¿Cuál es más rápido?

- `for`
- `for (in ... of)`
- `Array.forEach`
- `Array.map`
- `Array.reduce`

Curso de

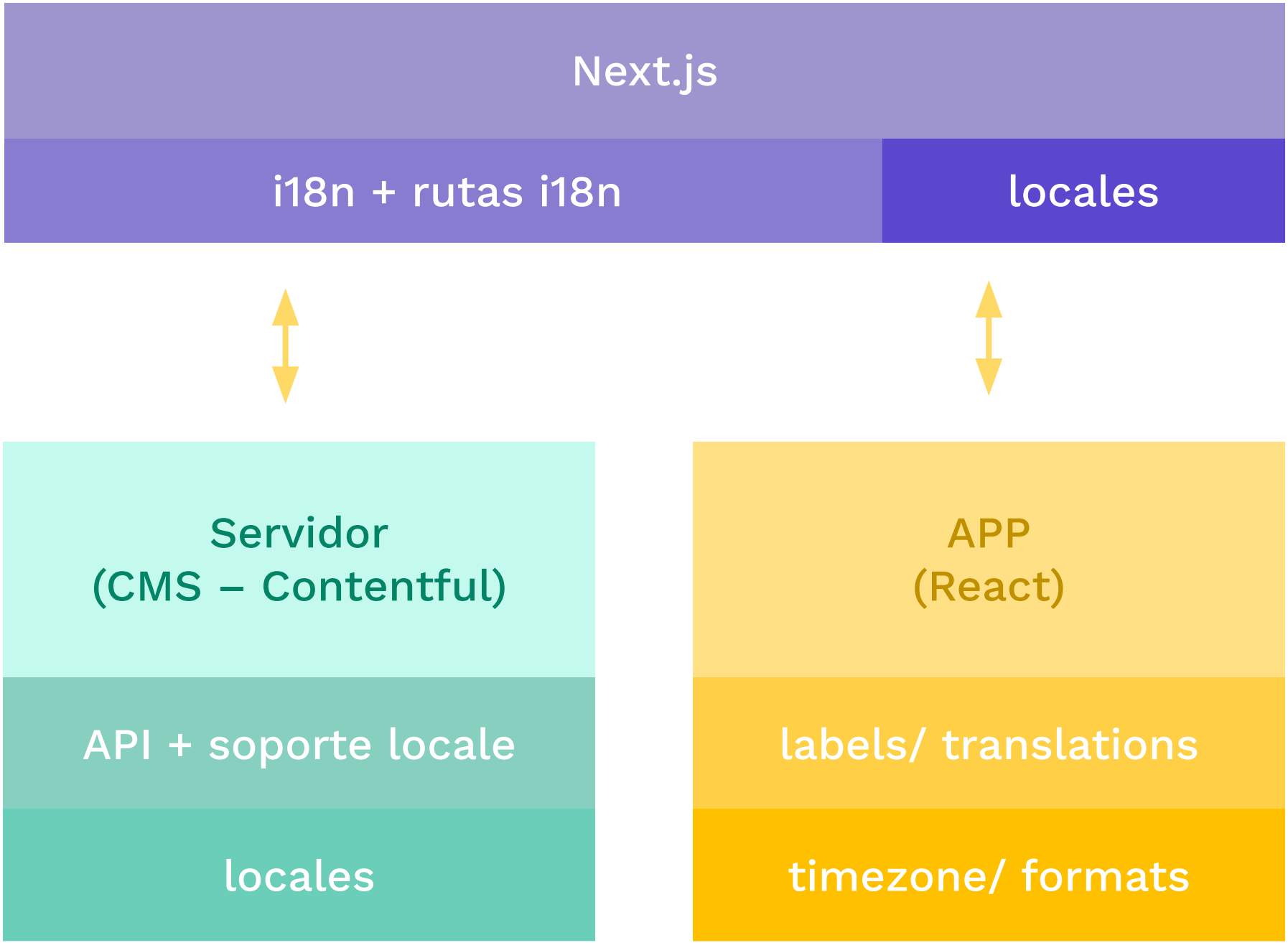
Grandes Datasets

@jonalvarezz 

¿Cuándo optimizar el código?


Curso de Optimización Web
129 Opiniones
AVANZADO
Descubre todos los conceptos, técnicas y herramientas para llevar tus aplicaciones web a otro nivel de performance, guiado bajo el modelo RAIL y primando siempre el análisis correspondiente para brindar la mejor experiencia de usuario.
• Explorar automatización de tareas con GitHub Actions
• Caché en servidores, CDN y con Service Workers
• Code Splitting.
• Estrategia de carga de Fuentes
Regístrate para tomar este curso
Ingresa tu email
EMPIEZA SIN COSTO
Al ingresar aceptas Términos de Servicio y Políticas de privacidad
Jonathan Alvarez
Senior Software Engineer en InterNations

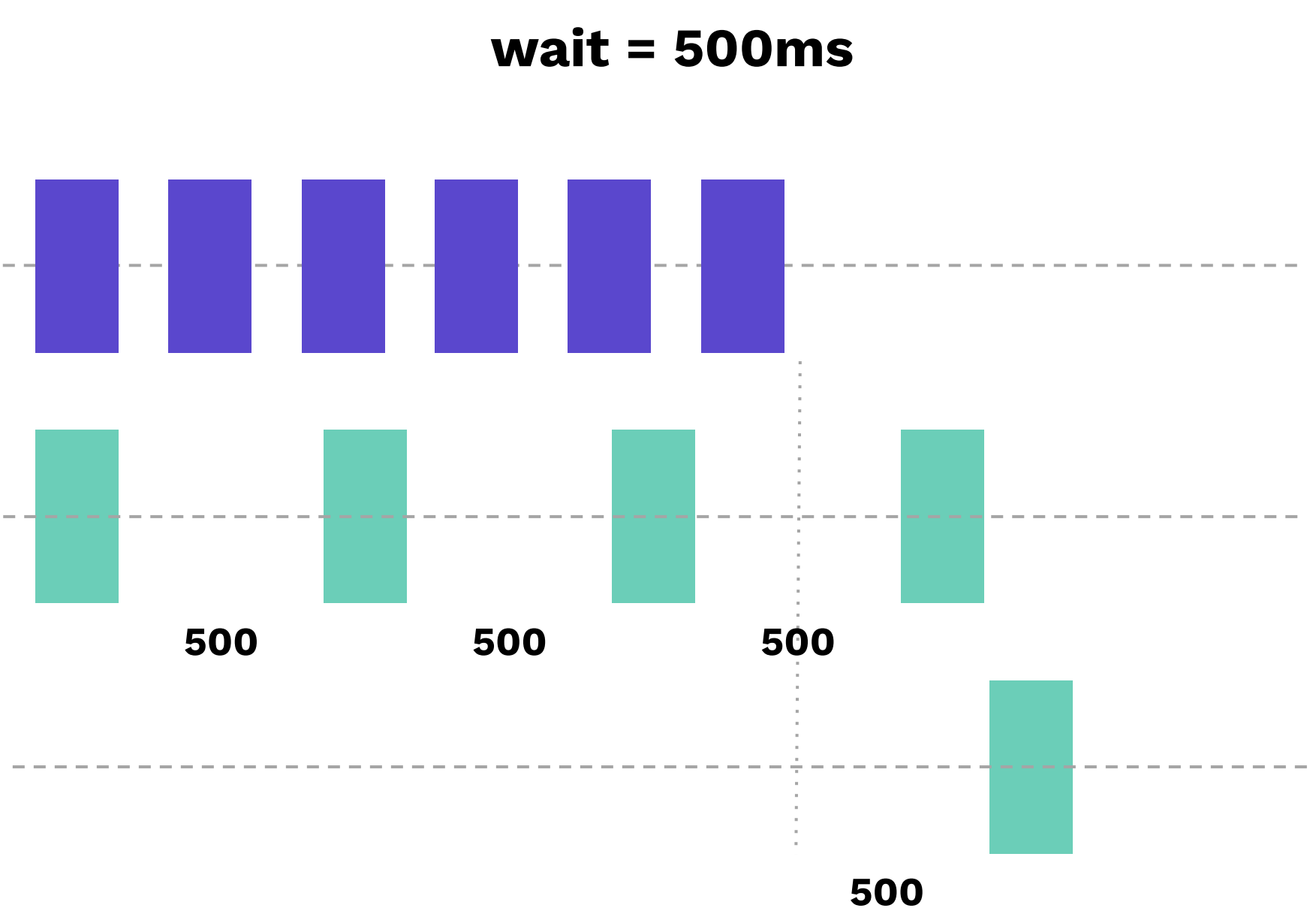
Throttling y debouncing

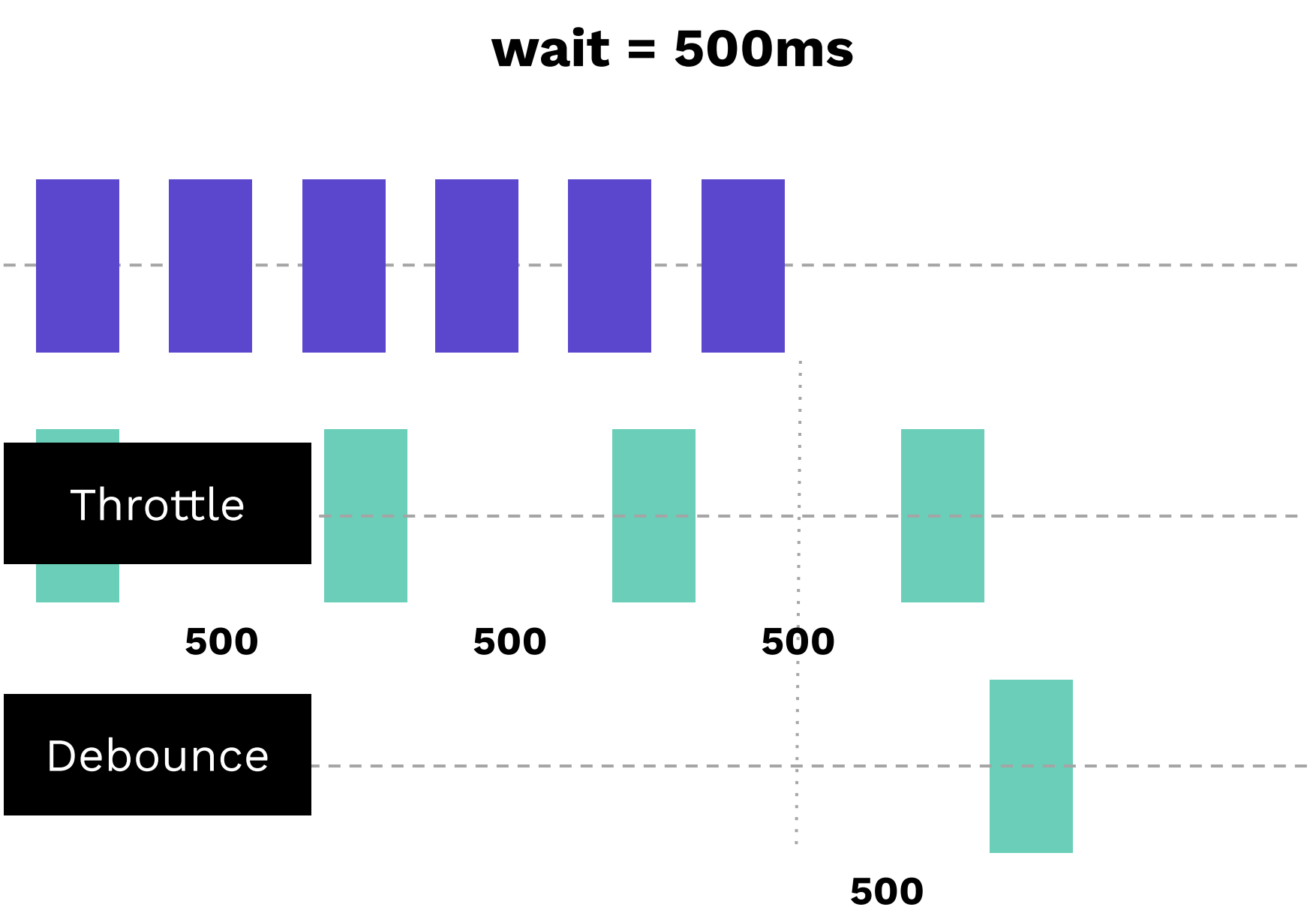

Next.js
i18n + rutas i18n
locales
Servidor
(CMS – Contentful)
API + soporte locale
locales
APP
(React)
labels/ translations
timezone/ formats

Analizando nuestra página de búsqueda

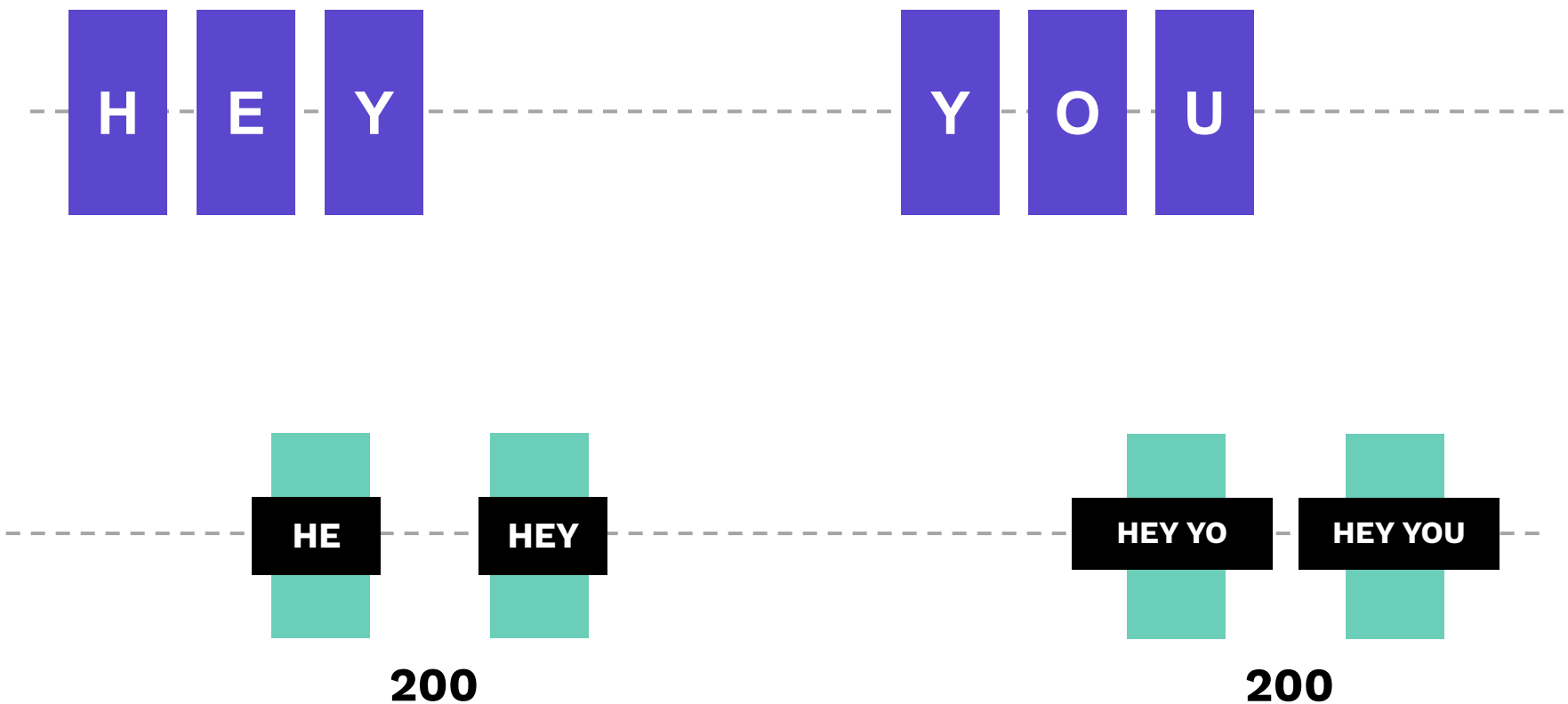
¿Por qué optimizar?

- Invocar una función menos veces.
- Proteger y optimizar recursos.

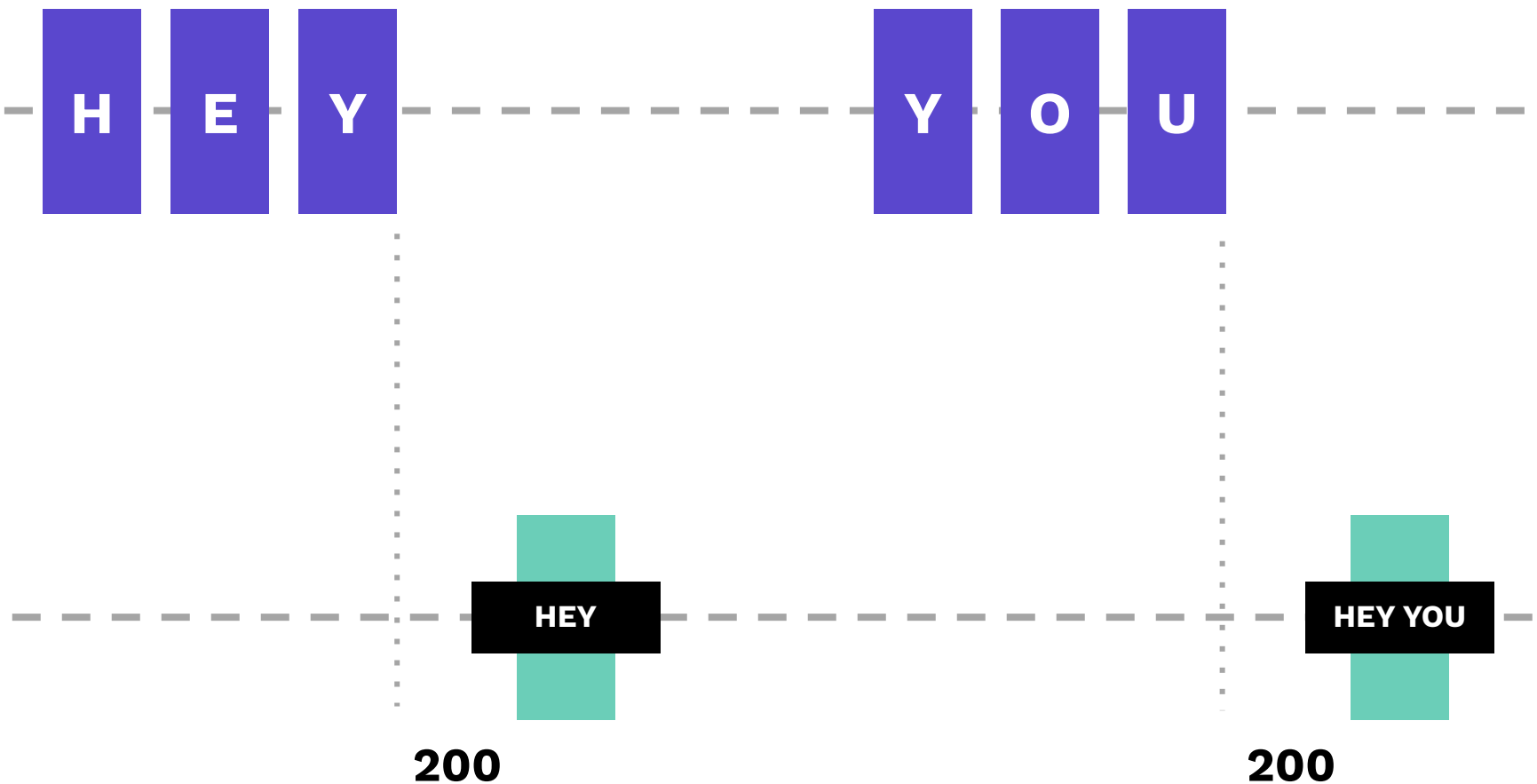

wait = 500ms
500
500
500
500

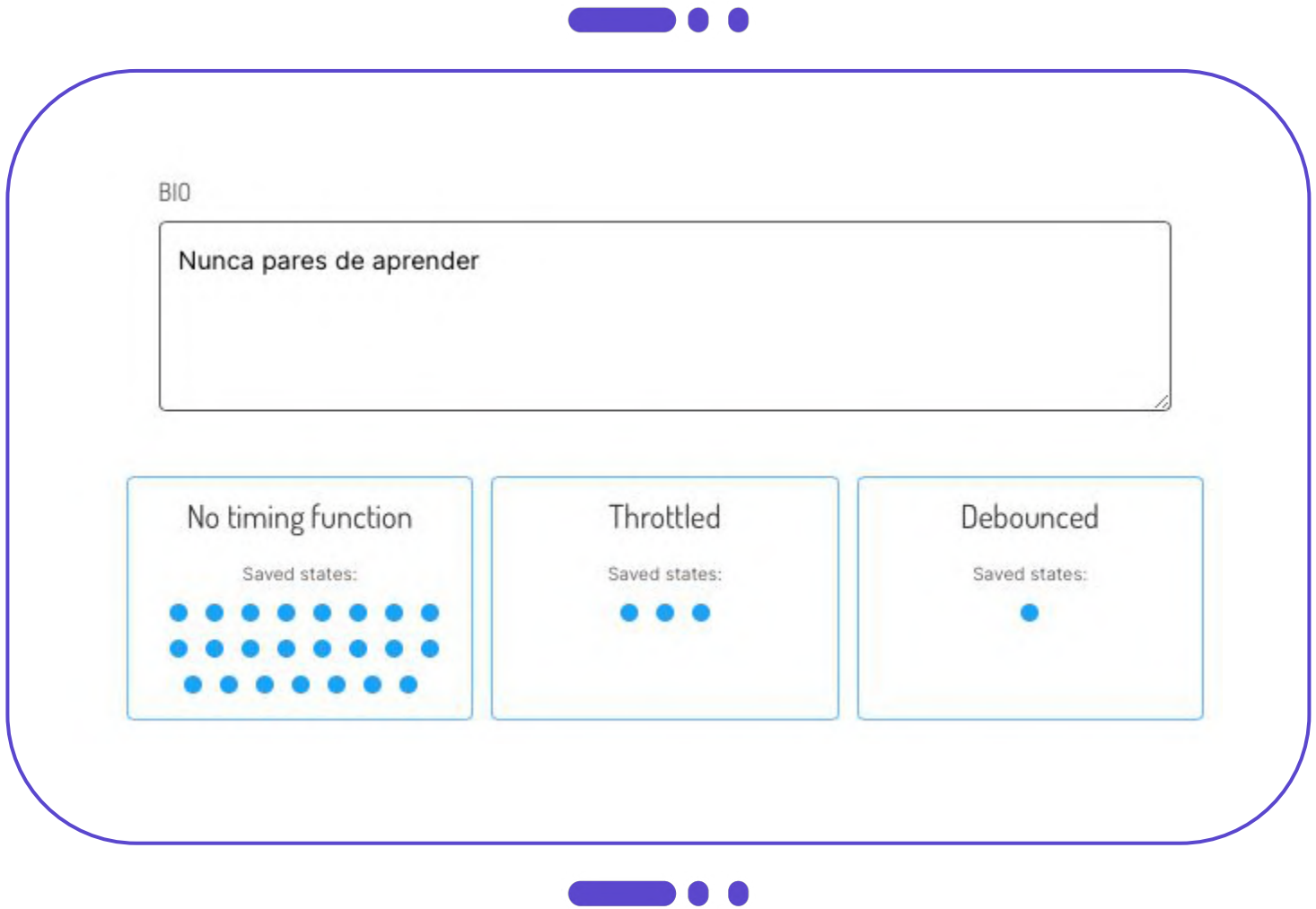

wait = 500ms
Throttle
500
500
500
Debounce
500

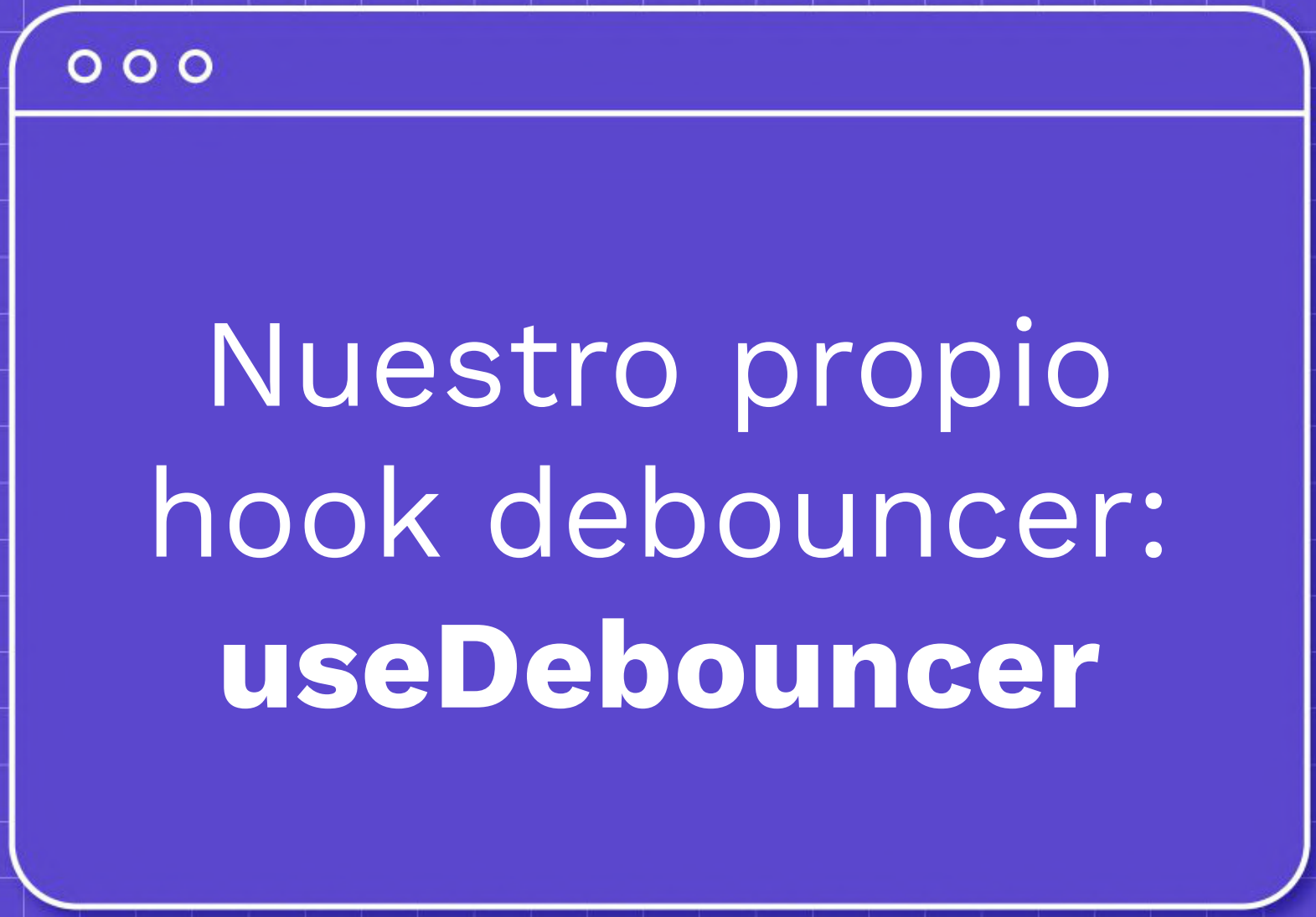
Throttle
wait = 200ms



Debounce
wait = 200ms




BIO
Nunca pares de aprender
No timing function
Saved states:
Throttled
Saved states:
Debounced
Saved states:


Nuestro propio
hook debouncer:
useDebouncer

Throttling y Debouncing con lodash en React


Caché en
memoria

Asincronía, caché y revalidación de recursos HTTP

Retos

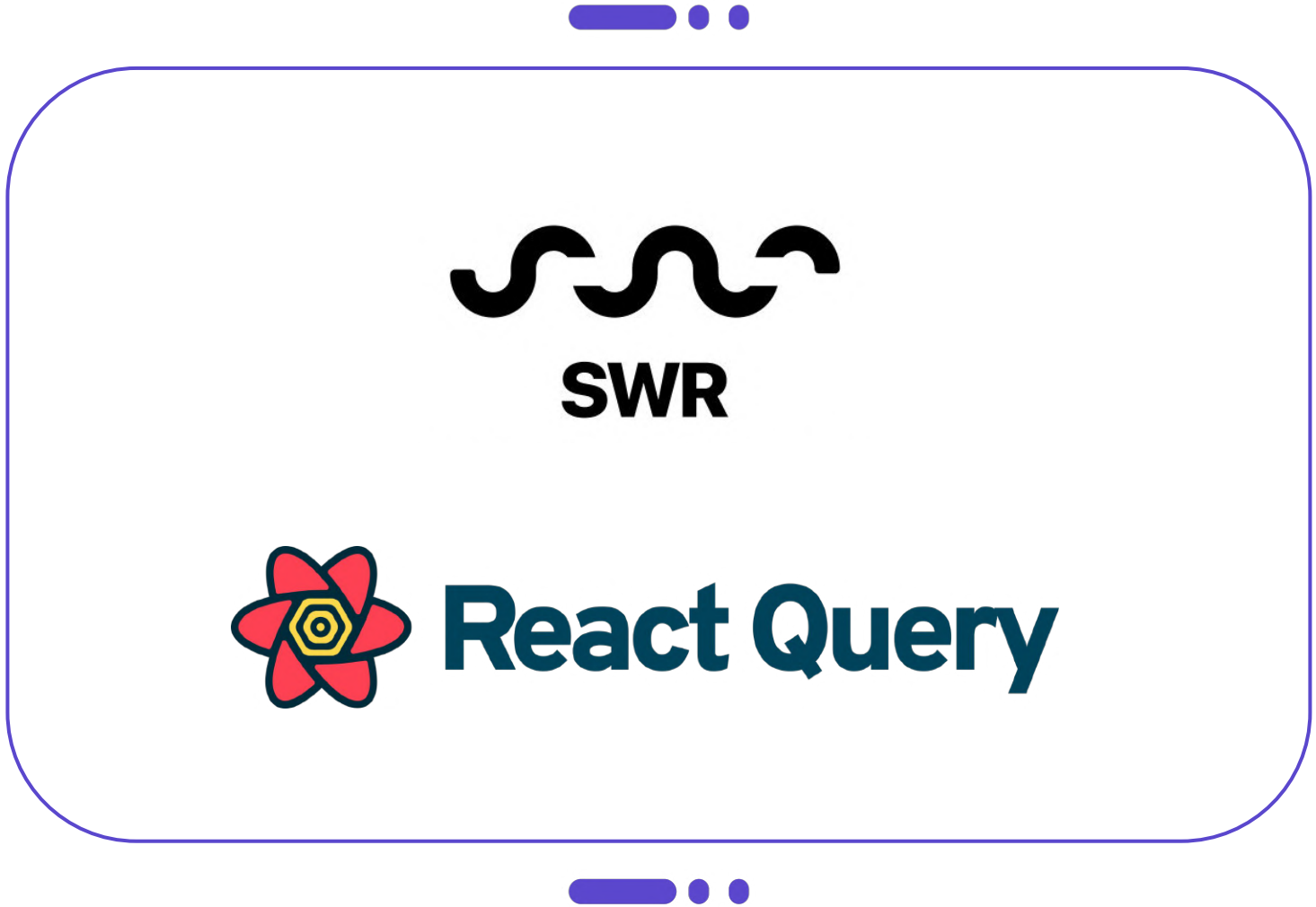
- Lógica de caché.
- Reintentos.
- Revalidación de datos.
- Manejo de errores.
- Peticiones seriales y paralelas.

“

En Redux, el manejo de efectos (async) termina siendo incrustado como una capa adicional dentro del middleware

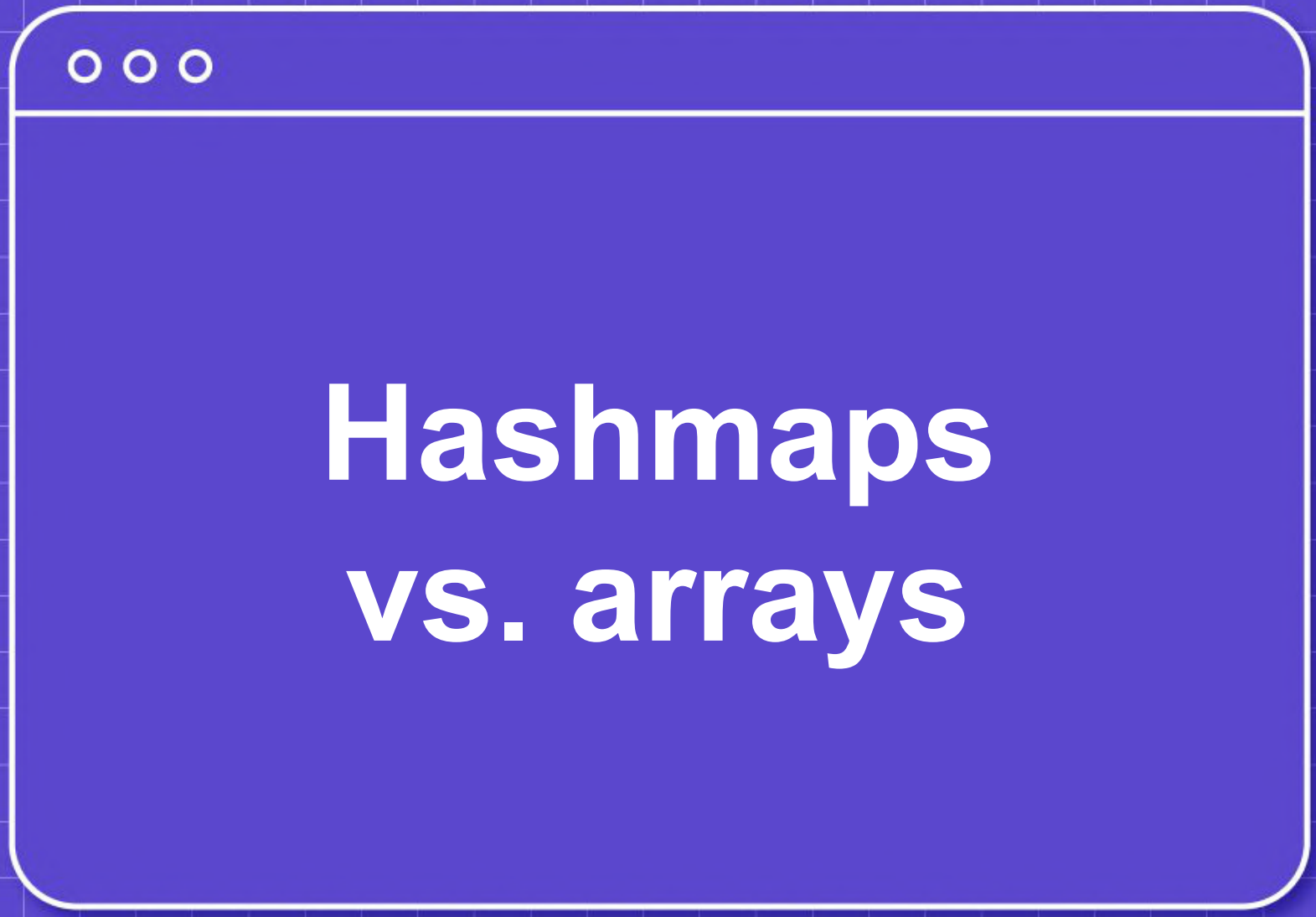
”

Enfoque stale-while-revalidate en el cliente

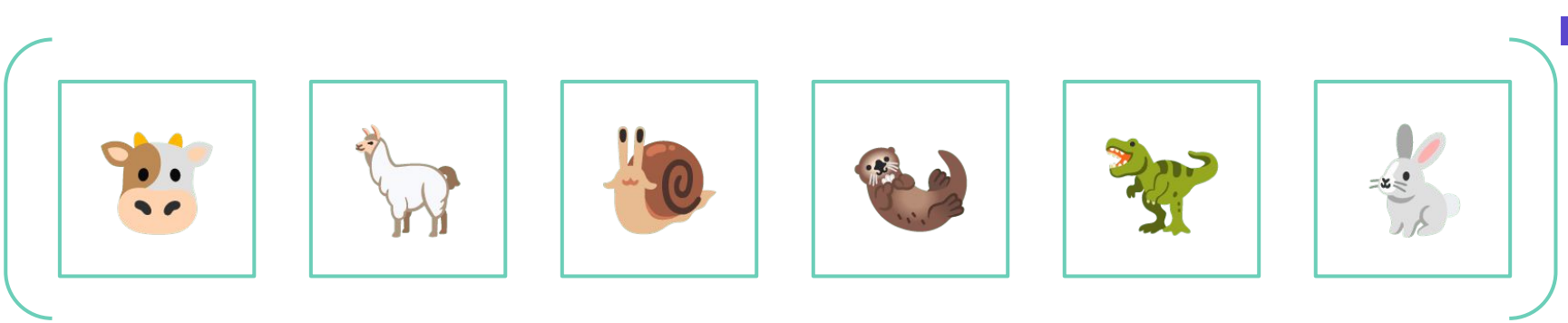

SWR
React Query

¡ Reto de la clase

Utilizar **react-query** o **use-swr** para reescribir las peticiones que se hacen en el cliente en la página de **“Top Stories”.**

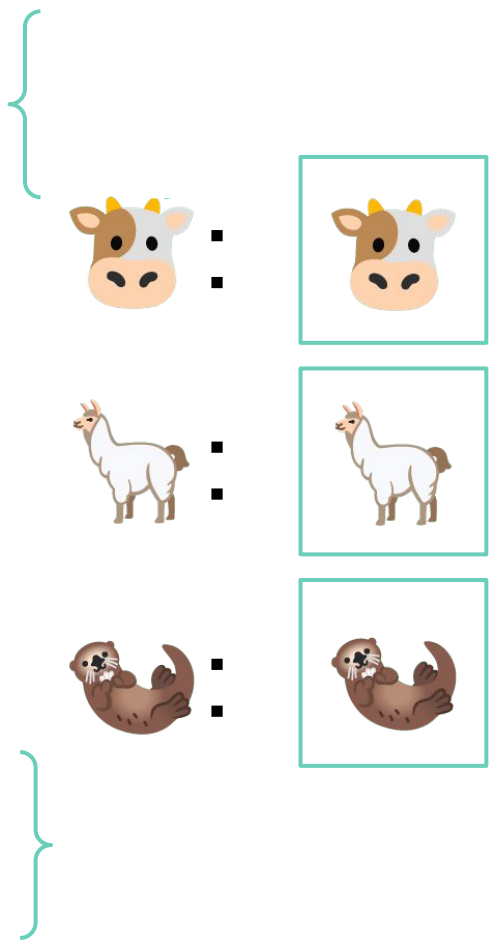

Hashmaps
vs. arrays

Array (lista)



O^n

Objeto (hashMap, key-value)



O^1

Utilidades

- Object.entries()

```
const bigList = [
  {id: '32dkfj', name: 'Jonas'},
  /* thousand of values */
]

const outputObject = {}

bigList.forEach((item) => {
  outputObject[item.id] = item
})
```

```
const bigList = [
  { id: "32dkfj", name: "Jonas" },
  /* thousand of values */
];

const outputObject = bigList.reduce(
  (output, item) => {
    output[item.id] = item;
    return output;
   }, {});
```

Utilidades

- Métodos de Array
- lodash.keyBy

Desventajas

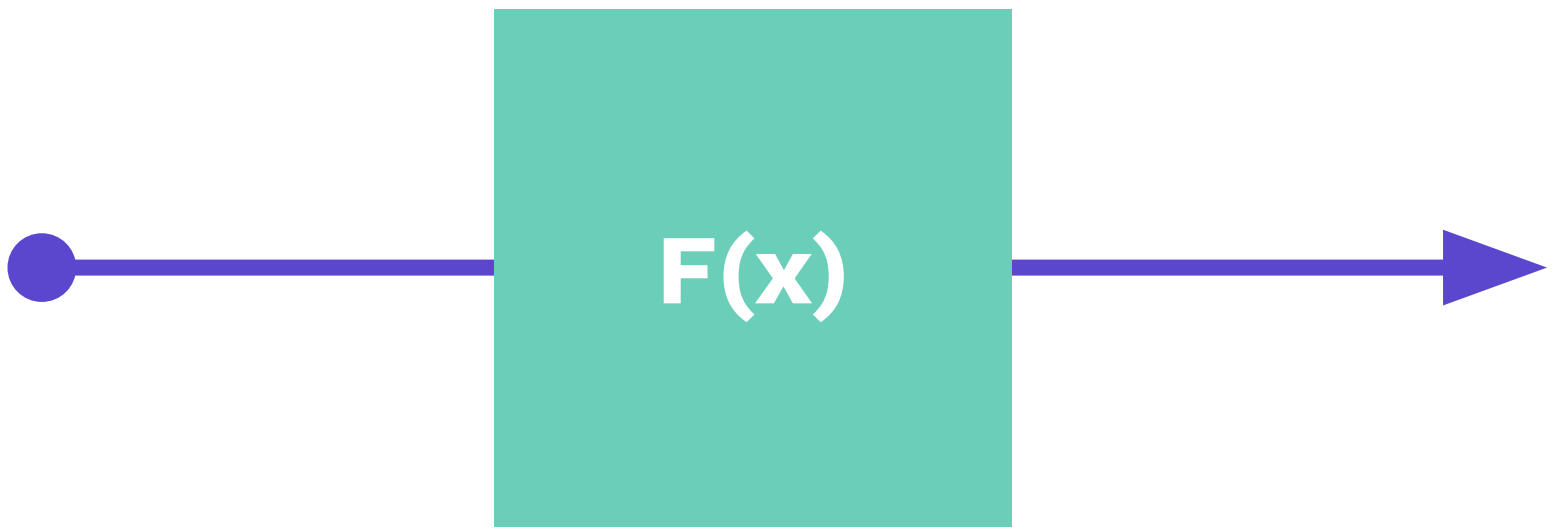
- Se debe serializar y deserializar.
- Agrega una capa de complejidad.

Utilidades para deserializar

- Object.values()
- Object.entries()

Memoización

Identificando re-renders problemáticos

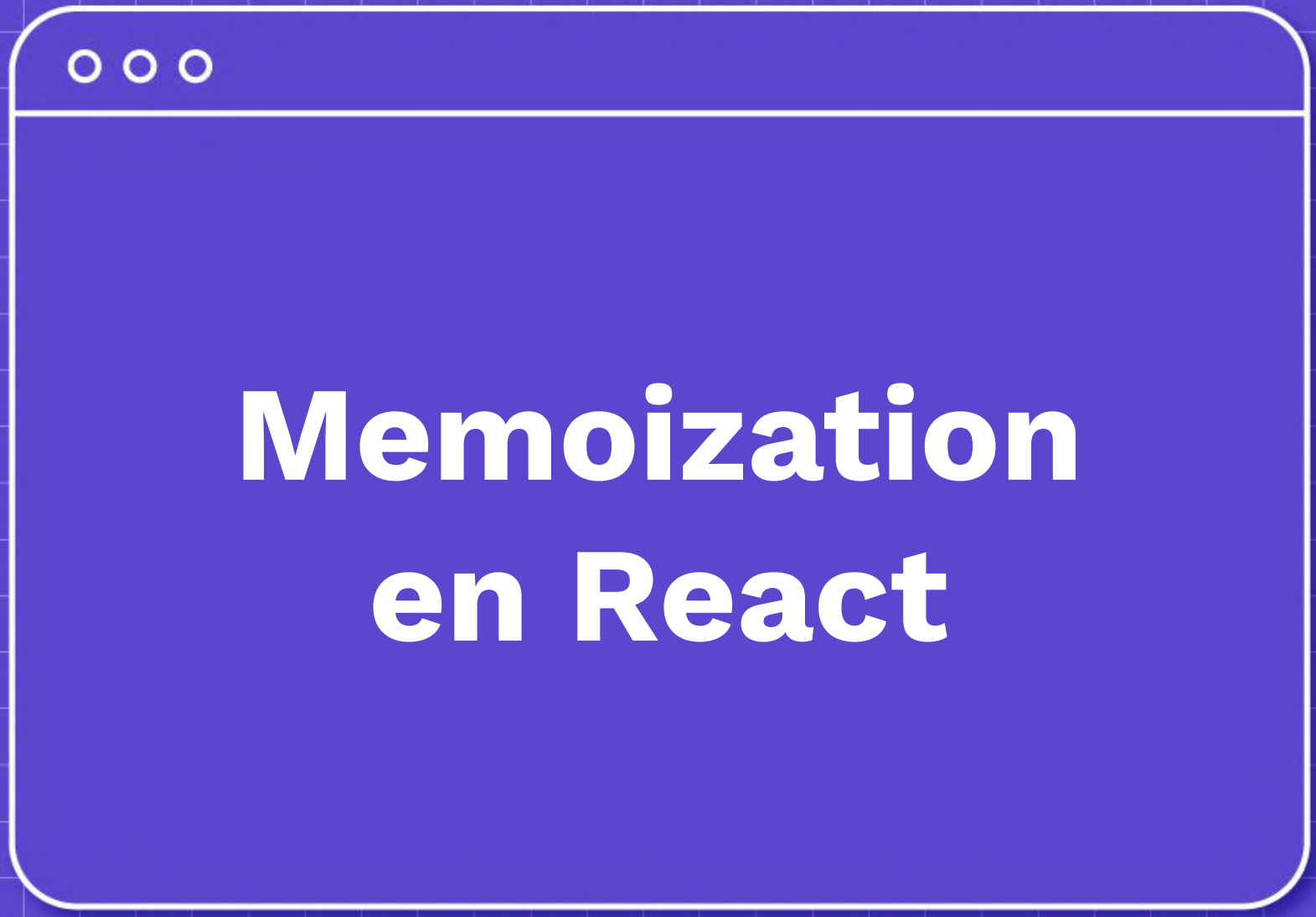

F(x)
INPUT
OUTPUT

```
function f (input1, input2, ...) {
  // calcular output
  return output
}
```

Memoización

1. Si el input es el mismo.

2. Retorno el valor calculado anteriormente.


Memoization
en React

Memoización en React

- React.PureComponent

Realiza un “shallow comparison” entre los props anteriores con los nuevos recibidos.

- React.memo / React.useMemo


Scrolling

Patrón windowing

“

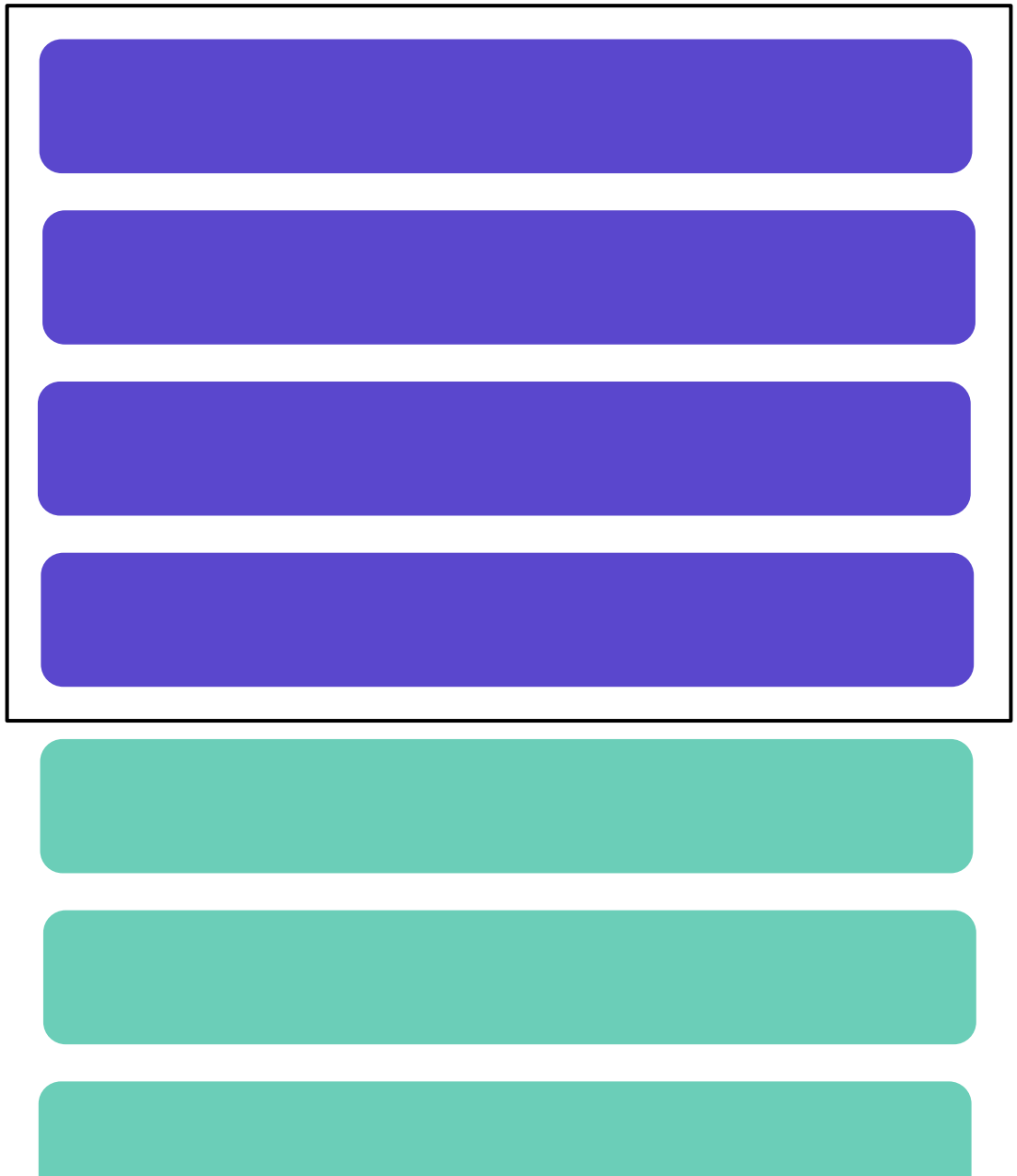
Pinta únicamente la parte visible de una lista de elementos muy grandes

”

“

Muy grandes = miles

”



react-window

EXAMPLES

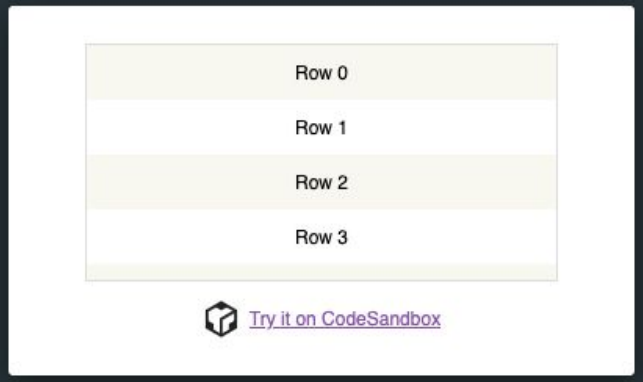
- Fixed Size List
- Variable Size List
- Fixed Size Grid
- Variable Size Grid
- Scrolling indicators
- Scrolling to an item
- Memoized List items
- RTL layout

COMPONENTS

- FixedSizeList
- VariableSizeList
- FixedSizeGrid
- VariableSizeGrid

METHODS

- areEqual
- shouldComponentUpdate




```
import { FixedSizeList as List } from 'react-window';

const Row = ({ index, style }) => (
  <div style={style}>Row {index}</div>
);

const Example = () => (
  <List
    height={150}
    itemCount={1000}
    itemSize={35}
    width={300}
  >
    {Row}
  </List>
);
```


```
import { FixedSizeList as List } from 'react-window';

const Column = ({ index, style }) => (
  <div style={style}>Column {index}</div>
);

const Example = () => (
  <List
    height={75}
    itemCount={1000}
    itemSize={100}
    layout="horizontal"
    width={300}
  >
    {Column}
  </List>
);
```

Próximos pasos