# Exercise 4: JSX and ES6

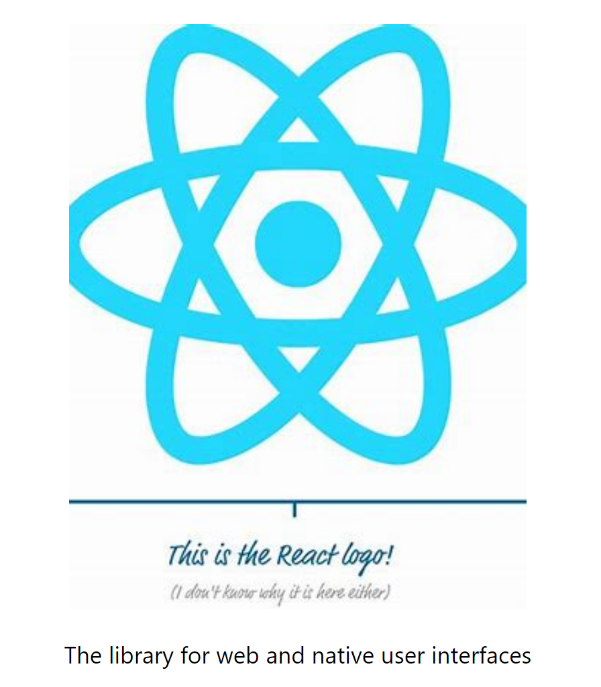
**Objectives and Outcomes**

JSX (JavaScript XML) and ES6 (ECMAScript 2015) are two important concepts in modern JavaScript development, often used together in frameworks like React. By the end of this exercise, you should have a good overview of JSX and ES6.

Exercises

1. Design this website as image below



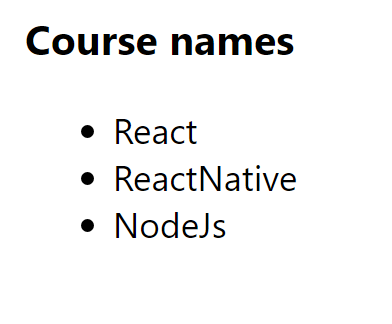
1. Design this website as image below
2. Create a navbar as image below with JSX



1. Display this text



1. Display list of course



Using ES6 and JSX

1. Do all requires that based on the variable as below

var people = [

{name: 'Jack', age: 50},

{name: 'Michael', age: 9},

{name: 'John', age: 40},

{name: 'Ann', age: 19},

{name: 'Elisabeth', age: 16}

]

* Find the first person off the people array is teenager (age >=10 and age <=20)
* Find the all person of the people array is teenager (age >=10 and age <=20)
* Check if every person of the people array is teenager (age >=10 and age <=20), which should return true or false
* Checks if any person of the people array is teenager (age >=10 and age <=20), which should return true or false.

1. Do all requires that based on the variable as below

var array = [1, 2, 3, 4]

* Applies a function passed as the first parameter against an accumulator and each element in the array (from left to right), thus reducing it to a single value. The initial value of the accumulator should be provided as the second parameter of the reduce function.
* Implementation of very simple functions (like the aforementioned sum or product) requires writing a lot of boilerplate. Is there any remedy for that? just try arrow functions!

1. Do all requires that based on three variables as below

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];

const person = {

name: "Costas",

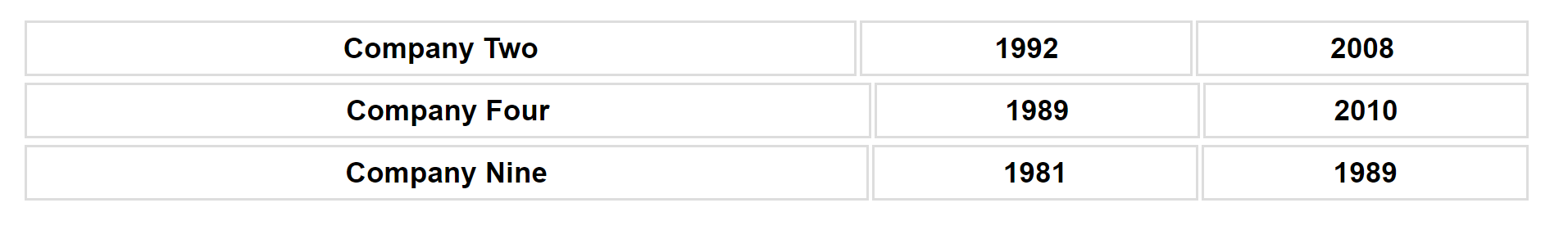
address: {

street: "Lalaland 12"

}

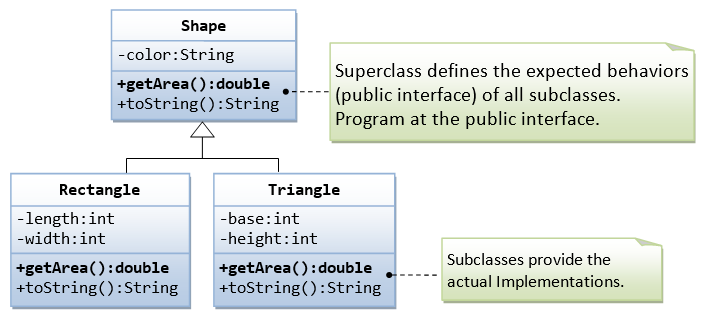
};

* Print the name of each company using forEach
* Print the name of each company that started after 1987
* Get only the companies that have category Retail, increment their start by 1 and append in the DOM a div that has the name, the category, the start and the end in paragraphs elements



* Sort the companies based on their end date in ascending order
* Sort the ages array in descending order
* Print the sum if you add all the ages using reduce
* Make a new object that has the properties of name and category same as the companies [0] and a method print that prints out the name, use object restructuring and ES6 JS
* Create a function that takes an unknown number of arguments that are numbers and return their sum;
* Make a function that takes an unknown number of arguments of any type and adds them in an array and returns the array, if the argument is an array, it should add its values to the array that will be returned by the function
* Destructuring the property street in a variable named street from the object person
* Write a function that every time you call it, it returns a number that increments starting from 0
* Create a function that destructors the query parameters of a URL and adds them in an object as key value pairs and then returns the object

1. Create classes as image below



1. Promises

Promise promises that you would get in future results of deferred or long-running tasks. Promise has two channels: the first for results, the second for potential errors. To get the result, you provide the callback function as the ‘then’ function parameter. To handle errors, you provide the callback function as the ‘catch’ function parameter.

Write promise function that displays random number larger than 5. If number is small than or equal to 5, please show notice: “Error”

**Conclusions**

In this exercise you have learnt to use JSX and ES6 features in your projects.