The "Submit" button and the "input type="submit"" are two different ways to create a button that submits a form in HTML. They serve the same fundamental purpose, which is to trigger the submission of a form, but they have slightly different use cases and appearances.

1. `<button>` Element with `type="submit"`:

- Use the `<button>` element when you need a highly customizable form submission button.
- You can add text, images, or other HTML elements within the `<button>` tags to create a custom-looking button.
- You can style it extensively using CSS to match your design requirements.
- It's a more flexible option when you want to create a unique button style.

```
<button type="submit">Submit Form</button>
```

2. `<input>` Element with `type="submit"`:

- Use the `<input>` element with `type="submit"` when you want a simple, standard form submission button.
- It's a self-contained element specifically designed for form submission.
- It has a default appearance that varies slightly across different web browsers.
- It's a straightforward option for creating a basic form submission button without additional customization.

```
<input type="submit" value="Submit Form">
```

In both cases, when the user clicks the button, it triggers the form to submit its data to the server for processing. The choice between them depends on your design and styling preferences. If you need a more customized button, use `<button>`. If you want a simple, standard button, use `<input type="submit">`.

1. **Customization:**

- *`<button>`: It allows for extensive customization. You can include text, images, icons, or even other HTML elements within the `<button>` tags. This flexibility enables you to create unique button styles.*
- *`<input type="submit">`: It's a simple form submission button. You can set its value attribute to change the button text, but customization options are more limited compared to `<button>`.*

```
<!-- <button> allows for custom content: -->
<button type="submit">
<img src="submit-icon.png" alt="Submit">
Submit Form
</button>

<!-- <input type="submit"> with limited customization: -->
<input type="submit" value="Submit Form">
```

2. **Styling:**

- `<button>`: You can style it extensively using CSS. This allows you to control button appearance, such as size, color, font, borders, and more.
- `<input type="submit">`: It has a default appearance that varies across browsers. While you can apply some CSS styling, it might not provide the same level of customization as `<button>`.

```css
/* CSS example to style a <button> */
button[type="submit"] {
background-color: #007bff;
color: #fff;
padding: 10px 20px;
border: none;
cursor: pointer;
}
```

3. **Accessibility:**

- `<button>`: It can be made more accessible by providing appropriate ARIA (Accessible Rich Internet Applications) attributes for screen readers. You can include ARIA roles and labels to improve accessibility.
- `<input type="submit">`: It also benefits from ARIA attributes, but customizing its accessibility features may require additional markup.

```html
<!-- Example of ARIA attributes for accessibility -->
<button type="submit" aria-label="Submit Form">
Submit
</button>
```

4. **Browser Consistency:**

- `<input type="submit">` is designed explicitly for form submissions, so it tends to have consistent behavior across different web browsers.
- `<button>` may have slight variations in behavior between browsers, especially when using JavaScript interactions. Testing is essential to ensure consistency.

5. **JavaScript Interactions:**

- `<button>`: It's more versatile when used with JavaScript. You can attach JavaScript event handlers directly to the `<button>` element, allowing you to perform custom actions before or after form submission.

```html
<!-- Example of using JavaScript with a <button> -->
<button type="submit" onclick="validateForm()">Submit Form</button>
```

- `<input type="submit">`: While you can use JavaScript with `<input type="submit">`, it often requires additional JavaScript code to interact with the form because it's a standalone input element.

6. **Form Structure:**

- Consider your form's structure and whether you need multiple submit buttons with different behaviors. In such cases, `<button>` may be more suitable since you can define different actions for each button using JavaScript.

```html
<!-- Example of multiple <button> elements with different actions -->
<button type="submit" onclick="validateForm()">Submit Form</button>
<button type="button" onclick="clearFormFields()">Clear Form</button>
```

7. **Compatibility with Older Browsers:**

- `<input type="submit">` tends to have better compatibility with older web browsers because it's a standard HTML input element.
- If you need to support older browsers, using `<input type="submit">` may be a safer choice.

8. **Markup Considerations:**

- In some cases, you may want to wrap form submission buttons in other elements for layout or styling purposes. `<button>` can be more versatile in such scenarios.

```html
<!-- Example of using <button> within a <div> for styling -->
<div class="submit-button">
<button type="submit">Submit Form</button>
</div>
```

9. **Cross-Browser Testing:**

- Regardless of your choice, it's essential to test your form in multiple web browsers to ensure consistent behavior and appearance.

10. **Accessibility Labels:**

- Accessibility is crucial for ensuring that your web forms are usable by individuals with disabilities. Both `<button>` and `<input type="submit">` can benefit from accessibility attributes.
- It's recommended to use attributes like `aria-label` or `aria-labelledby` to provide meaningful labels for screen readers and assistive technologies.

```html
<!-- Example of using aria-label for accessibility -->
<button type="submit" aria-label="Submit the form">Submit</button>
```

11. **Browser Behavior:**

- While both elements should trigger form submission when clicked, there can be subtle differences in browser behavior. For example, some browsers may use the text within the `<button>` element as the default label for screen readers.

12. **Button Roles:**

- The `<button>` element is more versatile in terms of assigning different roles to buttons. You can use it for form submission (`type="submit"`), triggering JavaScript functions (`type="button"`), or even as simple buttons (`type="button"`) for non-form-related actions.

```html
<!-- Example of using <button> for a non-form-related action -->
<button type="button" onclick="openModal()">Open Modal</button>
```

13. **Compatibility with CSS Frameworks:**

- If you're using a CSS framework like Bootstrap, they often provide pre-styled components for form submission buttons. In such cases, you may want to check the framework's documentation for recommendations on using `<button>` or `<input type="submit">` within their components.

14. **Semantic HTML:**

- When considering which element to use, think about the semantic meaning of the button. `<button>` is more semantically appropriate when the button represents an action, while `<input type="submit">` is a specialized input element for form submission.

15. **Coding Standards:**

- If you're working in a team or following specific coding standards or conventions, it's a good idea to adhere to those standards. Some projects or organizations may have guidelines on when to use `<button>` or `<input>` for form submission buttons.

16. **User Experience (UX):**

- Consider the overall user experience. How do the buttons fit into the design and flow of your website? A well-designed button, regardless of whether it's `<button>` or `<input>`, can enhance the user experience.

In conclusion, the choice between `<button>` and `<input type="submit">` depends on factors such as accessibility, customization, JavaScript interactions, form structure, and compatibility with frameworks. Both elements can effectively handle form submission, but understanding the specific requirements and constraints of your project will help you make an informed choice.