

[Home](#) > [puppeteer](#) > [Page](#) > [evaluate](#)

## Page.evaluate() method

**Signature:**

```
evaluate<T extends EvaluateFn>(pageFunction: T, ...args: SerializableOrJSHandle[]):  
    Promise<UnwrapPromiseLike<EvaluateFnReturnType<T>>>;
```

### Parameters

Parameter	Type	Description
pageFunction	T	a function that is run within the page
args	<a href="#">SerializableOrJSHandle</a> []	arguments to be passed to the pageFunction

**Returns:**

Promise<[UnwrapPromiseLike](#)<[EvaluateFnReturnType](#)<T>>>

the return value of `pageFunction`.

### Remarks

Evaluates a function in the page's context and returns the result.

If the function passed to `page.evaluateHandle` returns a Promise, the function will wait for the promise to resolve and return its value.

### Example 1

```
const result = await frame.evaluate(() => {  
    return Promise.resolve(8 * 7);  
});  
console.log(result); // prints "56"
```

You can pass a string instead of a function (although functions are recommended as they are easier to debug and use with TypeScript):

### Example 2

```
const aHandle = await page.evaluate('1 + 2');
```

To get the best TypeScript experience, you should pass in as the generic the type of `pageFunction`:

```
const aHandle = await page.evaluate<() => number>(() => 2);
```

### Example 3

[ElementHandle](#) instances (including [JSHandles](#)) can be passed as arguments to the `pageFunction` :

```
const bodyHandle = await page.$('body');
const html = await page.evaluate(body => body.innerHTML, bodyHandle);
await bodyHandle.dispose();
```