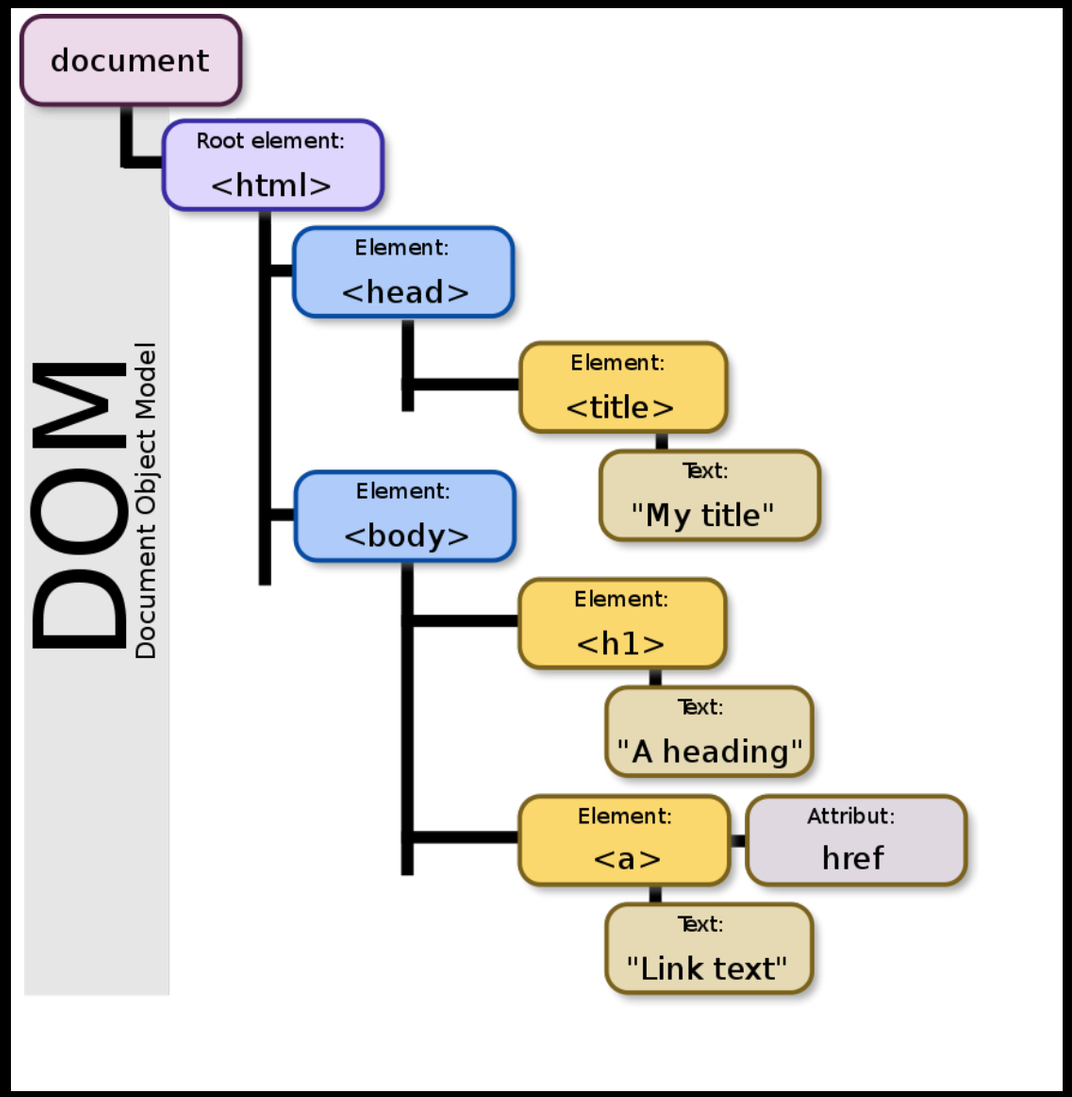
# REACT & JSX

# DOCUMENT OBJECT MODEL

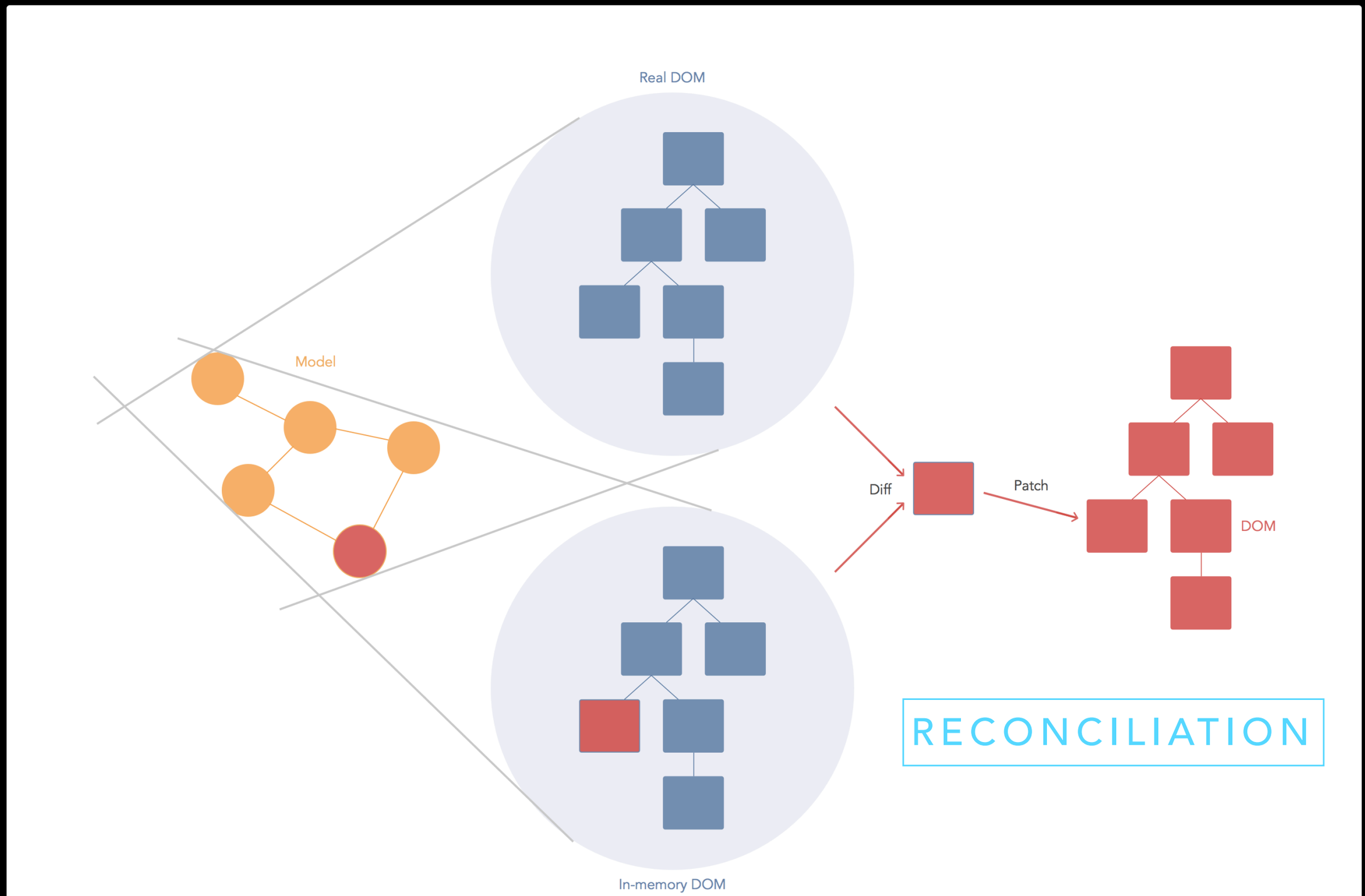



```
<html>
    <head>
        <title>
            My title
        </title>
    </head>
    <body>
        <h1>
            A heading
        </h1>
        <a href="http://www.google.com">
            Link text
        </a>
    </body>
</html>
```

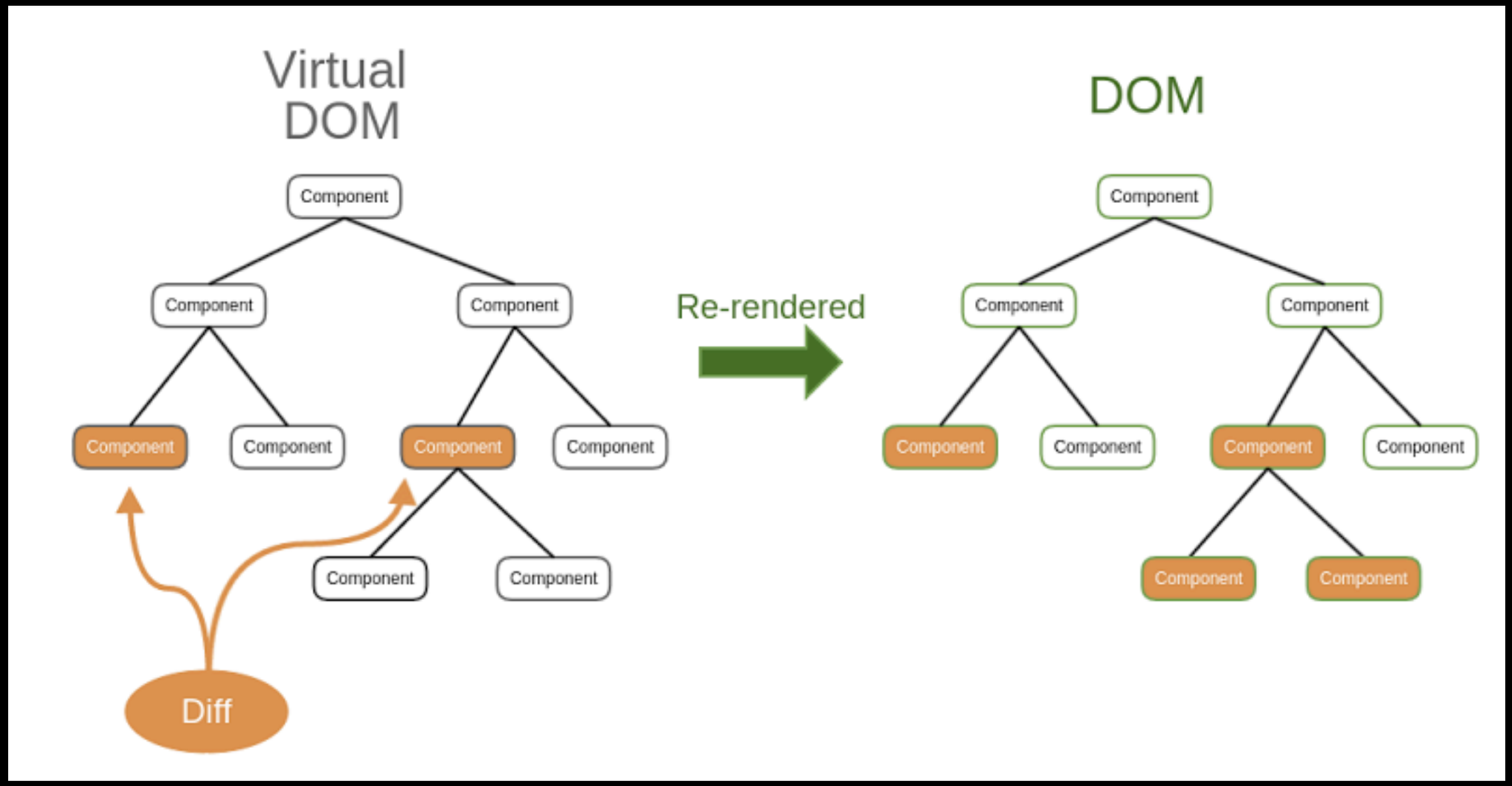
document.getElementById("TZA4S").removeChild(document.getElementById("Iga"))

OPEN UP CHROME TO DEMO

# REACT AND THE VIRTUAL DOM



# REACT AND THE VIRTUAL DOM




https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b

https://reactjs.org/docs/reconciliation.html

# GETTING SETUP

https://code.visualstudio.com/docs/nodejs/reactjs-tutorial

```
npm install -g create-react-app

create-react-app my-app

 cd my-app

 npm start
```

# MODIFYING INDEX.JS

- Don’t call the APP component lets start simple

```
import React from 'react';
import ReactDOM from 'react-dom';
```

```
ReactDOM.render(
    <h1>Hello, world!</h1>,
    document.getElementById('root')
  )
```

THIS IS STRANGE

Is it a string?

Is html?

IT IS CALLED JSX

Javascript XML

# JSX

LET, VAR AND CONST ARE REQUIRED IN REACT NATIVE

```
import React from 'react';
import ReactDOM from 'react-dom';

const element = <h1>Hello, world!</h1>;

ReactDOM.render(
    element,
    document.getElementById('root')
  )
```

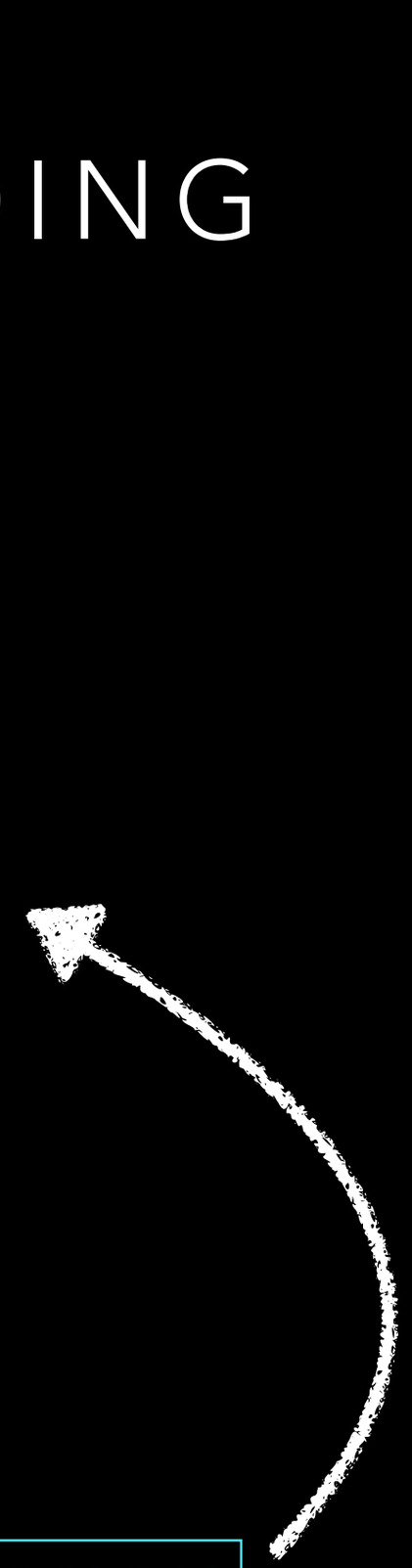
DEMO HOT SWAPS IN CHROME AND VISUAL STUDIO

# JSX EXPRESSION EMBEDDING

```
import React from 'react';
import ReactDOM from 'react-dom';

const name = 'DeShea Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
)
```



CAN BE ANY VALID JAVASCRIPT EXPRESSION

```
function buildWahoo(grit, grace, name) {
    let wahoo = {
    grit: 0,
    grace: 0,
    name: "Annoymous",
    readable: function(){
        return this.grit + " " + this.grace + " "+ this.name
    }

   }
    wahoo.grace = grace
    wahoo.grit = grit
    wahoo.name = name
    return wahoo

}

const element = (
  <h1>
    Hello, {buildWahoo(0.86, 0.9, "DeShea").readable()}!
  </h1>
)

ReactDOM.render(
  element,
  document.getElementById('root')
)
```

# JSX COMPILES TO REACT CREATE ELEMENT CALLS

```jsx
const element =
    <h1 className="greeting">
      Hello, world!
    </h1>
```

COMPILES TO THIS

```jsx
const element = React.createElement(
    'h1',
    {className: 'greeting'},
    'Hello, world!'
  )
```

```jsx
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
)
```

CREATES AND OBJECT SIMILAR TO THIS

```jsx
// Note: this structure is simplified
const element = {
  type: 'h1',
  props: {
    className: 'greeting',
    children: 'Hello, world!'
  }
};
```

# JSX IS AN EXPRESSION TOO

```
function getGreeting(user) {
    if (user) {
      return <h1>Hello, {buildWahoo(user)}!</h1>
    }
    return <h1>Hello, Anonymous.</h1>
  }
```

After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions:

# SPECIFYING ATTRIBUTES WITH JSX

```
let wahoo = {
    grit: 0,
    grace: 0,
    name: "Annoymous",
    avatarUrl: "http://www.w3schools.com/html/img_girl.jpg",
    readable: function(){
        return this.grit + " " + this.grace + " "+ this.name
    }
}

const tabElement = <div tabIndex="0"> </div>
const picture = <img src={wahoo.avatarUrl}></img>
```

# DISTINCTION BETWEEN ELEMENTS AND COMPONENT

REACT ELEMENTS ARE
WHAT GET RETURNED FROM COMPONENTS

# JSX ELEMENTS CAN CONTAIN CHILDREN

```
const element = (
    <div>
      <h1>Hello!</h1>
      <h2>Good to see you here.</h2>
    </div>
  )
```

# DEMONSTRATING THE VIRTUAL DOM

```
import React from 'react';
import ReactDOM from 'react-dom';


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);
```

# COMPONENTS AND PROPERTIES

```
function Clock() {
    let clockElement = <div>
                        <h1>This is a Clock</h1>
                        <h1>The Time is Now :
                       {new Date().toLocaleTimeString()}</h1>
                    </div>
    return  clockElement
  }
```

WE CAN ALSO MAKE THIS SELF CLOSING

```
<Clock/>
```

```
function tick() {
    const element = (
      <Clock></Clock>
    );
    ReactDOM.render(element, document.getElementById('root'))
  }

  setInterval(tick, 1000);
```

# WHAT IF WE HAD DIFFERENT TYPE OF CLOCKS

HOW COULD WE CUSTOMIZE THEM

THIS IS WHERE

PROPERTIES COME IN

# ASSIGNING PROPERTIES

FUNCTIONAL COMPONENTS MUST START WITH A CAPITAL

```
function Clock(props) {
    let clockElement = <div>
                        <h1>This is a { props.type} Clock</h1>
                        <h1>The Time is Now : {props.time}</h1>
                    </div>
    return  clockElement
}
```

# USING PROPERTIES

```
function tick() {
    const element = (
    <div>
        <Clock type={"English"} time ={new
                                Date().toLocaleTimeString()} />
        <Clock type={"Arabic"} time ={new
                            Date().toLocaleTimeString('ar-EG')} />
      </div>
    );
    ReactDOM.render(element, document.getElementById('root'))
  }

  setInterval(tick, 1000);
```

# PROPERTIES ARE READ ONLY

COMPONENTS MUST BE PURE FUNCTION

```
//Pure function does not modify its parameters
function sum(a, b) {
    return a + b;
  }

//Not a pure function because it modifies its parameters
function withdraw(account, amount) {
    account.total -= amount;
  }
```

THE MEANS THAT COMPONENTS CAN NOT MODIFY THEIR PROPERTIES

# CAN'T OVERRIDE PROPERTIES

```
function Clock(props) {
    props.type = "Override"
    let clockElement = <div>
                        <h1>This is a { props.type} Clock</h1>
                        <h1>The Time is Now : {props.time}</h1>
                    </div>
    return  clockElement
}
```

**TypeError: Cannot assign to read only property 'type' of object '#<Object>'**

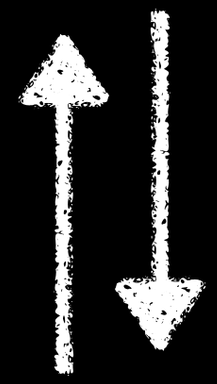
Clock
src/index.js:15

```
  12 |   }
  13 |
  14 | function Clock(props) {
> 15 |     props.type = "Override"
  16 |     let clockElement = <div>
  17 |                         <h1>This is a { props.type} Clock</h1>
  18 |                         <h1>The Time is Now : {props.time}</h1>
```

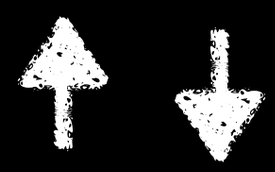
View compiled

# NEW SYNTAX FOR COMPONENTS

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



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



```jsx
<Welcome name="Sara" />
```

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

```jsx
const element = <Welcome name="Sara" />;
  ReactDOM.render(
    element,
    document.getElementById('root')
  )
```

```
  class Clock extends React.Component{
   render(){
        let clockElement = <div>
                          <h1>This is a { this.props.type} Clock</h1>
                          <h1>The Time is Now : {this.props.time}</h1>
                      </div>
        return  clockElement
    }
  }
```