**R&D (HTML, CSS, Bootstrap)**

**HTML (Hypertext Markup Language)**

**Basic Structure:**

* **<html>**
* The <html> tag is the root element of an HTML document. It defines the

beginning and end of an HTML document.

* Everything you want to display on a webpage is enclosed within this tag.
* **<head>**
* The <head> tag contains meta information about the HTML document.
* This information isn't displayed directly on the webpage but is crucial for the

browser and search engines.

* It usually includes things like the title, links to stylesheets, and meta tags for

things like character sets and viewport settings.

* **<body>**
* The <body> tag contains all the content that is visible on the webpage, like

text, images, videos, and other elements.

**Elements:**

* **<div>**
* The <div> tag is used to group or divide sections of content or elements together. It’s a Block-level element
* It's often used to create containers for styling with CSS or to organize large chunks of HTML.
* **<span>**
* The <span> tag is used for grouping inline elements together.
* It's useful for applying styles to small pieces of content without affecting the entire block.
* **<p>: Paragraph**
* The <p> tag is used to define paragraphs of text.
* It helps in structuring content by breaking it into readable blocks of text.
* Block-level element, automatically adds space above and below the paragraph.
* **<a>: Anchor**

The <a> tag is used to create hyperlinks to other pages, sections, or files.

it enables navigation between web pages or sections within a page.

* **<h1> to <h6>**

HTML headings are defined with the <h1> to <h6> tags, where <h1>

represents the most important heading and <h6> the least.

**Form Elements**:

* **<form>**
* The <form> tag is used to create an HTML form for user input.
* it wraps around all input elements to collect data and send it to a server.
* <input>
* The <input> tag is used to create various input fields (text, email, password, etc.).
* Allows the user to input data.
* <button>
* The <button> tag is used to create a clickable button.
* It's typically used for submitting a form, but it can also trigger JavaScript actions.
* type (can be "submit", "reset", or "button").
* <select>
* The <select> tag creates a dropdown list for selecting one or more options.
* It provides a way to offer a list of predefined options.
* <option> tags go inside <select> to define each item.
* <textarea>
* The <textarea> tag is used for larger text inputs, such as comments or feedback forms.
* It allows users to enter multiple lines of text.

**Common Input Types**:

* **Text (type="text")** => A single-line text input.
* **Email (type="email")**  => An input field specifically for email addresses.
* **Password (type="password")** => An input field for passwords, where characters are hidden.
* **Checkbox (type="checkbox")** => A checkbox allows the user to select multiple options.
* **Submit (type="submit") =>** A button that submits the form when clicked.

**Semantic Elements:**

* <header>
* The <header> tag is used to define the header of a webpage or section.
* Usually contains introductory content, like a logo, navigation links, or a title.
* It’s typically placed at the top of a webpage or section but can be used in multiple sections on the same page.
* <footer>
* The <footer> tag defines the footer of a webpage or section.
* Typically contains information like copyright, links to privacy policies, or contact info.
* <article>
* The <article> tag represents an independent, self-contained piece of content.
* It can be used for blog posts, news articles, forum posts, or any content that can be independently distributed.
* It should make sense on its own without any surrounding context.
* <section>
* The <section> tag is used to group related content within a webpage.
* It breaks the page into logical sections (like chapters, parts of an article, or thematic groups).
* Sections are more general than articles and may not stand alone.
* <nav>
* The <nav> tag is used to define a block of navigation links.
* it helps users navigate through the site (e.g., main menu, side menu).
* <main>
* The <main> tag contains the central content unique to the webpage.
* It excludes content that is repeated across pages like headers, footers, and sidebars.
* There should only be **one** <main> tag per page.
* <aside>
* The <aside> tag is used for content that is related to the main content but not essential.
* It's often used for sidebars, ads, or extra info.
* Content in an <aside> is supplementary and not central to the main flow of the document.

**Table Elements:**

* **<table>**
* The <table> tag defines the entire table structure.
* It's the main container for all other table-related tags.
* **<tr>: Table Row**
* The <tr> tag defines a row in the table.
* It groups table cells (both header cells and data cells) into a row.
* It can contain both header cells (<th>) and data cells (<td>).
* **<th>: Table Header Cell**
* The <th> tag defines a header cell in the table.
* It is used to create headings for the columns or rows, and the text inside is usually bold and cantered by default.
* Typically used inside the first row or first column of a table.
* **<td>: Table Data Cell**
* The <td> tag defines a data cell in the table.
* It contains the actual data (values) in the table.
* Can be used to display text, images, links, or any other content.
* **colspan: Merging Columns**
* The colspan attribute allows a cell to span (merge) across multiple columns.
* Useful when a single cell needs to take up the space of two or more columns.
* It's used in a <td> or <th> tag to specify how many columns the cell should span.
* **rowspan: Merging Rows**
* The rowspan attribute allows a cell to span across multiple rows.
* It merges cells vertically to occupy space in two or more rows.
* Like colspan, it's added to a <td> or <th> tag, specifying how many rows it should span.

**Lists:**

* **<ol> : Ordered Lists**
* An ordered list is a list where the items are numbered, either in ascending or descending order.
* Used when the sequence of items matters, like a list of steps or rankings.
* Each list item is automatically numbered by the browser.
* **<ul> : Unordered Lists**
* An unordered list is a list where items are **not** numbered, but instead are marked with bullets.
* Used when the order of items doesn’t matter, like a list of ingredients or random notes.
* The browser adds bullets by default.

**Links and Navigation:**

* **Internal Links**:
  + Internal links point to **another page or section** within the same website or document.
  + To link users to different parts of your website or even different sections on the same page (useful for navigation menus, table of contents, etc.).
* **External Links**:
* External links point to another website or resource outside of the current website.
* To direct users to other websites, social media platforms, or external resources.

**Images and Media:**

* **<img>**
* The <img> tag is used to embed images in a webpage.
* To display pictures, logos, icons, etc., as part of the website content.
* **Key Attributes**:
* **src**: Specifies the path to the image file.
* **alt**: Provides alternative text for the image (shown if the image cannot load, also helps with accessibility).
* **width** and height: Set the dimensions of the image (optional).
* **<video>**
* The <video> tag is used to embed video content in a webpage.
* To display video clips or movies directly within the web browser.
* **Key Attributes**:
  + **src**: Specifies the path to the video file.
  + **controls**: Displays video controls (play, pause, volume, etc.).
  + **autoplay**: Automatically starts the video when the page loads.
  + **loop**: Replays the video in a loop.
  + **muted**: Mutes the video on load.
  + **poster**: Image to display before the video starts playing.
* **<audio>**
* The <audio> tag is used to embed audio content in a webpage.
* To add sound files like music, podcasts, or any kind of audio to the website.
* **Key Attributes**:
  + **src**: Specifies the path to the audio file.
  + **controls**: Adds audio controls (play, pause, volume).
  + **autoplay**: Automatically plays the audio when the page loads.
  + **loop**: Replays the audio in a loop.
  + **muted**: Mutes the audio on load.

**CSS (Cascading Style Sheets)**

**Key Topics:**

**Selectors:**

* Element Selector:
* Selects HTML elements by their tag name.
* Apply styles to all elements of a specific type.
* **Class Selector:**
* Selects elements based on the value of their class attribute.
* Apply styles to one or more elements that share the same class.
* Use a dot (.) followed by the class name.
* **ID Selector:**
* Selects an element based on its unique id attribute.
* Apply styles to a specific element.
* Use a hash (#) followed by the ID name.
* **Pseudo-Classes:**
* Pseudo-classes are special keywords added to selectors that specify a special state of an element.
* Apply styles when elements are in a specific state, like when they are hovered, focused, or clicked.
* **:hover**: => Applies styles when the user hovers over an element with a mouse.
* **:focus**: => Applies styles when an element (like an input field) is focused, usually via keyboard or mouse click.
* **:active**: => Applies styles when an element (usually a link or button) is actively being clicked.
* **:nth-child(n)**:Selects elements based on their order within a parent element.
* **Pseudo-Elements:**
* Pseudo-elements are used to style specific parts of an element, like the first letter or line, or to insert content before or after an element.
* Allows you to target and style specific parts of an element’s content or to insert generated content.
* **::before**: => Inserts content **before** the content of an element.
* **::after**: => Inserts content **after** the content of an element.
* **::first-letter**: => Styles the first letter of an element.
* **::first-line**: => Styles the first line of an element.

**Box Model:**

* **Content**
* The actual content of the element, such as text, images, or other elements.
* This is where your main content (text, image, etc.) resides.
* **Padding**
* The space between the content and the border. Padding expands the size of the box around the content but keeps it within the border.
* Adds space **inside** the element, between the content and the border.
* **Border**
* A line that surrounds the padding and content. Borders can be styled with different colors, widths, and styles (e.g., solid, dashed).
* Surrounds the element's padding and content to create a visual boundary.
* **Margin**
* The space **outside** the border of an element, used to create space between the element and other elements.
* Creates space around the element but does not affect the element’s size directly.
* box-sizing: border-box;
* box-sizing is a CSS property that controls how the total width and height of an element are calculated.
* With box-sizing: border-box;, the padding and border are included in the element’s total width and height, which simplifies layout calculations.

**Positioning:**

* **Static Positioning (default)**
* This is the default position of an element. Elements with position: static; are placed according to the normal document flow, meaning they appear where they normally would on the page without any special positioning.
* No extra control over the element’s position.
* Cannot be moved with top, right, bottom, or left properties.
* **Relative Positioning**
* An element with position: relative; is positioned relative to its **original position** in the normal document flow.
* Moves the element without affecting the layout of surrounding elements.
* Can be shifted using top, right, bottom, or left.
* The element remains in the flow of the document (meaning it still occupies space in the layout).
* **Absolute Positioning**
* An element with position: absolute; is positioned relative to the **nearest positioned ancestor** (an ancestor with relative, absolute, fixed, or sticky positioning). If there is no such ancestor, it will be positioned relative to the **initial containing block** (usually the <html> or <body> element).
* The element is **removed from the document flow**, meaning it does not affect the position of other elements.
* Positioned using top, right, bottom, and left.
* **Fixed Positioning**
* An element with position: fixed; is positioned relative to the **viewport** (the browser window), meaning it stays in the same position even when the page is scrolled.
* The element is removed from the document flow.
* The element remains in a fixed position even if the user scrolls the page.
* Can be positioned using top, right, bottom, and left.
* **Sticky Positioning**
* An element with position: sticky; is positioned relative to the document flow until it reaches a specified position (e.g., top, bottom), at which point it becomes **fixed**.
* A hybrid of relative and fixed positioning.
* Starts off behaving like relative, but when you scroll past a certain point, it "sticks" to a position on the screen (like fixed).
* Requires a top, right, bottom, or left value to define where it becomes "sticky."
* **Z-Index**
* z-index is used to control the **stack order** of elements when they overlap. Elements with a higher z-index value will appear in front of those with a lower value.
* Only works on elements with a **position** of relative, absolute, fixed, or sticky.
* The default z-index value is auto, which means elements will appear in the order they are written in the HTML (last one is on top).
* The z-index property accepts both positive and negative numbers.

**Flexbox:**

* **Main Axis**:
* The primary axis along which flex items are laid out in a flex container.
* Direction is defined by the flex-direction property (default is row).
* **Cross Axis**:
* The axis perpendicular to the main axis.
* For a row direction, the cross axis runs vertically; for a column direction, it runs horizontally.
* **flex-direction**
* Specifies the direction in which flex items are placed in the flex container.
* **Values**:
  + row (default): Items are laid out in a row, from left to right.
  + row-reverse: Items are laid out in a row, from right to left.
  + column: Items are laid out in a column, from top to bottom.
  + column-reverse: Items are laid out in a column, from bottom to top.
* **flex-wrap**
* Determines whether flex items should wrap onto multiple lines when they overflow the container.
* **Values**:
  + nowrap (default): All items are laid out in a single line, potentially overflowing the container.
  + wrap: Items wrap onto multiple lines, creating additional space when necessary.
  + wrap-reverse: Items wrap onto multiple lines in reverse order.
* **justify-content**
* Aligns flex items along the main axis (horizontally in a row direction).
* **Values**:
  + flex-start (default): Items are packed toward the start of the flex container.
  + flex-end: Items are packed toward the end of the flex container.
  + center: Items are centered along the main axis.
  + space-between: Items are evenly distributed with the first item at the start and the last at the end.
  + space-around: Items are evenly distributed with equal space around them.
* **align-items**
* Aligns flex items along the cross axis (vertically in a row direction).
* **Values**:
  + stretch (default): Items stretch to fill the container.
  + flex-start: Items are aligned at the start of the cross axis.
  + flex-end: Items are aligned at the end of the cross axis.
  + center: Items are centered along the cross axis.
  + baseline: Items are aligned along their baseline.

**Grid Layout:**

* **grid-template-rows**
* Specifies the height of each row in the grid.
* You can use fixed units (like px, em, rem), percentages (%), or flexible units like fr (fraction of available space).
* **grid-template-columns**
* Specifies the width of each column in the grid, using the same units as for rows.
* **grid-area**
* Defines a grid item’s position and the area it spans across rows and columns.
* **grid-area** can be set in shorthand, specifying grid-row-start, grid-column-start, grid-row-end, and grid-column-end.
* **Gap Property**
* Adds space between grid items (row-gap and column-gap).
* **gap**: A shorthand to specify both row-gap and column-gap in a single declaration.
* **Alignment Properties for Grid**

Grid layout offers various alignment properties to position grid items within the grid container.

* **justify-items**
* **Aligns items along the row (horizontal axis)** in their grid area.
* Values: start, end, center, stretch (default).
* **align-items**
* **Aligns items along the column (vertical axis)** in their grid area.
* Values: start, end, center, stretch (default).
* **justify-content**
* Aligns the grid **within the grid container horizontally** (main axis).
* Values: start, end, center, space-between, space-around, space-evenly.

**align-content**

* Aligns the grid **within the grid container vertically** (cross axis).
* Values: start, end, center, space-between, space-around, space-evenly.

**Typography:**

* **font-family**
* Specifies the font or a list of fonts to be used for the text.
* **font-size**
* **Definition**: Specifies the size of the text.
* **Units**:
  + **px**: Fixed pixel size (e.g., 16px).
  + **em**: Relative to the parent element's font size (e.g., 1.2em).
  + **rem**: Relative to the root element's font size (usually 16px by default).
  + **%**: Relative to the parent element's size (e.g., 120%).
* **color**
* Sets the color of the text.
* **Values**:
  + Named colors (e.g., red, blue).
  + HEX values (e.g., #ff0000).
  + RGB values (e.g., rgb(255, 0, 0)).
  + HSL values (e.g., hsl(0, 100%, 50%)).
* **line-height**
* Controls the vertical spacing between lines of text (leading).
* **Values**:
  + A unitless number (e.g., 1.5), which is multiplied by the font size.
  + Fixed values like px, em, or percentages.
* **text-align**
* Specifies the horizontal alignment of the text.
* **Values**:
  + left (default for most languages).
  + right.
  + center.
  + justify (distributes text evenly across the width of the container).
* **text-decoration**
* Adds decorative lines to the text (such as underlining, overlining, or striking through).
* **Values**:
  + none: No decoration.
  + underline: Adds an underline.
  + overline: Adds a line above the text.
  + line-through: Strikes through the text.

**Responsive Design**:

* **Media Queries (@media)**
* Media queries allow you to apply CSS styles conditionally, based on the device's characteristics like screen width, height, resolution, etc.
* **Syntax**: You define breakpoints using @media rules, applying styles only when certain conditions are met.
* **Breakpoints for Responsiveness**
* Breakpoints are specific screen widths where the layout of a website changes to better fit the device. Commonly used breakpoints target typical device sizes such as mobile phones, tablets, and desktops.
* **Common Breakpoints:**
* **Mobile**: max-width: 480px
* **Tablet**: max-width: 768px
* **Laptop/Desktop**: min-width: 1024px
* **Large screens**: min-width: 1200px
* **Relative Units**
* Relative units scale elements relative to some other value, making layouts more adaptable to screen sizes.
* **Common Relative Units:**

1. **rem** (Root Em)
   * Relative to the font-size of the root element (<html>), typically 16px by default.
   * **1rem** = 16px (if root font size is 16px).
2. **em**
   * Relative to the font-size of its parent element.
   * If a parent element has a font size of 20px, then **1em** = 20px.
3. **% (Percent)**
   * Relative to the size of the parent element.
   * For example, setting an element’s width to 50% means it will take up half the width of its parent.
4. **vw (Viewport Width)**
   * **1vw** = 1% of the viewport width.
   * Used to make elements responsive to the browser’s width.
5. **vh (Viewport Height)**
   * **1vh** = 1% of the viewport height.
   * Useful for setting heights relative to the viewport, making layouts responsive to different screen heights.