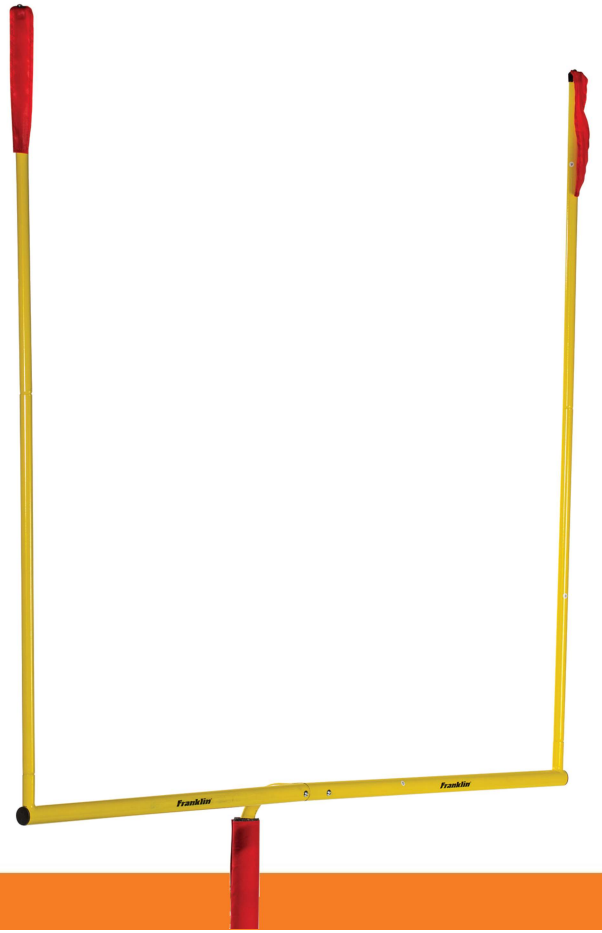
Effects

Develop Intelligence

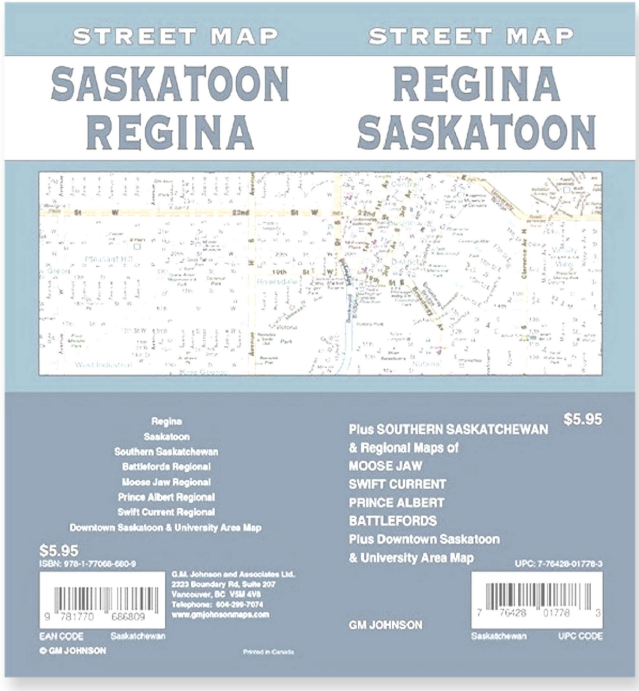
Goals

1. Describe the `useEffect` hook
2. List 3 use cases for `useEffect`
3. Explain when ***not*** to use `useEffect`



Roadmap

1. useEffect
2. Custom Hooks



Develop
Intelligence

useEffect

Rules for Components

Develop Intelligence

- Rendering should be *pure*
  - No side effects
  - Referential trasparency
- Side effects belong in event handlers
- What's missing: When you need rendering itself to create a side effect
  - e.g. Logging, fetching data, working with browser APIs

Effects

*Some components need to synchronize with external systems. For example, you might want to **control a non-React component** based on the React state, **set up a server connection**, or **send an analytics log** when a component appears on the screen. Effects let you run some code after rendering so that you can synchronize your component with some system outside of React.*

`useEffect`

- Runs ***after*** rendering
- Use `useEffect` for
  - Interacting with browser APIs
  - Interacting with a RESTful API
- Not for
  - Responding to user input-- i.e. event handlers
  - Transforming data for rendering

Example (I)

Develop Intelligence

```
1  import {useEffect,useState} from 'react';
2  export const App = ()=>{
3    const [contents,setContents] = useState('Loading...');
4    useEffect(()=>{
5      const timeOutID = setTimeout(()=>setContents('Done!'),5000);
6      return ()=>clearTimeout(timeOutID);
7    });
8    return <div>
9             <h1>Contents:</h1>
10            <p>{contents}</p>
11           </div>
12 };
```

Example (II)

```
import {useEffect, useState} from 'react';
export const Position = ()=>{
  const [position,setPosition] = useState({latitude:0,longitude:0});
  useEffect(()=>{
    fetch('http://www.geoplugin.net/json.gp')
      .then(r=>r.json())
      .then(result=>{
        setPosition({latitude:result.geoplugin_latitude,longitude:result.geop
    });
  },[]);
  return (
    <div>
      <h1>Latitude: {position.latitude}</h1>
      <h1>Longitude: {position.latitude}</h1>
    </div>
  )};
```

How to write an Effect

Develop Intelligence

1. Declare an effect
2. *Optionally* specify dependencies
3. *Optionally* add cleanup

What are dependencies?

Develop Intelligence

- By default, `useEffect` fires after each render
- Conserve resources by specifying state dependencies
- e.g.
  - **Position** (2 slides ago) has no dependencies because position doesn't change with subsequent renders

Cleanup

- Optionally, an effect can clean itself up when its component is unmounted
- Examples:
  - Unsubscribe from an observable
  - Cancel a promise
  - Close a connection

Promise Cancellation Pattern

- ECMAScript promises aren't cancellable
- BUT you can decide to ignore the resolution

```
useEffect(() => {
  let cancelled = false;
  async function getWeather() {
    const service = new WeatherService();
    const currentWeather = await service.getWeather();
    if (cancelled) return;
    setWeather(currentWeather);
  }
  getWeather();
  () => (cancelled = true);
}, []);
```

Be Careful

- Effects are *maybe* the most overused and abused part of react

Fail

```
const Header = ({ children }: PropsWithChildren) => {
  const [capitalized, setCapitalized] = useState(children);
  useEffect(()=>setCapitalized(children?.toString().toUpperCase()));
  return <h1>{capitalized}</h1>;
};
```

Better

```
const Header = ({ children }: PropsWithChildren) =>
   <h1>{children?.toString().toUpperCase()}</h1>;
```

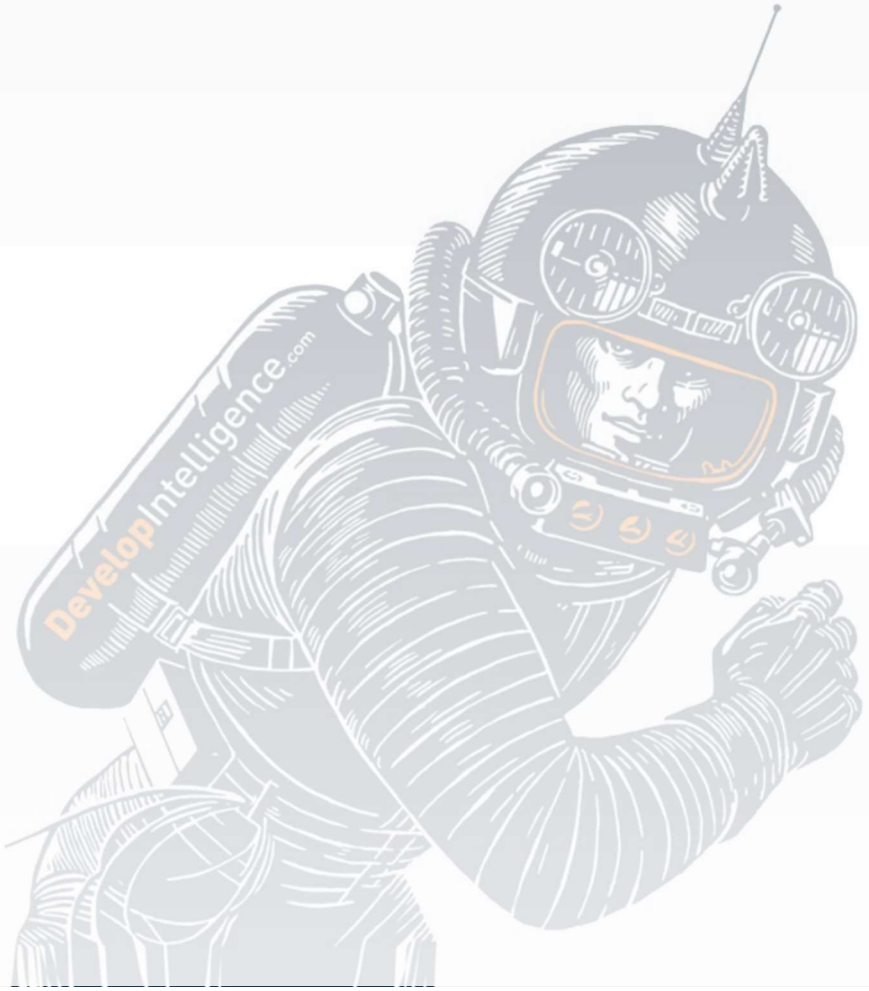
Infinite Loops

Develop Intelligence

```
const Ticker = () => {
  const [ticks, setTicks] = useState(0);
  useEffect(() => {
    setInterval(() => setTicks(ticks + 1), 1000);
  },[ticks]);
  return <h1>{ticks}</h1>;
};
```

Develop
Intelligence

Custom Hooks



Review

- You can only use hooks in 2 places:
  - A functional component
  - Another hook
- Sharing logic between components implies custom hooks

Custom Hooks

React comes with several built-in Hooks like useState, useContext, and useEffect. Sometimes, you'll wish that there was a Hook for some more specific purpose: for example, to fetch data, to keep track of whether the user is online, or to connect to a chat room. You might not find these Hooks in React, but you can create your own Hooks for your application's needs.

Example I: useFrozenState

- Problem: useState does't freeze

```tsx
interface State<T>{value:T;}

export const UseFrozenStateComponent = () => {
  const [count,] = useState<State<number>>({value:1});
  const handleClick = () =>
    count.value = count.value + 1;
  return (
    <div className="prose p-2 practice">
      <h1>Count: {count.value}</h1>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
};
```

Solution: useFrozenState

Develop Intelligence

```
interface Holder<T> extends Readonly<{
    content: T;
}> {}

export function useFrozenState<S>(
  initialState: S,
): [Holder<S>, setContent: (state: S) => void] {
  const [s, setS] = useState(initialState);
  const result = { content: s };
  Object.freeze(result);
  return [result, setS];
}
```

Example 2: Weather

Develop Intelligence

```
export function useWeather() {
  const [weather, setWeather] = useState<WeatherModel | null>(null);

  useEffect(() => {
    let cancelled = false;
    async function getWeather() {
      const service = new WeatherService();
      const w = await service.getWeather();
      if (cancelled) return;
      setWeather(w);
    }
    getWeather();
    () => (cancelled = true);
  }, []);

  return weather;
```

Lab: Weather

Instructions: `src/practice/weather`

Summary:

- Get your current location
  - Put it in a component
- Get your NOAA grid location
  - Put it in a component
- Get the current weather
  - Put it in a component!

Develop
Intelligence

Review

1. Describe the `useEffect` hook
2. List 3 use cases for `useEffect`
3. Explain when ***not*** to use `useEffect`