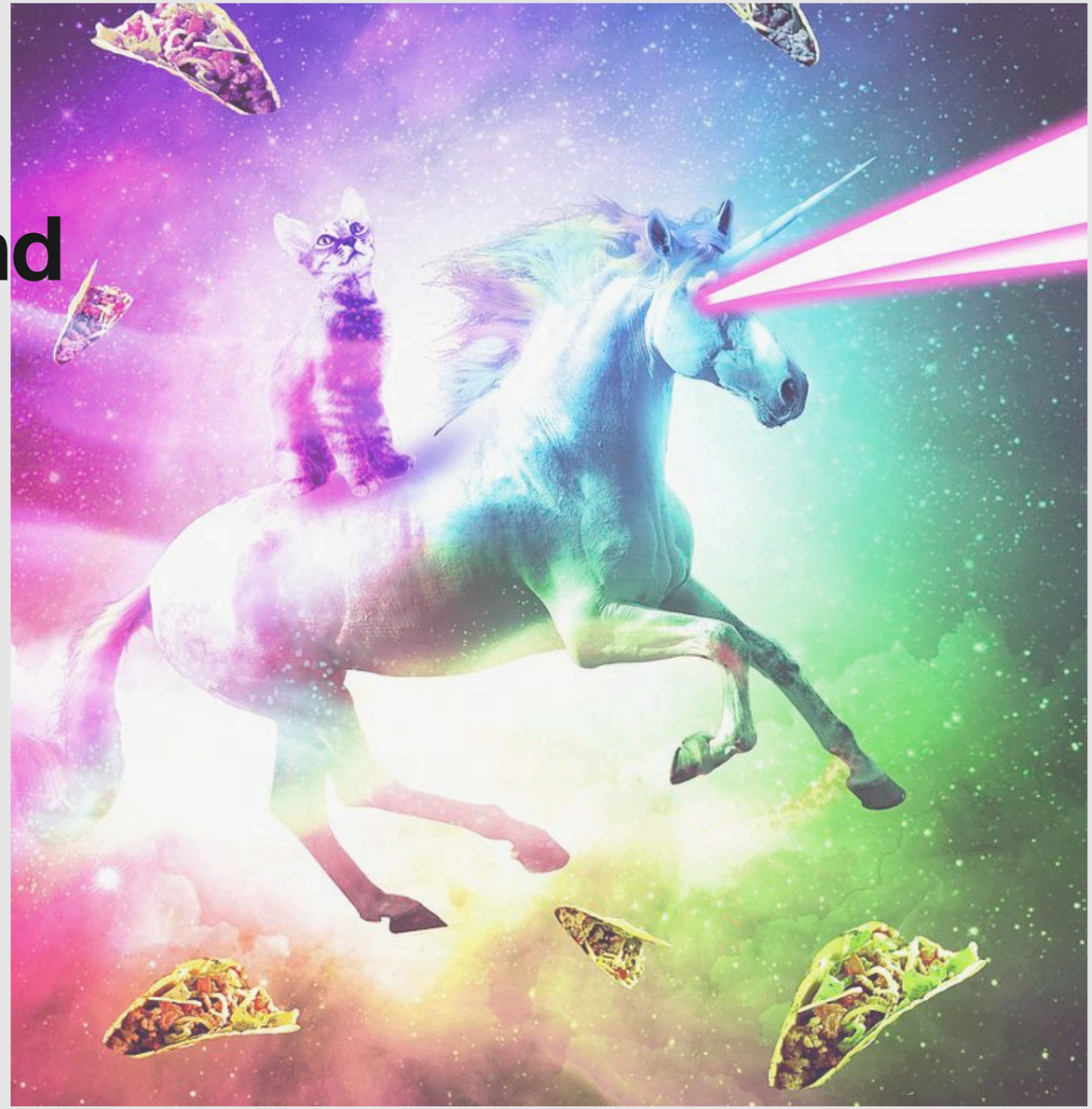
Ekosystem React

JavaScript Developer 2020/2021 *(jeśli przetrwamy)*

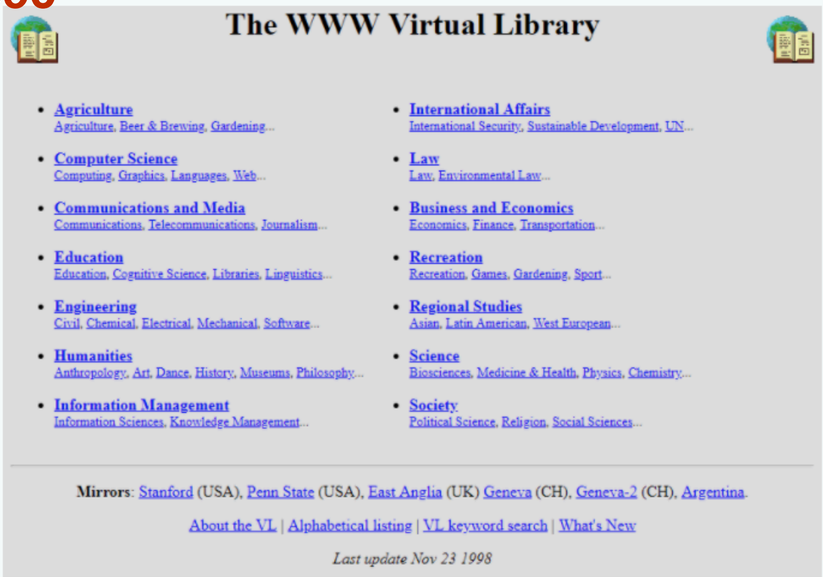
Zacznijmy od podstaw:

Magiczny świat Front End



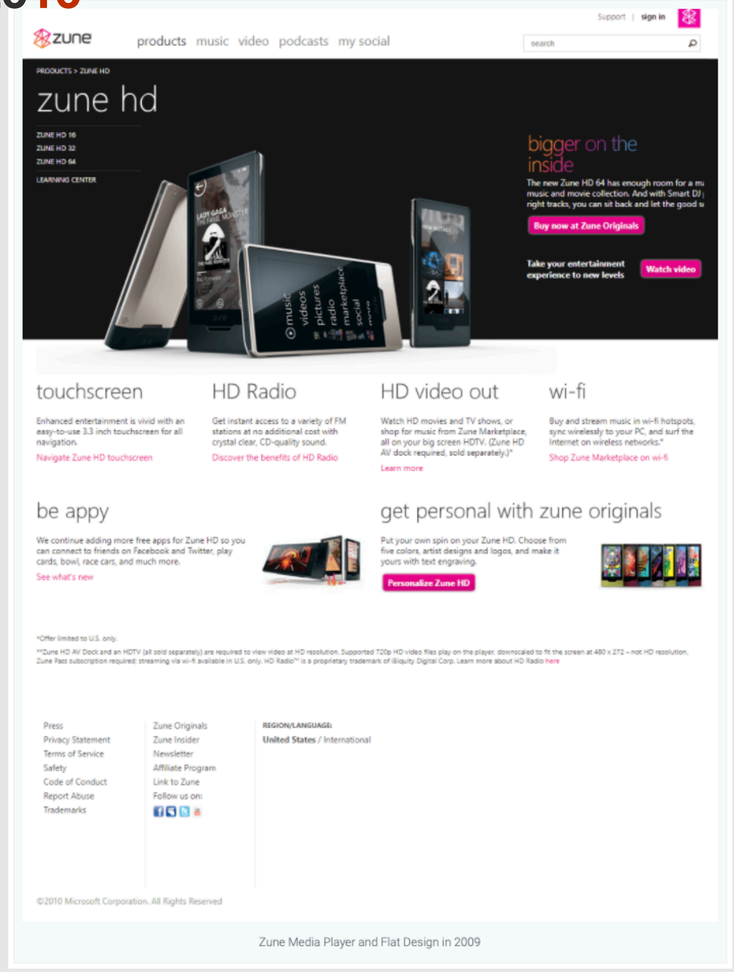
Trudne początki

1990

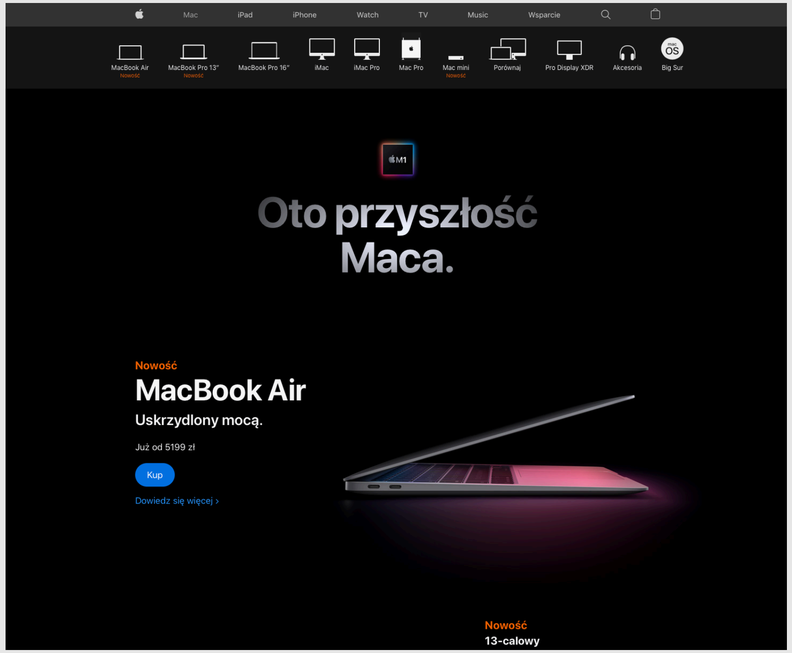


2000

2010



Zune Media Player and Flat Design in 2009



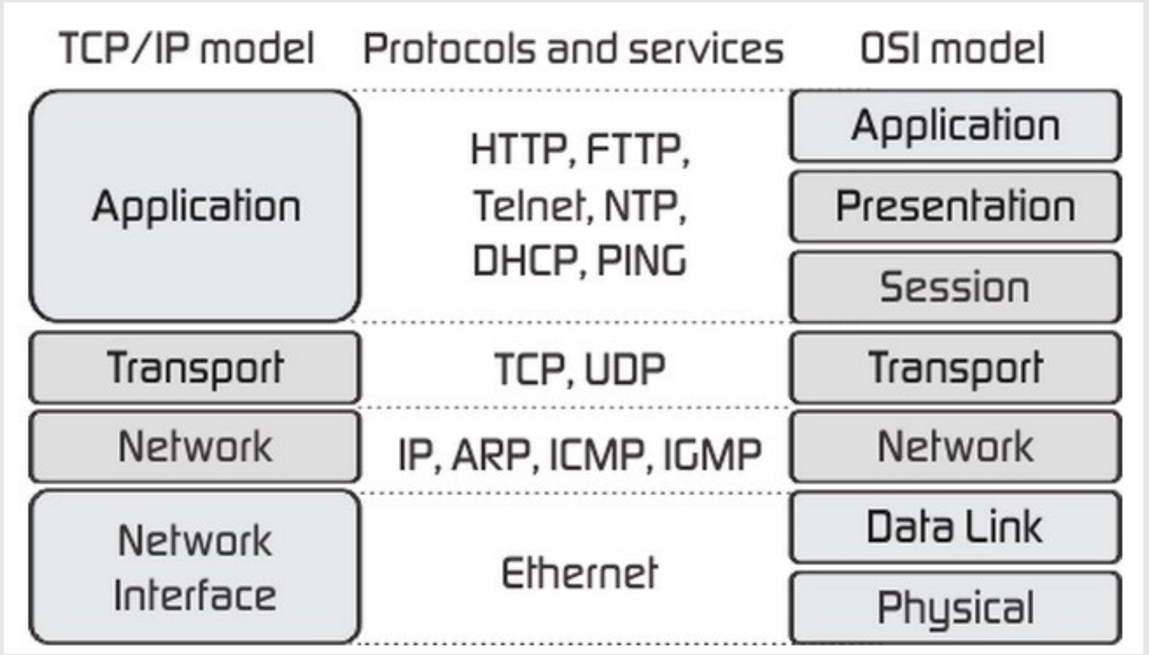
Protokół HTTP

HyperText Transfer Protocol

PROTOKÓŁ KOMUNIKACYJNY W WARSTWIE APLIKACJI MODELU OSI

Wymienia dane w zestawionym połączeniu

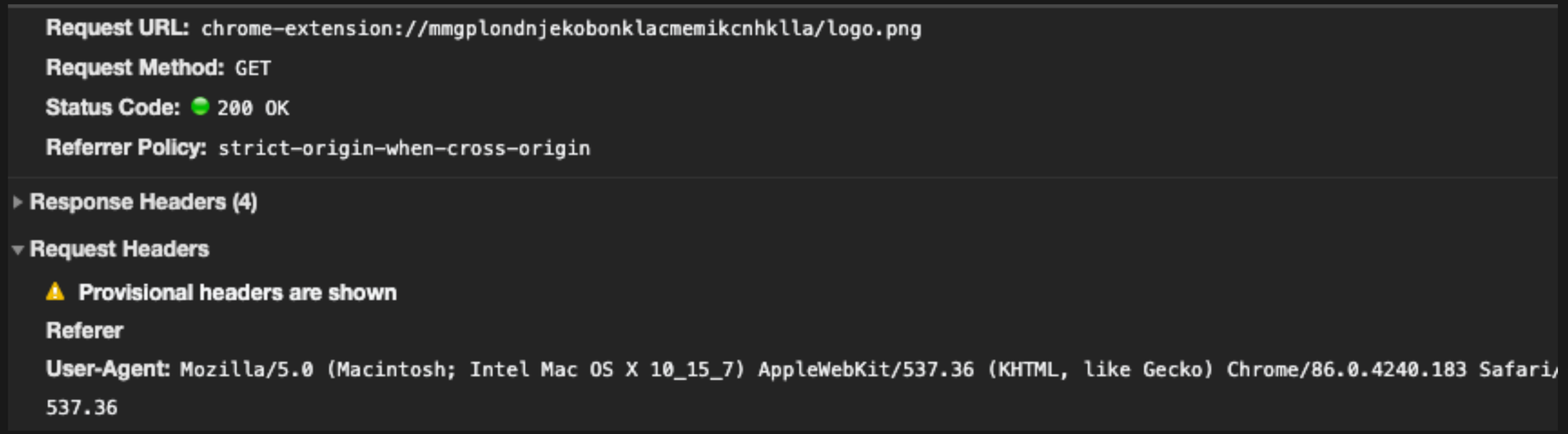
Wymiana odbywa się przy wykorzystaniu akcji **request** i **response**



HTTP Request

- METHOD:
  - **GET**
  - **POST**
  - PUT
  - PATCH
  - DELETE
- PATH
- VERSION
- HEADERS
- BODY

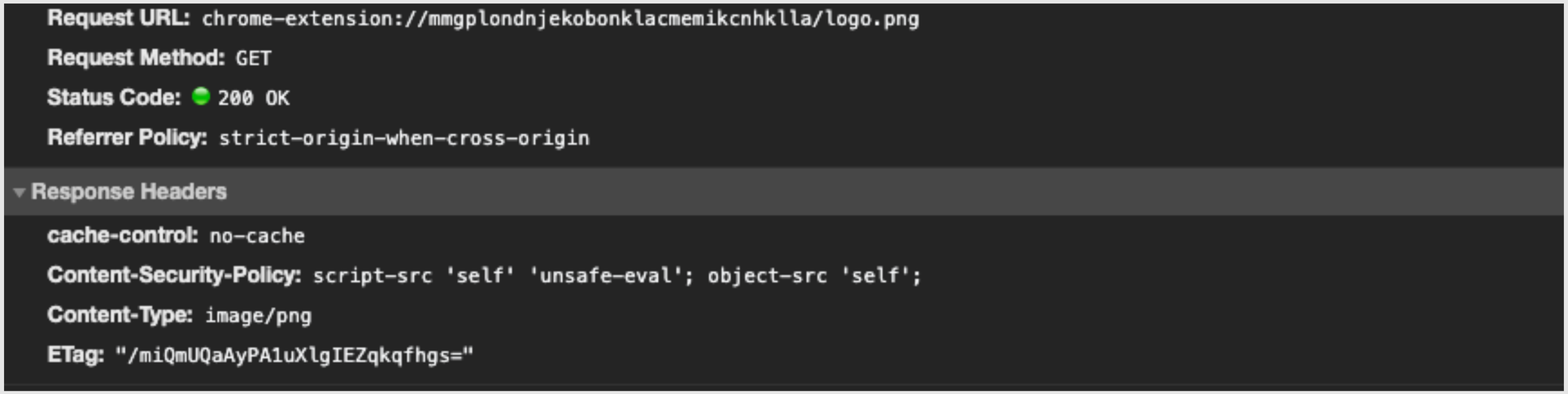
Example GET request



HTTP Response

- STATUS CODE
- STATUS MESSAGE
- VERSION
- HEADERS
- BODY

Example GET response



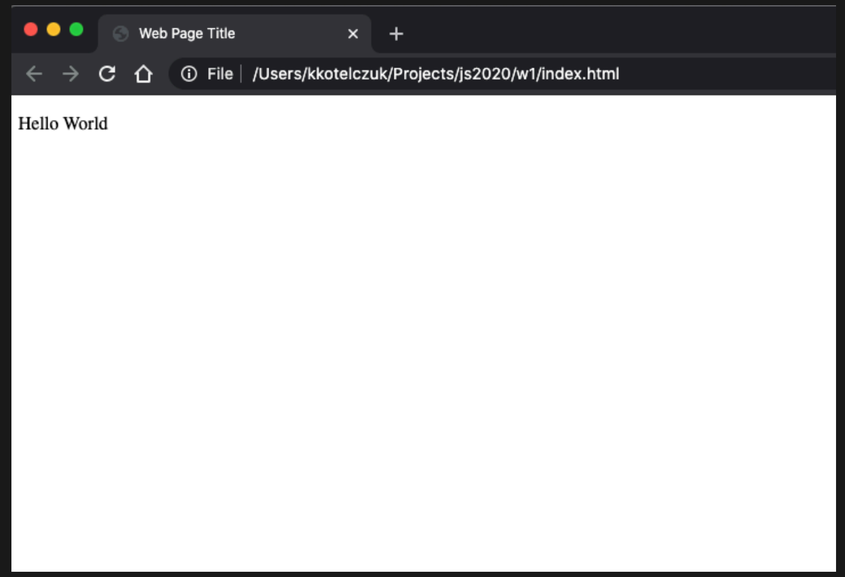
HTML

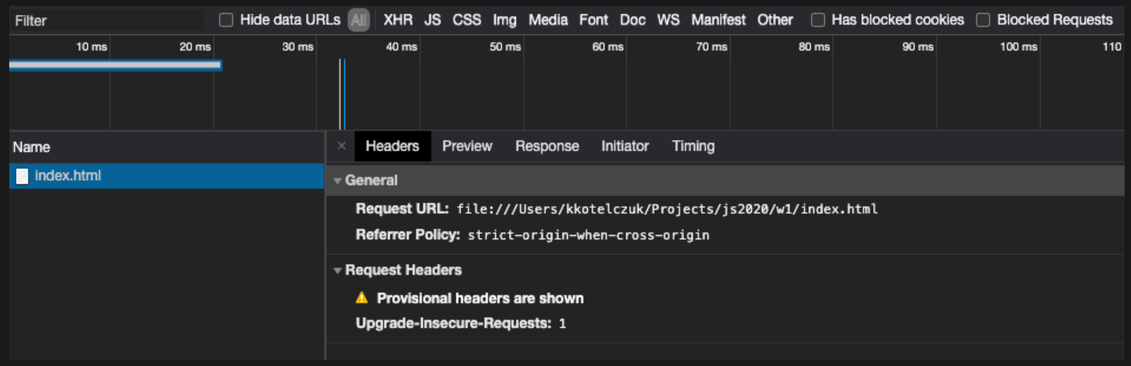
HyperText Markup Language

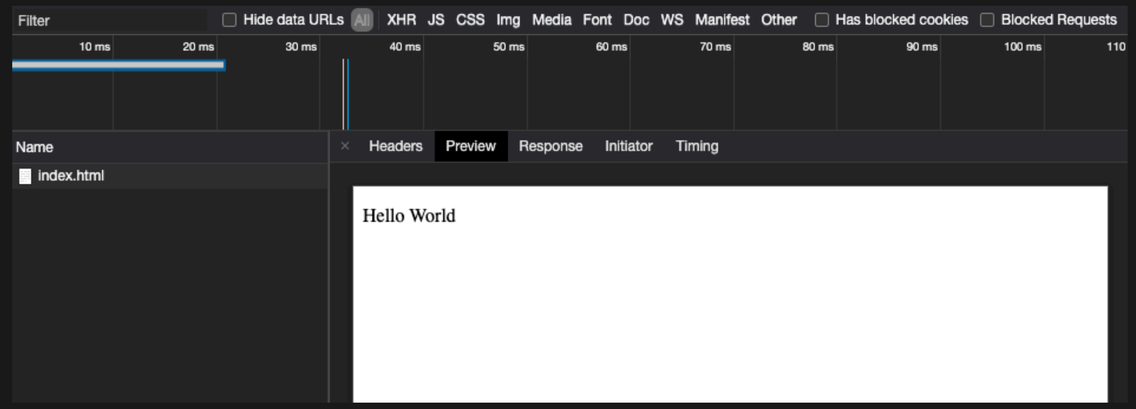
Podstawowa struktura HTML

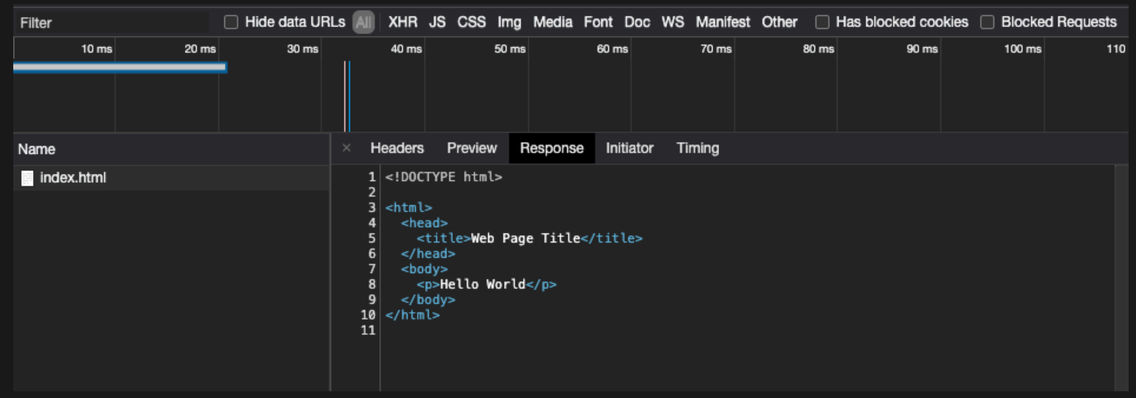
```
<!DOCTYPE html>

<html>
    <head>
        <title>Web Page Title</title>
    </head>
    <body>
         To też zadziała
        <p>Hello World</p>
    </body>
</html>
```

```
<Deklaracja typu pliku>

<Ramka dokumentu >
    <Znacznik sekcji nagłówka strony>
        <Znacznik nazwy strony >Web Page Title</Znacznik zakończenia nazwy strony>
    </Znacznik zakończenia sekcji nagłówka strony>
    <Znacznik "ciała" dokumentu>
        <Znacznik paragrafu>Hello World</Znacznik zakończenia paragrafu>
    </Znacznik zakończenia "ciała" dokumentu>
</Ramka dokumentu >
```


Web Page Title
File | /Users/kkotelczuk/Projects/js2020/w1/index.html
Hello World


Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Has blocked cookies
Blocked Requests
10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110
Name
index.html
Headers Preview Response Initiator Timing
General
Request URL: file:///Users/kkotelczuk/Projects/js2020/w1/index.html
Referrer Policy: strict-origin-when-cross-origin
Request Headers
Provisional headers are shown
Upgrade-Insecure-Requests: 1


Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Has blocked cookies
Blocked Requests
10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110
Name
index.html
Headers Preview Response Initiator Timing
Hello World


Filter
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
Has blocked cookies
Blocked Requests
10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110
Name
index.html
Headers Preview Response Initiator Timing
<!DOCTYPE html>
<html>
<head>
<title>Web Page Title</title>
</head>
<body>
Hello World
</body>
</html>

CSS

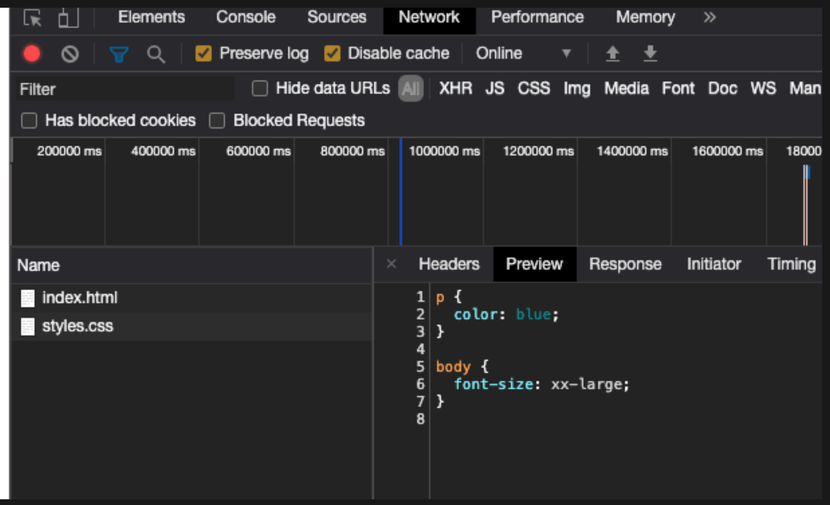
Cascading Style Sheets

```
p {
  color: blue;
}

body {
  font-size: xx-large;
}
```

```html
<!DOCTYPE html>

<html>
  <head>
    <title>Web Page Title</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

```css
p {
  color: blue;
}

body {
  font-size: xx-large;
}
```

Hello World



JavaScript

Tu już prawdopodobnie wszytko wiecie

Krótka lekcja historii

1993 Legendarne powstanie pierwszej wersji języka

A jego imię to JavaScript 1995

ŚREDNIOWIECZE

2005 Asynchroniczność

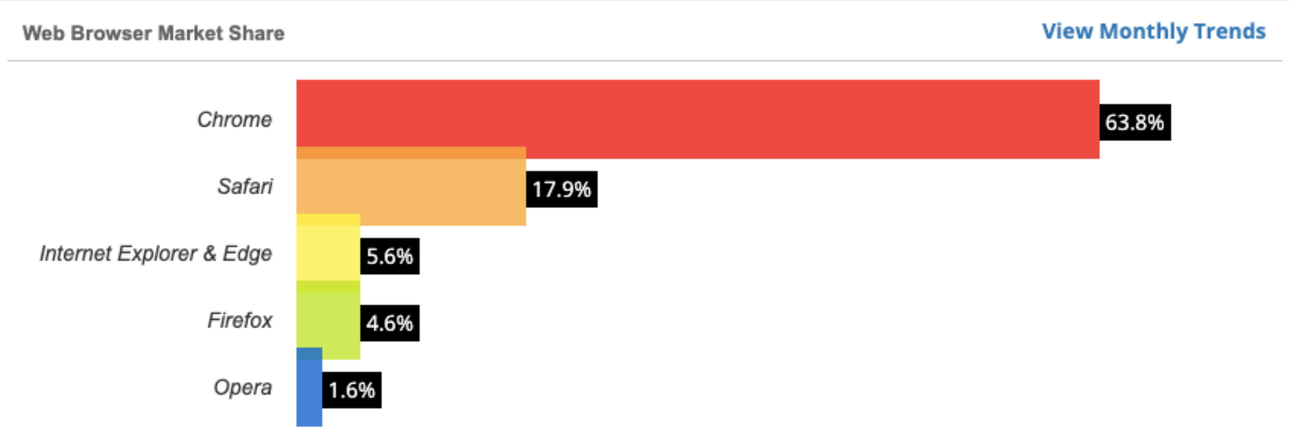
Odrodzenie dzięki jQuery 2006

2008 Wchodzimy na obroty z V8

Viva la revolution 2009

2016 Future is here

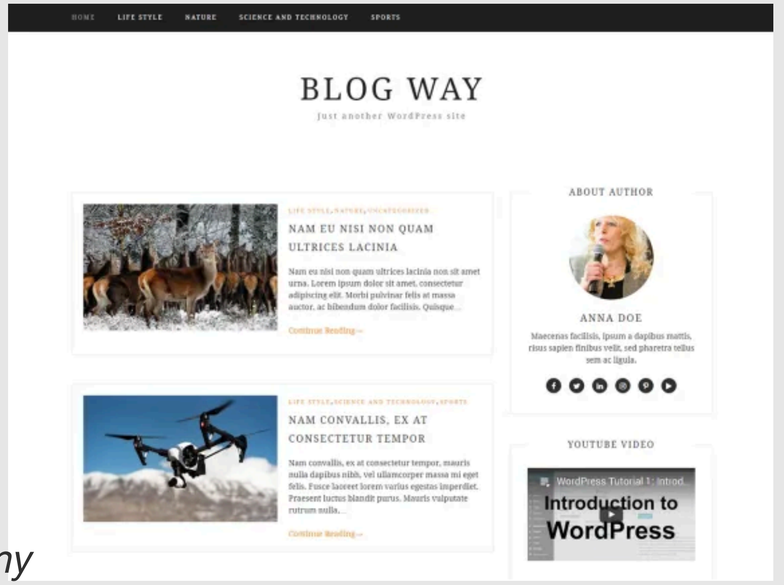
BROWSERS WARS




SPA
MPA

Multi Page Application

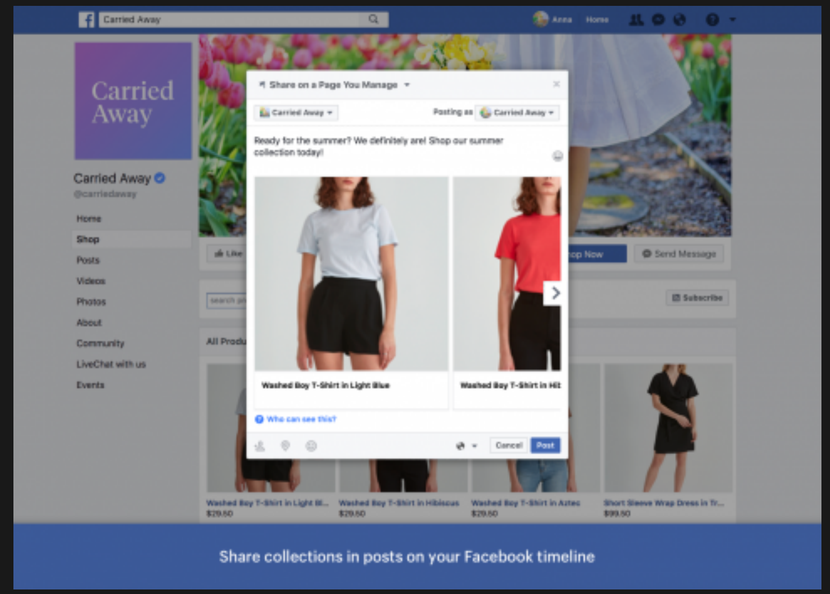
- Klasyczne strony
- Każde przejście na podstronę wymaga przeładowania zasobów
- Łatwe do pozycjonowania
- Klarowna struktura aplikacji



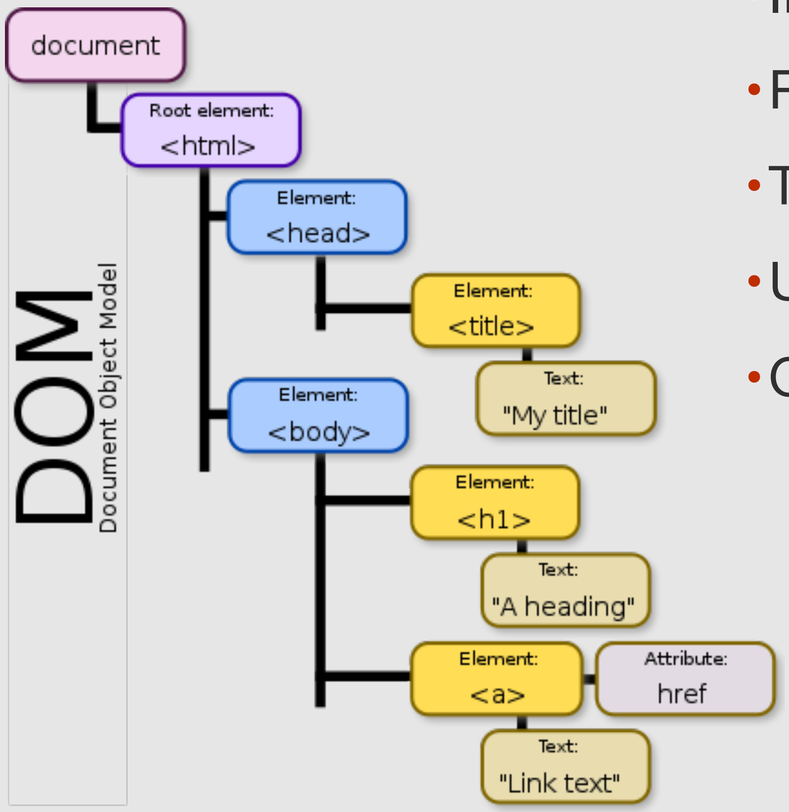
Dziś, wykorzystujemy głównie tam gdzie potrzebujemy dobrego SEO oraz nie mamy potrzeby często zmieniać zawartości strony

Single Page Application

- Strona składa się z 1 pliku js + 1 pliku HTML (zazwyczaj)
- Przechodzenie po stronie nie wymaga ciągłego ładowania wszystkich danych
- Znaczna część ciężaru aplikacji jest po stronie komputera użytkownika
- Pierwsze ładowanie trwa dłużej niż w MPA
- Proces twórczy jest bardziej skomplikowany



DOCUMENT OBJECT MODEL



- INTERFEJS
- Reprezentacja dokumentu HTML w formie drzewa logicznego
- Tworzony jest przez przeglądarkę
- Umożliwia JavaScript na modyfikowanie HTML’a
- Oraz dynamiczne reagowanie na wydarzenia

React

Rys historyczny

React v0.3.3

June 21, 2013 dodane przez Paul O'Shannessy

We have a ton of great stuff coming in v0.4, but in the meantime we're releasing v0.3.3. This release addresses some small issues people were having and simplifies our tools to make them easier to use.

React v15.0

April 07, 2016 dodane przez Dan Abramov

We would like to thank the React community for reporting issues and regressions in the release candidates on our issue tracker. Over the last few weeks we fixed those issues, and now, after two release candidates, we are excited to finally release the stable version of React 15.

Biblioteka framework

Kluczowe koncepcje

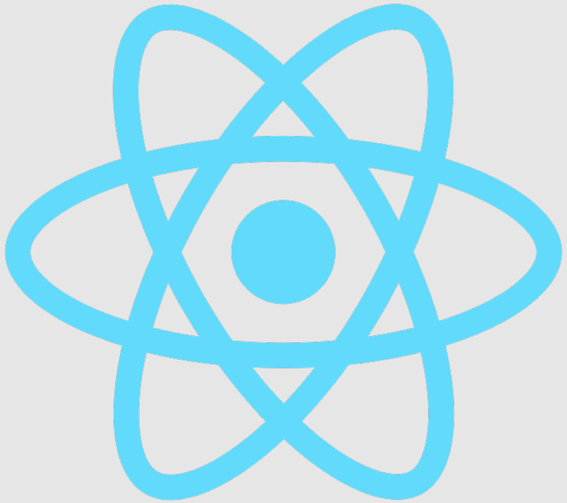
JSX

Wolność wyboru

One way data binding

Komponenty

Virtual DOM



JSX

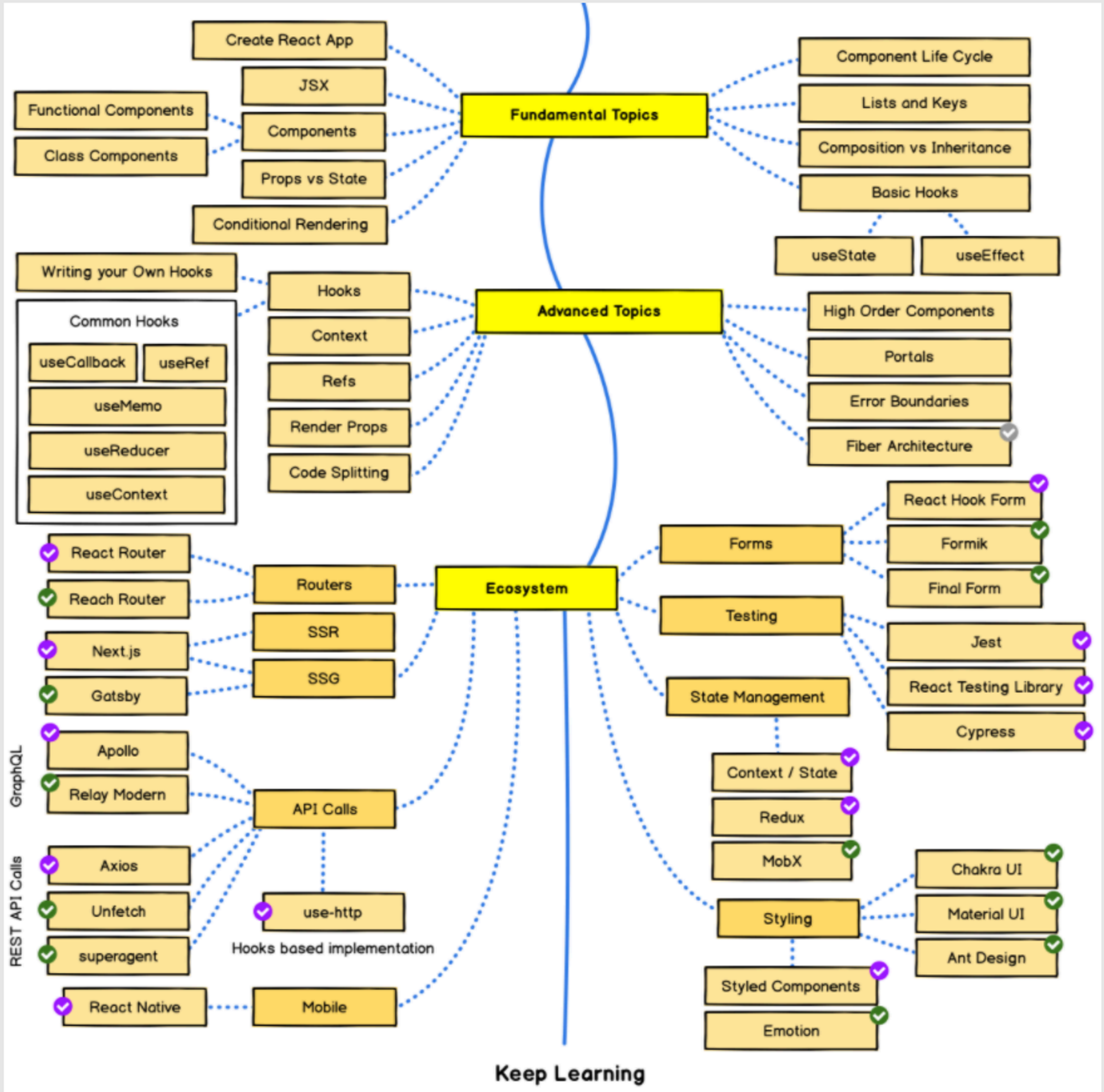
```
renderMyComponent = () => {
    const someData = [1,2,3,4,5,6]
    return (
        <div>
            {someData.map(item => <span>{item}</span>)}
        </div>
    )
}
```

```
<div>
  <span>1</span>
  <span>2</span>
  <span>3</span>
  <span>4</span>
  <span>5</span>
  <span>6</span>
</div>
```

Od czego zacząć

“Strive for progress, **not** perfection.”

– Unknown.



BABEL

Input:

```
// ES2020 nullish coalescing
function greet(input) {
    return input ?? "Hello world";
}
```

Output:

```
function greet(input) {
    return input != null ? input : "Hello world";
}
```

Środowisko programistyczne

01.

Przeglądarka

✅ Chrome, Safari, Edge
❌ Firefox, Opera, Internet Explorer
❓ Brave, Vivaldi

02.

System kontroli wersji Git

✅ Zainstalowany w systemie
❓ Aplikacja desktopowa
❓ Git dodany do konsoli

03.

Edytor tekstu

✅ Kartka papieru + ołówek
❌ Word, Wordpad, OpenOffice
❓ Notatnik
❓ VS CODE, Atom, WebStorm, VIM etc

04.

Niezbędne oprogramowanie

✅ Node.js
❓ Xcode command line tools
❓ Windows build tools

05.

Pomocnicze oprogramowanie

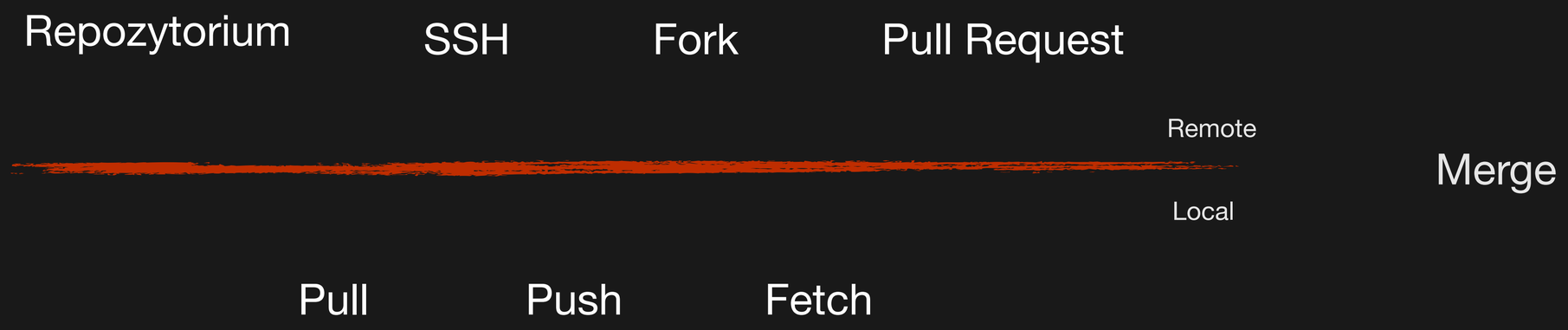
❓ Slack - desktop app
❓ M$ Teams - desktop app
❓ Bash for command line

06.

Warto znać

❓ Skróty klawiaturowe
❓ Strona MDN
❓ Strona z dokumentacją React

Git Flow


Repozytorium
SSH
Fork
Pull Request
Remote
Local
Merge
Pull
Push
Fetch

Renderowanie elementów

ReactDOM

ReactDOM.render(element, rootElement)

rootElement = document.getElementById(‘root’)

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

setInterval(tick, 1000);
```

Komponenty

Koncepcyjnie, komponenty są jak javascriptowe funkcje.

Przyjmują one arbitralne wartości na wejściu (nazywane “właściwościami” (ang. *props*))

zwracają reactowe elementy opisujące, co powinno się pojawić na ekranie.

```
function Welcome(props) {
  return <h1>Cześć, {props.name}</h1>;
}
```

```
class Welcome extends React.Component {
  render() {
    return <h1>Cześć, {this.props.name}</h1>;
  }
}
```

React traktuje komponenty zaczynające się od małej litery jako tagi drzewa *DOM.*
Na przykład, reprezentuje *HTML*-owy znacznik ‘div’,
ale już <Welcome /> reprezentuje komponent i wymaga, aby Welcome było w *zasięgu* (*ang. scope*).

Kompozycja komponentów

```
function Welcome(props) {
  return <h1>Cześć, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Wyodrębnianie komponentów

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <img className="Avatar"
          src={props.author.avatarUrl}
          alt={props.author.name}
        />
        <div className="UserInfo-name">
          {props.author.name}
        </div>
      </div>
      <div className="Comment-text">
        {props.text}
      </div>
      <div className="Comment-date">
        {formatDate(props.date)}
      </div>
    </div>
  );
}
```

```
 Comment(props) {
(
lassName="Comment">
 className="UserInfo">
vatar user={props.author} />
v className="UserInfo-name">
rops.author.name}
iv>
/>
 className="Comment-text">
ops.text}
/>
 className="Comment-date">
rmatDate(props.date)}
/>
>
```

```
function Avatar(props) {
        return (
          <img className="Avatar"
            src={props.user.avatarUrl}
            alt={props.user.name}
          />
        );
      }
```

```
    function UserInfo(props) {
      return (
        <div className="UserInfo">
          <Avatar user={props.user} />
          <div className=“UserInfo-name">
            {props.user.name}
          </div>
        </div>
      );
    }
```

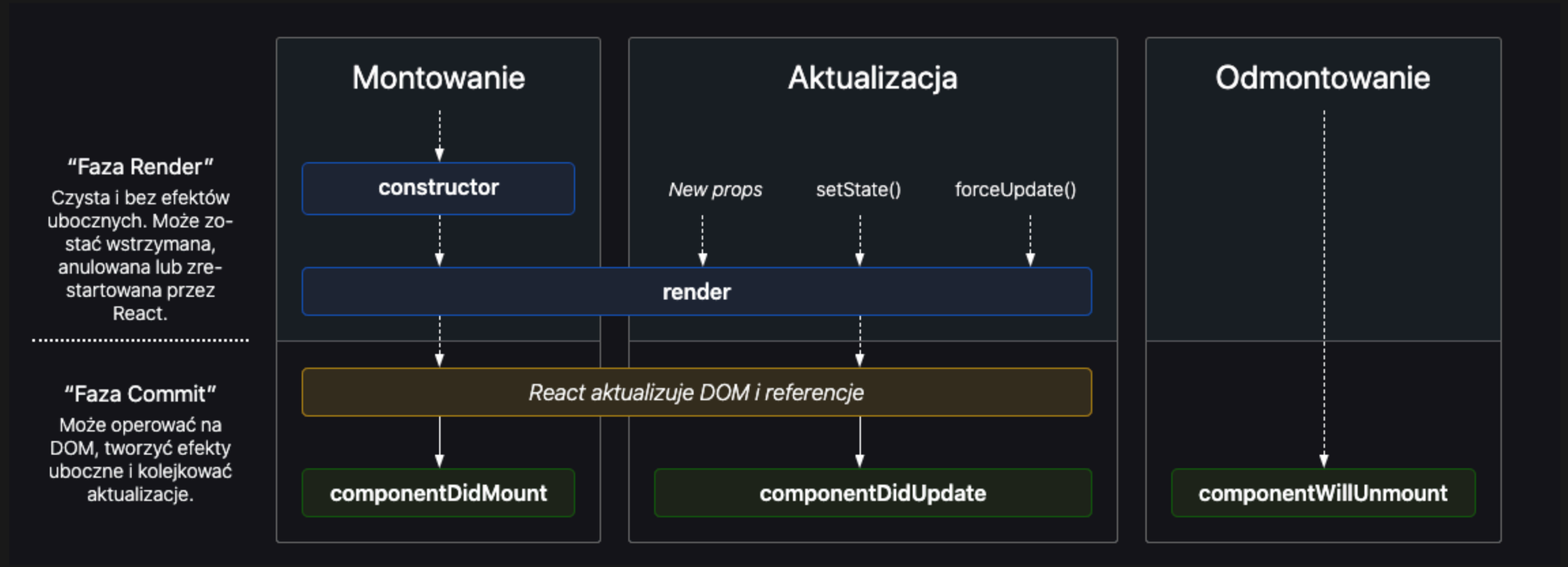
```
function Comment(props) {
  return (
    <div className="Comment">
      <UserInfo user={props.author} />
      <div className=“Comment-text">
        {props.text}
      </div>
      <div className=“Comment-date">
        {formatDate(props.date)}
      </div>
    </div>
  );
}
```

```
function Comment(props) {
        return (
          <div className="Comment">
            <div className="UserInfo">
              <img className="Avatar"
                src={props.author.avatarUrl}
                alt={props.author.name}
              />
              <div className="UserInfo-name">
                {props.author.name}
              </div>
            </div>
            <div className="Comment-text">
              {props.text}
            </div>
            <div className="Comment-date">
              {formatDate(props.date)}
            </div>
          </div>
        );
      }
```

Pure functions

Wszystkie komponenty reactowe muszą zachowywać się jak pure functions - nie mogą modyfikować swoich propsów

Uproszczony cykl życia komponentu



Kompozycja nad dziedziczenie

Stan komponentu

```
function Clock(props) {
  return (
    <div>
      <h1>Witaj, świecie!</h1>
      <h2>Aktualny czas: {props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}

function tick() {
  ReactDOM.render(
    <Clock date={new Date()} />,
    document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

```
ReactDOM.render(
  <Clock />,
  document.getElementById('root')
);
```

```
function Clock(props) {
  const [time, setTime] = useState(new Date())
  return (
    <div>
      <h1>Witaj, świecie!</h1>
      <h2>Aktualny czas: {time.toLocaleTimeString()}.</h2>
    </div>
  );
}
```

```
useEffect(() => {
  this.timerID = setInterval(
    () => this.tick(),
    1000
  );
  return () => clearInterval(this.timerID);
})
```

```
function Clock(props) {
  const [time, setTime] = useState(new Date())

  useEffect(() => {
    this.timerID = setInterval(
      () => this.tick(),
      1000
    );
    return () => clearInterval(this.timerID);
  })

  return (
    <div>
      <h1>Witaj, świecie!</h1>
      <h2>Aktualny czas: {time.toLocaleTimeString()}.</h2>
    </div>
  );
}


  ReactDOM.render(
    <Clock />,
    document.getElementById('root')
  );
```

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.tick(),
      1000
    );
  }

  componentWillUnmount() {
    clearInterval(this.timerID);
  }

  tick() {
    this.setState({
      date: new Date()
    });
  }

  render() {
    return (
      <div>
        <h1>Witaj, świecie!</h1>
        <h2>Aktualny czas: {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}

ReactDOM.render(
  <Clock />,
  document.getElementById('root')
);
```

Obsługa zdarzeń

Przez zdarzenie rozumiemy obsługę akcji wykonanych przez użytkownika

Zdarzenia reactowe pisane są camelCasem, a nie małymi literami.

W JSX procedura obsługi zdarzenia przekazywana jest jako funkcja, a nie łańcuch znaków.

```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

```
<a href="#" onclick="console.log('Kliknięto w link.'); return false">
  Kliknij mnie
</a>
```

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('Kliknięto w link.');
  }
  return (
    <a href="#" onClick={handleClick}>
      Kliknij mnie
    </a>
  );
}
```

SyntheticEvent

Enkapsulacja

Czyli renderowanie warunkowe

```jsx
function UserGreeting(props) {
  return <h1>Witamy ponownie!</h1>;
}

function GuestGreeting(props) {
  return <h1>Proszę się zarejestrować.</h1>;
}
```

```jsx
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}
ReactDOM.render(
  // Spróbuj zmienić na isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

&&

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Cześć!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          Masz {unreadMessages.length} nieprzeczytanych wiadomości.
        </h2>
      }
    </div>
  );
}

const messages = ['React', 'Re: React', 'Re:Re: React'];
ReactDOM.render(
  <Mailbox unreadMessages={messages} />,
  document.getElementById('root')
);
```

Uwaga - wyjątek

```
render() {
  const count = 0;
  return (
    <div>
      { count && <h1>Messages: {count}</h1>}
    </div>
  );
}
```

Inline if-else

```
render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      The user is <b>{isLoggedIn ? 'currently' : 'not'}</b> logged in.
    </div>
  );
}
```

If Guard

```
function WarningBanner(props) {
  if (!props.warn) {
    return null;
  }

  return (
    <div className="warning">
      Warning!
    </div>
  );
}
```

Listy i klucze

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) ⇒ number * 2);
console.log(doubled)

//logged: 2, 4, 6, 8, 10
```

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```

```
ReactDOM.render(
  <ul>{listItems}</ul>,
  document.getElementById('root')
);
```

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li>{number}</li>
  );
  return (
    <ul>{listItems}</ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
ReactDOM.render(
  <NumberList numbers={numbers} />,
  document.getElementById('root')
);
```

```
<li key={number.toString()}>
```

```jsx
function ListItem(props) {
  const value = props.value;
  return (
   <li key={value.toString()}>
      {value}
    </li>
  );
}

function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
   <ListItem value={number} />
  );
  return (
    <ul>
      {listItems}
    </ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
ReactDOM.render(
  <NumberList numbers={numbers} />,
  document.getElementById('root')
);
```

```jsx
function ListItem(props) {

  return <li>{props.value}</li>;
}

function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>

    <ListItem key={number.toString()} value={number} />
  );
  return (
    <ul>
      {listItems}
    </ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
ReactDOM.render(
  <NumberList numbers={numbers} />,
  document.getElementById('root')
);
```

Formularze

```
<form>
  <label>
    Name:
    <input type="text" name="name" />
  </label>
  <input type="submit" value="Submit" />
</form>
```

```jsx
const NameForm = () => {
  const [value, setValue] = useState('');

  const handleChange = event => {
    setValue(event.target.value);
  };

  const handleSubmit = event => {
    alert('A name was submitted: ' + value);
    event.preventDefault();
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={value} onChange={handleChange} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
};
```

```
<textarea>
  Hello there, this is some text in a text area
</textarea>
```

```
<textarea value={this.state.value} onChange={this.handleChange} />
```

```jsx
<option selected value="coconut">Coconut</option>
```

```jsx
<select value={value} onChange={handleChange}>
  <option value="grapefruit">Grapefruit</option>
  <option value="lime">Lime</option>
  <option value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

```jsx
<select multiple={true} value={['B', 'C']}>
```

```
class Reservation extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isGoing: true,
      numberOfGuests: 2
    };

    this.handleInputChange = this.handleInputChange.bind(this);
  }

  handleInputChange(event) {
    const target = event.target;
    const value = target.type === 'checkbox' ? target.checked : target.value;
    const name = target.name;
    this.setState({
      [name]: value
    });
  }

  render() {
    return (
      <form>
        <label>
          Is going:
          <input
            name="isGoing"
            type="checkbox"
            checked={this.state.isGoing}
            onChange={this.handleInputChange} />
        </label>
        <br />
        <label>
          Number of guests:
          <input
            name="numberOfGuests"
            type="number"
            value={this.state.numberOfGuests}
            onChange={this.handleInputChange} />
        </label>
      </form>
    );
  }
}
```