**Week 1 Day 3**

**Arrays**

* Useful array methods:
  1. *.filter()* filters an array according to a set of criteria. Filter takes true or false for each item and returns an array of values that fit the criteria. This method is non-mutating. Syntax:

*const filterItems = items.filter((item) => {*

*return item.price < 100*

*})*

* 1. *map()* is similar to filter – it maps each item of an array onto a new array according to a function. Syntax:

*const mapItems = items.map((item) => {*

*return item.price*

*})*

* 1. *find()* allows you to find a single item in an array. This will return the array element. Searching according to an object property in an array will still return the array element. Syntax:

*const findItem = items.find((item) => {*

*return item.name === ‘Album’*

*})*

* 1. *forEach()* returns undefined so return statement is not needed. *forEach()* is a mutating method. You do not assign *forEach()* to a new variable since it mutates the original. Syntax:

*items.forEach((item) => {*

*console.log(item.name)*

*})*

* 1. *some()* checks whether an array contains at least one of a given value and returns true/false (not a new array). Syntax:

*const hasInexpensiveItems = items.some((item) => {*

*return item.price <=100*

*})*

* 1. *every()* is effectively the same as some but it checks every array item. Syntax:

*const hasInexpensiveItems = items.every((item) => {*

*return item.price <=100*

*})*

* 1. *reduce()* takes a function and a starting value. The function takes an item and a property for what we want to reduce everything into. The property always comes first syntactically. In each iteration, the accumulator (property) is set to the return value of the function from the previous iteration – if no return then undefined. Good for cumulative operations. Syntax:

*const total = items.reduce((currentTotal, item) => {*

*return item.price/currentTotal*

*}, 0)*

* 1. *include()* checks that whatever is passed into the include method is in the array. Syntax: *const includesTwo = items.includes(2)*
* What is a REPL?
  1. A REPL is a read-evaluate-print loop. It is a simple programming environment that takes a user input, evaluates it and returns the output to the user
* How would you launch a node REPL?
  1. To launch a node REPL, install node and then run node in the command line without a file name.
* Task:

const basket = [  
 { name: 'Apple', priceInPence: 50, quantity: 4 },  
 { name: 'Orange', priceInPence: 80, quantity: 2 },  
 { name: 'Carrots', priceInPence: 20, quantity: 10 },  
 { name: 'Strawberries', priceInPence: 150, quantity: 1 }  
]

* 1. Which array method would I need to find the total number of items in my basket?
     + reduce
  2. Which array method would I need to see if there are "Grapes" in my basket?
     + some
  3. Which array method would I need to get a list of all the item names in my basket?
     + map
  4. Which array method would I need to find all the items over £1 in my basket?
     + filter