Page.$eval() method

This method runs `document.querySelector` within the page and passes the result as the first argument to the `pageFunction`.

Signature:

```
$eval<ReturnType>(selector: string, pageFunction: (element: Element, ...args:
unknown[]) => ReturnType | Promise<ReturnType>, ...args: SerializableOrJSHandle[]):
Promise<WrapElementHandle<ReturnType>>;
```

Parameters

Parameter	Type	Description
selector	string	the selector to query for
pageFunction	(element: Element, ...args: unknown[]) => ReturnType | Promise<ReturnType>	the function to be evaluated in the page context. Will be passed the result of `document.querySelector(selector)` as its first argument.
args	SerializableOrJSHandle[]	any additional arguments to pass through to `pageFunction`.

Returns:

Promise<WrapElementHandle<ReturnType>>

The result of calling `pageFunction`. If it returns an element it is wrapped in an ElementHandle, else the raw value itself is returned.

Remarks

If no element is found matching `selector`, the method will throw an error.

If `pageFunction` returns a promise `$eval` will wait for the promise to resolve and then return its value.

Example 1

```
const searchValue = await page.$eval('#search', el => el.value);
const preloadHref = await page.$eval('link[rel=preload]', el => el.href);
const html = await page.$eval('.main-container', el => el.outerHTML);
```

If you are using TypeScript, you may have to provide an explicit type to the first argument of the `pageFunction`. By default it is typed as `Element`, but you may need to provide a more specific sub-type:

Example 2

```
// if you don't provide HTMLInputElement here, TS will error
// as `value` is not on `Element`
const searchValue = await page.$eval('#search', (el: HTMLInputElement) => el.value);
```

The compiler should be able to infer the return type from the `pageFunction` you provide. If it is unable to, you can use the generic type to tell the compiler what return type you expect from `$eval`:

Example 3

```
// The compiler can infer the return type in this case, but if it can't
// or if you want to be more explicit, provide it as the generic type.
const searchValue = await page.$eval<string>(
 '#search', (el: HTMLInputElement) => el.value
);
```