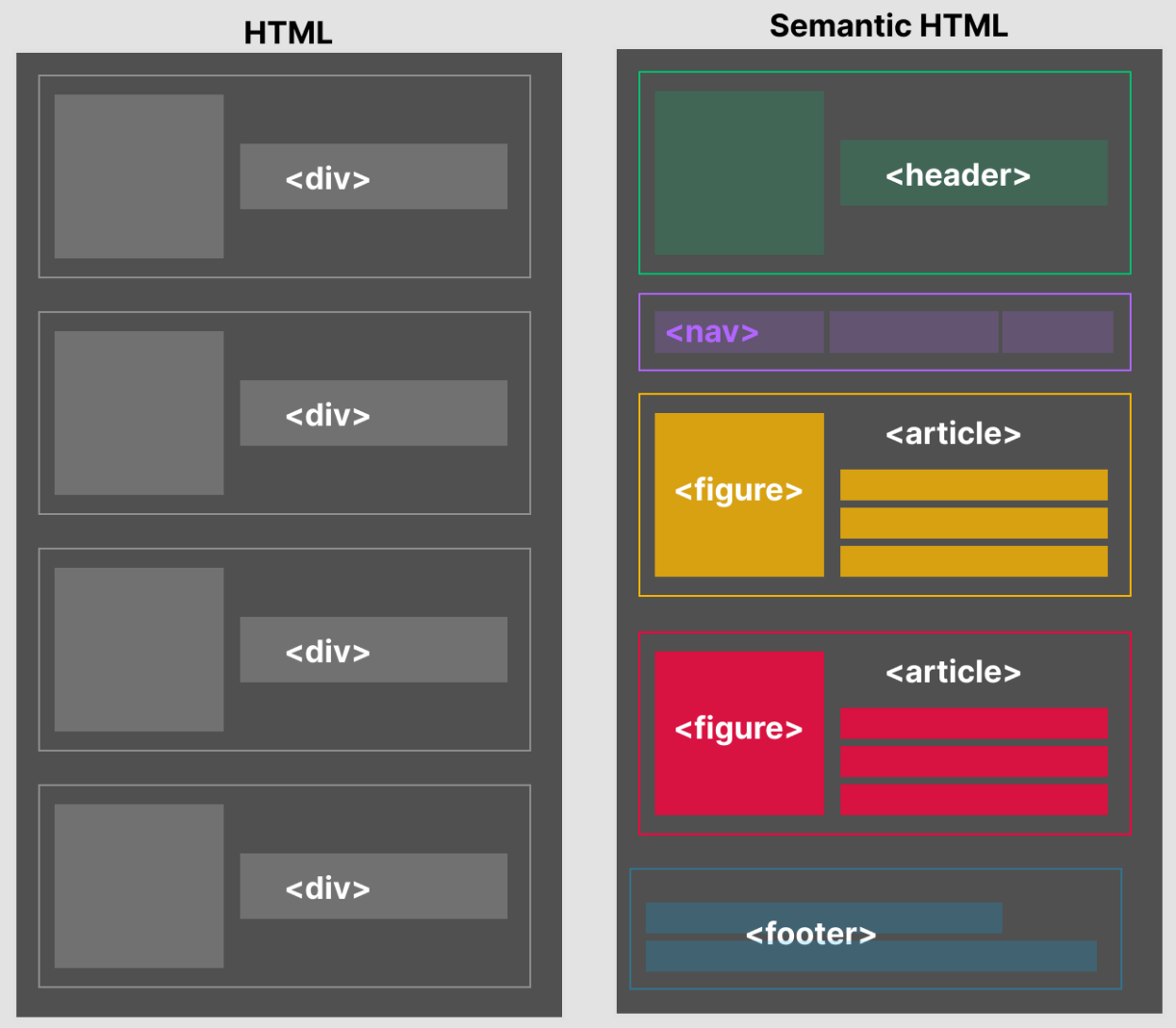
Week 1 - HTML Deep dive

Semantic HTML

HTML structure was originally created structure content on the web browser, such as headings, paragraphs, images, etc. But today machines are also reading web content for sharing or indexing purposes, if a human or machine reading the content the existing tags are enough, but to make the reading more efficient, the latest version of HTML introduced the semantics tags such as nav, header, footer, section, article, etc. This way it will be easy to figure out which content is what for the man or machine.

If you observe the below image the change can be observed. Each section of HTML is more readable and clear



Element Placement

Semantic HTML introduces elements that can tell developers exactly what the element does or where it's placed based on the name of that element. Some of these elements are `<header>`, `<nav>`, `<main>`, and `<footer>`. `<header>` describes the content at the top of the page `<body>`. It may include a logo, navigational links or a search bar. `<nav>` encapsulates the page's navigational links. It is often placed inside the `<header>` or `<footer>`. `<main>` encapsulates the main content of a page between the header/navigation and the footer

areas. `<footer>` includes the page's footer content at the bottom of the `<body>`.

Embedding media

Semantic HTML introduces us to `<video>`, `<audio>` and `<embed>`. `<video>` allows us to add videos to our website. `<audio>` allows us to implement audio into our website. `<embed>` can be used to implement any type of media. These elements are universal in that they all use the `src` attribute to link the source of the content. `<video>` and `<audio>` requires a closing tag while `<embed>` is a self-closing tag.

```
<!--Video Tag-->
<video src="somevideo.mp4">video not supported</video>

<!--Audio Tag-->
<audio src="someAduioFile.mp3"></audio>

<!--Embed tag-->
<embed src="someAnimation.gif"/>
```

`<figure>` and `<figcaption>`

The `<figure>` element is used to encapsulate media such as an image, diagram. or code snippet. The `<figcaption>` element is used to describe the media encapsulated within the `<figure>` element. Developers will normally use `<figcaption>` within the `<figure>` element to group the media and description. This way, if a developer decides to change the position of the media, the description will follow along with it.

```
<figure>
 <img src="dance.jpg">
 <figcaption>This is a image of baby dancing</figcaption>
</figure>
```

`<section>` and `<article>`

`<section>` defines elements in a document, such as chapters, headings, or any other area of the document with the same theme. `<article>` holds content that makes sense on its own such as articles, blogs, and comments. Generally developers will use `<section>` to define a theme for the webpage and use `<article>` to write independent content for that theme. This does not mean that `<article>` has to be used with `<section>`.

```
<section>
  <!--defines theme-->
  <h2>Top Sports league in America</h2>
<!--writes independent content relating to that theme-->
  <article>
    <p>One of the top sports league is the nba.</p>
  </article>
</section>
```

`<aside>` Aside Element

The `<aside>` element is used to mark additional information that can enhance another element but isn’t required in order to understand the main content. Usually, this information would be in a sidebar or a location where it doesn’t obstruct the main piece of content. An example of this would be an article that discusses how to take care of a dog and next to the article an ad would appear advertising a dog grooming product.

```
<article>
<!--Main Content-->
</article>
<aside>
<!--Additional information-->
</aside>
```

HTML Forms

`<input>`: Checkbox Type

When using an HTML `input` element, the `type="checkbox"` attribute will render a single checkbox item. To create a group of checkboxes related to the same topic, they should all use the same `name` attribute. Since it's a checkbox, multiple checkboxes can be selected for the same topic.

```
<input type="checkbox" name="breakfast" value="bacon">Bacon 🥓<br>
<input type="checkbox" name="breakfast" value="eggs">Eggs 🍳<br>
<input type="checkbox" name="breakfast" value="pancakes">Pancakes 🥞<br>
```

`<textarea>` Element

The `textarea` element is used when creating a text-box for multi-line input (e.g. a comment section). The element supports the `rows` and `cols` attributes which determine the height and width, respectively, of the element.

When rendered by the browser, `textarea` fields can be stretched/shrunk in size by the user, but the `rows` and `cols` attributes determine the initial size.

Unlike the `input` element, the `<textarea>` element has both opening and closing tags. The `value` of the element is the content in between these tags (much like a `<p>` element). The code block shows a `<textarea>` of size 10x30 and with a `name` of `"comment"`.

```
<textarea rows="10" cols="30" name="comment"></textarea>
```

`<form>` Element

The HTML `<form>` element is used to collect and send information to an external source.

`<form>` can contain various input elements. When a user submits the form, information in these input elements is passed to the source which is named in the `action` attribute of the form.

```
<form method="post" action="http://server1">
  Enter your name: <input type="text" name="fname">  <br/>
  Enter your age:  <input type="text" name="age">  <br/>
  <input type="submit" value="Submit">
</form>
```

`<input>`: Number Type

HTML input elements can be of type `number`. These input fields allow the user to enter only numbers and a few special characters inside the field.

The example code block shows an input with a type of `number` and a name of `balance`. When the input field is a part of a form, the form will receive a key-value pair with the format: `name: value` after form submission.

```
<input type="number" name="balance" />
```

`<input>` Element

The HTML `<input>` element is used to render a variety of input fields on a webpage including text fields, checkboxes, buttons, etc. `<input>` element have a `type` attribute that determines how it gets rendered to a page.

The example code block will create a text input field and a checkbox input field on a webpage.

```
<label for="fname">First name:</label>
<input type="text" name="fname" id="fname"><br>
<input type="checkbox" name="vehicle" value="Bike"> I own a bike
```

`<input>`: Range Type

A slider can be created by using the `type="range"` attribute on an HTML `input` element. The range slider will act as a selector between a minimum and a maximum value. These values are set using the `min` and `max` attributes respectively. The slider can be adjusted to move in different steps or increments using the `step` attribute.

The range slider is meant to act more as a visual widget to adjust between 2 values, where the relative position is important, but the precise value is not as important. An example of this can be adjusting the volume level of an application.

```
<input type="range" name="movie-rating" min="0" max="10" step="0.1">
```

`<select>` Element

The HTML `<select>` element can be used to create a dropdown list. A list of choices for the dropdown list can be created using one or more `<option>` elements. By default, only one `<option>` can be selected at a time.

The value of the selected `<select>`'s `name` and the `<option>`'s `value` attribute are sent as a key-value pair when the form is submitted.

```
<select name="rental-option">
  <option value="small">Small</option>
  <option value="family">Family Sedan</option>
```

```
  <option value="lux">Luxury</option>
</select>
```

Submitting a Form

Once we have collected information in a form we can send that information somewhere else by using the `action` and `method` attribute. The `action` attribute tells the form to send the information. A URL is assigned that determines the recipient of the information. The `method` attribute tells the form what to do with that information once it’s sent. An HTTP verb is assigned to the `method` attribute that determines the action to be performed.

```
<form action="/index3.html" method="PUT"></form>
```

`<input>`: Text Type

HTML `<input>` elements can support text input by setting the attribute `type="text"`. This renders a single row input field that users can type text inside.

The value of the `<input>`‘s `name` and `value` attribute of the element are sent as a key-value pair when the form is submitted.

```
<input type="text" name="username">
```

`<datalist>` Element

When using an HTML input, a basic search/autocomplete functionality can be achieved by pairing an `<input>` with

a `<datalist>`. To pair a `<input>` with a `<datalist>` the `<input>`'s `list` value must match the value of the `id` of the `<datalist>`. The `datalist` element is used to store a list of `<option>`s.

The list of data is shown as a dropdown on an `input` field when a user clicks on the input field. As the user starts typing, the list will be updated to show elements that best match what has been typed into the input field. The actual list items are specified as multiple `option` elements nested inside the `datalist`.

`datalist`s are ideal when providing users a list of pre-defined options, but to also allow them to write alternative inputs as well.

```
<input list="ide">

<datalist id="ide">
  <option value="Visual Studio Code" />
  <option value="Atom" />
  <option value="Sublime Text" />
</datalist>
```

`<input>`: Radio Button Type

HTML `<input>` elements can be given a `type="radio"` attribute that renders a single radio button. Multiple radio buttons of a related topic are given the same `name` attribute value. Only a single option can be chosen from a group of radio buttons.

The value of the selected/checked `<input>`'s `name` and `value` attribute of this element are sent as a key-value pair when the form is submitted.

```
<input name="delivery_option" type="radio" value="pickup" />
<input name="delivery_option" type="radio" value="delivery" />
```

Submittable Input

HTML `<input>` elements can have a type attribute set to submit, by adding `type="submit"`. With this attribute included, a submit button will be rendered and, by default, will submit the `<form>` and execute its action.

The text of a submit button is set to `Submit` by default but can also be changed by modifying the `value` attribute.

`<input>` `name` Attribute

In order for a form to send data, it needs to be able to put it into key-value pairs. This is achieved by setting the `name` attribute of the `input` element. The `name` will become the `key` and the `value` of the input will become the `value` the form submits corresponding to the key.

It's important to remember that the name is not the same as the ID in terms of form submission. The ID and the name of the input may be the same, but the value will only be submitted if the `name` attribute is specified.

In the code example, the first input will be submitted by the form, but the second one will not.

```
<input name="username" id="username" />
<input id="address" />
```

`<label>` Element

The HTML `<label>` element provides identification for a specific `<input>` based on matching values of the `<input>`'s `id` attribute and the `<label>`'s `for` attribute. By default, clicking on the `<label>` will focus the field of the related `<input>`.

The example code will create a text input field with the label text “Password: “ next to it. Clicking on “Password: “ on the page will focus the field for the related `<input>`.

```
<label for="password ">Password:</label>
<input type="text" id="password" name="password">
```

`<input>` Password Type

The HTML `<input>` element can have the attribute `type="password"` that renders a single row input field which allows the user to type censored text inside the field. It is used to type in sensitive information.

The value of this `<input>`’s `name` and `value` (actual value and not the censored version) attribute of this element are sent as a key-value pair when the form is submitted.

The code block shows an example of the fields for a basic login form - the username and password fields.

```
<input type="text" name="username" />
<input type="password" name="password" />
```

`required` Attribute

In HTML, input fields have an attribute called `required` which specifies that the field must include a value.

The example code block shows an input field that is required. The attribute can be written as `required="true"` or simply `required`.

```
<input type="password" name="password" required >
```

`max` Attribute

HTML `<input>`s of type `number` have an attribute called `max` that specifies the maximum value for the input field.

The code block shows an `input` number field that is set to have a maximum value of `20`. Any value larger than `20` will mark the input field as having an error.

```
<input type="number" max="20">
```

`maxlength` Attribute

In HTML, input fields with type `text` have an attribute called `maxlength` that specifies the maximum number of characters that can be entered into the field. The code block shows an input text field that accepts text that has a maximum length of 140 characters.

```
<input type="text" name="tweet" maxlength="140">
```

`pattern` Attribute

In a `text` input element, the `pattern` attribute uses a regular expression to match against (or validate) the value of the `<input>`, when the form is submitted.

```
<form action="/action_page.php">
  Country code:
  <input type="text" name="country_code"
         pattern="[A-Za-z]{3}"
         title="Three letter country code">
  <input type="submit">
</form>
```

`minlength` Attribute

In HTML, an input field of type `text` has an attribute that supports minimum length validation. To check that the input text has a minimum length, add the `minlength` attribute with the character count.

The example code block shows an example of a text field that has a minimum length of `6`.

```
<input type="text" name="username" minlength="6" />
```

HTML Form Validators

HTML forms allow you to specify different kinds of validation for your input fields to make sure that data is entered correctly before being submitted. HTML supports a number of different validators, including things like minimum value, minimum/maximum length, etc. The validators are specified as attributes on the `input` field.

`min` Attribute

In HTML, input fields with type `number` have an attribute called `min` that specifies the minimum value that can be entered

into the field. The code block provided shows an input number field that accepts a number with a minimum value of 1.

```
<input type="text" name="username" minlength="6" />
```

Exercise

#3 Create the form as discussed in the Week 1 Session (Personal Details Form)