# 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

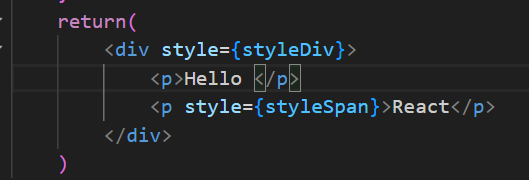
1. Design this website as image below



B1: tạo packet components sau đó tạo file HelloReact.js

B2: viết function HelloReact để trả về giao diện JSX





B3: định nghĩa các biến để style css cho giao diện



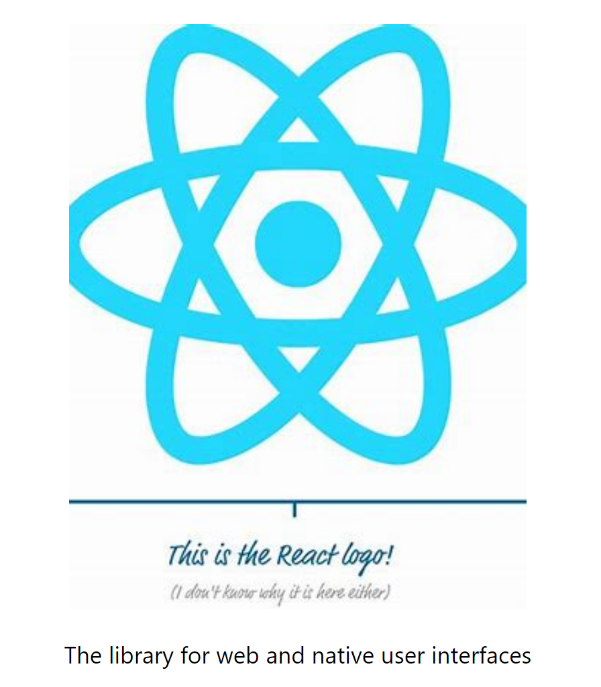
B4: Xuất component và import ở App.js





B5: gọi và khởi chạy hàm



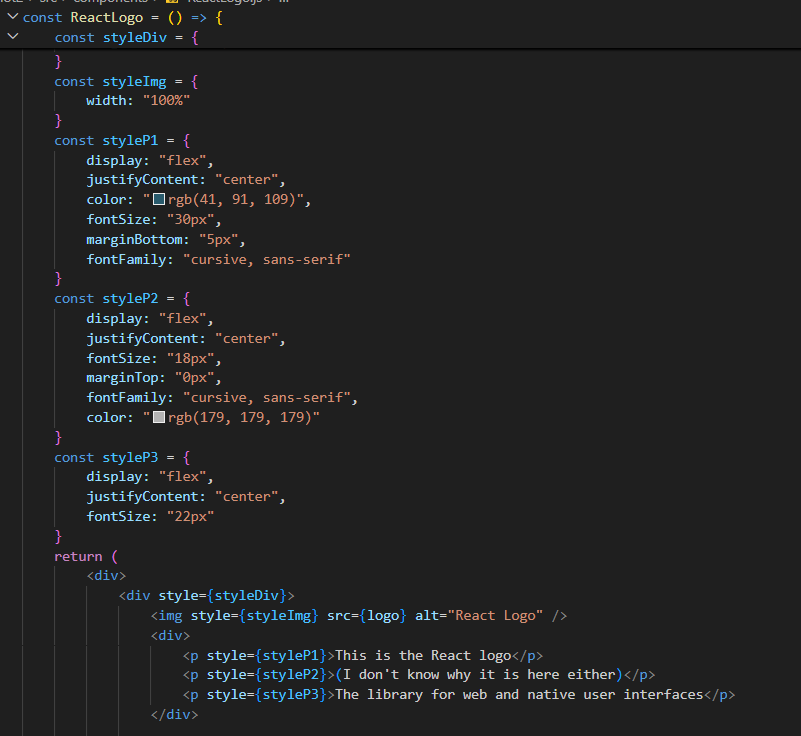
1. Design this website as image below

B1: tương tự câu 1…

B2: import logo react để sử dụng



B3: tạo hàm ReactLogo, trả về giao diện JSX, định nghĩa các biến để style cho giao diện



B4: Xuất component, import bên App.js, gọi và chạy hàm





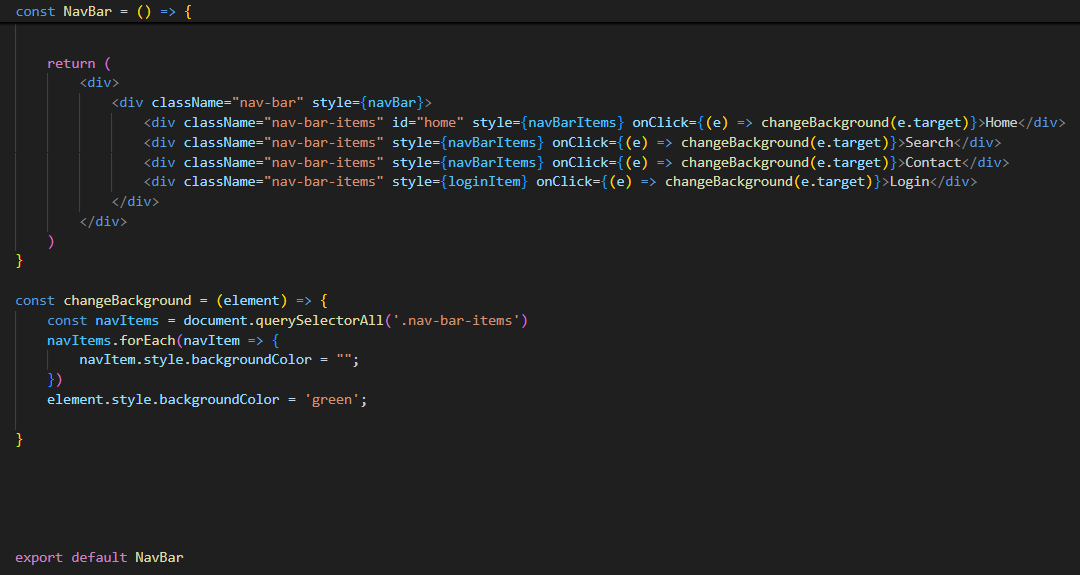


1. Create a navbar as image below with JSX



B1: tương tự các câu trên…

B2: Tạo hàm NavBar trả về giao diện, định nghĩa các biến style để style giao diện, xuất component, import ở App.js, gọi và chạy



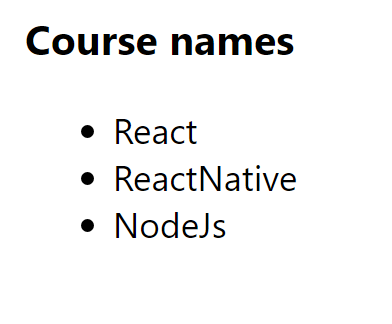


1. Display this text



Tương tự các bài trên

1. Display list of course



Tương tự các bài trên

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},

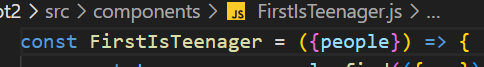
{name: 'Ann', age: 19},

{name: 'Elisabeth', age: 16}

]

* Find the first person off the people array is teenager (age >=10 and age <=20)

B1: Tạo file FirstIsTeenager.js, khởi tạo hàm FirstIsTeenager, Destructuring Prop nhận danh sách đối tượng thông qua prop {people} chứa thông tin name và age

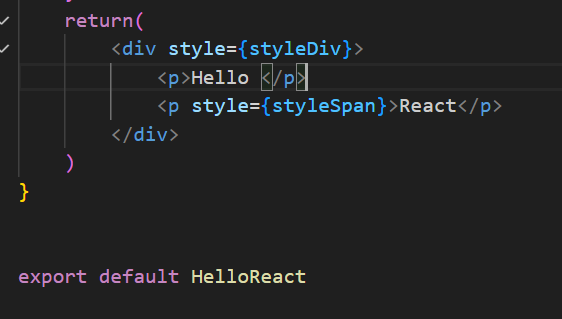


B2: khởi tạo biến teenager sẽ bằng people dùng phương thức .find() để tìm phẩn tử đầu tiên trong mảng thỏa mãn điều kiện 10<= age <= 20

Desrtucturing {age}: chỉ lấy thuộc tính age trong mảng people và kết quả sẽ được lưu trong biến teenager



B3: trả về JSX hiển thị nội dung, xuất component, import ở file App.js, gọi và truyền prop vào component và hiển thị ra kết quả







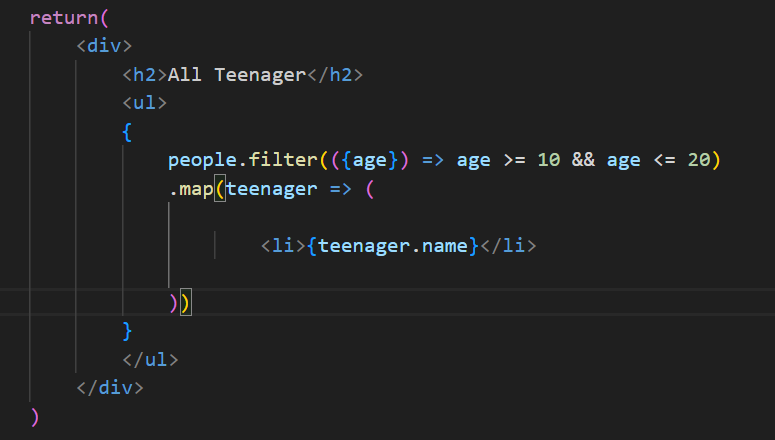
* Find the all person of the people array is teenager (age >=10 and age <=20)

B1: Tương tự như câu trên Khác mỗi tên file và hàm là AllIsTeenager



B2: Cho trả về JSX, bên trong sẽ dùng phương thức .filter() để lọc, destructuring {age} để lấy giá trị {age} của mỗi people và trả về kết mảng chứa các đối tượng có 10 <= age <= 20

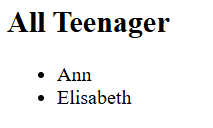
Dùng phưng thức .map() duyệt qua với mỗi phần tử teenager sẽ trả về một thẻ <li> để hiển thị tên



B3: Xuất component, import component gọi hàm truyền prop là people vào component và chạy







* Check if every person of the people array is teenager (age >=10 and age <=20), which should return true or false

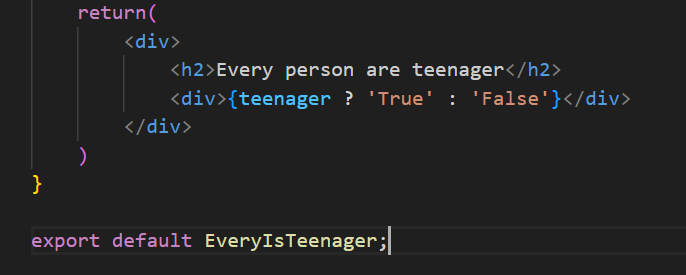
B1: Tương tự khác tên là EveryIsTeenager



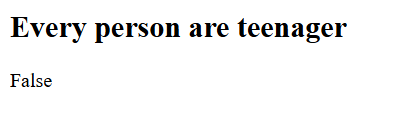
B2: khởi tạo biến teenager, dùng phương thức .every() để kiểm tra tất cả các phần tử có trong mảng thỏa điều kiện 10 <= age <= 20 và trả về true (nếu tất cả thỏa) hoặc false (nếu ít nhất 1 phần tử không thỏa điều kiện và lưu vào trong teenager



B3: Trả về giao diện JSX sử dụng toán tử để in ra True hoặc False, xuất component, import gọi hàm truyền prop people vào component và chạy





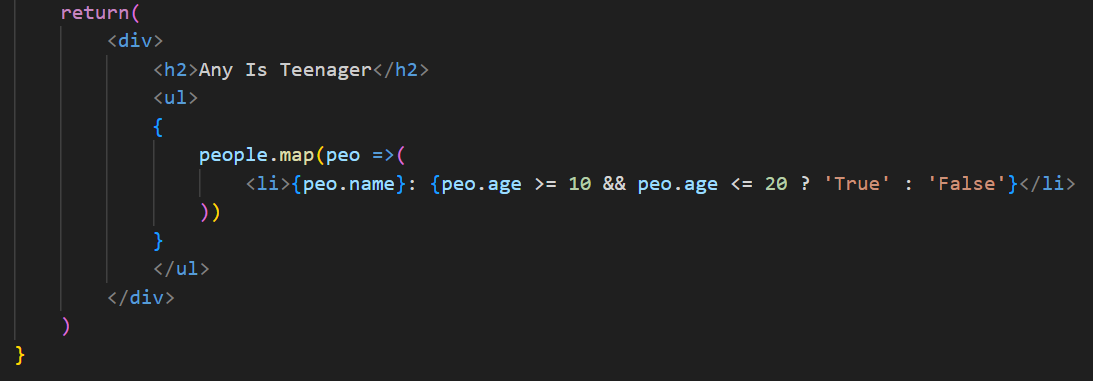


* Checks if any person of the people array is teenager (age >=10 and age <=20), which should return true or false.

B1: Tương tự khác tên là AnyIsTeenager



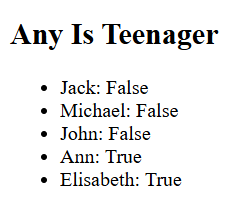
B2: Cho trả về giao diện JSX, bên trong dùng phương thức .map() để duyệt qua people trả về từng biến peo, mỗi biến peo sẽ xuất ra 1 thẻ <li> hiển thị tên và dùng toán tử để xác đinh tuổi có nằm trong điều kiện 10 <= age <= 20 để hiển thị True Hoặc False



B3: Xuất component, import, gọi hàm truyền prop là people vào component và chạy







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.

B1: Tương tự, khác tên là Accumulator



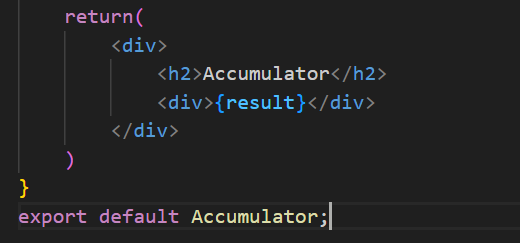
B2: Khai biến result, array sử dụng phương thức .reduce() (phương thức này dùng để thu gọn mảng thành 1 giá trị duy nhất và ở đây sẽ là tính tổng tích lũy các phần tử



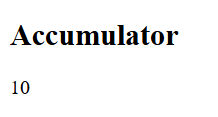
a là giá trị tích lũy, b là giá trị hiện tại của mảng, 0 là giá trị khởi tạo a ban đầu sẽ = 0

sau khi tính xong sẽ lưu vào result

B4: trả về giao diện in ra result, xuất component, import, gọi hàm và truyền prop là array vào component và in ra kết quả







* 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"

}

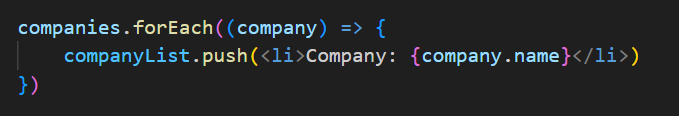
};

* Print the name of each company using forEach

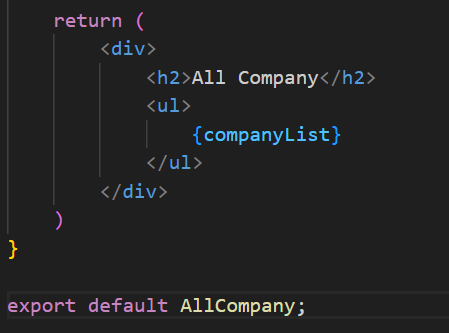
B1: Tương tự, khác tên là AllCompany



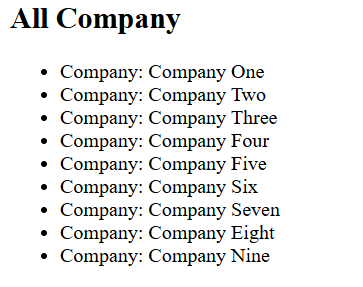
B2: Khởi tạo mảng companyList, dùng phương thức .forEach() để lập lại từng phần tử trong companies lấy thông tin hiện tại thông qua đối tượng company và sử dụng .push() để đẩy phần tử <li> chứa tên company vào trong mảng companyList



B3: Trả về giao diện JSX xuất, import component gọi và truyền prop vào và chạy







* Print the name of each company that started after 1987

B1: Tương tự khác tên là CompanyAfter1987



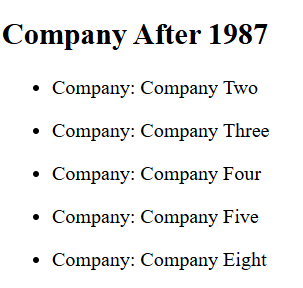
B2: Trả về JSX bên trong sử dụng phương thức .filter() để lọc companies, destructuring {start} để lấy giá trị năm bắt đầu và trả về mảng thỏa điều kiện start > 1987 và sau đó sử dụng .map() duyệt qua từng biến company và mỗi lần như vậy xuất ra 1 thẻ <li> và tên công ty



B3: xuất, import component gọi hàm, truyền prop vào và chạy







* 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

B1: tương tự các quy trình trên

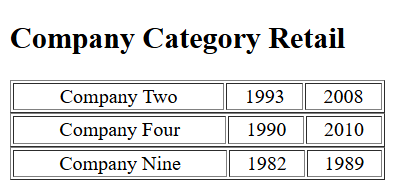


B2: cũng tương tự như bài trên khác ở đây sẽ là destructuring {category} để filter so sánh tuyệt đối có = giá trị ‘Retail’ không sau đó là map, duyệt ra từng company trả về 1 table với các công ty có category ‘Retail’



B3: Xuất và import component, gọi và truyền prop vào





* Sort the companies based on their end date in ascending order

B1: tương tự khác tên CompanySort

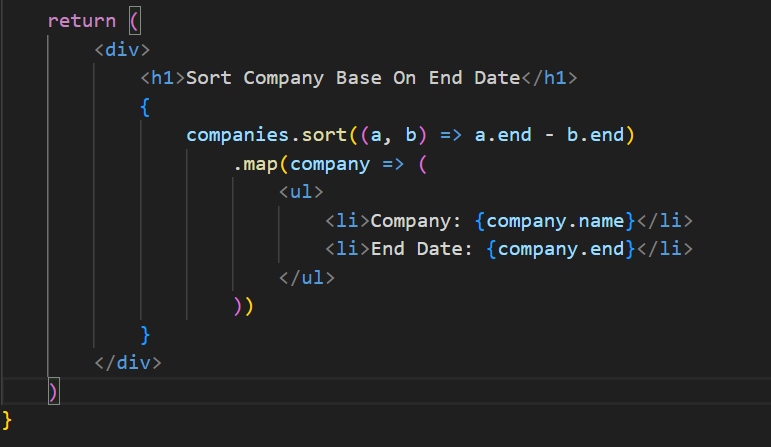


B2: Trả về giao diện JSX sử dụng .sort() để sort company và sẽ sắp xếp theo thuộc tính end  
a.end nhỏ hơn b.end trả về 1 giá trị âm => a đứng trước b

a.end lớn hơn b.end trả về giá trị dương => a đứng sau b

a.end = b.end 2 phần tử không đổi

sau đó dùng map để duyệt qua và trả về company name và company end



B3: xuất và import component gọi và truyền prop vào



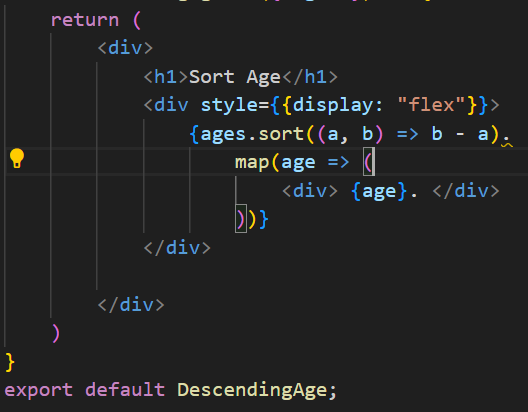


* Sort the ages array in descending order

B1: tương tự tên là DescendingAge

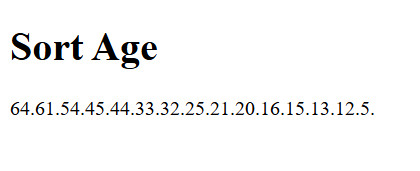


B2: Trả về JSX, sử dụng sort() để sắp xếp tuổi giảm dần, .map() để duyệt qua từng tuổi và trả về giao diện



B3: xuất và import component, gọi và truyền prop ages và chạy



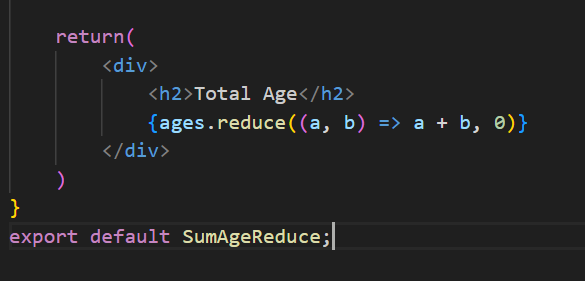


* Print the sum if you add all the ages using reduce

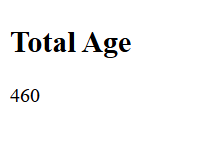
B1: Tương tự khác tên SumAgeReduce



B2: trả về JSX sử dụng .reduce() để rút gọn tính tổng a là tích lũy khởi đầu = 0, b là giá trị qua từng age



B3: xuất và import component, gọi và truyền prop vào và chạy



* 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

B1: tương tự khác tên NewObject

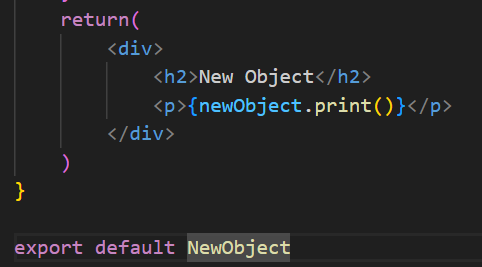


B2: Destructuring lấy trực tiếp phần tử name và category từ companies[0]



B3: Tạo đối tượng newObject có thuộc tích name và category được lấy ở trên, thêm 1 phương thức in ra

B4: trả về giao diện JSX



B5: Xuất và import component, gọi và truyền prop companies và chạy



* 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

B1: tương tự khác tên AddValueToArray và sử dụng cú pháp …rest để thu thập tất cả tham số vào mảng args

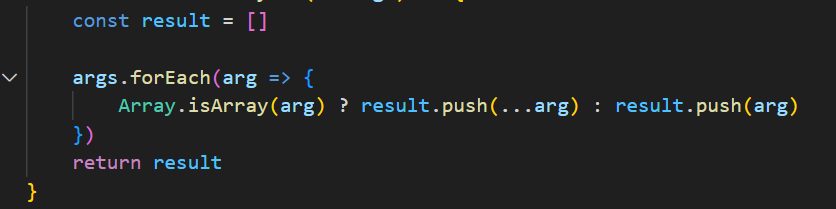


B2: Khởi tạo mảng result, dùng forEach() với args để duyệt qua từng phần tử

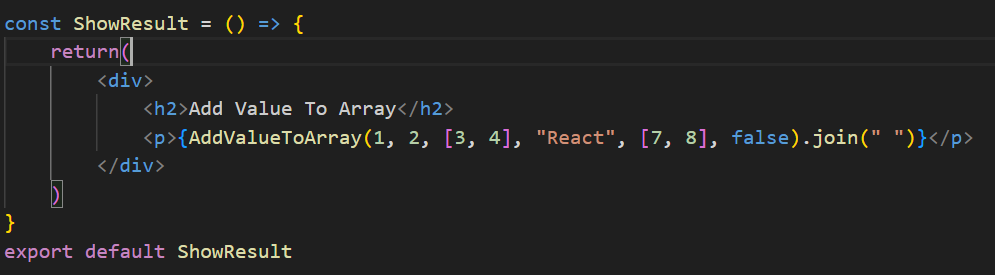
Sau đó dùng toán tử và hàm kiểm tra Array.isArray(arg) để kiểm tra xem có phải array không

Nếu đúng thì sẽ push(…arg) vào result dấu … là toán tử spread dùng để giải nén các phần tử của mảng còn không thì sẽ push thẳng vào push(arg)

Kết thúc forEach và return result

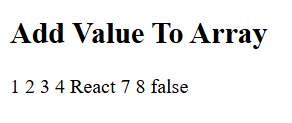


B3: Tạo hàm ShowResult, trả về JSX trong thẻ <p> gọi hàm AddValueToArray, truyền vào args và sử dụng .join(“ “) để mỗi phần tử cách nhau 1 khoản cách



B4: Xuất và import component, gọi và chạy





* Destructuring the property street in a variable named street from the object person

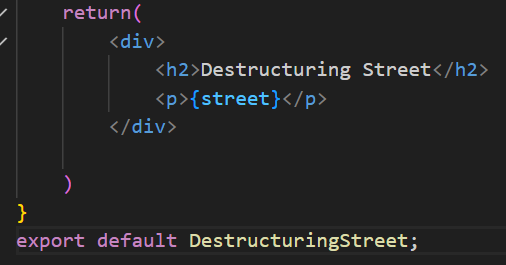
B1: tương tự, khác tên là DestructuringStreet



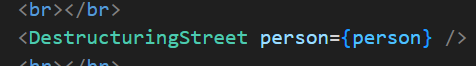
B2: lấy thuộc tính address từ đối tượng person sau đó lấy tiếp thuộc tính street từ address và gán nó vào street

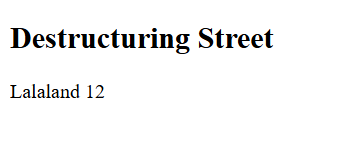


B3: trả về JSX



B4: Xuất và import component, truyền prop là person vào và chạy’



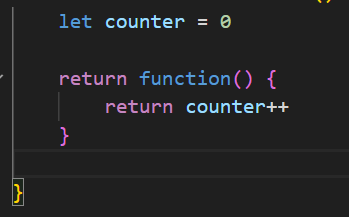


* Write a function that every time you call it, it returns a number that increments starting from 0

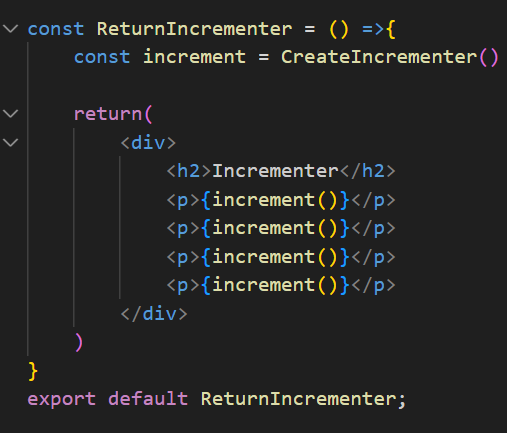
B1: tương tự khác tên CreateIncrementer



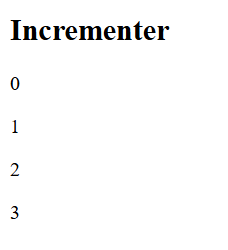
B2: khởi tạo biến counter = 0 sau đó trả về một function ẩn danh function này không có tên nhưng mỗi khi gọi sẽ tăng biến counter lên +1



B3: Khởi tạo hàm trả về JSX để in ra giá trị

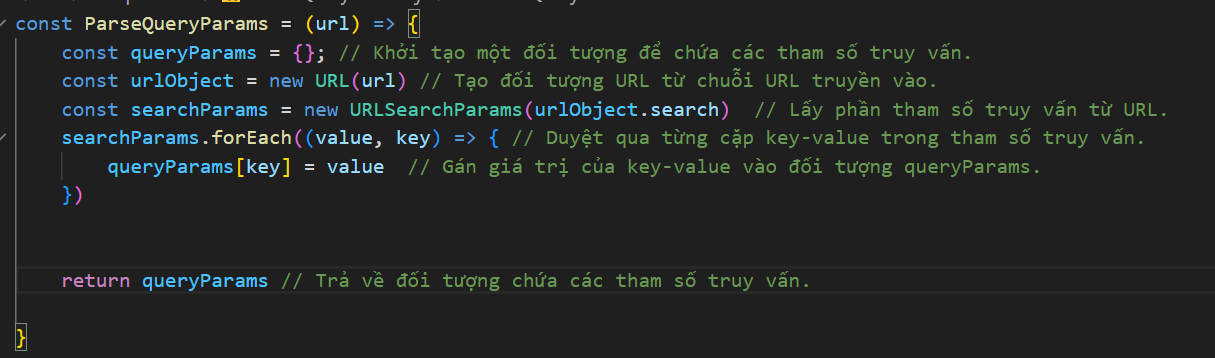


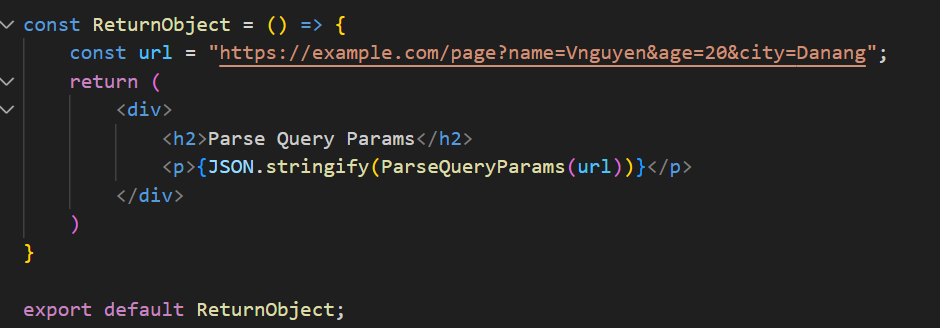
B4: Xuất và import component, gọi và chạy



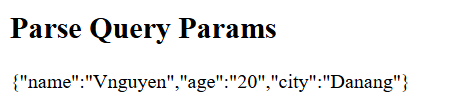
* Create a function that destructors the query parameters of a URL and adds them in an object as key value pairs and then returns the object

B1: tương tự nhưng khác tên ParaseQueryParams

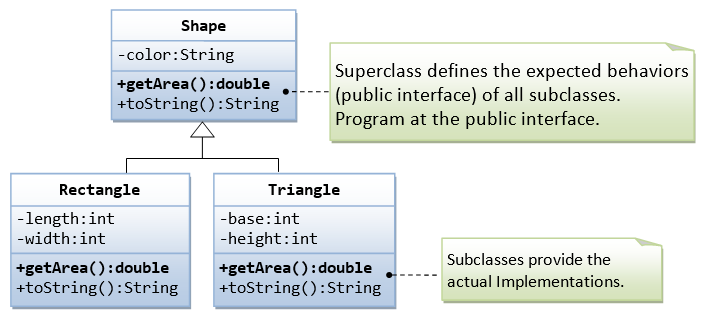
B2: 

B3: Khai báo 1 url để truyền vào ParseQueryParams, trả về JSX, xuất và import gọi và chạy  






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.