Things to Know for Front end Technology’s

1. Importance of HTML DOCTYPE?

The HTML DOCTYPE declaration is a crucial element of an HTML document because it informs the web browser about the version of HTML being used in the document.

Here are a few reasons why the DOCTYPE declaration is important:

Specifies the HTML version: The DOCTYPE declaration specifies the version of HTML that the web page is written in, which helps the browser to render the document correctly.

Ensures browser compatibility: Different web browsers have different ways of rendering HTML documents, and the DOCTYPE declaration helps to ensure that the web page is displayed consistently across various browsers.

Validates HTML code: The DOCTYPE declaration also helps to validate the HTML code. HTML validators use the DOCTYPE to check whether the document complies with the standard.

Improves accessibility: The DOCTYPE declaration helps to improve the accessibility of the web page by ensuring that the document is correctly structured and coded. This is particularly important for users who rely on assistive technologies to access the web.

In summary, the DOCTYPE declaration is an essential part of an HTML document that ensures that the web page is rendered correctly, is compatible with different browsers, is valid and accessible

2. What is the syntax to link external style sheet?

To link an external style sheet to an HTML document, you need to use the HTML link tag in the document's head section.

Here's the basic syntax:

```
<head>
  <link rel="stylesheet" type="text/css" href="path/to/stylesheet.css">
</head>
```

Let's break this down:

The link element is used to link external resources to an HTML document.

The rel attribute specifies the relationship between the HTML document and the linked resource. In this case, the value of rel should be set to "stylesheet" to indicate that the linked resource is a style sheet.

The type attribute specifies the MIME type of the linked resource. In the case of a style sheet, the value of type should be set to "text/css".

The href attribute specifies the path to the external style sheet. The path can be an absolute or relative URL.

Make sure to replace "path/to/stylesheet.css" with the actual path to your external style sheet. Once you have added this link tag to your head section, the HTML document will apply the styles defined in the external style sheet to the document's content.

3. Difference between HTML, HTML5, CSS and JAVA Script

HTML, HTML5, CSS, and JavaScript are all programming languages used for web development, but they serve different purposes.

HTML (Hypertext Markup Language) is used for creating the structure and content of a web page. It defines the elements and tags used to create headings, paragraphs, images, links, forms, tables, and more. HTML is a markup language, not a programming language, and it is used to describe the structure of a web page rather than to add functionality to it.

HTML5 is the latest version of HTML, and it includes new features that make it easier to create more dynamic and interactive web pages. Some of the new features in HTML5 include video and audio playback, the ability to create animations and graphics with canvas and SVG, and new input types for forms.

CSS (Cascading Style Sheets) is used for controlling the presentation and layout of a web page. It allows you to define the fonts, colors, margins, borders, and other visual properties of HTML elements. CSS is used to separate the presentation of a web page from its content, which makes it easier to maintain and update the design of a site.

JavaScript is a scripting language used for adding interactivity and functionality to a web page. It allows you to create dynamic effects like pop-ups, drop-down menus, image sliders, and more. JavaScript can also be used for form validation, AJAX requests, and other advanced features. JavaScript is often used in combination with HTML and CSS to create interactive and responsive web pages.

In summary, HTML is used for creating the structure and content of a web page, CSS is used for controlling the presentation and layout of a web page, JavaScript is used for adding interactivity and functionality to a web page, and HTML5 is the latest version of HTML with new features for creating more dynamic and interactive web pages.

4. Difference between span and div

'span' and 'div' are both HTML elements used for grouping and styling content, but they are used in different contexts and have different default display properties.

div (short for division) is a block-level element that is used to group larger sections of content together. It can contain other block-level elements, such as headings, paragraphs, lists, and other divs. Divs are commonly used to create layout structures and to separate different sections of a web page. By default, divs start on a new line and take up the full width of their parent container.

span is an inline-level element that is used to group smaller pieces of content together, such as text or inline images. Spans are commonly used to apply styling to specific parts of text, such as changing the color, font, or background. By default, spans do not start on a new line and only take up as much width as their content requires.

Here is an example of using div and span elements in HTML:

```
<div class="section">
  <h1>Heading</h1>
  <p>This is a paragraph of text.</p>
  <span class="highlight">This is a highlighted piece of text.</span>
</div>
```

In this example, the div element is used to group together the heading, paragraph, and highlighted text as a section. The span element is used to apply the "highlight" class to the text and style it differently from the rest of the paragraph.

In summary, div is a block-level element used for grouping larger sections of content, while span is an inline-level element used for grouping smaller pieces of content and applying styling to specific parts of text.

5. How JavaScript differs from Java?

JavaScript and Java are two different programming languages that share some similarities but are also fundamentally different.

Here are some of the main differences between JavaScript and Java:

Purpose: JavaScript is primarily used for adding interactivity and dynamic effects to web pages, while Java is used for developing desktop applications, web applications, mobile apps, and more.

Syntax: JavaScript has a syntax similar to C languages, such as C++, while Java has a syntax similar to C++ and C#. JavaScript is a dynamic language, which means that variables are not statically typed, whereas Java is a statically typed language.

Execution: JavaScript code is executed by a web browser's JavaScript engine, while Java code is compiled into bytecode that is executed by the Java Virtual Machine (JVM).

Object-oriented programming: Both JavaScript and Java support object-oriented programming, but they differ in their approach. JavaScript is a prototype-based language, which means that objects inherit from other objects, while Java is a class-based language, which means that objects are created from classes.

Concurrency: JavaScript is single-threaded, which means that it can only perform one task at a time, while Java supports multithreading, which means that it can perform multiple tasks at the same time.

In summary, while JavaScript and Java share some similarities, they are different programming languages used for different purposes. JavaScript is primarily used for adding interactivity to web pages, while Java is used for developing a wide range of applications. They also differ in their syntax, execution, object-oriented programming approach, and concurrency.

6. Difference between visibility: hidden and display: none?

‘visibility: hidden’ and display:none are both CSS properties used for hiding elements on a web page, but they work differently.

visibility: hidden hides an element but preserves the space it occupies on the web page. This means that the element is not visible, but it still takes up space and affects the layout of the page. The hidden element is still present in the HTML document and can still be targeted by JavaScript or other CSS styles. Elements with visibility: hidden are not rendered on the page but are still part of the document flow.

display: none also hides an element, but it removes the element from the document flow and does not preserve the space it occupies on the web page. This means that the element is not visible and does not affect the layout of the page. The element is not present in the HTML document and cannot be targeted by JavaScript or other CSS styles. Elements with

display: none are completely removed from the document flow and are not rendered on the page.

Here is an example of using visibility: hidden and display: none:

=>CSS code:

```
.hidden {
  visibility: hidden;
}
.hidden2 {
  display: none;
}
```

=>HTML code:

```
<p>This is some text.</p>
<p class="hidden">This is some hidden text.</p>
<p>This is some more text.</p>
<p class="hidden2">This is some more hidden text.</p>
<p>This is even more text.</p>
```

In this example, the first hidden paragraph is hidden with visibility: hidden, which means that it is not visible but still takes up space on the page. The second hidden paragraph is hidden with display: none, which means that it is not visible and does not take up space on the page.

In summary, visibility: hidden hides an element but preserves its space on the web page, while display: none hides an element and removes it from the document flow, so it does not take up space on the web page.

7. Difference between inline and block level elements?

HTML elements are classified as either inline or block-level elements based on how they are displayed on the web page.

Block-level elements occupy the full width of their parent container and start on a new line. They can contain other block-level and inline elements. Block-level elements are often used

for larger content sections, such as paragraphs, headings, lists, forms, and images. By default, block-level elements have a line break before and after the element.

Examples of block-level elements include:

<h1> to <h6>

<p>

, <ol>,

<form>

<table>

Inline elements, on the other hand, only occupy the width of their content and do not start on a new line. They can be nested inside other inline and block-level elements. Inline elements are often used for smaller content sections, such as text and images within paragraphs. By default, inline elements do not have a line break before or after the element.

Examples of inline elements include:

<a>

, , <u>

Here is an example of using block-level and inline elements in HTML:

```
<div>
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
  <ul>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
  </ul>
```

```
  <img src="image.jpg" alt="An image">
  <a href="#">This is a link</a>
</div>
```

In this example, the block-level elements (, <h1>, <p>, <ul>) occupy the full width of the parent container and start on a new line, while the inline elements (<img>, <a>) only occupy the width of their content and do not start on a new line.

In summary, block-level elements are larger sections of content that occupy the full width of their parent container and start on a new line, while inline elements are smaller content sections that only occupy the width of their content and do not start on a new line.

8. What are selectors?

In CSS, selectors are patterns used to select and target specific HTML elements on a web page that you want to apply styles to. CSS selectors can be used to apply styles to specific elements or groups of elements based on their tag name, class, ID, attributes, or even their position in the HTML document.

There are several types of selectors in CSS:

Type selectors: Select elements based on their tag name, such as p for paragraphs or h1 for headings.

Class selectors: Select elements based on their class attribute, such as .my-class for elements with a class of my-class.

ID selectors: Select elements based on their ID attribute, such as #my-id for elements with an ID of my-id.

Attribute selectors: Select elements based on their attribute value, such as [type="text"] for elements with a type attribute of text.

Pseudo-class selectors: Select elements based on their state, such as :hover for elements when the user hovers over them or :nth-child() for elements based on their position in the parent container.

Pseudo-element selectors: Select and style a specific part of an element, such as ::before and ::after for adding content before or after an element.

Here is an example of using CSS selectors to target elements on a web page:

```
/* Select all paragraphs and set their font size to 16 pixels */
p {
  font-size: 16px;
}

/* Select all elements with class 'my-class' and set their background color to blue */
.my-class {
  background-color: blue;
}

/* Select the element with ID 'my-id' and set its text color to red */
#my-id {
  color: red;
}

/* Select all elements with a 'type' attribute of 'text' and set their border to 1 pixel solid black */
[type="text"] {
  border: 1px solid black;
}

/* Select the first child of a <ul> element and set its text color to green */
ul li:first-child {
  color: green;
}
```

```
/* Select and add content before all elements with class 'my-class' */

.my-class::before {

  content: "Before";

}
```

In this example, CSS selectors are used to apply styles to specific HTML elements based on their tag name, class, ID, attribute, and position in the parent container. This allows developers to selectively style elements on a web page, making it more visually appealing and user-friendly.

9. Which property is used to set the line style for the outline?

The outline-style property is used to set the line style for the outline in CSS. The outline property is a shorthand property that sets the outline-style, outline-width, and outline-color properties in one declaration.

Here is an example of using the outline-style property to set the line style for an outline:

```
/* Set the outline style for an element */

.element {

  outline-style: dotted;

}
```

In this example, the outline-style property is set to dotted, which creates a dotted line for the outline. Other values that can be used for the outline-style property include solid, double, groove, ridge, inset, and outset, among others.

Note that the outline property can also be used to set the line style for the outline. Here is an example:

```
/* Set the outline style for an element using the 'outline' shorthand property */

.element {

  outline: 2px dotted red;

}
```

In this example, the outline property is set to a red, dotted outline with a width of 2 pixels. The first value of the outline property sets the width, the second value sets the line style, and the third value sets the color of the outline.

10. Try applying as many attributes and restrictions and observe the form behaviour

Try out yourself in a browser to see how it behaves.

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required minlength="3" maxlength="20" pattern="[A-Za-z]+">
  <br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <br>
  <label for="password">Password:</label>
  <input type="password" id="password" name="password" required minlength="6">
  <br>
  <label for="age">Age:</label>
  <input type="number" id="age" name="age" required min="18" max="100">
  <br>
  <label for="gender">Gender:</label>
  <select id="gender" name="gender" required>
    <option value="">Select Gender</option>
    <option value="male">Male</option>
    <option value="female">Female</option>
    <option value="other">Other</option>
  </select>
  <br>
  <label for="terms">
    <input type="checkbox" id="terms" name="terms" required>
```

```
    I agree to the terms and conditions
  </label>
  <br>
  <input type="submit" value="Submit">
</form>
```

In this example, the form has several attributes and restrictions applied:

The required attribute is added to several input fields, which means that the user must fill in the field before submitting the form.

The minlength and maxlength attributes are used to specify the minimum and maximum number of characters allowed in the name input field.

The pattern attribute is used to specify a regular expression pattern that the name input field must match. In this case, it must contain only alphabetical characters.

The type attribute is set to "email" for the email input field, which ensures that the user enters a valid email address.

The type attribute is set to "password" for the password input field, which hides the characters that the user types.

The min and max attributes are used to specify the minimum and maximum age that can be entered in the age input field.

The select element is used to create a dropdown menu for the gender input field, and the required attribute is added to ensure that the user selects an option.

The checkbox input field is used to create a checkbox for the terms and conditions, and the required attribute is added to ensure that the user agrees to the terms before submitting the form.

By applying these attributes and restrictions, the form becomes more user-friendly and reduces the likelihood of errors or invalid submissions. You can try filling out this form and observing how it behaves when you try to submit it with invalid or missing input.

11. Understand the difference between target:blank and target:self

target:blank and target:self are two values of the target attribute in HTML that determine where a link should be opened when clicked.

target:blank: When a link has target="_blank", it will open the linked document in a new browser window or tab, depending on the user's browser settings. The new window/tab will typically not have any of the user's browsing history, cookies or session information. This is often used for external links, or for links to pages within the same website that should be opened separately from the current page.

target:self: When a link has target="_self", it will open the linked document in the same frame or window that the link is in. This is the default behavior if the target attribute is not specified. This is often used for links to pages within the same website, or for links that should replace the current page with the linked page.

It's worth noting that the use of target="_blank" has some potential drawbacks, including:

It can be annoying for users who prefer to control when new tabs or windows are opened.

It can be used for malicious purposes, such as opening a new tab with an unwanted advertisement or pop-up.

Therefore, it's generally recommended to use target="_blank" sparingly and only for links that genuinely need to be opened in a new window or tab. In general, it's best to use target="_self" or simply omit the target attribute altogether if the linked document should be opened in the same frame or window as the current page.

12. Know keywords and existing functions of JavaScript?

JavaScript is a programming language that is primarily used for web development, and it has a wide range of keywords and built-in functions that can be used to manipulate web page content, interact with users, and more. Here are some examples of keywords and functions in JavaScript:

Keywords:

var: Used to declare a variable in JavaScript.

if: Used for conditional statements that execute different code based on whether a condition is true or false.

for: Used for loop statements that repeat a block of code a specified number of times.

function: Used to define a function in JavaScript, which is a reusable block of code that can be called from other parts of the script.

Functions:

alert(): Displays an alert box with a message and an OK button.

console.log(): Outputs a message to the console, which can be useful for debugging.

document.getElementById(): Retrieves an element from the HTML document by its ID attribute.

document.createElement(): Creates a new HTML element that can be added to the document.

document.write(): Writes text or HTML code directly to the document.

Math.random(): Generates a random number between 0 and 1.

Date.now(): Returns the current date and time in milliseconds since January 1, 1970.

These are just a few examples of the many keywords and functions available in JavaScript. As a programming language, JavaScript can be used to create complex web applications, manipulate the content and style of web pages, and interact with web APIs and services.

13. Understand position properties?

In CSS, there are several position properties that allow you to control the positioning of an element within its containing element or within the viewport. Here's an overview of the different position properties:

static: This is the default position value for all elements. Elements with position: static are positioned according to the normal flow of the document. The top, bottom, left, right, and z-index properties have no effect on static elements.

relative: Elements with position: relative are positioned relative to their normal position in the document flow. You can use the top, bottom, left, and right properties to move the element up, down, left, or right from its original position. Setting a z-index value on a relative element will only affect its stacking order relative to other elements with position: relative.

absolute: Elements with position: absolute are positioned relative to the nearest positioned ancestor element, if any; otherwise, they are positioned relative to the initial containing block (usually the <html> element). You can use the top, bottom, left, and right properties to move the element relative to its containing element. Elements with position: absolute are taken out of the normal flow of the document, so other elements will flow around them.

fixed: Elements with position: fixed are positioned relative to the viewport and remain in the same position even when the page is scrolled. You can use the top, bottom, left, and right properties to position the element on the viewport.

sticky: Elements with position: sticky are positioned like relative elements until they reach a specified threshold, at which point they become fixed elements. You can use the top, bottom, left, and right properties to specify the threshold where the element becomes fixed.

Each of these position values has its own set of properties and behaviors, and choosing the right position value can be important for achieving the desired layout and behavior in your web page.

14. What is a box model?

In CSS, every element on a web page is considered a rectangular box, and the box model is a way to describe how those boxes are laid out on the page.

The box model consists of four parts:

Content: This is the actual content of the box, such as text, images, or other HTML elements.

Padding: This is the space between the content and the border of the box.

Border: This is a border that surrounds the padding and content of the box.

Margin: This is the space between the border of the box and other elements on the page.

Together, these four parts define the total size of the box, and they can all be styled individually using CSS properties. For example, you can set the width and height of the content area using the width and height properties, set the amount of padding using the padding property, set the style and color of the border using the border property, and set the amount of margin using the margin property.

Understanding the box model is important for creating web page layouts and ensuring that elements are spaced and sized correctly on the page. It can also help you debug layout issues and understand how different CSS properties interact with each other.

15. Difference between HTML4 and HTML5 for the following tags Header, menu, content, article and footer

HTML5 introduced several new semantic tags that help describe the structure of a web page more accurately than HTML4. Here's how some of these tags differ in their usage between HTML4 and HTML5:

Header: In HTML4, the header tag was not defined, so authors often used a div or table element to create a header section. In HTML5, the header tag is a new semantic tag that is used to define a header section of a web page, typically containing a logo, a navigation menu, or other introductory content.

Menu: In HTML4, there was no specific tag to define a menu. Authors often used a table or list element to create a navigation menu. In HTML5, the nav tag is a new semantic tag that is used to define a section of a web page that contains a navigation menu.

Content: In HTML4, authors often used a div element to define the main content section of a web page. In HTML5, the main tag is a new semantic tag that is used to define the primary content section of a web page. Additionally, HTML5 introduced several new semantic tags for defining different types of content, such as article, section, and aside.

Article: In HTML4, authors often used a div element to define an article section of a web page. In HTML5, the article tag is a new semantic tag that is used to define an independent, self-contained content section, such as a blog post or a news article.

Footer: In HTML4, authors often used a div element to create a footer section of a web page. In HTML5, the footer tag is a new semantic tag that is used to define a footer section of a web page, typically containing copyright information, contact details, or other secondary content.

Overall, the introduction of semantic tags in HTML5 helps web developers create more accessible, structured, and meaningful web pages. By using these tags appropriately, it is easier for search engines and other tools to understand the content and structure of a web page, which can lead to better indexing and search rankings.

16. Debugging in JavaScript.

Debugging is the process of finding and fixing errors, or bugs, in your code. In JavaScript, there are several techniques you can use to debug your code:

Using console.log(): One of the simplest ways to debug JavaScript code is to use console.log() to print out values of variables, objects, or other data types at different points in your code. This can help you see the flow of your code and identify where things may be going wrong.

Using breakpoints: Another useful technique is to set breakpoints in your code using the browser's developer tools. This allows you to pause the execution of your code at a specific line and inspect the state of your variables and objects.

Using the debugger statement: You can also use the debugger statement in your code to create a breakpoint at a specific line. When the code reaches that line, the debugger will pause the execution and allow you to inspect the state of your code.

Using try-catch blocks: If your code is throwing errors, you can use a try-catch block to catch those errors and handle them gracefully. This can help you isolate the errors and prevent them from crashing your application.

Using a linter: A linter is a tool that analyzes your code for potential errors and coding conventions. By using a linter, you can catch errors early on in the development process and ensure that your code follows best practices.

By using these debugging techniques, you can identify and fix errors in your JavaScript code more efficiently, which can save you time and improve the quality of your code.

17. What are Cookies?

Cookies are small text files that are stored on a user's computer by a web server. They are commonly used to store user preferences and login information, and to track user behavior on a website. When a user visits a website, the web server sends a cookie to the user's browser, which stores it on the user's computer. The next time the user visits the website, the browser sends the cookie back to the server, which can use the information in the cookie to personalize the user's experience.

Cookies can be used to store a wide range of information, such as user preferences, login credentials, shopping cart contents, and browsing history. They can also be used to track user behavior, such as the pages they visit and the links they click on. This information can be used by websites to personalize content, provide targeted advertising, and improve the user experience.

There are two main types of cookies: session cookies and persistent cookies. Session cookies are temporary cookies that are deleted when the user closes their browser. Persistent cookies, on the other hand, are stored on the user's computer for a longer period of time, and can be used to remember user preferences and login information across multiple sessions.

Cookies are widely used on the internet, but they have also been the subject of privacy concerns. Critics argue that cookies can be used to track users without their consent, and that they can be used to collect sensitive information, such as personal data and browsing history. As a result, many web browsers allow users to block or delete cookies, and some websites have implemented privacy policies and cookie consent mechanisms to address these concerns.

18. Understand table structure?

In HTML, tables are used to display data in a structured format, with rows and columns. The basic structure of an HTML table consists of the following elements:

Table: The table element is used to define the table.

Table Row: The tr element is used to define a row in the table.

Table Header: The th element is used to define a header cell in a row. The header cells are usually displayed in bold.

Table Data: The td element is used to define a data cell in a row.

Table Caption: The caption element is used to provide a title or description for the table.

Table Body: The tbody element is used to group the body content in the table.

Table Head: The thead element is used to group the header content in the table.

Table Foot: The tfoot element is used to group the footer content in the table.

Here's an example of a basic HTML table structure:

css

Copy code

```
<table>
 <caption>Example Table</caption>
 <thead>
  <tr>
   <th>Header 1</th>
   <th>Header 2</th>
   <th>Header 3</th>
  </tr>
 </thead>
 <tbody>
  <tr>
   <td>Row 1, Column 1</td>
```

```
    <td>Row 1, Column 2</td>
    <td>Row 1, Column 3</td>
   </tr>
   <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
    <td>Row 2, Column 3</td>
   </tr>
  </tbody>
  <tfoot>
   <tr>
    <td>Footer 1</td>
    <td>Footer 2</td>
    <td>Footer 3</td>
   </tr>
  </tfoot>
</table>
```

In this example, the table element defines the table. The caption element provides a title for the table. The thead element groups the header content in the table, with each header cell defined using the th element. The tbody element groups the body content in the table, with each data cell defined using the td element. The tfoot element groups the footer content in the table, with each footer cell defined using the td element.

19. Explain cell padding and cell spacing.

In HTML tables, cellpadding and cellspacing are attributes that can be used to add space between the cells in a table.

Cellpadding- is used to add space between the content of a cell and the edge of the cell. This attribute is defined using a numeric value, which specifies the amount of padding in pixels.

Cellspacing- is used to add space between adjacent cells in a table. This attribute is also defined using a numeric value, which specifies the amount of space in pixels.

Here's an example of how to use cellpadding and cellspacing in an HTML table:

```
<table cellpadding="10" cellspacing="5">
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</table>
```

In this example, the cellpadding attribute is set to 10, which adds 10 pixels of space between the content of each cell and the edge of each cell. The cellspacing attribute is set to 5, which adds 5 pixels of space between adjacent cells in the table.

It's worth noting that cellpadding and cellspacing are deprecated in HTML5 and should be replaced with CSS styling. Specifically, the padding property can be used to add padding to cells, and the border-spacing property can be used to add space between cells in a table.

20. What are form attributes?

HTML forms have a number of attributes that can be used to specify various properties of the form. Here are some commonly used form attributes:

action: Specifies the URL where the form data should be sent when the form is submitted.

method: Specifies the HTTP method to use when submitting the form, typically GET or POST.

name: Specifies a name for the form, which can be used to reference the form in JavaScript or server-side code.

enctype: Specifies the encoding type to use when submitting the form data. The two most common values are application/x-www-form-urlencoded and multipart/form-data.

target: Specifies where to display the response that's returned after the form is submitted. This can be _self to display the response in the same window, _blank to display it in a new window, or a named frame or window.

autocomplete: Specifies whether the browser should enable autocomplete for form fields. The two most common values are on and off.

novalidate: Specifies that the form should not be validated when it's submitted. This can be useful during development or when submitting data to a server that doesn't require validation.

There are also a number of other attributes that can be used to specify additional properties of individual form elements, such as type, value, placeholder, required, and disabled. These attributes can be used to specify the type of form element (e.g. text input, checkbox, select box), its default value, a placeholder text that appears in the input field before the user types anything, whether the field is required, and whether it's disabled (i.e. cannot be edited by the user).