UNIT No 1

Unit Name : Working with HTML5 new tags and elements to enhance the structure and functionality of web pages:

HTML5 | Introduction

Introduction: HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is an abbreviation of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text document within the tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

Features:

- It has introduced new multimedia features which supports both audio and video controls by using <audio> and <video> tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including <header> <footer>, <article>, <section> and <figure> are added.
- Drag and Drop- The user can grab an object and drag it further dropping it to a new location.
- Geo-location services- It helps to locate the geographical location of a client.

- Web storage facility which provides web application methods to store data on the web browser.
- Uses SQL database to store data offline.
- Allows drawing various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e., <!doctype html>
- Easy character encoding i.e., <meta charset="UTF-8">

Removed elements from HTML 5: There are many elements which are depreciated from HTML 5 are listed below:

Removed Elements	Use Instead Elements
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	

<frameset>	
<noframes>	
<isindex>	
<strike>	CSS, <s> or <del>
<tt>	CSS

New Added Elements in HTML 5:

- **<article>:** The <article> tag is used to represent an article. More specifically, the content within the <article> tag is independent from the other content of the site (even though it can be related).
- **<aside>:** The <aside> tag is used to describe the main object of the web page in a shorter way like a highlighter. It basically identifies the content that is related to the primary content of the web page but does not constitute the main intent of the primary page. The <aside> tag contains mainly author information, links, related content and so on.
- **<figcaption>:** The <figcaption> tag in HTML is used to set a caption to the figure element in a document.
- **<figure>:** The <figure> tag in HTML is used to add self-contained content like illustrations, diagrams, photos or codes listing in a document. It is related to main flow, but it can be used in any position of a document and the figure goes with the flow of the

document and if it is removed it should not affect the flow of the document.

- **<header>:** It contains the section heading as well as other content, such as a navigation links, table of contents, etc.
- **<footer>:** The <footer> tag in HTML is used to define a footer of HTML document. This section contains the footer information (author information, copyright information, carriers etc.). The footer tag is used within body tag. The <footer> tag is new in the HTML 5. The footer elements require a start tag as well as an end tag.
- **<main>:** Delineates the main content of the body of a document or web app.
- **<mark>:** The <mark> tag in HTML is used to define the marked text. It is used to highlight the part of the text in the paragraph.
- **<nav>:** The <nav> tag is used to declaring the navigational section in HTML documents. Websites typically have sections dedicated to navigational links, which enables user to navigate the site. These links can be placed inside a nav tag.
- **<section>:** It demarcates a thematic grouping of content.
- **<details>:** The <details> tag is used for the content/information which is initially hidden but could be displayed if the user wishes to see it. This tag is used to create interactive widget which user can open or close it. The content of details tag is visible when open the set attributes.
- **<summary>:** The <summary> tag in HTML is used to define a summary for the <details> element. The <summary> element is used

along with the <details> element and provides a summary visible to the user. When the summary is clicked by the user, the content placed inside the <details> element becomes visible which was previously hidden. The <summary> tag was added in HTML 5. The <summary> tag requires both starting and ending tag.

- **<time>:** The <time> tag is used to display the human-readable data/time. It can also be used to encode dates and times in a machine-readable form. The main advantage for users is that they can offer to add birthday reminders or scheduled events in their calendars and search engines can produce smarter search results.
- **<bdi>:** The <bdi> tag refers to the Bi-Directional Isolation. It differentiates a text from other text that may be formatted in different direction. This tag is used when a user generated text with an unknown direction.
- **<wbr>:** The <wbr> tag in HTML stands for word break opportunity and is used to define the position within the text which is treated as a line break by the browser. It is mostly used when the used word is too long and there are chances that the browser may break lines at the wrong place for fitting the text.
- **<datalist>:** The <datalist> tag is used to provide autocomplete feature in the HTML files. It can be used with input tag, so that users can easily fill the data in the forms using select the data.
- **<keygen>:** The <keygen> tag in HTML is used to specify a key-pair generator field in a form. The purpose of <keygen> element is to provide a secure way to authenticate users. When a form is

submitted then two keys are generated, private key and public key. The private key stored locally, and the public key is sent to the server. The public key is used to generate client certificate to authenticate user in future.

- **<output>:** The <output> tag in HTML is used to represent the result of a calculation performed by the client-side script such as JavaScript.
- **<progress>:** It is used to represent the progress of a task. It also defines how much work is done and how much is left to download a task. It is not used to represent the disk space or relevant query.
- **<svg>:** It is the Scalable Vector Graphics.
- **<canvas>:** The <canvas> tag in HTML is used to draw graphics on web page using JavaScript. It can be used to draw paths, boxes, texts, gradient and adding images. By default, it does not contain border and text.
- **<audio>:** It defines the music or audio content.
- **<embed>:** Defines containers for external applications (usually a video player).
- **<source>:** It defines the sources for <video> and <audio>.
- **<track>:** It defines the tracks for <video> and <audio>.
- **<video>:** It defines the video content.

Advantages:

- All browsers supported.

- More device friendly.
- Easy to use and implement.
- HTML 5 in integration with CSS, JavaScript, etc. can help build beautiful websites.

Disadvantages:

- Long codes have to be written which is time consuming.
- Only modern browsers support it.

Example 1:

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML 5</title>
   <style>
      h1 {
         font-size:50px;
      }
   </style>
</head>
<body>
   <h1>Web Technology</h1>
</body>

</html>
```

Example 2:

```
<!DOCTYPE html>
<html>

<head>
  <title>HTML 5 Demo</title>

  <style>
    .GFG {
      font-size:40px;
      font-weight:bold;
      color:green;
    }
    body {
      text-align:center;
    }
  </style>
</head>

<body>
  <div class = "GFG">WT</div>
  <aside>
    <div>Web Technology with UIUX</div>
  </aside>
</body>

</html>
```

Difference between HTML and HTML5

HTML stands for **Hyper Text Markup Language**. It is used to design web pages using a markup language. HTML is a combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate (at the

note for the computer) text so that a machine can understand it and manipulate text accordingly.

Features of HTML:

- It allows the creation of hyperlinks with the `<a>` tag, connecting different web pages.
- Uses tags to mark elements and content, such as headings (`<h1>` to `<h6>`).
- It supports embedding images (`<img>`), videos (`<video>`), and audio (`<audio>`) for multimedia content.
- It provides form elements like `<form>`, `<input>`, and `<button>` for user input and data submission.
- Semantic tags like `<article>`, `<section>`, and `<nav>` for better document structure and accessibility.

HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM). It has introduced various new features like drag and drop, geo-location services

Features of HTML5:

- Introduced new semantic elements like `<header>`, `<footer>`, `<section>`, and `<article>` for improved structure.
- Enhances multimedia capabilities with native support for audio and video elements.

- Provides the `localStorage` API, allowing web applications to store data locally on the user's device.
- Enables websites to access a user's geographical location.
- Uses SQL database to store data offline.

HTML	HTML5
It didn't support audio and video without the use of flash player support.	It supports audio and video controls with the use of <audio> and <video> tags.
It uses cookies to store temporary data.	It uses SQL databases and application cache to store offline data.
Does not allow JavaScript to run in the browser.	Allows JavaScript to run in the background. This is possible due to JS Web worker API in HTML5.
Vector graphics are possible in HTML with the help of various technologies such as VML, Silver-light, Flash, etc.	Vector graphics are additionally an integral part of HTML5 like SVG and Canvas.
It does not allow drag and drop effects.	It allows drag and drop effects.
Not possible to draw shapes like circle, rectangle, triangle etc.	HTML5 allows to draw shapes like circle, rectangle, triangle etc.
It works with all old browsers.	It supported by all new browser like Firefox, Mozilla, Chrome, Safari, etc.

<HTML>,<Body> , and <Head> tags are mandatory while writing a HTML code.	These tags can be omitted while writing HTML code.
Older version of HTML are less mobile-friendly.	HTML5 language is more mobile-friendly.
Doctype declaration is too long and complicated.	Doctype declaration is quite simple and easy.
Elements like nav, header were not present.	New element for web structure like nav, header, footer etc.
Character encoding is long and complicated.	Character encoding is simple and easy.
It is almost impossible to get true GeoLocation of user with the help of browser.	One can track the GeoLocation of a user easily by using JS GeoLocation API.
It can not handle inaccurate syntax.	It is capable of handling inaccurate syntax.
Being an older version , it is not fast , flexible , and efficient as compared to HTML5.	It is efficient, flexible and more fast in comparison to HTML.
Attributes like charset, async and ping are absent in HTML.	Attributes of charset, async and ping are a part of HTML 5.

There are many HTML elements which have been modified or removed from HTML5. Some of them are listed below:

Element	In HTML5
<applet>	Changed to <object>

<acronym>	Changed to <abbr>
<dir>	Changed to
<frameset>	Removed
<frame>	Removed
<noframes>	Removed
<strike>	No new tag. CSS is used for this
<big>	No new tag. CSS is used for this
<basefont>	No new tag. CSS is used for this
<font>	No new tag. CSS is used for this
	No new tag. CSS is used for this
<tt>	No new tag. CSS is used for this

Many new elements are added in HTML5 like nav, audio, figcaption, progress, command, time, datalist, video, figure, meter, data, section, time, aside, canvas, summary, rp, rt, details, wbr, header, footer, keygen, embed, article, hgroup, bdi, mark, output, source, track, section, ruby and many more.

HTML Semantic Elements

Semantic elements = elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: `<div>` and `<span>` - Tells nothing about its content.

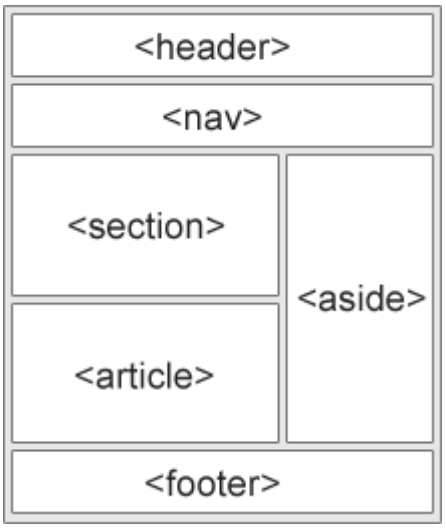
Examples of semantic elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Semantic Elements in HTML

Many web sites contain HTML code like: to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML <section> Element

The `<section>` element defines a section in a document.

"A section is a thematic grouping of content, typically with a heading."

Examples of where a `<section>` element can be used:

- Chapters
- Introduction
- News items
- Contact information

A web page could normally be split into sections for introduction, content, and contact information.

Example

Two sections in a document:

```
<section>
<h1>WWF</h1>
```

```
<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>
<section>
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.</p>
</section>
```

HTML <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.

Examples of where the <article> element can be used:

- Forum posts
- Blog posts
- User comments
- Product cards
- Newspaper articles

Example

Three articles with independent, self-contained content:

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p>
</article>
<article>
<h2>Mozilla Firefox</h2>
```

```
<p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p>
</article>

<article>
<h2>Microsoft Edge</h2>
<p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p>
</article>
```

Example 2

Use CSS to style the <article> element:

```
<html>
<head>
<style>
.all-browsers {
  margin: 0;
  padding: 5px;
  background-color: lightgray;
}

.all-browsers > h1, .browser {
  margin: 10px;
  padding: 5px;
}
.browser {
  background: white;
}
.browser > h2, p {
  margin: 4px;
  font-size: 90%;
}
</style>
</head>
<body>
```

```
<article class="all-browsers">
  <h1>Most Popular Browsers</h1>
  <article class="browser">
    <h2>Google Chrome</h2>
     <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p>
  </article>
  <article class="browser">
    <h2>Mozilla Firefox</h2>
      <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p>
  </article>
  <article class="browser">
    <h2>Microsoft Edge</h2>
    <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p>
  </article>
</article>
</body>
</html>
```


Nesting <article> in <section> or Vice Versa?

The `<article>` element specifies independent, self-contained content.

The `<section>` element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with `<section>` elements containing `<article>` elements, and `<article>` elements containing `<section>` elements.

HTML <header> Element

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

Note: You can have several <header> elements in one HTML document. However, <header> cannot be placed within a <footer>, <address> or another <header> element.

Example

A header for an <article>:

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
   <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML <footer> Element

The <footer> element defines a footer for a document or section.

A <footer> element typically contains:

- authorship information

- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several `<footer>` elements in one document.

Example

A footer section in a document:

```
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

HTML <nav> Element

The `<nav>` element defines a set of navigation links.

Notice that NOT all links of a document should be inside a `<nav>` element. The `<nav>` element is intended only for major blocks of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Example

A set of navigation links:

```
<nav>
```

```
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

HTML <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be indirectly related to the surrounding content.

Example

Display some content aside from the content it is placed in:

```
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>
<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
```

Example 2

Use CSS to style the <aside> element:

```
<html>
<head>
<style>
```

```
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style>
</head>
<body>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>
<aside>
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>
</body>
</html>
```

HTML <figure> and <figcaption> Elements

The `<figure>` tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

The `<figcaption>` tag defines a caption for a `<figure>` element. The `<figcaption>` element can be placed as the first or as the last child of a `<figure>` element.

The `<img>` element defines the actual image/illustration.

Example

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

"A semantic Web allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.

Tag	Description
<article>	Defines independent, self-contained content
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links

<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

HTML Multimedia

Multimedia on the web is sound, music, videos, movies, and animations. Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more. Web pages often contain multimedia elements of different types and formats. Multimedia elements (like audio or video) are stored in media files. The most common way to discover the type of a file, is to look at the file extension. Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

The HTML <video> Element

To show a video in HTML, use the <video> element:

```
<video width="320" height="240" controls autoplay muted> <source src="movie.mp4" type="video/mp4"> <source src="movie.ogg" type="video/ogg"> Your browser does not support the video tag. </video>
```

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

To start a video automatically, use the autoplay attribute. Add muted after autoplay to let your video start playing automatically (but muted).

The HTML <audio> Element

To play an audio file in HTML, use the <audio> element:

```
<audio controls autoplay muted>  <source src="horse.ogg"
type="audio/ogg">  <source src="horse.mp3" type="audio/mpeg"> Your
browser does not support the audio element. </audio>
```

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

To start an audio file automatically, use the autoplay attribute. Add muted after autoplay to let your audio file start playing automatically (but muted).

Adding youtube videos:

Youtube videos can also be directly added to your web page using the embed video option on any youtube video. **<iframe>** element is used to add the video with the src attribute but a shortcut to that is simply to go to any youtube video and right-click on the video, then select the copy embed code option.

Example 3: In this code we will see how to add youtube videos with the help of iframe tag in an HTML document.

```
<!DOCTYPE html>
<html>
<head>
   <title>Page Title</title>
</head>
```

```
<body>

            <iframe width="942" height="530" src=
"https://www.youtube.com/embed/" title= "Introduction to
Python | Sample Video for Python Foundation Course"
frameborder="0" allow= "accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope;
picture-in-picture; web-share" allowfullscreen>

   </iframe>

</body>

</html>
```

<embed>:

It is used as a container for embedding plug-ins such as flash animations.

Syntax:

```
<embed attributes>
```


Example 4: In this example, we will use <embed> tag with an example.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: green;
    }
  </style>
</head>
<body>
  <p>Embed Sample</p>
```

```
  <embed src=
"https://media.org/wp-content/uploads/20210723103530/simplescreesdfdsrecorder2021071.gif" width="300px" height="300px">
</body>
</html>
```

<source>:

As you can observe that <audio>, <video> elements contain the <source> element, the <source> tag is used to attach multimedia files.

Syntax:

```
<source src="" type="">
  ...
</source>
```

Example 5: In this example, we will use <source> tag with an example.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>
<body>
    <center>
        <h2>Source Tag</h2>
        <audio controls>
```

```
            <source src="audio.mp3"
                    type="audio/mp3">
        </audio>
    </center>
</body>
</html>
```

<track>:

It is used to specify subtitles, caption files, or different files containing text, that ought to be visible once the media is taking part in it.Thus it is a simple sector for the <audio> and <video> elements.

Syntax:

```
<track Attribute>
```

Example 6: In this example, we will use <track> tag with an example.

```
<!DOCTYPE html>
<html>

<head>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>
<body>
    <h2>Track Tag: Both Audio and Video</h2>
    <video width="300" height="300" controls>
```

```
            <source src="myvid.mp4" type="video/mp4">
            <track src=
"https://media.org/wp-content/uploads/11.mp4"
                   kind="subtitle" srclang="en"
                      label="English">
      </video>
</body>
</html>
```

The HTML <form> Elements

The HTML <form> element can contain one or more of the following form elements:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

The <input> Element

One of the most used form elements is the <input> element.

The <input> element can be displayed in several ways, depending on the type attribute.

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

HTML Input Types

Here are the different input types you can use in HTML:

`<input type="text">` defines a single-line text input field.

`<input type="password">` defines a password field.

`<input type="submit">` defines a button for submitting form data to a form handler. The form-handler is typically a server page with a script for processing input data. The form-handler is specified in the form's `action` attribute.

`<input type="reset">` defines a reset button that will reset all form values to their default values.

`<input type="radio">` defines a radio button.Radio buttons let a user select ONLY ONE of a limited number of choices.

`<input type="checkbox">` defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

`<input type="button">` defines a button.

The `<input type="color">` is used for input fields that should contain a color. Depending on browser support, a color picker can show up in the input field.

The `<input type="date">` is used for input fields that should contain a date. Depending on browser support, a date picker can show up in the input field.

The `<input type="datetime-local">` specifies a date and time input field, with no time zone. Depending on browser support, a date picker can show up in the input field.

The `<input type="email">` is used for input fields that should contain an e-mail address.Depending on browser support, the e-mail address can be automatically validated when submitted.Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

The `<input type="hidden">` defines a hidden input field (not visible to a user).A hidden field lets web developers include data that cannot be seen or

modified by users when a form is submitted. A hidden field often stores what database record that needs to be updated when the form is submitted.

The `<input type="month">` allows the user to select a month and year. Depending on browser support, a date picker can show up in the input field.

The `<input type="number">` defines a numeric input field. You can also set restrictions on what numbers are accepted with `min and maxu` attribute.

The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes

The `<input type="tel">` is used for input fields that should contain a telephone number.

The `<input type="time">` allows the user to select a time (no time zone). Depending on browser support, a time picker can show up in the input field.

The `<input type="url">` is used for input fields that should contain a URL address. Depending on browser support, the url field can be automatically validated when submitted. Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

The `<input type="week">` allows the user to select a week and year. Depending on browser support, a date picker can show up in the input field.

The `placeholder` attribute specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The `required` attribute is a boolean attribute.When present, it specifies that an input field must be filled out before submitting the form. The `autocomplete` attribute specifies if browsers should try to predict the value of an input field or not. You can also specify which type of value you expect in the input field.

Here is Example using all input types :

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>The input Element</h2>
<form action="/action_page.php">
 <label for="fname">First name:</label><br>
    <input    type="text"    id="fname"    name="fname"    required
placeholder="Your  name"><br>
<label for="username">Username:</label><br>
 <input type="text" id="username" name="username" autocomplete><br>
input   type="radio"   id="html"   name="fav_language"   value="HTML">
<label for="html">HTML</label><br>
  <input   type="radio"   id="css"   name="fav_language"   value="CSS">
<label for="css">CSS</label><br>
    <input    type="radio"    id="javascript"    name="fav_language"
value="JavaScript">
 <label for="javascript">JavaScript</label>
<input  type="checkbox"  id="vehicle1"  name="vehicle1"  value="Bike">
<label for="vehicle1"> I have a bike</label><br>
  <input  type="checkbox"  id="vehicle2"  name="vehicle2"  value="Car">
<label for="vehicle2"> I have a car</label><br>
  <input  type="checkbox"  id="vehicle3"  name="vehicle3"  value="Boat">
<label for="vehicle3"> I have a boat</label>
<input  type="button"  onclick="alert('Hello  World!')"  value="Click
Me!">
<label  for="favcolor">Select  your  favorite  color:</label>   <input
type="color" id="favcolor" name="favcolor">

<label for="birthday">Birthday:</label>
 <input type="date" id="birthday" name="birthday">
<label  for="datemax">Enter  a  date  before  1980-01-01:</label>  <input
type="date" id="datemax" name="datemax" max="1979-12- 31"><br><br>
 <label  for="datemin">Enter  a  date  after  2000-01-01:</label>  <input
type="date" id="datemin" name="datemin" min="2000-01-02">
<label  for="birthdaytime">Birthday  (date  and  time):</label>  <input
type="datetime-local" id="birthdaytime" name="birthdaytime">
<label for="email">Enter your email:</label>
 <input type="email" id="email" name="email">
<input  type="image"  src="img_submit.gif"  alt="Submit"  width="48"
height="48">
```

```
<label for="myfile">Select a file:</label>
 <input type="file" id="myfile" name="myfile">
<input type="hidden" id="custId" name="custId" value="3487">
<label  for="bdaymonth">Birthday  (month  and  year):</label>   <input
type="month" id="bdaymonth" name="bdaymonth">
<label  for="quantity">Quantity  (between  1  and  5):</label>   <input
type="number" id="quantity" name="quantity" min="1" max="5">
<label for="quantity">Quantity:</label>
 <input type="number" id="quantity" name="quantity" min="0" max="100"
step="10" value="30">
<label for="vol">Volume (between 0 and 50):</label>
 <input type="range" id="vol" name="vol" min="0" max="50">
<label for="phone">Enter your phone number:</label>
        <input        type="tel"        id="phone"        name="phone"
pattern="[0-9]{3}-[0-9]{2}- [0-9]{3}">
<label for="appt">Select a time:</label>

 <input type="time" id="appt" name="appt">
<label for="homepage">Add your homepage:</label>
 <input type="url" id="homepage" name="homepage">
<label for="week">Select a week:</label>
 <input type="week" id="week" name="week">
 <input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

The <select> Element

The `<select>` element defines a drop-down list. The `<option>` element defines an option that can be selected. By default, the first item in the drop-down list is selected. To define a pre-selected option, add the `selected` attribute to the option.Use the `size` attribute to specify the number of visible values. Use the
`multiple` attribute to allow the user to select more than one value.

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="3">
 <option value="volvo" selected>Volvo</option>
 <option value="saab">Saab</option>
 <option value="fiat">Fiat</option>
 <option value="audi">Audi</option>
</select>
```

The <textarea> Element

The `<textarea>` element defines a multi-line input field (a text area).The `rows` attribute specifies the visible number of lines in a text area. The `cols` attribute specifies the visible width of a text area.

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

The <fieldset> and <legend> Elements

The `<fieldset>` element is used to group related data in a form. The `<legend>` element defines a caption for the `<fieldset>` element.

```
<form action="/action_page.php">
 <fieldset>
 <legend>Personalia:</legend>
 <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
<label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
<input type="submit" value="Submit">
 </fieldset>
</form>
```

HTML Canvas Graphics

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

What is HTML Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Canvas is supported by all major browsers.

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

Here is an example of a basic, empty canvas:

Example

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">


</canvas>
```

Add a JavaScript

After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

Here are some examples:

Draw a Line

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
</script>
```

Draw a Circle

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
```

```
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
</script>
```

Draw a Text

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
```

Stroke Text

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
</script>
```

Draw Linear Gradient

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
```

```
// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

Draw Circular Gradient

Example

```
  <script>
  var c = document.getElementById("myCanvas");
  var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

Draw Image

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
</script>
```

HTML SVG Graphics

SVG (Scalable Vector Graphics)

SVG defines vector-based graphics in XML, which can be directly embedded in HTML pages.

SVG graphics are scalable, and do not lose any quality if they are zoomed or resized:

SVG is supported by all major browsers.

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines graphics in XML format
- Each element and attribute in SVG files can be animated
- SVG integrates with other standards, such as CSS, DOM, XSL and JavaScript

---

The <svg> Element

The HTML `<svg>` element is a container for SVG graphics.

SVG has several methods for drawing paths, rectangles, circles, polygons, text, and much more.

SVG Circle

Example

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
     <circle   cx="50"   cy="50"   r="40"   stroke="green"   stroke-width="4"
fill="yellow" />
</svg>

</body>
</html>
```

SVG Rectangle

Example

```
<svg width="400" height="120">
  <rect x="10" y="10" width="200" height="100" stroke="red" stroke-width="6"
fill="blue" />
</svg>
```

SVG Rectangle with Opacity and Rounded Corners

Example

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

SVG Star

Example

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Gradient Ellipse and Text

SVG

Example

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1">
      <stop offset="0%" stop-color="yellow" />
      <stop offset="100%" stop-color="red" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
     <text   fill="#ffffff"   font-size="45"   font-family="Verdana"   x="50"
y="86">SVG</text>
  Sorry, your browser does not support inline SVG.
</svg>
```

Differences Between SVG and Canvas

SVG is a language for describing 2D graphics in XML, while Canvas draws 2D graphics, on the fly (with JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers to SVG graphics.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of SVG and Canvas

The table below shows some important differences between Canvas and SVG:

SVG	Canvas
• Resolution independent • Support for event handlers • Good text rendering capabilities • Slow rendering if complex • Not suited for game applications	• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games

Grail Layout

Holy Grail is a layout pattern that's very common on the web. It consists of a header, a main content area with fixed-width navigation on the left, content in the middle and a fixed-width sidebar on the right and then a footer.

Holy Grail has been achieved using variety of methods, probably most notably recently with Flexbox. CSS Grid Layout is yet another method, and it should prove to be the most appropriate and straightforward way when browser support gets better. It was designed especially to easily accomplish full page layouts.

For the classic holy grail layout, there is a header, footer, left sidebar, right sidebar, and main content. It's similar to the previous layout, but now with sidebars!

To write this entire grid using a single line of code, use the grid-template property. This enables you to set both the rows and columns at the same time.

The property and value pair is: grid-template: auto 1fr auto / auto 1fr auto. The slash between the first and second space-separated lists is the break between rows and columns.

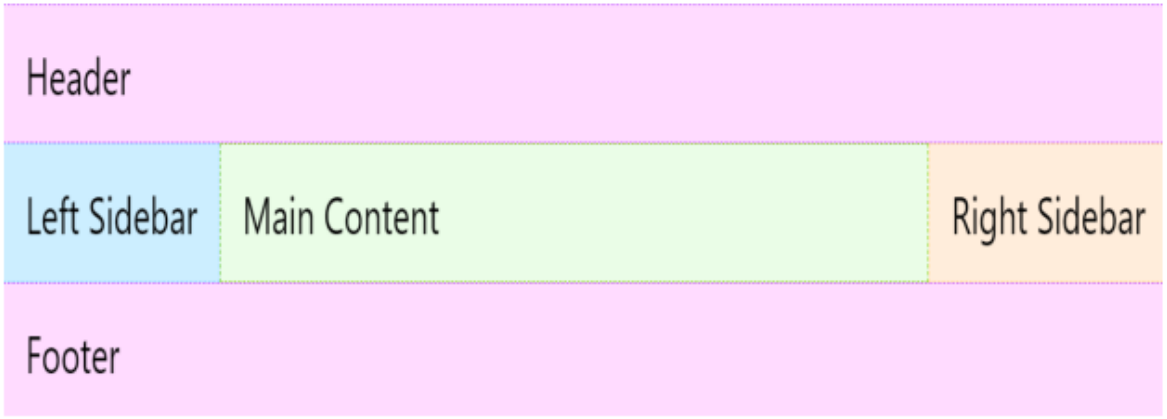
```
.parent {

  display: grid;

  grid-template: auto 1fr auto / auto 1fr auto;

}
```

As in the last example, where the header and footer had auto-sized content, here the left and right sidebar are automatically sized based on their children's intrinsic size. However, this time it is horizontal size (width) instead of vertical (height).



```
<div class="parent">
  <header class="section coral">Header</header>
  <div class="left-side section blue">Left Sidebar</div>
  <main class="section green"> Main Content</main>
  <div class="right-side section yellow">Right Sidebar</div>
  <footer class="section coral">Footer</footer>
</div>
```

GIT Command

SETUP & INIT Configuring user information, initializing and cloning repositories

- git init :initialize an existing directory as a Git repository
- git clone [url]: retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT Working with snapshots and the Git staging area

- git status :show modified files in working directory, staged for your next commit
- git add [file] : add a file as it looks now to your next commit (stage)
- git reset [file] : unstage a file while retaining the changes in working directory
- git diff : diff of what is changed but not staged
- git diff –staged : diff of what is staged but not yet commited
- git commit -m "[descriptive message]": commit your staged content as a new commit snapshot

BRANCH & MERGE Isolating work in branches, changing context, and integrating changes

- git branch :list your branches. a * will appear next to the currently active branch
- git branch [branch-name] :create a new branch at the current commit
- git checkout :switch to another branch and check it out into your working directory
- git merge [branch] :merge the specified branch's history into the current one
- git log : show all commits in the current branch's history

SHARE & UPDATE Retrieving updates from another repository and updating local

- git remote add [alias] [url] :add a git URL as an alias
- git fetch [alias] fetch :down all the branches from that Git remote
- git merge [alias]/[branch] :merge a remote branch into your current branch to bring it up to date
- git push [alias] [branch] :Transmit local branch commits to the remote repository branch
- git pull :fetch and merge any commits from the tracking remote branch

1. Create a GitHub Account:

https://github.com/ and sign up for a free account.

2. Install Git:

Download and install Git for your operating system from https://git-scm.com/downloads.

3. Set Up Git:

- Open a terminal and configure your username and email for Git

commits:

- git config --global user.name "Your Name"
- git config --global user.email "your_email@example.com"

4. Create a Repository:

- Login to your GitHub account.
- Click "New repository" and give it a name and description (optional).
- Choose between Public (visible to everyone) or Private (only accessible to you).
- This creates a remote repository on GitHub to store your project.

5. Clone a Repository:

- Open a terminal and navigate to your desired local directory.
- Use the git clone command followed by the remote repository URL to download it:
- git clone https://github.com/username/repository.git
- This creates a local copy (clone) of the remote repository.

6. Create Branches:

- Branches allow you to work on different versions of your project without affecting the main code.
- Use the git branch command to list existing branches and git checkout to switch branches.
- To create a new branch, use git branch new_branch_name.

7. Make Changes and Commit:

- Make changes to your project files.
- Use git add to stage files for the next commit.
- Run git commit -m "Commit message" to capture the changes with a descriptive message.
- This creates a snapshot of your project at that point.

8. Push Changes to GitHub:

- After committing, use git push origin branch_name to upload your local commits to the remote repository on GitHub.

- origin is the default remote name, and branch_name is the branch you're pushing.

9. Pull Requests:

- When working on a shared project, use Pull Requests to propose changes from your branch to the main codebase.
- Create a Pull Request on GitHub, outlining the changes you made.
- Reviewers can discuss and suggest modifications before merging your changes.

10. Merge Changes:

- Once a Pull Request is approved, you can merge your branch's changes into the main branch.
- This integrates your work into the project's main codebase.
- Use the git merge command in the terminal to perform the merge.

11. Forking:

- Forking creates a copy of someone else's repository on your GitHub account.
- This allows you to make your own changes without affecting the original project.
- You can then create Pull Requests to propose your modifications back to the original repository.

12. Collaboration:

- GitHub facilitates collaboration by allowing multiple users to work on a project.
- You can assign tasks, discuss changes in Pull Requests, and track project progress.
- Utilize features like code reviews and issue tracking for effective collaboration.

Figma

Figma is a versatile design tool that operates entirely in the cloud, allowing for seamless collaboration among team members in real-time. Its interface consists of several key components:

- Canvas: The canvas is the main workspace where you design your interface. It's where you place and arrange elements like shapes, text, and images.
- Layers Panel: This panel displays all the layers in your design, allowing you to easily select, organize, and manipulate them. Layers are stacked hierarchically based on their order and grouping.
- Toolbar: The toolbar provides access to various tools and features for designing, editing, and prototyping your interface. It includes tools like selection, shapes, text, and more.
- Hierarchy: Hierarchy refers to the arrangement of elements in a design to establish a visual order and guide the viewer's attention. It involves organizing elements based on importance, emphasizing key components, and creating a sense of structure.
- Contrast: Contrast involves creating visual differences between elements to make them stand out and attract attention. This can be achieved through variations in color, size, shape, texture, or typography.
- Color Theory: Color theory explores the principles of color and how they impact visual communication. It includes concepts such as color harmony, contrast, saturation, and the psychological effects of different colors.
- Typography: Typography deals with the selection, arrangement, and styling of text in a design. It encompasses aspects like font choice, size, spacing, alignment, and hierarchy.
- Frames: Frames are containers that hold design elements. They can be resized, rotated, and nested within one another. Frames can also have constraints to control how their contents respond to changes in size.
- Artboards: Artboards represent individual screens or pages within a design project. They help organize content and maintain context between different parts of a project, such as different screens in a mobile app or pages in a website.

Tools in Figma

- Selection Tool: Used to select and manipulate objects on the canvas.
- Frame Tool: Creates frames, which act as containers for content and help organize designs.
- Shapes: Offers a variety of shape tools for drawing basic shapes like rectangles, circles, polygons, etc.
- Text: Allows you to add and format text within your designs.
- Pen Tool: Used for vector drawing and creating custom shapes or paths with precision.

Layers in Figma

- Layers in Figma represent individual elements within your design, such as shapes, text, images, etc.
- Layers are organized hierarchically in the layers panel, allowing you to easily select, arrange, and group them.
- Proper layer organization is essential for maintaining a tidy and manageable design project.

Vector Editing

Figma provides robust vector editing capabilities, allowing you to create and edit vector graphics with precision.

The pen tool and other vector tools enable you to draw custom shapes, paths, and illustrations directly within Figma.

Text and Color Styles

Text Styles: Text styles allow you to create and apply consistent text formatting across your designs. You can define attributes such as font, size, color, and spacing as reusable styles.

Color Styles: Similarly, color styles enable you to define and apply consistent color palettes throughout your designs. You can create color swatches and reuse them across your project.

Components in Figma

Components are reusable design elements that can be shared and reused across multiple instances within a project.

They help maintain design consistency by ensuring that changes made to one instance of a component are reflected across all instances.

Components can range from simple UI elements like buttons and icons to more complex elements like navigation bars and cards.

Prototyping in Figma

Figma's prototyping features allow you to create interactive prototypes by adding links, transitions, and animations to your designs.

You can define interactions between different frames/artboards, simulate user flows, and gather feedback from stakeholders or users.

Prototyping in Figma helps visualize and test the usability and functionality of your designs before implementation.