Example #3 – Lesson 5

[[1, 2], [3, 4]].map(arr => {

console.log(arr[0]);

return arr[0];

});

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action** | **Performed On** | **Side Effect** | **Return Value** | **Is Return Value Used?** |
| Method call map( ) | Outer array | None | Array copy w/ transformed elements [1, 3] | Yes, input in new copy of arr |
| Callback execution | Each sub array | None | value at index[0] of each sub array | Yes, used by map( ) for returned arr copy |
| Element reference ([0]) | Each sub array | None | Element at index 0 of each sub array | Yes, used by console.log and by the callback function |
| Method call console.log | Element at index 0 of each sub array | Outputs a string representation of an integer | Undefined | No |

Examples #4 – Lesson 5

let myArr = [[18, 7], [3, 12]].forEach(arr => {

return arr.map(num => {

if (num > 5) {

return console.log(num);

}

});

});

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action** | **Performed On** | **Side Effect** | **Return Value** | **Is Return Value Used?** |
| Assignment | N/A | None | undefined | No |
| Method call forEach() | Outer array | None | undefined | Yes, assigned to myArr |
| Out callback execution | Each sub array | None | Copy of each sub array, transformed by map | No, forEach doesn’t use the return value of the callback function |
| Method call map() | On each sub array | None | Modified copy of each sub array | Yes, explicitly returned by the callback |
| Callback execution | Each element of the sub arrays | None | Undefined | Yes, used by map for transformation |
| Comparison | On each element of the sub arrays | None | True or False | Yes, to either run or not the if code block |
| Method call console.log() | Each element of the sub arrays that returns true from the comparison | Outputs string representation of a Number | Undefined | Yes, by the explicit return from the callback |
|  |  |  |  |  |

Examples #5 – Lesson 5

[[1, 2], [3, 4]].map(arr => {

return arr.map(num => num \* 2);

});

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action** | **Performed On** | **Side Effect** | **Return Value** | **Is Return Value Used?** |
| Method call map() | Outer array | none | A copy of the outer array, modified | No |
| Callback execution | Each Sub array | none | Copy of each sub array, modified | Yes, by map() to modify the outer array |
| Method call  map() | Each sub array | none | Copy of each sub array, modified | Yes, by the explicit return, to callback of the outer function |
| Callback execution | Each element of each sub array | none | Each element \*2 | Yes, by map() to transform each sub array |
| (\*) multiplication | Each element of each sub array | none | Each element \* 2 | Yes, by the implicit return on the callback function on each iteration |

Examples #6 – Lesson 5

[{ a: 'ant', b: 'elephant' }, { c: 'cat', d: 'dog' }].filter(object => {

return Object.keys(object).every(key => object[key][0] === key);

});

// => [ { c: 'cat', d: 'dog' } ]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action** | **Performed On** | **Side Effect** | **Return Value** | **Is Return Value Used?** |
| Method call map() | Outer array | none | A copy of the outer array, modified | No |
| Callback execution | Each Sub array | none | Copy of each sub array, modified | Yes, by map() to modify the outer array |
| Method call  map() | Each sub array | none | Copy of each sub array, modified | Yes, by the explicit return, to callback of the outer function |
| Callback execution | Each element of each sub array | none | Each element \*2 | Yes, by map() to transform each sub array |
| (\*) multiplication | Each element of each sub array | none | Each element \* 2 | Yes, by the implicit return on the callback function on each iteration |