Intro to React
Lab 1
Exercises

1 Online References

1.1 Basic references

https://www.w3schools.com/REACT/DEFAULT.ASP

1.2 Intermediate references

https://ibaslogic.com/react-tutorial-for-beginners

https://www.geeksforgeeks.org/reactjs-components/

https://scrimba.com/learn/learnreact

1.3 Official references

https://react.dev/learn

2 Component basics

1. Create a root component (`App.js`) that returns in its JSX content
   a) HTML headings and paragraphs
   b) HTML images (link to an image on a website, for e.g. Unsplash, Pixelbay, etc)
   c) HTML tables
   d) HTML lists
   e) HTML links

Solution: `components/exercise-solution`
`App-Q1.js`

2.1 Nesting components

2. Create 3 child components with suitable names that return a HTML table, HTML list and HTML link respectively. Embed several instances of each of these child components in the JSX returned by the top level root component.

Solution: `components/exercise-solution`
`App-Q2.js`

2.2 Exporting and importing components

3. Place the definition of the 3 child function components from the previous question (Q2) in a separate JavaScript source code file (`MyChildren.js`). Export one of these components with default export and the other 2 with a named export. Import all these 3 components into your main root component file App.js.

Solution: `components/exercise-solution`
`App-Q3.js`
`MyChildren.js`

2.3 Using JSX in components

2.4 Embedding variables / expressions in JSX

4. Create 3 variables (`destination`, `openStyle`, `textContent`) in the root component function and embed them into the `href` and `target` attributes as well as the text content of an `<a>` element in the JSX of the root component. These variables can have any appropriate values corresponding to the required values for the specified attributes.

See:
https://www.tutorialrepublic.com/html-tutorial/html-links.php

5. Declare an object called `styleObject` in the root component function and give it 3 properties that correspond to the original CSS properties: `text-align`, `text-decoration` and `text-transform`. The properties in `styleObject` can have any appropriate values of your choice. Use `styleObject` to style a `<p>` element in the JSX of the root component

See:
https://www.tutorialrepublic.com/css-tutorial/css-text.php

Solution: `jsx/exercise-solution`
`App-Q4-Q5.js`

3 Props

6. Create a child component ChildLink which extracts 3 properties (`destination`, `openStyle`, `textContent`) from the `props` object that it receives and uses the values of these properties for the `href` and `target` attributes as well as the text content of an `<a>` element in the JSX of the child component. The JSX of the root component should reference this child component and pass these 3 appropriate values for these properties to it.

Solution: `props/exercise-solution`
`App-Q6.js`

3.1 Accessing props via parameter destructuring

7. Rewrite the solution for the previous question (Q6) using parameter destructuring in ChildLink and provide default values for the `destination` and `textContent` properties. The JSX of the root component should then reference this ChildLink twice: the first time with all property values explicitly specified and the second time with only the `openStyle` property

value specified, in order to demonstrate the use of the default values for the 2 other properties.

Solution: `props/exercise-solution App-Q7.js`

3.2 Using children property to access content between JSX tags

8. Rewrite the solution for the previous question (Q7) where the content of the <a> element in the JSX of ChildLink is provided via the `children` property passed via the `props` object from the root parent component, instead of via the `textContent` props.

Solution: `props/exercise-solution App-Q8.js`

3.3 Conditional rendering

9. Create a ChildLink component that receives props `myAge` from the parent root component and returns a <a> element. The `myAge` props should be a number and if it is within the following ranges shown in the table below, then <a> element returned by ChildLink should link to the specified URLs. Try out 3 ChildLink references in the JSX of root component to verify that 3 different <a> elements are returned.

Range of myAge	**Destination of the <a> element**
0 - 10	https://www.malaymail.com/
11 - 20	https://www.nst.com.my/
21 and higher	https://www.utusan.com.my/

Solution: `props/exercise-solution App-Q9.js`

10. Create a ChildImage component that receives props `animal` from the parent root component and returns a <img> element. The `animal` props should be a string with the value `cat`, `dog` or `mouse`, and the <img> element returned from the ImageLink component should link to correspondingly to picture of these animals on a remote website (such as Unsplash). If none of these values are passed, you should not return any JSX at all. Try out 4 different ChildImage references in the JSX of root component to verify that 3 different <img> elements are returned or nothing is returned.

https://dev.to/ocxigin/how-to-link-unsplash-images-in-html-and-css-5dd5

Solution: `props/exercise-solution`
`App-Q10.js`

3.4 Rendering lists

11. Consider the incomplete implementation of App.js below. Complete it so that the child component ListOfLinks returns a list of <li> elements that each encapsulate a ChildLink component, with the properties of the objects in the linkInfo array being passed as props to this component.

```
function ChildLink({destination, actualName}) {

    return (
      <a href={destination}>{actualName}</a>
    );

}


function ListOfLinks() {

  let linkInfo = [
    { destination: "https://www.malaymail.com/",   actualName : 'Malay Mail'   },
    { destination: "https://www.nst.com.my/",   actualName : 'NST'   },
    { destination: "https://www.utusan.com.my/",   actualName : 'Utusan'   }
  ];

  let listChildLinks = [];

  // Complete implementation here
}

export default function App() {

  return (
    <>
      <h2>List of links</h2>
      <ListOfLinks/>
    </>

  );
}
```

Solution: `props/exercise-solution`
`App-Q11.js`

4 Create React project folder structure

5 Styling in React

There are several ways to apply styling to a React app

5.1 Using inline styles

12. Use an inline style approach to style a <div> containing a <p> returned from the root component. You can use any of the relevant CSS properties related to text:

https://www.tutorialrepublic.com/css-tutorial/css-text.php

Solution: `styles/exercise-solution`
`App-Q12.js`

5.2 Importing a separate style sheet

13. Place style sheet rules in app.css and import them into App.js to style one or more <a> elements in App.js. You can use any of the relevant CSS properties related to text:

https://www.tutorialrepublic.com/css-tutorial/css-links.php

Solution: `styles/exercise-solution`
`App-Q13.js`
`App-Q13.css`