**Basic Algorithm Truncate**

**Function truncateText**(text, num) {

let lengthText = text.length;

if (lengthText > num && num > 3){

Let slice = lengthText.slice(0, (num)) + ‘…’;

return slice;

} else if (lengthText > num && num <= 3) {

Let slice = lengthText.slice(0, num) + ‘…’;

Return slice;

} else {

Return text;

}

truncateText("A-tisket a-tasket A green and yellow basket", 8);

**Basic Algorithm Finders Keeper**

**Function findElement**(arr, func) {

Let num = 0;

For (var i=0; I <= arr.length; i++){

Num = arr[i];

If(func(num)){

Return num;

}

}

Return undefined;

**}**

findElement([1, 2, 3, 4], num => num % 2 === 0);

**Basic Algorithm Boo Who**

**Function booWho**(bool){

Return typeof bool === ”Boolean”;

}

booWho(null);

**Basic Algorithm Falsy Bouncer**

**Function** bouncer(arr){

Return arr.filter(Boolean);

}

Bouncer([7, “ate”, “”, false, 9]);

**Basic Algorithm Where Do I Belong**

**Function** getIndexToIns(arr, num) {

arr.sort(function(i,b){

return i-b;

});

for(let i=0; i < arr.length; i++){

if(arr[i] >= num)

return i;

}

return arr.length;

}

getIndexToIns([40, 60], 50);

**Basic Algorithm Mutations**

**function** mutation(arr) {

let test, target;

test = arr[1].toLowerCase();

target = arr[0].toLowerCase();

for(let i = 0; i < test.length; i++){

if(target.indexOf(test[i]) < 0)

return false;

}

return true;

}

mutation(["hello", "hey"]);

**Basic Algorithm Chunky Monkey**

**function** chunkArrayInGroups(arr, size) {

var arr2 = [];

for (var i = 0; i < arr.length; i+=size) {

arr2.push(arr.slice(i , i+size));

}

return arr2;

}