ElementHandle class

ElementHandle represents an in-page DOM element.

Signature:

```
export declare class ElementHandle<ElementType extends Element = Element> extends
JSHandle<ElementType>
```

Extends: JSHandle<ElementType>

Remarks

ElementHandles can be created with the Page.$() method.

```
const puppeteer = require('puppeteer');

(async () => {
 const browser = await puppeteer.launch();
 const page = await browser.newPage();
 await page.goto('https://example.com');
 const hrefElement = await page.$('a');
 await hrefElement.click();
 // ...
})();
```

ElementHandle prevents the DOM element from being garbage-collected unless the handle is disposed. ElementHandles are auto-disposed when their origin frame gets navigated.

ElementHandle instances can be used as arguments in Page.$eval() and Page.evaluate() methods.

If you're using TypeScript, ElementHandle takes a generic argument that denotes the type of element the handle is holding within. For example, if you have a handle to a `<select>` element, you can type it as `ElementHandle<HTMLSelectElement>` and you get some nicer type checks.

The constructor for this class is marked as internal. Third-party code should not call the constructor directly or create subclasses that extend the `ElementHandle` class.

Methods

Method	Modifiers	Description
$(selector)		Runs `element.querySelector` within the page. If no element matches the selector, the return value resolves to `null`.
$$(selector)		Runs `element.querySelectorAll` within the page. If no elements match the selector, the return value resolves to `[]`.
$$eval(selector, pageFunction, args)		This method runs `document.querySelectorAll` within the element and passes it as the first argument to `pageFunction`. If there's no

		element matching `selector`, the method throws an error.If `pageFunction` returns a Promise, then `frame.$$eval` would wait for the promise to resolve and return its value.
$eval(selector, pageFunction, args)		This method runs `document.querySelector` within the element and passes it as the first argument to `pageFunction`. If there's no element matching `selector`, the method throws an error.If `pageFunction` returns a Promise, then `frame.$eval` would wait for the promise to resolve and return its value.
$x(expression)		The method evaluates the XPath expression relative to the elementHandle. If there are no such elements, the method will resolve to an empty array.
asElement()		
boundingBox()		This method returns the bounding box of the element (relative to the main frame), or `null` if the element is not visible.
boxModel()		This method returns boxes of the element, or `null` if the element is not visible.
click(options)		This method scrolls element into view if needed, and then uses Page.mouse to click in the center of the element. If the element is detached from DOM, the method throws an error.
contentFrame()		Resolves to the content frame for element handles referencing iframe nodes, or null otherwise
focus()		Calls focus on the element.
hover()		This method scrolls element into view if needed, and then uses Page.mouse to hover over the center of the element. If the element is detached from DOM, the method throws an error.
isIntersectingViewport()		Resolves to true if the element is visible in the current viewport.
press(key, options)		Focuses the element, and then uses Keyboard.down() and Keyboard.up().
screenshot(options)		This method scrolls element into view if needed, and then uses Page.screenshot() to take a screenshot of the element. If the element is detached from DOM, the method throws an error.
select(values)		Triggers a `change` and `input` event once all the provided options have been selected. If there's no `<select>` element matching `selector`, the method throws an error.
tap()		This method scrolls element into view if needed, and then uses Touchscreen.tap() to tap in the center of the element. If the element is detached from DOM, the method throws an error.
type(text, options)		Focuses the element, and then sends a `keydown`, `keypress`/`input`, and `keyup` event for each character in the text.To press a special key, like `Control` or `ArrowDown`, use ElementHandle.press().

uploadFile(filePaths)		This method expects `elementHandle` to point to an input element.