

```
function getPonies(size = 10, page = 1) {
  // ...
  server.get(size, page);
}
```

Now it is perfectly clear that the `size` parameter will be `10` and the `page` parameter will be `1` if not provided.

NOTE

There is a small difference though, as now `0` or `""` are valid values and will not be replaced by the default one, as `size = size || 10` would have done. It will be more like `size = size === undefined ? 10: size;`.

The default value can also be a function call:

```
function getPonies(size = defaultSize(), page = 1) {
  // the defaultSize method will be called if size is not provided
  // ...
  server.get(size, page);
}
```

or even other variables, either global variables, or other parameters of the function:

```
function getPonies(size = defaultSize(), page = size - 1) {
  // if page is not provided, it will be set to the value
  // of the size parameter minus one.
  // ...
  server.get(size, page);
}
```

Note that if you try to access parameters on the right, their value is always `undefined`:

```
function getPonies(size = page, page = 1) {
  // size will always be undefined, as the page parameter is on its right.
  server.get(size, page);
}
```

This mechanism for parameters can also be applied to values, for example when using a destructuring assignment:

```
const { timeout = 1000 } = httpOptions;
// you now have a variable named 'timeout',
// with the value of 'httpOptions.timeout' if it exists
// or 1000 if not
```