

# Loeng 13

Klientrakendus (JavaScript)

# Kas saaks ilma JavaScriptita?

Home (Front Page)

ois2.ttu.ee/uusois/uus\_ois2.tud\_leht

Logi sisse

**TAL TECH**  
Tallinna Tehnikaülikooli  
õppeinfosüsteem

Üldinfo



ESILEHT

ÕPPEAINED

ÕPPEKAVAD

TUNNIPLAANID

AKADEEMILINE KALENDER



Rakenduskõrgharidusõpe

**Bakalaureuseõpe**

Magistriõpe

Doktoriõpe

Integreeritud õpe

**INFOTEHNOLOOGIA TEADUSKOND**

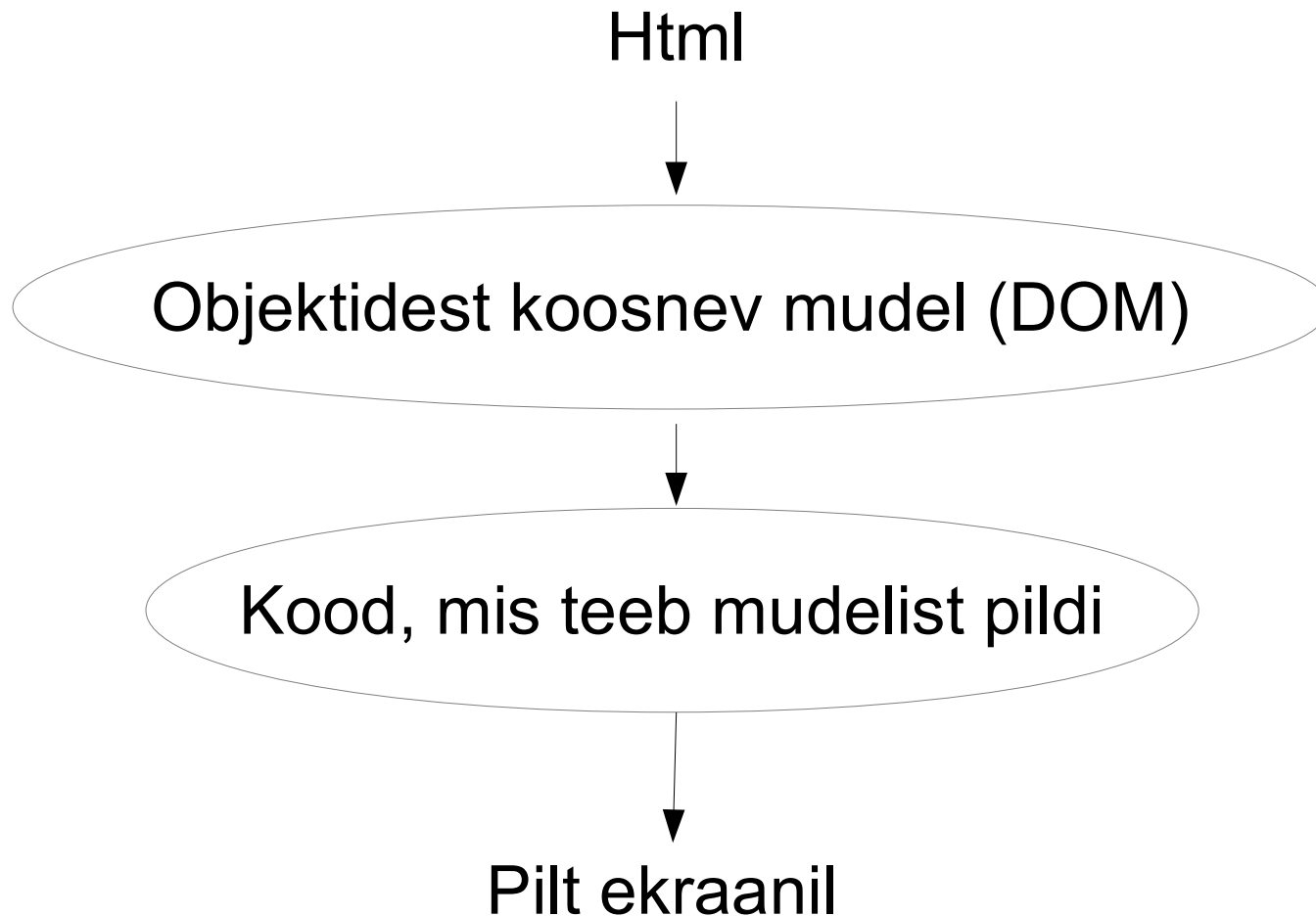
Õppekava nimetus ▲ ▼	Õppekava kood ▲ ▼	Õppekavaversiooni kood ▲ ▼
+ Arvutisüsteemid	IACB17	IACB17/17
+ IT süsteemide administreerimine	IAAB17	IAAB17/19
- IT süsteemide arendus	IADB17	IADB17/17
<div><div>Õppekava juht/programmijuht:</div><div>Õppetöö keel:</div><div>Maht (EAP):</div><div>Nominaalne õppeaeg:</div><div>Õppekava eesmärgid:</div></div> <div>Meelis Antoi</div> <div>eesti keel</div> <div>180</div> <div>6 semestrit</div> <div>Õppekava eesmärk on ette valmistada teoreetiliste teadmiste ja praktiliste oskustega tarkvaraarendajaid, töötamaks kaasaegse info- ja kommunikatsioonitehnoloogia valdkonnas tegelevates ettevõtetes. Õppekava lõpetaja on võimeline töötama suuremahuliste IT projektide arenduses nii iseseisvalt kui ka suuremas meeskonnas ja/või jätkama õpinguid info-ja kommunikatsioonitehnoloogia valdkonna magistriõppes.</div>		
+ Informaatika	IAIB17	IAIB17/19
+ Küberturbe tehnoloogiad	IVSB17	IVSB17/17
+ Äriinfotehnoloogia	IABB17	IABB17/17

**INSENERITEADUSKOND**

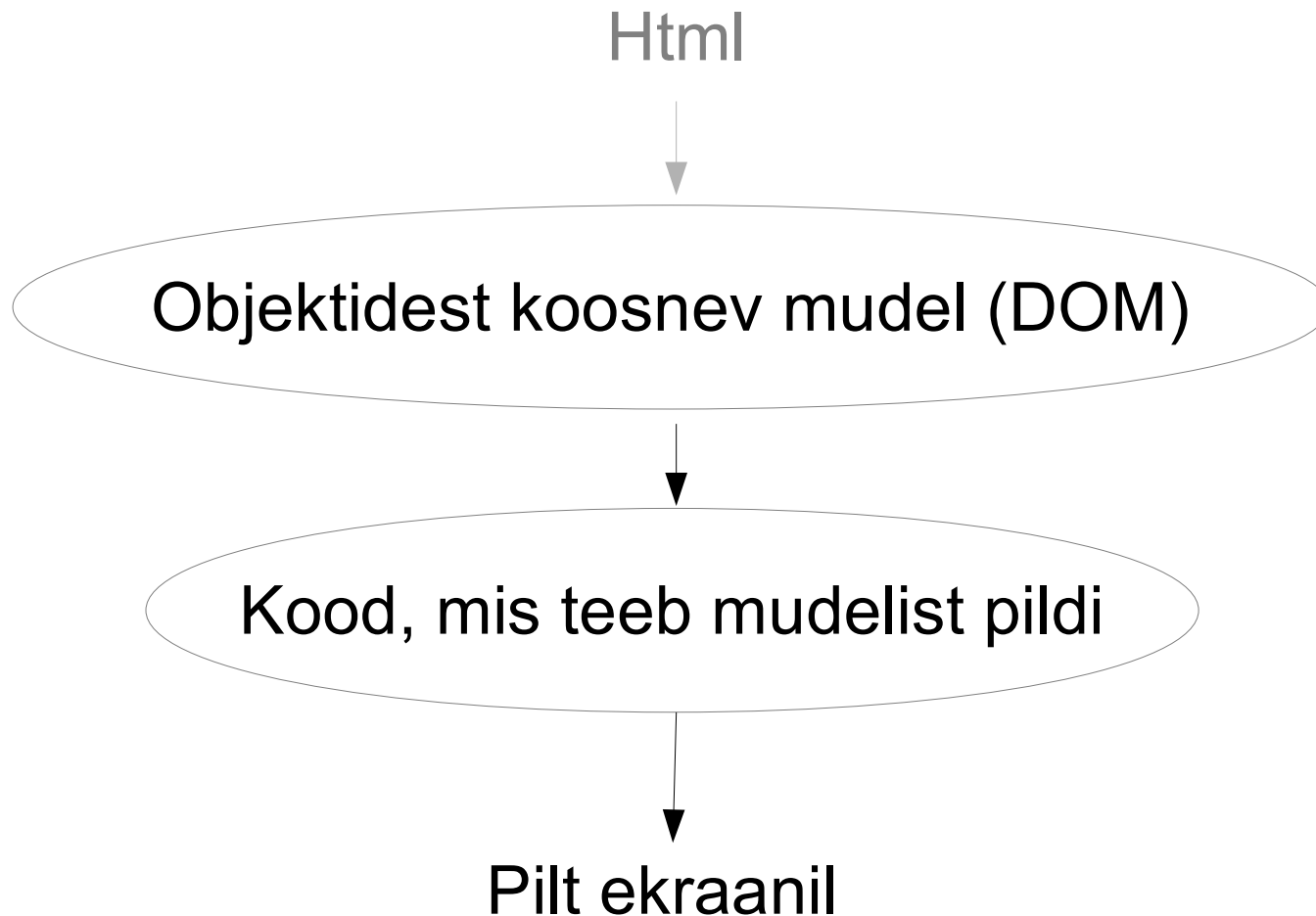
[Vaata rohkem..](#)

# Mida teeb JavaScripti raamistik?

# Pildi joonistamine Html-ist



# Pildi joonistamine JavaScriptist



# DOM-i muutmine

```
<html>
```

```
<body>
```

```
<script>
```

```
    const p = document.createElement('h1');
```

```
    h1.innerText = 'Hello';
```

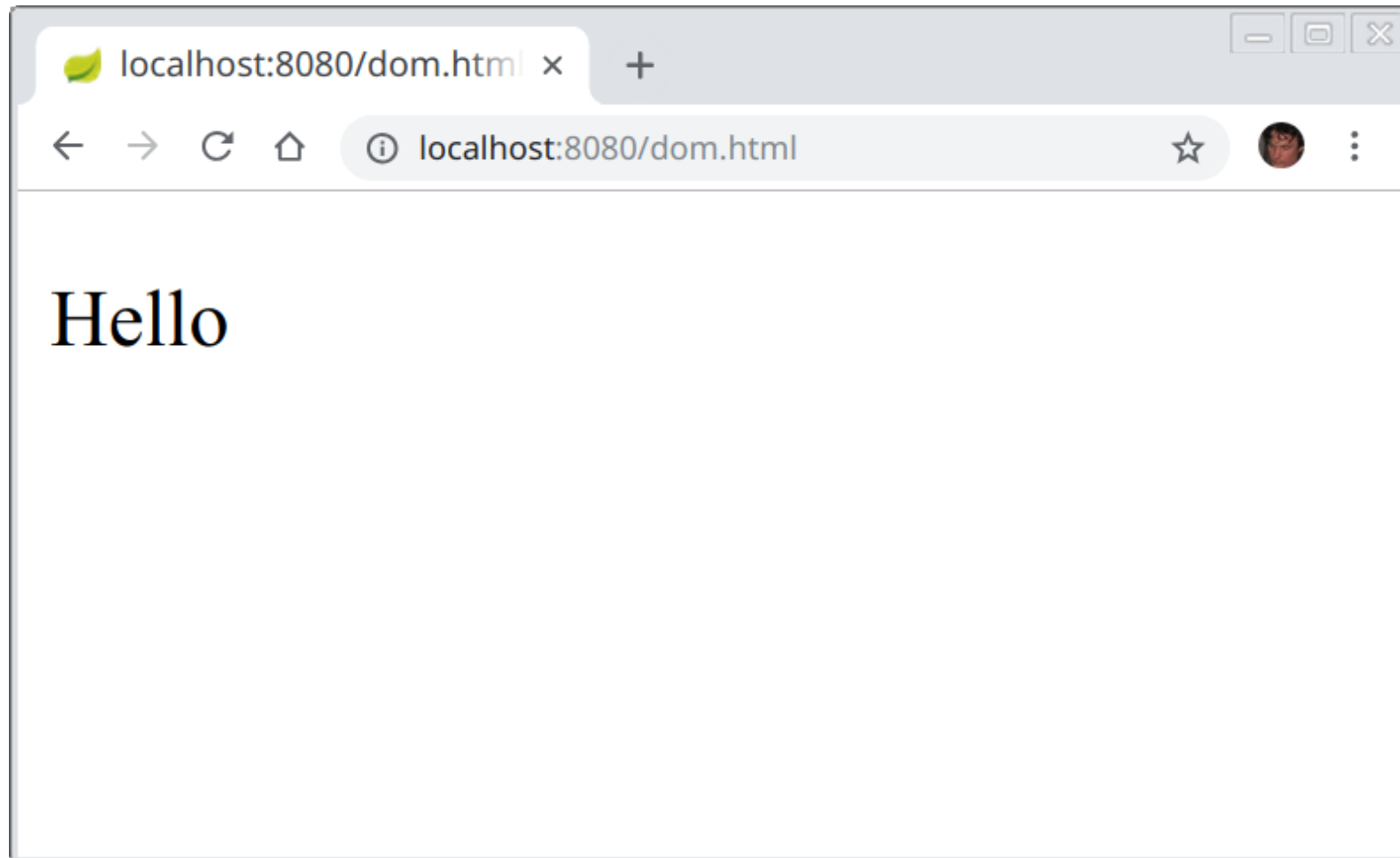
```
    document.body.appendChild(h1);
```

```
</script>
```

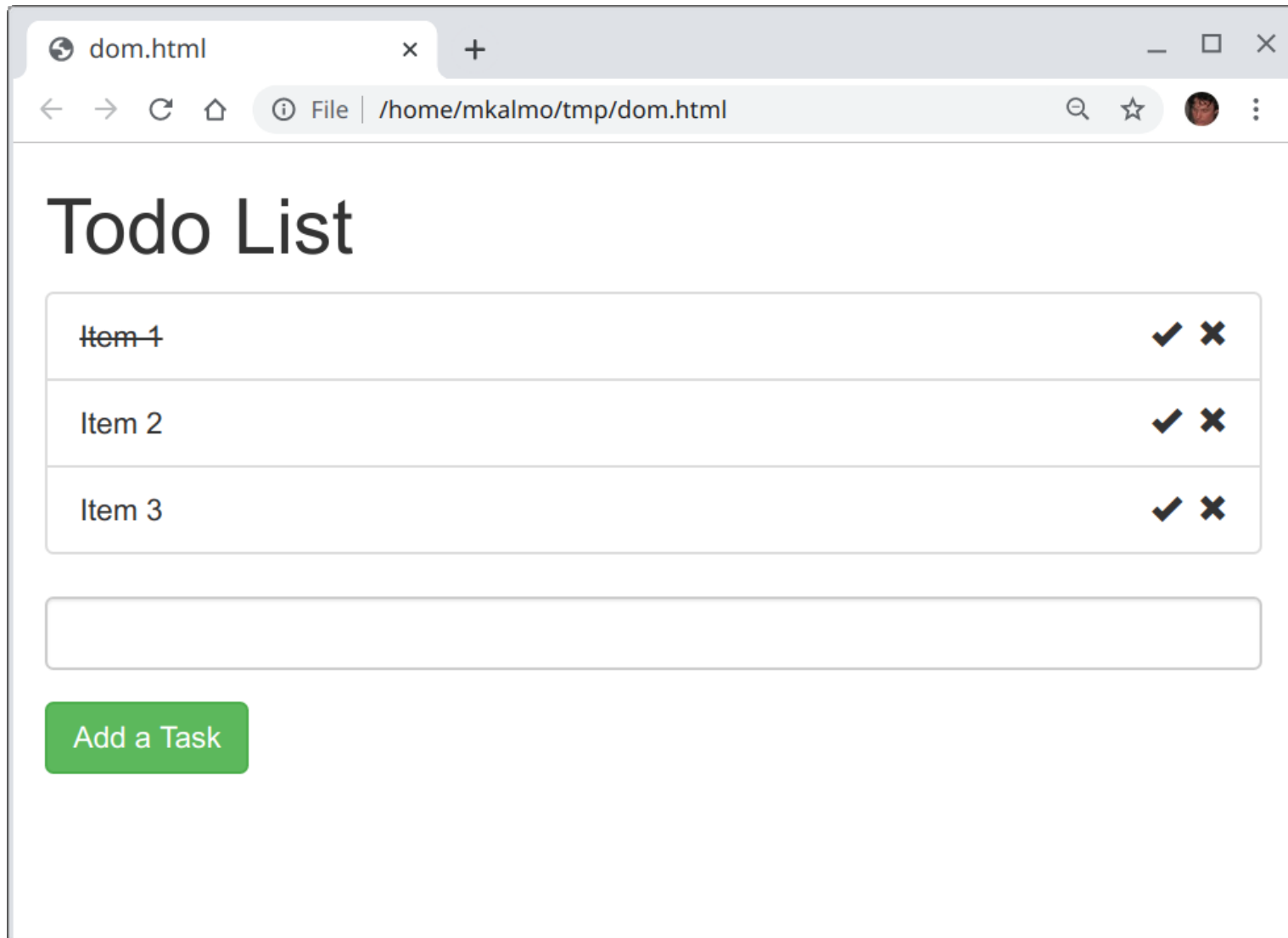
```
</body>
```

```
</html>
```

# DOM-i muutmine



# DOM-i muutmine (näide)





# JavaScripti raamistikud

- Angular, React, Vue, ...

# Raamistik (termin)

# Raamistik (termin)

- Inversion of Control (IOC)
- Teeb osa arhitektuurilisi otsuseid

# Vue /vjuː/

- Kood ja mallid eraldi
- Html-il põhinev malli keel
- Ei vaja tingimata ehitamist (build), CDN on piisav

# Vue (mall)

```
<body>
<div class="col-sm-offset-3 col-sm-6" id="app">
  ...

  <ul class="list-group">
    <li class="list-group-item" v-for="item in items">
      <span>{{ item.text }}</span>
      ...
    </li>
  </ul>

  <button type="button"
    v-on:click="addNewItem"
    class="btn btn-success">Add a Task</button>

  <script src="vue-code.js"></script>
```

# Veebitehnoloogiate aine

```
<table class="list-table">
  <thead>
    <tr>
      <th scope="col">Eesnimi</th>
      <th scope="col">Perekonnanimi</th>
      <th scope="col">Telefon</th>
    </tr>
  </thead>
  <tbody>
    <tr tpl-foreach="$persons as $person">
      <td>
        {{ $person->firstName }}
      </td>
      <td>
        {{ $person->lastName }}
      </td>
    </tr>
  </tbody>
</table>
```

# Vue (kontroller)

```
const vm = new Vue({  
  el: '#app',  
  
  data: {  
    newItemText: '',  
    items: [ ... ]  
  },  
  
  methods: {  
    addNewItem: () => {  
      ...  
    },  
  
    markAsDone: (id) => {  
      ...  
    },  
  
    ...  
  }  
})
```

# Vue (näide)



# Vue tööpõhimõte

- DOM-i järeltöötlus

index.html

```
<script src="https://unpkg.com/.../vue.js"></script>

...

<ul class="list-group">
  <li class="list-group-item" v-for="item in items">
    <span>{{ item.text }}</span>
  </li>
</ul>

...

<script src="vue-code.js"></script>
```

# React

```
<script src="https://unpkg.com/...
```

```
...
```

```
<div id="root"></div>
```

```
<script>
```

```
  const li = React.createElement("li", {}, 'Item 1');
```

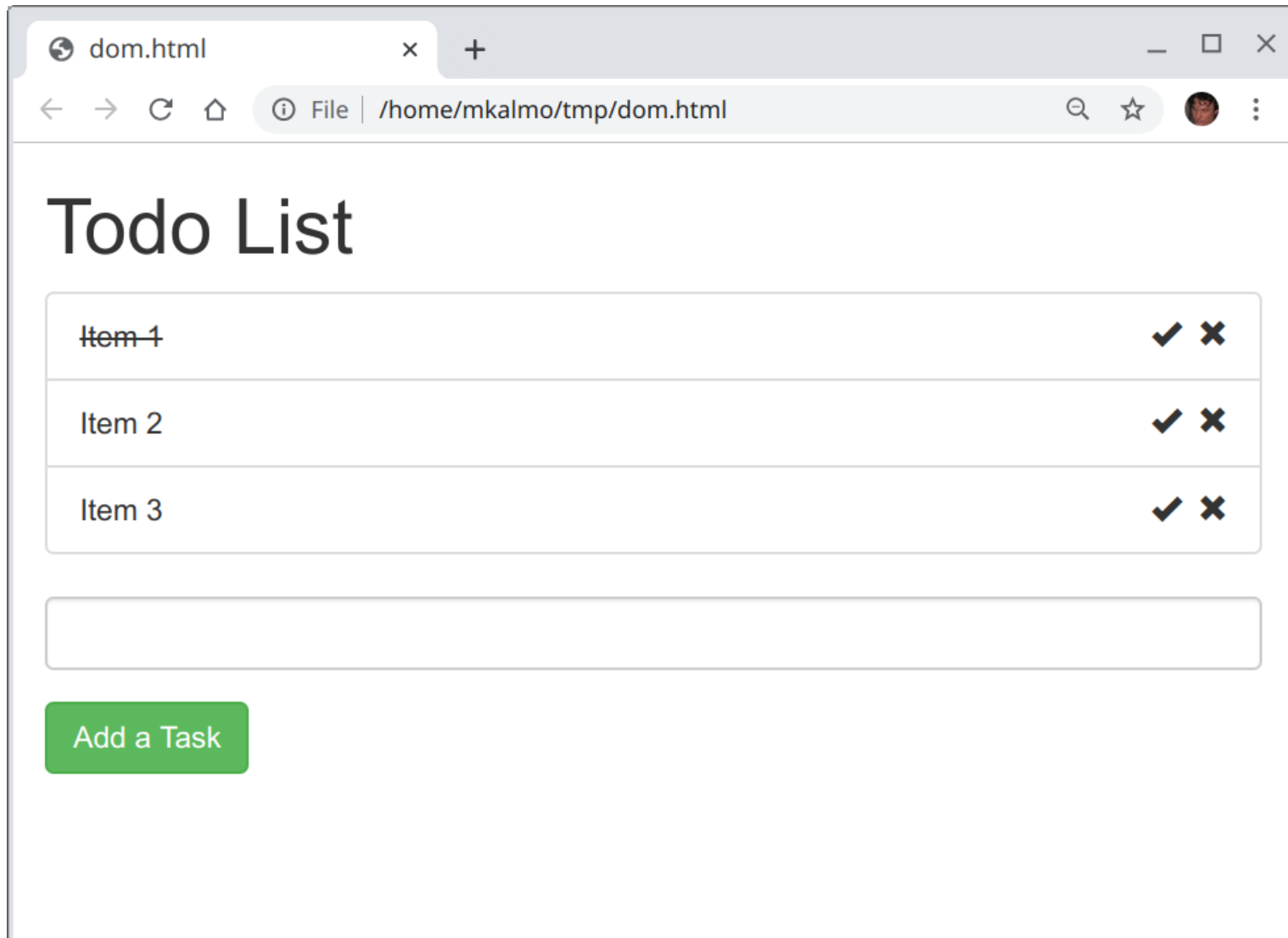
```
  const ul = React.createElement("ul", {}, li);
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

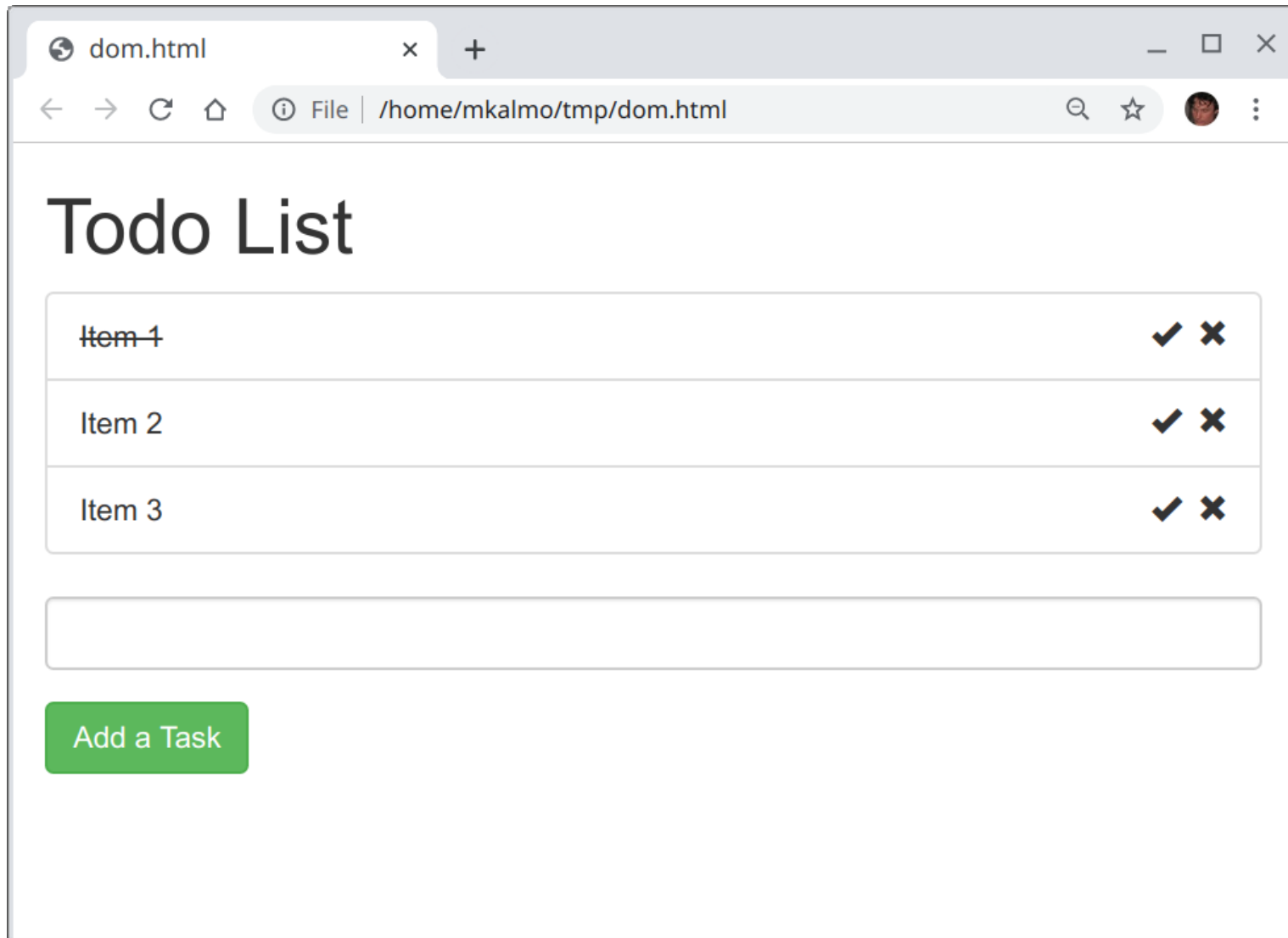
```
</script>
```

```
...
```

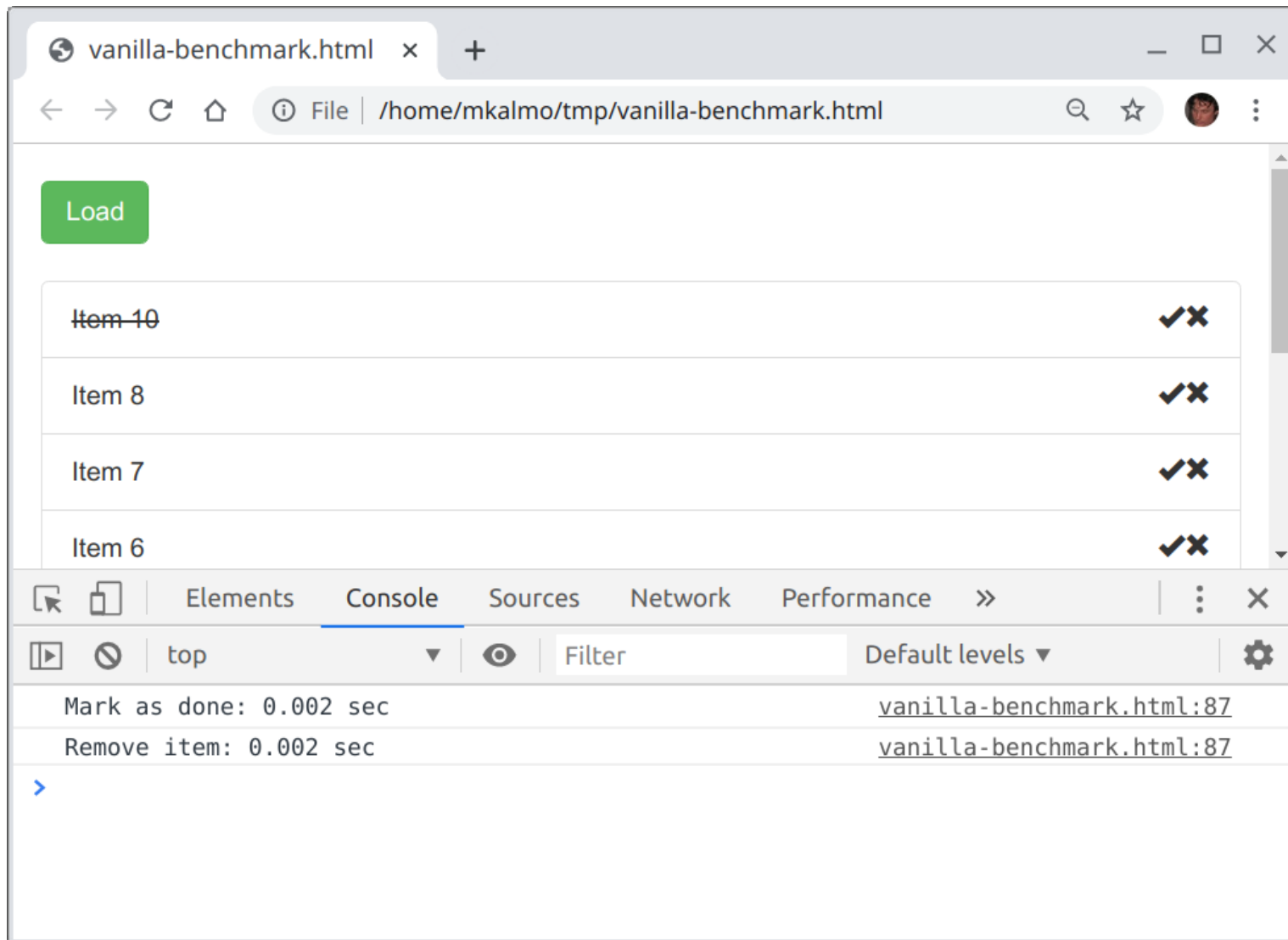
# Muudatused (DOM + tagarakendus)



# Virtual DOM



# Päris DOM (benchmark)



# Päris DOM (benchmark)

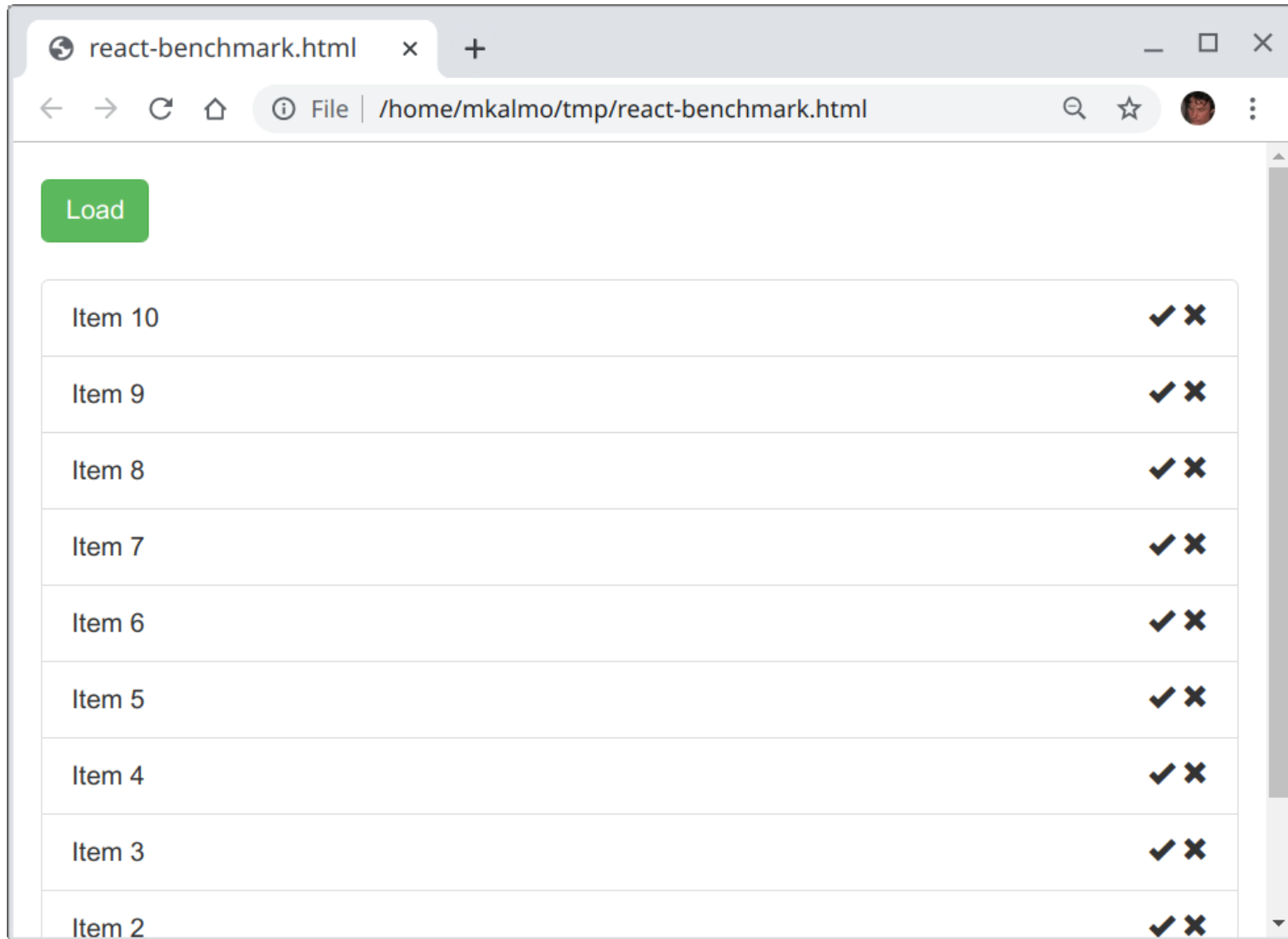
The screenshot shows a web browser window with the address bar displaying `vanilla-benchmark.html`. The page content includes a green **Load** button and a list of items. The console is open, showing the following log entries:

Message	Source
Load data: 0.968 sec	<code>vanilla-benchmark.html:87</code>
Mark as done: 1.077 sec	<code>vanilla-benchmark.html:87</code>
Remove item: 1.128 sec	<code>vanilla-benchmark.html:87</code>

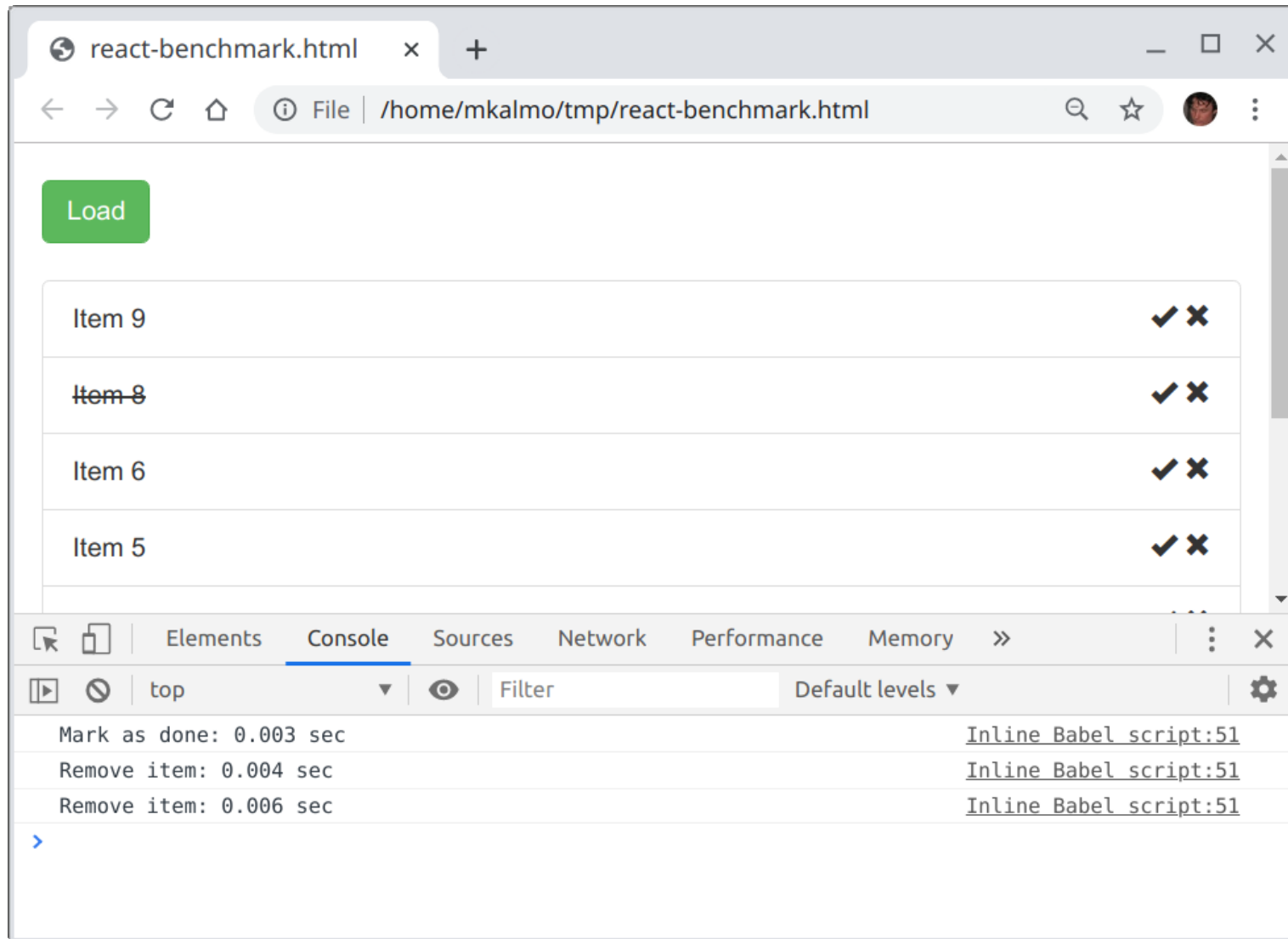
The list of items in the application is as follows:

Item	Status
Item 10000	✓✗
Item 9999	✓✗
Item 9997	✓✗
Item 9996	✓✗

# Virtual DOM (benchmark)



# Virtual DOM (benchmark)





# Virtual DOM (benchmark)

The screenshot shows a web browser window with the address bar displaying `react-benchmark.html` and the file path `/home/mkalmo/tmp/react-benchmark.html`. The application has a green **Load** button and a list of items. The Chrome DevTools console is open, showing the following log entries:

Message	Source
Load data: 1.798 sec	<a href="#">Inline Babel script:51</a>
Load data: 0.355 sec	<a href="#">Inline Babel script:51</a>
Mark as done: 0.334 sec	<a href="#">Inline Babel script:51</a>
Remove item: 0.442 sec	<a href="#">Inline Babel script:51</a>
Remove item: 0.539 sec	<a href="#">Inline Babel script:51</a>

# Reacti mallide keel (JSX)

```
<script src="https://unpkg.com/.../babel.js" ...
```

```
...
```

```
<div id="root"></div>
```

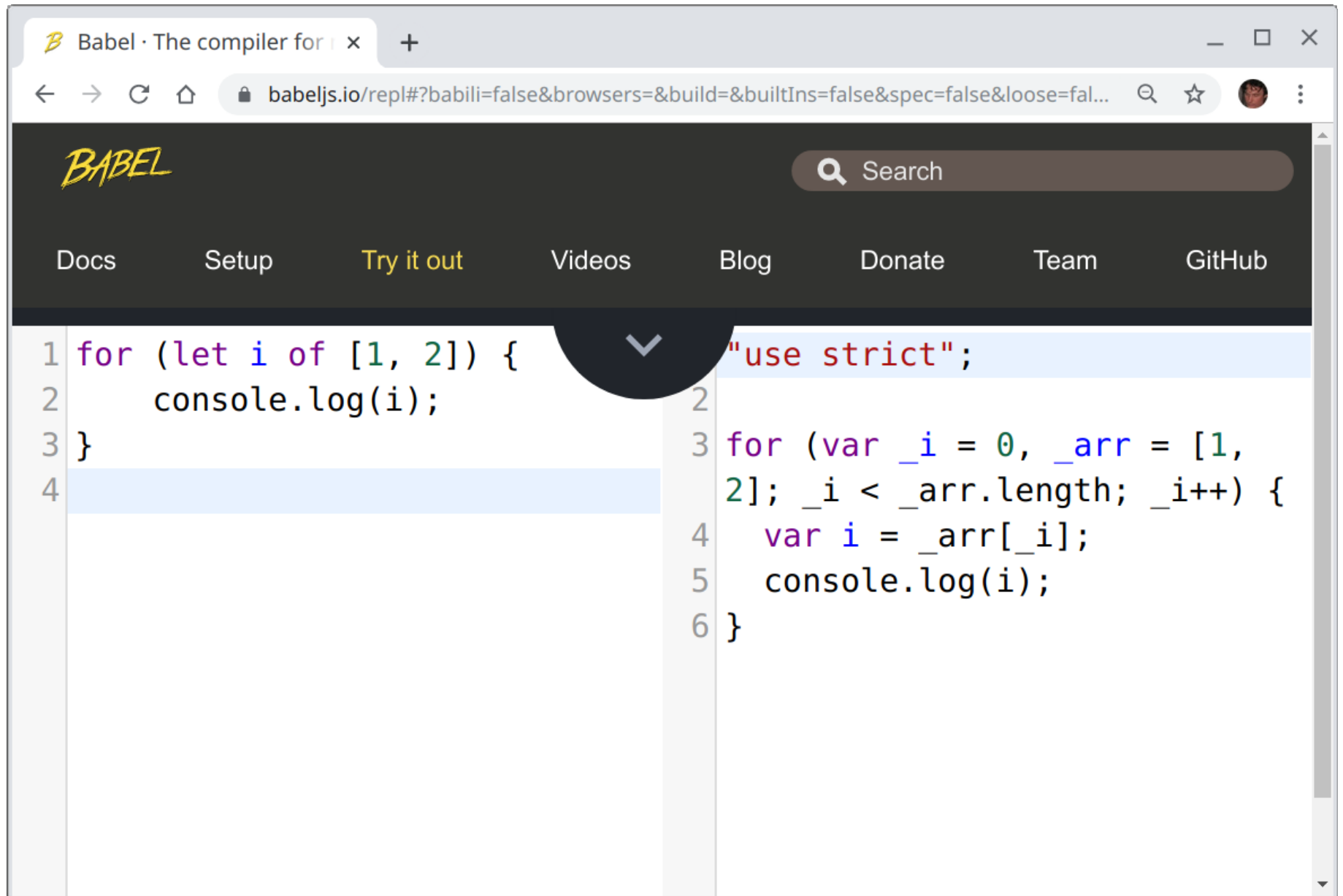
```
<script type="text/babel">
```

```
  const ul = <ul><li>Item 1</li></ul>;
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

```
</script>
```

# Babel



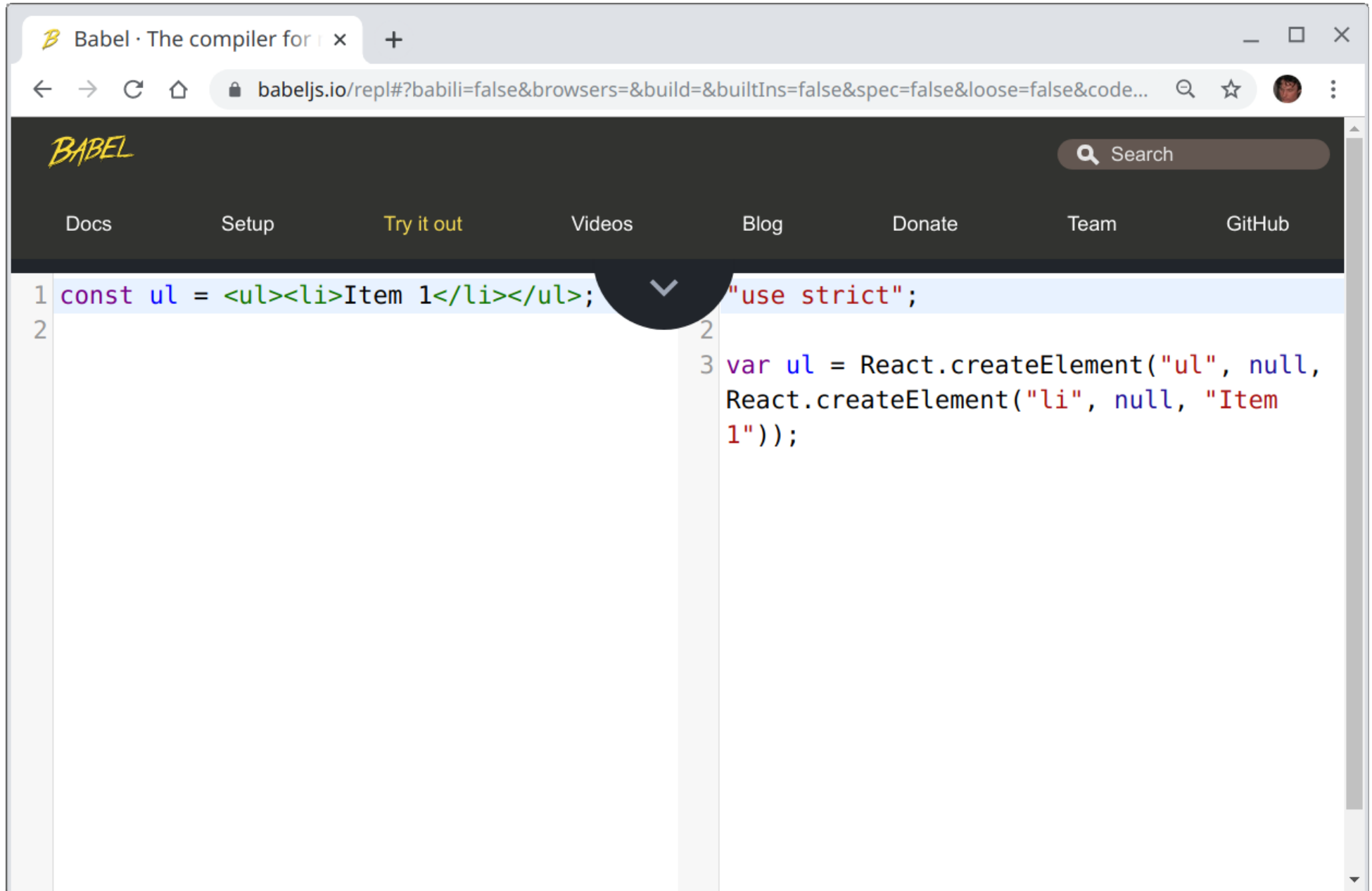
The screenshot shows the Babel REPL interface in a web browser. The browser's address bar displays the URL `babeljs.io/repl#?babili=false&browsers=&build=&builtIns=false&spec=false&loose=fal...`. The Babel logo is in the top left, and a search bar is in the top right. A navigation menu includes links for Docs, Setup, Try it out (highlighted), Videos, Blog, Donate, Team, and GitHub. The main area is a split editor. The left pane contains the input code: 

```
1 for (let i of [1, 2]) {  
2   console.log(i);  
3 }  
4
```

 The right pane shows the transformed code: 

```
1 "use strict";  
2  
3 for (var _i = 0, _arr = [1,  
4   2]; _i < _arr.length; _i++) {  
5   var i = _arr[_i];  
6   console.log(i);  
7 }
```

# Reacti mallide keel (JSX)



The screenshot shows the Babel REPL interface in a web browser. The browser's address bar displays the URL `babeljs.io/repl#?babili=false&browsers=&build=&builtIns=false&spec=false&loose=false&code...`. The Babel logo is in the top left, and a search bar is in the top right. A navigation bar contains links for Docs, Setup, Try it out (highlighted), Videos, Blog, Donate, Team, and GitHub. The main area is split into two panels. The left panel contains the input code: 

```
1 const ul = <ul><li>Item 1</li></ul>;  
2
```

 The right panel contains the output code: 

```
1 "use strict";  
2  
3 var ul = React.createElement("ul", null,  
  React.createElement("li", null, "Item  
  1"));
```

# Reacti mallide keel (JSX)

...

```
<div id="root"></div>
```

```
<script type="text/babel">
```

```
  const ul = <div><br /></div>;
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

```
</script>
```

# Reacti mallide keel (JSX)

...

```
<script type="text/babel">
```

```
  const name = 'Item 1';
```

```
  const ul = <ul><li>{ name }</li></ul>;
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

```
</script>
```

# Reacti mallide keel (JSX)

...

```
<script type="text/babel">
```

```
  const items = [1, 2, 3];
```

```
  const ul = <ul>
    {
      items.map(i => <li>Item { i }</li>)
    }
  </ul>;
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

```
</script>
```

# Reacti mallide keel (JSX)

...

```
<script type="text/babel">
```

```
  const items = [1, 2, 3];
```

```
  const ul = <ul>
    {
      getItems()
    }
    </ul>;
```

```
  ReactDOM.render(ul, document.getElementById('root'));
```

```
  function getItems() { ... }
```

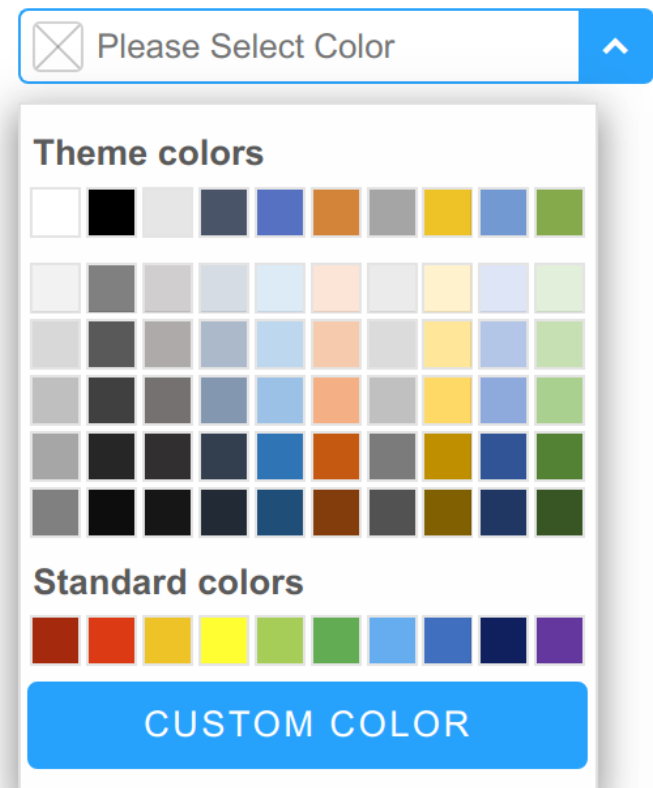
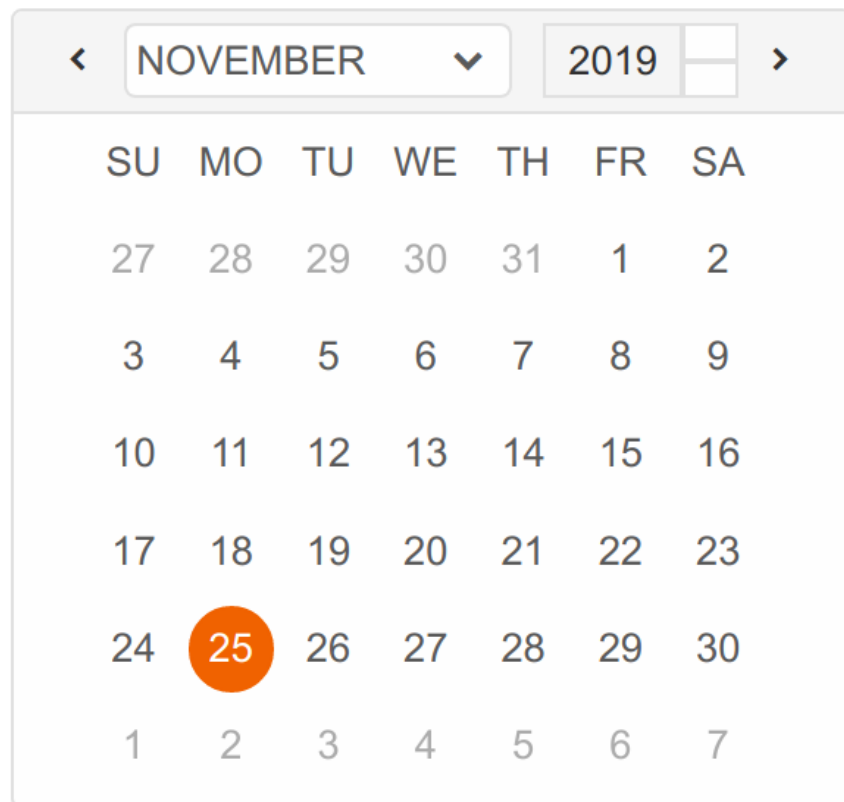
```
</script>
```



# React monoliit (näide)

# Komponendid

- Html + Css + JavaScript



<https://www.htmlelements.com/demos/>

# Komponendid

- [https://developer.mozilla.org/en-US/docs/Web/Web\\_Components](https://developer.mozilla.org/en-US/docs/Web/Web_Components)

# Komponendid

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Web Component Example</title>
  <script src="main.js" defer></script>
  <link href="main-style.css" rel="stylesheet">
</head>
<body>
<h1>Web Component Example</h1>

<my-component data-text="Sample text"/>

</body>
</html>
```

# Komponendid

- Angular, React, Vue, ...

# Komponendid Reactis

```
const content = <div>
    ...
    <Calendar date={ new Date() } />
    ...
</div>;

ReactDOM.render(content, document.getElementById('root'));
```

# Analoogia

- Monoliitne Java rakendus kasutab ArrayList-i
- Nt. kalendri komponendi kasutamine

# Analoogia

- Objektorienteeritult kirjutatud rakendus
- Kogu rakenduse komponentideks jagamine



# Komponendid

## Todo List

Item 1

✓ ✕

Item 2

✓ ✕

Item 3

✓ ✕

Add a Task

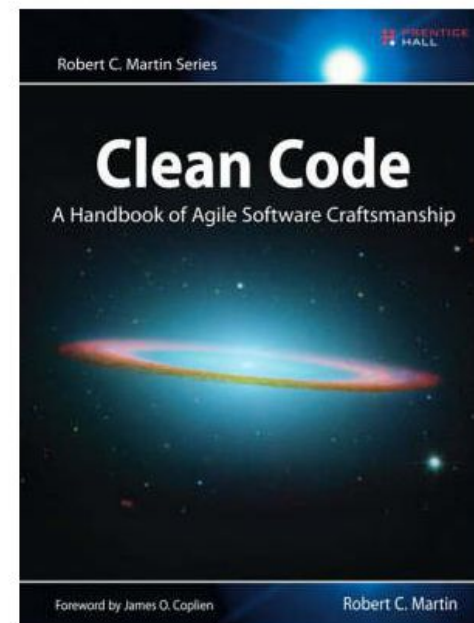
# Komponendid

- Loogika jagamine komponentide vahel
- Seosed ja sõltuvused
- Suhtlus komponentide vahel

Item 1	✓ x
Item 2	✓ x
Item 3	✓ x

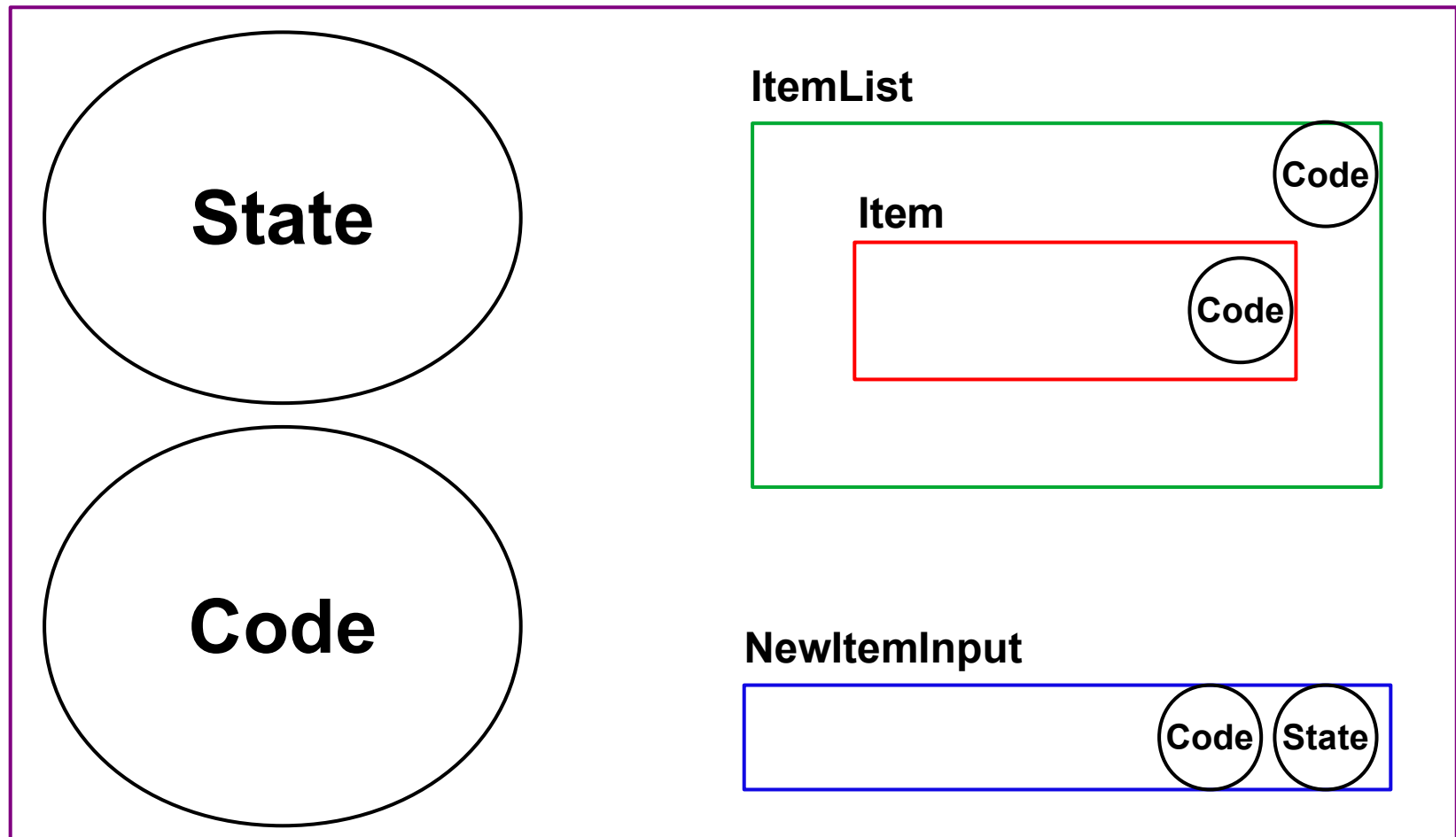
Add a Task

# Analoogia: seosed ja sõltuvused

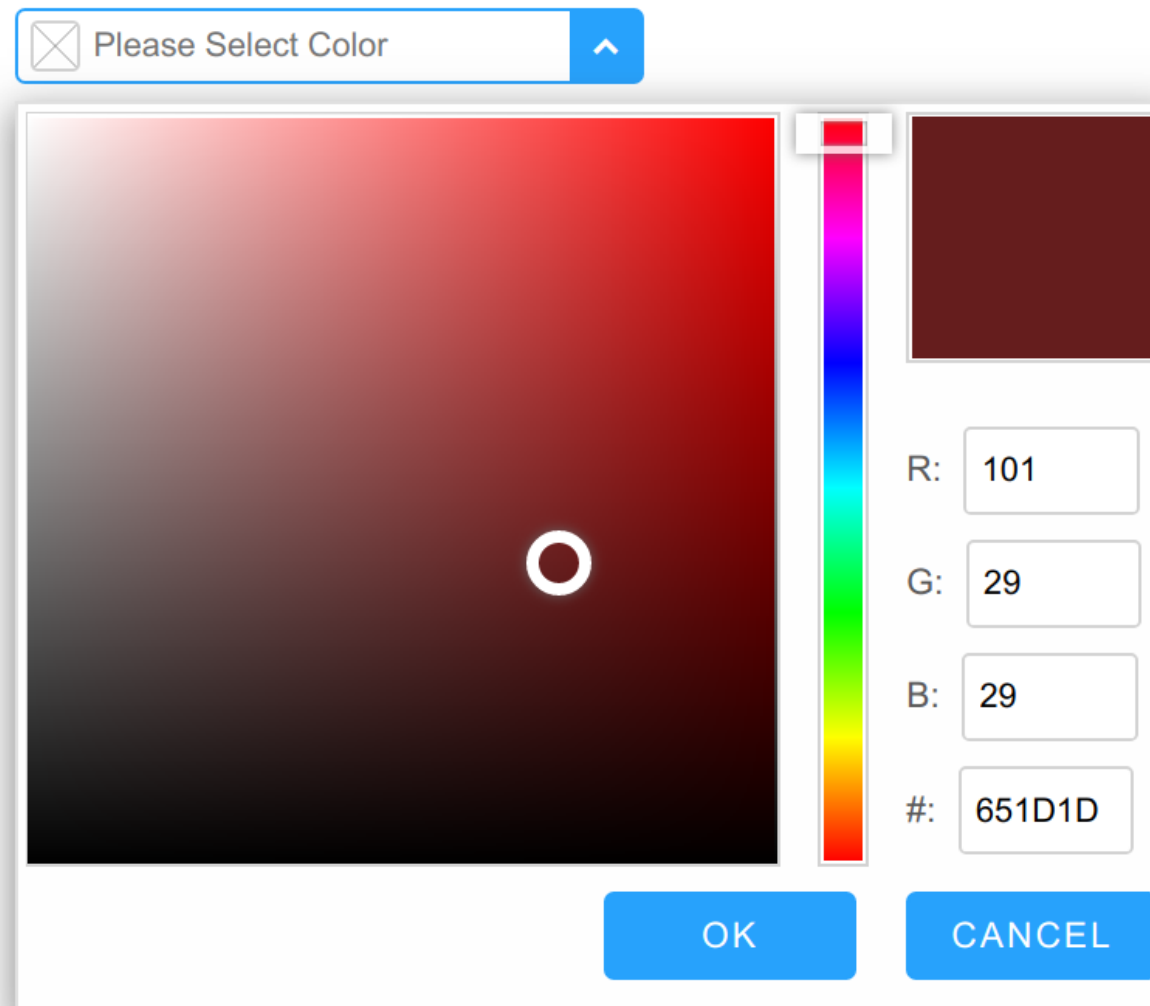


# Komponentide seosed ja sõltuvused

App



# Komponendi olek



# Reacti rakendus komponentidega (näide)

# Tööriistad

- Translaator (nt. JSX)
- Paketeerija
- Hot Swap
- ...

# Paketeeriija

```
import React, {Component} from 'react';  
import Http from './Http';
```

```
import ItemList from './ItemList';  
import ItemInput from './ItemInput';
```

```
class TodoApp extends Component { ...
```



# Tööriistad

- `npx create-react-app <app name>`

# Kaitsmised

# Eksami teemad

- Servlet
- Projekti ehitamine
- Jdbc
- Spring Core
- Spring Mvc
- Jpa
- Spring Security