

Let's get lazy with React

A look into React newer features



/ricardani/react-suspense-demo

react@experimental
react-dom@experimental

Index

Strict Mode

Error Boundaries

Code Splitting

Index

Strict Mode

Error Boundaries

Code Splitting

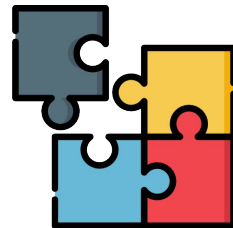
Suspense for Data Fetching

Concurrent Mode

Index



Strict Mode
Error Boundaries
Code Splitting



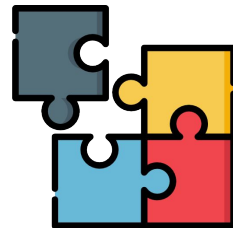
Suspense for Data Fetching

Concurrent Mode

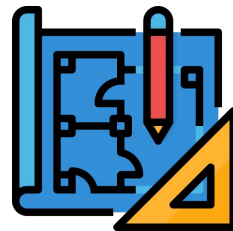
Index



Strict Mode
Error Boundaries
Code Splitting



Suspense for Data Fetching
Concurrent Mode



React Hooks

React Hooks

```
const [isLoading, setIsLoading] = useState(false);
```

<https://reactjs.org/docs/hooks-state.html>

React Hooks

Current State

Default Value

```
const [isLoading, setIsLoading] = useState(false);
```

Update Function

The diagram illustrates the components of the `useState` hook call. The text `const [isLoading, setIsLoading] = useState(false);` is shown. The `isLoading` and `setIsLoading` are grouped in a blue box, with an arrow from 'Current State' pointing to it. The `useState` function is in purple, and the `false` argument is in orange and boxed in blue, with an arrow from 'Default Value' pointing to it. An arrow from 'Update Function' points to the `setIsLoading` function.

React Hooks

```
useEffect(() => { ...some code }, []);
```

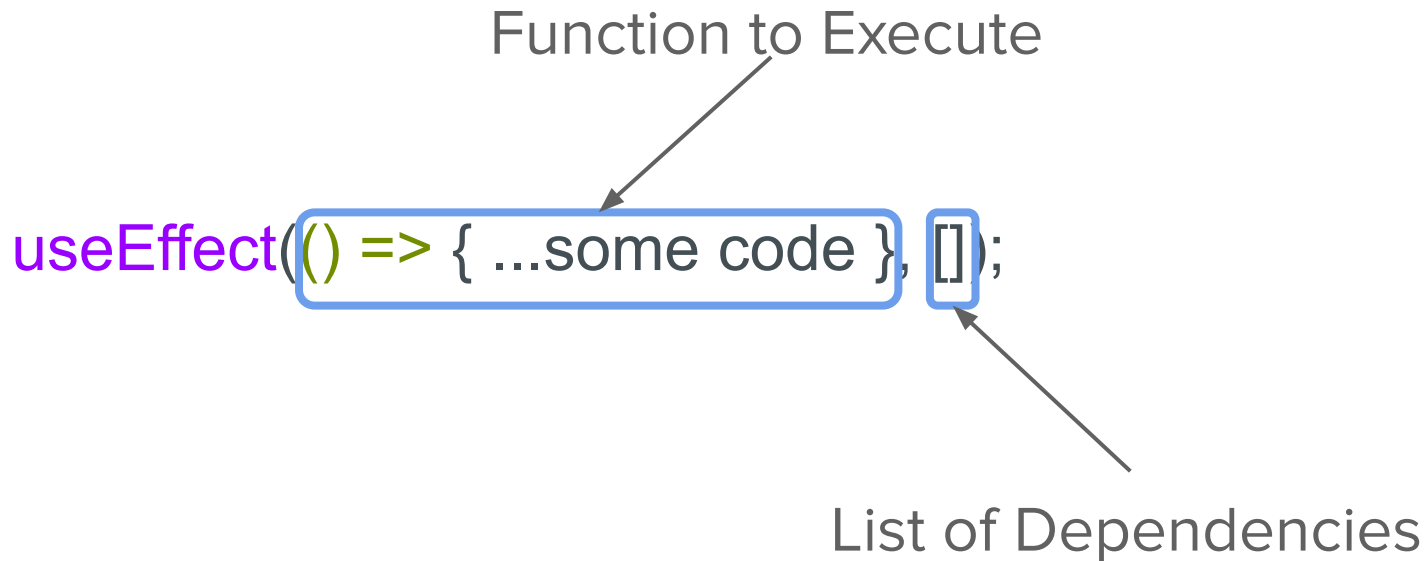
<https://reactjs.org/docs/hooks-effect.html>

React Hooks

Function to Execute

```
useEffect(() => { ...some code }, []);
```

List of Dependencies

The diagram shows the `useEffect` hook signature. The text `useEffect` is in purple. The function `() => { ...some code }` is enclosed in a blue rounded rectangle, with an arrow pointing from the text "Function to Execute" above it. The dependency array `[]` is also enclosed in a blue rounded rectangle, with an arrow pointing from the text "List of Dependencies" below it.

<https://reactjs.org/docs/hooks-effect.html>

Demo Project

Strict Mode

Strict Mode

Highlights Potential Problems

Unsafe Lifecycles

Legacy and Deprecated Code

Strict Mode

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



The diagram illustrates how a standard React element is wrapped in Strict Mode. An arrow points from the `<App />` in the code above to the `<App />` in the code below. The `<React.StrictMode>` and `</ React.StrictMode>` are enclosed in blue rounded rectangles.

```
<React.StrictMode> <App /> </ React.StrictMode>
```

Error Boundaries

Error Boundaries

What happens when an error occurs
inside of a component ?

<https://reactjs.org/docs/error-boundaries.html>

Error Boundaries

// These functions are called when an error has been thrown in a child component

```
static getDerivedStateFromError(error) {
```

```
  // Returns a new state
```

```
}
```

```
componentDidCatch(error, errorInfo) {
```

```
  // Used to log more information about the error
```

```
}
```

<https://reactjs.org/docs/error-boundaries.html>

Error Boundaries

// These functions are called when an error has been thrown in a child component

```
static getDerivedStateFromError(error) {
```

```
  // Returns a new state
```

```
}
```

```
componentDidCatch(error, errorInfo) {
```

```
  // Used to log more information about the error
```

```
}
```

<https://reactjs.org/docs/error-boundaries.html>

No Hook Version

Error Boundaries

// These functions are called when an error has been thrown in a child component

```
static getDerivedStateFromError(error) {
```

```
  // Returns a new state
```

```
}
```

```
componentDidCatch(error, errorInfo) {
```

```
  // Used to log more information about the error
```

```
}
```

<https://reactjs.org/docs/error-boundaries.html>

No Hook Version
Yet

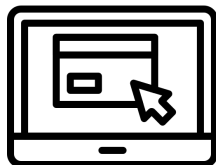
Code Splitting

Code Splitting

Why should we use code splitting?

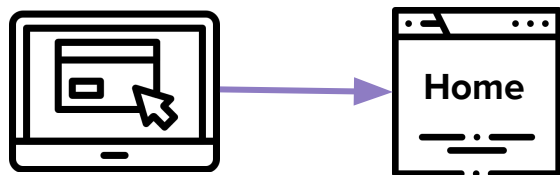
<https://reactjs.org/docs/code-splitting.html>

Code Splitting



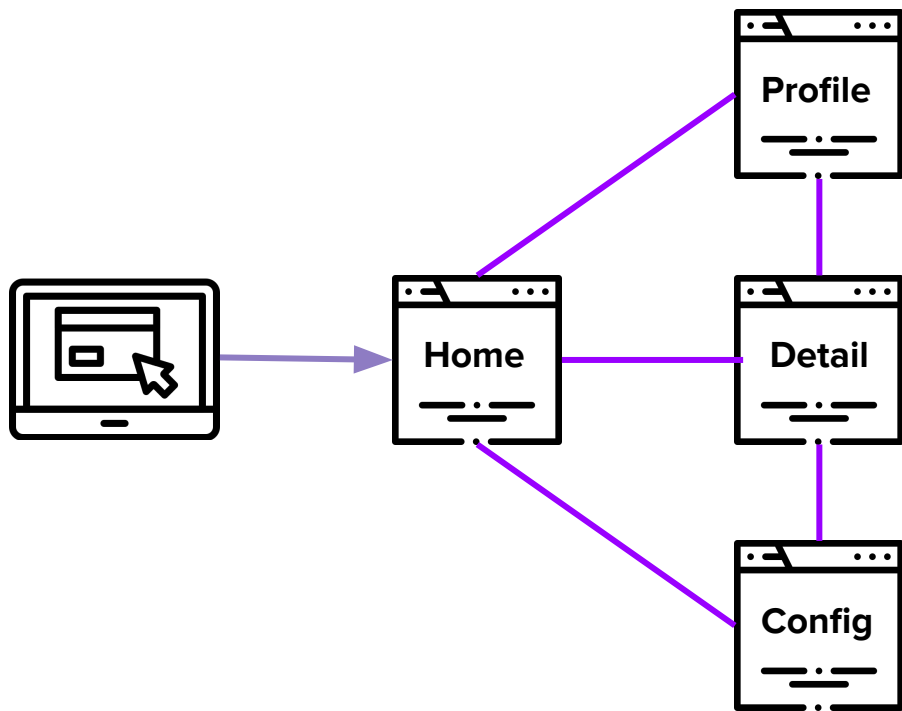
<https://reactjs.org/docs/code-splitting.html>

Code Splitting



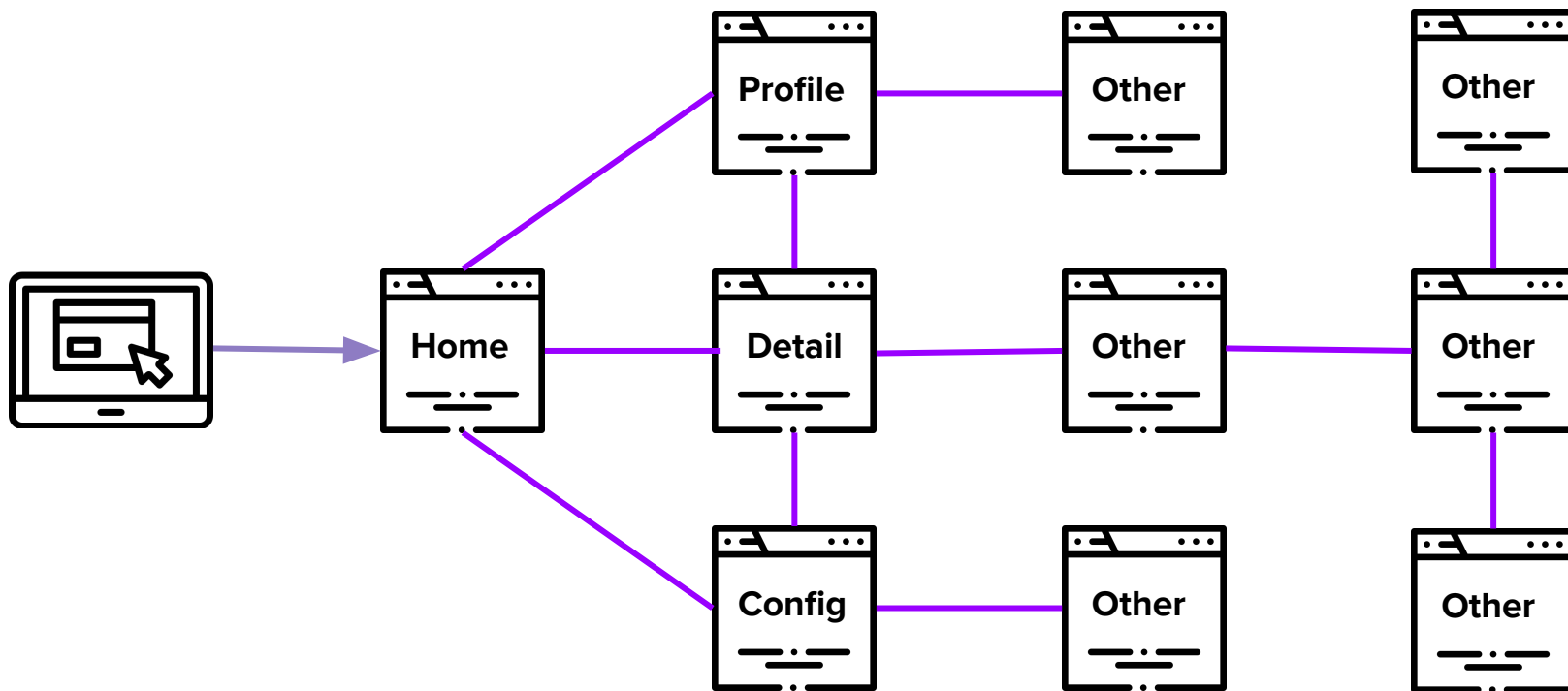
<https://reactjs.org/docs/code-splitting.html>

Code Splitting



<https://reactjs.org/docs/code-splitting.html>

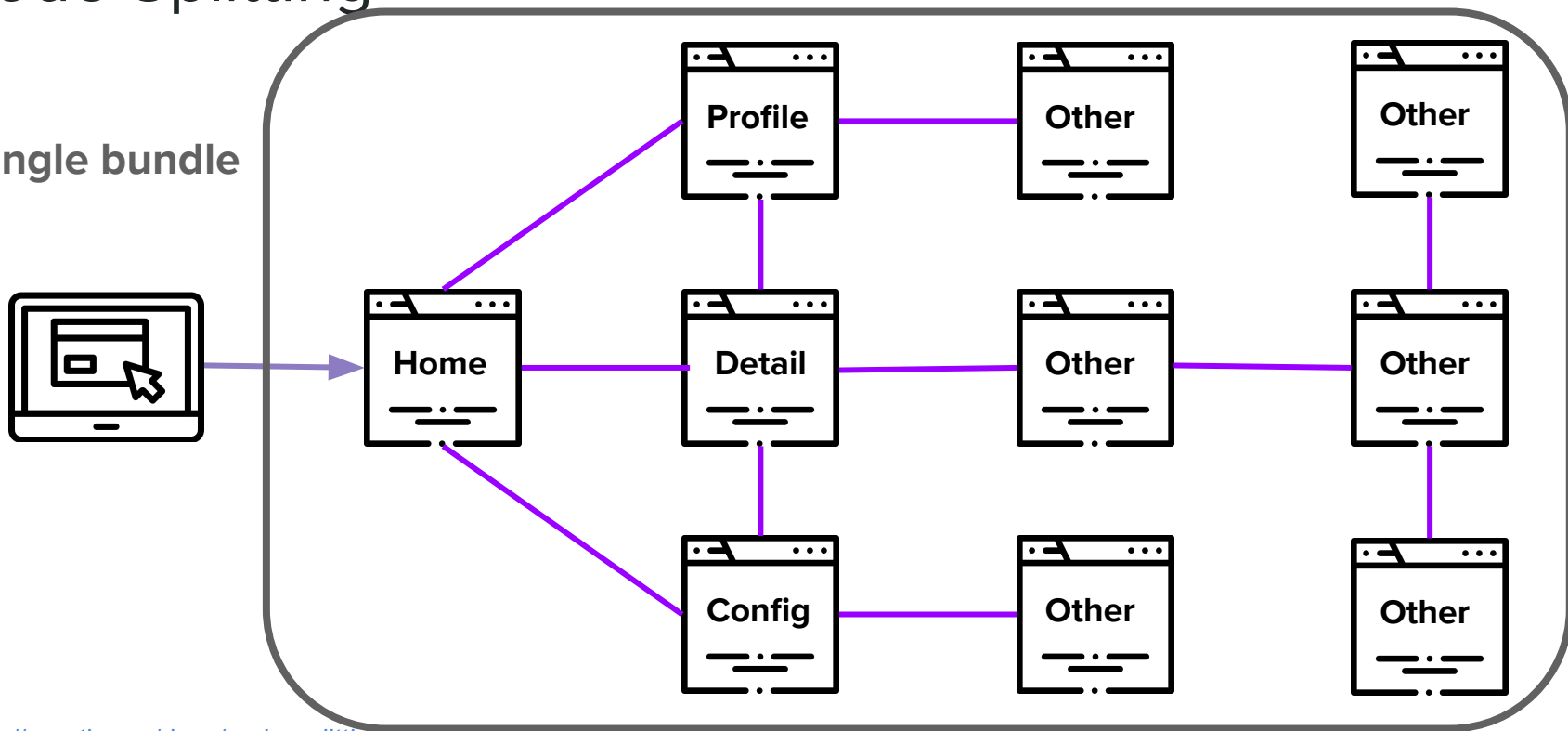
Code Splitting



<https://reactjs.org/docs/code-splitting.html>

Code Splitting

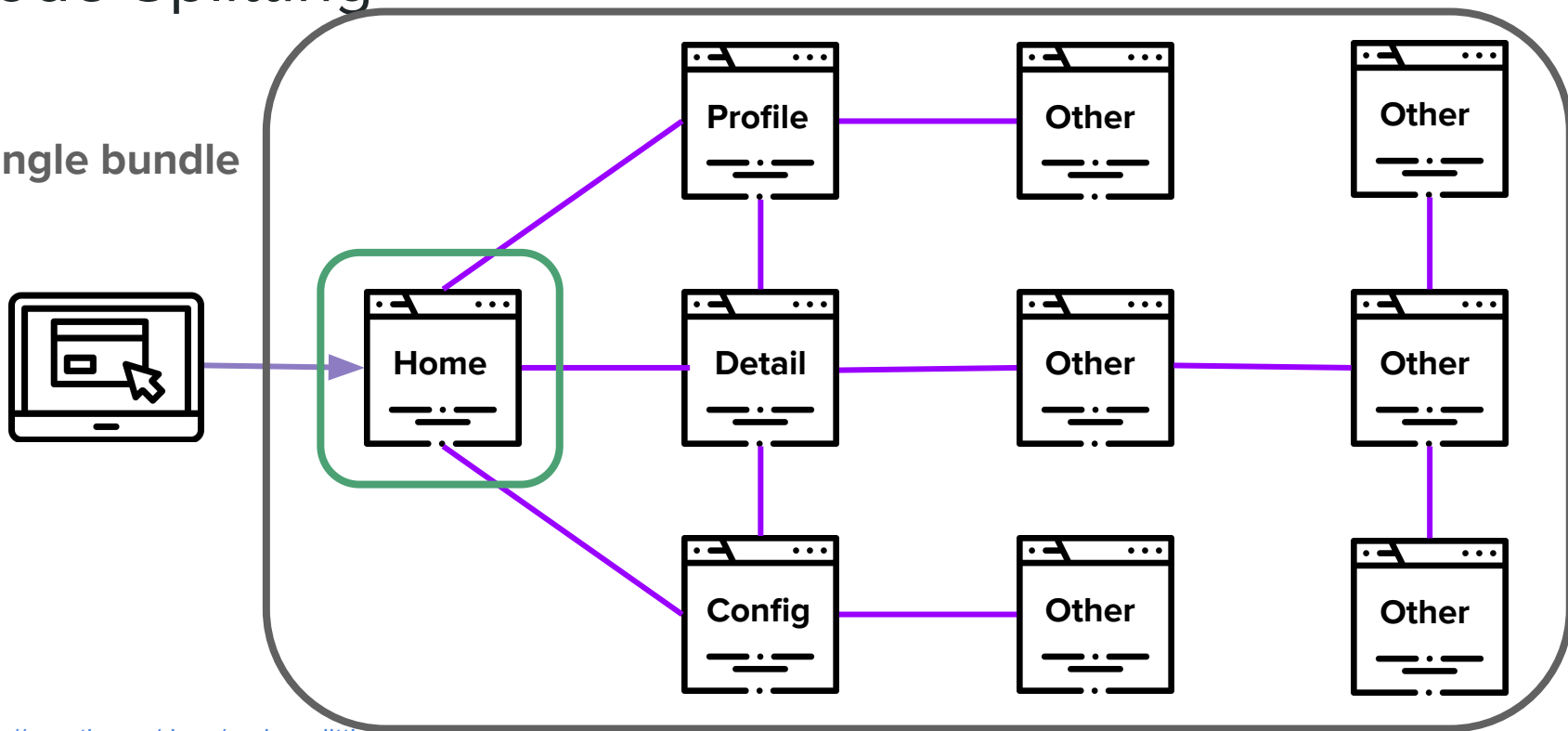
Single bundle



<https://reactjs.org/docs/code-splitting.html>

Code Splitting

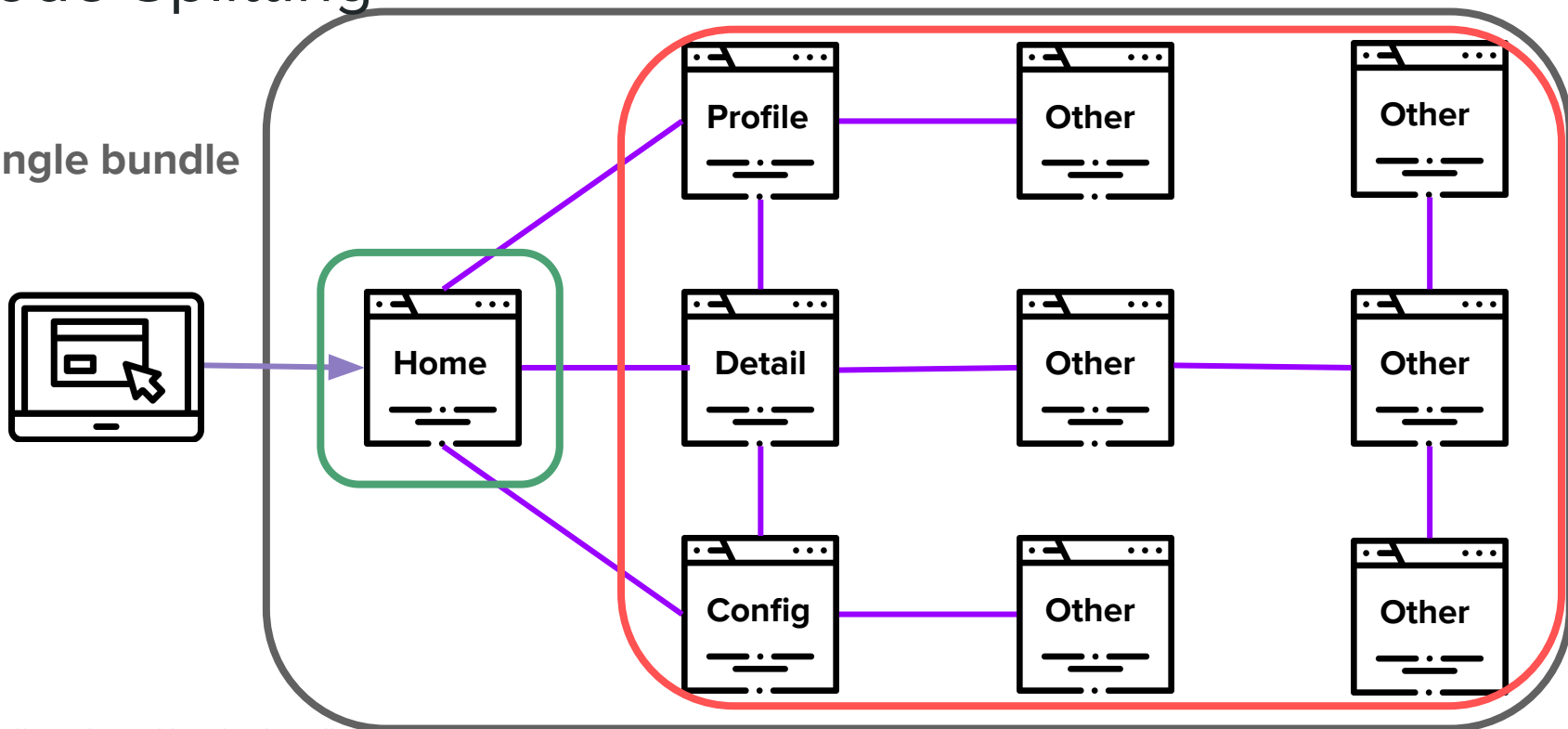
Single bundle



<https://reactjs.org/docs/code-splitting.html>

Code Splitting

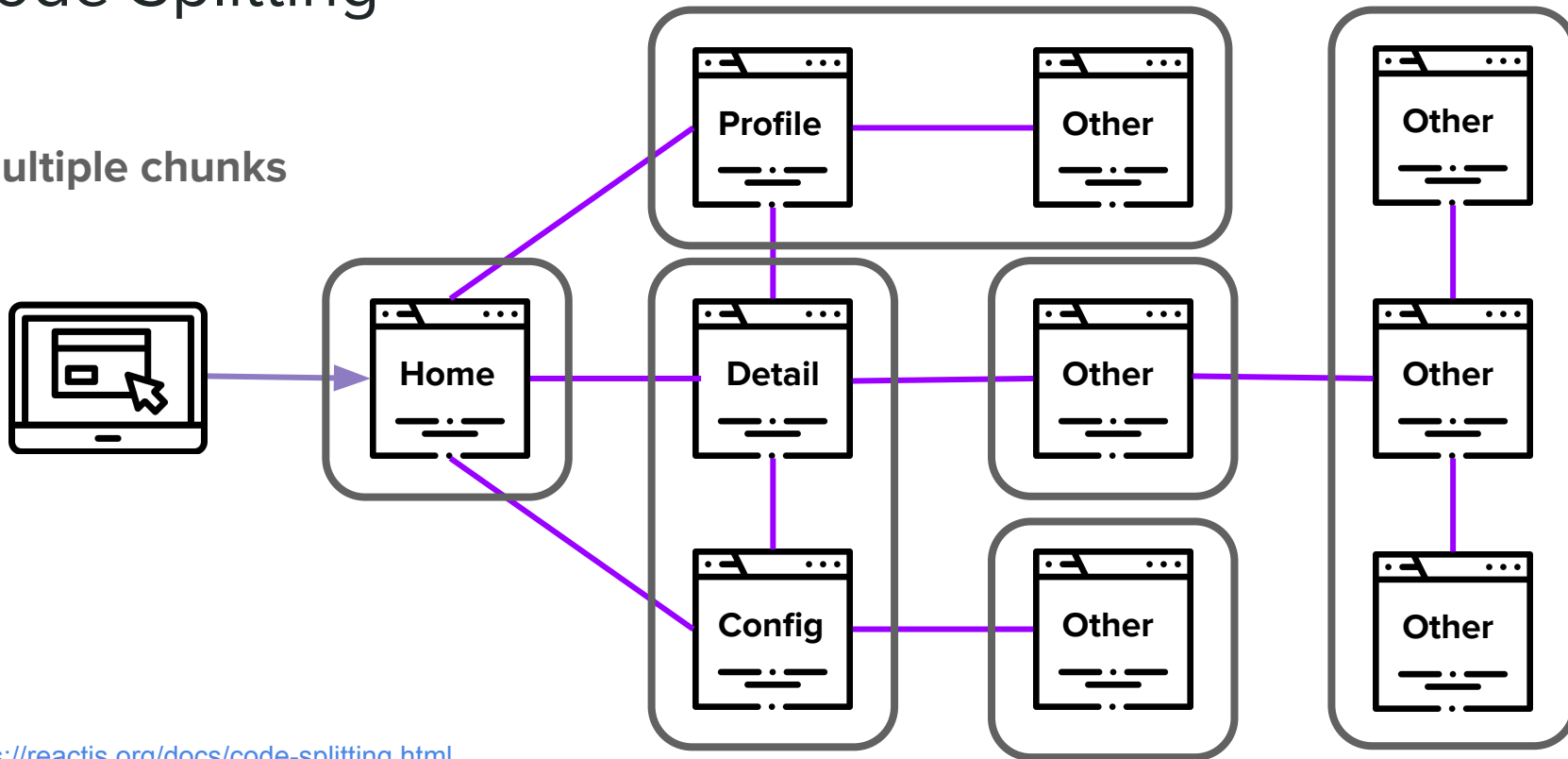
Single bundle



<https://reactjs.org/docs/code-splitting.html>

Code Splitting

Multiple chunks



<https://reactjs.org/docs/code-splitting.html>

Code Splitting

How do you do it?

<https://reactjs.org/docs/code-splitting.html>

Code Splitting

Cons

<https://reactjs.org/docs/code-splitting.html>

Code Splitting

Cons

- Handling lazy components
- A request for each chunk
- More Loading screens

Code Splitting

Cons

- Handling lazy components
- A request for each chunk
- More loading screens

Pros

<https://reactjs.org/docs/code-splitting.html>

Code Splitting

Cons

- Handling lazy components
- A request for each chunk
- More loading screens

Pros

- Reduced bundle size
- Faster first page impression



Suspense for Data Fetching

Suspense for Data Fetching



Why not use Suspense for data fetching?

<https://reactjs.org/docs/concurrent-mode-suspense.html>



Concurrent Mode

Concurrent Mode

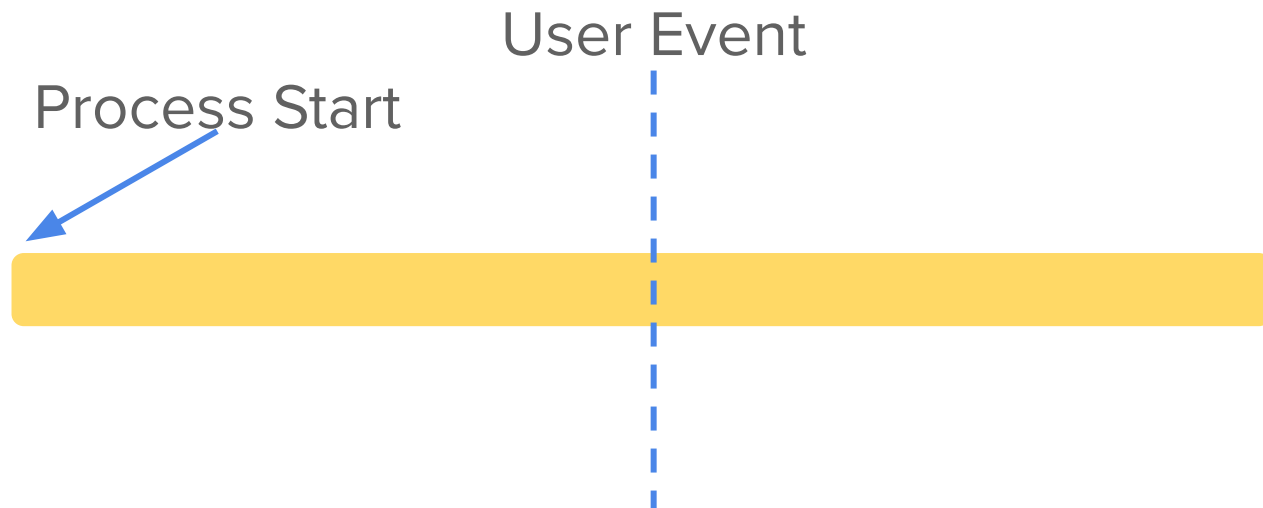


Process Start



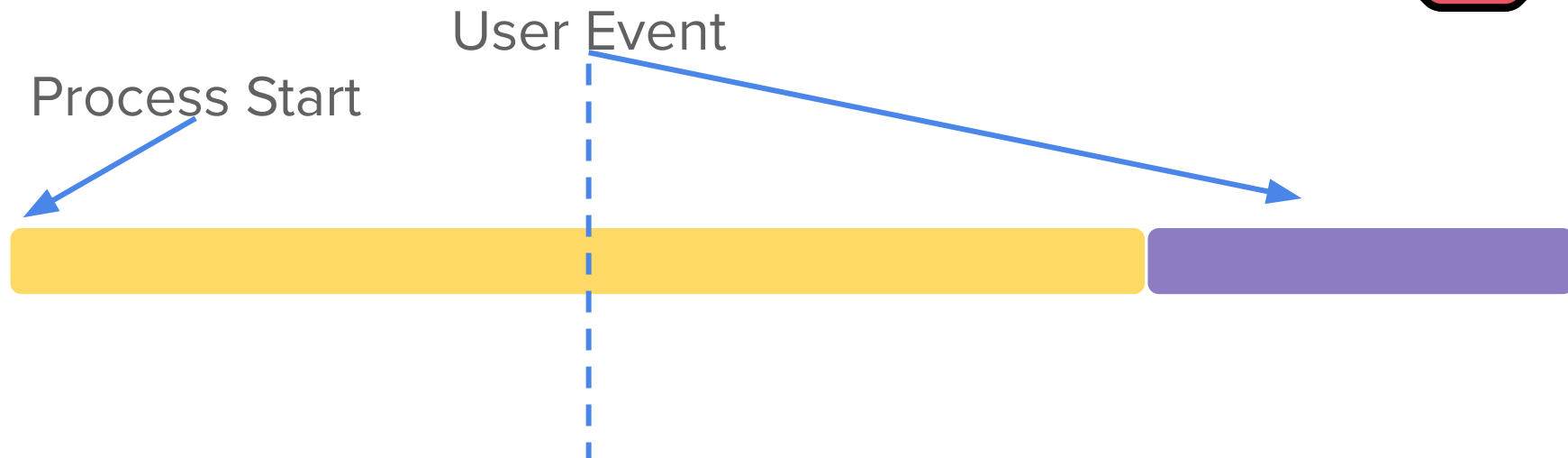
<https://reactjs.org/docs/concurrent-mode-intro.html>

Concurrent Mode

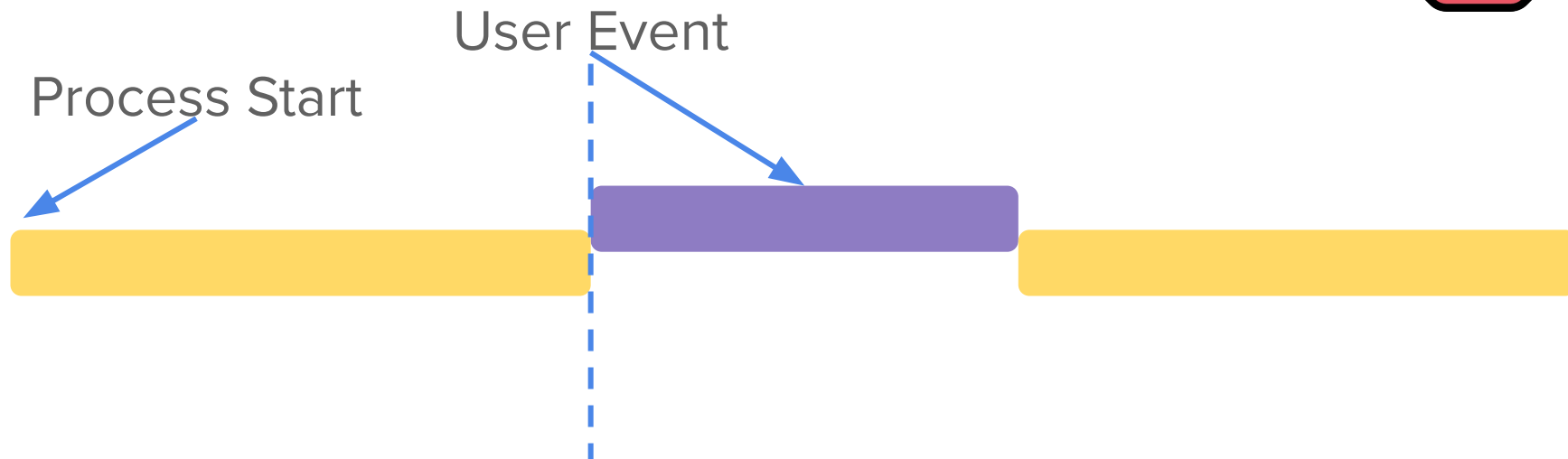


<https://reactjs.org/docs/concurrent-mode-intro.html>

Concurrent Mode



Concurrent Mode



<https://reactjs.org/docs/concurrent-mode-intro.html>

Concurrent Mode



Can work on multiple tasks and switch between them

Can partially render a tree without committing the result

Does not block the main thread

<https://reactjs.org/docs/concurrent-mode-intro.html>

Concurrent Mode



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



Enable Concurrent Mode

```
ReactDOM.createRoot( document.getElementById('root') ).render( <App /> )
```

Concurrent Mode



SuspenseList

useTransition

useDeferredValue

Let's get lazy with React

A look into React newer features



/ricardani/react-suspense-demo