

30 Days Of React: Props



LinkedIn



Follow @asabeneh

776

Author: [Asabeneh Yetayeh](#)

October, 2020

[<< Day 4](#) | [Day 6 >>](#)



- [Props](#)
 - [Props in Functional Component](#)
 - [What is props?](#)
 - [Props object](#)
 - [Different data type props](#)
 - [String props type](#)
 - [Number props type](#)
 - [Boolean props type](#)
 - [Array props type](#)
 - [Object props type](#)
 - [Function prop types](#)
 - [Destructuring props](#)
 - [propTypes](#)
 - [defaultProps](#)
- [Exercises: Components and Props](#)
 - [Exercises: Level 1](#)
 - [Exercises: Level 2](#)
 - [Exercises: Level 3](#)

Props

Props in Functional Component

In the previous day, we saw how to inject different data types to React component JSX. Now, let us see how we use it in component and also how to pass different data as props.

What is props?

Props is a special keyword in React that stands for properties and is being used to pass data from one component to another and mostly from parent component to child component. We can say props is a data carrier or a means to transport data.

I hope you are familiar with the JavaScript function. Most of the time, functions with parameters are smart and they can take dynamic data likewise props is a way we pass data or parameter to a component. Let's see the difference between a function and a component.

```
// function syntax

const getUserInfo = (firstName, lastName, country) => {
  return `${firstName} ${lastName}. Lives in ${country}.`
}

// calling a functions

getUserInfo('Asabeneh', 'Yeteyeh', 'Finland')

//component syntax

// User component, component should start with an uppercase
const User = (props) => {
  return (
    <div>
      <h1>
        {props.firstName}
        {props.lastName}
      </h1>
      <small>{props.country}</small>
    </div>
  )
}

// calling or instantiating a component, this component has three properties and
we call them props:firstName, lastName, country
<User firstName = 'Asabeneh', lastName='Yetayeh' country = 'Finland' />
```

In the previous section, we injected data as follows and today we will change these data to props.

```
const welcome = 'Welcome to 30 Days Of React'
const title = 'Getting Started React'
const subtitle = 'JavaScript Library'
```

```

const author = {
  firstName: 'Asabeneh',
  lastName: 'Yetayeh',
}
const date = 'Oct 4, 2020'

// Header Component
const Header = () => (
  <header>
    <div className='header-wrapper'>
      <h1>{welcome}</h1>
      <h2>{title}</h2>
      <h3>{subtitle}</h3>
      <p>
        {author.firstName} {author.lastName}
      </p>
      <small>{date}</small>
    </div>
  </header>
)

```

Instead of injecting data we can also pass the data as props. React props are similar to parameters in functions.

Props object

React props is an object which you get instantly when you create a React component. Before we pass properties to the component, let's check what do we get in the props object.

```

import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
const Header = (props) => {
  console.log(props) // empty object, {}
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{welcome}</h1>
        <h2>{title}</h2>
        <h3>{subtitle}</h3>
        <p>
          {author.firstName} {author.lastName}
        </p>
        <small>{date}</small>
      </div>
    </header>
  )
}

// The App, or the parent or the container component

```

```
// Functional Component
const App = () => {
  return (
    <div className='app'>
      <Header />
    </div>
  )
}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)
```

In the above `console.log(props)`, you would get an empty object `{}`. That means if you do not pass any attributes or properties when you instantiate the component, the props will be empty otherwise it will be populated with the data you passed as attributes and the proper name of these attributes are props.

Let's start with a simple example. In the example below, the welcome string has been passed as props in the Header components.

```
import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
const Header = (props) => {
  console.log(props) // {welcome:'Welcome to 30 Days Of React'}
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{props.welcome}</h1>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
  return (
    <div className='app'>
      <Header welcome='Welcome to 30 Days Of React' />
    </div>
  )
}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)
```

Now, when you do `console.log(props)` you should get the following object, that means the `welcome` property we passed to the `Header` component can be found inside the `props` object.

```
{
  welcome: 'Welcome to 30 Days Of React'
}
```

As you can see in the above code, we passed only single props to `Header` component, the `welcome` props. A component can have one or many props. Props could be different data types. It could be a string, number, boolean, array, object or a function. We will cover different kind of props in the next sections.

Different data type props

String props type

The data type of the props we pass an attribute to the component is a string.

```
import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
const Header = (props) => {
  console.log(props)
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{props.welcome}</h1>
        <h2>{props.title}</h2>
        <h3>{props.subtitle}</h3>
        <p>
          {props.firstName} {props.lastName}
        </p>
        <small>{props.date}</small>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => (
  <div className='app'>
    <Header
      welcome='Welcome to 30 Days Of React'
      title='Getting Started React'
      subtitle='JavaScript Library'
      firstName='Asabeneh'
      lastName='Yetayeh'
      date='Oct 4, 2020'
    />
  </div>
)
```

```

    </div>
  )

  const rootElement = document.getElementById('root')
  ReactDOM.render(<App />, rootElement)

```

If you check on the browser console, you will get the following object.

```

{
  firstName: "Asabeneh",
  lastName: "Yetayeh",
  date: "Oct 4, 2020"
  subtitle: "JavaScript Library"
  title: "Getting Started React"
  welcome: "Welcome to 30 Days Of React"
}

```

Since you are a JavaScript ninja by now, you know what do do with this object.

As you can see from the above example, the value of the props are written statically. However, if we want to apply some logic it is hard to implement with statically written data, so it will be better to use a variable as props. Let's see the following example:

```

import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
const Header = (props) => (
  <header>
    <div className='header-wrapper'>
      <h1>{props.welcome}</h1>
      <h2>{props.title}</h2>
      <h3>{props.subtitle}</h3>
      <p>
        {props.firstName} {props.lastName}
      </p>
      <small>{props.date}</small>
    </div>
  </header>
)

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const welcome = 'Welcome to 30 Days Of React'
  const title = 'Getting Started React'
  const subtitle = 'JavaScript Library'
  const firstName = 'Asabeneh'
  const lastName = 'Yetayeh'

```

```

const date = 'Oct 4, 2020'

return (
  <div className='app'>
    <Header
      welcome={welcome}
      title={title}
      subtitle={subtitle}
      firstName={firstName}
      lastName={lastName}
      date={date}
    />
  </div>
)
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

Number props type

Let's use a number props to a component

```

import React from 'react'
import ReactDOM from 'react-dom'

const Age = (props) => <div>The person is {props.age} years old.</div>
const Weight = (props) => (
  <p>The weight of the object on earth is {props.weight} N.</p>
)

// The App, or the parent or the container component
// Functional Component
const App = () => {
  let currentYear = 2020
  let birthYear = 1820
  const age = currentYear - birthYear
  const gravity = 9.81
  const mass = 75

  return (
    <div className='app'>
      <Age age={age} />
      <Weight weight={gravity * mass} />
    </div>
  )
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

Boolean props type

We can pass boolean data types to a React component.

```
import React from 'react'
import ReactDOM from 'react-dom'

const Status = (props) => {
  // ternary operator to check the status of the person
  let status = props.status ? 'Old enough to drive' : 'Too young for driving'
  return <p>{status}</p>
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
  let currentYear = 2020
  let birthYear = 2015
  const age = currentYear - birthYear // 15 years

  let status = age >= 18

  return (
    <div className='app'>
      <Status status={status} />
    </div>
  )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```

Array props type

In programming arrays and objects are the most frequently used data structure to solve different problems and store data in a more structured way. Therefore, we encounter data in the form of an array quite often. Let's pass an array as props to a component

```
import React from 'react'
import ReactDOM from 'react-dom'

const Skills = (props) => <ul>{props.skills}</ul>

const App = () => (
  <div className='app'>
    <Skills skills={['HTML', 'CSS', 'JavaScript']} />
  </div>
)

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```


If you see the result on the browser, the skills elements needs formatting. Therefore before we render, it should have some elements between each skill. To modify the array and to add a li element we can use map method. You should be very familiar with the functional programming map, filter and reduce to feel good at React if not please go back to day 1 JavaScript refresher. Let's apply map to modify the array.

```
import React from 'react'
import ReactDOM from 'react-dom'

// Skills Component
const Skills = (props) => {
  // modifying the skills array
  const skillList = props.skills.map((skill) => <li>{skill}</li>)
  return <ul>{skillList}</ul>
}

const App = () => (
  <div className='app'>
    <Skills skills={['HTML', 'CSS', 'JavaScript']} />
  </div>
)

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```

We will go in-depth about list and map in other sections. Now, let's see an object as a props.

Object props type

We may pass an object as props to a React component. Let's see an example. We can change the previous Header props to object. For the time being let's change a few properties for better understanding.

```
import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
const Header = (props) => {
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{props.data.welcome}</h1>
        <h2>{props.data.title}</h2>
        <h3>{props.data.subtitle}</h3>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
```

```

const data = {
  welcome: 'Welcome to 30 Days Of React',
  title: 'Getting Started React',
  subtitle: 'JavaScript Library',
}

return (
  <div className='app'>
    <Header data={data} />
  </div>
)
}
const rootElement = document.getElementById('root')
// we render the JSX element using the ReactDOM package
ReactDOM.render(<App />, rootElement)

```

Now, let's change all the previous Header properties to an object.

```

import React from 'react'
import ReactDOM from 'react-dom'

const showDate = (time) => {
  const months = [
    'January',
    'February',
    'March',
    'April',
    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}

// Header Component
const Header = (props) => {
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{props.data.welcome}</h1>
        <h2>{props.data.title}</h2>
        <h3>{props.data.subtitle}</h3>
        <p>
          {props.data.author.firstName} {props.data.author.lastName}

```

```

        </p>
        <small>{showDate(props.data.date)}</small>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const data = {
    welcome: 'Welcome to 30 Days Of React',
    title: 'Getting Started React',
    subtitle: 'JavaScript Library',
    author: {
      firstName: 'Asabeneh',
      lastName: 'Yetayeh',
    },
    date: new Date(), // date needs to be formatted to a human readable format
  }

  return (
    <div className='app'>
      <Header data={data} />
    </div>
  )
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

When we use an object as props we usually destructure the data to access the values. Destructuring makes our code easy to read. We will soon see the destructuring of props but before that let's see function as props for a React component.

Function prop types

We can pass a function as props type to a React component. Let's see some examples

```

import React from 'react'
import ReactDOM from 'react-dom'

// A button component

const Button = (props) => <button onClick={props.onClick}>{props.text}</button>

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const sayHi = () => {
    alert('Hi')
  }
}

```

```

    return (
      <div className='app'>
        <Button text='Say Hi' onClick={sayHi} />
      </div>
    )
  }
  const rootElement = document.getElementById('root')
  // we render the JSX element using the ReactDOM package
  ReactDOM.render(<App />, rootElement)

```

Even we can write a function inside the curly bracket

```

import React from 'react'
import ReactDOM from 'react-dom'

// A button component

const Button = (props) => <button onClick={props.onClick}>{props.text}</button>

// The App, or the parent or the container component
// Functional Component
const App = () => {
  return (
    <div className='app'>
      <Button text='Say Hi' onClick={() => alert('Hi')} />
    </div>
  )
}
const rootElement = document.getElementById('root')
// we render the JSX element using the ReactDOM package
ReactDOM.render(<App />, rootElement)

```

Now, lets implement different functions as props

```

import React from 'react'
import ReactDOM from 'react-dom'

// A button component

const Button = (props) => <button onClick={props.onClick}>{props.text}</button>

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const greetPeople = () => {
    alert('Welcome to 30 Days Of React Challenge, 2020')
  }
}

```

```

    return (
      <div className='app'>
        <Button text='Greet People' onClick={greetPeople} />
        <Button text='Show Time' onClick={() => alert(new Date())} />
      </div>
    )
  }
  const rootElement = document.getElementById('root')
  ReactDOM.render(<App />, rootElement)

```

In the above example, `onClick` is a props to hold the `greetPeople` function. HTML has `onclick`, `onmouseover`, `onhover`, `onkeypress` and etc event handlers. In React, these handlers are in camelCase. For instance `onClick`, `onMouseOver`, `onKeyPress` etc. We will cover events in React in detail in other section.

Let's see some more functions as props to give a clear understanding how to handle function as props in a React component.

This component shows month, date and year as an alert box.

```

import React from 'react'
import ReactDOM from 'react-dom'

// Function to display time in Mon date, year format eg Oct 4, 2020
const showDate = (time) => {
  const months = [
    'January',
    'February',
    'March',
    'April',
    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}

// A button component

const Button = (props) => <button onClick={props.onClick}>{props.text}</button>

// The App, or the parent or the container component
// Functional Component

```

```

const App = () => {
  const handleTime = () => {
    alert(showDate(new Date()))
  }
  const greetPeople = () => {
    alert('Welcome to 30 Days Of React Challenge, 2020')
  }
  return (
    <div className='app'>
      <Button text='show time' onClick={handleTime} />
      <Button text='Greet People' onClick={greetPeople} />
    </div>
  )
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

Destructuring props

By now, I believe you are a JavaScript ninja and you know about destructuring arrays and objects. Destructuring code to some extent makes easy to read. Let us destructure the props in Header component. Everything we passed as props is stored in props object. Therefore, props is an object and we can destructure the properties. Let's destructure some of the props we wrote in object props example. We can destructure in many ways:

1. Step by step destructuring

```

import React from 'react'
import ReactDOM from 'react-dom'

const showDate = (time) => {
  const months = [
    'January',
    'February',
    'March',
    'April',
    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}
// Header Component

```

```

const Header = (props) => {
  const data = props.data
  const { welcome, title, subtitle, author, date } = data
  const { firstName, lastName } = author
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{welcome}</h1>
        <h2>{title}</h2>
        <h3>{subtitle}</h3>
        <p>
          {firstName} {lastName}
        </p>
        <small>{showDate(date)}</small>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const data = {
    welcome: 'Welcome to 30 Days Of React',
    title: 'Getting Started React',
    subtitle: 'JavaScript Library',
    author: {
      firstName: 'Asabeneh',
      lastName: 'Yetayeh',
    },
    date: new Date(),
  }

  return (
    <div className='app'>
      <Header data={data} />
    </div>
  )
}

const rootElement = document.getElementById('root')
// we render the JSX element using the ReactDOM package
ReactDOM.render(<App />, rootElement)

```

2. Destructuring in one line

```

import React from 'react'
import ReactDOM from 'react-dom'

const showDate = (time) => {
  const months = [
    'January',
    'February',

```

```

    'March',
    'April',
    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}

// Header Component
const Header = (props) => {
  const data = props.data
  const {
    welcome,
    title,
    subtitle,
    author: { firstName, lastName },
    date,
  } = data

  return (
    <header>
      <div className='header-wrapper'>
        <h1>{welcome}</h1>
        <h2>{title}</h2>
        <h3>{subtitle}</h3>
        <p>
          {firstName} {lastName}
        </p>
        <small>{showDate(date)}</small>
      </div>
    </header>
  )
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const data = {
    welcome: 'Welcome to 30 Days Of React',
    title: 'Getting Started React',
    subtitle: 'JavaScript Library',
    author: {
      firstName: 'Asabeneh',
      lastName: 'Yetayeh',
    },
  },

```



```

    date: new Date(),
  }

  return (
    <div className='app'>
      <Header data={data} />
    </div>
  )
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

3. Destructuring the props inside the parenthesis

```

import React from 'react'
import ReactDOM from 'react-dom'

const showDate = (time) => {
  const months = [
    'January',
    'February',
    'March',
    'April',
    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}

// Header Component
const Header = ({
  data: {
    welcome,
    title,
    subtitle,
    author: { firstName, lastName },
    date,
  },
}) => {
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{welcome}</h1>

```

```

        <h2>{title}</h2>
        <h3>{subtitle}</h3>
        <p>
            {firstName} {lastName}
        </p>
        <small>{showDate(date)}</small>
    </div>
</header>
)
}

// The App, or the parent or the container component
// Functional Component
const App = () => {
    const data = {
        welcome: 'Welcome to 30 Days Of React',
        title: 'Getting Started React',
        subtitle: 'JavaScript Library',
        author: {
            firstName: 'Asabeneh',
            lastName: 'Yetayeh',
        },
        date: new Date(),
    }

    return (
        <div className='app'>
            <Header data={data} />
        </div>
    )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

Now, let's destructure all the components we had and assemble them together. We pass props from one component to another typically from parent to a child component. For instance in the Main component techs, user, greetPeople and handleTime props have been passed from the parent component Main to child components TechList and UserCard. Below, you will get all the codes destructured and cleaned.

```

import React from 'react'
import ReactDOM from 'react-dom'
import asabenehImage from './images/asabeneh.jpg'

// Fuction to show month date year

const showDate = (time) => {
    const months = [
        'January',
        'February',
        'March',
        'April',
    ]
}

```

```

    'May',
    'June',
    'July',
    'August',
    'September',
    'October',
    'November',
    'December',
  ]

  const month = months[time.getMonth()].slice(0, 3)
  const year = time.getFullYear()
  const date = time.getDate()
  return ` ${month} ${date}, ${year}`
}

// Header Component
const Header = ({
  data: {
    welcome,
    title,
    subtitle,
    author: { firstName, lastName },
    date,
  },
}) => {
  return (
    <header>
      <div className='header-wrapper'>
        <h1>{welcome}</h1>
        <h2>{title}</h2>
        <h3>{subtitle}</h3>
        <p>
          {firstName} {lastName}
        </p>
        <small>{showDate(date)}</small>
      </div>
    </header>
  )
}

// TechList Component
const TechList = ({ techs }) => {
  const techList = techs.map((tech) => <li key={tech}>{tech}</li>)
  return techList
}

// User Card Component
const UserCard = ({ user: { firstName, lastName, image } }) => (
  <div className='user-card'>
    <img src={image} alt={firstName} />
    <h2>
      {firstName}
      {lastName}
    </h2>
  </div>
)

```

```

    </h2>
  </div>
)

// A button component

const Button = ({ text, onClick, style }) => (
  <button style={style} onClick={onClick}>
    {text}
  </button>
)

// CSS styles in JavaScript Object
const buttonStyles = {
  backgroundColor: '#61dbfb',
  padding: 10,
  border: 'none',
  borderRadius: 5,
  margin: 3,
  cursor: 'pointer',
  fontSize: 18,
  color: 'white',
}

// Main Component
const Main = ({ user, techs, greetPeople, handleTime }) => (
  <main>
    <div className='main-wrapper'>
      <p>Prerequisite to get started react.js:</p>
      <ul>
        <TechList techs={techs} />
      </ul>
      <UserCard user={user} />
      <Button text='Greet People' onClick={greetPeople} style={buttonStyles} />
      <Button text='Show Time' onClick={handleTime} style={buttonStyles} />
    </div>
  </main>
)

// Footer Component
const Footer = ({ copyright }) => (
  <footer>
    <div className='footer-wrapper'>
      <p>Copyright {copyright.getFullYear()}</p>
    </div>
  </footer>
)

// The App, or the parent or the container component
// Functional Component
const App = () => {
  const data = {
    welcome: 'Welcome to 30 Days Of React',
    title: 'Getting Started React',
  }

```

```

    subtitle: 'JavaScript Library',
    author: {
      firstName: 'Asabeneh',
      lastName: 'Yetayeh',
    },
    date: new Date(), // date needs to be formatted to a human readable format
  }
  const date = new Date()
  const techs = ['HTML', 'CSS', 'JavaScript']
  // copying the author from data object to user variable using spread operator
  const user = { ...data.author, image: asabenehImage }

  const handleTime = () => {
    alert(showDate(new Date()))
  }
  const greetPeople = () => {
    alert('Welcome to 30 Days Of React Challenge, 2020')
  }

  return (
    <div className='app'>
      <Header data={data} />
      <Main
        user={user}
        techs={techs}
        handleTime={handleTime}
        greetPeople={greetPeople}
      />
      <Footer copyright={date} />
    </div>
  )
}
const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)

```

propTypes

The propTypes package helps us to assign the data types of the props we passed to a component.

defaultProps

The defaultProps can be used when we want to have some default prop types for a component.

We will cover propTypes in detail in other sections.

Exercises: Components and Props

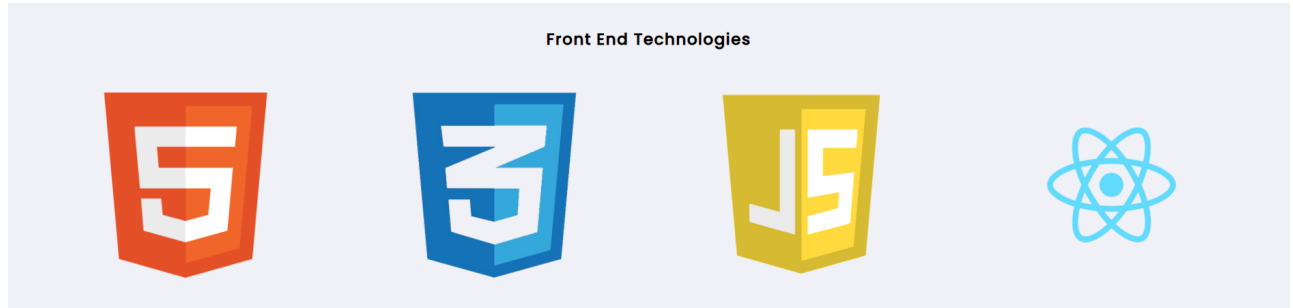
Exercises: Level 1

1. What is props in a React component ?
2. How do you access props in a React component ?

3. What data types can we pass as props to components ?
4. What is a propTypes?
5. What is a default propTypes?

Exercises: Level 2

1. Create a functional component and display the following images

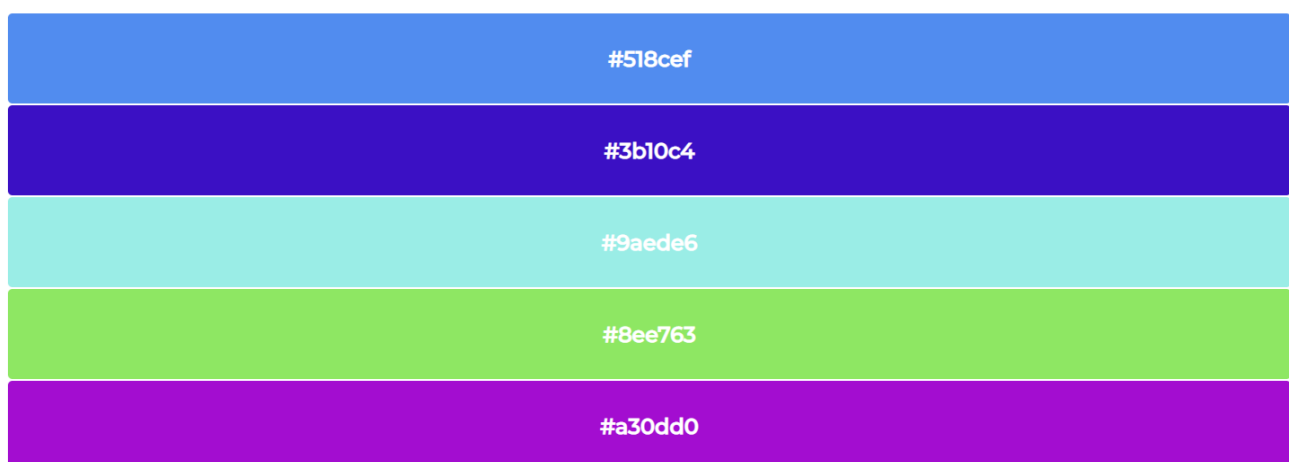


2. Use functional component to create the following design


A "SUBSCRIBE" form with a light blue background. It includes the text "Sign up with your email address to receive news and updates." Below this are three input fields labeled "First name", "Last name", and "Email". At the bottom is a red "Subscribe" button.

Exercises: Level 3

1. Use the given hexadecimal color generator in the example to create these random colors. If you don't know how to generate the hexadecimal color you can use [dummy data generator](#)



2. Use functional component to design the following user card.



ASABENEH YETAYEH ✓

Senior Developer, Finland

SKILLS

HTML

CSS

Sass

JS

React

Redux

Node

MongoDB

Python

Flask

Django

NumPy

Pandas

Data Analysis

MYSQL

GraphQL

D3.js

Gatsby

Docker

Heroku

Git

🕒 Joined on Aug 30, 2020

🎉 CONGRATULATIONS ! 🎉

[<< Day 4](#) | [Day 6 >>](#)