HTML:

1- Explain the difference between the div and span tags in HTML

|  |  |
| --- | --- |
| <div> | <span> |
| * division or a section * a container for HTML elements - which is then styled with CSS or manipulated with JavaScript * easily styled by using the class or id attribute * block-level element | * inline container used to mark up a part of a text, or a part of a document * easily styled by CSS or manipulated with JavaScript using the class or id attribute * an inline element |

2-What are semantic HTML tags? Provide examples of semantic tags.

semantic element clearly describes its meaning to both the browser and the developer EX:

|  |  |
| --- | --- |
| Semantic | Non semantic |
| * <article> * <aside> * <details> * <figcaption> * <figure> * <footer> * <header> * <main> * <mark> * <nav> * <section> * <summary> * <time> | * <div> * <span> |

3- How do you use the data-\* attribute in HTML?

used to store custom data private to the page or application

ability to embed custom data attributes on all HTML elements

The stored (custom) data can then be used in the page's JavaScript to create a more engaging user experience (without any Ajax calls or server-side database queries).

EX

<head>

<script>

function showDetails(animal) {

var animalType = animal.getAttribute("data-animal-type");

alert("The " + animal.innerHTML + " is a " + animalType + ".");

}

</script>

</head>

<body>

<h1>Species</h1>

<p>Click on a species to see what type it is:</p>

<ul>

<li onclick="showDetails(this)" id="owl" data-animal-type="bird">Owl</li>

<li onclick="showDetails(this)" id="salmon" data-animal-type="fish">Salmon</li>

<li onclick="showDetails(this)" id="tarantula" data-animal-type="spider">Tarantula</li> </ul> </body>

4- Explain the difference between the and tags in HTML5.

|  |  |
| --- | --- |
| article | Section |
| 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.  EX:   * Forum posts * Blog posts * User comments * Product cards * Newspaper articles | defines a section in a document.  According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."  EX:   * Chapters * Introduction * News items * Contact information |

5-hyperlink in new tab

|  |  |
| --- | --- |
| \_blank | Opens the linked document in a new window or tab |
| \_self | Opens the linked document in the same frame as it was clicked (this is default) |
| \_parent | Opens the linked document in the parent frame |
| \_top | Opens the linked document in the full body of the window |
| framename | Opens the linked document in the named iframe |

<a href="https://www.w3schools.com/" target="\_blank">Visit W3Schools!</a>

CSS:

1-box model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

Explanation:

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

2- center element horizontally and vertically

<!DOCTYPE html>

<html>

<head>

<style>

.container {

height: 200px;

position: relative;

border: 3px solid green;

}

.center {

margin: 0;

position: absolute;

top: 50%;

left: 50%;

-ms-transform: translate(-50%, -50%);

transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<div class="container">

<div class="center">

<p>I am vertically and horizontally centered.</p>

</div>

</div>

</body>

</html>

3- inline and inline-block

inline and inline-block allows to set a width and height on the element.

|  |  |
| --- | --- |
| inline | inline-block |
| Display: inline  top and bottom margins/paddings are not respected  <style>  span.a {  display: inline;  width: 100px;  height: 100px;  padding: 5px;  border: 1px solid blue;  background-color: yellow;  }  </style>  <body>  <h2>display: inline</h2>  <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>  </body> | display: inline-block, the top and bottom margins/paddings are respected  <style>  span.b {  display: inline-block;  width: 100px;  height: 100px;  padding: 5px;  border: 1px solid blue;  background-color: yellow;  }  </style>  </head>  <body>  <h2>display: inline-block</h2>  <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>  </body> |

4- media

use the @media rule to include a block of CSS properties only if a certain condition is true.

If the browser window is 600px or smaller, the background color will be lightblue

@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}

5-padding and margin

|  |  |
| --- | --- |
| padding | Margin |
| Clears an area around the content.  The padding is transparent | Clears an area outside the border.  The margin is transparent |

Bootstrap

1-container

Containers are a fundamental building block of Bootstrap that contain, pad, and align your content within a given device or viewport.

Bootstrap comes with three different containers:

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

<div class= “container”>Hello</div>

<div class= “container-fluid”>Hello</div>

2-How do you use Bootstrap's grid system to create a responsive layout?

grid system uses a series of containers, rows, and columns to layout and align content. It’s built with flexbox and is fully responsive. Below is an example and an in-depth explanation for how the grid system comes together.

* **Our grid supports six responsive breakpoints.** Breakpoints are based on min-width media queries, meaning they affect that breakpoint and all those above it (e.g., .col-sm-4 applies to sm, md, lg, xl, and xxl). This means you can control container and column sizing and behavior by each breakpoint.
* **Containers center and horizontally pad your content.** Use .container for a responsive pixel width, .container-fluid for width: 100% across all viewports and devices, or a responsive container (e.g., .container-md) for a combination of fluid and pixel widths.
* **Rows are wrappers for columns.** Each column has horizontal padding (called a gutter) for controlling the space between them. This padding is then counteracted on the rows with negative margins to ensure the content in your columns is visually aligned down the left side. Rows also support modifier classes to uniformly apply column sizing and gutter classes to change the spacing of your content.
* **Columns are incredibly flexible.** There are 12 template columns available per row, allowing you to create different combinations of elements that span any number of columns. Column classes indicate the number of template columns to span (e.g., col-4 spans four). widths are set in percentages so you always have the same relative sizing.
* **Gutters are also responsive and customizable.** Gutter classes are available across all breakpoints, with all the same sizes as our margin and padding spacing. Change horizontal gutters with .gx-\* classes, vertical gutters with .gy-\*, or all gutters with .g-\* classes. .g-0 is also available to remove gutters.
* **Sass variables, maps, and mixins power the grid.** If you don’t want to use the predefined grid classes in Bootstrap, you can use our grid’s source Sass to create your own with more semantic markup. We also include some CSS custom properties to consume these Sass variables for even greater flexibility for you.

3-Explain the difference between the .col-xs, .col-sm, .col-md, and .col-lg classes in Bootstrap.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Col-xs  Extra small <768px | Col-sm  Small >=768px | Col-md  Medium >=992px | Col-lg  Large >=1200px |
| **Suitable for** | Phones | Tablets | Small laptops | Laptops & Desktops |
| **Grid behaviour** | Horizontal at all times | Collapsed to start, horizontal above breakpoints |  | Collapsed to start, horizontal above breakpoints |
| **Container width** | None (auto) |  |  | 1170px |
| **# of columns** | 12 | 12 | 12 | 12 |
| **Column width** | Auto | ~62px | ~81px | ~97px |
| **Gutter width** | 30px (15px on each side of a column) | 30px (15px on each side of a column) | 30px (15px on each side of a column) | 30px (15px on each side of a column) |
| **Nestable** | Yes | Yes | Yes | Yes |
| **Offsets** | Yes | Yes | Yes | Yes |
| **Column ordering** | Yes | Yes | Yes | Yes |

4-How do you create a responsive navigation menu using Bootstrap?

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="#">WebSiteName</a>

</div>

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#">Page 1</a></li>

<li><a href="#">Page 2</a></li>

<li><a href="#">Page 3</a></li>

</ul>

</div>

</nav>

5-