Chapter 4 Exercises

The Sum of a Range

```
function range(start, end) {
  var myArray = [];
  for (i= start; i <= end; i++) {
    myArray.push(i);
  };
  return myArray;
};

function sum(rangeArray) {
  var rangeSum = 0;
  for (i=0; i <= rangeArray.length; i++) {
    rangeSum += i;
  };
  return rangeSum;
};

console.log(range(1, 10));
// → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
console.log(range(5, 2, -1));
// → [5, 4, 3, 2]
console.log(sum(range(1, 10)));
// → 55
```

Reversing An Array

```
function reverseArray(array) {
  var revArray = [];
  for (i = 0; i < array.length; i++) {

    var temp = array[i];
    revArray.unshift(temp);


  };
  return revArray;
};
```

```
function reverseArrayInPlace(array) {

  var half = (array.length)/2;
  var half = Math.floor(half);

 for (i = 0; i < half; i ++) {
    var temp = array[i];
    var farthest = array[array.length - 1 - i];
    array[i] = farthest;
    array[array.length - 1 - i] = temp;
  };

  return array;

};


console.log(reverseArray(["A", "B", "C"]));
// → ["C", "B", "A"];

var arrayValue = [1, 2, 3, 4, 5];

reverseArrayInPlace(arrayValue);
console.log(arrayValue);
// → [5, 4, 3, 2, 1]
```

A List

```
function arrayToList(myArray) {
        for ( i = (myArray.length -1); i >= 0; i-- )
    {
      var singleObj = {};
      singleObj = myArray[i];
      var List = {value: singleObj, rest: List};
    };
        return List;
};


function listToArray(myList) {
  var resultArray = [];
  for (var node = myList; node; node = node.rest) {
    resultArray.push(node.value);
  };
  return resultArray;
```

```
  //produces an array from a list
};

function prepend(myElement, myList) {

  //takes an element and a list and creates a new list that adds
  //the element to the front of the input list
};

function nth(myList, myNumber) {

  //takes a list and a number and returns
  //the element at the given position in the list

  //or undefined when there is no such element
  //recursive
};

console.log(arrayToList([10, 20]));
// → {value: 10, rest: {value: 20, rest: null}}
console.log(listToArray(arrayToList([10, 20, 30])));
// → [10, 20, 30]
console.log(prepend(10, prepend(20, null)));
// → {value: 10, rest: {value: 20, rest: null}}
console.log(nth(arrayToList([10, 20, 30]), 1));
// → 20
```

// prepend

```
function prepend(myElement, myList) {
  if (myList == null) {
        var myNewList = {value: singleObj, rest: myList};
  }
  else {
    var myNewList = myList;
  };
  var singleObj = myElement;
  var myNewList = {value: singleObj, rest: myList};
  return myNewList;
};
```

// nth

```
function nth(myList, myNumber) {

  console.log(myList);
  var counter = 0;
  for (var node = myList; node; node = node.rest) {
        counter ++;
    if (counter == myNumber) {
      console.log("bob", node.value);
      return node.value;
    }

  };
  return undefined;

};
```

//nth recursive

```
var counter = 0;

function nth(myList, myNumber) {
//console.log(counter);
  if (myList.rest == null) {
    console.log ("null", myList.value);
    return false;

  }

  else if (counter == myNumber) {
    console.log ("here", myList.value);
    return myList.value;

  }

  else {
    counter ++;
    console.log("counter",counter);
    myList = myList.rest;
    nth(myList, myNumber);
    //console.log("made it here");
  };
};
```

egghead.io react lessons

// deep comparison

```
function deepEqual(value1, value2) {

  if (( ( (typeof value1) == "object") && ( (typeof value1) != null) )  && ( ( (typeof value2)
== "object") && ( (typeof value2) != null) )) {
    if (value1 === value2) {
      return true;
    }

    else {
      return (value1 === value2);
    };
  };


};
```

//deep comparison 2

```
// take 2 values
// return true if value1 = value2 or value1 & value 2 are objects
//w/same properties whose values are equal
//recursive call to figure out whether values are equal
//


function deepEqual(value1, value2) {

  if (( ( (typeof value1) == "object") && ( (typeof value1) != null) )  && ( ( (typeof value2)
== "object") && ( (typeof value2) != null) )) {

    console.log(Object.keys(value1)[0]);
        console.log(Object.keys(value2));
    //deep comparison
    //console.log(value1.keys);

        // count properties in objects and compare number
                // false if different
        //if the same go over one objects properties and compare to other object
        // compare values of properties with a recursive call to deep equal
        // false if different
        //return true at end of function if true
```

```
    //for(variable in object) {};



  }

  else {
      return (value1 === value2);
  };
};

var obj = {here: {is: "an"}, object: 2};
console.log(deepEqual(obj, obj));
// → true
console.log(deepEqual(obj, {here: 1, object: 2}));
// → false
console.log(deepEqual(obj, {here: {is: "an"}, object: 2}));
// → true
```