Recursion

id = 0;

function recursion1(obj, arr, lev) {

if (!lev) lev = 0;

var i = 0;

var keys = obj instanceof Object ? Object.keys(obj) : [];

for (var key of keys) {

var objval = obj[key];

arr[lev] = i + 1;

arr = arr.slice(0, lev + 1);

id++;

console.log(id, lev+1, i+1, arr);

lev++;

recursion1(objval, arr, lev);

lev--;

i++;

}

}

recursion1(foo,[]);

var foo = {

bar: 'a',

child: {

b: 'b',

grand: {

greatgrand: {

c: { d: 'd', e: 'e', f: [{ g: 'g', h: 'h' }, { i: 'i' }] }

, j: ['k', 'l']

}

}

}

}

// https://christianlydemann.com/how-to-do-tree-searching-with-javascript/

function containsInNestedObjectDF(obj, val) {

if (obj === val) {

return true;

}

const keys = obj instanceof Object ? Object.keys(obj) : [];

for (const key of keys) {

const objval = obj[key];

const isMatch = containsInNestedObjectDF(objval, val);

if (isMatch) {

return true;

}

}

return false;

}

console.log( containsInNestedObjectDF(foo, "i"));

/\*

https://stackoverflow.com/questions/2549320/looping-through-an-object-tree-recursively

\*/

// use this recursive function with a parse funciton

function parseObjectProperties(obj, parse) {

for (var k in obj) {

if (typeof obj[k] === 'object' && obj[k] !== null) {

console.log(k);

parseObjectProperties(obj[k], parse)

} else if (obj.hasOwnProperty(k)) {

parse(obj[k])

}

}

}

//\*\*\*

// then apply to the property the task you want, in this case just console

parseObjectProperties(foo, function (prop) {

console.log(prop)

})