

Bab 7

Debugging React App

1. Module Introduction

1. Module Introduction

- sebagai react developer, nantinya anda juga akan menulis kode yang mengandung kesalahan dan bug
- Oleh karena itu dalam bab ini kita akan mempelajari cara melakukan debugging react
- Pertama-tama kita akan belajar memahami pesan kesalahan sehingga mempermudah kita menemukan dan memperbaiki kesalahan
- Beberapa kesalahan tidak disebabkan oleh penulisan syntax, namun juga disebabkan karena kesalahan logika
- Dan kita akan belajar cara menemukan kesalahan tersebut dengan bantuan developer tools pada browser,
- Kita juga akan belajar tentang react strict mode, pengertian, cara menggunakan, dan mengapa kita harus menggunakannya

1. Module Introduction

- Kita juga akan belajar terkait react devtools, cara menginstallnya dan apa yang bisa kita lakukan dengan react devtools

2. The Starting Project

2. The Starting Project

- Untuk percobaan bab ini kita akan menggunakan proyek latihan yaitu kalkulator investasi yang dibuat diawal kursus
- Ini bukanlah aplikasi react yang kompleks hanya terdiri dari beberapa state dan komponen
- Proyek inilah yang kita gunakan ketika ingin mempelajari aspek dan fitur ketika ingin memperbaiki kesalahan pada project react
- Kita akan membuat beberapa kesalahan di project ini dan belajarcara melihat,memahami,dan memperbaikinya

2. The Starting Project



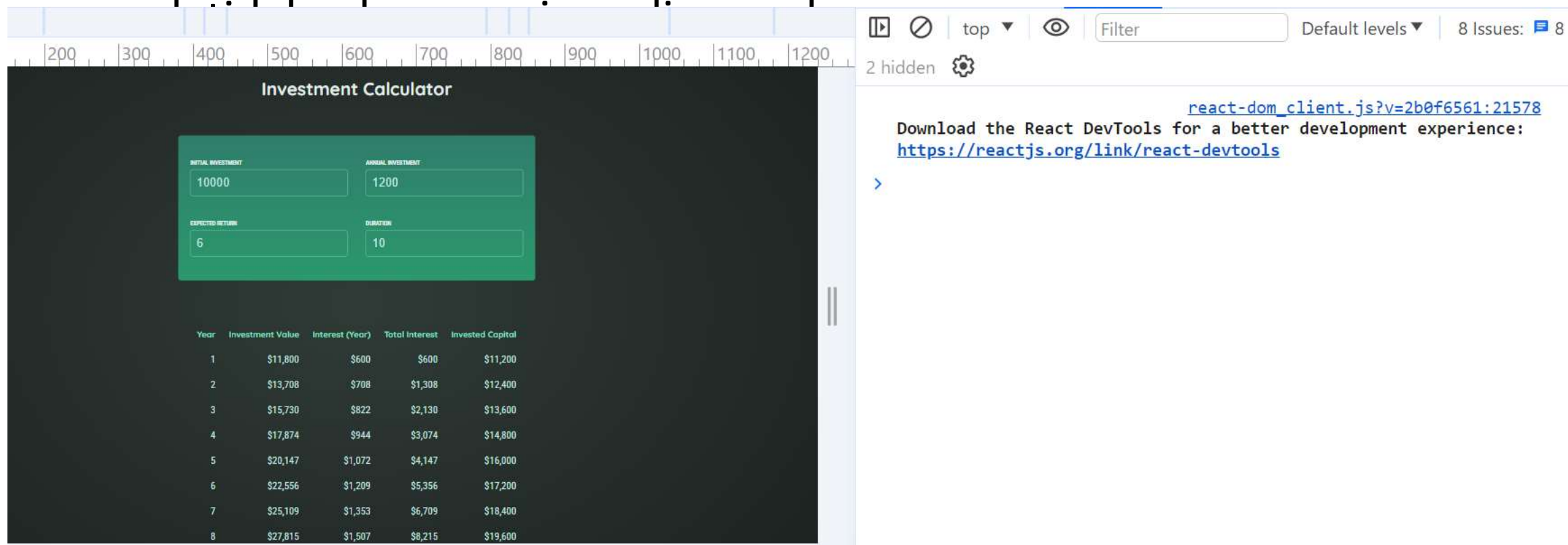
The image shows a digital interface for an "Investment Calculator". At the top center is an icon of a green money bag with a dollar sign. Below the icon, the title "Investment Calculator" is displayed in white text. The main input area is a green rectangle containing four white input fields arranged in a 2x2 grid. The fields are labeled "INITIAL INVESTMENT", "ANNUAL INVESTMENT", "EXPECTED RETURN", and "DURATION". The values entered are 10000, 1200, 6, and 10 respectively. Below the input area is a table with five columns: "Year", "Investment Value", "Interest (Year)", "Total Interest", and "Invested Capital". The table contains three rows of data for years 1, 2, and 3.

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600

3. Understanding React Error Message

3. Understanding React Error Message

- Ketika kita membuka proyek web investasi, semua tampil berjalan



The screenshot shows a web application titled "Investment Calculator" running in a browser. The application has a dark theme. The input fields are as follows:

INITIAL INVESTMENT	ANNUAL INVESTMENT
10000	1200

EXPECTED RETURN	DURATION
6	10

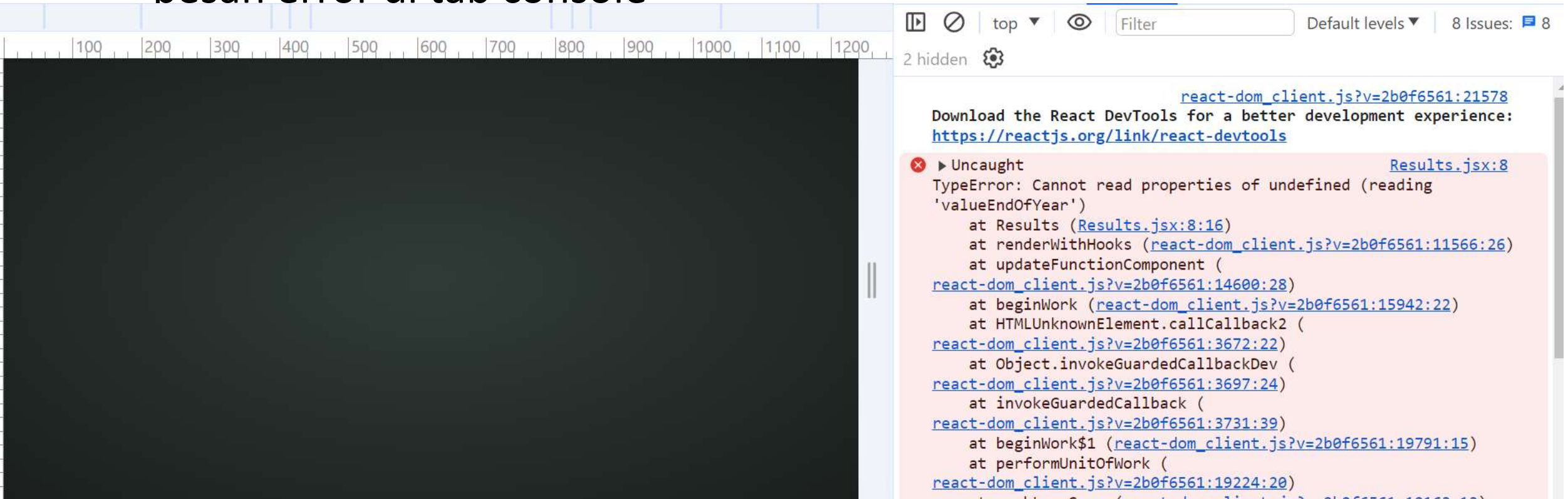
Below the input fields is a table showing the investment results over 8 years:

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000
6	\$22,556	\$1,209	\$5,356	\$17,200
7	\$25,109	\$1,353	\$6,709	\$18,400
8	\$27,815	\$1,507	\$8,215	\$19,600

On the right side, the React DevTools error console is open, showing a message from `react-dom_client.js?v=2b0f6561:21578` with the text: "Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>".

3. Understanding React Error Message

- Namun ketik kita memasukkan angka nol atau negatif ke kolom duration, program akan rusak, layar akan menjadi blank serta banya pesan error di tab console



3. Understanding React Error Message

- Ketika melihat banyak teks berwarna merah dilayar, banyak programmer yang merasa kewalahan
- Namun anda tidak perlu khawatir sebab error di react cukup membantu dan berwawasan
- Dalam pesan ini dikatakan gagal membaca property yang bernilai undefined, property yang dimaksud adalah valueEndOfYear
- Karena propertynya undefined, kemungkinan objek nya juga bernilai undefined



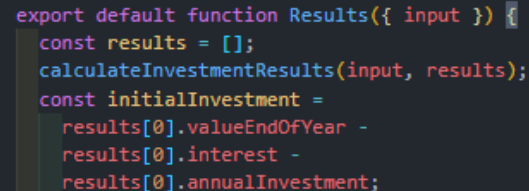
```
Uncaught  
TypeError: Cannot read properties of undefined (reading  
  'valueEndOfYear')  
    at Results (Results.jsx:8:16)  
    at renderWithHooks (react-dom_client.js?v=2b0f6561:11566:26)  
    at updateFunctionComponent (  
      react-dom_client.js?v=2b0f6561:14600:28)  
    at beginWork (react-dom_client.js?v=2b0f6561:15942:22)  
    at HTMLUnknownElement.callCallback2 (  
      react-dom_client.js?v=2b0f6561:3672:22)  
    at Object.invokeGuardedCallbackDev (  
      react-dom_client.js?v=2b0f6561:3697:24)  
    at invokeGuardedCallback (  
      react-dom_client.js?v=2b0f6561:3731:39)  
    at beginWork$1 (react-dom_client.js?v=2b0f6561:19791:15)  
    at performUnitOfWork (  
      react-dom_client.js?v=2b0f6561:19224:20)  
    at workLoopSync (react-dom_client.js?v=2b0f6561:19163:13)
```

3. Understanding React Error Message

- Kita harus mulai memperbaiki kesalahan dengan mencari lokasi property valueEndOfYear,
- Dibawah pesan error terdapat staktrace, ini adalah kumpulan kode yang menyebabkan error terjadi
- Disini tertulis error berasal dari results di file results.jsx yang menghasilkan komponenline 8 spasi 16



```
Uncaught  
TypeError: Cannot read properties of undefined (reading  
'valueEndOfYear')  
    at Results (Results.jsx:8:16)  
    at renderWithHooks (react-dom_client.js?v=2b0f6561:11566:26)  
    at updateFunctionComponent (react-dom_client.js?v=2b0f6561:14600:28)  
    at beginWork (react-dom_client.js?v=2b0f6561:15942:22)  
    at HTMLUnknownElement.callCallback2 (react-dom_client.js?v=2b0f6561:3672:22)  
    at Object.invokeGuardedCallbackDev (react-dom_client.js?v=2b0f6561:3697:24)  
    at invokeGuardedCallback (react-dom_client.js?v=2b0f6561:3731:39)  
    at beginWork$1 (react-dom_client.js?v=2b0f6561:19791:15)  
    at performUnitOfWork (react-dom_client.js?v=2b0f6561:19224:20)  
    at workLoopSync (react-dom_client.js?v=2b0f6561:19163:13)
```



```
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
  const initialInvestment =  
    results[0].valueEndOfYear -  
    results[0].interest -  
    results[0].annualInvestment;  
}
```

3. Understanding React Error Message

- Di kode ini results[0] tidak terdefinisi, sehingga kita perlu pergi ke kode dimana nilai result di panggil
- Result ini awalnya array yang dijadikan argumen dari sebuah function
- Function tersebut memanipulasi array result, function tersebut adalah calculateInvestmentResult yang ada di file investment.js

```
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
  const initialInvestment =  
    results[0].valueEndOfYear -  
    results[0].interest -  
    results[0].annualInvestment;  
  return (  
    <div>  
      <p>Initial Investment: {initialInvestment}</p>  
      <p>Value End of Year: {results[0].valueEndOfYear}</p>  
      <p>Interest: {results[0].interest}</p>  
      <p>Annual Investment: {results[0].annualInvestment}</p>  
    </div>  
  );  
}
```

```
export function calculateInvestmentResults(  
  expectedReturn,  
  duration,  
  results) {  
  let investmentValue = initialInvestment;  
  
  for (let i = 0; i < duration; i++) {  
    const interestEarnedInYear = investmentValue * (expectedReturn / 100);  
    investmentValue += interestEarnedInYear + annualInvestment;  
    results.push({  
      year: i + 1, // year identifier  
      interest: interestEarnedInYear, // the amount of interest earned in this year  
      valueEndOfYear: investmentValue, // investment value at end of year  
      annualInvestment: annualInvestment, // investment added in this year  
    });  
  }  
}
```

3. Understanding React Error Message

- Di file results.js , results awalnya sebuah array kosong yang akan diisi oleh function calculateInvestment
- Results tersebut digunakan untuk mengisi initialInvestment
- Kesalahan terjadi karena array results tetap menjadi array kosong

```
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
  const initialInvestment =  
    results[0].valueEndOfYear -  
    results[0].interest -  
    results[0].annualInvestment;  
  return /
```

```
export function calculateInvestmentResults(  
  expectedReturn,  
  duration,  
, results) {  
  let investmentValue = initialInvestment;  
  
  for (let i = 0; i < duration; i++) {  
    const interestEarnedInYear = investmentValue * (expectedReturn / 100);  
    investmentValue += interestEarnedInYear + annualInvestment;  
    results.push({  
      year: i + 1, // year identifier  
      interest: interestEarnedInYear, // the amount of interest earned in this year  
      valueEndOfYear: investmentValue, // investment value at end of year  
      annualInvestment: annualInvestment, // investment added in this year  
    });  
  }  
}
```

3. Understanding React Error Message

- Di file investment.js nilai array result baru diisi melalui perulangan
- Jadi ketika perulangan tidak terjadi array result tetap menjadi array kosong
- Looping ini bsru dijalankan ketika nilai duration lebih kecil dari 1 yang awalnya bernilai dari nol

```
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
  const initialInvestment =  
    results[0].valueEndOfYear -  
    results[0].interest -  
    results[0].annualInvestment;  
  return (  
    <div>  
      <p>Initial Investment: {initialInvestment}</p>  
      <p>Expected Return: {input.expectedReturn}</p>  
      <p>Duration: {input.duration}</p>  
      <p>Annual Investment: {input.annualInvestment}</p>  
      <p>Results:</p>  
      <table>  
        <thead>  
          <tr>  
            <th>Year</th>  
            <th>Interest Earned</th>  
            <th>Value End of Year</th>  
            <th>Annual Investment</th>  
          </tr>  
        </thead>  
        <tbody>  
          {results.map((result, index) => (  
            <tr>  
              <td>{index + 1}</td>  
              <td>{result.interest}</td>  
              <td>{result.valueEndOfYear}</td>  
              <td>{result.annualInvestment}</td>  
            </tr>  
          ))}  
        </tbody>  
      </table>  
    </div>  
  );  
}
```

```
export function calculateInvestmentResults({  
  expectedReturn,  
  duration,  
}, results) {  
  let investmentValue = initialInvestment;  
  
  for (let i = 0; i < duration; i++) {  
    const interestEarnedInYear = investmentValue * (expectedReturn / 100);  
    investmentValue += interestEarnedInYear + annualInvestment;  
    results.push({  
      year: i + 1, // year identifier  
      interest: interestEarnedInYear, // the amount of interest earned in this year  
      valueEndOfYear: investmentValue, // investment value at end of year  
      annualInvestment: annualInvestment, // investment added in this year  
    });  
  }  
}
```

3. Understanding React Error Message

- Nilai i awal pada perulangan adalah nol, perulangan baru bisa dijalankan ketika nilai i lebih kecil dari duration
- Nilai duration berasal dari form input
- Ketika kita memasukkan angka 0 atau negatif, atau tidak diisi, perulangan tidak akan terjadi karena kondisi tidak terpenuhi sehingga
- Array results tetap kosong dan ini tidak bisa dihindari

```
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
  const initialInvestment =  
    results[0].valueEndOfYear -  
    results[0].interest -  
    results[0].annualInvestment;  
  return (  
    <div>  
      <p>Initial Investment: {initialInvestment}</p>  
      <p>Results: {results}</p>  
    </div>  
  );  
}
```

```
export function calculateInvestmentResults(  
  expectedReturn,  
  duration,  
  results) {  
  let investmentValue = initialInvestment;  
  
  for (let i = 0; i < duration; i++) {  
    const interestEarnedInYear = investmentValue * (expectedReturn / 100);  
    investmentValue += interestEarnedInYear + annualInvestment;  
    results.push({  
      year: i + 1, // year identifier  
      interest: interestEarnedInYear, // the amount of interest earned in this year  
      valueEndOfYear: investmentValue, // investment value at end of year  
      annualInvestment: annualInvestment, // investment added in this year  
    });  
  }  
}
```

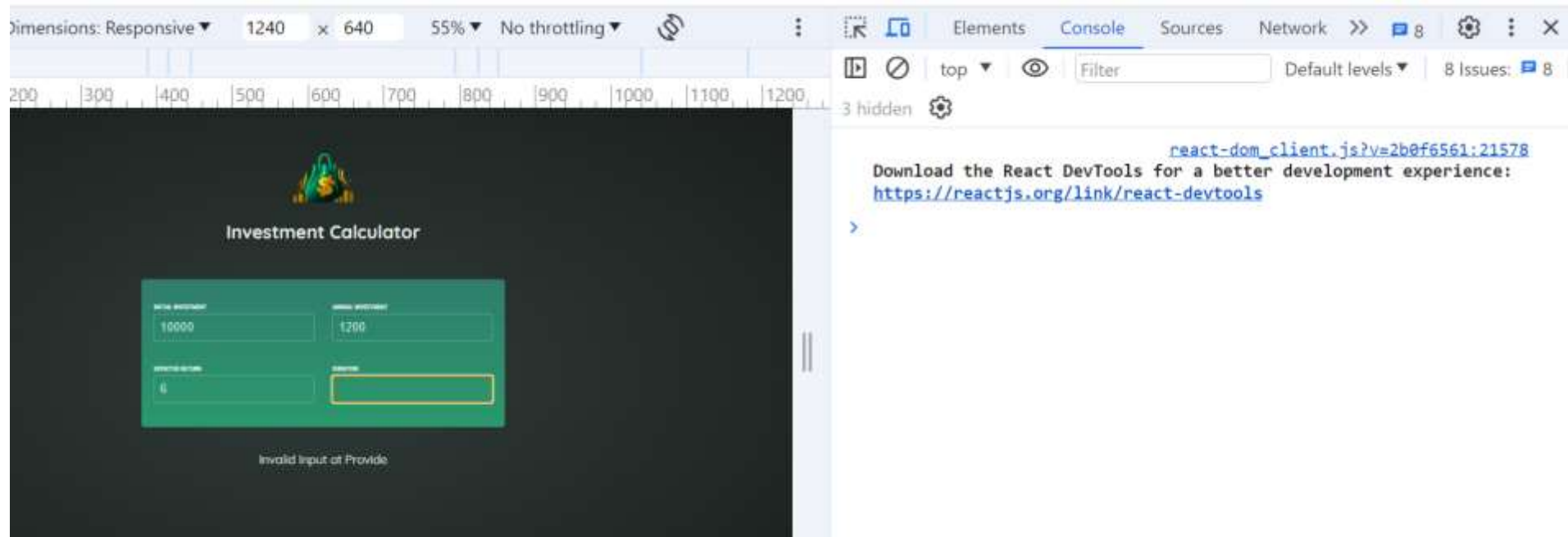

3. Understanding React Error Message

- Karena nilai duration berasal dari form, kita perlu mengecek dengan pengkondisian sebelum nilai array results digunakan
- Kita cek di results.js apakah panjang result === 0 artinya apakah array results kosong? Jika iya maka function results akan mengembalikan elemen berupa pesan kesalahan yang tampil dilayar
- Sehingga ketika array results kosong, kode initialInvestment yang menggunakan array results tidak dijalankan sehingga error menghilang dan aplikasi tidak crash
- Ketika ada kesalahan kita tidak boleh membiarkan aplikasi berhenti, aplikasi harus tetap berjalan tapi kita harus memberitahu user apa kesalahan yang terjadi

3. Understanding React Error Message

```
Codeium: Refactor | Explain | Generate JSDoc | X
export default function Results({ input }) {
  const results = [];
  calculateInvestmentResults(input, results);

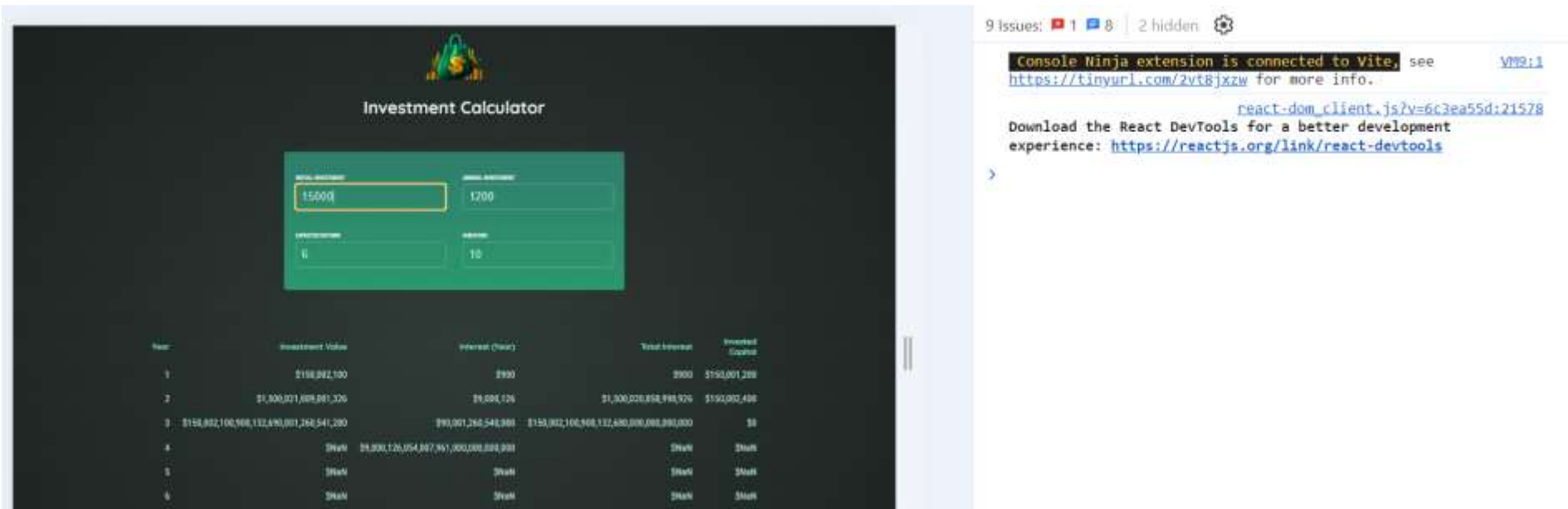
  if (results.length === 0) {
    return <p className="center">Invalid Input Dat Provide</p>;
  }
  const initialInvestment =
    results[0].valueEndOfYear -
    results[0].interest -
    results[0].annualInvestment;
}
```



4. Using the browser debugger and breakpoints

4. Using the browser debugger and breakpoints

- Tidak semua error menyebabkan pesan kesalahan
- Terkadang ada juga error yang menyebabkan kesalahan logika
- Misal jika kita ubah initial Investment menjadi 15000 , hasilnya angka di tabel akan meledak dan banyak bernilai NaN



The screenshot shows a web application titled "Investment Calculator" with a green input form. The form has four fields: "Initial Investment" (set to 15000), "Annual Interest Rate" (set to 1200), "Compounding Period" (set to 12), and "Years" (set to 10). Below the form is a table with 5 columns: Year, Investment Value, Interest (Year), Total Interest, and Invested Capital. The table shows data for years 1 through 6. The values for years 4, 5, and 6 are NaN, indicating a calculation error due to the high initial investment value.

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$158,882,100	\$900	\$900	\$159,001,288
2	\$1,300,021,889,881,326	\$1,888,126	\$1,300,028,858,988,926	\$159,002,488
3	\$158,882,106,968,132,680,001,268,541,280	\$10,001,268,548,988	\$158,882,106,968,132,680,008,088,000,000	\$8
4	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN

On the right side of the screenshot, the browser's console shows 9 issues. The first issue is a message from the Console Ninja extension: "Console Ninja extension is connected to Vite, see <https://tinyurl.com/2vi8jxzw> for more info." Below this, there is a link to "react-dom_client.js?v=6c3ea55d:21578" and a message to "Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>".

4. Using the browser debugger and breakpoints

- Dan menemukan kesalahan ini sekarang bisa sedikit lebih sulit, karena tidak ada pesan kesalahan di konsol.
- dalam hal menangani kesalahan seperti ini, sebagai langkah pertama, Anda harus selalu mencoba untuk berpikir secara logis. Bagian mana dari kode Anda yang mungkin bertanggung jawab atas kesalahan ini?
- pada awalnya, jika saya memuat ulang aplikasi ini, semuanya tampak berfungsi dengan baik. Jadi, tampaknya tidak mungkin kesalahan berasal dari function `calculateInvestmentResult`.
- Kode di function `calculateInvestmentResult` tampaknya bekerja dengan benar, karena jika tidak, kita tidak akan mendapatkan hasil yang benar di sini pada awalnya, tabel hasil investasi juga berhasil di render

4. Using the browser debugger and breakpoints

- Tampaknya juga tidak mungkin bahwa kesalahan berasal dari komponen Result di Result.jsx dan mungkin dari pembuatan komponen untuk menampilkan data tabel di sini, karena, sekali lagi, pada awalnya semuanya tampak berfungsi. Ini hanya rusak jika kita mengedit nilai tersebut.
- sehingga error yang terjadi terkait ketika kita mengambil input pengguna, yang secara teknis terjadi di komponen App (App.jsx)

4. Using the browser debugger and breakpoints

- karena kita mengangkat state yang mengelola input pengguna dari komponen UserInput (UserInput.jsx) ke komponen App., di dalam menangani perubahan, kita menyimpan nilai yang dimasukkan di dalam objek state userInput

```
Codeium: Refactor | Explain | Generate JSDoc | X
1 export default function UserInput({ onChange, userInput }) {
2   return (
3     <section id="user-input">
4       <div className="input-group">
5         <p>
6           <label>Initial Investment</label>
7           <input
8             type="number"
9             required
10            value={userInput.initialInvestment}
11            onChange={(event) =>
12              onChange('initialInvestment', event.target.value)
13            }
14          />
15        </p>
16        <p>
```

```
Codeium: Refactor | Explain | Generate JSDoc | X
function App() {
  const [userInput, setUserInput] = useState({
    initialInvestment: 10000,
    annualInvestment: 1200,
    expectedReturn: 6,
    duration: 10,
  });

  Codeium: Refactor | Explain | Generate JSDoc | X
  function handleChange(inputIdentifier, newValue) {
    setUserInput((prevUserInput) => {
      return {
        ...prevUserInput,
        [inputIdentifier]: newValue,
      };
    });
  }

  return (
    <>
      <Header />
      <UserInput userInput={userInput} onChange={handleChange} />
      <Results input={userInput} />
    </>
  );
}
```

4. Using the browser debugger and breakpoints

- Dan sepertinya kode di sini bertanggung jawab atas kesalahan ini.oleh karena itu, apa yang harus kita lakukan di sini adalah melihat kode tersebut pada saat kode tersebut dieksekusi dengan nilai yang sebenarnya digunakan selama eksekusi.
- Dan kita dapat melakukannya di browser pada proyek React ini dengan membuka tab Sources di inspect element di browser Dan di sana, Anda akan melihat bahwa di bawah host lokal, Anda akan menemukan struktur folder yang terlihat sangat mirip dengan apa yang kita dapatkan di proyek kita.
- Yang paling penting, ada folder src. Dan apa yang terjadi di sini adalah konfigurasi proyek ini pada dasarnya menghasilkan kode yang dapat digunakan oleh browser untuk menunjukkan struktur folder Anda di browser.

4. Using the browser debugger and breakpoints

The screenshot displays a web browser window with the 'Investment Calculator' application. The application has a dark theme and a green header. It features four input fields for 'Initial Investment' (10000), 'Annual Investment' (1200), 'Expected Return' (6), and 'Duration' (10). Below the inputs is a table showing the investment growth over 10 years.

Year	Investment Value	Interest (Compd)	Total Interest	Invested Capital
1	\$10,000.00	\$600.00	\$600.00	\$10,600.00
2	\$10,600.00	\$636.00	\$1,236.00	\$11,236.00
3	\$11,236.00	\$674.16	\$1,910.16	\$11,910.16
4	\$11,910.16	\$714.61	\$2,624.77	\$12,624.77
5	\$12,624.77	\$757.48	\$3,382.25	\$13,382.25
6	\$13,382.25	\$802.93	\$4,185.18	\$14,185.18
7	\$14,185.18	\$851.11	\$5,036.29	\$15,036.29
8	\$15,036.29	\$902.18	\$5,938.47	\$15,938.47
9	\$15,938.47	\$956.31	\$6,894.78	\$16,894.78
10	\$16,894.78	\$1,013.68	\$7,908.46	\$17,908.46

The Chrome DevTools Sources panel is open, showing the file structure of the application. The file 'App.jsx' is selected, and the code editor displays the following JavaScript code:

```
function App() {
  const [userInput, setUserInput] = useState({
    initialInvestment: 10000,
    annualInvestment: 1200,
    expectedReturn: 6,
    duration: 10,
  });

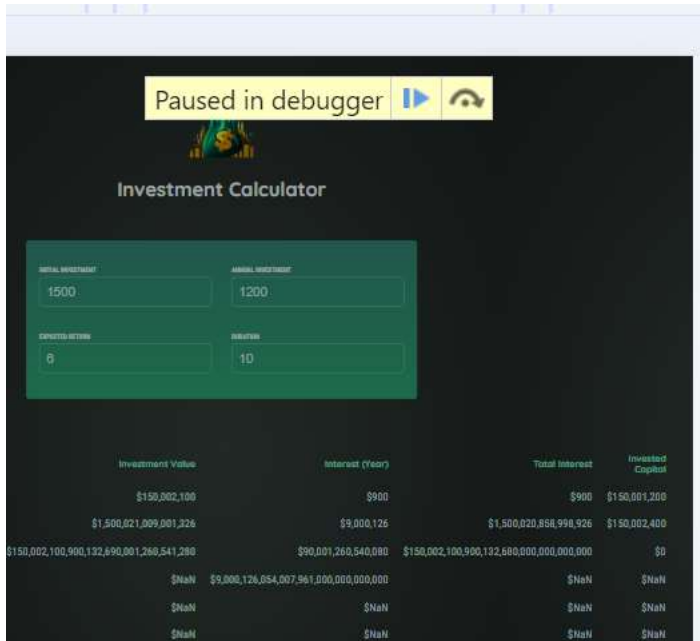
  function handleChange(inputIdentifier, newValue) {
    setUserInput((prevUserInput) => {
      return {
        ...prevUserInput,
        [inputIdentifier]: newValue,
      };
    });
  }

  return (
    <>
      <Header />
      <UserInput userInput={userInput} onChange=
      <Results input={userInput} />
    </>
  );
}
```

4. Using the browser debugger and breakpoints

- Tapi sekarang Anda bisa menyelami kode ini, Dan hal yang menyenangkan dari fitur ini adalah bahwa file ini tidak hanya berisi kode yang sama seperti yang kita tulis di vs code. Itu saja tidak akan terlalu berguna karena kita dapat melihat kode di vs code juga, tetapi sekarang kita juga dapat menempatkan break point di sini dengan mengklik nomor baris.
- Dan setelah Anda menambahkan break point, eksekusi kode akan berhenti pada baris kode ini ketika mencapai titik tersebut.

4. Using the browser debugger and breakpoints



Paused in debugger

Investment Calculator

INITIAL INVESTMENT: 1500
ANNUAL INVESTMENT: 1200
EXPECTED RETURN: 6
DURATION: 10

Investment Value	Interest (Year)	Total Interest	Invested Capital
\$150,002,100	\$900	\$900	\$150,001,200
\$1,500,021,009,001,326	\$9,000,126	\$1,500,020,858,998,926	\$150,002,400
\$150,002,100,900,132,690,001,260,541,280	\$90,001,260,540,080	\$150,002,100,900,132,680,000,000,000,000	\$0
\$NaN	\$9,000,126,054,007,961,000,000,000,000	\$NaN	\$NaN
\$NaN	\$NaN	\$NaN	\$NaN
\$NaN	\$NaN	\$NaN	\$NaN

workspace // (index) App.jsx

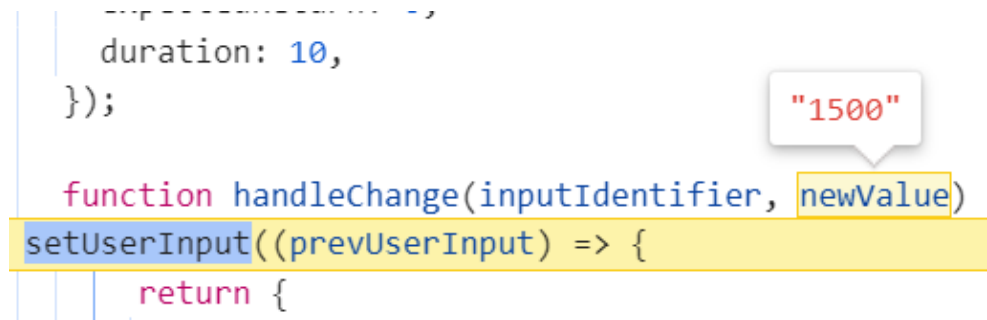
- top
 - localhost:5173
 - @vite
 - node_modules
 - src
 - assets
 - components
 - util
 - *App.jsx
 - App.jsx
 - index.css
 - index.jsx
 - index.jsx
 - (index)
 - @react-refresh

```
2  
3 import Header from './components/Header.jsx';  
4 import UserInput from './components/UserInput.js';  
5 import Results from './components/Results.jsx';  
6  
7 function App() {  
8   const [userInput, setUserInput] = useState({  
9     initialInvestment: 10000,  
10    annualInvestment: 1200,  
11    expectedReturn: 6,  
12    duration: 10,  
13  });  
14  
15  function handleChange(inputIdentifier, newValue)  
16    setUserInput((prevUserInput) => {  
17    return {  
18      ...prevUserInput,  
19      [inputIdentifier]: newValue,  
20    }  
21  }
```

4. Using the browser debugger and breakpoints

- kita juga dapat mengarahkan kursor ke beberapa bagian dari kode kita seperti parameter ini untuk melihat nilai aktual yang diterima di sini untuk eksekusi spesifik ini. Kita juga dapat melihat nilai baru yang diterima di sini, dalam hal ini adalah 1500 yang juga saya masukkan di kolom input.

```
duration: 10,  
});  
  
function handleChange(inputIdentifier, newValue) {  
  setUserInput((prevUserInput) => {  
    return {
```



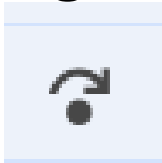


4. Using the browser debugger and breakpoints

- kita juga dapat mengarahkan kursor ke beberapa bagian dari kode kita seperti parameter ini untuk melihat nilai aktual yang diterima di sini untuk eksekusi spesifik ini. Kita juga dapat melihat nilai baru yang diterima di sini, dalam hal ini adalah 1500 yang juga saya masukkan di kolom input.

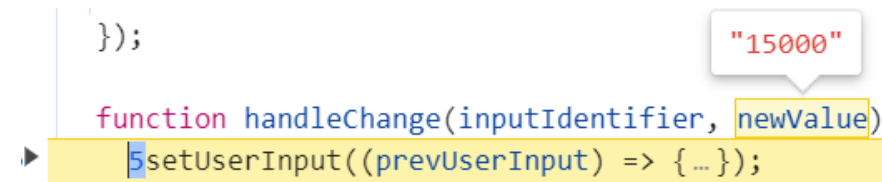
```
duration: 10,  
});  
  
function handleChange(inputIdentifier, newValue) {  
  setUserInput((prevUserInput) => {  
    return {
```

4. Using the browser debugger and breakpoints

- Dan sekarang kita dapat menggunakan kontrol di bagian bawah di sini untuk menelusuri kode kita selangkah demi selangkah.
- Dengan tombol ini, kita dapat melompat ke dalam fungsi yang akan dieksekusi jika kita sedang menjalankan beberapa fungsi yang didefinisikan di sini. 
- Dengan tombol ini, kita dapat melompat keluar dari fungsi ini, 
- dan dengan tombol ini, kita dapat langsung melompat ke pernyataan berikutnya. Dan ini memungkinkan kita untuk menelusuri kode langkah demi langkah dan melihat bagaimana segala sesuatunya berjalan di sini dan bagaimana segala sesuatunya berperilaku, dan dengan nilai mana kita bekerja. 

4. Using the browser debugger and breakpoints

- dalam kasus ini, jika saya memuat ulang lagi, memasukkan nilai initialInvestment sebesar 15000 dan mematahkannya lagi sehingga berhenti lagi, kita dapat melihat bahwa nilai baru itu tampaknya bertipe string yang ditunjukkan oleh tanda kutip di sini.



```
});  
function handleChange(inputIdentifier, newValue)  
setUserInput((prevUserInput) => {...});
```

- itulah masalahnya, karena kita kemudian melakukan operasi matematis dengan nilai string tersebut. dalam JavaScript, ketika menggunakan string dan angka yang digabungkan dalam perhitungan matematis, hal itu dapat menyebabkan kesalahan, itulah mengapa angkanya menjadi kacau dan banyak nilai NaN

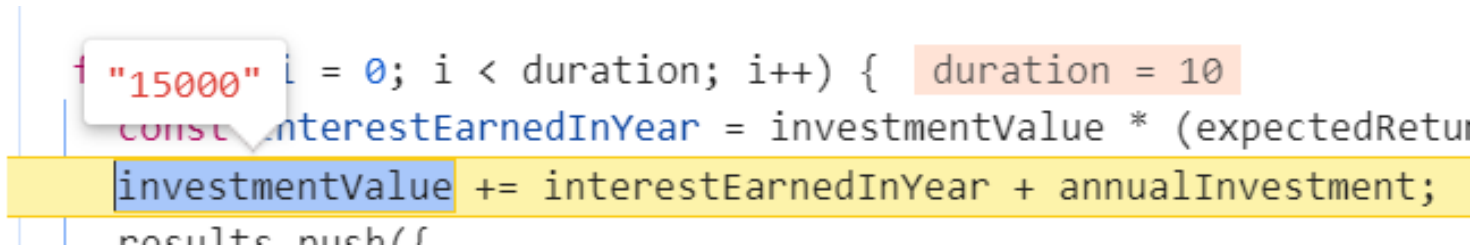
4. Using the browser debugger and breakpoints

- jika saya melanjutkan eksekusi kode, semuanya kacau . kita dapat melihat kesalahan ini sedikit lebih baik jika kita benar-benar menghapus break point ini dan menambahkannya di file investment.js,
- bisa melihat bahwa pada akhirnya eksekusi kode berhenti di sini dan kita melihat bahwa initial investment kita adalah sebuah string.

```
duration,  
, results) { results = []  
let investmentValue = initialInvestment;  
  
for (let i = 0; i < duration; i++) {
```


4. Using the browser debugger and breakpoints

- Dan ketika kita melangkah melalui kode ini dengan tombol panah ke bawah kita masuk ke dalam perulangan,
- kita dapat melihat bahwa sekarang masalahnya pasti ada di baris 17.
- Nilai investmentValue adalah "15.000"



```
for ("15000" i = 0; i < duration; i++) { duration = 10
  const interestEarnedInYear = investmentValue * (expectedRetu
  investmentValue += interestEarnedInYear + annualInvestment;
  results.push(i
```

- dan kita kemudian menambahkan dua angka, yaitu interestEarnedInYear + annualInvestment

4. Using the browser debugger and breakpoints

```
for (let i = 0; i < 900; i++) { duration = 10  
  const interestEarnedInYear = investmentValue * (expectedReturn  
  investmentValue += interestEarnedInYear + annualInvestment;  
}
```

```
for (let i = 0; i < duration; i++) { duration = 10  
  const interestEarnedInYear = investmentValue * (expectedReturn  
  investmentValue += interestEarnedInYear + annualInvestment;  
}
```

- Kita menambahkan dua angka ini ke sebuah string investmentValue.
- Itulah mengapa di browser kita melihat angka yang panjang atau bahkan bernilai NaN
- untuk memperbaikinya , kita harus memaskan newValue di function handleChange pada App.jsx di konversi ke tipe data number sebelum disimpan

4. Using the browser debugger and breakpoints

- untuk memperbaikinya , kita harus memaskan `newValue` di function `handleChange` pada `App.jsx` di konversi ke tipe data `number` sebelum disimpan. Setelah diperbaiki kode akan berjalan normal

```
Codeium: Refactor | Explain | Generate JSDoc | X
function handleChange(inputIdentifier, newValue) {
  setUserInput((prevUserInput) => {
    return {
      ...prevUserInput,
      [inputIdentifier]: +newValue,
    };
  });
}
```

4. Using the browser debugger and breakpoints

INITIAL INVESTMENT

15000

ANNUAL INVESTMENT

1200

EXPECTED RETURN

6

DURATION

10

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$17,100	\$900	\$900	\$16,200
2	\$19,326	\$1,026	\$1,926	\$17,400
3	\$21,686	\$1,160	\$3,086	\$18,600
4	\$24,187	\$1,301	\$4,387	\$19,800
5	\$26,838	\$1,451	\$5,838	\$21,000
6	\$29,648	\$1,610	\$7,448	\$22,200

5. Understanding React's Strict Mode

5. Understanding React's Strict Mode

- Nah selain pesan kesalahan, kesalahan logis dan alat pengembang browser, ada juga alat lain, bisa dibilang, dapat membantu Anda menangkap dan memperbaiki beberapa jenis kesalahan.
- Dan untuk mendemonstrasikan fitur ini, saya menyunting Results.jsx dan saya memindahkan array results ini dari fungsi komponen ke tepat di bawah pernyataan impor,

```
import { calculateInvestmentResults, formatter } from "../util/investment.js";

export default function Results({ input }) {
  const results = [];
  calculateInvestmentResults(input, results);
}
```

```
import { calculateInvestmentResults, formatter } from "../util/investment.js";
const results = [];
Codeium: Refactor | Explain | Generate JSDoc | X
export default function Results({ input }) {
  calculateInvestmentResults(input, results);
}
```

5. Understanding React's Strict Mode

- Sekarang, saya melakukan ini karena suatu alasan, jika Anda memuat ulang, semuanya tampak berfungsi. Namun ketika Anda melakukan perubahan di input initial investment, Anda akan melihat bahwa ada sesuatu yang tidak beres karena sekarang tabelnya menjadi semakin panjang dan kami juga memiliki pesan kesalahan di konsol browser, beberapa kesalahan jika Anda melihatnya, dan semua kesalahan ini adalah tentang dua child yang memiliki key yang sama. Jadi, mereka tampaknya terkait dengan array list kita di sini, karena dalam daftar itulah kita mengeluarkan elemen yang memiliki penyangga utama.

5. Understanding React's Strict Mode

INITIAL INVESTMENT
15000

ANNUAL INVESTMENT
1200

EXPECTED RETURN
6

RISK LEVEL
10

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000
6	\$22,556	\$1,209	\$5,356	\$17,200
7	\$25,109	\$1,353	\$6,709	\$18,400
8	\$27,815	\$1,507	\$8,215	\$19,600
9	\$30,684	\$1,669	\$9,884	\$20,800
10	\$33,725	\$1,841	\$11,725	\$22,000
1	\$2,260	\$60	-\$8,940	\$11,200
2	\$1,596	\$136	-\$8,804	\$12,400

Console Ninja extension is connected to Vite, see <https://tinyurl.com/2vt8jxzw> for more info.

react-dom_client.js?v=b8d7ee19:21578

Download the React DevTools for a better development experience:
<https://reactjs.org/link/react-devtools>

Warning: Encountered two children with the same key, `1`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.
at tbody
at table
at Results (<http://localhost:5173/src/components/Results.jsx:19:35>)
at App (<http://localhost:5173/src/App.jsx:24:37>)

Warning: Encountered two children with the same key, `2`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.
at tbody
at table
at Results (<http://localhost:5173/src/components/Results.jsx:19:35>)
at App (<http://localhost:5173/src/App.jsx:24:37>)

Warning: Encountered two children with the same key, `3`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.
at tbody
at table
at Results (<http://localhost:5173/src/components/Results.jsx:19:35>)
at App (<http://localhost:5173/src/App.jsx:24:37>)

5. Understanding React's Strict Mode

- Dan tentu saja cukup jelas di sini bahwa masalahnya adalah bahwa daftar ini terus bertambah panjang dan semakin panjang, alih-alih dihapus dan diganti saat kita mengedit input tersebut. Sekarang, kita dapat kembali menggunakan debugger atau melihat lebih dekat kode kita untuk mencari tahu apa yang salah di sini. Tetapi sebelum kita melakukan itu, perlu dicatat bahwa kita hanya melihat masalah ini segera setelah kita mulai mengedit beberapa nilai input di initial investment yang mungkin bisa dibilang adalah sesuatu yang akan kita lakukan ketika menguji aplikasi ini. Namun demikian, ini bukan kesalahan yang langsung kami lihat. Dan setidaknya ini adalah kesalahan yang bisa langsung ditampilkan oleh React saat aplikasi dijalankan.

5. Understanding React's Strict Mode

- Sebagai contoh, kita dapat membungkus komponen App dengan tag pembuka dan penutup komponen StrictMode.

```
index.js src/index.js
1  import ReactDOM from "react-dom/client";
2  import { StrictMode } from "react";
3
4  import App from "../App.jsx";
5  import "../index.css";
6
7  ReactDOM.createRoot( document.getElementById("root") ).render(
8    <StrictMode>
9      <App />
10    </StrictMode>
11  );
12
```

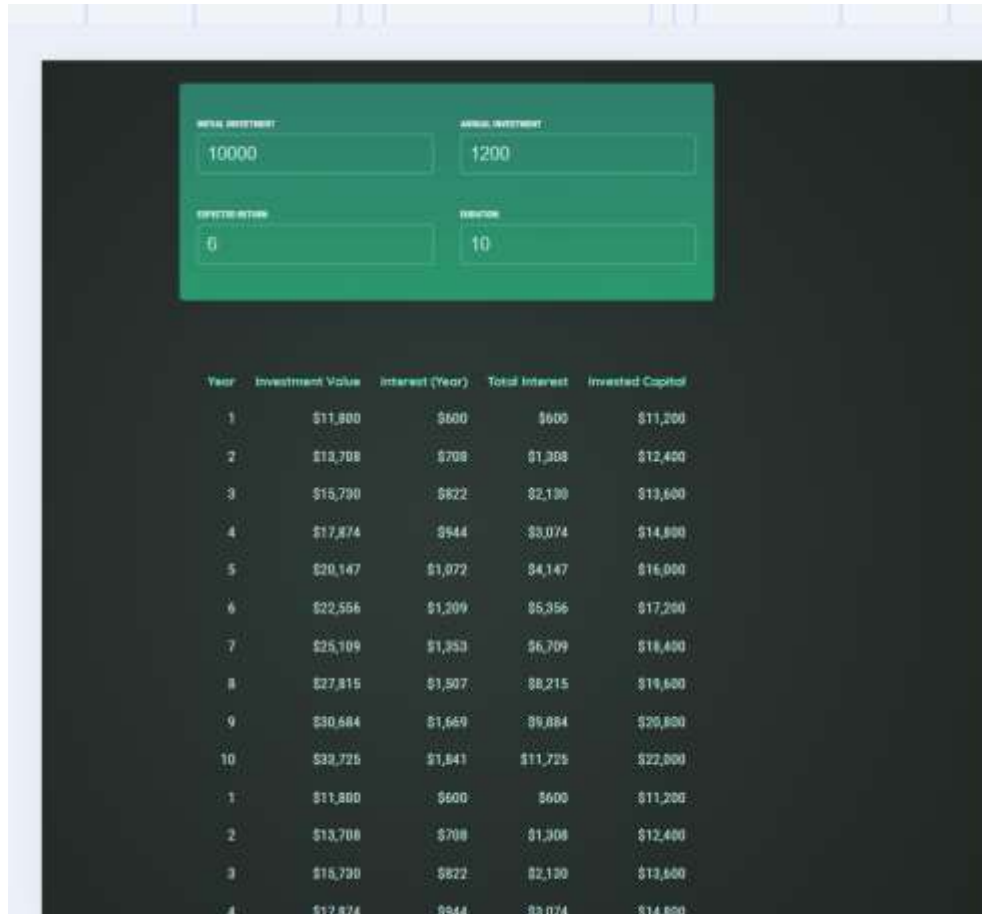
5. Understanding React's Strict Mode

- StrictMode melakukan beberapa hal di belakang layar yang dapat membantu kita menangkap masalah tertentu dalam aplikasi
- Sebagai contoh, salah satu hal terpenting yang dilakukan oleh komponen StrictMode adalah StrictMode akan mengeksekusi setiap fungsi komponen dua kali, bukan hanya satu kali.
- ini hanya dilakukan selama pengembangan. Jadi, jika Anda menyiapkan aplikasi Anda untuk production dan Anda akan mengunggahnya ke server, StrictMode tidak akan lagi mengeksekusi setiap komponen dua kali karena hal ini tentu saja akan sedikit memengaruhi kinerja aplikasi Anda.

5. Understanding React's Strict Mode

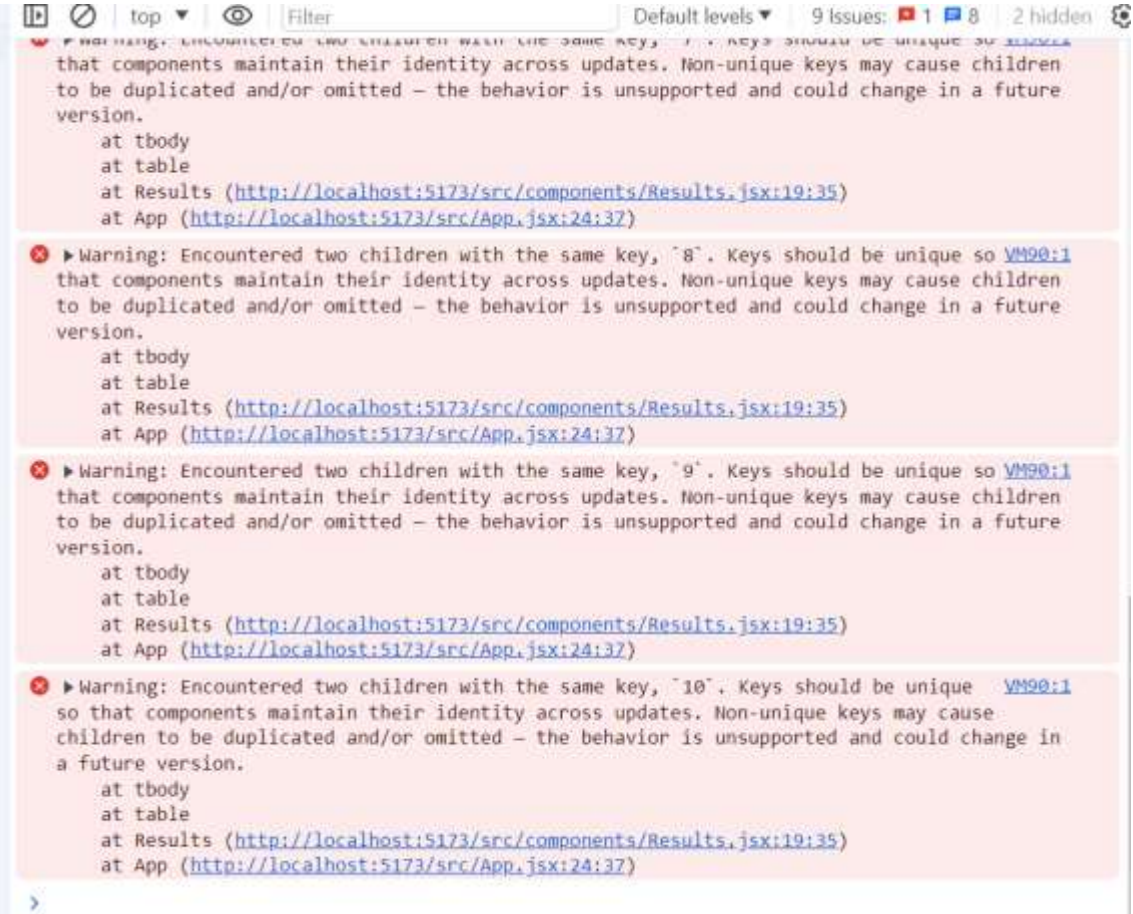
- ketika setiap komponen dieksekusi dua kali, ini membantu Anda menangkap kesalahan. karena sekarang jika saya memuat ulang, setiap komponen dieksekusi dua kali termasuk komponen App, saya mulai dengan dua kali tabel di sini

5. Understanding React's Strict Mode



The screenshot shows a web application with a green header and a dark blue table. The header has four input fields: 'INITIAL INVESTMENT' (10000), 'ANNUAL INVESTMENT' (1200), 'EXPECTED RETURN' (0), and 'PERIOD' (10). The table below has five columns: 'Year', 'Investment Value', 'Interest (Year)', 'Total Interest', and 'Invested Capital'. The table contains 10 rows of data, showing the growth of an investment over time.

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,790	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000
6	\$22,596	\$1,209	\$5,356	\$17,200
7	\$25,109	\$1,353	\$6,709	\$18,400
8	\$27,815	\$1,507	\$8,215	\$19,600
9	\$30,684	\$1,669	\$9,884	\$20,800
10	\$32,725	\$1,841	\$11,725	\$22,000



The screenshot shows the Chrome DevTools console with several warnings. The warnings are about encountering two children with the same key, which is a common issue in React when rendering lists. The warnings are located in the 'Results' component at line 19:35 and in the 'App' component at line 24:37. The warnings are repeated for keys '8', '9', and '10'.

Warning: Encountered two children with the same key, `8`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.

Warning: Encountered two children with the same key, `9`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.

Warning: Encountered two children with the same key, `10`. Keys should be unique so that components maintain their identity across updates. Non-unique keys may cause children to be duplicated and/or omitted – the behavior is unsupported and could change in a future version.

5. Understanding React's Strict Mode

- sepertinya saya memiliki beberapa kesalahan dalam kode saya yang menyebabkan tabel ini bertambah dan bertambah. Jadi, tentu saja, hal itu tidak memperbaiki masalah, dan juga belum tentu memberi tahu saya, apa yang menyebabkan masalah ini, tetapi langsung muncul ke permukaan bahwa ada masalah.
- Saya bahkan tidak perlu menyunting input initial investment untuk mengetahui bahwa ada sesuatu yang salah.
- Namun, berkat StrictMode yang diaktifkan dan melingkupi seluruh aplikasi saya, kesalahan ini bisa langsung terlihat.

5. Understanding React's Strict Mode

- Dan kesalahan di sini berasal dari fakta bahwa dalam komponen Results saya, saya membuat array results ini di luar fungsi komponen dan array results dieksekusi hanya sekali karena fungsi komponen ini akan dieksekusi ulang oleh React setiap kali state pada komponen yang terlihat berubah,

```
import { calculateInvestmentResults, formatter } from "../util/investment.js";
const results = [];
Codeium: Refactor | Explain | Generate JSDoc | X
export default function Results({ input }) {
  calculateInvestmentResults(input, results);
```

5. Understanding React's Strict Mode

- jadi setiap kali input dari initial investment berubah dalam kasus ini, kode lain di dalam berkas ini, sehingga pembuatan array ini tidak akan dieksekusi ulang, yang hanya dieksekusi satu kali. Dan oleh karena itu, yang terjadi adalah sebuah array dibuat dan kemudian di dalam fungsi `calculateInvestmentResults`, semakin banyak item ditambahkan ke satu array yang sama di dalam memori karena array tersebut tidak pernah di-reset
- Itulah mengapa kita harus mengambil array results ini dan memindahkannya kembali ke dalam komponen Results sehingga sebuah array baru dibuat ulang setiap kali fungsi komponen ini dijalankan.

5. Understanding React's Strict Mode

```
import { calculateInvestmentResults, formatter } from "../util/investment.js";  
const results = [];  
Codeium: Refactor | Explain | Generate JSDoc | ✕  
export default function Results({ input }) {  
  calculateInvestmentResults(input, results);  
}
```

```
import { calculateInvestmentResults, formatter } from "../util/investment.js";  
Codeium: Refactor | Explain | Generate JSDoc | ✕  
export default function Results({ input }) {  
  const results = [];  
  calculateInvestmentResults(input, results);  
}
```

5. Understanding React's Strict Mode

- Dan dengan itu, kita dapat melihat bahwa jika kita memuat ulang aplikasi ini, kesalahan ini akan hilang dan tidak ada kesalahan di konsol.

5. Understanding React's Strict Mode

Investment Calculator

INITIAL INVESTMENT	ANNUAL INVESTMENT
15000	1200
EXPECTED RETURN	DURATION
6	10

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$17,100	\$900	\$900	\$16,200
2	\$19,326	\$1,026	\$1,926	\$17,400
3	\$21,686	\$1,160	\$3,086	\$18,600
4	\$24,187	\$1,301	\$4,387	\$19,800
5	\$26,838	\$1,451	\$5,838	\$21,000
6	\$29,648	\$1,610	\$7,448	\$22,200
7	\$32,627	\$1,779	\$9,227	\$23,400
8	\$35,785	\$1,958	\$11,185	\$24,600
9	\$39,132	\$2,147	\$13,332	\$25,800
10	\$42,680	\$2,348	\$15,680	\$27,000

Console Ninja extension is connected to Vite, see <https://tinyurl.com/2vt8jxzw> for more info. VM116:1

react-dom_client.js?v=b8d7ee19:21578

Download the React DevTools for a better development experience:

<https://reactjs.org/link/react-devtools>

>

6. Using the react devtools (browser extension)

6. Using the react devtools (browser extension)

- ada ekstensi tambahan yang dapat di instal ke dalam browser Anda untuk membuat hidup Anda sebagai pengembang React menjadi lebih mudah. Dan ekstensi tersebut bernama React developer tools.



React Developer Tools

Add to Chrome

 **Featured** 4.0 ★ (1.5K ratings)

Extension

Developer Tools

4,000,000 users

6. Using the react devtools (browser extension)

- ada ekstensi tambahan yang dapat di instal ke dalam browser Anda untuk membuat hidup Anda sebagai pengembang React menjadi lebih mudah. Dan ekstensi tersebut bernama React developer tools.



React Developer Tools

Add to Chrome

 **Featured** 4.0 ★ (1.5K ratings)

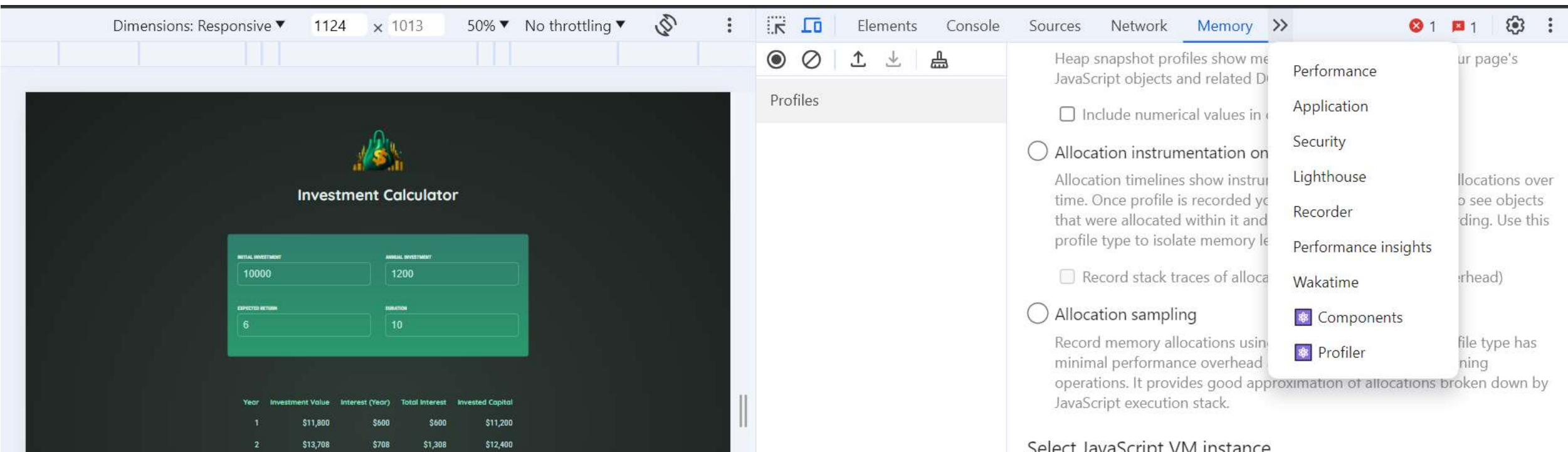
Extension

Developer Tools

4,000,000 users

6. Using the react devtools (browser extension)

- jika Anda membuka inspect element Anda akan menemukan dua tab baru, yaitu tab components dan profiler



The screenshot displays the React DevTools browser extension interface. The main window shows an "Investment Calculator" application with input fields for Initial Investment (10000), Annual Investment (1200), Expected Return (6), and Duration (10). Below the inputs is a table showing investment results over two years.

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400

The right-hand panel shows the "Memory" tab selected. A dropdown menu is open, listing various tools: Performance, Application, Security, Lighthouse, Recorder, Performance insights, Wakatime, Components, and Profiler. The "Components" and "Profiler" options are highlighted with a blue background.

Below the dropdown menu, the text "Select JavaScript VM instance" is visible.

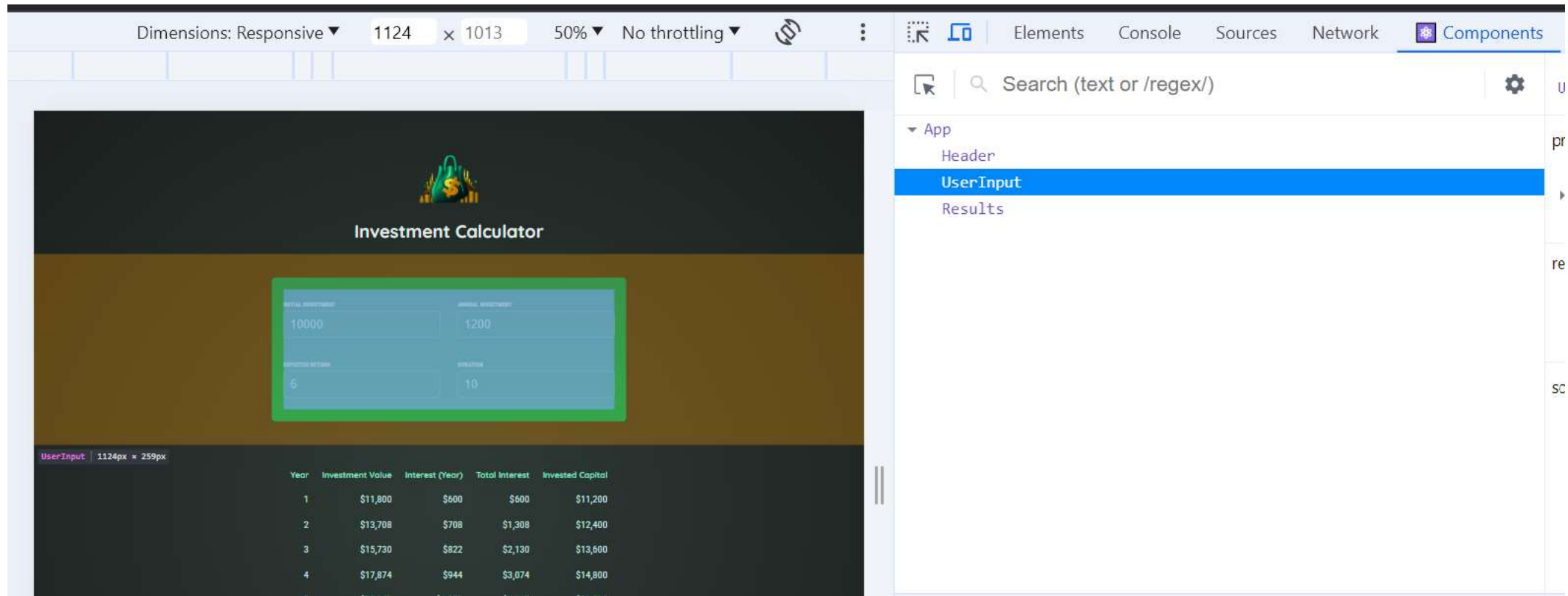
6. Using the react devtools (browser extension)

- jika Anda membuka inspect element Anda akan menemukan dua tab baru, yaitu tab components dan profiler
- tab profiler ini berfungsi menemukan dan memperbaiki masalah performa pada aplikasi React Anda, dan oleh karena itu kita akan melihat lebih dekat ada tab ini di akhir kursus saat kita, secara umum, melihat lebih dekat pada performa React dan cara meningkatkan dan mengoptimalkannya. Untuk saat ini, tab komponen inilah yang akan digunakan.

6. Using the react devtools (browser extension)

- Pada tab components Anda menemukan pohon komponen aplikasi Anda. Jadi komponen aplikasi berupa Header, UserInput, dan komponen Results sebagai anak. Dan jika Anda mengarahkan kursor ke salah satu komponen
- ini, komponen ini juga disorot di sebelah kiri layar di elemen yang dimaksud. Jadi, Anda dapat dengan cepat melihat bagian mana dari UI yang dikontrol dan dirender oleh komponen yang bisa sangat berguna untuk menganalisis dan memahami pohon komponen yang lebih kompleks, dan antarmuka pengguna yang lebih kompleks.

6. Using the react devtools (browser extension)

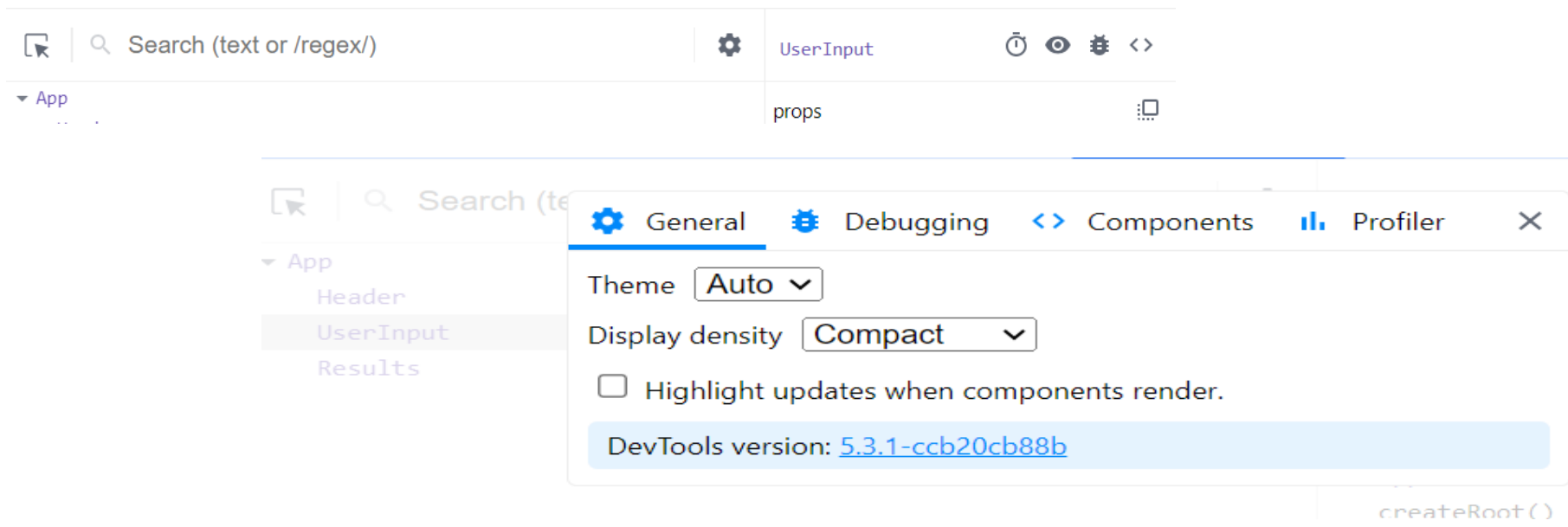


The screenshot shows the React DevTools interface with the Components panel open. The 'UserInput' component is highlighted in blue. The application is an 'Investment Calculator' with a dark theme. The input fields are highlighted with a green border. The table below shows the calculated results for the input values.

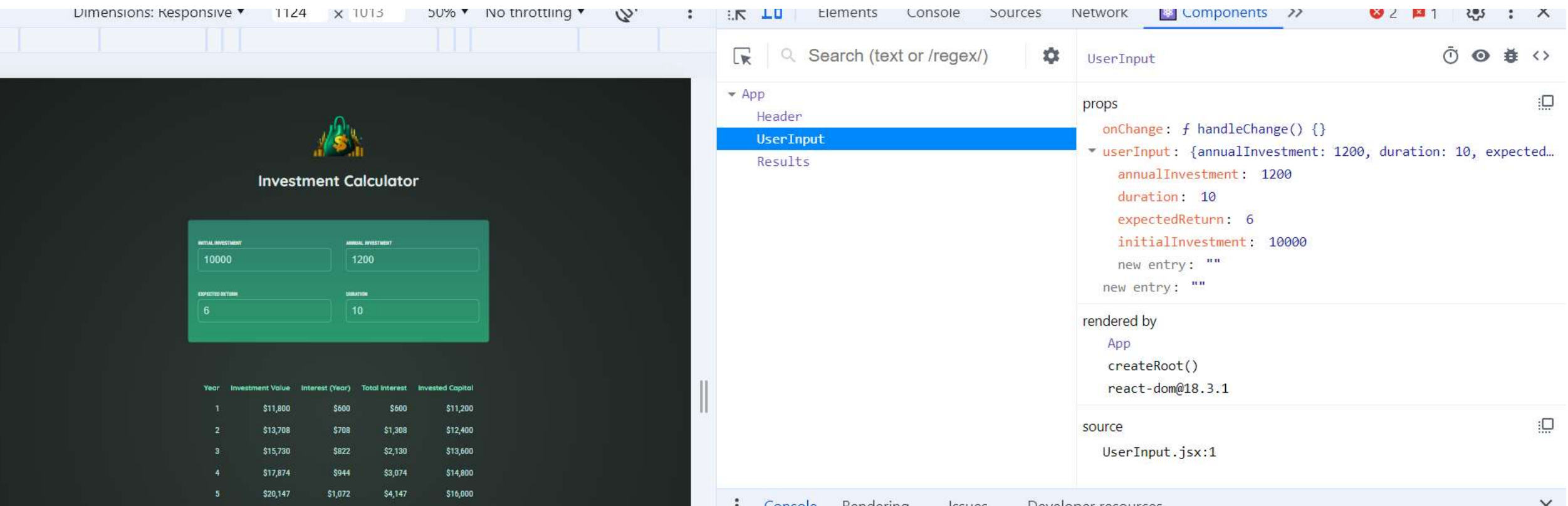
Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000

6. Using the react devtools (browser extension)

- Jika Anda mengklik komponen roda gigi, Anda juga dapat mengontrol tampilan alat pengembang. Contohnya, mode warna, densitas tampilan, dan kemudian juga untuk pohon komponen ini, juga jika pohon komponen harus diperluas secara default, dan beberapa hal lainnya.



6. Using the react devtools (browser extension)

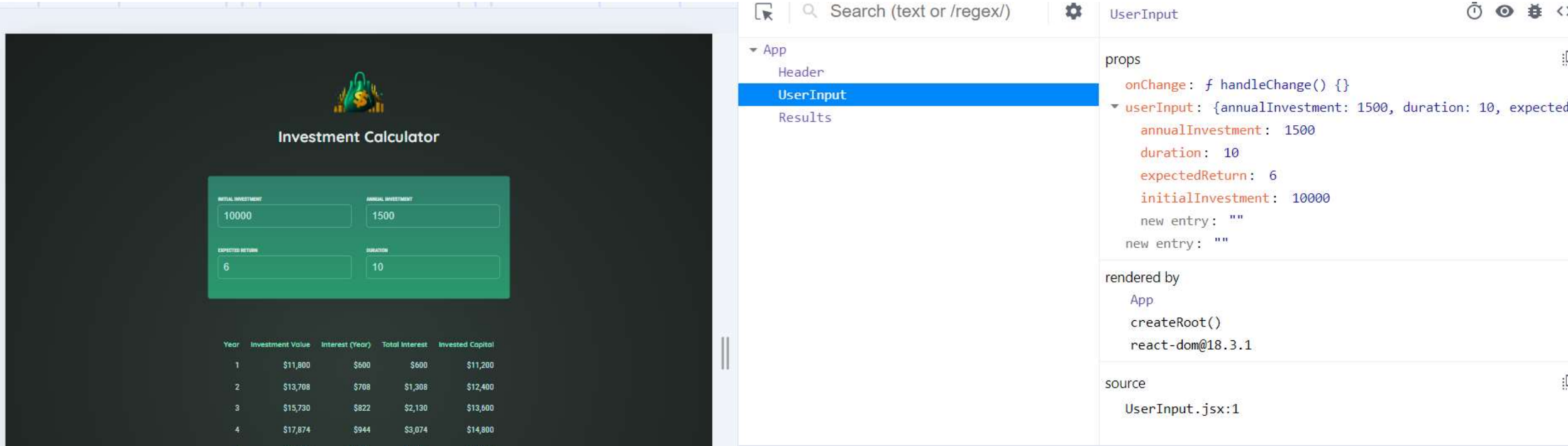


The screenshot displays the React DevTools interface. On the left, the 'App' component tree shows the hierarchy: App > Header > **UserInput** > Results. The 'UserInput' component is selected, and its props are shown on the right. The props include an `onChange` function and a `userInput` object with values for `annualInvestment`, `duration`, `expectedReturn`, and `initialInvestment`. Below the props, the 'rendered by' section shows the component was rendered by `App` using `createRoot()` from `react-dom@18.3.1`. The 'source' section shows the file `UserInput.jsx:1`.

The application window shows an 'Investment Calculator' with a green header and a dark background. It features four input fields: 'INITIAL INVESTMENT' (10000), 'ANNUAL INVESTMENT' (1200), 'EXPECTED RETURN' (6), and 'DURATION' (10). Below the inputs is a table showing the investment results over 5 years.

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000

6. Using the react devtools (browser extension)



The screenshot shows a web application titled "Investment Calculator" with a green input form and a table of results. The form contains the following values:

INITIAL INVESTMENT	ANNUAL INVESTMENT	EXPECTED RETURN	DURATION
10000	1500	6	10

Below the form is a table showing the investment results over 4 years:

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800

The React DevTools component inspector is open on the right, showing the component tree with "UserInput" selected. The props for the selected component are:

```
props
  onChange: f handleChange() {}
  userInput: {annualInvestment: 1500, duration: 10, expectedReturn: 6, initialInvestment: 10000, new entry: ""}
```

The component is rendered by the App component, created by createRoot() from react-dom@18.3.1. The source file is UserInput.jsx:1.

6. Using the react devtools (browser extension)

- Jika komponen mengelola state, seperti komponen App Anda juga akan melihatnya daftar state/hook .
- Anda dapat melihat bahwa komponen App dalam aplikasi ini menggunakan hook use state. untuk membuat satu state, satu nilai state, dan Anda juga dapat mengedit nilai state tersebut di sini.
- Contohnya, saya bisa mengubah duration menjadi 12 di sini, dan melihat bagaimana hal itu tercermin dalam UI.

6. Using the react devtools (browser extension)

The screenshot shows a web application titled "Investment Calculator" with a green input form and a table of results. The form contains the following values:

- INITIAL INVESTMENT: 10000
- ANNUAL INVESTMENT: 1200
- EXPECTED RETURN: 6
- DURATION: 12

The table below shows the investment growth over 5 years:

Year	Investment Value	Interest (Year)	Total Interest	Invested Capital
1	\$11,800	\$600	\$600	\$11,200
2	\$13,708	\$708	\$1,308	\$12,400
3	\$15,730	\$822	\$2,130	\$13,600
4	\$17,874	\$944	\$3,074	\$14,800
5	\$20,147	\$1,072	\$4,147	\$16,000

React DevTools is open on the right, showing the component tree and state. The component tree lists: App, Header, UserInput, and Results. The state of the App component is:

```
State: {annualInvestment: 1200, duration: 12, expectedReturn: 6, initialInvestment: 10000}
```

The state is rendered by `createRoot()` from `react-dom@18.3.1`. The source file is `App.jsx:8`.

6. Using the react devtools (browser extension)

- react developer tools adalah tempat yang tepat untuk memahami dengan cepat bagaimana tampilan pohon komponen Anda, bagian mana dari UI yang dikontrol oleh komponen yang mana,
- Anda juga dapat menggunakannya untuk mengedit pohon komponen Anda, dan dengan cepat merasakan bagaimana perubahan pada props atau state dapat direfleksikan pada UI.