

2. Basic HTML Document Structure

html

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Page</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

- `<!DOCTYPE html>`: Declares the document type/Version
- `<html>`: Root element
- `<head>`: Contains metadata and title
- `<body>`: Contains visible content

3. Text Formatting Tags

Tag	Purpose	Example
`<h1>`	Heading	`<h1>Title</h1>`
`<p>`	Paragraph	`<p>Text here</p>`
`<strong>`,`<b>`	Bold	`<strong>Bold text</strong>`
`<em>`,`<i>`	Italic	`<em>Italic text</em>`
` `	Line Break	`Line 1 Line 2`

4. Attributes:

Attributes in HTML provide additional information about an

element, controlling its behavior, appearance, or relationship to other elements. They’re always written inside the opening tag and follow the format: `name="value"`.

Attribute	Used With	Purpose
`href`	`<a>`	Link destination
`src`	`<img>`	Image source
`alt`	`<img>`	Alternative text for accessibility
`type`	`<input>`	Specifies input type (text, email, etc.)
`value`	`<input>`	Pre-filled value
`placeholder`	`<input>`	Hint text inside field
`id`	Any element	Unique identifier
`class`	Any element	CSS styling hook
`style`	Any element	Inline CSS
`disabled`	`<input>`, `<button>`	Makes element inactive

5. Links and Images

Html

```
<a href="https://moeezmir.netlify.app”>Visit Site</a>
<img src="image.jpg" alt="Description">
```

- `<a>`: Creates a hyperlink
  - `Download` :Download the file
  - `Target=”_blank”` : open link in new tab
- `<img>`: Displays an image
  - `alt`: Describes the image for accessibility
  - `height`: give height to our image
  - `width`: give width to our image

6. Lists:

HTML lists are essential for organizing content—whether you're building a navigation menu, outlining features, or structuring quiz options.

Unordered List: Used for items where order doesn’t matter—like features, tags, or categories.

Html

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
</ul>
```

Ordered List: Used when the order of items matters—like steps in a process or rankings.

Html

```
<ol>
  <li>Step 1</li>
  <li>Step 2</li>
</ol>
```

Description List: Used for key-value pairs—like definitions, FAQs, or labeled data.

```
<dl>
  <dt>HTML</dt>
  <dd>Markup language for structuring web content</dd>

  <dt>CSS</dt>
  <dd>Stylesheet language for designing web pages</dd>
</dl>
```

7. Tables:

An HTML table is a grid-like structure created using the `<table>` tag. It's made up of rows (`<tr>`) and columns (`<td>` or `<th>`), allowing you to present data in a clean, readable format.

html

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Moeez</td>
    <td>50</td>
  </tr>
</table>
```

- `<table>`: Defines a table
- `<tr>`: Table row
- `<th>`: Table header

- `<td>`: Table data

8. Forms:

An HTML form is a structured interface that enables users to enter and submit data through various input controls. It acts as a communication bridge between the user and the backend system, allowing information to be captured, validated, and processed—whether for login, search, feedback, or data entry.

Html

```
<form action="submit.php" method="post">
  <label for="name">Name :< /label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
```

- `<form>`: Creates a form
- `action`: Where data is sent
- `method`: GET or POST
- `<input>`: Input field
- `<type>` : defines what data to store
- `<label>`: Describes input

9. Semantic Tags:

Give meaning to structure Instead of using generic `<div>` or `<span>`, semantic tags like `<header>`, `<footer>`, `<article>`, and `<section>` tell browsers and developers what each part of the page is for.

Tag	Purpose
`<header>`	Top section
`<nav>`	Navigation links
`<main>`	Main content
`<section>`	Thematic grouping
`<footer>`	Bottom section

10. Non-Semantic Tags:

Non-semantic tags are **generic containers** in HTML that don’t convey any meaning about the content inside them. They’re used

purely for layout or styling purposes, not for describing the role of the content.

Tag	Purpose
	Block-level container for layout/styling
	Inline-level container for styling/text

1. CSS
2. Include external css file

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

3. Internal styles

```
<style>
div
 { color: #444;
}
</style>
```

4. Inline styles

```
<tag style="property: value"> </tag>
```

5. Selectors

```
*              all elements
div            all div tags
div p          paragraphs inside divs
div >          pall p tags, one level deep in div
div + p        p tags immediately after div
div ~ p        p tags preceded by div
.classname          all elements with class
#idname             element with ID
div.classname       divs with certain classname
div#idname          div with certain ID
#idname             *all elements inside #idname
```

6. Pseudo classes

```
a:link          link in normal state
a:active        link in clicked state
a:hover         link with mouse over it
p::after{content:"yo";}     add content after p
p::before                   add content before p
```

```
input:checked        checked inputs
input:disabled       disabled  inputs
input:enabl          denabled  inputs
input:focus          input has focus
input:in-range       value in range
input:out-of-range   input value  out of range
input:valid          input with   valid value
input:invalid        input with   invalid value
input:optional        no required attribute
input:required        input with requred attribute
div:empty             element with no children
```

7. Media Queries :Viewport is 480 pixels or smaller

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

8. Media Queries :Viewport width smaller OR height smaller

```
@media screen and (max-width: 600px), (max-height:
500px) {}
```

9. Font

```
font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
font-style: italic;
font-variant: small-caps;
font-weight: bold;
font-size: larger;
font: style variant weight size family;
```

10. Text

```
text-align: justify;
letter-spacing: .15em;
text-decoration: underline;
word-spacing: 0.25em;
text-transform: uppercase;
```

```
line-height: normal;
```

11. Background

```
background-image: url("Path");
background-position: right top;
background-size: cover;
background-repeat: no-repeat;
background-attachment: scroll;
background-color: yellow;
background: color image repeat attachment position;
```

12. Border

```
border-width: 5px;
border-style: solid;
border-color: aqua;
border-radius: 15px;
border: width style color;
```

13. Animations

CSS animations allow one to animate transitions or other media files on the web page.

```
animation-name: myanimation;
animation-duration: 10s;
animation-timing-function: ease;
animation-delay: 5ms;
animation-iteration-count: 3;
animation-direction: normal;
animation-play-state: running;
animation-fill-mode: both;
```

14. Transitions

Transitions let you define the transition between two states of an element.

```
transition-property: none;
transition-duration: 2s;
transition-timing-function: ease-in-out;
transition-delay: 20ms;
```

15. Flexbox

Flexbox is a layout of CSS that lets you format HTML easily. Flexbox makes it simple to align items vertically and horizontally using rows and columns. Items will "flex" to different sizes to fill the space. And overall, it makes the responsive design more manageable.

```
display: flex;
flex-direction: row | row-reverse | column | column-
reverse;
flex-wrap: nowrap | wrap | wrap-reverse;
flex-flow: column wrap;
justify-content: flex-start | flex-end | center |
space-between | space-around | space-evenly | start |
end | left | right ... + safe | unsafe;
align-items: stretch | flex-start | flex-end | center
| baseline | first baseline | last baseline | start |
end | self-start | self-end + ... safe | unsafe;
align-content: flex-start | flex-end | center |
space-between | space-around | space-evenly | stretch
| start | end | baseline | first baseline | last
baseline + ... safe | unsafe;
```

16. Child Properties (flex items)

```
order: 5; /* default is 0 */
flex-grow: 4; /* default 0 */
flex-shrink: 3; /* default 1 */
flex-basis: | auto; /* default auto */
```

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
align-self: auto | flex-start | flex-end | center | baseline | stretch;
```

17. Grid

Grid layout is a 2-Dimensional grid system to CSS that creates complex responsive web design layouts more easily and consistently across browsers.

```
display: grid | inline-grid;
grid-template-columns: 12px 12px 12px;
grid-template-rows: 8px auto 12px;
grid-template: none | grid-template-rows / grid-template-columns;
column-gap: line-size;
row-gap: line-size;
justify-items: start | end | center | stretch;
align-items: start | end | center | stretch;
place-items: center;
justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
align-content: start | end | center | stretch | space-around | space-between | space-evenly;
grid-auto-flow: row | column | row dense | column dense;
```

18. Properties

```
font-family     font of the element
font-style      font style: normal, italic, oblique
font-variant    set small-caps
font-weight     use bold or thin characters
margin-bottom  bottom margin
margin-left     left margin
```

```
margin-right    right margin
margin-top      margin top
max-height      maximum height of element
max-width       maximum width of element
min-height      minimum height
min-width       minimum width
outline-offset gap between the element and the
outline
outline-style outline style
outline-width  outline width
overflowhide    display or scroll if the content
overflows its container
overflow-x      horizontal overflow
overflow-y      vertical overflow
paddingpadding between the element border and content
padding-bottom padding bottom
padding-left padding left
padding-right padding right
padding-top    padding top
text-overflow the way how overflowed content is
marked (ellipsis)
text-shadow    text shadow
visibility     visibility:hidden elements leave a gap
white-space    how are white-spaces handled
width          width of an element
word-break     text breaking rules when text reaches
the end of the container
word-spacing  size of white space between words
word-wrap      break long words and wrap onto the next
line
z-index        stack order of the element
```

1. Console in js

```
document.write("hello world");
document.write(2 + 2);
document.write(["irtiza", "umaisa",
"saniya"]);//Array
document.write([22, 1, 44, 55, 66]); //Array
//OBJECTS
document.write({
    name: "irtiza",
    marks: 77
});
console.warn('this is for warning');
console.error("Error messages");
console.clear();  //clear console
```

2. Document .write in js

```
document.write("hello world");
document.write(2 + 2);
document.write(["irtiza", "umaisa",
"saniya"]);//Array
document.write([22, 1, 44, 55, 66]); //Array
//OBJECTS
document.write({
    name: "irtiza",
    marks: 77
});
```

3. Let & Const

```
// let & const
let age;
age = 2;
age = 33;
document.write(age);
```

```
const section = "9th";
// section = "10th" this will through an error
document.write(section);
```

4. Datatype

```
//DATA TYPES PREMETIVE AND REFRENCE DATA TYPE
//PREMETIVE
let string = "this is a string";
document.write(string, typeof (string));
let number = 33;
document.write(number, typeof (number));
let booLean = true;
document.write("Data type is " + (typeof booLean));
let nullVar = null;
document.write("Data type is ", typeof (nullVar));
let undef = undefined;
document.write("Data type is " + (typeof undef));
// Reference Data Types
let myarr = [1, 2, 3, 4, false, "string"];
document.write("Data type is " + (typeof myarr));
// Object Literals
let stMarks = {
    harry: 89,
    Shubham: 36,
    Rohan: 34
}
document.write(typeof stMarks);
//FUNCTION
function findName() {
}
document.write(typeof findName);
//DATE
let date = new Date();
document.write(typeof date);
```

5. Type conversion

```
let myVAr=10;
document.write(myVAr, (typeof myVAr));
//method one String() Number()
let myVArr = String(10);
document.write(myVArr, (typeof myVArr));
// method second toString
let myVarrr=11;
document.write(myVarrr.toString());
// method third parseInt parseFloat
let number = parseInt('10.11');//output 10
number = parseFloat('10.11');//output 10.11
document.write(number, (typeof number));
//toFixed() used to show decimal
// TYPE COERSION
let coersion = '10';
let coersion2 = 20;
document.write(coersion + coersion2);//addition
concatinate
```

6. Strings

```
 //strings
let namee = "irtiza";
nickName = "meezan";
//concatinate method one
document.write(namee + nickName);
//concattinate method 2
document.write(namee.concat(" is my name"));
//functions of strings
let newName = "functions of strings";
document.write(newName);
document.write(newName.length);
```

```
document.write(newName.toLowerCase());
document.write(newName.toUpperCase());
//index of
document.write(newName[0]);
document.write(newName.charAt(0));
document.write(newName.indexOf('s'));
document.write(newName.lastIndexOf("s"));
//check
document.write(newName.endsWith('strings'));
//returns boolean
document.write(newName.includes('strings'));
//returns boolean
document.write(newName.substring(0, 9)); //return
strings from 0 to 9
document.write(newName.slice(0, 9)); //from start
document.write(newName.slice(-4)); //from end
document.write(newName.split(' '));//make an array
document.write(newName.replace('strings',
'replaced'));
//Template litrals
let myHtml = ` hello ${newName} template litrals `;
document.body.innerHTML = myHtml;
```

7. Arrays

```
let marks = [33, 22, 44, 66, 77, 99];
const fruits = ['orange', 'apple', 'banana'];
const mixed = [55, 'strings', [3, 8]];
//array constructor
const arr = new Array(23, 44, 55, 77, 'yeap buddy');
//index starts from 0
document.write(marks[2]); //output 44
//property & method
document.write(arr.length);
//check array is present
```

```
document.write(Array.isArray(arr)); //return boolean
arr[1] = 'irtiza';
document.write(arr);
//index of
let value = marks.indexOf(33);
document.write(value);
//mutating or modifying arrays
marks.push(34); //push at the end
marks.unshift(99); //push at the starting
marks.pop();//delete end element
marks.shift(); //delete start element
marks.splice(1, 2);//takes two paramter start and how
many i.e delete element from 1 to two more
marks.reverse();//reverse of an array
let marks2 = [1, 2, 3, 4, 5];
marks  =  marks.concat(marks2);
document.write(marks);
//OBJECTS
let myObj = {
    name: 'faizan', //key : values
    channel: 'wreckeverthing',
    isActive: true,
    markas: [22, 22, 222, 11, 12]
}
document.write(myObj); //show obj
//access objects with two methods
//Method one
document.write(myObj.markas); //show name inobj
document.write(myObj['markas']); //show markas inobj
//Method two
document.write(myObj.name); //show name in obj
document.write(myObj['name']); //show name in obj
```

8. if Else and Or And

```
const age = 23;
if (age == 13) { //simple condition check
    document.write("Age is 23");
}
else if (age == 22) { //nested if else
    document.write("Age is 22");
}
else if (age === 23) { //condition and datatype check
    document.write("Age is 23");
}
else if (age != 23) { //not equal to
    document.write("Age if not 23");
}
else if (age !== 23) { //not equal to and check
datatype
    document.write("Age if not 23");
}
else {
    document.write("Age is not 23 or 22");
}
//boolean
const driver = true;
if (driver) {
    document.write("driver is present");
}
else {
    document.write("driver is absent");
}
//AND && OR ||
if (driver && age) {//AND = both condition should be
true
    document.write("driver and age is defined");
}
```

```
if (driver || khaberkos) {  //OR = atleast one
condition  should be true

    document.write("one is defined");
}
               }
```

9. Ternary operator

```
const irtiza = 22;
document.write(irtiza == 22 ? 'irtiza is 22' :
'irtiza is not 22');
```

10. Switch cases

```
switch (irtiza) {
case 20:
document.write("irtiza  is 20");
break;
case 21:
document.write("irtiza  is 21");
break;
case 22:
document.write("irtiza  is 22");
break;
default:
document.write("age is unknown");
break;
```

11. Loops for,while,dowhile,foreach,forin

```
      let a = 1;
while (a < 10) { //while loop
    document.write(a);
    a++;
}
for (a = 10; a < 20; a++) { //for loop
```

```
    document.write(a);
}
do { //do-while loop
    document.write(a)
    a++;
} while (a < 10);
//foreach
let arr = [1, 2, 3, 4, 5];
arr.forEach(function (element) //element,index,array
can also be used
{
    document.write(element);
});
//break; to break a loop
//continue to skip a loop
let obj = {
    name: 'irtiza',
    age: 11,
    os: 'kali-linux',
    pro: 'professional'
}
```