**Higher Order Functions & Arrays**

**Datasets used**

const companies = [

  { name: "Company One", category: "Finance", start: 1981, end: 2004 },

  { name: "Company Two", category: "Retail", start: 1992, end: 2008 },

  { name: "Company Three", category: "Auto", start: 1999, end: 2007 },

  { name: "Company Four", category: "Retail", start: 1989, end: 2010 },

  { name: "Company Five", category: "Technology", start: 2009, end: 2014 },

  { name: "Company Six", category: "Finance", start: 1987, end: 2010 },

  { name: "Company Seven", category: "Auto", start: 1986, end: 1996 },

  { name: "Company Eight", category: "Technology", start: 2011, end: 2016 },

  { name: "Company Nine", category: "Retail", start: 1981, end: 1989 },

];

const ages = [33, 12, 20, 16, 5, 54, 21, 44, 61, 13, 15, 45, 25, 64, 32];

**ForEach**

**Usecase** - ForEach is used is an array-specific method that is used to loop through an array

**Syntax** – normal function and arrow function **(index and allElements are optional)**

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

  console.log(companies[i]);

}

companies.forEach(function (element, index, allElements) {

  console.log(element);

  console.log(index);

  console.log(allElements);

});

companies.forEach((element, index, allElements) => {

  console.log(element);

  console.log(index);

  console.log(allElements);

});

A screenshot of a computer

Description automatically generated with medium confidence etc. until all elements are logged (9 elements)

**Filter**

**Usecase** – Filter is used to filter items out of an array. Means that you don’t have to push the items into another array. For example, extracting all ages over 21 from the ages array using a for loop.

let canDrink = [];

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

  if (ages[i] >= 21) {

    canDrink.push(ages[i]);

  }

}

**Syntax** – normal function and arrow function **(index and allElements are optional)**

let canDrinkFilterMethod = ages.filter(function (element) {

  if (element >= 21) {

    return true; // return true means that the item will be pushed to the variable

  }

});

Graphical user interface, text, application, website

Description automatically generated

Arrow function refined as one liner

let canDrinkFilterMethodArrow = ages.filter((ages) => {

  ages >= 21;

});

More Examples:

Filter out all companies that are retail

const retailCompanies = companies.filter(function (element) {

  if (element.category === "Retail") {

    return true;

  }

});

console.log(retailCompanies);

const retailCompaniesArrow = companies.filter(

  (element) => element.category === "Retail"

);

console.log(retailCompaniesArrow);

Text

Description automatically generated

Companies that started in the 1980s

const start1980s = companies.filter(function (element) {

  if (element.start >= 1980 && element.start < 1990) {

    return true;

  }

});

console.log(start1980s);

const start1980sArrow = companies.filter(

  (element) => element.start >= 1980 && element.start < 1990

);

console.log(start1980sArrow);

Text

Description automatically generated

Companies that lasted at least 10 years

const lasted10years = companies.filter(function (element) {

  if (element.end - element.start >= 10) {

    return true;

  }

});

console.log(lasted10years);

const lasted10yearsArrow = companies.filter(

  (element) => (element.end = element.start >= 10)

);

console.log(lasted10years);

**Map**

**Usecase** – Map is used to create new arrays from a current array.

**Syntax** – element parameter is mandatory**. Index, wholeArray and thisArg** (replacement for this keyword) are optional. Normal and Arrow function -

const companyNames = companies.map(function (element, index, wholeArray, thisArg) {

  return element.name;

});

const companyNamesArrow = companies.map( element => element.name);

Graphical user interface, text

Description automatically generated

Any value that is put in the return statement will replace all the elements in the generated array. For example:

const mapOtherValues = companies.map(function (element) {

  return "other values";

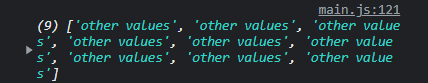
});

console.log(mapOtherValues);

const mapOtherValuesArrow = companies.map(

  (element) => "other values arrow function"

);



const companiesStartAndEndMap = companies.map(function (element) {

  return `${element.name} [${element.start} - ${element.end}] `;

});

console.log(companiesStartAndEndMap);

const companiesStartAndEndMapArrow = companies.map(

  (element) => `${element.name} [${element.start} - ${element.end}]`

);

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

const agesSquareRt = ages.map((element) => Math.sqrt(element));

console.log(agesSquareRt);

Text

Description automatically generated

Maps can be chained. For example, all the ages will be multiplied by 10 then 0.5 will be added to them

Text

Description automatically generated

Text

Description automatically generated

**Sort**

**Usecase** – Sorts an array depending on a condition given in the sort function.

**Syntax –** Sort can be functionless. Optional parameters include firstElement and secondElement for comparison, a compare function or an inline function with firstElement and secondElement.

const sortedCompanies = companies.sort(function (c1, c2) {

  if (c1.start > c2.start) {

    return 1;

  } else {

    return -1;

  }

});

console.log(sortedCompanies);

Return **1** or negative **-1**. 1 takes the element to the bottom of the array, -1 takes the element to the top of the array

**Graphical user interface, text

Description automatically generated**

console.log(sortedCompanies);

const sortedCompaniesArrow = companies.sort((a, b) =>

  a.start > b.start ? 1 : -1

);

Shorter form using ternary operator

Graphical user interface, text

Description automatically generated



Just sorting the numbers doesn’t work, because sort() just look at the first number.

const sortAgesProperly = ages.sort(function (a, b) {

  return a - b;

});

console.log(sortAgesProperly);

if a – b gives a negative value, a will be above b in sorted array. (so ascending)

if b – a, the sorting will be descending.

A screenshot of a computer

Description automatically generated with medium confidence

**Reduce**

**Usecase** – Sorts an array depending on a condition given in the sort function.

**Syntax –** Sort can be functionless. Optional parameters include firstElement and secondElement for comparison, a compare function or an inline function with firstElement and secondElement.