**UNIVERSITY OF HELSINKI – FULL STACK COURSE**

**Naming Conventions**

* **Components should always start with a capital letter, to avoid confusion with existing.**

**Part 1 –**

* JSX might look like HTML but is actually more similar to javascript, as it is compiled into javascript by Babel (which is a compiler used in ReactJS).
* As JSX is more similar to JSX all of the attributes in it follow camelCase notation. (example – className attribute).
* JSX returned in a functional component or class component is actually converted to React.createElement().
* Generally, every returned component must contain a root element or must return an array of components if there is no root element.

const App = () => {

return [

<h1>Greetings</h1>,

<Hello name="Maya" age={26 + 10} />,

<Footer />

]

}

* To avoid returning such array of elements, or using extra <div>, we can use empty fragments.

const App = () => {

const name = 'Peter'

const age = 10

return (

<>

<h1>Greetings</h1>

<Hello name="Maya" age={26 + 10} />

<Hello name={name} age={age} />

<Footer />

</>

)

}

* Regarding imports and exports in React (suitable ES6), to import a “export default” object from file we can use any name in the import, but if it isn’t a “export default” then we have to use object de-structuring( that is {some\_name} or import \* as some\_name )all the items in the file are exported as key-value pairs. - <https://stackoverflow.com/a/48441827>.
* We generally use latest standards of JavaScript in reactjs and not all browsers can understand it, hence it has to transpiled to the older versions that is understood by the browser, one of which is Babel.
* Javascript engine is responsible for translating the code to machine-level instructions, whereas the Javascript Runtime Engine provides other additional libraries like DOM API or Web API.
* In javascript – var ,let , const can be used to declare variables, but var is not preferred because of the un-predictability of its behavior based on scope and etc.
* “this” keyword doesn’t exist in arrow functions, where functions defined using “function()” this keyword can be used but behavior has to be carefully monitored.
* States are used in React to keep track of changes and re-render the component in case of any changes.
* Every time a state is changed, it causes the whole component along with it’s child elements to be re-rendered.
* Using object de-structuring and array de-structuring whenever necessary.
* It is mandatory to not update the state directly, therefore before updating the state it is essential to mutate the elements in state and update it and then set it.

**Part 2 –**

* In Functional Programming languages like javascript functions are values, that is they can be assigned to other variables like any other data type like string or integer.
* Functions that are being sent to other functions are called as call-back functions, whereas functions which accept these call-back functions are called as higher-order functions.
* In ES6, for arrow functions if the function only has one line and then we can neglect the curly braces and it is implicitly returned.
* trim() is used to remove line breaks or new lines at the beginning and at the end of the string.
* reduce() is like a super-set for all high-order list functions, as it can perform all types of transformations.
* What are closures – these turn open expressions into closed expressions, that is if a lambda function has variables which are defined somewhere else (in its vicinity) their definition has to be provided to the function, this is what closures do, provide the definitions of undefined variables from surrounding environments.( <https://stackoverflow.com/a/36878651>).
* **Never mutate state directly, but rather create a copy, change it and then assign it.**
* ‘==’ is loose equality as it converts the operands to a common type if different and hence it’s performance varies, ‘===’ is strict equality and doesn’t do such conversions and return false if not of the same type, **hence , in almost all cases strict equality is the best option.**
* array.some() compares elements by reference and hence, is not suitable for array of objects.
* A javascript engine is a literally an interpreter that converts javascript to commands to be executed by a processor, whereas a runtime environment is more than an engine it provides with web APIs, DOM object etc. in case of a browser, which are required by the javascript code in order to run in the browser. ( <http://dolszewski.com/javascript/javascript-runtime-environment/>)