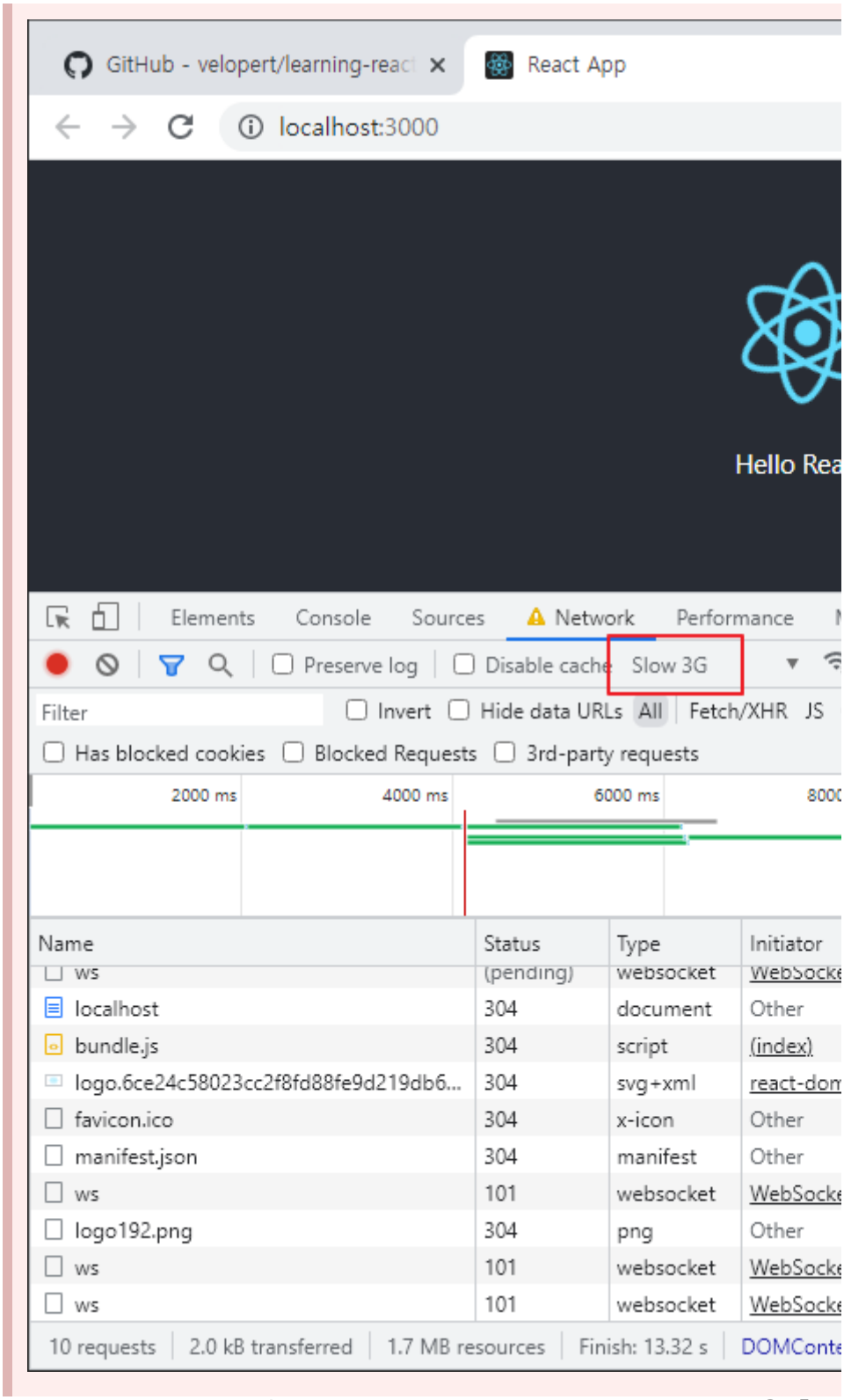
}>

19
Co
Sp

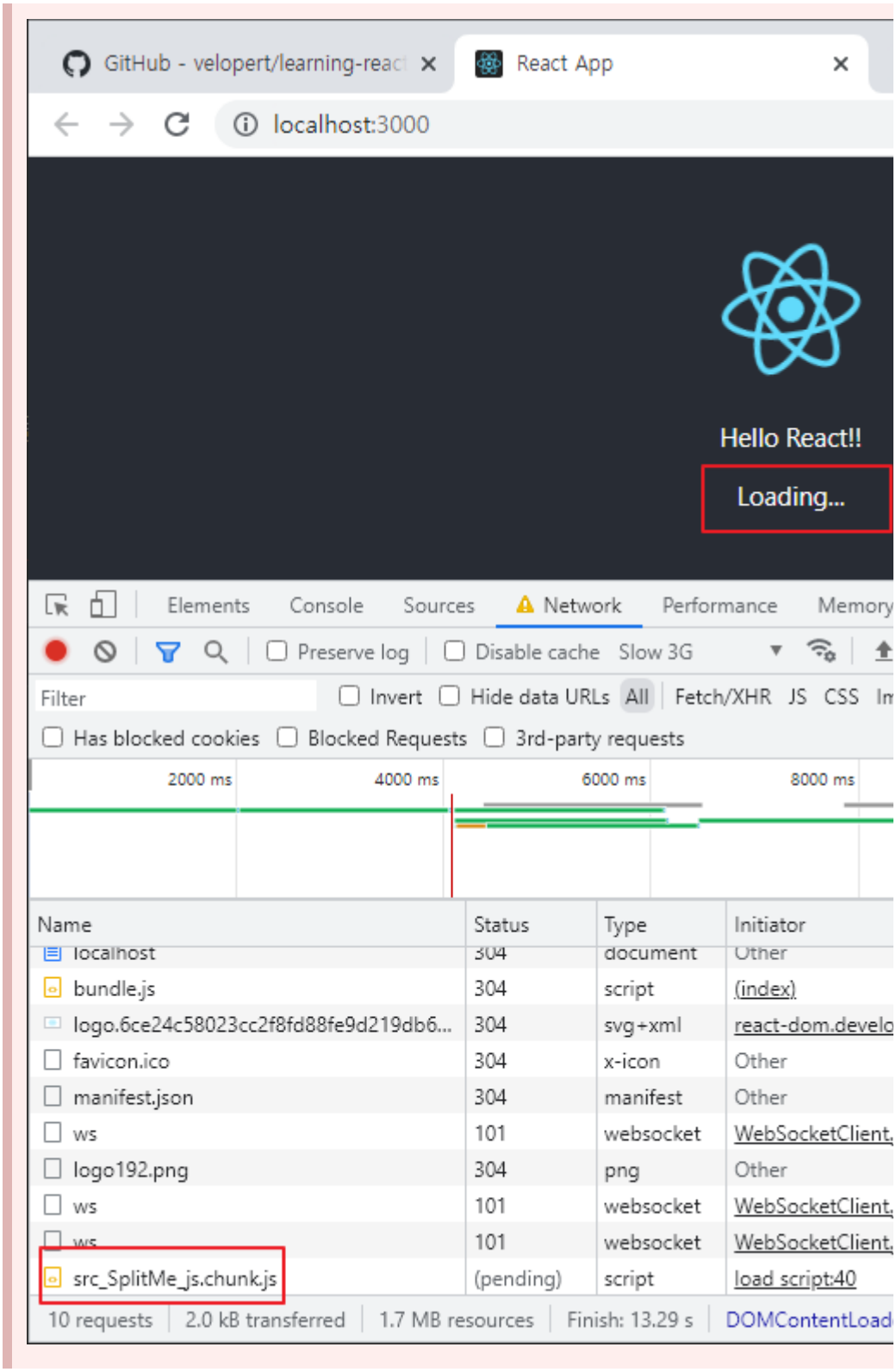
##### src/App.js - 다시 함수형 컴퍼넌트로 수정

```js
import React, { useState, Suspense } from 'react';
import logo from './logo.svg';
import './App.css';

const SplitMe = React.lazy(() => import('./SplitMe'));

function App () {

  const [ visible, setVisible ] = useState(false);

  const onClick = () => {
    setVisible(true);
  };

  return (
    <div className="App">
    <header className="App-header">
      <img src={logo} className="App-logo" alt="logo" />
      <p onClick={onClick}>Hello React!!</p>
      <Suspense fallback={<div>Loading...</div>}>
        {visible && <SplitMe />}
      </Suspense>
    </header>
  </div>
  )
}

export default App;
```

- 크롬개발자도구 Network에서 online을 slow3G로 변경후 SplitMe가 스플링된 것을




- 확인 후 Hello React를 클릭하면 클릭후에 스플릿된 코드가 로딩되는 것을 확인




### 19.2.3 Loadable Components를 통한 코드 스플리팅

- Loadable Components는 스플리팅을 편하게 도와주는 서드파티 라이브러리
- 좋은 점은 서버사이드 랜더링을 지원한다는 것이다(React.lazy와 Suspense는 아
- 또한, 랜더링하기 전에 필요할 때 스플리팅파일을 미리 불러올 수 있는 기능을
- 서버 사이드 랜더링이란 웹 서비스의 초기로딩속도 개선, 캐싱 및 검색엔진 최적
- 라이브러리 설치 yarn add @loadable/component
- 사용법은 React.lazy와 비슷하지만 Suspense를 사용할 필요는 없다.

**src/App.js - loadable component 적용**

```
import React, { useState, Suspense } from 'react';
import logo from './logo.svg';
```

```
import './App.css';
import loadable from '@loadable/component';

const SplitMe =loadable(() => import('./SplitMe'), {
  fallback: <div>Loading...</div>
  // 로딩중에 다른 UI를 보여주고 싶을 경우 : import {Bars, Radio} from 'react-i
});

function App () {

  const [ visible, setVisible ] = useState(false);

  const onClick = () => {
    setVisible(true);
  };

  return (
    <div className="App">
    <header className="App-header">
      <img src={logo} className="App-logo" alt="logo" />
      <p onClick={onClick}>Hello React!!</p>
      <Suspense fallback={<div>Loading...</div>}>
        {visible && <SplitMe />}
      </Suspense>
    </header>
  </div>
  )
}

export default App;
```

**src/App.js - 컴퍼넌트를 미리 불러오는(preload)방법**

- preload()함수를 이용해서 mouse를 올려만 놓아도 SplitMe가 로딩이 시작된다.
- Loadable Components는 미리불러오기기능외에도 타임아웃, 로딩UI디레이, SSR호
    - 참고: https://www.smooth-code.com/open-source/loadable-components/do

```
import React, { useState, Suspense } from 'react';
import logo from './logo.svg';
import './App.css';
import loadable from '@loadable/component';

const SplitMe =loadable(() => import('./SplitMe'), {
  fallback: <div>Loading...</div>
});

function App () {

  const [ visible, setVisible ] = useState(false);

  const onClick = () => {
    setVisible(true);
  };

  const onMouseOver = () => {
    SplitMe.preload();
  };

  return (
```

```
    <div className="App">
    <header className="App-header">
      <img src={logo} className="App-logo" alt="logo" />
      <p onClick={onClick} onMouseOver={onMouseOver}>Hello React!!</p>
      <Suspense fallback={<div>Loading...</div>}>
        {visible && <SplitMe />}
      </Suspense>
    </header>
  </div>
  )
}

export default App;
```

## 19.3 정리

- SSR할 계획이 없다면 React.lazy와 Suspense를 구현하고 계획이 있다면 Loadabl
- 리엑트 공식문서에서도 SSR할 경우 Loadable Components를 사용하도록 권장
- 향후 지원할 수도 있으니 공식문서를 다시 한번 확인
    - https://reactjs.org/docs/code-splitting.html#reactlazy

- 

App

```
im
lo
fr
'.
im
'.

fu
```

```
Ap
{

re
(

<d
cl

<h
cl
he

<i
sr
{l
cl
lo
al
/>

<p
Re
</

</

</

);
}

ex
de
Ap
```

- 

jsx에서 emm 사용하기

-

-

-

19
자

# 바스크립트함수비동기로딩

```
src
```

```
ex
de
fu
no
{

al
Re
}
```

```
App
```

```
im
lo
fr
'.
im
'.
im
no
fr
'.

fu
Ap
{


co
on
=
()
=>
{

no

}
```

```
re
(

<d
cl

<h
cl
he

<i
sr
{l
cl
lo
al
/>

<p
on
{o
Re
</

</

</

);
}

ex
de
Ap
```

•

-

App - 함수형태로선언

-

-

```
im
lo
fr
'.
im
'.

fu
Ap
{


co
on
```

```
=
()
=>
{

im
=>
re

}

re
(

<d
cl

<h
cl
he

<i
sr
{l
cl
lo
al
/>

<p
on
{o
Re
</

</

</

);
}

ex
de
Ap
```

- 

19
Re
와
Su
를통한컴퍼넌트코드스플리팅

- 

# 19st 를 사용한 코드 스플리팅

src

```
co
Sp
=
()
=>
{

re
<d
}
ex
de
Sp
```

•

```
src

im
lo
fr
'.
im
'.

cl
Ap
ex
Co
{


st
=
{

Sp
nu

}


ha
```

```
=
as
()
=>
{

co
lo
=
aw
im

th

Sp
lo

})

}


re
{

co
{
Sp
}
=
th

re
(

<d
cl

<h
cl
he

<i
sr
{l
cl
lo
al
/>

<p
on
{t
Re
</

{S
```

```
&&
<S
/>

</

</

)

}
}

ex
de
Ap
```

19 Re와 Su사용하기

-

-

-

```
//
Sus
imp
{
Sus
}
fro
're
```

Loa