`useEffect` is a React Hook used for managing side effects in functional components. Side effects typically include operations like data fetching, subscriptions, or manually changing the DOM, which need to be performed after rendering but are not part of the component rendering itself.

Here's a breakdown of how `useEffect` works:

1. **Declaration**: You import `useEffect` from the React library:

```javascript
import React, { useEffect } from 'react';
```

2. **Syntax**: The basic syntax of `useEffect` looks like this:

```javascript
useEffect(() => {
  // Side effect code goes here
}, [dependencies]);
```

3. **Callback Function**: The first argument to `useEffect` is a callback function. This function contains the code that you want to run as a side effect. This code can include data fetching, subscription setup, DOM manipulation, or any other side effect.

4. **Dependencies Array**: The second argument to `useEffect` is an optional array of dependencies. If provided, the effect will only re-run if one of the values in the dependencies array changes. If the dependencies array is omitted, the effect will run after every render.

5. **Return Function (Optional)**: The function returned by `useEffect` can be used for cleanup. This cleanup function runs when the component unmounts or before running the effect again due to changes in dependencies.

Here's an example of using `useEffect` to fetch data when a component mounts:

```javascript
import React, { useState, useEffect } from 'react';

function MyComponent() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Function to fetch data
    const fetchData = async () => {
      const response = await fetch('https://api.example.com/data');
      const result = await response.json();
      setData(result);
    };

    fetchData(); // Call the function to fetch data when component mounts

    // Cleanup function (optional)
    return () => {
      // Code to cancel any ongoing operations or clean up resources
    };
  }, []); // Empty dependency array means this effect runs only once after mounting

  return (
    <div>
      {/* Render data */}
      {data ? <p>{data}</p> : <p>Loading...</p>}
    </div>
  );
}
```

In this example, `useEffect` is used to fetch data from an API when the component mounts. The effect runs only once after the component mounts, thanks to the empty dependency array `[]`. If you had dependencies, you would include them in the array, and the effect would re-run whenever any of those dependencies change.