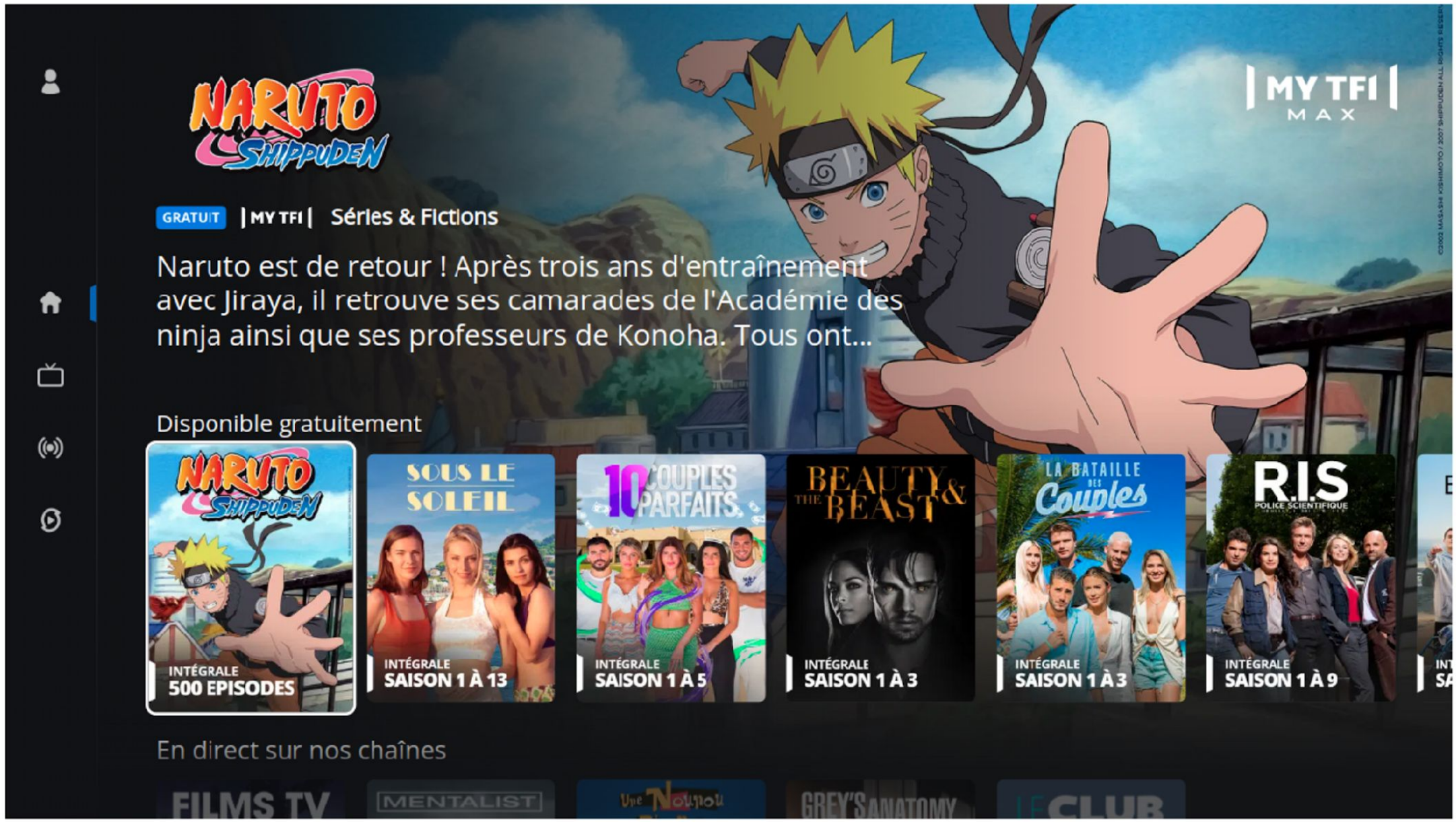
Bringing the best of React to TV navigation

A React-Like Solution for Remote Control

MY TF1


NARUTO SHIPPUDEN
MY TF1 MAX
GRATUIT | MY TF1 | Séries & Fictions
Naruto est de retour ! Après trois ans d'entraînement avec Jiraya, il retrouve ses camarades de l'Académie des ninja ainsi que ses professeurs de Konoha. Tous ont...
Disponible gratuitement
INTÉGRALE 500 EPISODES
SOUS LE SOLEIL
INTÉGRALE SAISON 1 À 13
10 COUPLES PARFAITS
INTÉGRALE SAISON 1 À 5
BEAUTY & THE BEAST
INTÉGRALE SAISON 1 À 3
LA BATAILLE DES Couples
INTÉGRALE SAISON 1 À 3
R.I.S POLICE SCIENTIFIQUE
INTÉGRALE SAISON 1 À 9
En direct sur nos chaînes
FILMS TV
MENTALIST
GREY'S ANATOMY
LE CLUB

About us

Pierre Poupin

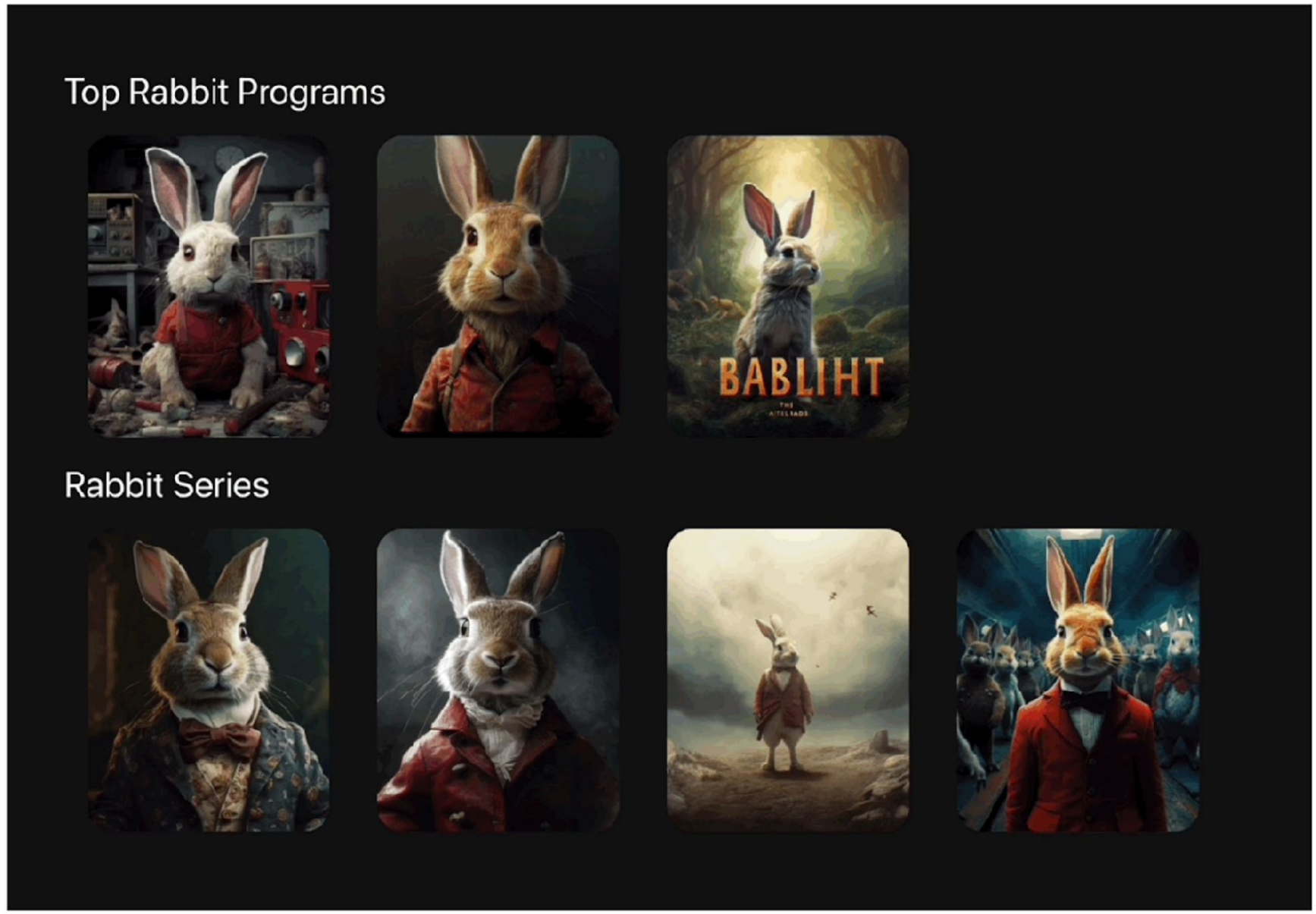
github.com/pierpo

Mathieu Fedrigo

twitter.com/mat_fdg

Context: Smart TVs

A simple TV App



Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

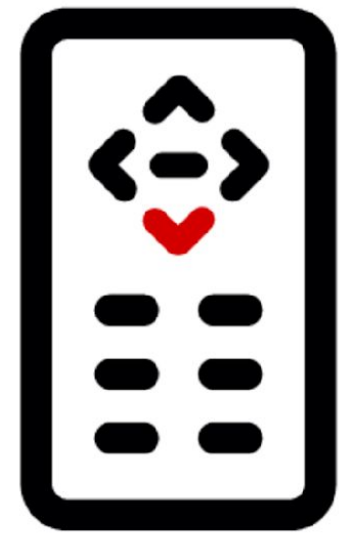
Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series



Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

Rabbit Series

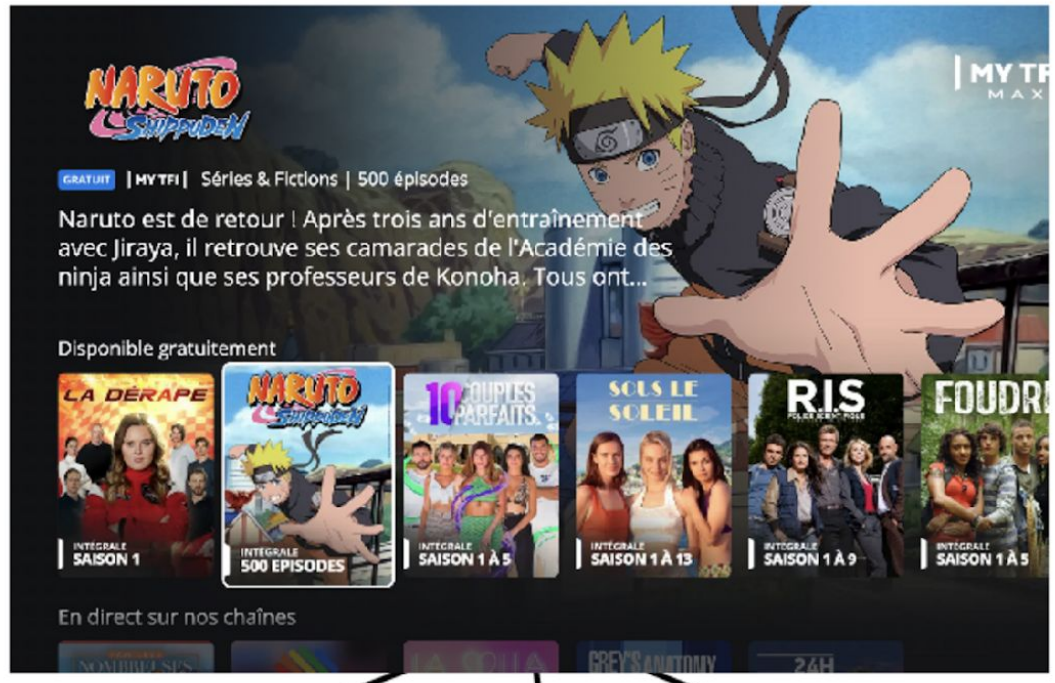
Top Rabbit Programs

Rabbit Series

Top Rabbit Programs

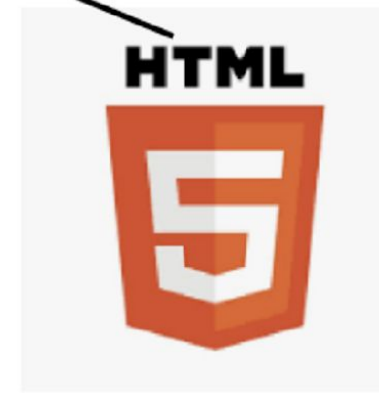
Rabbit Series

Our goal


NARUTO SHIPPUDEN
GRATUIT | MYTF1 | Séries & Fictions | 500 épisodes
Naruto est de retour ! Après trois ans d'entraînement avec Jiraya, il retrouve ses camarades de l'Académie des ninja ainsi que ses professeurs de Konoha. Tous ont...
Disponible gratuitement
LA DÉRAPE
INTÉGRALE SAISON 1
NARUTO SHIPPUDEN
INTÉGRALE 500 EPISODES
10 COUPLES PARFAITS
INTÉGRALE SAISON 1 À 5
SOUS LE SOLEIL
INTÉGRALE SAISON 1 À 13
R.I.S
INTÉGRALE SAISON 1 À 9
FOUDRE
INTÉGRALE SAISON 1 À 5
En direct sur nos chaînes


androidtv


tv+


HTML

Our goal

Android codebase

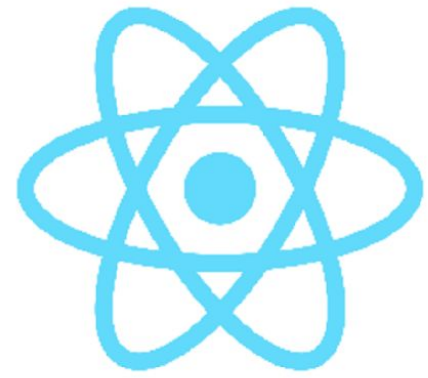
iOS codebase

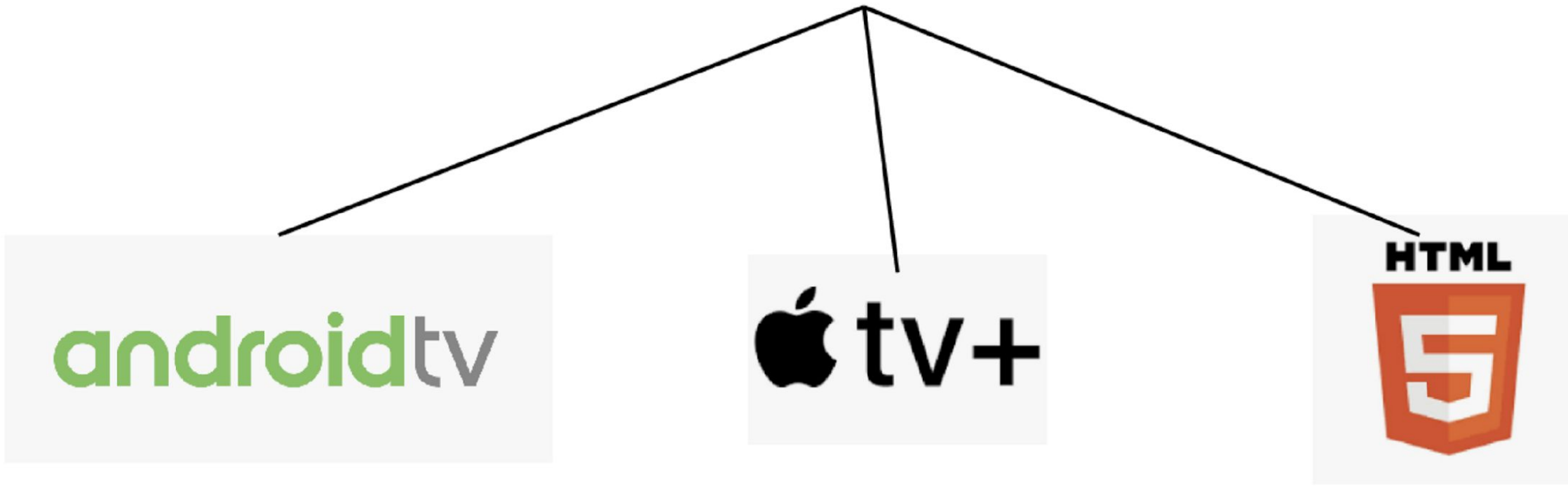
Web codebase

Our goal

React multiplatform codebase

Thanks React Native! Classic!





Our goal

But... How do I handle spatial navigation with React Native?

press "right"

How do I handle spatial navigation?

Native TV OS components?

How do I handle spatial navigation?

Native TV OS components?

❌ Not available on the web!

❌ Not multiplatform...

How do I handle spatial navigation?

Native TV OS components?

❌ Not available on the web!

❌ Not multiplatform...

Custom navigation system in React?

How do I handle spatial navigation?

Native TV OS components?

❌ Not available on the web!

❌ Not multiplatform...

Custom navigation system in React?

✅ Multiplatform

🤯 but it's a hard problem!

Why is it hard?

Many edge cases of "where am I when *[something]* happens?"

rows, grids, nested grids, an element disappears, the focused element disappears, a modal appears...

Why is it hard?

Many edge cases of "where am I when *[something]* happens?"

rows, grids, nested grids, an element disappears, the focused element disappears, a modal appears...

Keeping track of the positions of the elements in React

Why is it hard?

Many edge cases of "where am I when *[something]* happens?"

rows, grids, nested grids, an element disappears, the focused element disappears, a modal appears...

Keeping track of the positions of the elements in React

We will cover:

- The expressive API we should have
- How we handled all this logic
- How we plugged this all to React!

A hard problem with many solutions

First iteration: we faced dead-ends!

- We had a **very imperative logic** ☐
  - **Manual** spatial registering of components
  - ...in duplication of the React JSX declarations
- Which lead to a very **leaky abstraction** 🚰

A hard problem with many solutions

First iteration: we faced dead-ends!

- Result of the first iteration:
  - ☐ Lots of bugs
  - 🐇 No dynamic updates (add/remove elements)
  - 🌀 Layout could not be too deeply nested

The declarative API

What we want to do

Top Rabbit Programs

Rabbit Series

The declarative API

Basic Program

```tsx
const Program = () => <ProgramLayout />;

export const Page = () => (
  <View>
    <Program />
    <Program />
  </View>
);
```

The declarative API

Focusable Program

```
const Program = () => (
  <Node>
    <ProgramLayout />
  </Node>
);

export const Page = () => (
  <View>
    <Program />
    <Program />
  </View>
);
```

The declarative API

Vertical container

```
const Program = () => (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);

export const Page = () => (
  <Node orientation="vertical">
    <View>
      <Program />
      <Program />
    </View>
  </Node>
);
```

The declarative API

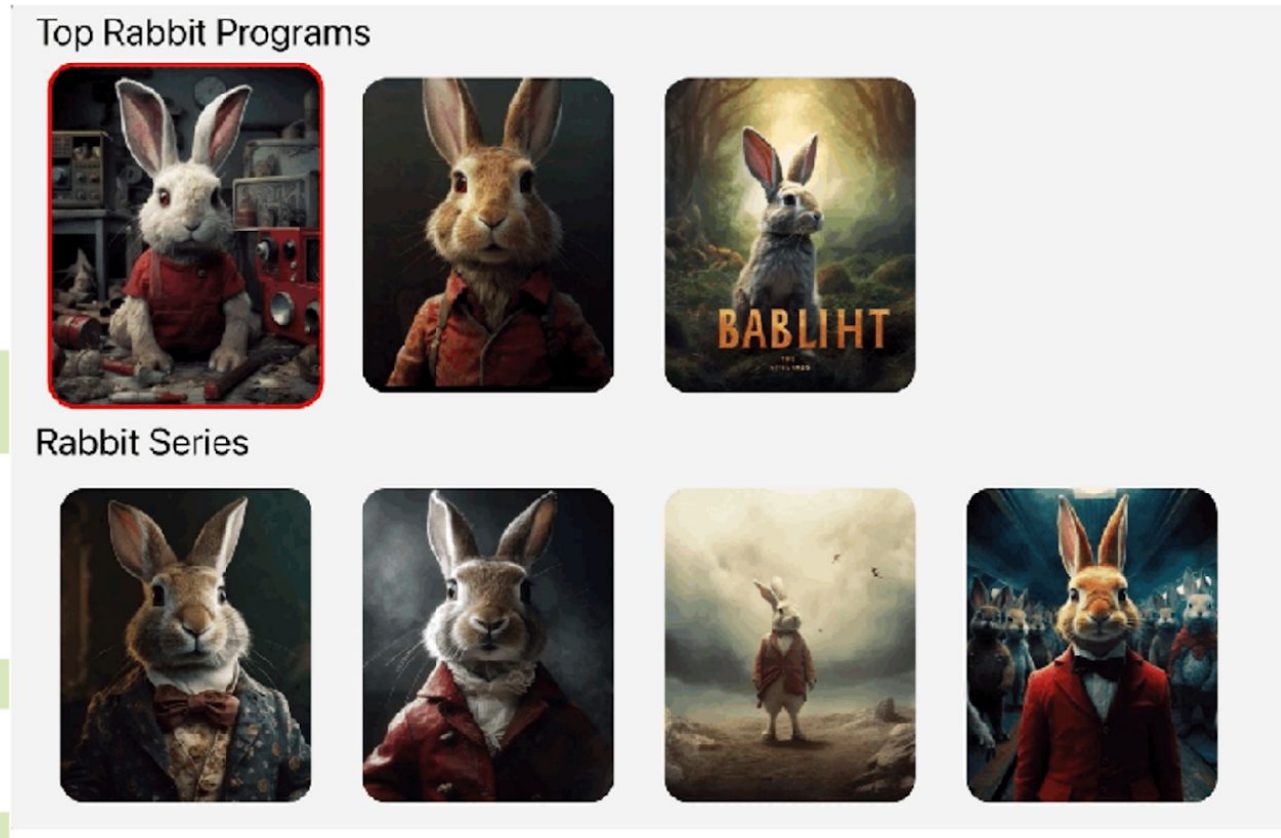
Horizontal container → Row

```tsx
const Program = () => (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);

export const Page = () => (
  <Node orientation="horizontal">
    <View style={{ flexDirection: 'row' }}>
      <Program />
      <Program />
    </View>
  </Node>
);
```

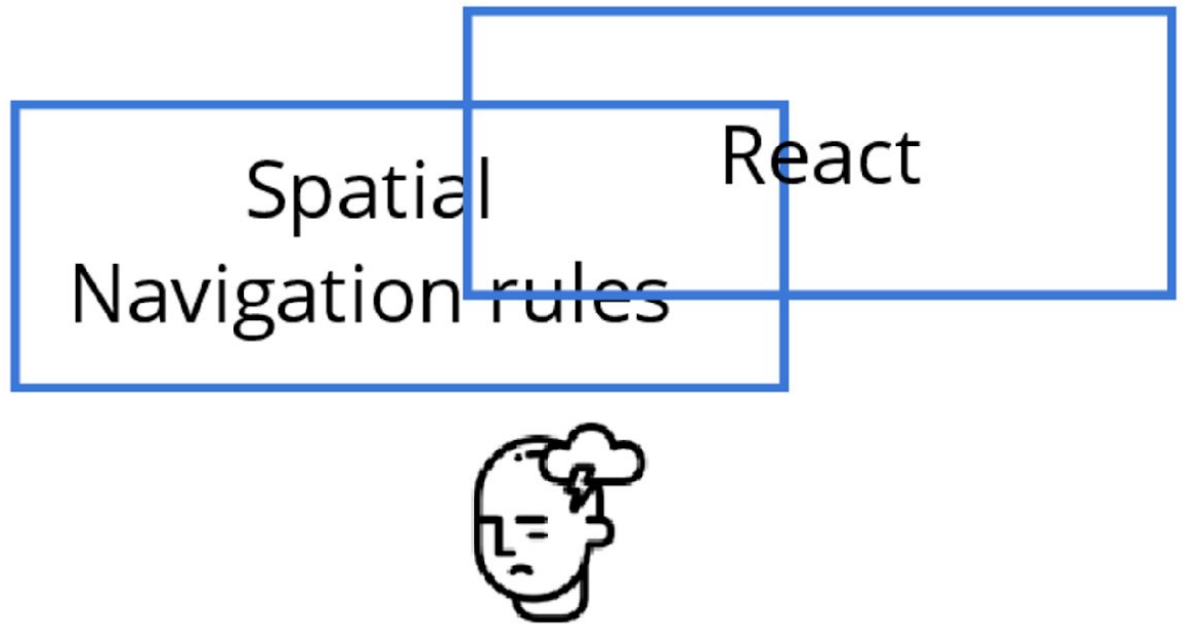
The declarative API

Whole page

```
const Program = () => (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);

export const Page = () => (
  <>
    <Typography>Top Rabbit Programs</Typography>
    <Node orientation="horizontal">
      <View style={{ flexDirection: 'row' }}>
        <Program />
        <Program />
        <Program />
      </View>
    </Node>
    <Typography>Rabbit Series</Typography>
    <Node orientation="horizontal">
      <View style={{ flexDirection: 'row' }}>
        <Program />
        <Program />
        <Program />
        <Program />
      </View>
    </Node>
  </>
);
```



The Spatial Navigation problem



The Spatial Navigation problem

Spatial Navigation rules

React

externalize the navigation logic!

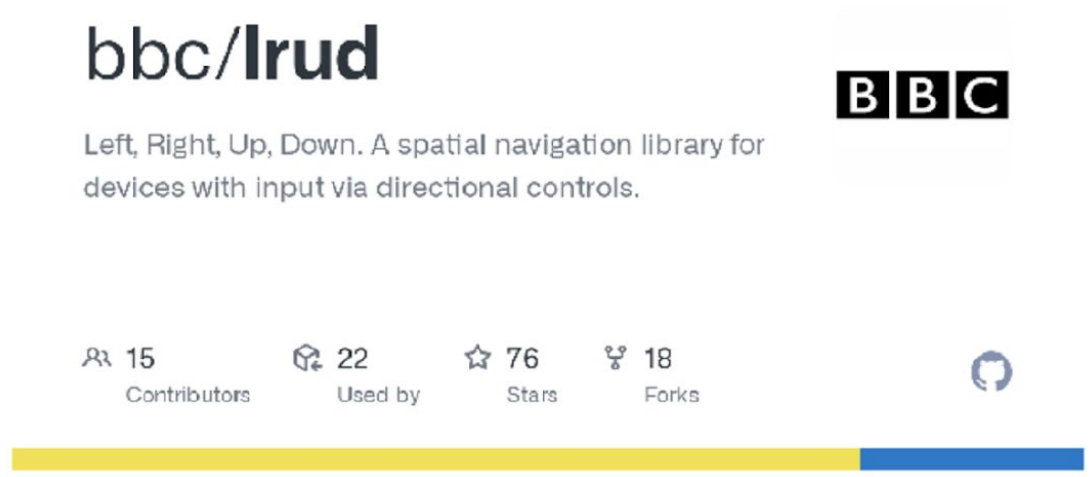
The Spatial Navigation problem

Is there a lib for that? 📚

The Spatial Navigation problem

Is there a lib for that? 📚

github.com/bbc/lrud

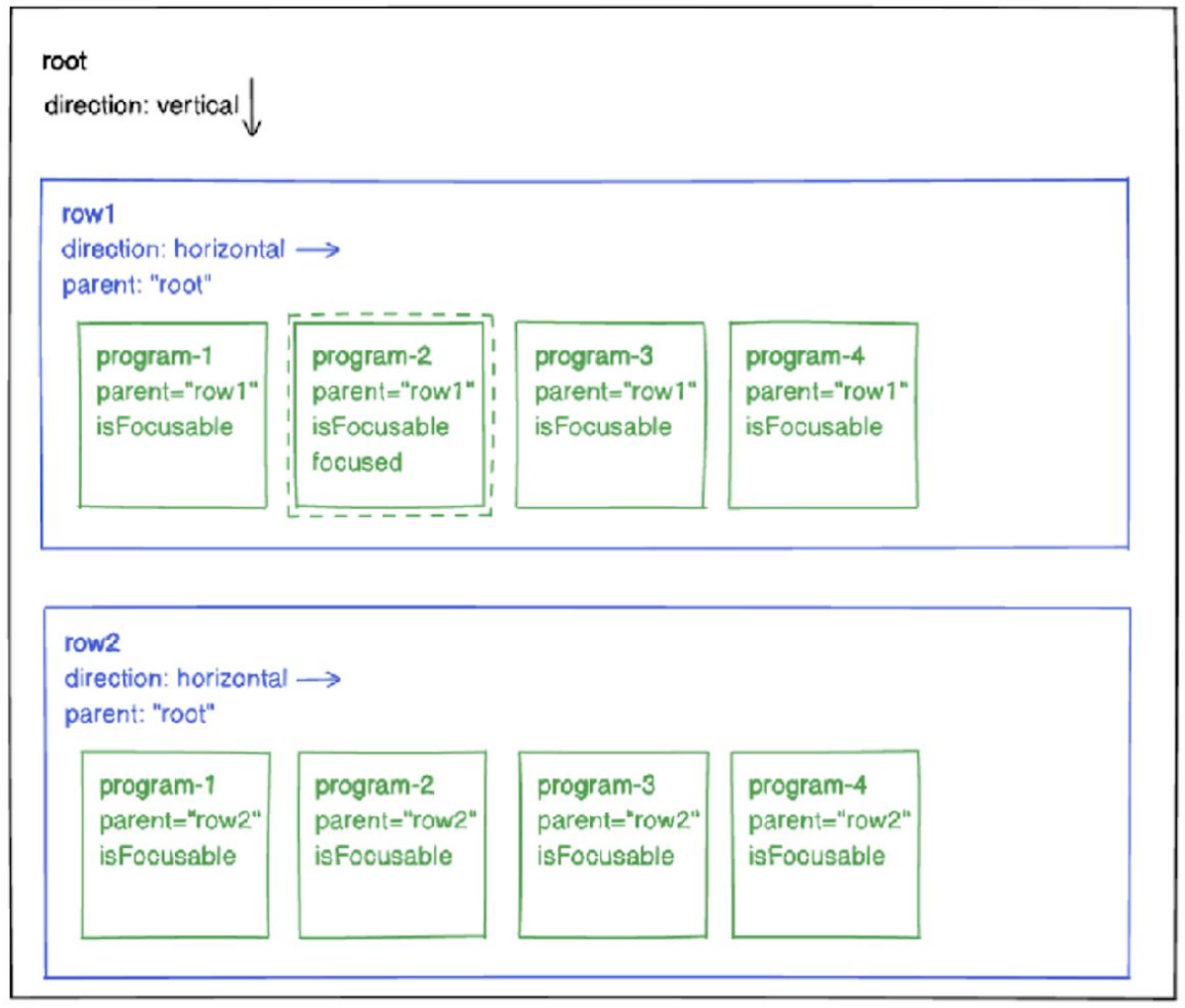


LRUD = left-right-up-down

How LRUD works

Represent a spatial layout with a data structure

It's UI agnostic! 🦴



How LRUD works

Represent a spatial layout with a data structure

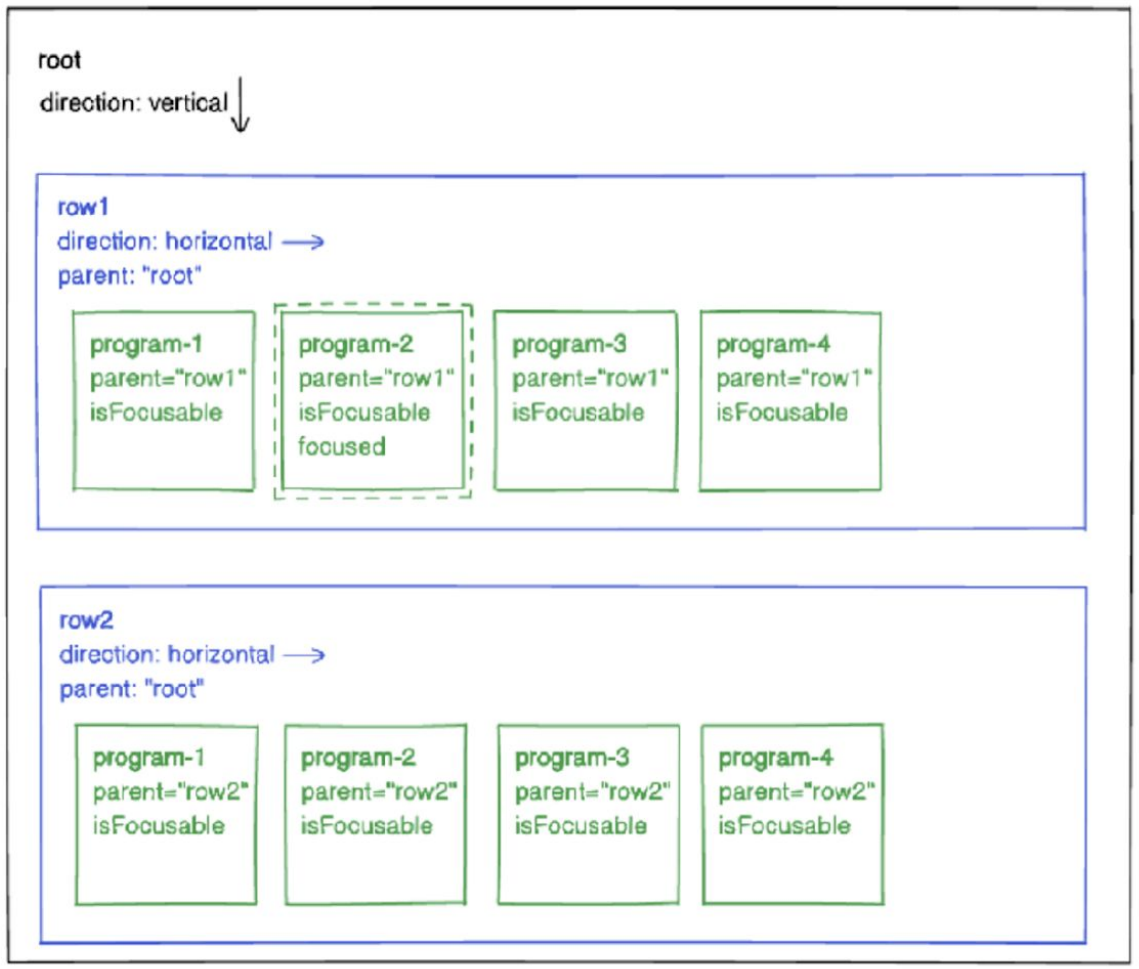
It's UI agnostic! 🦴

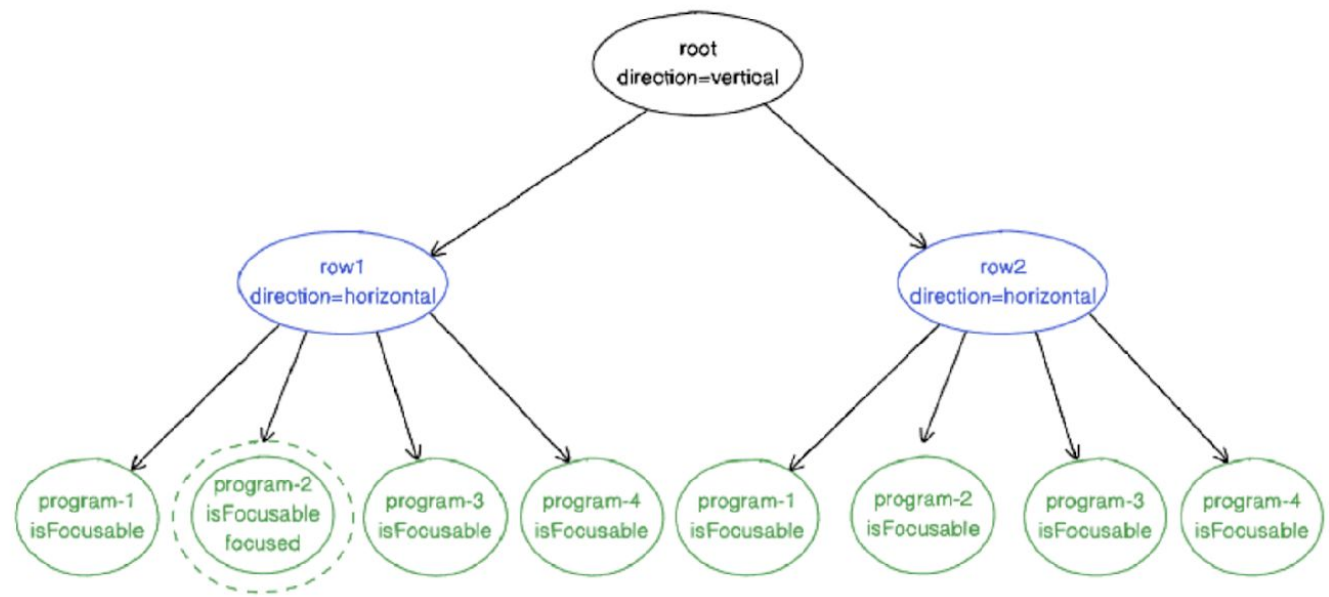
```
// not the exact structure, but global idea
{
  root: {
    id: 'root-element',
    orientation: 'vertical',
    children: [
      { id: 'row1', orientation: 'horizontal', children: [
        {id: 'program-1-1', isFocusable: true }
      ]},
      { id: 'row2', orientation: 'horizontal', children: [
        {id: 'program-2-1', isFocusable: true }
      ]}
    ]
  }
}
```

How LRUD works

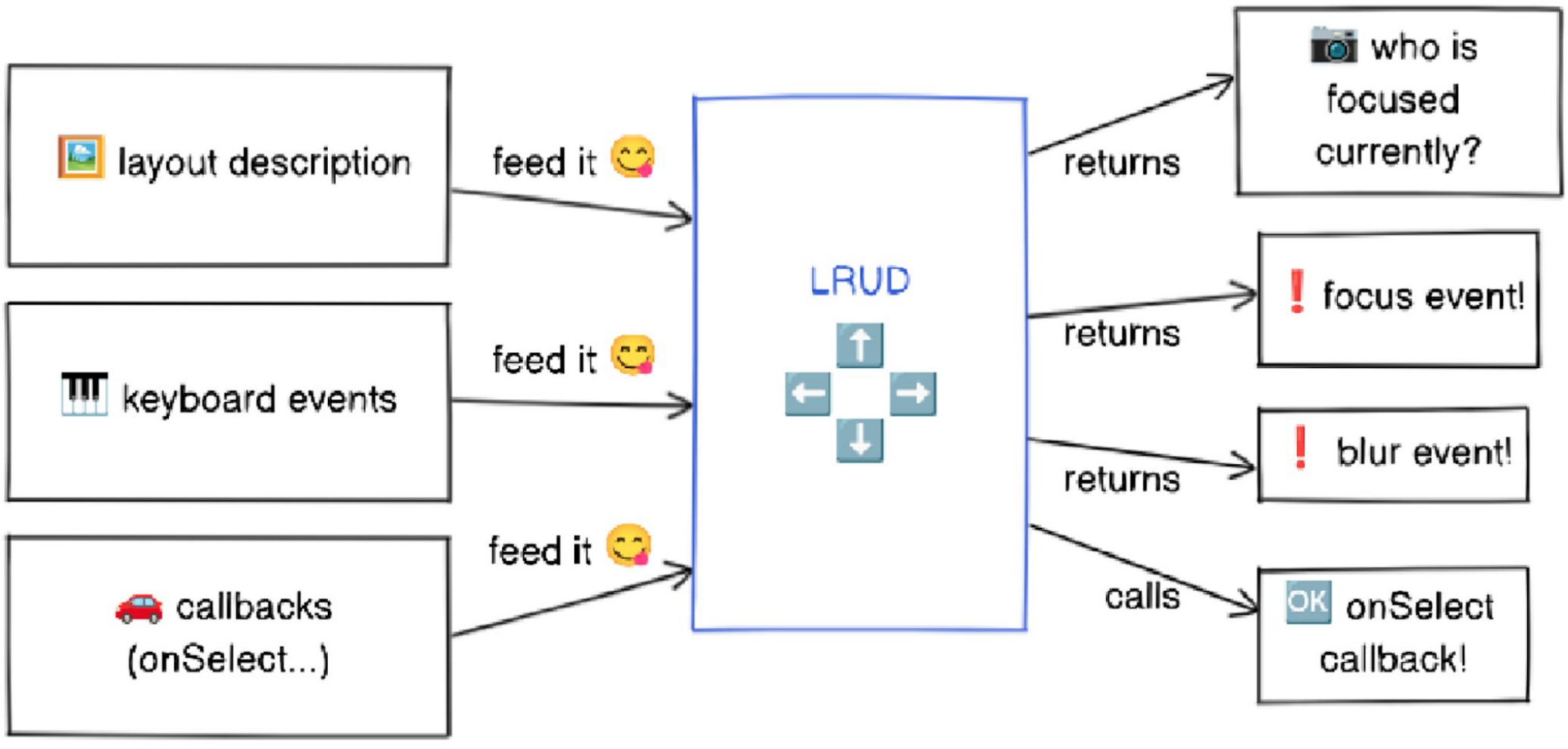
LRUD's representation is actually a tree!

(just like React 😎 it's a match! ❤️🔥)



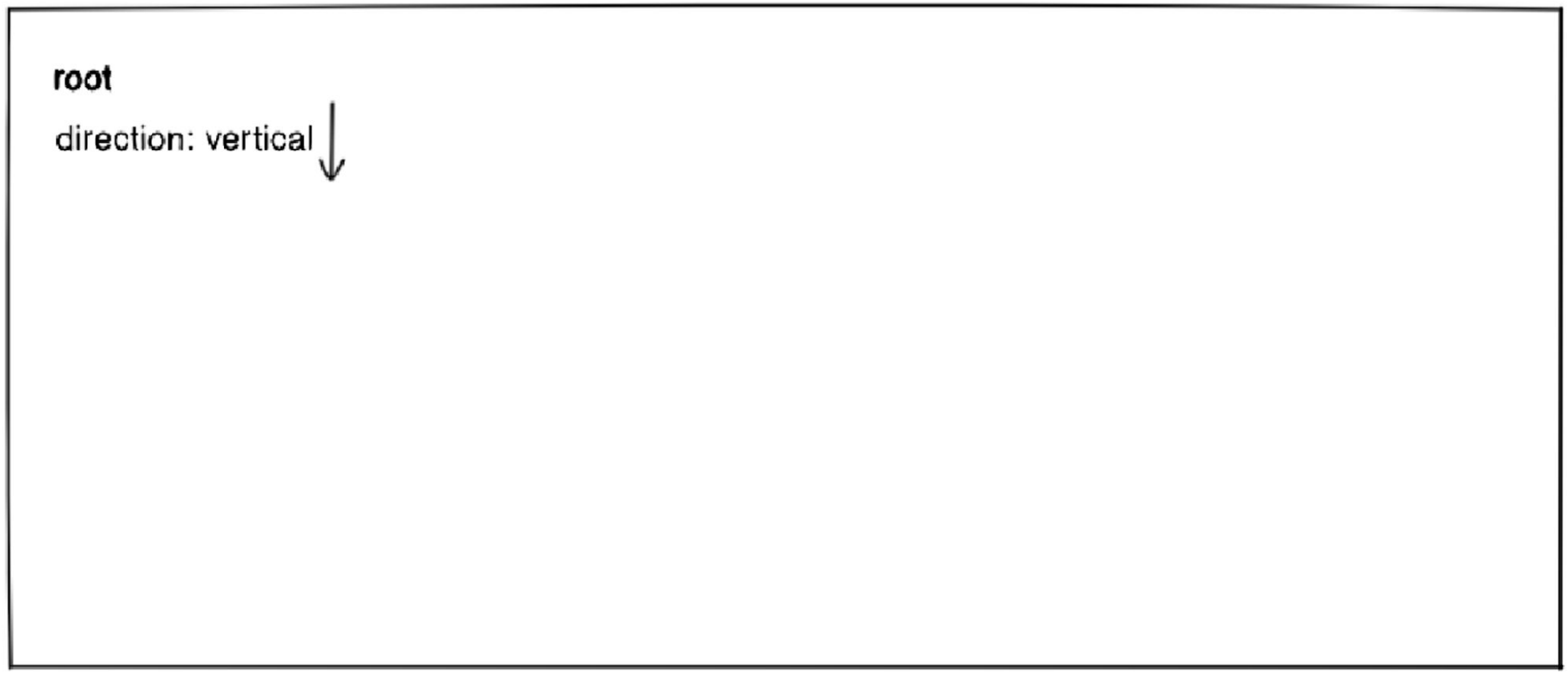


LRUD: Summary



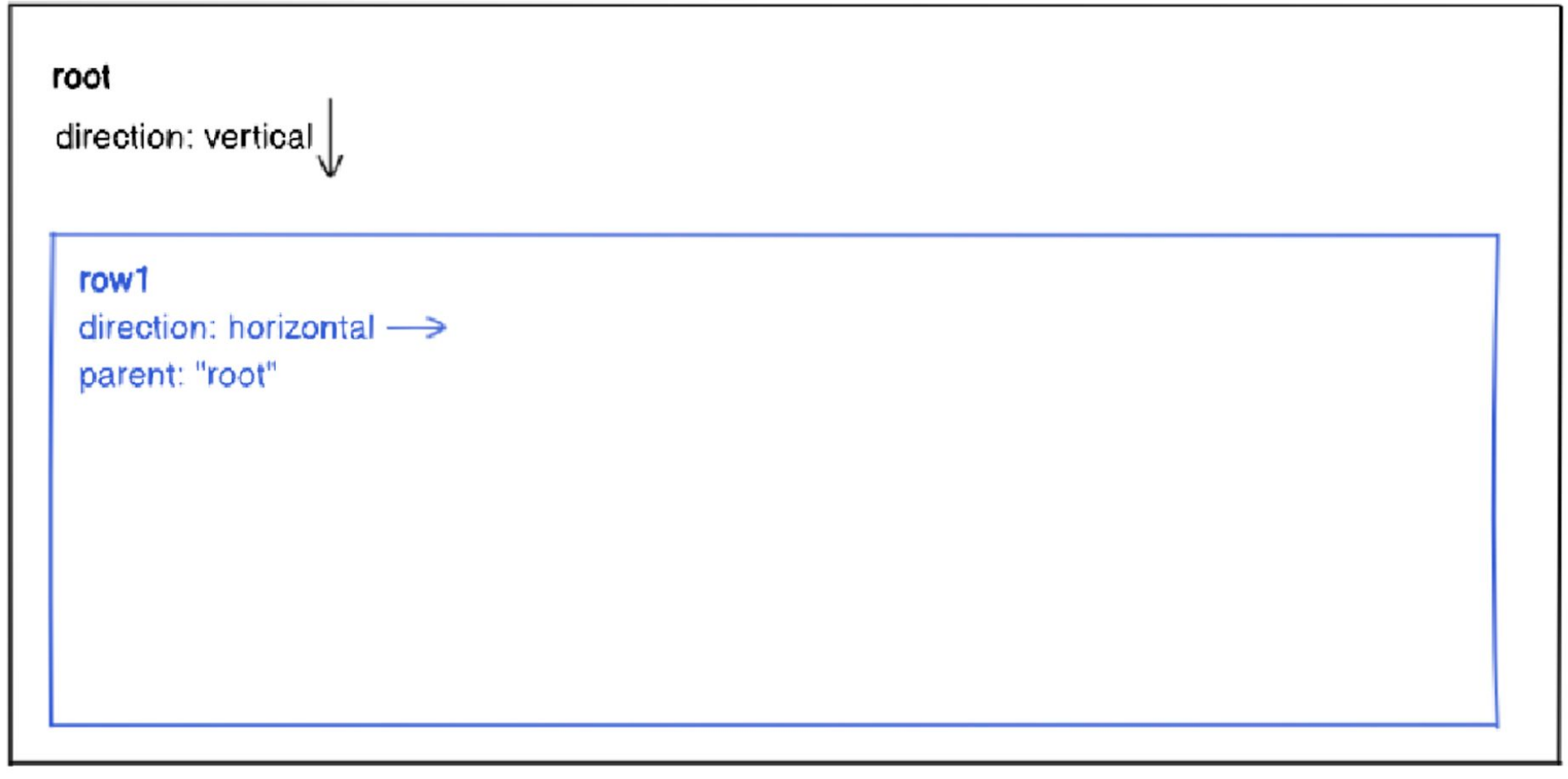
Register the root

```
1 lrud.registerElement('root', { direction: 'vertical' })
```



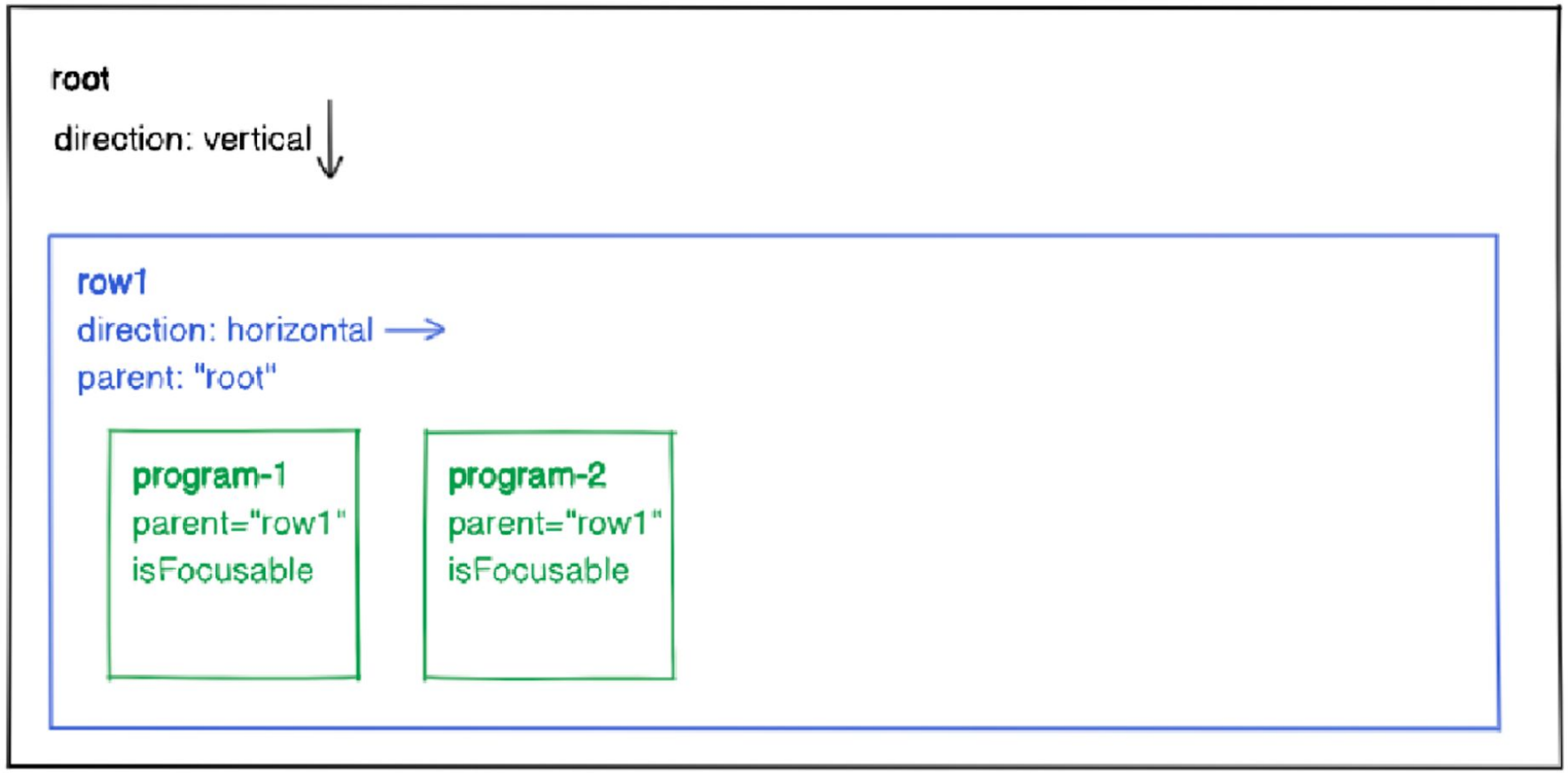
Register a row

```
1 lrud.registerElement('root', { direction: 'vertical' })
2 lrud.registerElement('row1', { parent: 'root', direction: 'horizontal' })
```



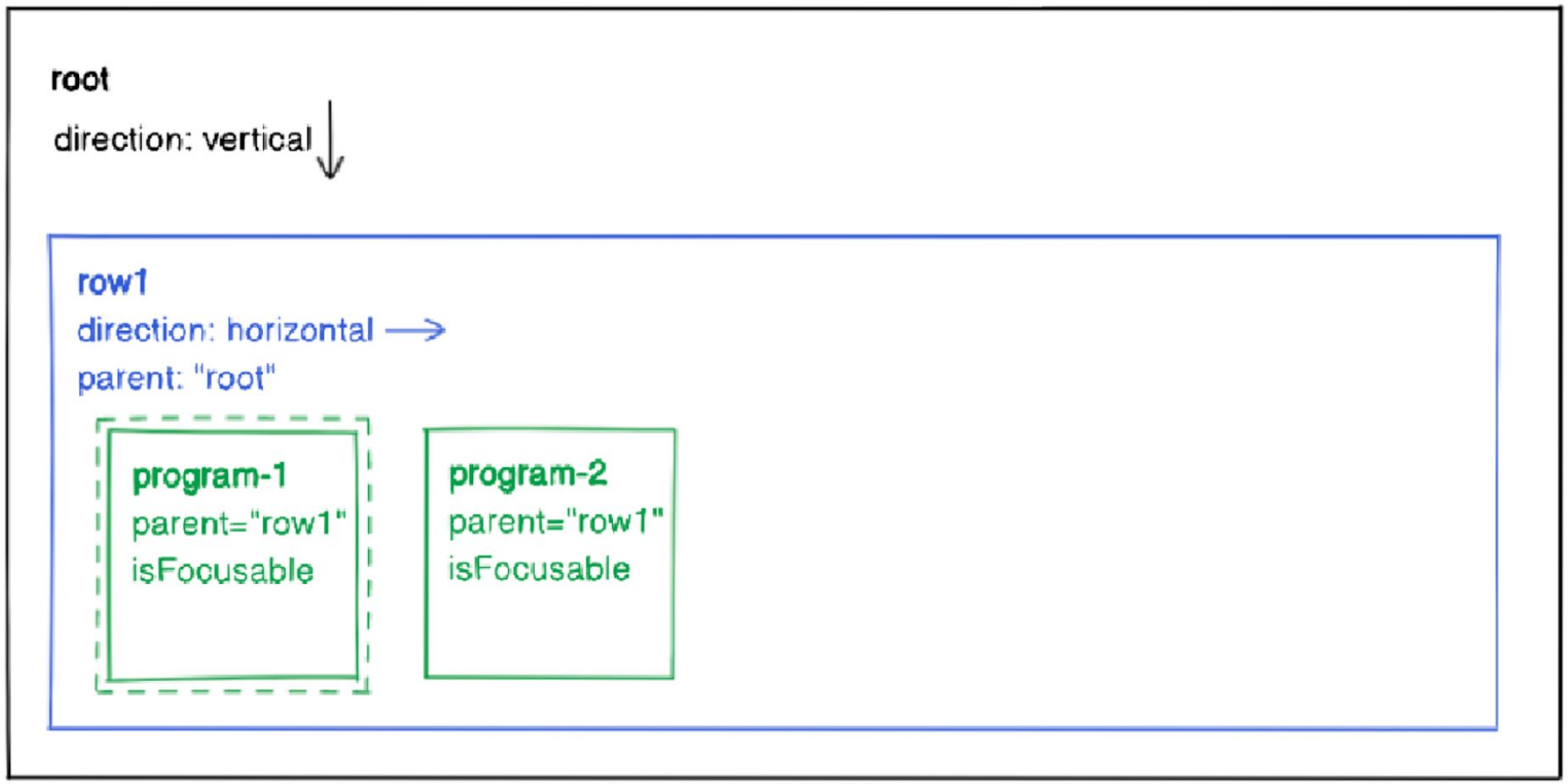
Register programs

```
1 lrud.registerElement('root', { direction: 'vertical' })
2 lrud.registerElement('row1', { parent: 'root', direction: 'horizontal' })
3 lrud.registerElement('program-1', { parent: 'row1', isFocusable: true })
4 lrud.registerElement('program-2', { parent: 'row1', isFocusable: true })
```



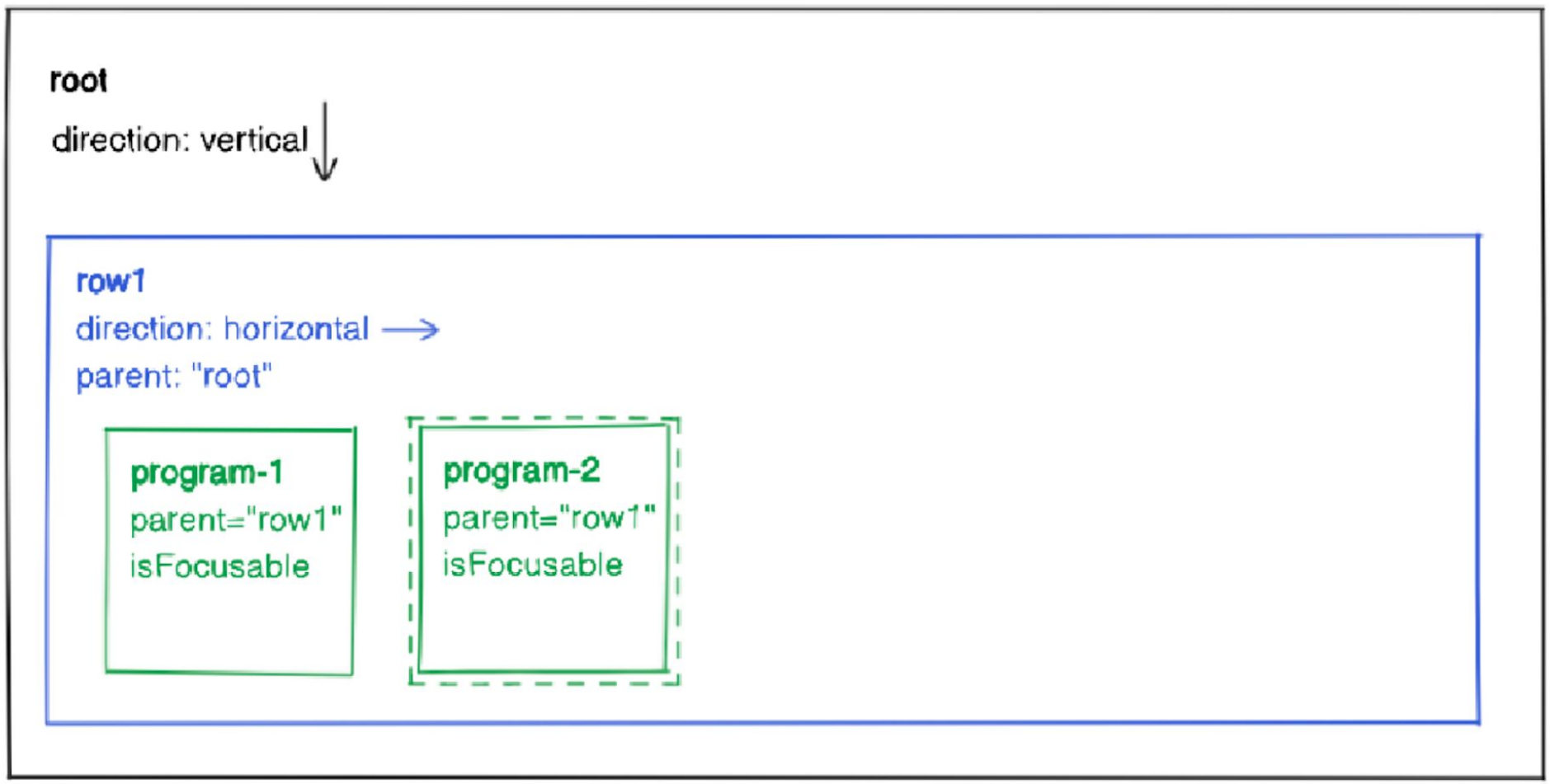
Press right!

```
1 lrud.registerElement('root', { direction: 'vertical' })
2 lrud.registerElement('row1', { parent: 'root', direction: 'horizontal' })
3 lrud.registerElement('program-1', { parent: 'row1', isFocusable: true })
4 lrud.registerElement('program-2', { parent: 'row1', isFocusable: true })
5 lrud.handleKeyEvent('right') // actually, any key to get first focus
```



Press right (again)!

```
1 lrud.registerElement('root', { direction: 'vertical' })
2 lrud.registerElement('row1', { parent: 'root', direction: 'horizontal' })
3 lrud.registerElement('program-1', { parent: 'row1', isFocusable: true })
4 lrud.registerElement('program-2', { parent: 'row1', isFocusable: true })
5 lrud.handleKeyEvent('right')
6 lrud.handleKeyEvent('right')
```



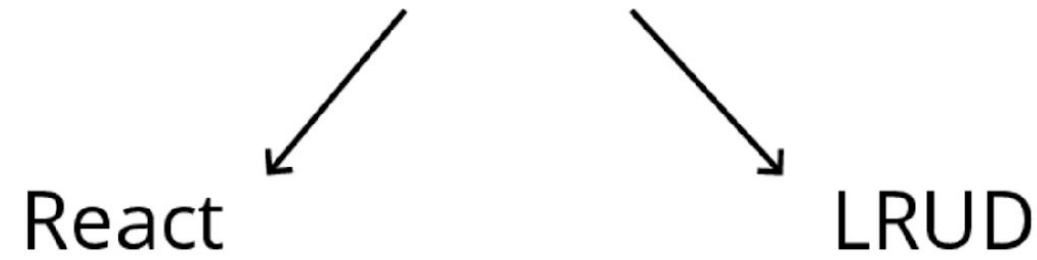
LRUD + React

a love story with many chapters

LRUD + React

a love story with many chapters

2 trees that need to live together



LRUD setup

LRUD setup

```
export const lrud = new Lrud();
```

LRUD setup

```
export const lrud = new Lrud();

lrud.registerNode('root', { orientation: 'vertical' });
```

LRUD setup

```
export const lrud = new Lrud();

lrud.registerNode('root', { orientation: 'vertical' });

window.addEventListener('keydown', ({ code }) ⇒
  lrud.handleKeyEvent({ direction: mapKeyCode(code) }, { forceFocus: true }),
);
```

Empty node

```
const Program = () ⇒ (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);
```

Empty node

```tsx
const Program = () ⇒ (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);
```

```tsx
export const Node = ({ children }: Props) ⇒ {
  return children;
};
```

Empty node

```tsx
const Program = () => (
  <Node isFocusable>
    <ProgramLayout />
  </Node>
);
```

```tsx
type Props = {
  isFocusable?: boolean;
  orientation?: 'horizontal' | 'vertical';
  children: React.ReactNode;
};

export const Node = ({ children }: Props) => {
  return children;
};
```

Register Node

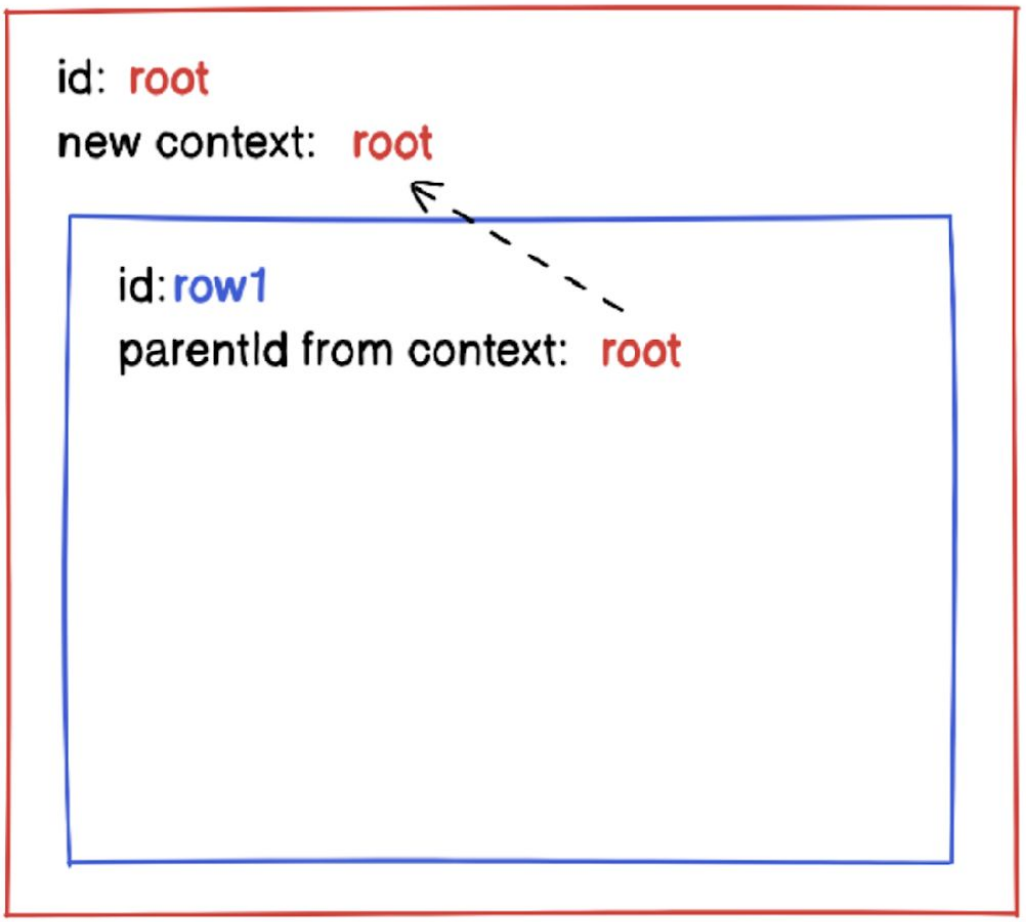
```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const id = useId();

  useEffect(() => {
    lrud.registerNode(id, {
      parent: '??????????',
      isFocusable,
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return children;
};
```

Basic Context

id: root
new context: root

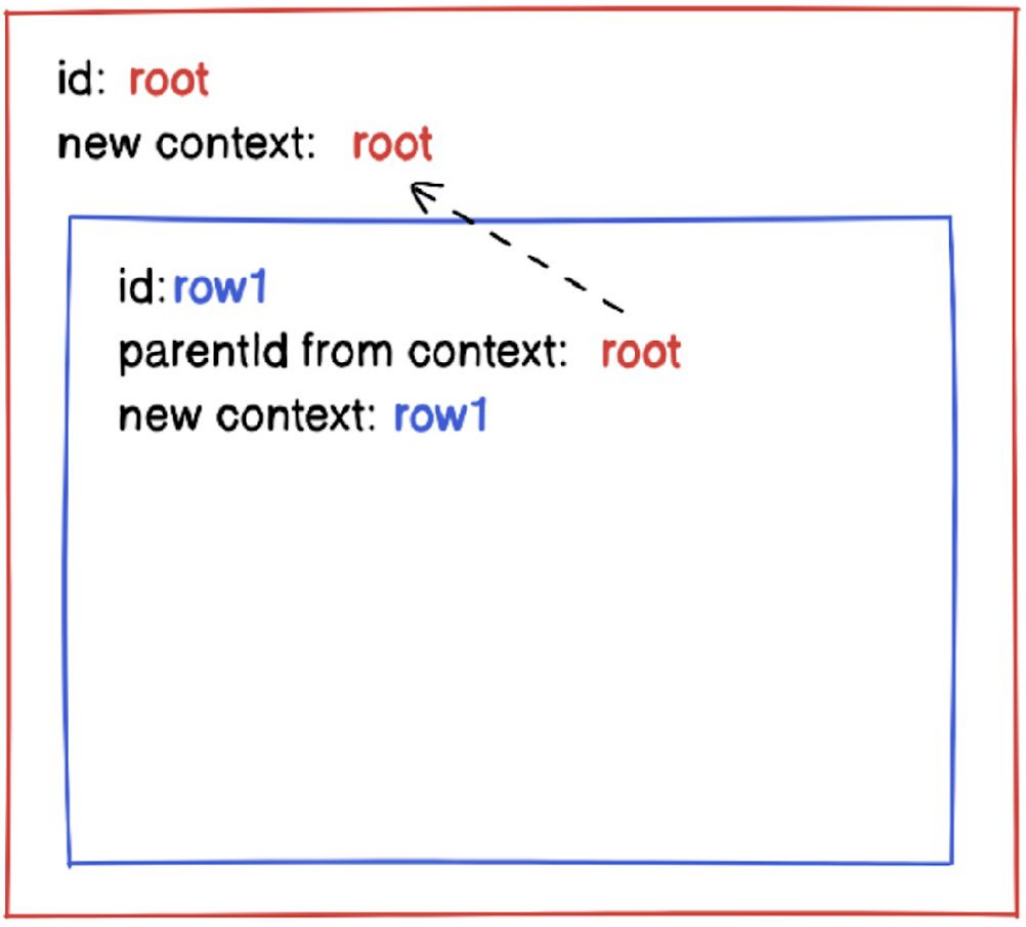
```
export const ParentIdCtx = createContext('root');

export const Example = () ⇒ (
  <>
    I have no parent
    My id: {"root"}
    <ParentIdCtx.Provider value="root">
      </ParentIdCtx.Provider>
  </>
);
```

Basic Context



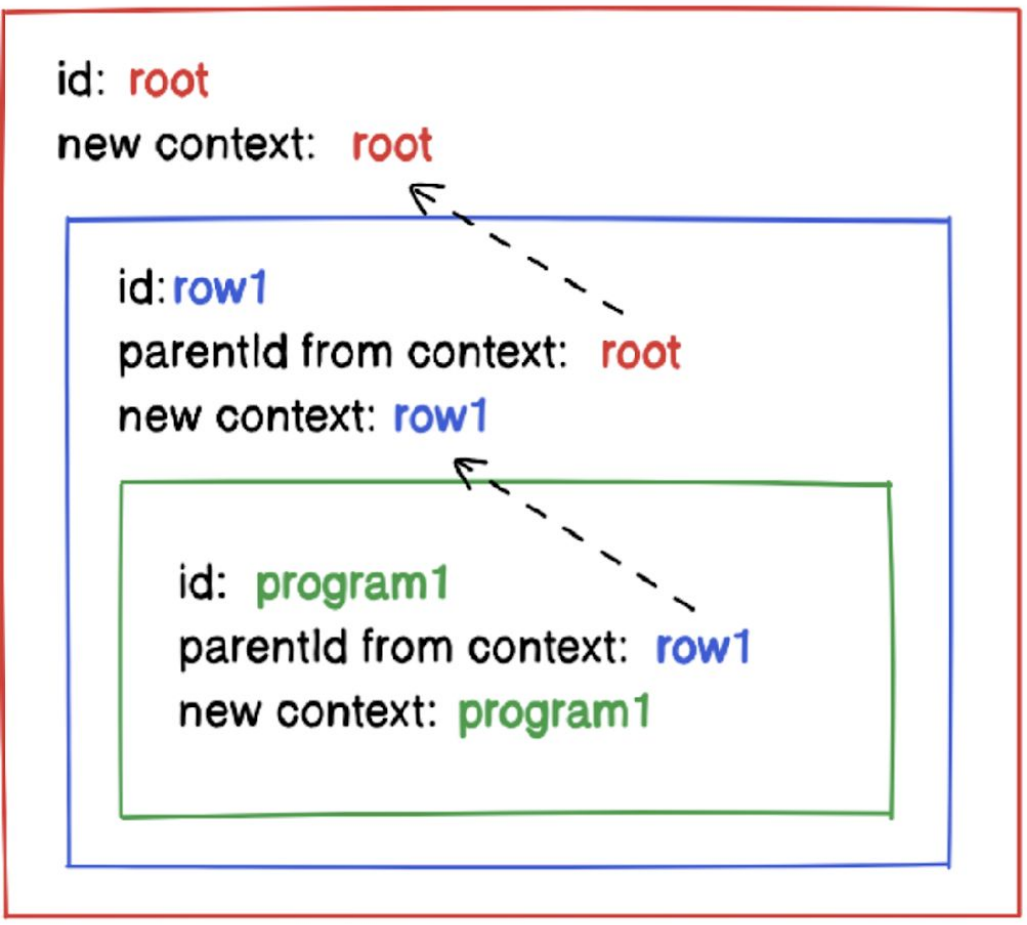

```
export const ParentIdCtx = createContext('root');

export const Example = () ⇒ (
  <>
    I have no parent
    My id: {"root"}
    <ParentIdCtx.Provider value="root">

      <>
        My parent: {useContext(ParentIdCtx) === "root"}
        My id: {"row1"}

      </>

    </ParentIdCtx.Provider>
  </>
);
```

Nested Context




```
export const ParentIdCtx = createContext('root');

export const Example = () ⇒ (
  <>
    I have no parent
    My id: {"root"}
    <ParentIdCtx.Provider value="root">


      <>
        My parent: {useContext(ParentIdCtx) === "root"}
        My id: {"row1"}
        <ParentIdCtx.Provider value="row1">









        </ParentIdCtx.Provider>
      </>

    </ParentIdCtx.Provider>
  </>
);
```

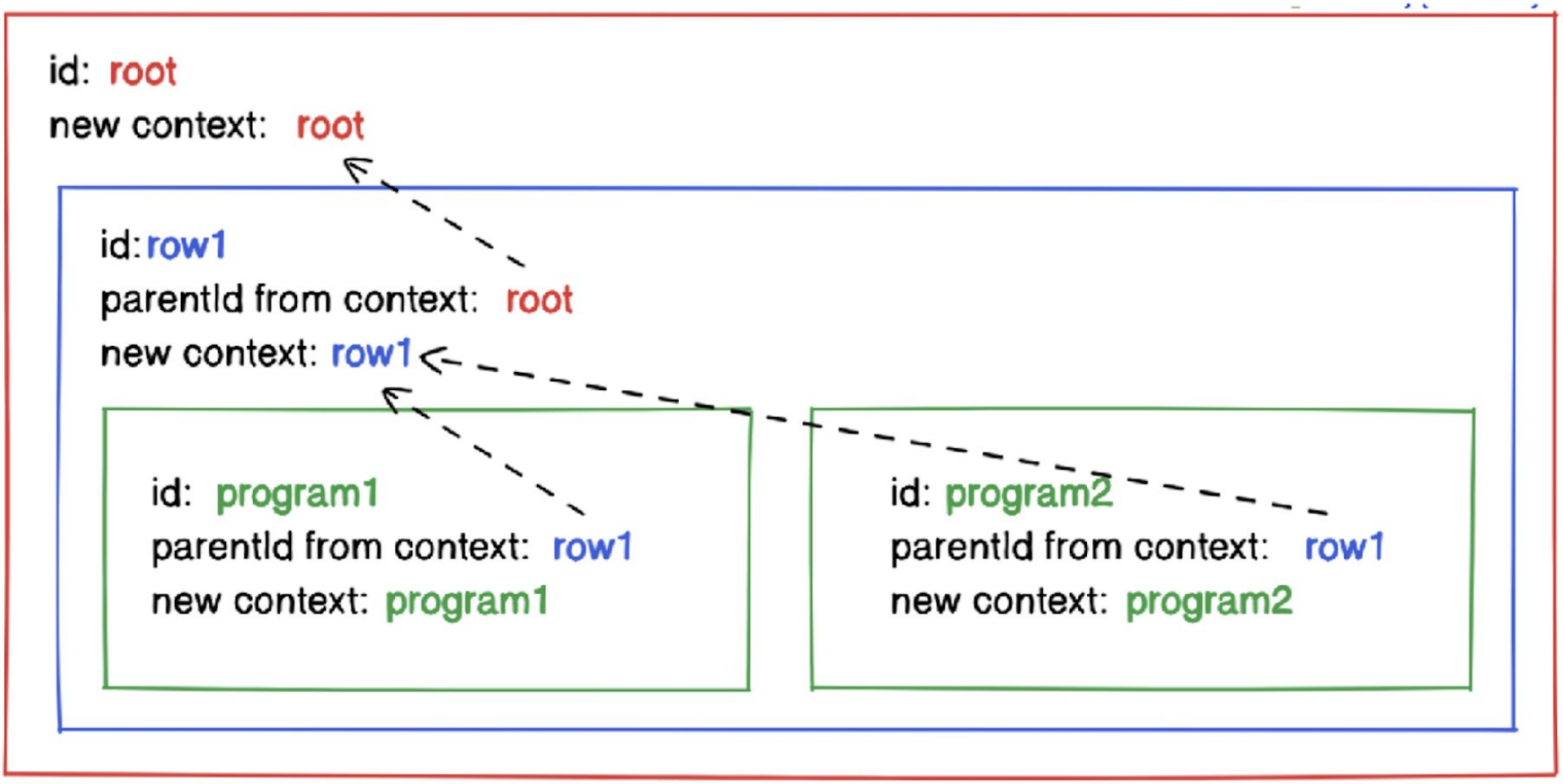
Nested Context




```jsx
export const ParentIdCtx = createContext('root');

export const Example = () => (
  <>
    I have no parent
    My id: {"root"}
    <ParentIdCtx.Provider value="root">

      <>
        My parent: {useContext(ParentIdCtx) === "root"}
        My id: {"row1"}
        <ParentIdCtx.Provider value="row1">

          <>
            My parent: {useContext(ParentIdCtx) === "row1"}
            My id: {"program1"}
          </>

        </ParentIdCtx.Provider>
      </>

    </ParentIdCtx.Provider>
  </>
);
```

Nested Context

```
export const ParentIdCtx = createContext('root');
```




```
="root">

ParentIdCtx) === "root"}

alue="row1">

ext(ParentIdCtx) === "row1"}
```

```
        </ParentIdCtx.Provider>
      </>

    </ParentIdCtx.Provider>
  </>
);
```

Give id to children

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const id = useId();

  useEffect(() => {
    lrud.registerNode(id, {
      parent: '??????????',
      isFocusable,
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children}
    </ParentIdContext.Provider>
  );
};

export const ParentIdContext = createContext('root');
```

Get parentId

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const parentId = useContext(ParentIdContext);
  const id = useId();

  useEffect(() => {
    lrud.registerNode(id, {
      parent: parentId,
      isFocusable,
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children}
    </ParentIdContext.Provider>
  );
};
```

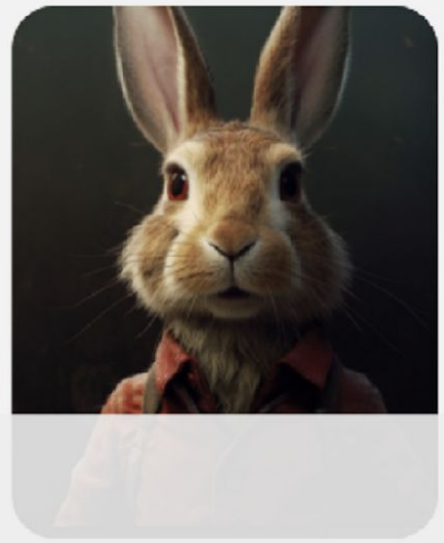
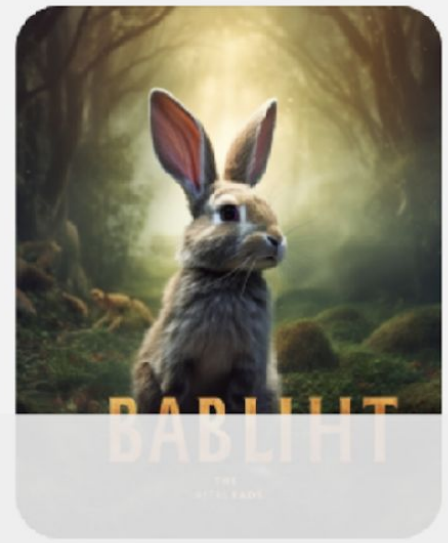
Nothing happens?

Top Rabbit Programs

Rabbit Series

Nothing happens?

Top Rabbit Programs

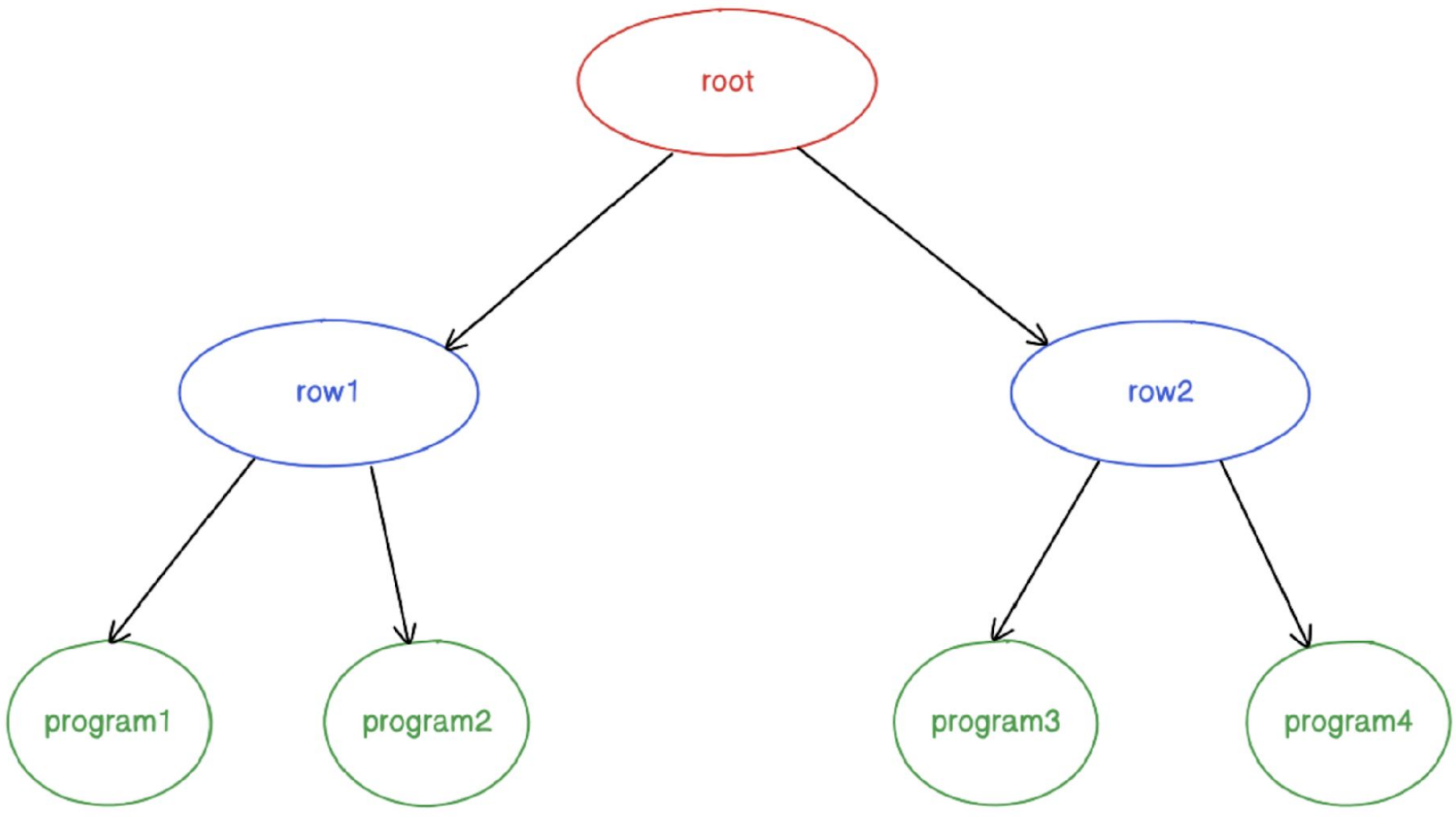
Rabbit Stories

LRUD ERROR: Cannot find "row1"



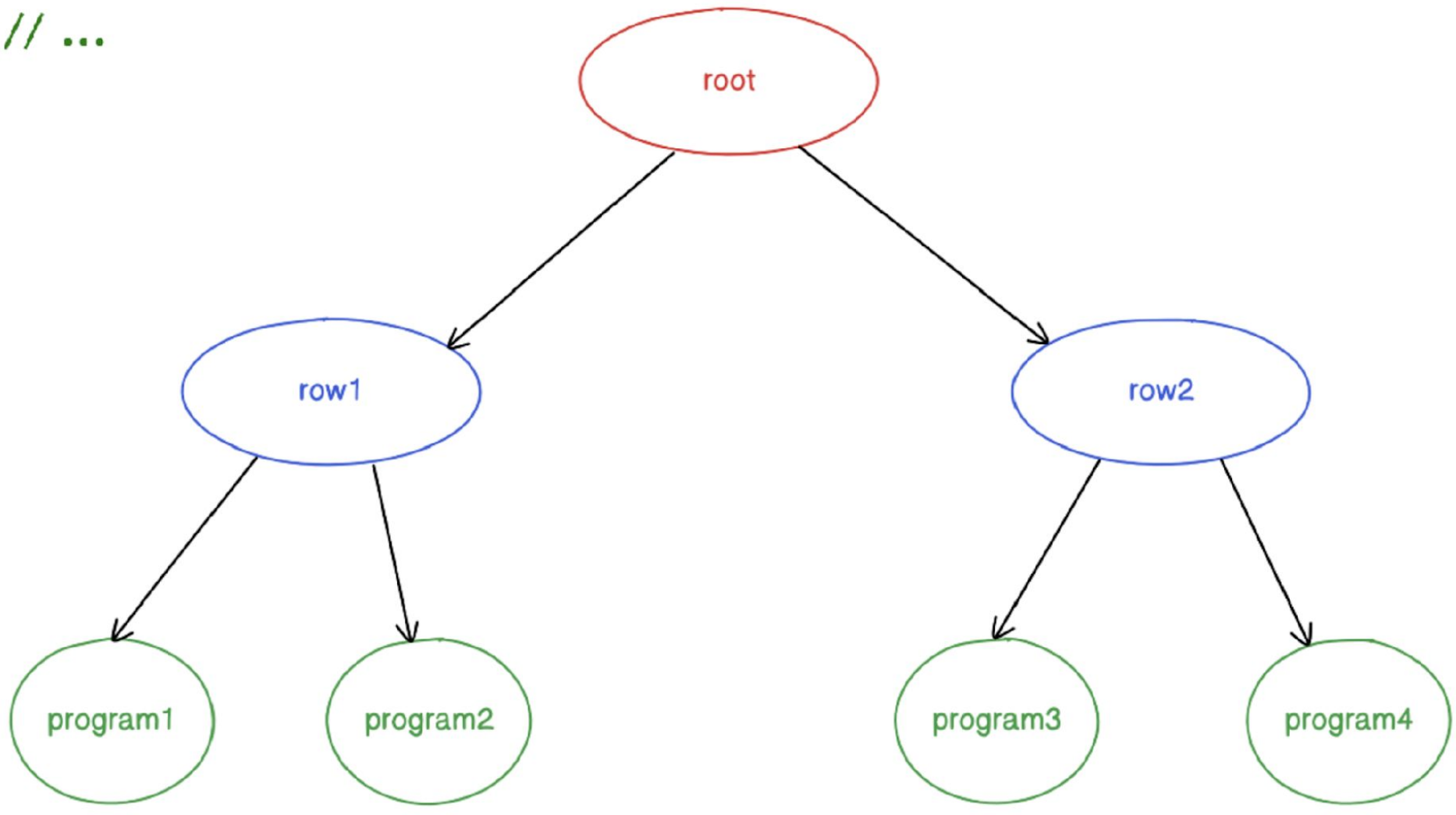

Render order

Render order



Render order

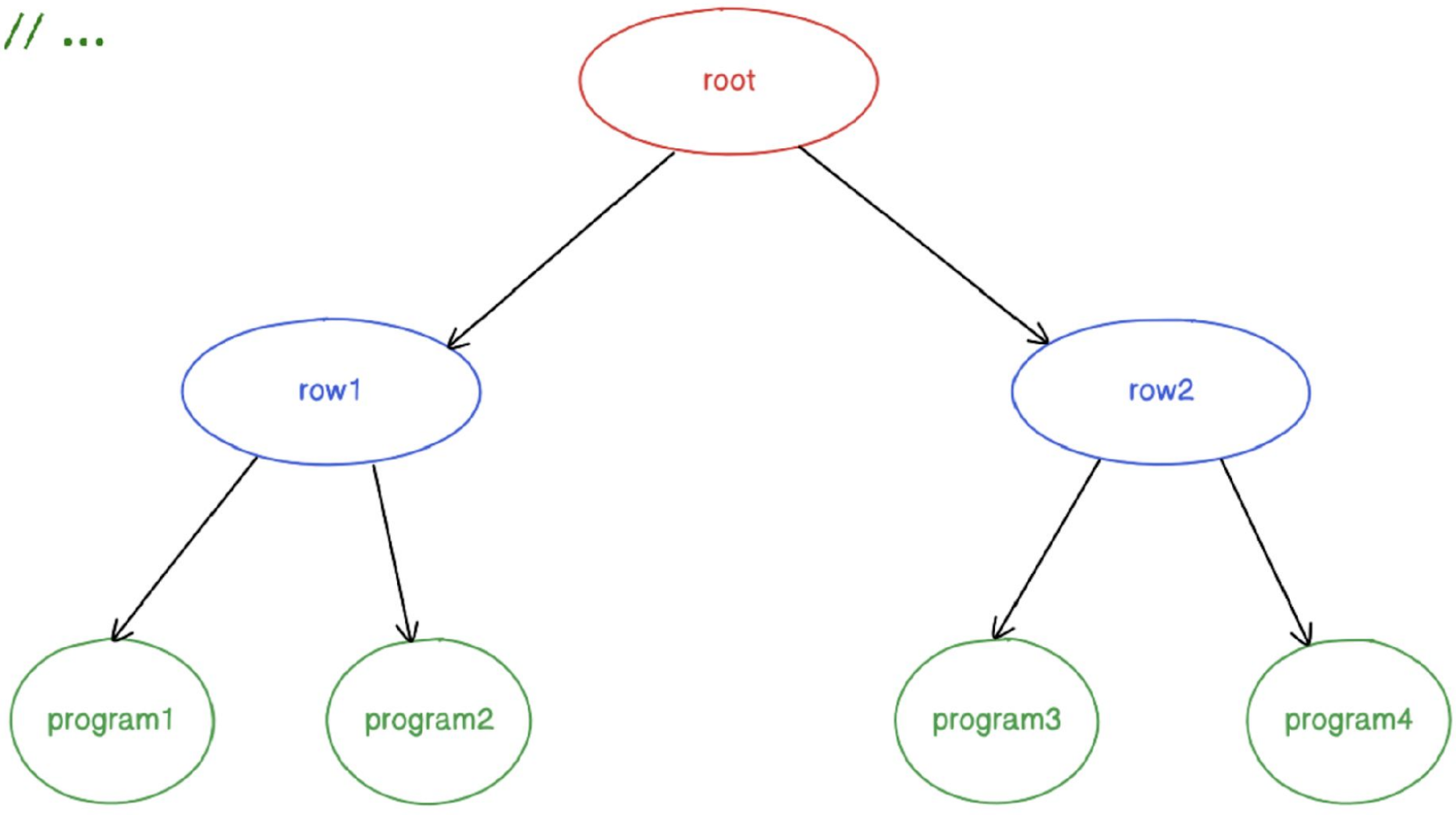
```
const Component = () ⇒ {
  console.log('render');

  return // ...
};
```



Render order

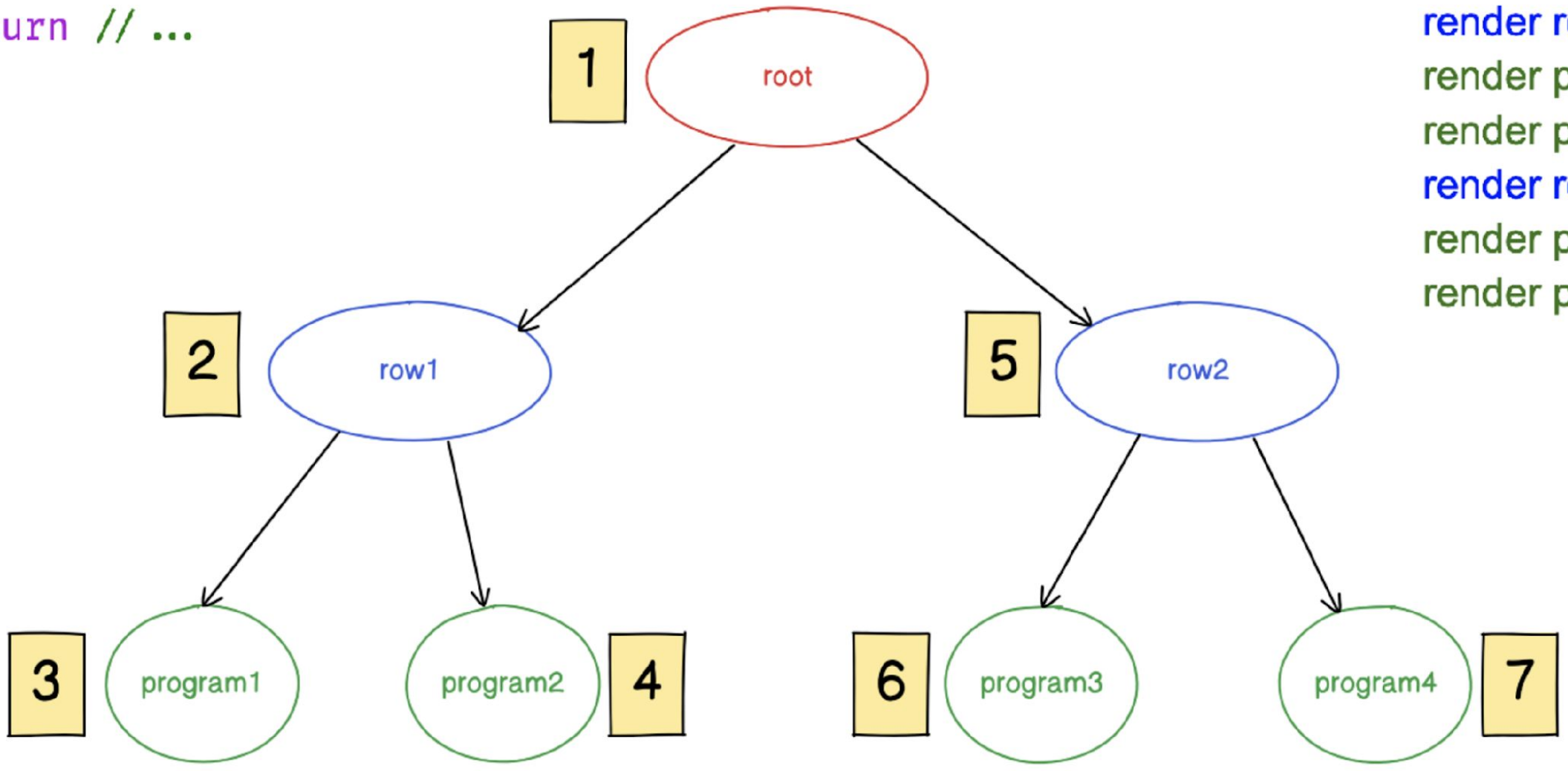
```
const Component = () ⇒ {
  console.log('render');

  return // ...
};
```

Logs order:



Render order

```
const Component = () ⇒ {
  console.log('render');

  return // ...
};
```



Logs order:

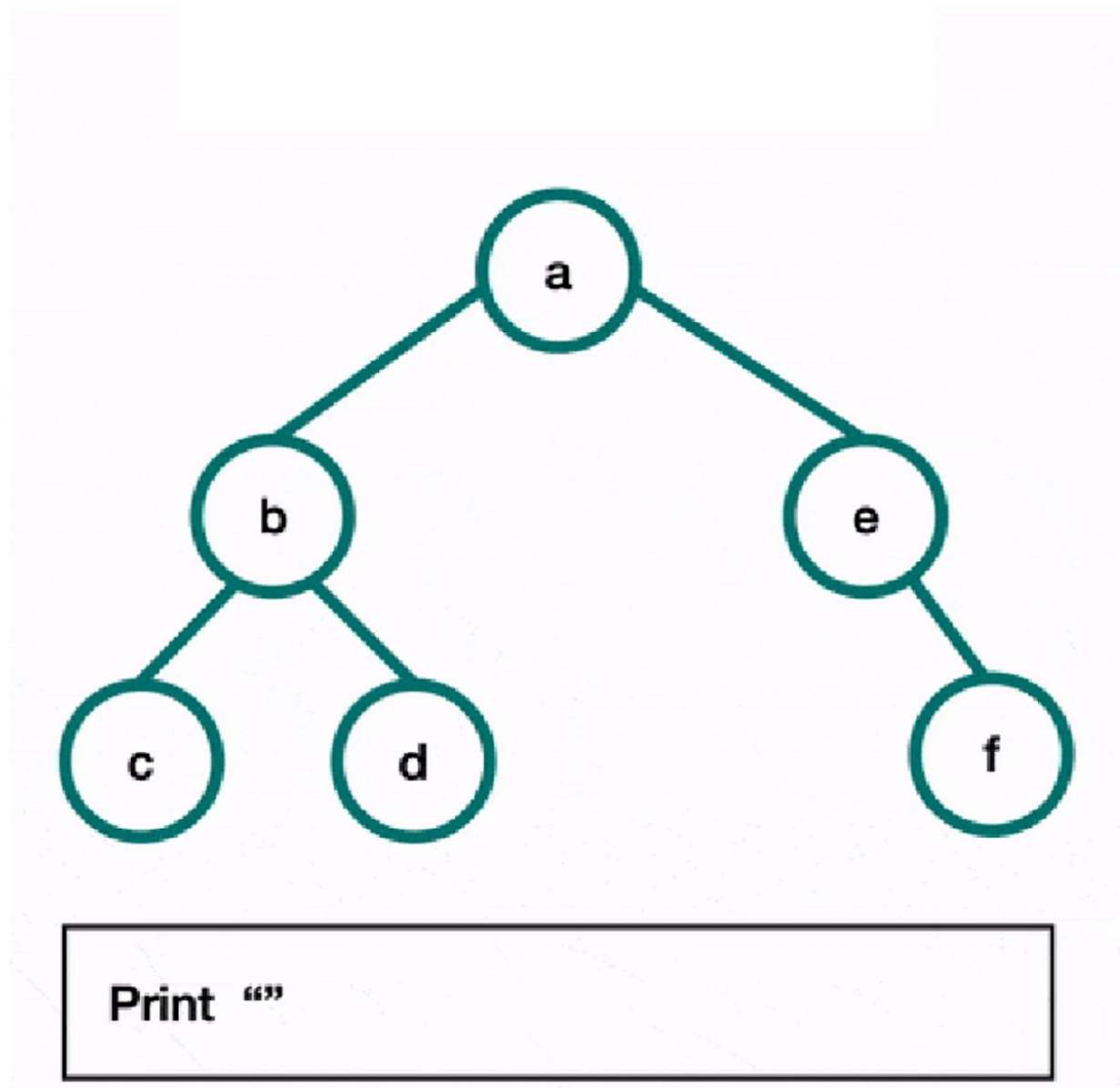
render root
render row1
render program1
render program2
render row2
render program3
render program4

Render order

Tree traversal is

depth-first, **pre**-order (NLR)

Node → left path → right path



Logs order:

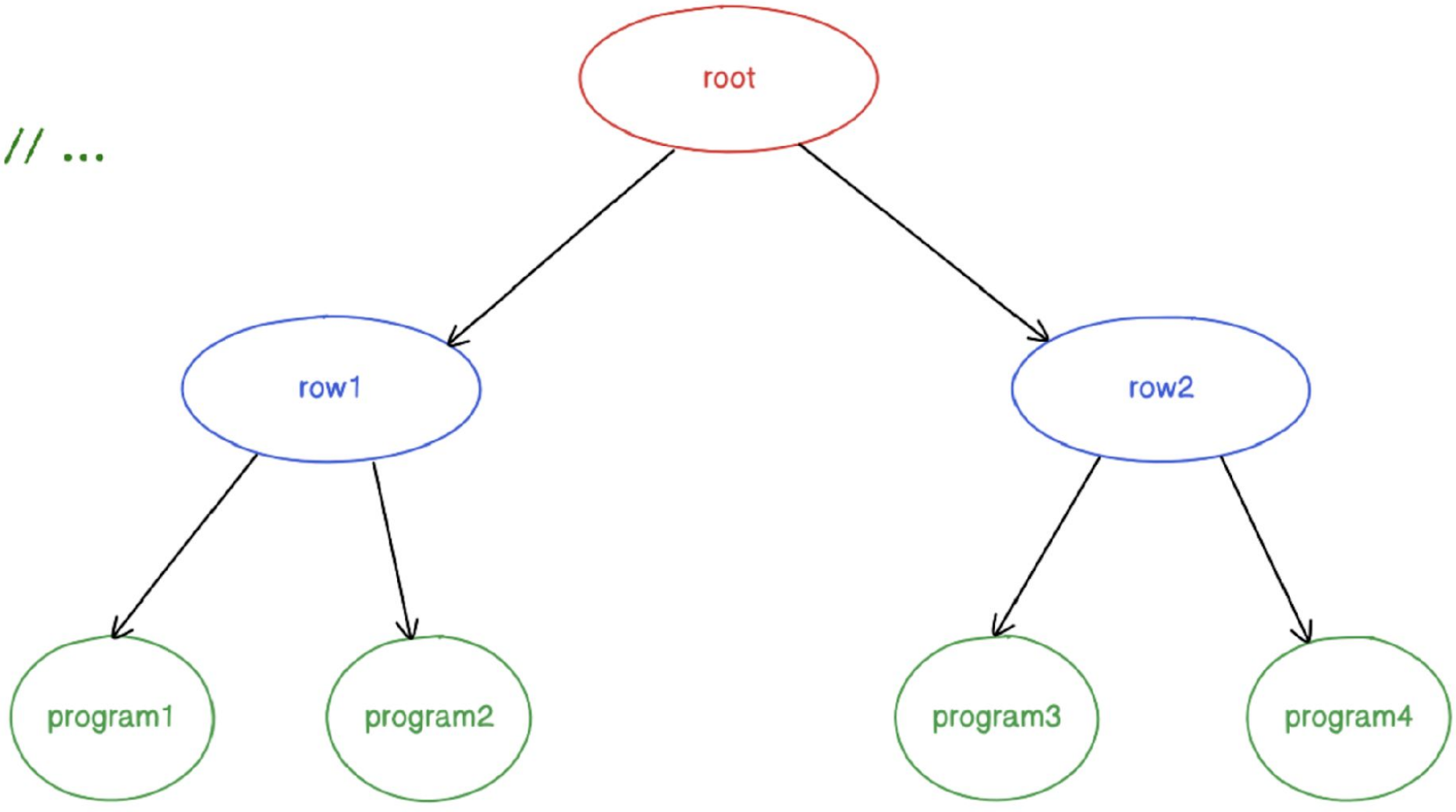
render root
render row1
render program1
render program2
render row2
render program3
render program4

Credit: skilled.dev

useEffect order

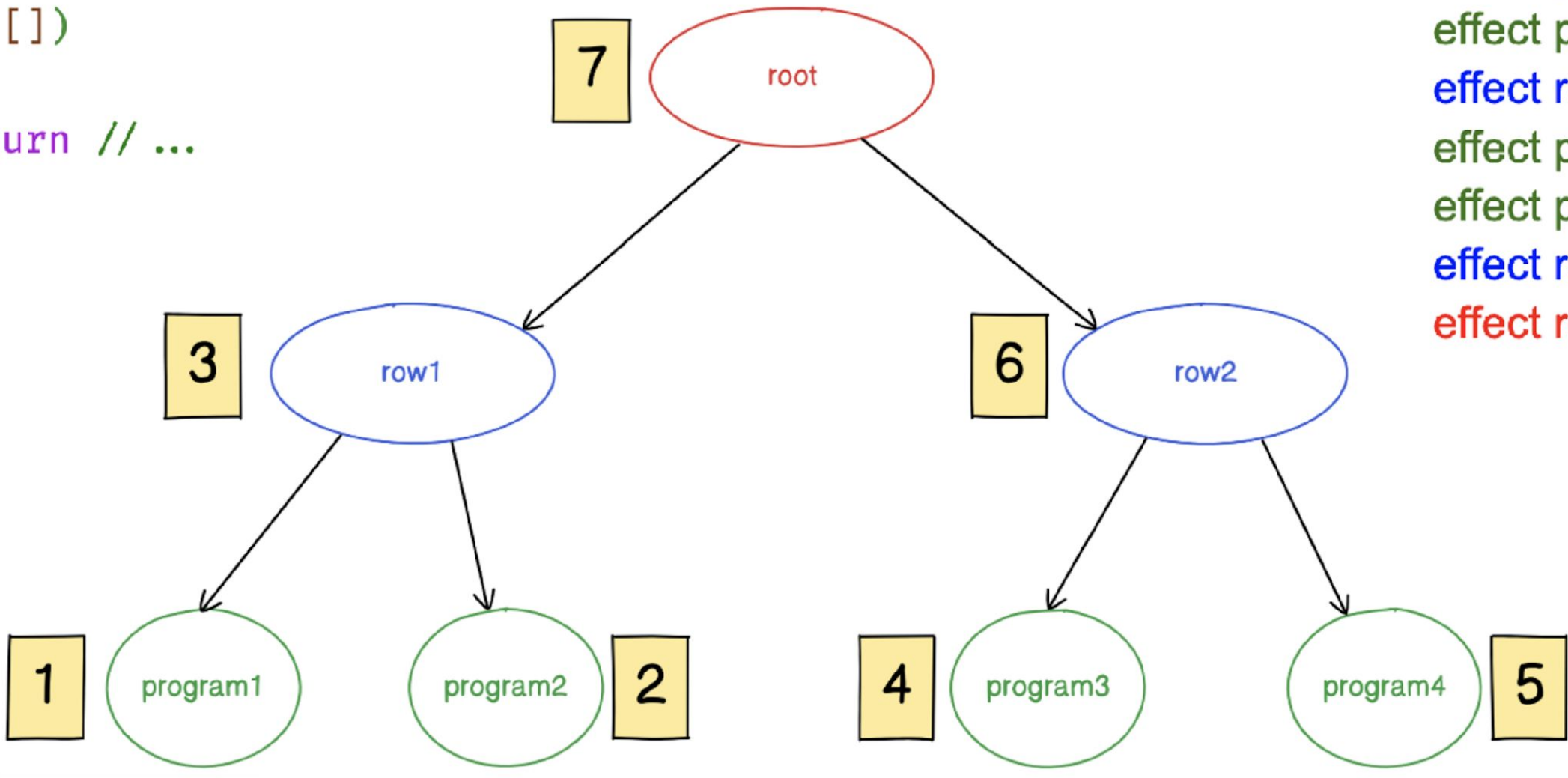
```
const Component = () ⇒ {
  useEffect(() ⇒ {
    console.log('effect');
  }, [])

  return // ...
};
```

Logs order:



useEffect order

```
const Component = () ⇒ {
  useEffect(() ⇒ {
    console.log('effect');
  }, [])

  return // ...
};
```



Logs order:

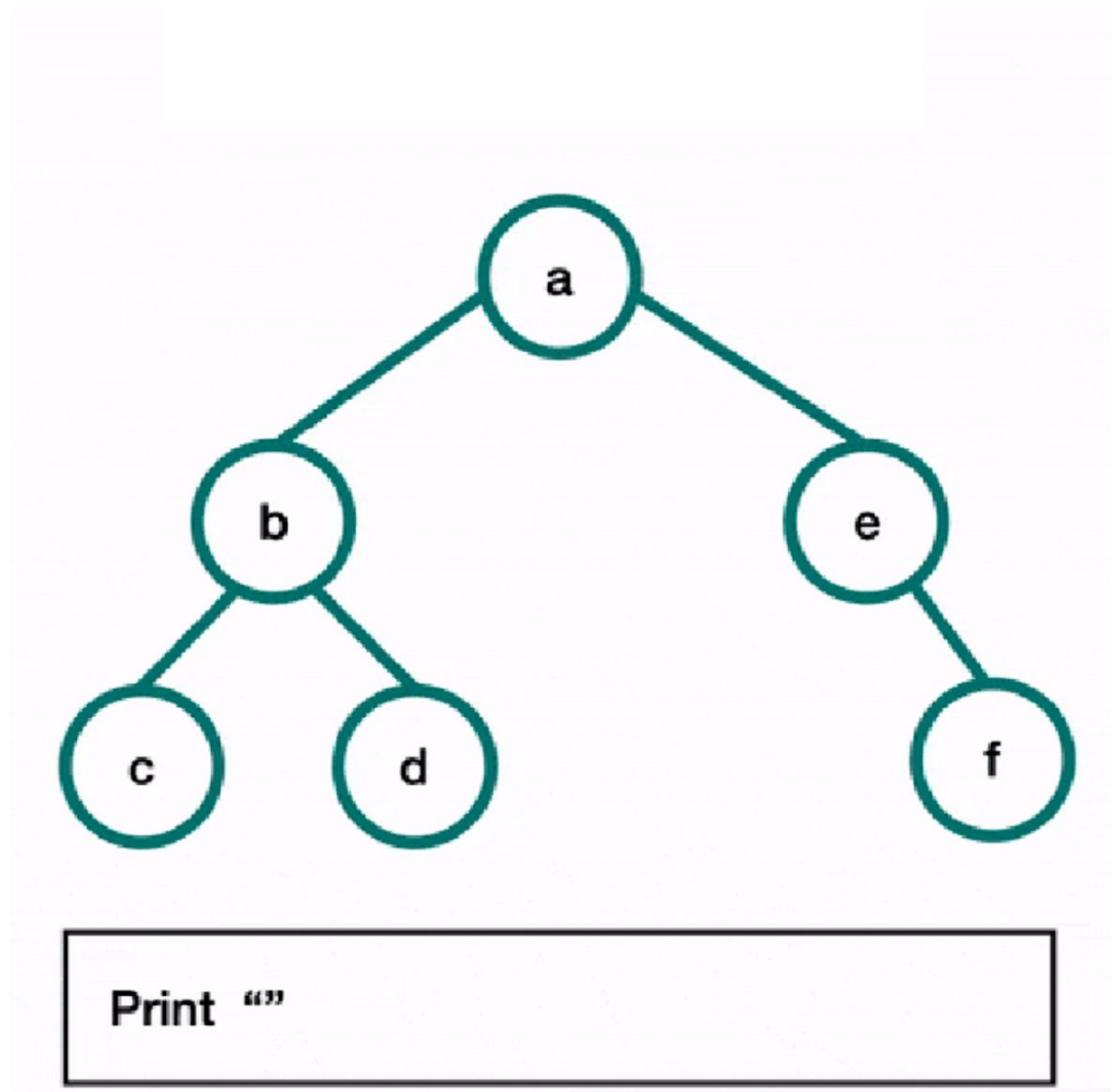
effect program1
effect program2
effect row1
effect program3
effect program4
effect row2
effect root

useEffect order

Tree traversal is

depth-first, **post**-order (LRN)

left path → right path → Node



Logs order:

effect program1
effect program2
effect row1
effect program3
effect program4
effect row2
effect root

Credit: skilled.dev

Recap

Order render

render root
render row1
render program1
render program2
render row2
render program3
render program4

Order useEffect

effect program1
effect program2
effect row1
effect program3
effect program4
effect row2
effect root

A custom useEffect-ish

```
export const useBeforeMountEffect: typeof useEffect = (
  _callback,
  _dependencies,
) ⇒ {
  // Like useEffect, but happens during render
};
```

Basic

```
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  _dependencies,
) => {
  callback();
};
```

Basic

```ts
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  _dependencies,
) => {
  const isMounted = useRef(false);

  if (!isMounted.current) {
    callback();
    isMounted.current = true;
  }
};
```

Basic

```
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  _dependencies,
) => {
  callback();
  isMounted.current = true;
  }
};
```

⚠️ don't try this at home

Dependencies

```
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  dependencies,
) => {
  const isMounted = useRef(false);
  const prevDependencies = useRef<DependencyList>();

  if (!isMounted.current || dependenciesChanged(prevDependencies, dependencies)) {
    callback();
    prevDependencies.current = dependencies;
    isMounted.current = true;
  }
};
```

Destructor

```ts
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  dependencies,
) => {
  const isMounted = useRef(false);
  const prevDependencies = useRef<DependencyList>();
  const prevDestructor = useRef<() => void>();

  if (!isMounted.current || dependenciesChanged(prevDependencies, dependencies)) {
    prevDestructor.current?.(); // Call previous destructor
    prevDestructor.current = callback() || undefined; // Call effect & store destructor
    prevDependencies.current = dependencies;
    isMounted.current = true;
  }
};
```

Unmount

```tsx
export const useBeforeMountEffect: typeof useEffect = (
  callback,
  dependencies,
) => {
  const isMounted = useRef(false);
  const prevDependencies = useRef<DependencyList>();
  const prevDestructor = useRef<() => void>();

  if (!isMounted.current || dependenciesChanged(prevDependencies, dependencies)) {
    prevDestructor.current?.(); // Call previous destructor
    prevDestructor.current = callback() || undefined; // Call effect & store destructor
    prevDependencies.current = dependencies;
    isMounted.current = true;
  }

  useEffect(() => {
    return () => prevDestructor.current?.(); // Call the destructor on unmount too
  }, []);
};
```

Fix registration order

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const parentId = useContext(ParentIdContext);
  const id = useId();

  useBeforeMountEffect(() => {
    lrud.registerNode(id, {
      parent: parentId,
      isFocusable,
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children}
    </ParentIdContext.Provider>
  );
};
```

Nothing happens?

Top Rabbit Programs

Rabbit Series

Focused state

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const parentId = useContext(ParentIdContext);
  const id = useId();

  const [isFocused, setIsFocused] = useState(false);

  useBeforeMountEffect(() => {
    lrud.registerNode(id, {
      parent: parentId,
      isFocusable,
      onBlur: () => setIsFocused(false),
      onFocus: () => setIsFocused(true),
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children}
    </ParentIdContext.Provider>
  );
};
```

Pass focused state

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const parentId = useContext(ParentIdContext);
  const id = useId();

  const [isFocused, setIsFocused] = useState(false);

  useBeforeMountEffect(() => {
    lrud.registerNode(id, {
      parent: parentId,
      isFocusable,
      onBlur: () => setIsFocused(false),
      onFocus: () => setIsFocused(true),
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children({ isFocused })}
    </ParentIdContext.Provider>
  );
};
```

Pass focused state

```tsx
export const Node = ({
  isFocusable = false,
  orientation = 'vertical',
  children,
}: Props) => {
  const parentId = useContext(ParentIdContext);
  const id = useId();

  const [isFocused, setIsFocused] = useState(false);

  useBeforeMountEffect(() => {
    lrud.registerNode(id, {
      parent: parentId,
      isFocusable,
      onBlur: () => setIsFocused(false),
      onFocus: () => setIsFocused(true),
      orientation,
    });

    return () => lrud.unregisterNode(id);
  }, []);

  return (
    <ParentIdContext.Provider value={id}>
      {children({ isFocused })}
    </ParentIdContext.Provider>
  );
};
```

```tsx
const Program = () => (
  <Node isFocusable>
    {({ isFocused }) => (
      <ProgramLayout isFocused={isFocused} />
    )}
  </Node>
);

export const Page = () => (
  <>
    <Typography>Top Rabbit Programs</Typography>
    <Node orientation="horizontal">
      <View style={{ flexDirection: 'row' }}>
        <Program />
        <Program />
      </View>
    </Node>
    <Typography>Rabbit Series</Typography>
    <Node orientation="horizontal">
      <View style={{ flexDirection: 'row' }}>
        <Program />
        <Program />
      </View>
    </Node>
  </>
);
```

*That's not it

We had many other challenges to tackle

- Handling the scroll
- Handle pages on top of each other
- Side menu that is common between pages
- Navigation in virtualized lists
- Refreshing virtualized lists with new data
- Handling the default focus

The React-way = ❤️

Adopting a React-friendly declarative API was such a huge productivity boost

Thank you!

Check out https://blog.bam.tech !