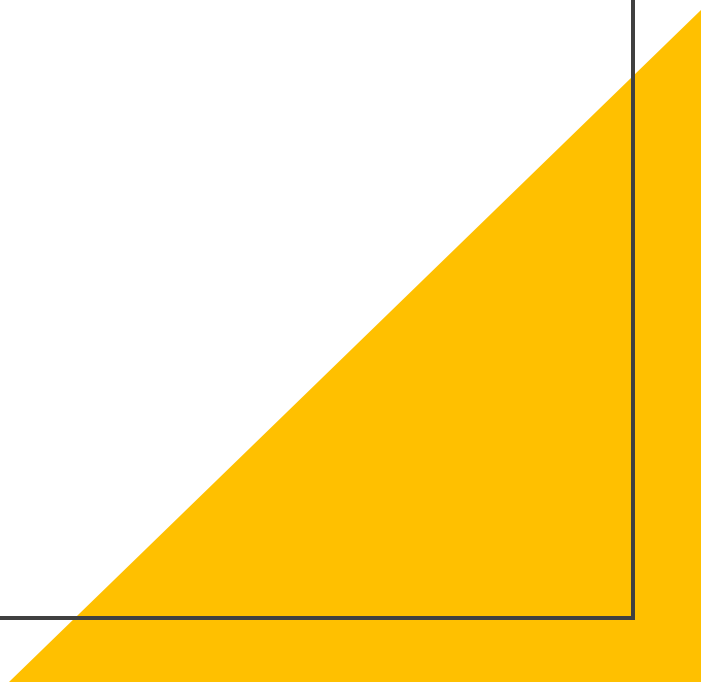



# OORSR

0 predmetu



# O predmetu

- [Obstoječe informacije](#) 
- Vsebina
  - Web Components, React
  - Sodobni pristopi CSS
    - BEM, SCSS, CSS-in-JS
  - Vabljeni predavanja (Angular, Vue.js?)
  - Kolokviji

# O predmetu

- <https://roadmap.sh/>

# Spletne komponente (Web components)

Gregor Jošt



# Spletne komponente

- Predstavljajo samostojen, samozadosten nabor HTML, CSS in JavaScript
- So neodvisne od drugih delov spletne strani, kar omogoča lažje vzdrževanje in ponovno uporabo
- Omogočajo boljšo modularnost in organizacijo kode
- So združljive s standardi spletnega razvoja (HTML5)

# Spletne komponente

- Modularnost: deljenje aplikacij na manjše enote
- Ponovna uporaba: ponovna uporaba kode v različnih projektih
- Neodvisnost od platforme: delujejo na vseh sodobnih brskalnikih ([sort of](#))
- Skrivanje implementacijskih podrobnosti

# Spletne komponente

- (Lahko so) neodvisne od ogrodij
  - <https://angular.io/guide/elements>
  - <https://legacy.reactjs.org/docs/web-components.html>
  - <https://vuejs.org/guide/extras/web-components>
- Primer brez ogrodja: <https://jsbin.com/vesayasuwe/edit?html,output>

# Spletne komponente

---

```
<!-- Define a custom web component -->
<my-button></my-button>

<script>
  // Define the custom web component class
  class MyButton extends HTMLElement {
    constructor() {
      super();

      // Create a shadow DOM for encapsulation
      const shadow = this.attachShadow({ mode: 'open' });

      // Create a button element
      const button = document.createElement('button');
      button.textContent = 'Click Me';

      // Append the button to the shadow DOM
      shadow.appendChild(button);

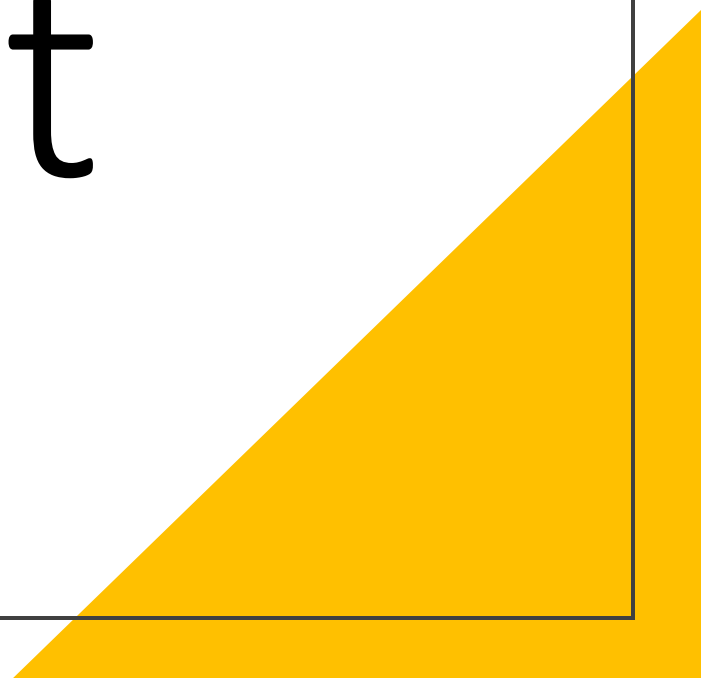
      // Add event listener to the button
      button.addEventListener('click', () => {
        alert('Button clicked!');
      });
    }
  }

  // Define the custom element 'my-button' using custom element API
  customElements.define('my-button', MyButton);
</script>
```



# Uvod v React

Gregor Jošt



# Uvod v React

---

- Poglavlja:
  - Kaj je React?
  - Osnovni pojmi
  - Kako deluje React?
  - Vzpostavitev okolja

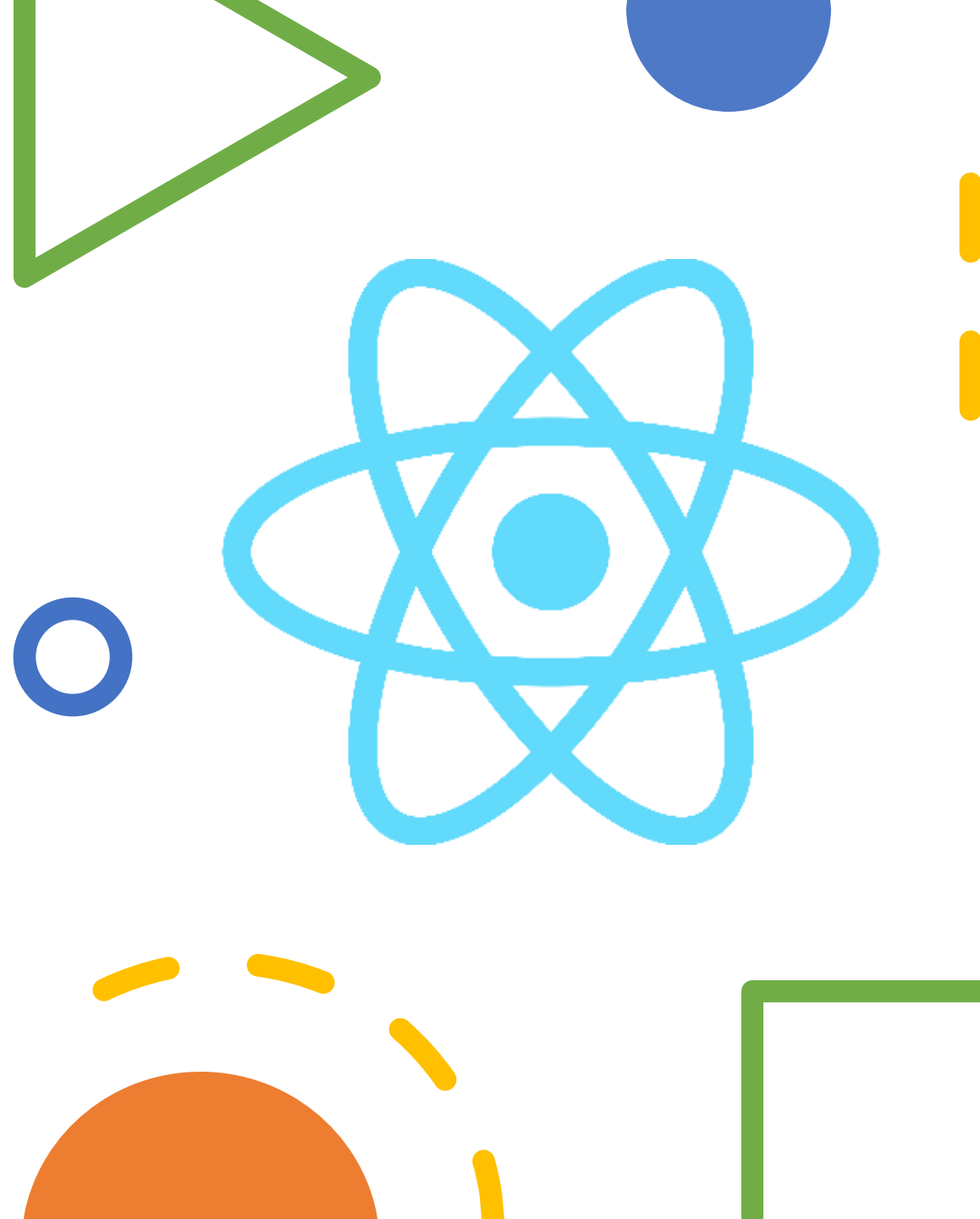


# Uvod v React

Kaj je React?

# Kaj je React?

**Knjižnica** JavaScript za razvoj uporabniških vmesnikov (UV)



# Kaj je React?

- Primeri spletnih strani oz. aplikacij, ki uporabljajo React



Microsoft

amazon



BBC

Walmart

TESLA



NETFLIX



airbnb



Dropbox



Uber



slack



reddit

The New York Times



PayPal



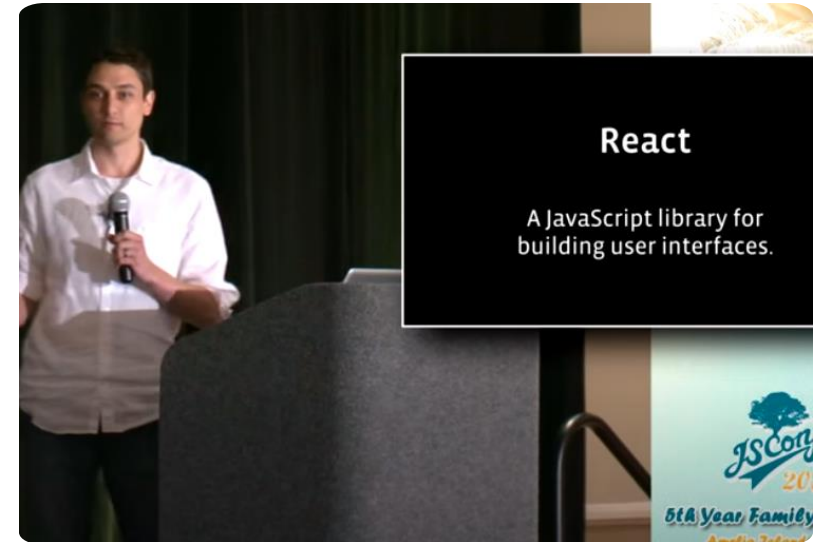
# Kaj je React?

- Razvit s strani Facebook-a (Meta)
  - <https://github.com/jordwalke>
  - Trenutno **več kot 1000** odprtokodnih "contributors"
- Začetki segajo v leto 2011, prva stabilna (odprtokodna) različica 2013
  - **Facebook Ads**
  - <https://github.com/jordwalke/FaxJs>

```
var HelloMessage = React.createClass({
  render: function() {
    return (
      <div>Hello {this.props.name}</div>
    );
  }
});
```

# Kaj je React?

- Ob predstavitvi ogrodja na konferenci **JSConfUS 2013** je bil odziv precej negativen
  - <https://www.youtube.com/watch?v=GW0rj4sNH2w>
  - *"The audience was skeptical. **Most people** thought **React** was a **huge step backward**"*



# Kaj je React?

- Trenutno zelo priljubljen med razvijalci
  - "*React.js completes its fifth year as most wanted*",  
vir: <https://survey.stackoverflow.co/2022/>
  - "*Node.js and React.js are the two most common web technologies used by all respondents.*", vir:  
<https://survey.stackoverflow.co/2023/>





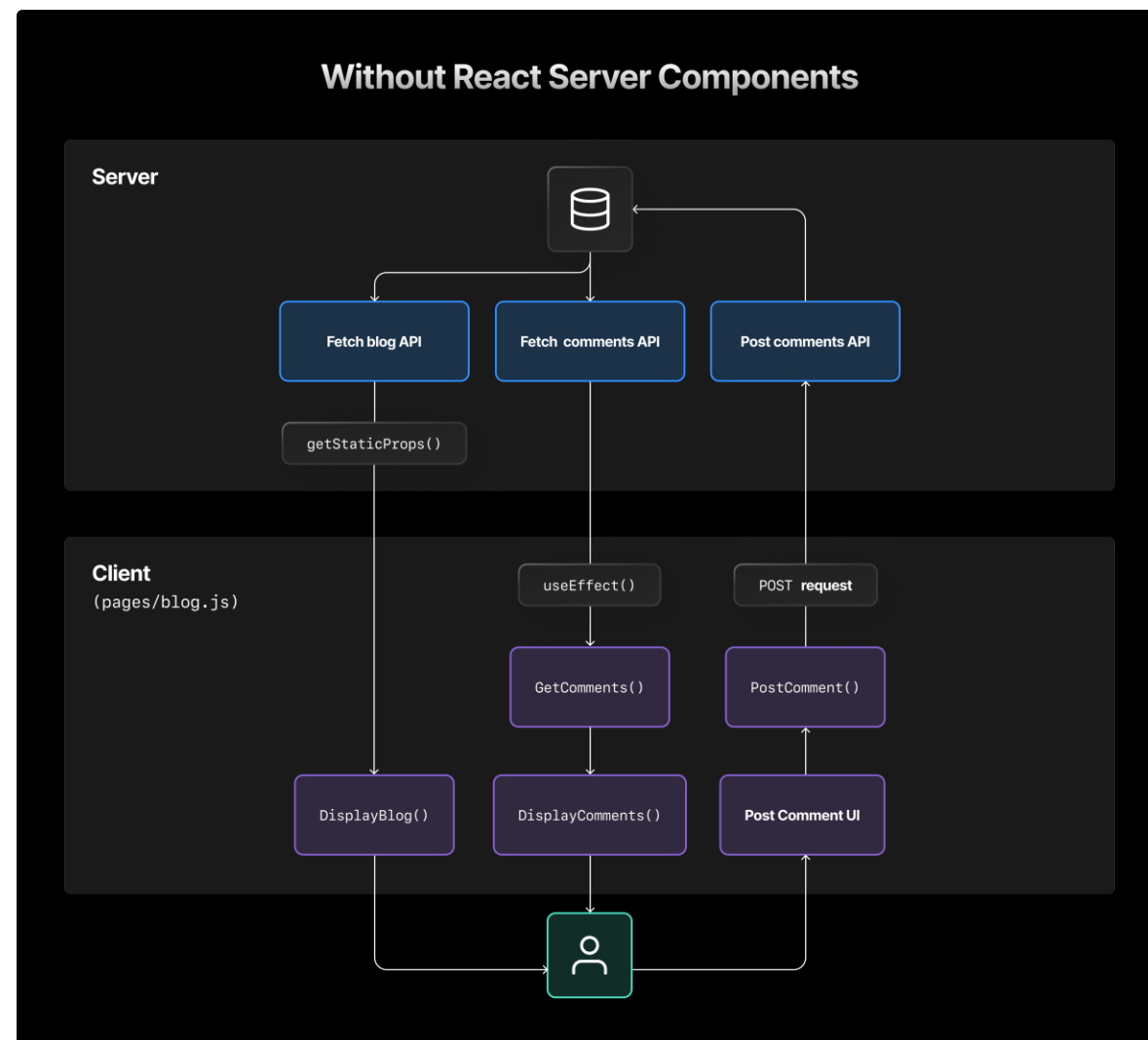
# Kaj je React?

- React 0.3.0 (2013): Prva (javna) različica React-a
- React 0.12.0 (2014): Podpora **ES6 razredom** za implementacijo komponent
- React 15.0.0 (2016): Umaknjena podpora **createClass**
- React 16.0.0 (2017): Dodani fragmenti, **Fiber**
- React 16.3.0 (2018): Dodan **context** API
- React 16.8.0 (2019): The One With **Hooks**
- Aktualna različica: React 18.3.1 (April 26, 2024)
- React 19 "beta"



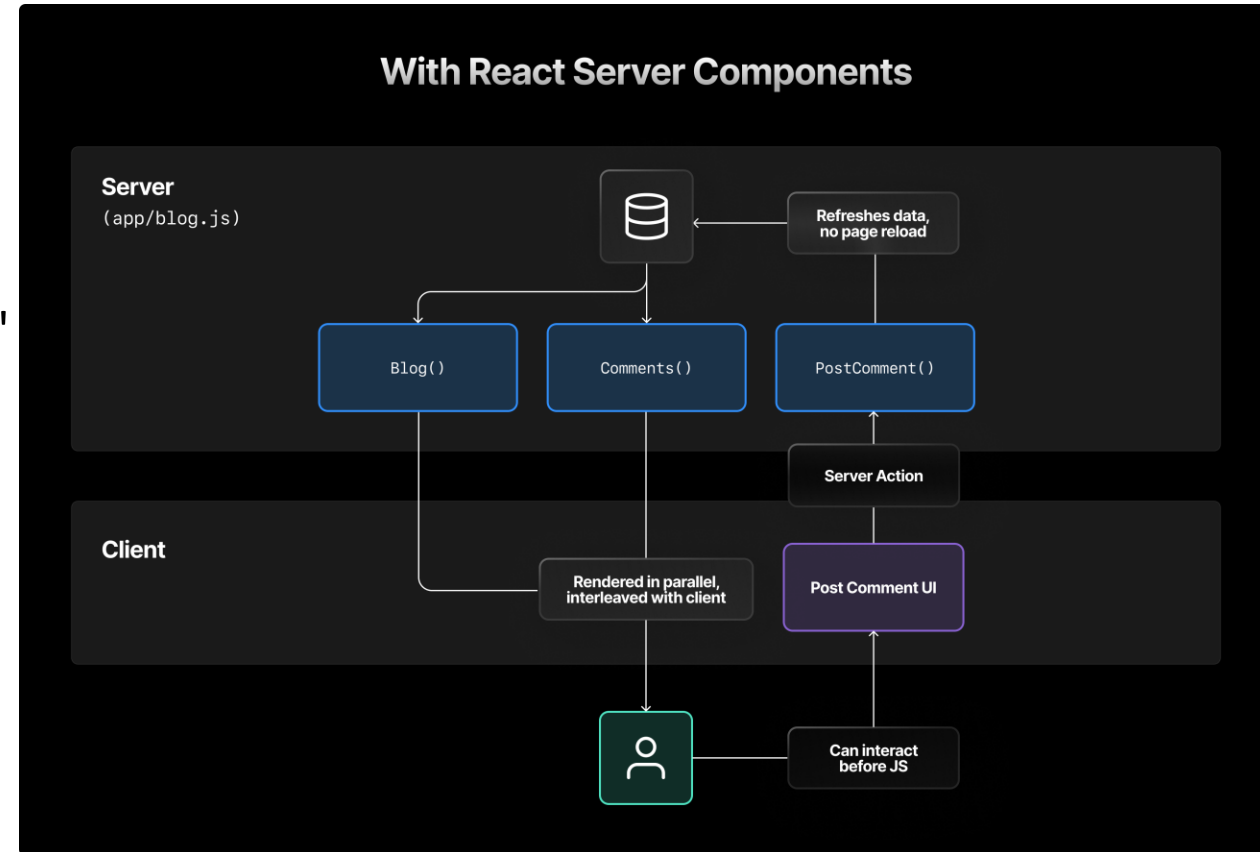
# Kaj je React?

- Prihodnost React:
  - Server Components
    - Directives: "use client" and "use server"
    - Actions, useOptimistic



# Kaj je React?

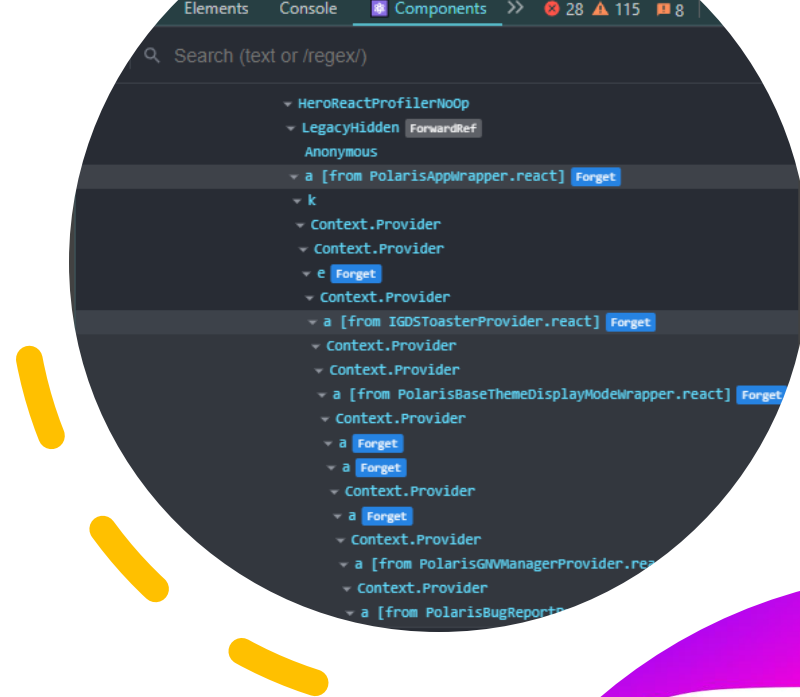
- Prihodnost React:
  - Server Components
    - Directives: "use client" and "use server"
    - Actions, useOptimistic



<https://vercel.com/blog/understanding-react-server-components>

# Kaj je React?

- Prihodnost React:
  - React Compiler is no longer a research project: the compiler now powers instagram.com in production, and we are working to ship the compiler across additional surfaces at Meta and to prepare the first open source release.



# Kaj je React?

```
function Counter() {  
  const [count, setCount] = useState(0);  
  
  function increment() {  
    setCount(count + 1);  
  }  
  
  return (  
    <div>  
      <h1>Counter App</h1>  
      <p>Count: {count}</p>  
      <button onClick={increment}>Increment</button>  
    </div>  
  );  
}
```



# Kaj je React?



*React **components** are JavaScript **functions**.*

*Want to show some content conditionally?  
Use an **if** statement.*

*Displaying a list? Try array **map()**.*

*Learning React is **learning programming**.*

Vir: <https://react.dev/>

# Kaj je React?

- React je **deklarativen**
  - Programska koda opisuje **KAJ** naj aplikacija počne in **NE KAKO**
  - Ne opisujemo korak za korakom kako spremeniti DOM
  - Definiramo **izgled UV**, React pa poskrbi za **izvedbo**
- **Imperativen** pristop zahteva, da podamo seznam ukazov
  - Npr. knjižnica jQuery ponuja skupek funkcij za neposredno manipulacijo DOM-a (ustvari, dodaj, odstrani)

# Kaj je React?

jQuery (imperativen pristop)

```
const $list = $("#list");

items.forEach(item => {
  const $li = $("- ").text(item);
  $list.append($li);
});

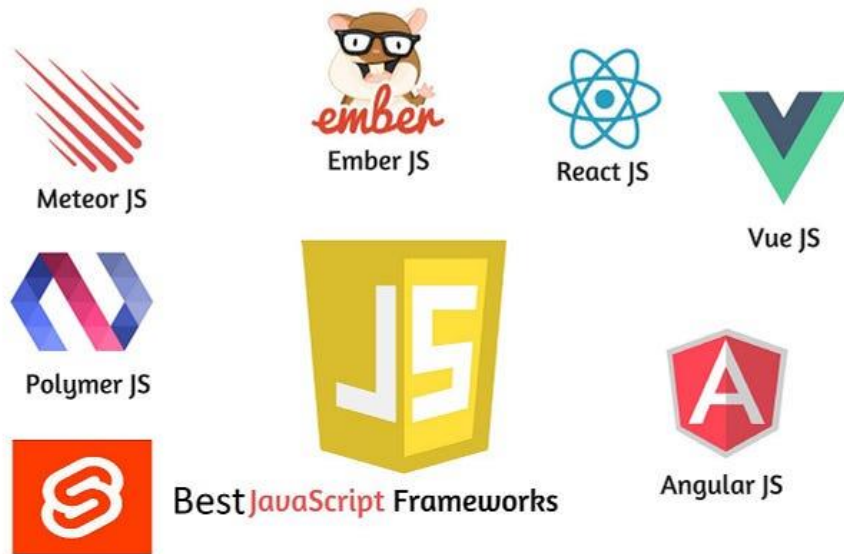
```

React (deklarativen pristop)

```
function ItemList({ items }) {
  return (
    <ul>
      {items.map((item) => (
        <li key={item}>{item}</li>
      ))}
    </ul>
  );
}
```



# Kaj je React?



- Alternative React-u

- Vue.js
- Angular
- Svelte
- Ember

# Kaj je React?

*React is a library.*

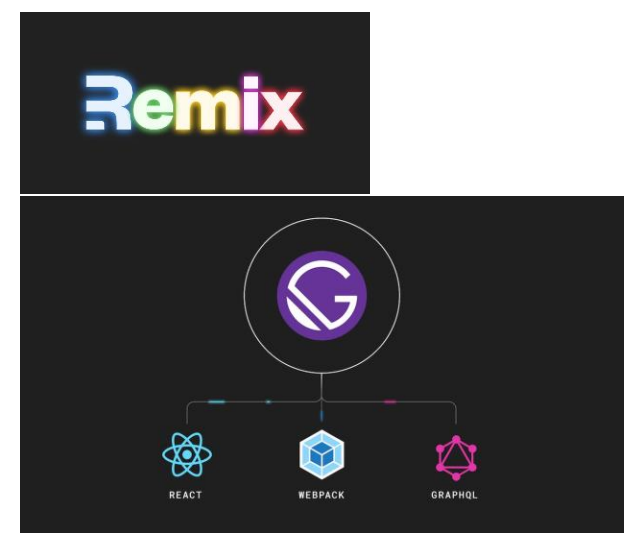
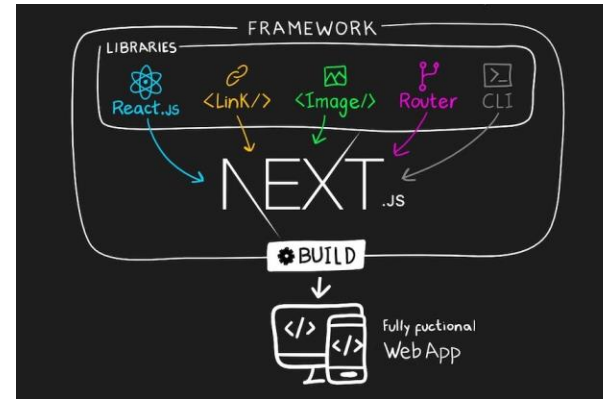
*It lets you put components together, but it **doesn't** prescribe how to do **routing** and **data fetching**.*

*To build an **entire app with React**, we recommend a full-stack React framework like **Next.js** or **Remix**.*

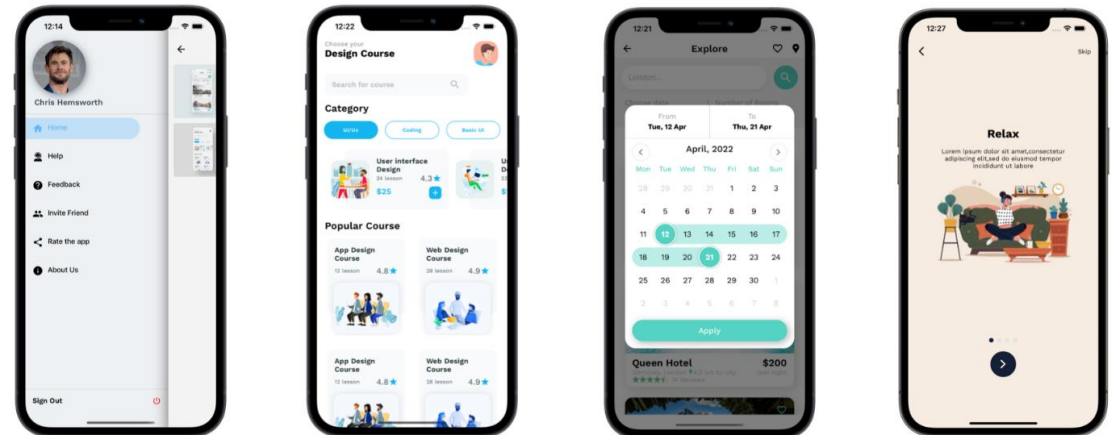


# Kaj je React?

- Priljubljena ogrodja



## React Native UI Templates





# Uvod v React

Osnovni pojmi

# Osnovni pojmi

- JSX
- Komponenta
- Lastnosti komponente (angl. props)
- Stanje (angl. state)
- Kavljji (angl. hooks)
- Virutalni (navidezni) DOM (angl. virtual DOM)
- Izris (angl. rendering)

# Osnovni pojmi – JSX

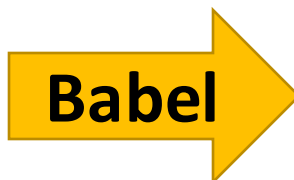
- **JavaScript XML** oz. JSX je sintaktična razširitev jezika JavaScript
- Omogoča pisanje HTML-ju podobno označevalno kodo znotraj jezika JavaScript
- JSX se prevede v programsko kodo JavaScript preden se izvede
  - V ta namen se pogosto uporablja [Babel](#)
  - `createElement(type, props, ...children)`

# Osnovni pojmi – JSX

## JSX

```
const heading =  
<h1>Predavanje OORSR</h1>;
```

```
const list = (  
  <ul id="obveznosti">  
    <li>Vaje</li>  
    <li>Predavanja</li>  
  </ul>  
>);
```



## Prevedno v JavaScript

```
const heading = React.createElement("h1", null,  
  "Predavanje OORSR");
```

```
const list = React.createElement(  
  "ul",  
  {  
    id: "obveznosti",  
  },  
  React.createElement("li", null, "Vaje"),  
  React.createElement("li", null, "Predavanja")  
>);
```

# Osnovni pojmi – JSX

- **React** in **JSX** se lahko uporabljata **neodvisno** en od drugega
- JSX **ni obvezen** za razvoj aplikacij React
  - Uporaba je ZELO pogosta
- JSX se uporablja tudi v drugih knjižnicah, npr. Preact, Inferno, in **Vue.js**
- <https://transform.tools/html-to-jsx>

```
import { defineComponent, ref } from '@vue/composition-api'

export default defineComponent({
  setup() {
    const count = ref(0)
    const increment = () => count.value++

    return {
      count,
      increment
    }
  },

  render() {
    return (
      <div>
        <button onClick={this.increment}>Increment</button>
        <span>Count: {this.count}</span>
      </div>
    )
  }
})
```



# Osnovni pojmi – JSX, pravila uporabe

- Vedno vrnemo en, krovni element

```
const heading = (  
  <div>  
    <h1>Predavanje OORSR</h1>  
  </div>  
);
```



```
const heading =  
React.createElement(  
  "div",  
  null,  
  React.createElement("h1", null,  
    "Predavanje OORSR")  
);
```

```
const heading = (  
  <div>  
    <h1>Predavanje OORSR</h1>  
  </div>  
  <span>Vsebina</span>  
);
```



```
/repl.jsx: Adjacent JSX elements must be  
wrapped in an enclosing tag. Did you want a  
JSX fragment <>...</>? (5:2) 3 |  
<h1>Predavanje OORSR</h1> 4 | </div> > 5 |  
<span>Vsebina</span> | ^ 6 | );
```

# Osnovni pojmi – JSX, pravila uporabe

- Vsi elementi se morajo zaključiti

```
const imageWrong = ;
```

```
const imageRight = ;
```

# Osnovni pojmi – JSX, pravila uporabe

- Zaviti oklepaji `{}` za prikaz dinamičnih vrednosti
- Odpre se možnost vključitve sintakse JavaScript znotraj JSX
- Vključijo se znotraj elementov ali kot lastnost/atribut

```
const dinamicnoIme = "Predmet";  
const naslov = <h1>{dinamicnoIme}</h1>;  
const slika = ;
```



```
const dinamicnoIme = "Predmet";  
const naslov = React.createElement("h1", null, dinamicnoIme);  
const slika = React.createElement("img", {  
  src: "...",  
  alt: dinamicnoIme,  
});
```

# Osnovni pojmi – JSX, pravila uporabe

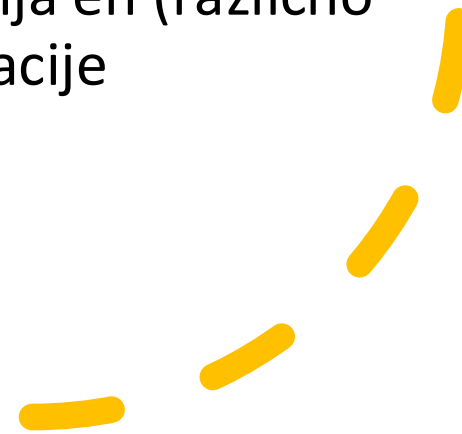
- Pri pisanju lastnosti se uporablja večzačetnica (**camelCase**)
- Atributi JSX postanejo lastnosti objekta v jeziku JavaScript
  - Imena lastnosti ne smejo vsebovati vezajev ali rezerviranih besed
- Veliko atributov HTML v Reactu je pisanih kot *camelCase*
- Izjeme? *data-*

```
const heading = (  
  <h1 className="heading"  
style={{ backgroundColor: "red"  
}}>  
    Predavanje  
  </h1>  
);
```

```
const heading = React.createElement(  
  "h1",  
  {  
    className: "heading",  
    style: {  
      backgroundColor: "red",  
    },  
  },  
  "Predavanje"  
);
```

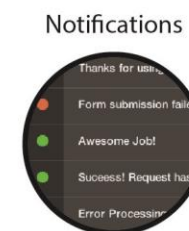
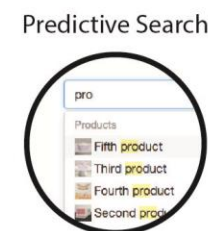
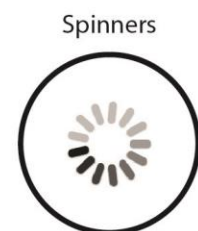
# Osnovni pojmi – komponente

- Komponenta predstavlja osnovo sodobnega razvoja pročelja (**frontend**) sistema
  - Koncept **ni specifičen** za knjižnico React
- UV deli na več neodvisnih enot, ki omogočajo ponovno uporabo
- Skupaj tvorijo celoten UV
  - Posamezna komponenta predstavlja en (različno velik) delček celotne spletne aplikacije



# Osnovni pojmi – komponente

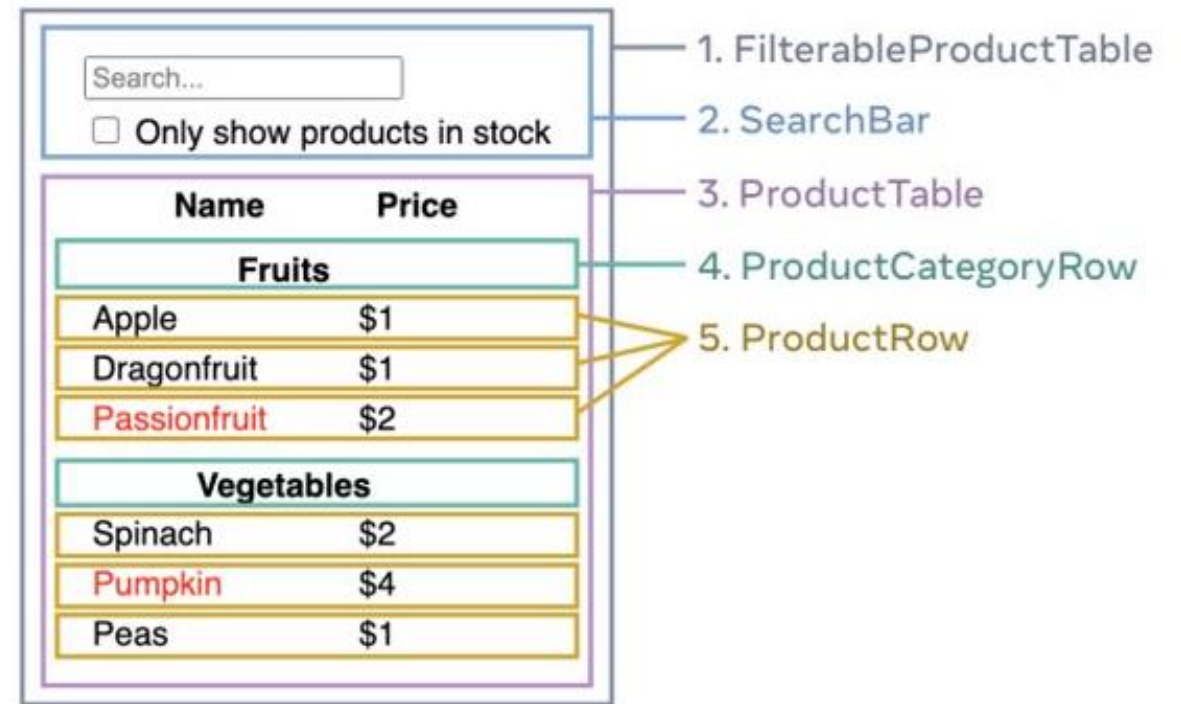
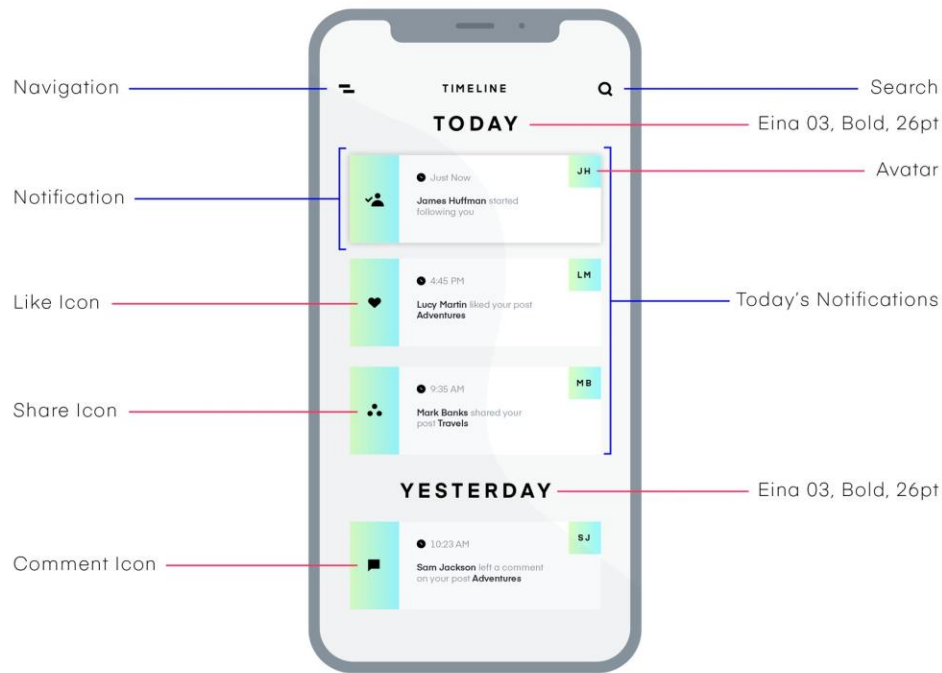
- Komponenta je lahko preprost gumb, slika ali obrazec za prijavo v sistem, lahko pa predstavlja celotno spletno stran



# Osnovni pojmi – komponente

- V knjižnici React velja:
  - Poimenovanje vsake izmed komponent naj bo **z veliko začetnico**
  - Komponente naj bodo kar se le da **majhne**
  - Komponente naj bodo kar se le da **preproste** (in, če je le možno, brez stanja)
  - Uporaba sintakse **JSX** za izris
  - Preferirana uporaba **funkcijskih komponent**
  - Komponenta **lahko izriše** drugo komponento, **ne sme** pa je **definirati**

# Osnovni pojmi – komponente





# Osnovni pojmi – lastnosti komponente

- Lastnosti (angl. ***properties***, pogosto skrajšano na ***props***) so primarni način medsebojnega komuniciranja komponent
- Vsaka glavna komponenta lahko pošlje podrejenim komponentam podatke/informacije preko props
- *Props* lahko delujejo kot atributi HTML, vendar niso
  - Lahko so kakršnekoli vrednosti oz. tipa
- *Props* so edini parameter, ki ga prejmejo funkcijske komponente

# Osnovni pojmi – lastnosti komponente

```
type VizitkaProps = {  
  avtor: string;  
  naslov: string;  
};  
  
const Vizitka = (props: VizitkaProps) => {  
  return (  
    <div className="vizitka">  
      <div className="vizitka__glava">  
        <h1>{props.naslov}</h1>  
      </div>  
      <div className="vizitka__noga">Avtor: {props.avtor}</div>  
    </div>  
  );  
};
```

# Osnovni pojmi – kavlji

- Kavlji (angl. **hooks**) so „novost“ knjižnice React
- Omogočajo deljenje in ponovno uporabo logike, ki hrani stanje
- Zaradi kavljev so funkcijske komponente nadomestile razredne komponente
  - Uporaba kavljev ni obvezna, lahko delujejo vzporedno z obstoječo kodo
- *Hooks* so posebne funkcije, ki omogočajo uporabo specifičnih funkcionalnosti React
- Najpogostejše uporabljen *hook* je **useState**

# Osnovni pojmi – stanje komponente

- Stanje v komponentah si lahko predstavljamo kot podatke, ki si jih mora komponenta zapomniti
  - vpis podatka v vnosno polje,
  - izdelek v košarici,
  - trenutno prikazano sliko v galeriji slik ipd.
- Stanje je torej **spomin** komponente
- Stanje komponente se običajno spreminja z uporabo **useState** *hook-a*
- Sprememba stanja proži ponovni izris komponente!

# Osnovni pojmi – stanje komponente (kviz)

```
import { useState } from "react";

const Stevec = (): JSX.Element => {
  const [steveno, setSteveno] = useState<number>(0);

  const pritiskNaGumb = () => {
    setSteveno(steveno + 1);
    console.log(steveno);
  };

  return (
    <div>
      {steveno}
      <button onClick={pritiskNaGumb}>Povečaj</button>
    </div>
  );
};
```



# Osnovni pojmi – virtualni DOM

- **DOM** (Document Object Model) je programski vmesnik za spletne strani
- DOM predstavi spletno stran kot vozlišča in objekte
- Kot objektno usmerjena predstavitev spletne strani jo lahko spreminjamo s skriptnim jezikom
- DOM v osnovi definira:
  - HTML elemente kot objekte
  - lastnosti HTML elementov
  - metode za dostop do HTML elementov
  - dogodke za HTML elemente

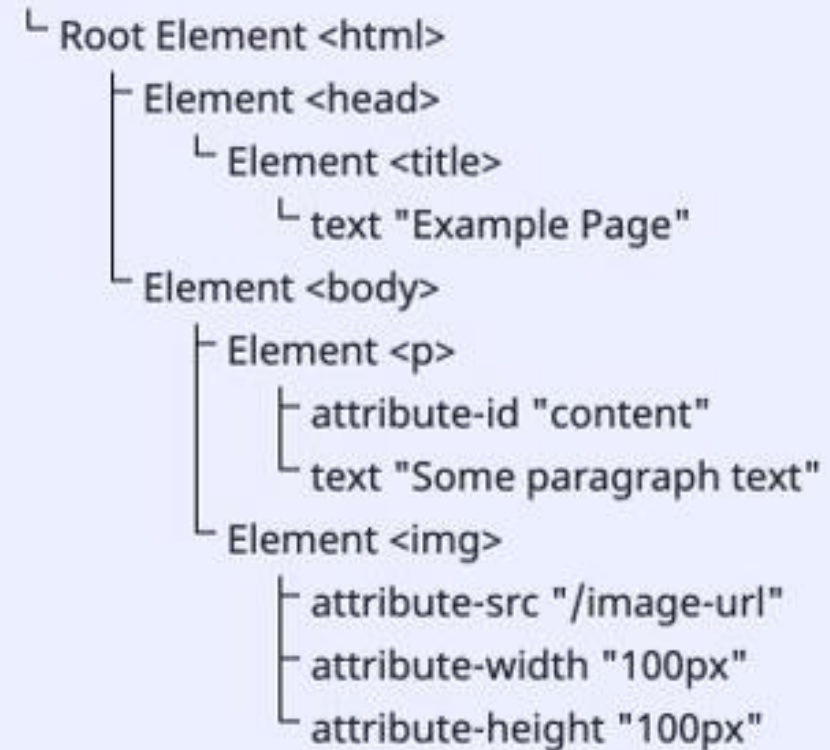
# Osnovni pojmi – virtualni DOM

## HTML

```
<html lang="en">
  <head>
    <title>Example Page</title>
  </head>
  <body>
    <p id="content">Some paragraph text</p>
    
  </body>
</html>
```

## DOM

### Document



# Osnovni pojmi – virtualni DOM

- Slabosti manipulacije DOM-a
  - Pogoste posodobitve DOM-a so lahko potratne (počasnejše delovanje spletnih aplikacij)
  - Vsaka sprememba v DOM-u lahko proži "reflow" in "repaint"
    - Reflow: brskalnik ponovno izračuna postavitev elementov ob dodajanju/odstranjevanju
    - Repaint: postopek posodabljanja vizualnega izgleda elementov

```
// Proži reflow, ko preberemo lastnost
const container = document.getElementById('container');
const containerWidth = container.offsetWidth;

// Proži repaint, ko spremenimo barvo
const box = document.getElementById('box');
box.style.backgroundColor = 'blue';
```



# Osnovni pojmi – virtualni DOM

- Slabosti manipulacije DOM-a
  - Privzeto nimamo na voljo algoritma, ki bi skrbel za optimizacijo posodobitve DOM-a
  - Nimamo (standardnega) načina upravljanja stanja aplikacije
  - Neposredne posodobitve DOM-a lahko pomenijo več "hroščev" v kodi

# Osnovni pojmi – virtualni DOM

- **Virtualni (navidezni) DOM**  
je abstraktna predstavitev DOM-a
- Gre za objekt JavaScript,  
ki hrani kopijo DOM-a v pomnilniku
  - Dejanski DOM je počasen (posodobitve)
  - JavaScript je hitrejši
- Je bolj vzorec kot dejanska tehnologija
- Vse komponente v Reactu komunicirajo zgolj z virtualnim DOM-om

# Osnovni pojmi – virtualni DOM

- Vsaka sprememba stanja aplikacije ustvari posodobljeno verzijo virtualnega DOM-a
- Osnovna ideja uporabe virtualnega DOM-a je, da se „navidezna“ predstavitev UV sinhronizira z dejanskim DOM-om
- Z uporabo virtualnega DOM-a zmanjšamo število posodobitev dejanskega DOM-a

# Osnovni pojmi – virtualni DOM

```
function MyComponent(props) {  
  return (  
    <div>  
      <h1>{props.title}</h1>  
      <p>{props.description}</p>  
    </div>  
  );  
}
```

```
{  
  type: "div",  
  props: {  
    children: [  
      {  
        type: "h1",  
        props: {  
          children: "My Component Title",  
        },  
      },  
      {  
        type: "p",  
        props: {  
          children: "My Component Description",  
        },  
      },  
    ],  
  },  
}
```

# Osnovni pojmi – virtualni DOM



**ДЭН**

@dan\_abramov



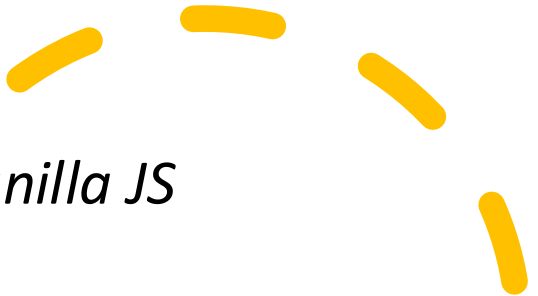
I wish we could retire the term “virtual DOM”. It made sense in 2013 because otherwise people assumed React creates DOM nodes on every render. But people rarely assume this today.

“Virtual DOM” sounds like a workaround for some DOM issue. But that’s not what React is.

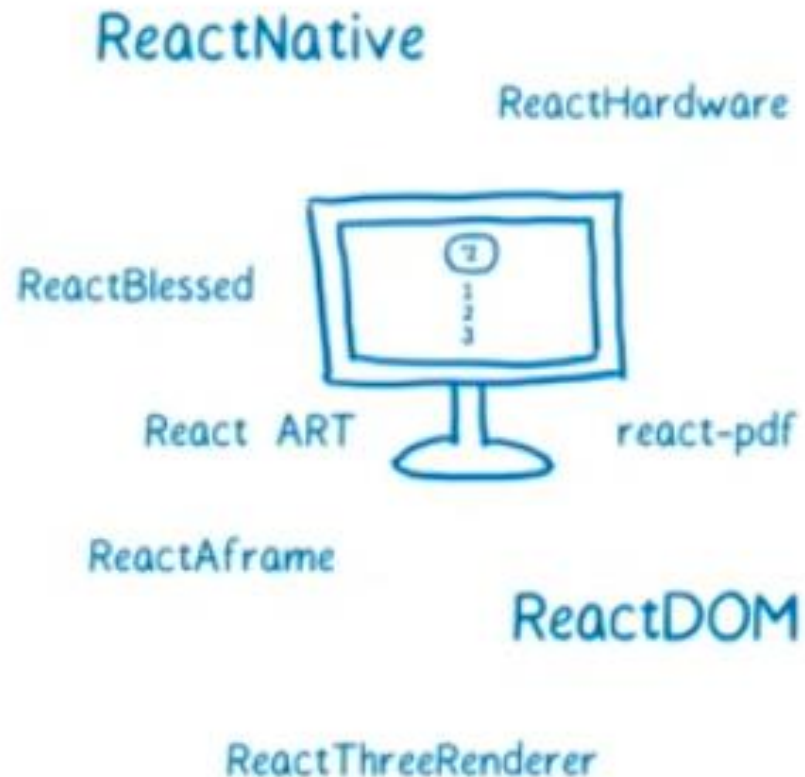
2:52 PM · Nov 24, 2018



# Osnovni pojmi – virtualni DOM

- 
- *We use React rather than Vanilla JS because it is*
  - ***Declarative,***
  - ***Offers Reusable Components and***
  - ***helps to easily build complex UI***
  - *while abstracting away difficult parts.*

# renderer



## Osnovni pojmi – izris

- Izris oz. **rendering** je (*ongoing*) proces pretvorbe komponent React v elemente HTML in njihov prikaz v brskalniku
- React upošteva kombinacijo stanja in *props*
- Za izris skrbi knjižnica **ReactDOM** (in ne React!)
  - React ne upravlja z brskalnikom oz. DOM-om
  - **ReactDOM** je vezni člen med **React** in dejanskim **DOM**

# Osnovni pojmi – izris

- Poznamo začetni izris (**initial render**) in ponovni izris (**re-render**)
  - Začetni zris se izvede, ko se aplikacija prvič zažene

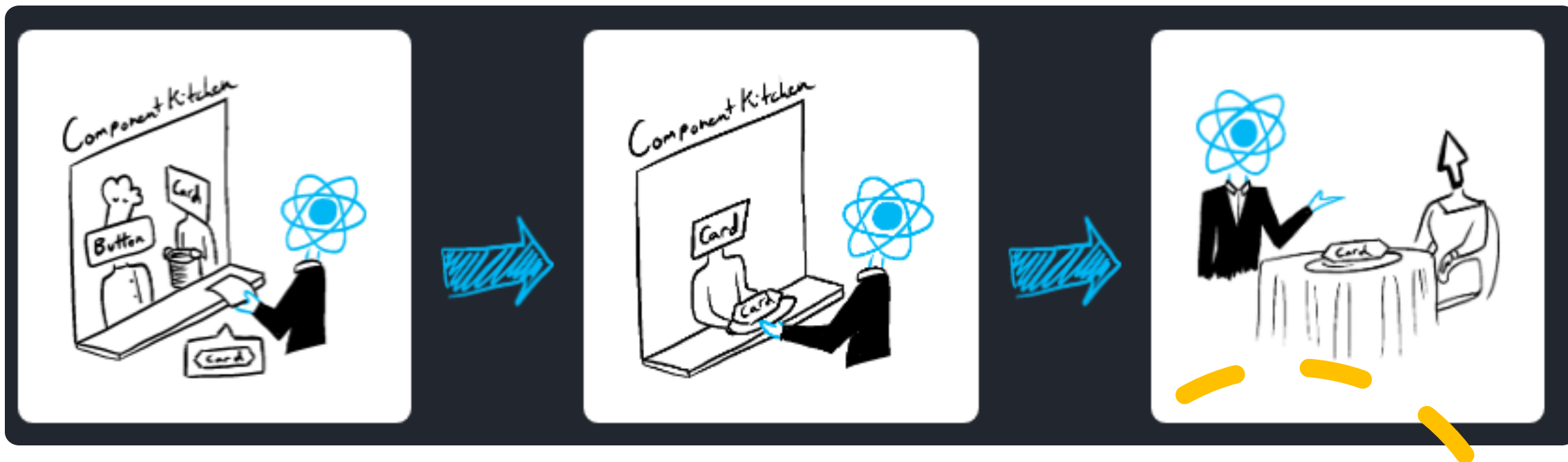
```
import Image from './Image.js';  
import { createRoot } from 'react-dom/client';  
  
const root = createRoot(document.getElementById('root'))  
root.render(<Image />);
```

- Ponovni zris se izvede, ko se spremeni stanje komponente (oz. stanje nadrejene komponente)



# Osnovni pojmi – izris

- Izris se izvaja v treh fazah:
  - Proženje izrisa
    - Začetni zris
    - Sprememba stanja
  - Izris komponente
    - React kliče našo komponento in ugotavlja, kaj je potrebno izrisati
    - Proces je rekurziven, dokler ne pridemo do konca drevesa komponent
  - Faza potrjevanja – dejanska sprememba v DOM-u
    - Spremenijo se zgolj vozlišča DOM, kjer je React zaznal spremembo od prejšnjega stanja
    - DOM ostane nespremenjen, če ni razlik od prejšnjega stanja



Osnovni pojmi – izris

# Osnovni pojmi – izris

- Če se **komponenta ponovno izriše**, se bodo ponovno izrisale tudi vse **komponente znotraj nje!**
- Pri tem React **ne upošteva**, ali so se *props* spremenili ali ne
- Če torej kličemo *setState* na najbolj krovni komponenti, bo prožilo ponovni izris **čisto vseh komponent** v aplikaciji.
  - Dejanski DOM se ne posodobi, kjer ni bilo dejanskih sprememb
  - Vseeno React izvaja kalukacije za ugotavljanje sprememb
  - Dobra praksa: naj se sprememba stanja komponente izvaja na najnižjem možnem nivoju.

# Osnovni pojmi – izris

```
const root = ReactDOM.createRoot(document.getElementById("root"));

function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}</h2>
    </div>
  );
  root.render(element);
}

setInterval(tick, 1000);
```

**Hello, world!**

**It is 12:26:46 PM.**



Primerjava z JavaScript kodo?



# Uvod v React

Kako deluje React?

# Kako deluje React?

- V osnovi lahko opredelimo kako deluje React v naslednjih korakih
  1. Začetni izris
  2. V komponentah se zgodi sprememba
  3. Primerjava razlik prejšnjega in trenutnega stanja komponente
  4. Usklajevanje virtualnega DOM-a z dejanskim glede na spremembe



# Kako deluje React?

## 1. V fazi začetnega izrisa React:

- Prejme seznam komponent in njihovih lastnosti
- Ustvari kopijo pravega DOM-a (virtualni DOM)
- JSX se pretvori v funkcije JavaScript (Babel)
- Izriše komponente v obliki elementov HTML v dejanski DOM

# Kako deluje React?

## 2. Ko pride do sprememb v stanju komponente:

- React ustvari novo (posodobljeno) verzijo virtualnega DOM-a
- Kliče naše komponente in ugotavlja, kaj je potrebno prikazati na zaslonu
- Specifično: kličejo se funkcije, katerih sprememba stanja ja prožila ponovni izris



# Kako deluje React?

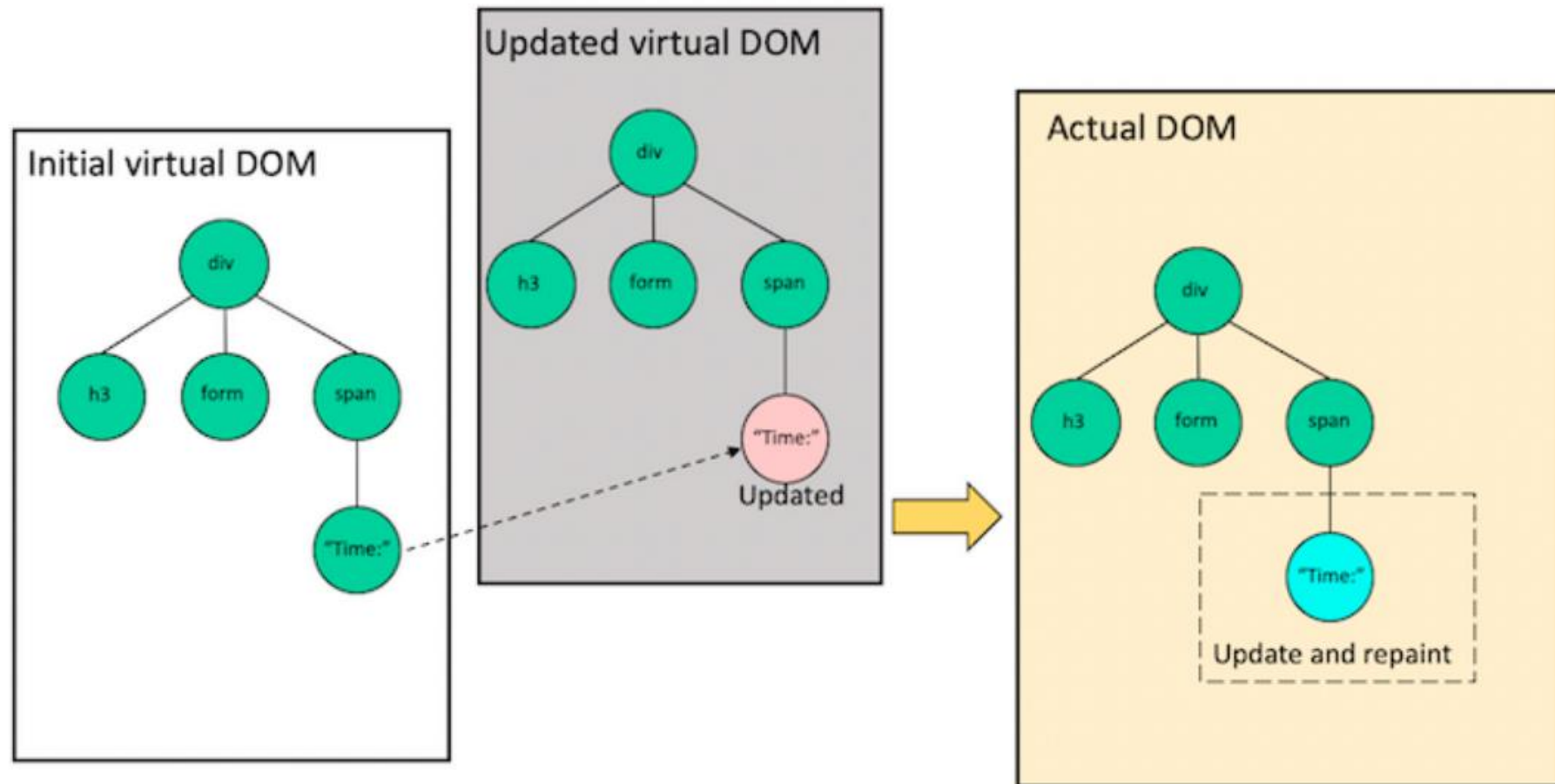
## 3. Nato se izvede primerjava razlik (**diffing**)

- Algoritem, ki primerja prejšnjo (staro) in trenutno (novo) različico virtualnega DOM-a in opredeli, kje se razlikujeta
- Rezultat **diffinga** je seznam identificiranih sprememb
  - npr. katere komponente so bile dodane, odstranjene, spremenjene...
- React nudi zelo učinkovit in relativno hiter algoritem

# Kako deluje React?

4. Sledi uskljevanje (**reconciliation**), ki poskrbi, da se opredeljene spremembe sinhronizirajo z dejanskim DOM-om
- Sinhronizacija poteka v paketu, kar zmanjša št. posodobitev DOM-a
  - Algoritem usklajevanja je pred verzjo React 16 temeljil na skladu (**stack-based**)
  - React 16 je vpeljala nov algoritem uskljevanja, imenovan **Fiber**

# Kako deluje React?



# Kako deluje React?

- **Stack-based reconciliation** v verzijah React < 16 temelji na skladu opravil
- Deluje tako, da rekurzivno obišče vsa vozlišča in gradi sklad opravil, potrebnih za posodobitev DOM-a
  - Ni mogoče ustaviti in nadaljevati dela
  - Ne pozna prioritete opravil
  - Sinhron pristop, kjer se blokira glavna nit

# Kako deluje React?

- Verzija React 16 je vpeljala nov algoritem uskljevanja, imenovan **Fiber**
  - Vpelje koncept vlaken (**fiber**), ki predstavlja enoto dela (**unit of work**)
  - Enoto dela lahko ustvarimo, nadaljujemo z izvajanjem, dajemo prioritete...
  - Fiber deluje **asinhrono**!
  - Zagotovi boljše delovanje in odzivnost aplikacij
  - Prehod na Fiber ni imel vpliva na delo razvijalcev aplikacij React
- Primerjava delovanja obeh algoritmov:  
<https://claudiopro.github.io/react-fiber-vs-stack-demo/>

# Kako deluje React?

---

- Povzetek:
  - React je knjižnica za implementacijo spletnih strani in aplikacij
  - React uporablja JSX, poseben jezik, ki je podoben HTML
  - JSX sintaksa se pretvori v kodo JavaScript, ki se lahko izvaja v brskalniku
  - Poznamo tudi koncept stanja (spomina) komponente
  - Ko se stanje spremeni, se izvede **diffing**, da pridobimo seznam sprememb
  - **Diffing** primerja prejšnjo in trenutno različico aplikacije
  - V ta namen uporabljamo **virtualni DOM**, ki je kopija pravega DOM-a
  - Na koncu sledi usklajevanje (**reconciliation**) virtualnega DOM-a s pravim



# Uvod v React

Vzpostavitev okolja

# Vzpostavitev okolja



**Andrew Clark**

@acd lite



If you use React, you should be using a React framework. If your existing app doesn't use a framework, you should incrementally migrate to one. If you're creating a new React project, you should use a framework from the beginning.

8:52 PM · Jan 23, 2023 · **790.7K** Views



# Vzpostavitev okolja

- Vzpostavitev okolja lahko dosežemo z uporabo ogrodij React (npr. Next.js, Remix, Gatsby) ali orodij oz. brez
- V nadaljevanju si bomo pogledali naslednja ogrodja:
  - Brez orodij
  - Orodje *create-react-app*
  - Orodje *vite*

# Vzpostavitev okolja – brez orodij

- Vzpostavitev okolja brez ogrodji ali orodij
  - Predpogoj: namestitev Node.js
- Koraki (poenostavljeno):
  1. Ustvarimo imenik in package.json datoteko
  2. Namestimo potrebne knjižnice (**node modules**)
  3. Ustvarimo strukturo aplikacije
  4. Dodamo **Webpack** in **Babel**
  5. Zagon aplikacije



# Vzpostavitev okolja brez orodij

1. Ustvarimo imenik in  
package.json datoteko

```
D:\Installation Files\apps>cd from-scratch  
  
D:\Installation Files\apps\from-scratch>npm init -y  
Wrote to D:\Installation Files\apps\from-scratch\package.json:  
  
{  
  "name": "from-scratch",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

# Vzpostavitev okolja – brez orodij

## 2. Namestimo potrebne knjižnice

- Med pomembnejšimi:
  - *Webpack*: združi vse datoteke JavaScript v eno
  - *babel-loader*: deluje z Webpack da pretvori ES6+ v verzijo, podprto s strani več brskalnikov
  - *@babel/preset-react*: razširi podporo Babel, da vključuje tudi JSX
  - *webpack-dev-server*: omogoča uporabo Webpacka z razvojnim strežnikom, ki zagotavlja „živo ponovno nalaganje“ (angl. live reloading)
  - *css-loader*: omogoča Webpacku pretvorbo datoteke CSS v niz JavaScript
  - *style-loader*: vstavi niz JavaScript v HTML DOM

```
D:\Installation Files\apps\from-scratch>npm i webpack babel-loader @babel/preset-react @babel/core babel-preset-react html-webpack-plugin webpack-dev-server css-loader style-loader @babel/plugin-proposal-class-properties webpack-cli -D && npm i react react-dom
npm WARN deprecated core-js@2.6.12: core-js@<3.23.3 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whims, feature detection in old core-js versions could cause a slowdown up to 100x even if nothing is polyfilled. Some versions have web compatibility issues. Please, upgrade your dependencies to the actual version of core-js.

added 434 packages, and audited 435 packages in 1m

54 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 4 packages, and audited 439 packages in 5s

54 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

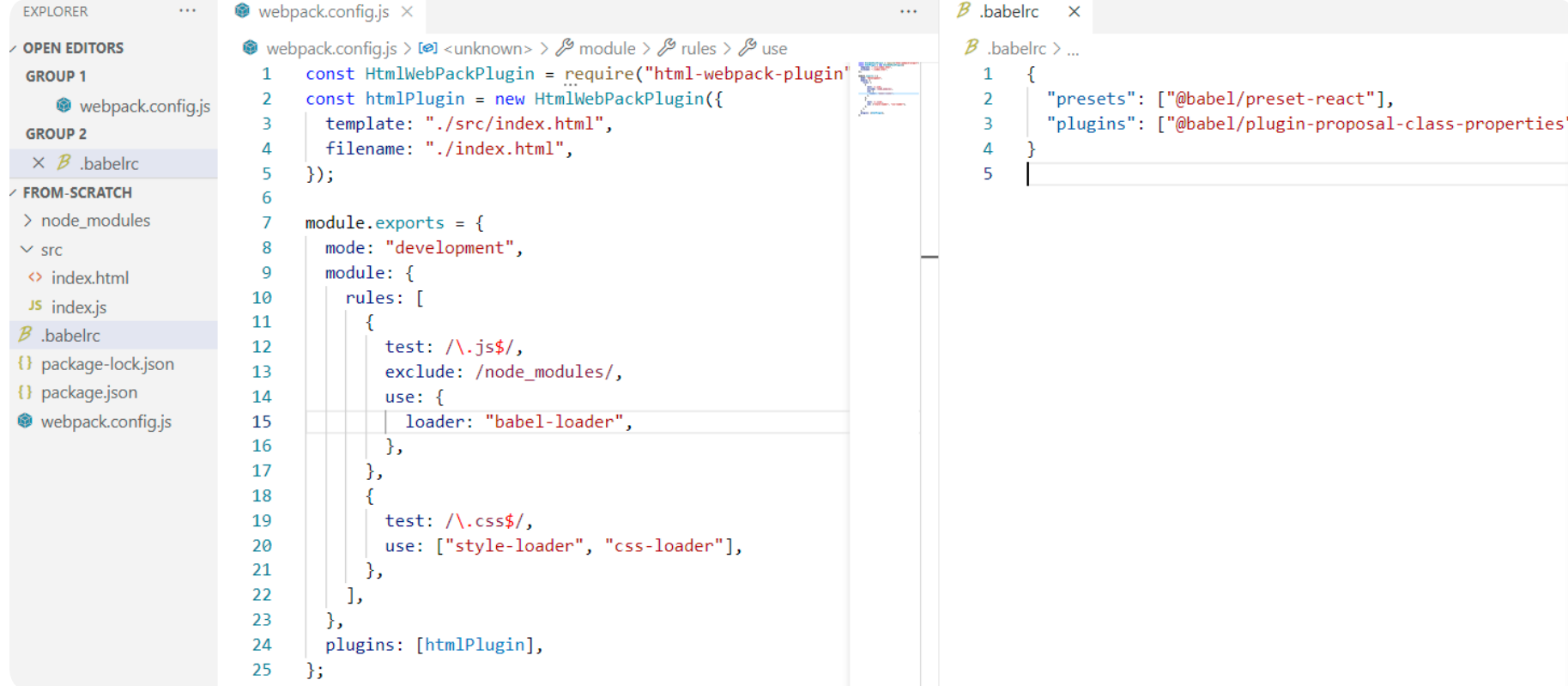
# Vzpostavitev okolja – brez orodij

## 2. Namestimo potrebne knjižnice

- `npm i webpack babel-loader @babel/preset-react @babel/core babel-preset-react html-webpack-plugin webpack-dev-server css-loader style-loader @babel/plugin-proposal-class-properties webpack-cli -D && npm i react react-dom`

Vzpostavitev  
okolja – brez  
orodij

3. Ustvarimo strukturo aplikacije in  
ustrezne datoteke
- app/src/index.html
  - app/src/index.js



Vzpostavitev  
okolja – brez  
orodij

#### 4. Dodamo **Webpack** in **Babel**

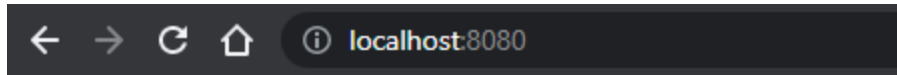
- app/webpack.config.js
- app/.babelrc



# Vzpostavitev okolja – brez orodij

## 5. Zagon aplikacije

- V package.json posodobimo ukaz  
"start": "webpack serve --config webpack.config.js"
- Poženemo `npm run start`
- Odpremo <http://localhost:8080/> stran v brskalniku



**This is my React app!**

```
package.json > {} scripts > start
1  {
2    "name": "from-scratch",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "webpack serve --config webpack.config.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "devDependencies": {
13     "@babel/core": "^7.21.3",
14     "@babel/plugin-proposal-class-properties": "^7.18.6",
15     "@babel/preset-react": "^7.18.6",
16     "babel-loader": "^9.1.2",
17     "babel-preset-react": "^6.24.1",
18     "css-loader": "^6.7.3",
19     "html-webpack-plugin": "^5.5.0",
20     "style-loader": "^3.3.2",
21     "webpack": "^5.76.3",
22     "webpack-cli": "^5.0.1",
23     "webpack-dev-server": "^4.13.1"
24   },
25   "dependencies": {
26     "react": "^18.2.0",
27     "react-dom": "^18.2.0"
28   }
29 }
30
```



# Vzpostavitev okolja – create-react-app

- Uporaba orodja **create-react-app** ima naslednje prednosti:
  - Vzpostavitev okolja z enim ukazom
  - Ni potrebno razumeti vseh podrobnosti konfiguracije ampak se osredotočimo na razvoj
  - Vzdržujemo samo eno odvisnost
  - Uporablja webpack, Babel, ESLint in ostalo
  - Možnost **eject**
- `npx create-react-app my-app`
- `npm run build`

# Vzpostavitev okolja – create-react-app

- Podpira predloge (**templates**)
  - `npx create-react-app my-app --template [template-name]`
  - Npr: `npx create-react-app my-app --template typescript`
  - Seznam predlog: [https://www.npmjs.com/search?q=cra-template-\\*](https://www.npmjs.com/search?q=cra-template-*)

# Vzpostavitev okolja – create-react-app

- Struktura projekta
  - Vse datoteke JavaScript in HTML morajo biti znotraj imenika src
  - Za delovanje mora obstajati public/index.html in src/index.js (točno na tej poti in s tem imenom)

```
my-app/  
  README.md  
  node_modules/  
  package.json  
  public/  
    index.html  
    favicon.ico  
  src/  
    App.css  
    App.js  
    App.test.js  
    index.css  
    index.js  
    logo.svg
```

# Vzpostavitev okolja – create-react-app

**Create React App is Finally Dead**  
55K views • 9 days ago

Theo - t3.gg

Sorry for the short vid, edited this in like 5 minutes lol just wanted to get the good news out! Check out the new docs site ...

4K

Intro | Recommendations | Conclusion 3 chapters

**Create React App is Officially DEAD!**  
6.5K views • 7 days ago

Brian Design

Create React App is Officially DEAD! The react dev website has just been released and there is no mention of Create React App ...

**REACT.DEV LAUNCHED! Goodbye Create React App?**  
88K views • 9 days ago

Codevolution

Business - codevolution.business@gmail.com React Create React App.

**Don't Use Create React App in 2023**  
64K views • 3 months ago

Theo - t3.gg

My last Create React App video was way too long and I wanted to start a new series so...yolo. Create T3 App: <https://create.t3.gg/> ...

**Is Create-React-App Dead!?**  
1.7K views • 2 months ago

Nx - Smart, Fast, Extensible

Well, at least that's a discussion @t3dotgg kicked off...and a lot of developers agree, me included! In fact, we saw this coming and ...

Intro | PR to remove CRA | Replace CRA with what? | Using the Nx React Standalone template | Usin... 8 chapters

**Stop Using Create React App**  
267K views • 4 months ago

Web Dev Simplified

Create React App is an incredible tool that made it easy to setup and work with React, but as the years have gone by CRA has ...

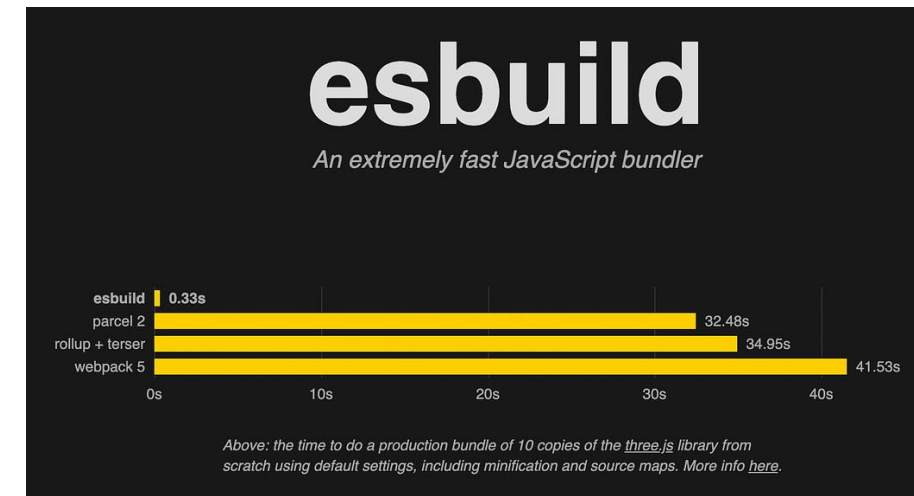
Introduction | CRA Cons | Vite | Astro/Gatsby | Next.js | Full Stack Frameworks 6 chapters

# Vzpostavitev okolja – vite

- Vite (francoska beseda za hiter) je trenutno najbolj priljubljeno orodje za izgradnjo odjemalskih spletnih aplikacij
- Razvil ga je **Evan You** leta **2020**
- Privzeto ga uporablja ogrodje **Vue**
- Uporablja sodobne funkcije spletnih brskalnikov (npr. **ES moduli**) za hitro gradnjo in odziven čas
- Podpira **več ogrodij** za razvoj aplikacij, med drugimi tudi React in Vue

# Vzpostavitev okolja – vite

- V primerjavi s CRA je vite:
  - Hitrejši pri zagonu strežnika za razvoj
  - Hitrejši pri posodobitvi aplikacije ob spremembi datotek
  - Podpora sodobnim tehnologijam
    - Hot Module Replacement(HMR), ki temelji na ESM (EMACScript modules)
    - Ne uporablja Webpack-a (Webpack združi (bundle) celotno aplikacijo preden jo zažene)
  - Uporaba esbuild
  - Manjša aplikacija (odvisno od dodatnih modulov)
    - V osnovi CRA 338 MB, vite 110 MB



# Vzpostavitev okolja – vite

```
D:\Installation Files\apps>npm create vite@latest my-react-app -- --template react-ts
Scaffolding project in D:\Installation Files\apps\my-react-app...

Done. Now run:

  cd my-react-app
  npm install
  npm run dev

D:\Installation Files\apps>cd my-react-app

D:\Installation Files\apps\my-react-app>npm i

added 83 packages, and audited 84 packages in 35s

9 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\Installation Files\apps\my-react-app>npm run dev

> my-react-app@0.0.0 dev
> vite

VITE v4.2.1 ready in 4076 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h to show help
```

