Page No.
Date

pollyfill

→ to make older browser understand our new code, The new code is converted into a older code which browser can understand called pollyfill.

→ babel do this conversion automatically

Example :- ES6 is the newer version of Javascript. If I'm working on 1999 browser, my browser will not understand what is const, new promise, etc., so their is a replacement code for this functionalities which is compatible with older version of browser

→ so, This is happen when we write "browser list" → our code is converted to older one.

Babel

→ It is a Javascript package / Library used to convert code written in newer version of JS (ECMA script 2015, 2016, 2017, etc.) into code that can be run in older JS Engines.

To run our app, command is :-

```
npx parcel index.html
```

we always don't have to write this command. Generally, we build a script inside package.json which run this command in an Easy way.

package.json

```
"script" : {
    "start" : "parcel index.html";
    "test" : "Just"
}
```

so, to run the project, we have to write

`npm run start` => shortcut :- `npm start`

"start" script Execute this command

Build command

```
npx parcel build index.html
```

```
"script" : {
    build : "parcel build index.html";
}
```

Note

`npx = npm run`

so, `npm run build` will build a project.

Page No.
Date

console.log

→ console.log are not removed automatically by the parcel. we have to configure our project to remove it

→ There is a package which helps to remove console.log :-

↳ babel-plugin-transform-remove-console

→ Before installing this package, create a folder called .babelrc } For configuration

→ And include :-

```
{
  "plugins": [["transform-remove-console",
  {"exclude": ["error", "warn"]}]]
}
```

→ Then, build : npm run build

& see that all console.log are removed.

Page No.
Date

Render

→ means updating something in the DOM

Reconciliation

→ help to make React application fast and efficient by minimizing the amount of work that need to be done to update the changes

→ so, you don't have to worry about what changes on Every update.

Ex:-

```
<ul>
  <li> first </li>
  <li> second </li>
</ul>
```

```
<ul>
  <li> first </li>
  <li> second </li>
  <li> third </li>
</ul>
```

For adding new child in above case react has to re-render Everything, it will not give you a good performance.

SOLUTION - INTRODUCTION OF KEYS

```
<ul>
  <li key="1"> first </li>
  <li key="2"> second </li>
  <li key="3"> third </li>
</ul>
```

siblings

→ For adding siblings always use a key attributes for uniquely identify in REACT.

React.createElement ⟹ { } ⟹ HTML (DOM)

Page No.
Date

React.createElement()

→ React.createElement() is creating an object.

→ This object is converted into HTML code and puts upon DOM.

If you want to build a big HTML structure, then using 'createElement()' is not a good solution,
So there comes JSX

JSX

when Facebook created React, the major concept behind bringing react was that we want to write a lots of HTML using Javascript. because Javascript is very performant.

```
const heading = React.createElement(
"h1",
{ id = "title"
  key = "1"
},
"hello"
)
```

```
const heading = <h1>Hello</h1>
```

→ This is JSX

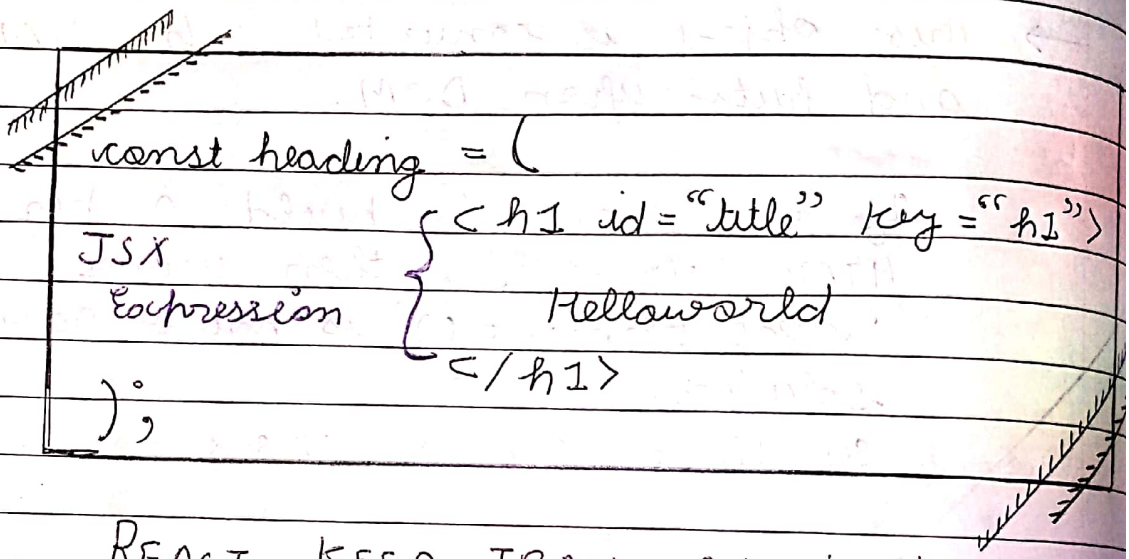
Babel comes along with parcel

Page No.
Date

* JSX is not "HTML inside Javascript"

⟹ JSX has "HTML-like syntax"

This is valid JS code :-



REACT KEEP TRACK OF KEY

→ JSX uses React.createElement() behind the scenes.

⟹ JSX ⟹ React.createElement() ⟹ object ⟹ HTML (DOM)

→ Babel converts JSX to React.createElement

Advantages of JSX

→ Developer Experience
→ syntactical sugar
→ Readability
→ less code
→ maintainability
→ No Repetition

COMPONENT

"Everything is a component in React"

There are two types of React component are :-

1. Functional component — NEW WAY of writing code
2. class Based component — OLD WAY

→ Functional component

☆ it is nothing but a Javascript Function

☆ It is a normal JS function which returns some piece of react element (here JSX)

Example :-

```
const HeaderComponent = ( ) => {
    return <h1> Helloworld </h1>
};
```

⊙ For any component Name start with capital letter.

⊙ TO render Functional component, write :-

```
<HeaderComponent/>
```

Page No.
Date

React Element

```
const heading = (
    <h1 id="title
        key="h1">
    Hello world
    </h1>
);
```

converting it into arrow function will make functional component

React component

```
const heading = () =>
    return (
    <h1 id="title"
        key="h1">
    Hello world
    </h1>
);
};
```

React Element is finally an object

Functional component is finally a Function

---

* The Amazing thing

① * const Title = () => {
    <h1> Hello world </h1>
}

} it is a functional component

* const HeaderComponent = () => {
    return (

        <Title/>
        <h2> React </h2>
        <h2> Hellow </h2>

    );
};

This is component composition

instead of this we can write {Title()}

Page No.
Date

(2)

* const title = (
    <h1> Hello </h1>
);

} this is normal variable

* const HeaderComponent = () => {
    return (

            {title}
            <h2> Hello </h2>
            <h2> World </h2>

    );
};

(3)

Whenever you write JSX, you can write any piece of Javascript code b/w parenthesis { } it will work.

Component composition

↳ If I have to use a component inside a component Then it is called component composition / composing components