When accessing and manipulating the content of a `<p>` tag, the differences between `textContent` and `innerText` are minimal, but they still exist. Here's a breakdown:

1. \*\*`textContent`\*\*:

- `textContent` returns the full text content of the element, including all text nodes and whitespace, regardless of CSS styling.

- When setting `textContent`, it replaces all existing child nodes of the element with a single text node containing the specified text.

Example:

```html

<p id="myParagraph">Hello <span style="display: none;">World</span></p>

```

```javascript

const paragraph = document.getElementById('myParagraph');

console.log(paragraph.textContent); // Output: "Hello World"

paragraph.textContent = 'New Text';

console.log(paragraph.textContent); // Output: "New Text"

```

2. \*\*`innerText`\*\*:

- `innerText` returns the visible text content of the element, as it would be rendered visually.

- It respects CSS styling, so any text that is visually hidden with CSS will not be included in the result.

- When setting `innerText`, it replaces the visible text content of the element while preserving the structure of child nodes.

Example:

```html

<p id="myParagraph">Hello <span style="display: none;">World</span></p>

```

```javascript

const paragraph = document.getElementById('myParagraph');

console.log(paragraph.innerText); // Output: "Hello"

paragraph.innerText = 'New Text';

console.log(paragraph.innerText); // Output: "New Text"

```

In the context of accessing and manipulating the content of a `<p>` tag, both `textContent` and `innerText` may produce similar results, especially if there are no CSS styles affecting the visibility of text. However, `innerText` may be more suitable if you want to consider CSS styling and exclude visually hidden text.