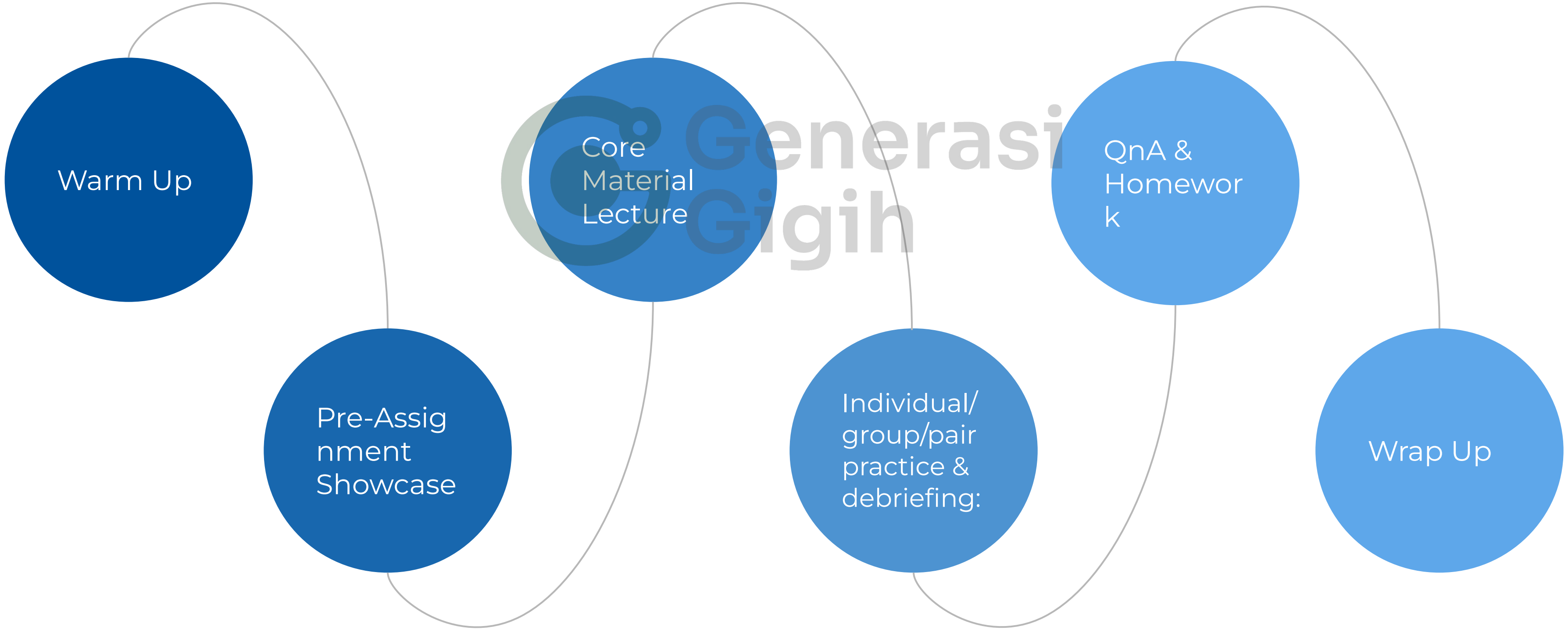
Full Stack Engineer

Module 4 Session 2: Component Part 2, Props, and Conditional Rendering

Our Agenda


Warm Up
Pre-Assig nment Showcase
Core Material Lecture
Individual/ group/pair practice & debriefing:
QnA & Homewor k
Wrap Up

Generasi Gigih

Let's Warm Up!

What is JSX and Component? How to write the code with component and JSX?

Generasi Gigih

Showcase Time!

Let's Discuss

Generasi Gigih

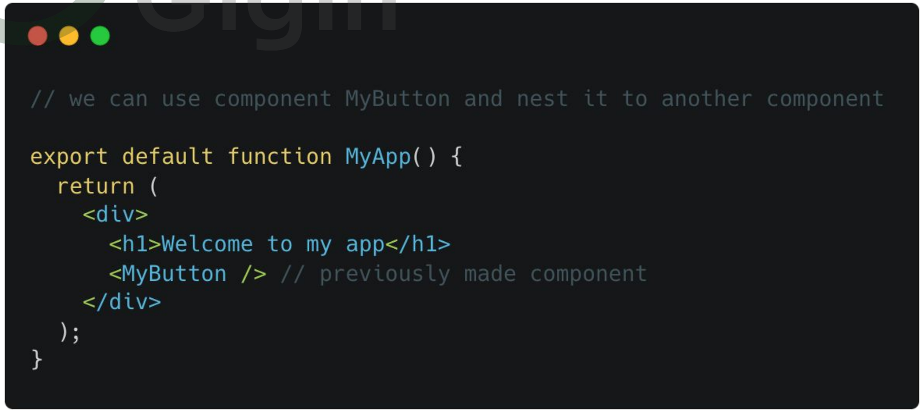
Props

Communicating between components

Intermezzo

Remember parent component and child component? (previously mentioned when learning how to nest component)

Please explain



```
// we can use component MyButton and nest it to another component

export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton /> // previously made component
    </div>
  );
}
```

Part 1: Intro to Props

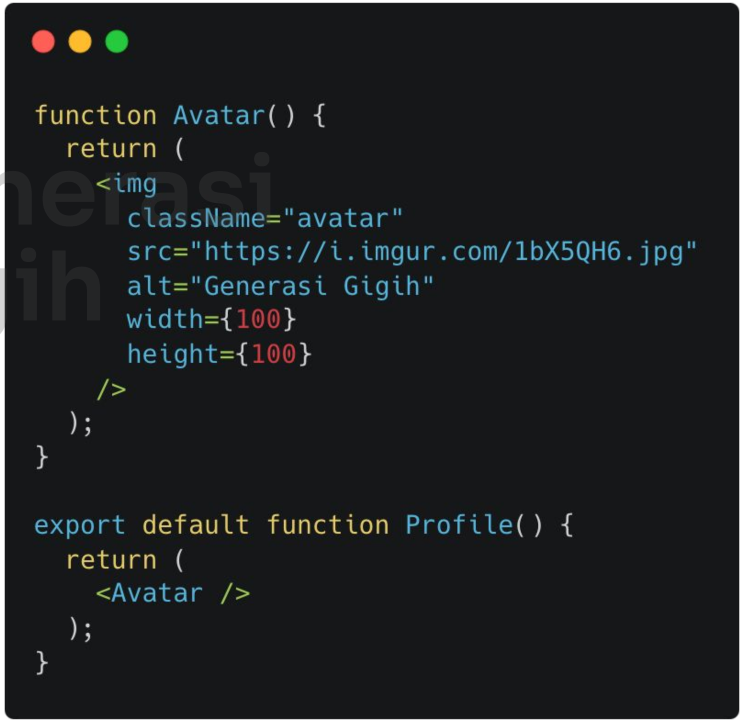
- React components use props to communicate with each other.
- Every parent component can **pass some information to its child** components by giving them props
- Props are **immutable**, meaning its values cannot be changed
- Props can be any Javascript values (objects, array, functions, etc)

Part 2: Parent and Child Components (no props)

read first

- Profile – parent component
- Avatar – child component of Profile component

- No props on Avatar component (only calling <Avatar and then close />)



```
function Avatar() {
  return (
    <img
      className="avatar"
      src="https://i.imgur.com/1bX5QH6.jpg"
      alt="Generasi Gigih"
      width={100}
      height={100}
    />
  );
}

export default function Profile() {
  return (
    <Avatar />
  );
}
```

Part 3: Parent and Child Components (with props)

Steps to use props:

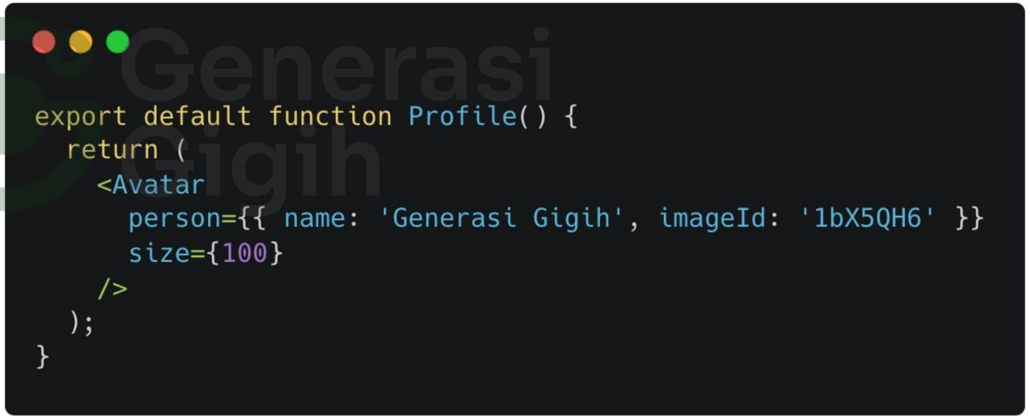
1. Pass props to child component
2. Read/ retrieve props inside the child component

We'll see one by one..

Part 3.1: Pass props to child component

Profile is passing 2 props to **Avatar**:

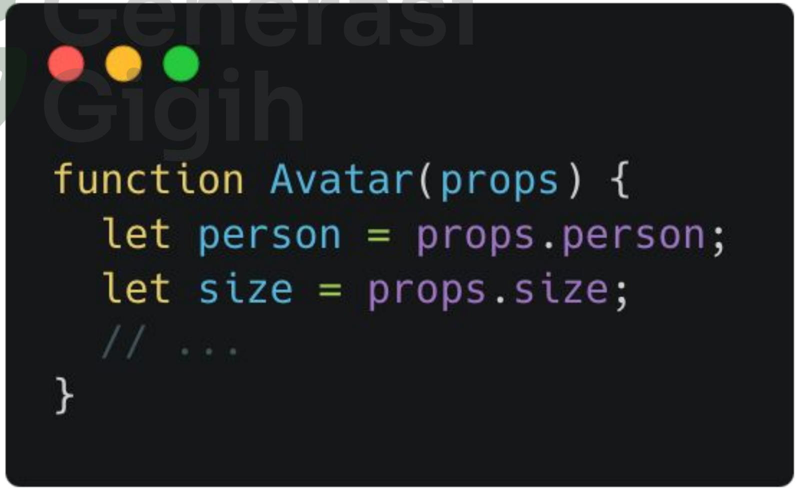
1. Person (object)
2. Size (number)



```
export default function Profile() {
  return (
    <Avatar
      person={{ name: 'Generasi Gigih', imageId: '1bX5QH6' }}
      size={100}
    />
  );
}
```

Part 3.2: Read props inside child component

Person and size can now be read inside Avatar component

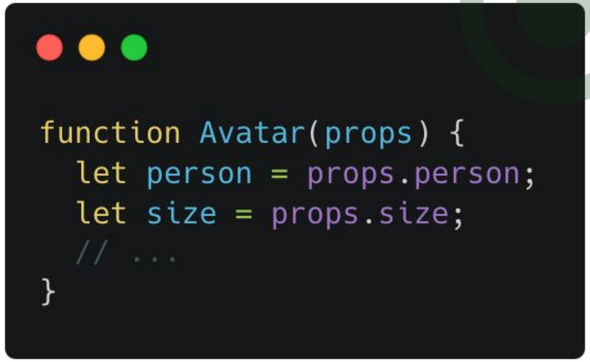


```
function Avatar(props) {
  let person = props.person;
  let size = props.size;
  // ...
}
```

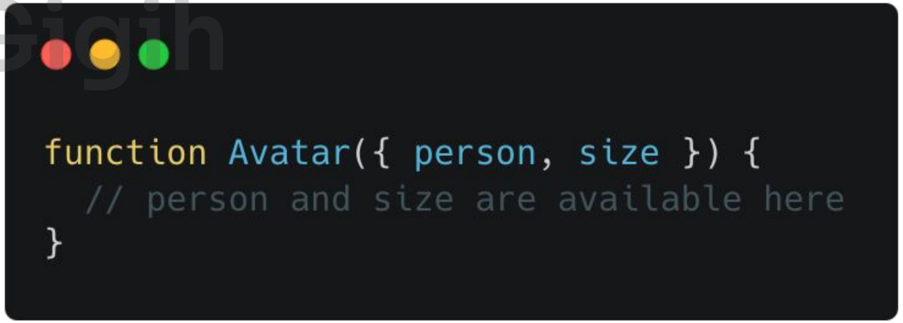
Intermezzo

Remember syntax to destructure an object? (lesson on module 1)

How if we want to destructure props right away?



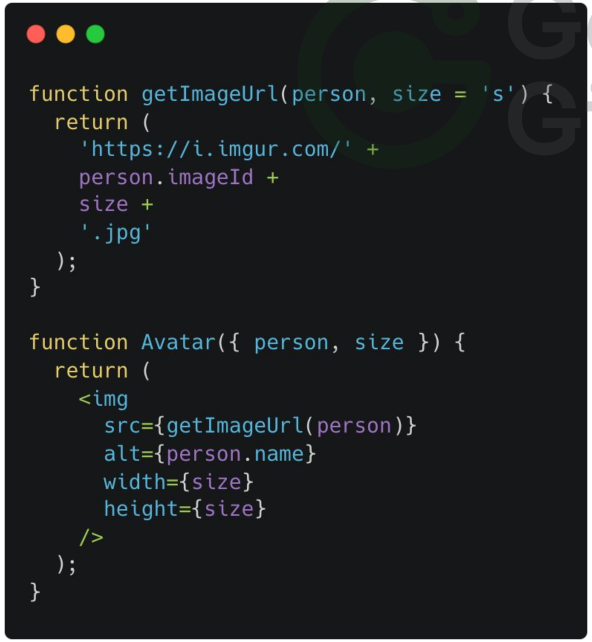
```
function Avatar(props) {
  let person = props.person;
  let size = props.size;
  // ...
}
```



```
function Avatar({ person, size }) {
  // person and size are available here
}
```

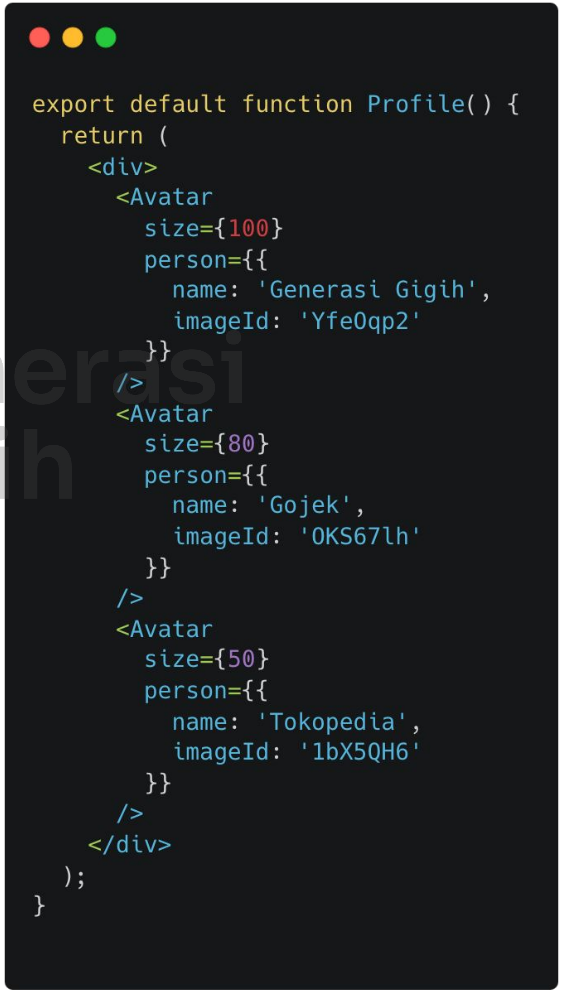
If you forget please read again:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment#Unpacking_fields_from_objects_passed_as_a_function_parameter

Part 3.3: Display multiple child component with different props (read first)



```
function getImageUrl(person, size = 's') {
  return (
    'https://i.imgur.com/' +
    person.imageId +
    size +
    '.jpg'
  );
}

function Avatar({ person, size }) {
  return (
    <img
      src={getImageUrl(person)}
      alt={person.name}
      width={size}
      height={size}
    />
  );
}
```

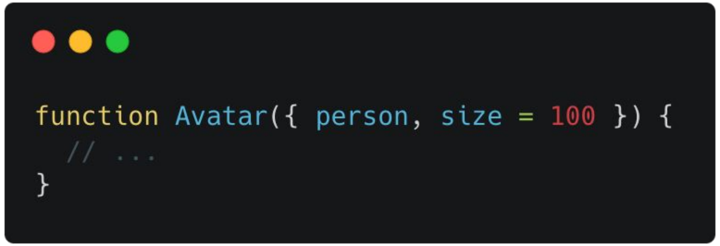


```
export default function Profile() {
  return (
    <div>
      <Avatar
        size={100}
        person={{
          name: 'Generasi Gigih',
          imageId: 'YfeOqp2'
        }}
      />
      <Avatar
        size={80}
        person={{
          name: 'Gojek',
          imageId: 'OKS67lh'
        }}
      />
      <Avatar
        size={50}
        person={{
          name: 'Tokopedia',
          imageId: '1bX5QH6'
        }}
      />
    </div>
  );
}
```

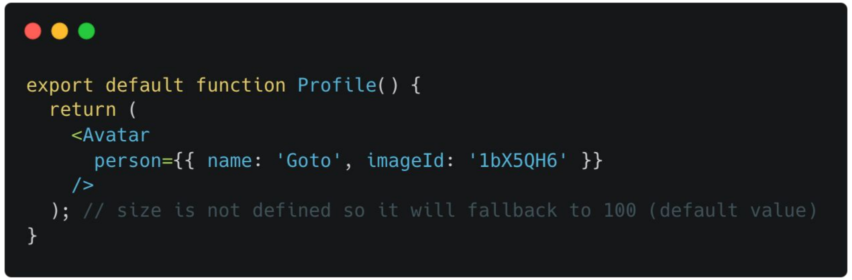
Part 3.4: Default value for props

If you want to give a prop a default value to fall back on **when no value is specified**:

1. Destructure
2. Put = and the default value



```
function Avatar({ person, size = 100 }) {
  // ...
}
```



```
export default function Profile() {
  return (
    <Avatar
      person={{ name: 'Goto', imageId: '1bX5QH6' }}
    />
  ); // size is not defined so it will fallback to 100 (default value)
}
```

Part 3.5: Props Recap

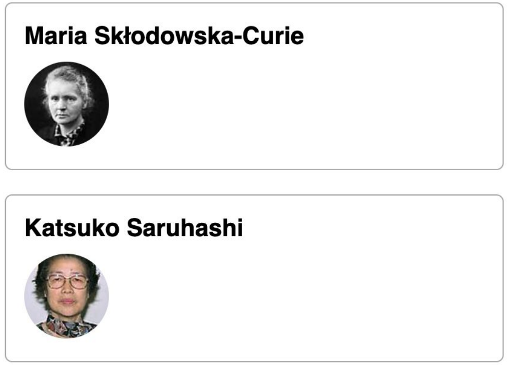
- To pass props, add them to the JSX, just like you would with HTML attributes.
- To read props, use the function Avatar({ person, size }) destructuring syntax.
- You can specify a default value like size = 100, which is used for missing and undefined props.
- Props are **read-only snapshots** in time: every render receives a new version of props.
- **You can’t change props**. When you need interactivity, you’ll need to set state (will learn later)

Generasi Gigih

Hands on

Continue from session 1 exercise, **bold is new on session 2**:

1. Create several objects that contains name and image url for avatar
2. Create a **child** component that contains header and image called Avatar
3. **Create a parent component that call child components and pass objects as props to the child components** (child 1 object 1, child 2 object 2 and so on)
4. **Try to have one empty name and use *default props***
5. **Output: child components will be rendered x times based on how many objects previously made**


Maria Skłodowska-Curie

Katsuko Saruhashi

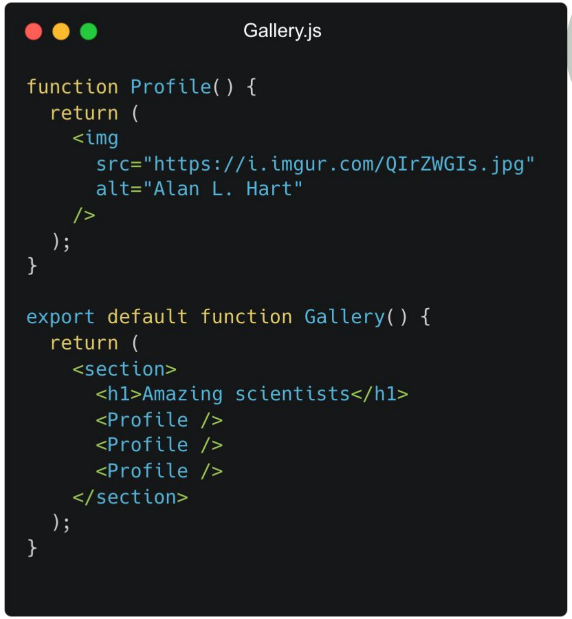
Generasi Gigih

Default and Named Exports

More about exporting and importing components

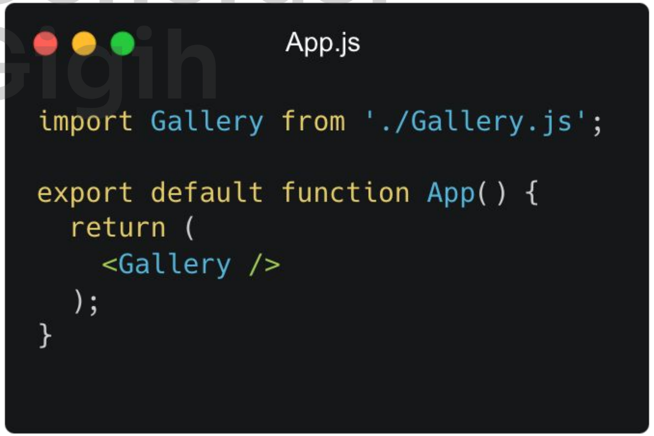
Intermezzo

Remember how to import and export component? Explain the code below


Gallery.js

```
function Profile() {
  return (
    <img
      src="https://i.imgur.com/QIrZWGIs.jpg"
      alt="Alan L. Hart"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```


App.js

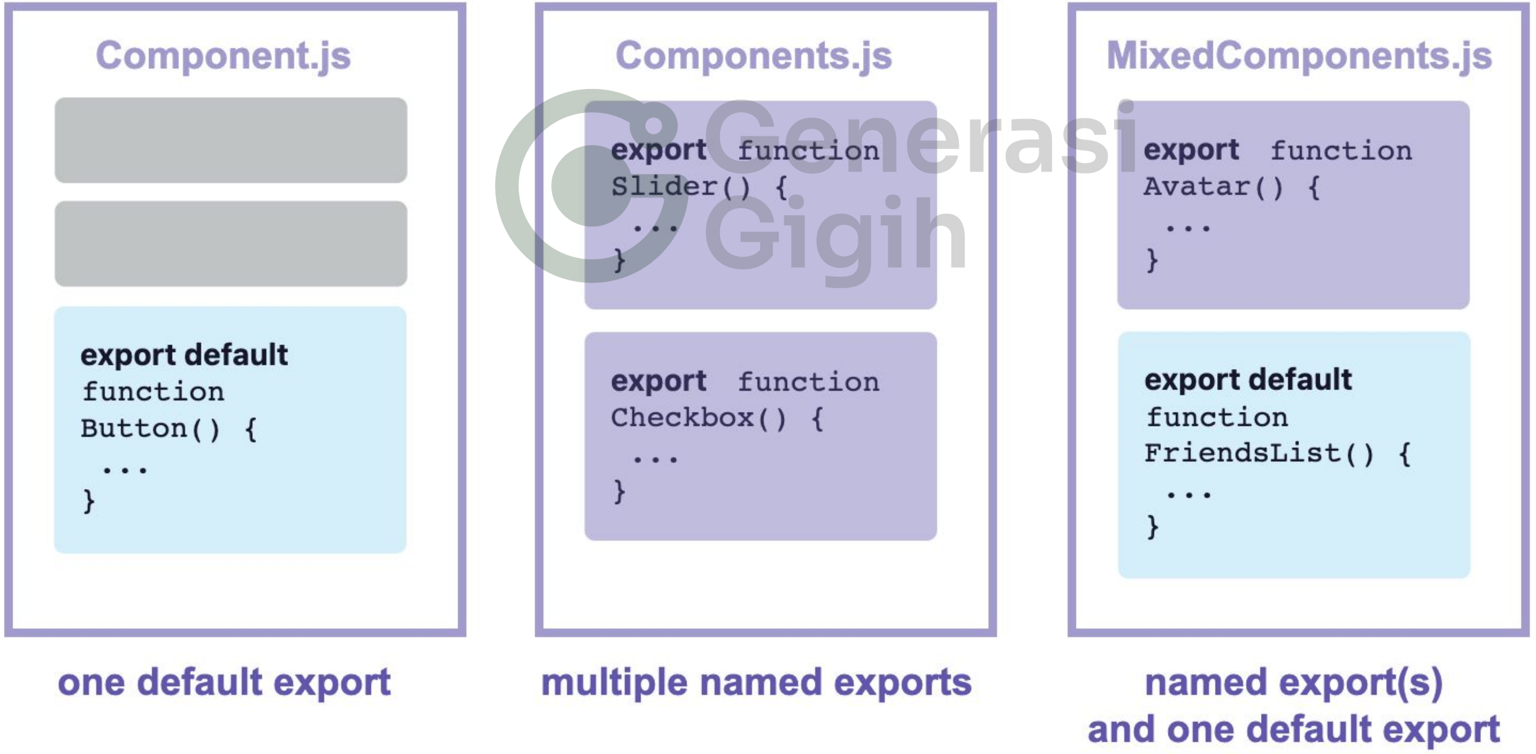
```
import Gallery from './Gallery.js';

export default function App() {
  return (
    <Gallery />
  );
}
```

Part 1: Default vs named export (1)

- Two primary ways to export values with JavaScript: **default exports** and **named exports** (our examples have only used default exports)
- A file can have **no more than one default export**, but it can have as many named exports as you like.
- People often use **default exports if the file exports only one component**, and use named exports if it exports multiple components and values.

Part 2: Default vs named export (2)



```
export default
function
Button() {
  ...
}
```

```
export  function
Slider() {
  ...
}
```

```
export  function
Checkbox() {
  ...
}
```

```
export  function
Avatar() {
  ...
}
```

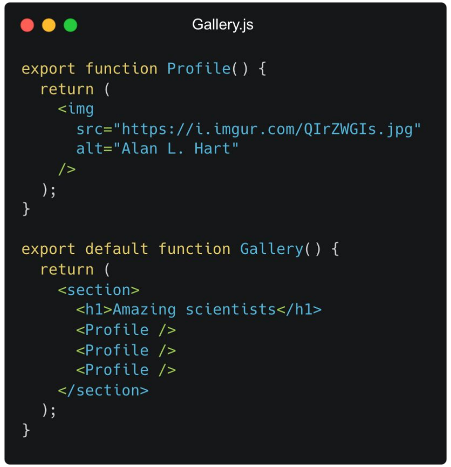
```
export default
function
FriendsList() {
  ...
}
```

Part 2: Default vs named export (3)

How to import components depends on how you export it

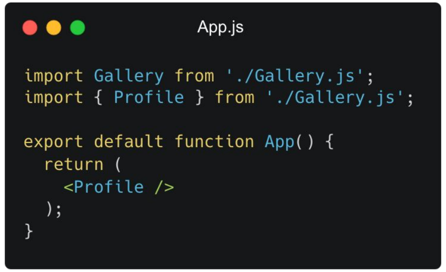
Syntax	Export statement	Import statement
Default	`export default function Button() {}`	`import Button from './Button.js';`
Named	`export function Button() {}`	`import { Button } from './Button.js';`

Part 5: Export and import multiple files (read first)


Gallery.js

```
export function Profile() {
  return (
    <img
      src="https://i.imgur.com/QIrZWGIs.jpg"
      alt="Alan L. Hart"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```

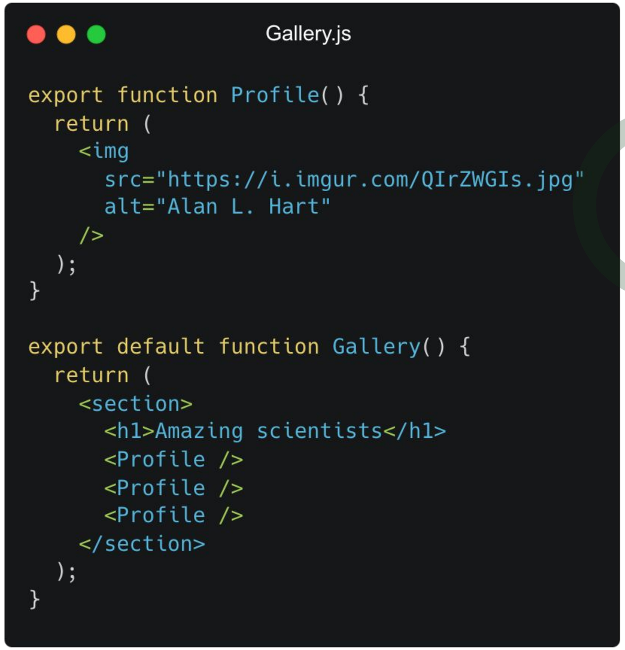

App.js

```
import Gallery from './Gallery.js';
import { Profile } from './Gallery.js';

export default function App() {
  return (
    <Profile />
  );
}
```

- Gallery.js:
  - **Exports** the Profile component as a named export called Profile.
  - **Exports** the Gallery component as a default export.
- App.js:
  - **Imports** Profile as a named import called Profile from Gallery.js.
  - **Imports** Gallery as a default import from Gallery.js.
  - **Exports** the root App component as a default export.

Exercise


Gallery.js

```
export function Profile() {
  return (
    <img
      src="https://i.imgur.com/QIrZWGIs.jpg"
      alt="Alan L. Hart"
    />
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```

Raise your hand and type on chat:

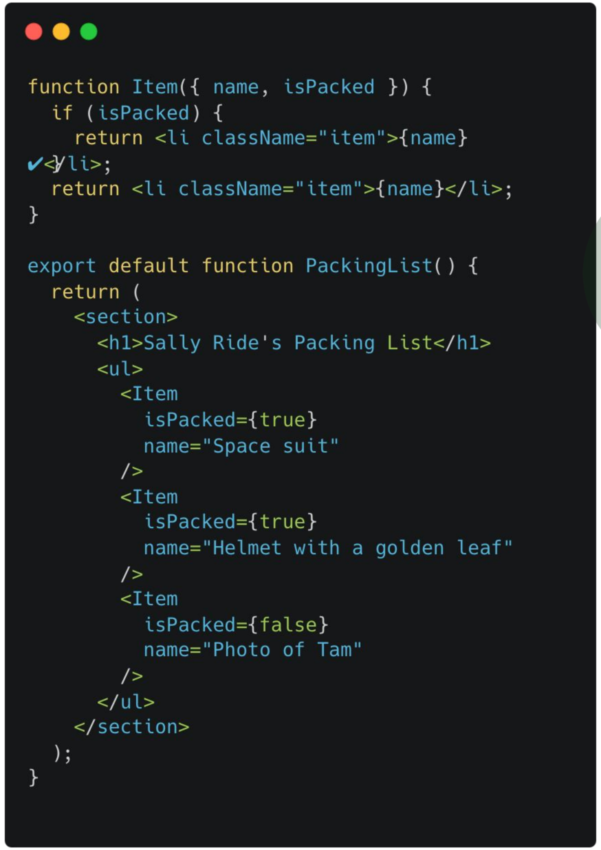
- Syntax to import Profile from Gallery.js
- Syntax to import default export from Gallery.js

Generasi Gigih

Conditional Rendering

Displaying components with conditions

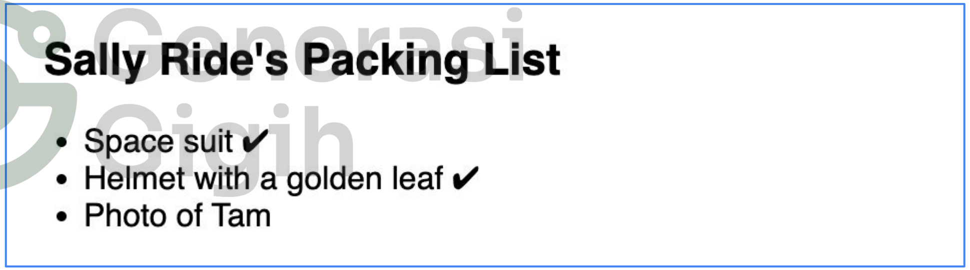
Conditional Rendering (read first)



```
function Item({ name, isPacked }) {
  if (isPacked) {
    return <li className="item">{name}
✔</li>;
  return <li className="item">{name}</li>;
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

This code will render:

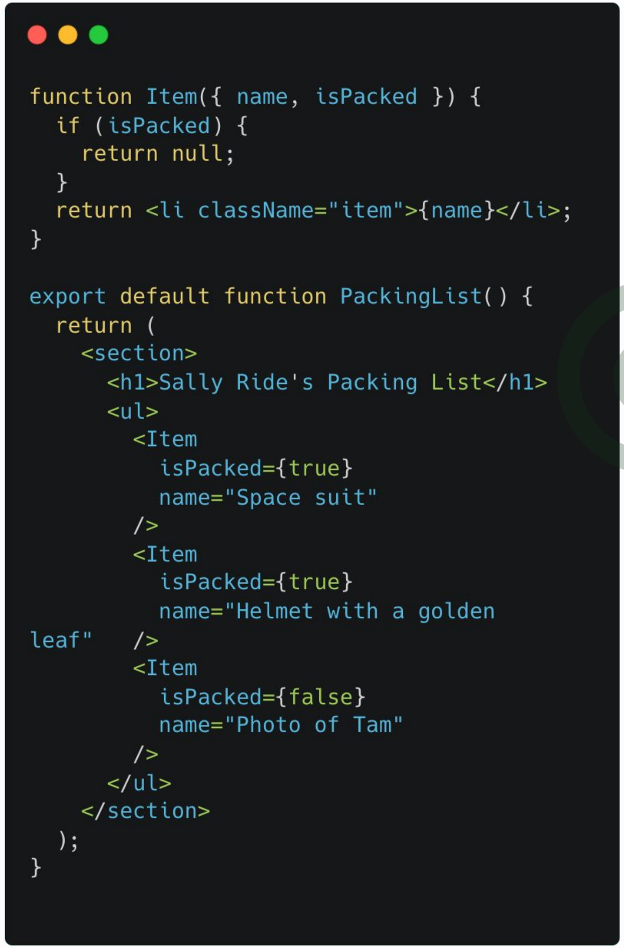


Sally Ride's Packing List

- Space suit ✔
- Helmet with a golden leaf ✔
- Photo of Tam

Please take a look at this code

✔ is an example for conditional rendering. If the condition is true it will be rendered.

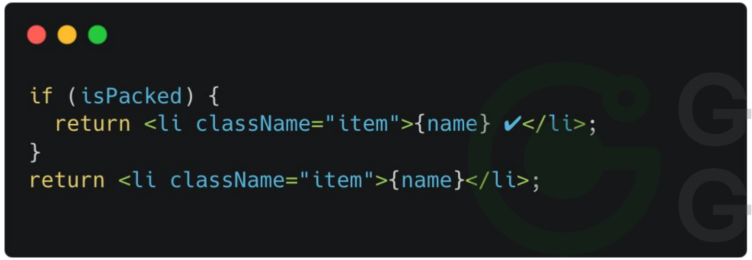


```
function Item({ name, isPacked }) {
  if (isPacked) {
    return null;
  }
  return <li className="item">{name}</li>;
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden
leaf"    />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

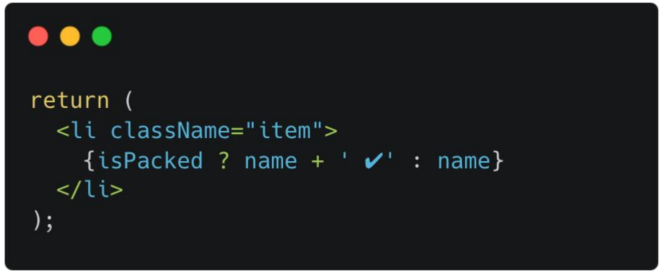
What will be the output of this code?

Conditional ternary operator (:)



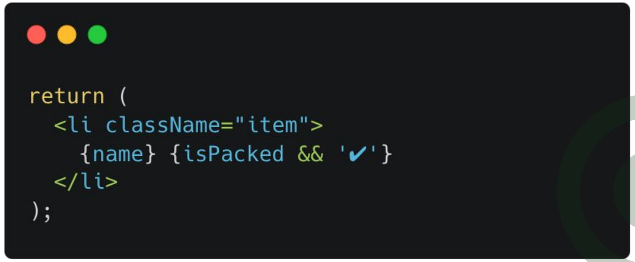
```
if (isPacked) {
  return <li className="item">{name} ✔</li>;
}
return <li className="item">{name}</li>;
```

Both are completely equivalent



```
return (
  <li className="item">
    {isPacked ? name + ' ✔' : name}
  </li>
);
```

Logical AND operator (&&)



```
return (
  <li className="item">
    {name} {isPacked && '✔'}
  </li>
);
```

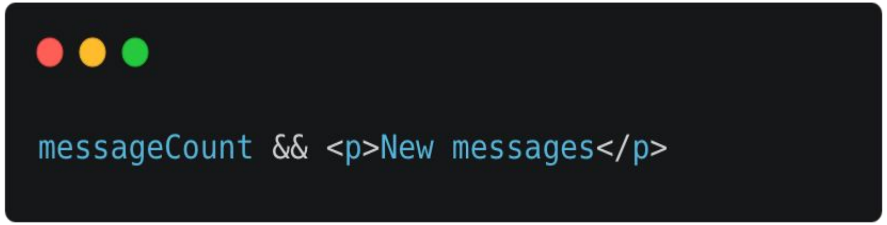
You can read this as "if isPacked, then (&&) render the checkmark, otherwise, render nothing".

- A JavaScript && expression returns the value of its right side (the checkmark) if the left side (our condition) **is true**.

- If the condition is false, the whole expression becomes false. React considers false as a "hole" in the JSX tree, just like null or undefined, and **doesn't render anything in its place**.

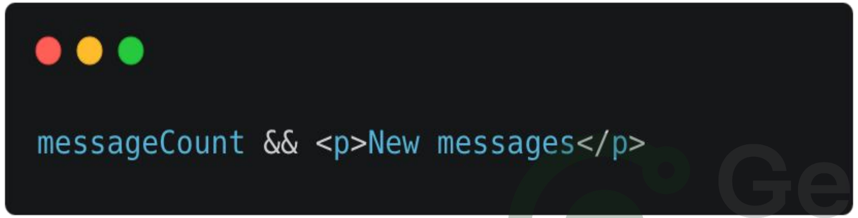
IMPORTANT NOTES

- Don’t put numbers on the left side of &&.
- To test the condition, JavaScript converts the left side to a boolean automatically.
- If the left side is 0, then the whole expression gets that value (0), and
- React will happily **render 0** rather than nothing.



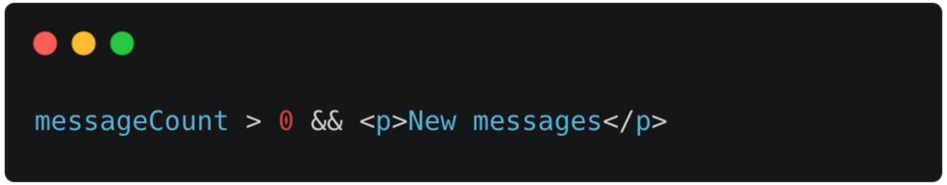
```
messageCount && <p>New messages</p>
```

What is a more correct way to write this expression? Answer on next slide



```
messageCount && <p>New messages</p>
```

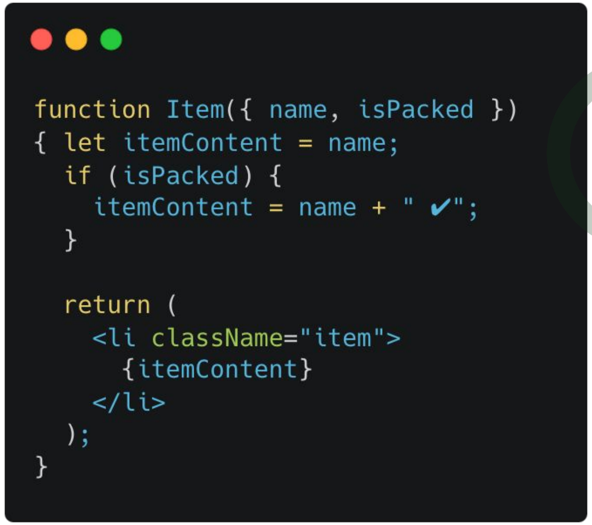
Wrong



```
messageCount > 0 && <p>New messages</p>
```

Right

Using variable



```
function Item({ name, isPacked })
{ let itemContent = name;
  if (isPacked) {
    itemContent = name + " ✔";
  }

  return (
    <li className="item">
      {itemContent}
    </li>
  );
}
```

Please take a look at this code and try to understand it.

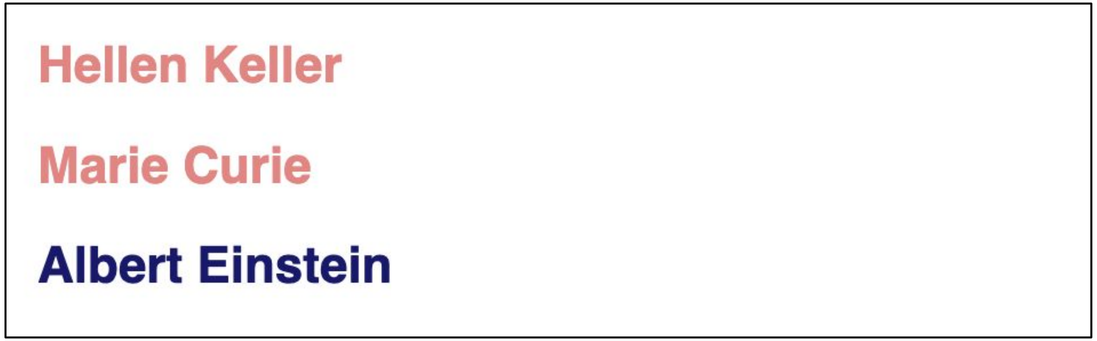
We can also conditionally assign a variable with JSX, then escape to JS and render the variable.

Generasi Gigih

Hands on

Conditional rendering

1. Create several objects that contain name and gender (female or male)
2. Return **all the objects' name in h1**
3. If gender is female h1 will be in lightcoral color
4. If gender is male h1 will be in midnight blue color



Hellen Keller

Marie Curie

Albert Einstein

Generasi Gigih

What we learned

1. Props
2. Export and import component
3. Conditional rendering

Generasi Gigih

Q&A!

Homework

Create anything you like using what you’ve learned so far!

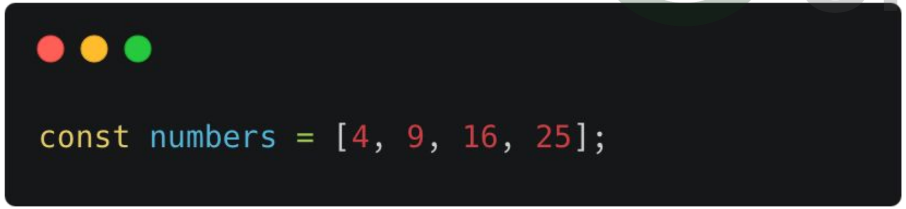
Generasi Gigih

Warm up for session 3

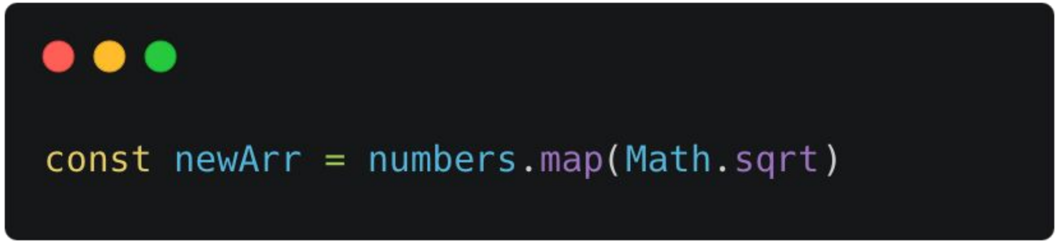
Part 1: JS map function

Remember how to display array of object using map?

Who can return a new array with the square root of all element values?
(you may google first what javascript function can return square root)



```
const numbers = [4, 9, 16, 25];
```

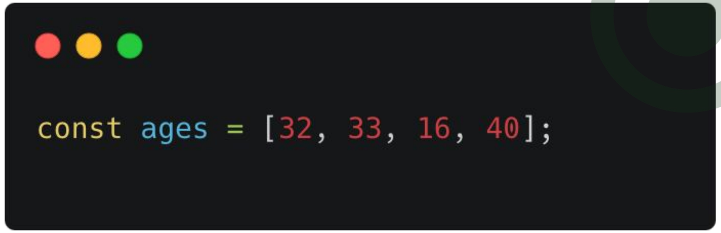


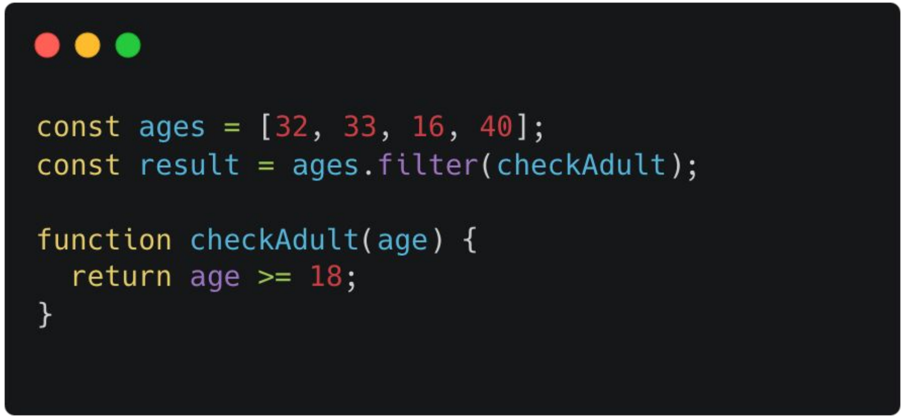
```
const newArr = numbers.map(Math.sqrt)
```

Part 2: JS filter function

Remember how to filter array?

Return an array of all values in ages[] that are 18 or over:


```javascript
const ages = [32, 33, 16, 40];
```


```javascript
const ages = [32, 33, 16, 40];
const result = ages.filter(checkAdult);

function checkAdult(age) {
  return age >= 18;
}
```

See you in the next session!