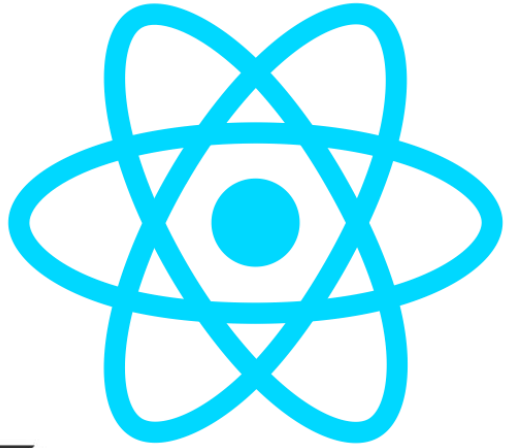
Cognixia

A Collabera LEARNING SOLUTIONS COMPANY



# ECMASCRIPT 6 / ECMASCRIPT 2015

# ECMASCRIPT 6 / ECMASCRIPT 2015

WHAT'S NEW?

- Variabletypes
- Template Strings
- Arrow functions
- Modules
- Classes

# IMMUTABLE VARIABLES

```
const MY_CONSTANT = 1;

MY_CONSTANT = 2;          //Error
```

# BLOCK-SCOPED VARIABLES

```
if(true) {

  let x = 1;

}

console.log(x); // undefined
```

```
for(let i = 0, l = list.length; i < l; i++) {
                    // do something with list[i]
}
console.log(i); // undefined
```

# Template Strings

```
//Multi Line String

const mlStrings = `In ES5,

this is not legal`;


//Interpolate variable bindings
const city = ’Bangalore’;
const time = ‘today’;

//In ES5
const city = 'Bangalore';
const time = 'today';
console.log('Hello '+ city +', How are you '+ time +'?');

//In ES6
 console.log(`Hello ${city}, How are you ${time}?`);
```

# ARROW FUNCTIONS

```
let books = [
        {title: 'X', price: 10},
        {title: 'Y', price: 15}
];
let titles = books.map( item => item.title );


// ES5 equivalent:
var titles = books.map(function(item) {
return item.title;
});
```

# ARROW FUNCTIONS

```
let book = {
  title:   'X',
  sellers:  ['A','B'],
  printSellers() {
    this.sellers.forEach((seller)  => {
        console.log(seller  +  '  sells  '  +this.title);
    });
  }
}
```

# MODULES

```
//  lib/math.js
export  function  sum(x,  y) {
  return  x  +  y;
}

export  var  pi  =  3.141593;


//  app.js
import  {  sum,  pi  }  from  "lib/math";
console.log('2PiVal = '  +  sum(pi,  pi));
```

# MODULES

```
// in lib/myfunc.js
export default function() {
        console.log('Echo function..');
}


// in app.js
import doSomething from 'lib/myfunc.js';
doSomething();
```

# CLASSES

```
class Vehicle {
        constructor(name) {
                this.name = name;
                this.kind = 'vehicle';
        }
        getName() {
                retun this.name;
        }
}


// Create an instance
let myVehicle = new Vehicle('Rocky');
```

# CLASSES

```
class Car extends Vehicle(){
        constructor(name) {
                super(name);
                this.kind = "Car";
        }
}


let myCar = new Car("Bumpy");

myCar.getName(); // Bumpy
myCar instanceof Car; // true
myCar instanceof Vehicle; // true
```

# SPREAD OPERATOR

```
let values = [1,2,4];
let updatedNumbers = [...values, 5];


let moreNumbers = [...values,5,6,..values];


// ES5 equivalent
let values = [1,2,4];
// Iterate, push, repeat ...
// Iterate, push, repeat ...
```

# SPREAD OPERATOR

```
let values = [1,2,4];


doSomething(...values);


function  doSomething(x,y,z) {
 // x = 1, y = 2, z = 4;
}
```

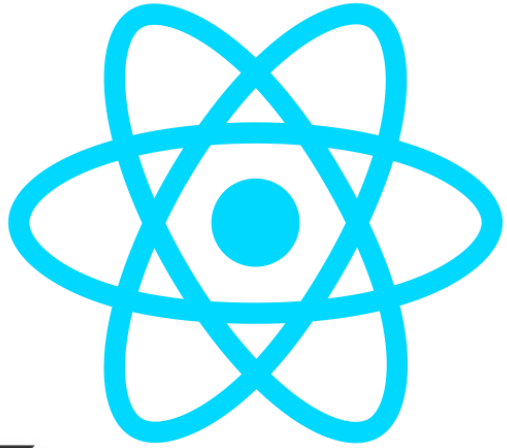
# Maps

The Map object holds key-value pairs. Any value (both objects and primitive values) may be used as either a key or a value.

```
// define a map object
let numMap = new Map([    [ 1, 'one' ],    [ 2, 'two' ],    [ 3, 'three' ]]);


// get & set over Map
numMap.get(1); // one


numMap.set('KEY_FOUR','four');


for (let value of map.values()) {
                    console.log(value); // one, two, three, four
}
```

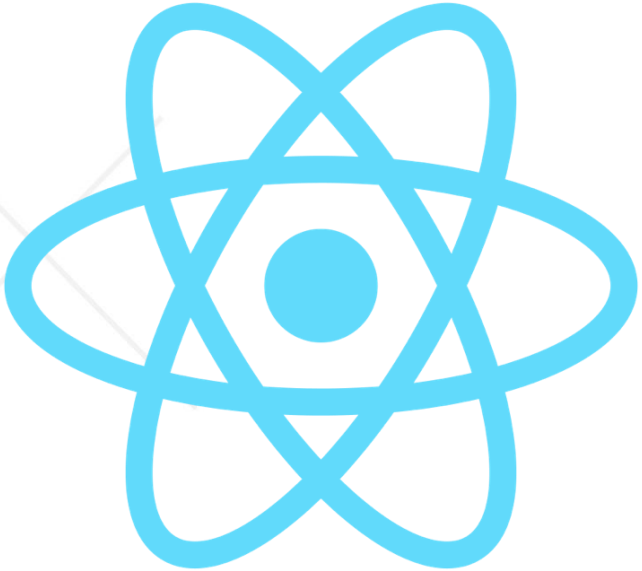
Cognixia

A Collabera LEARNING SOLUTIONS COMPANY



# React

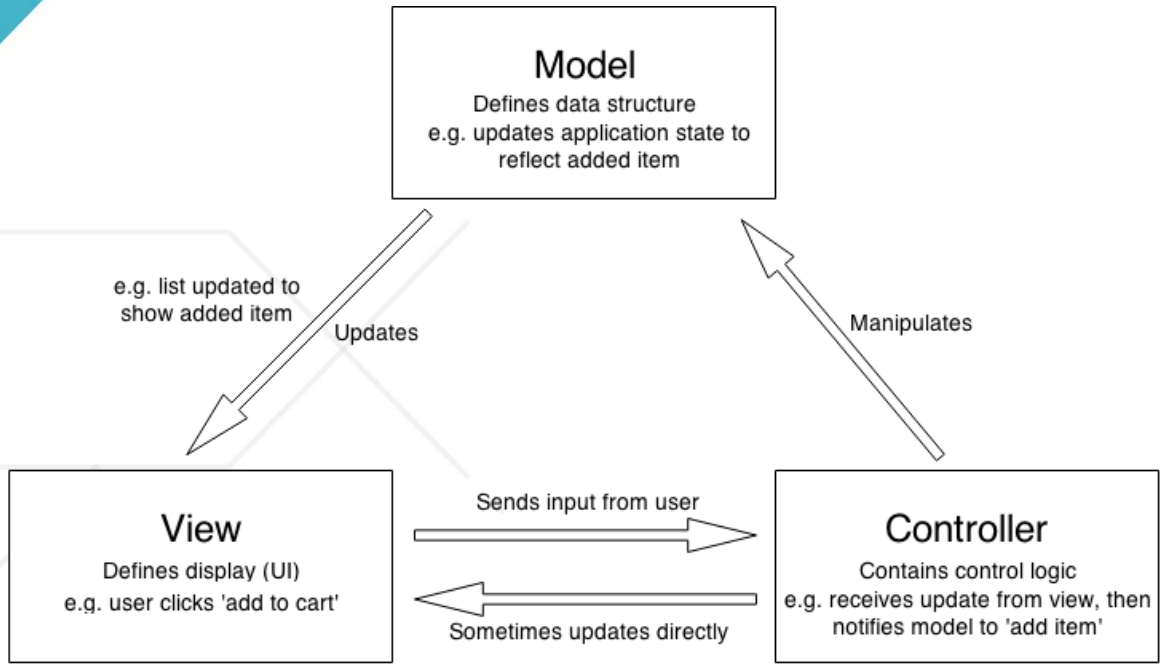
# React



# What is React?

- "A JavaScript library for building user interfaces." from official Documentation
- React is just the **View**
  - React is generally thought of as the view layer in an application. You might have used a library such as Handlebars or jQuery in the past.
  - Just like jQuery manipulates UI elements, or Handlebars templates are inserted onto the page, React components change what the user sees.

# ModelViewController



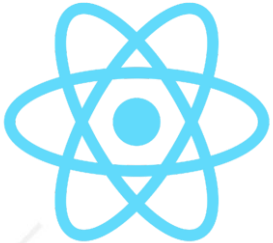
# ModelViewController

Thin views/templates Models and controllers that grows..

...and grows

Until most of your time is spent keeping them in sync

# We need a better model

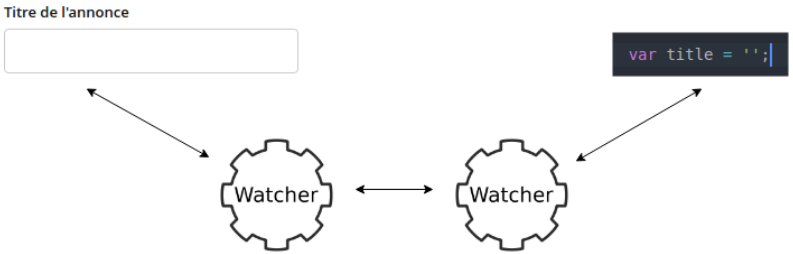


React is a JavaScript Library for building user interfaces.

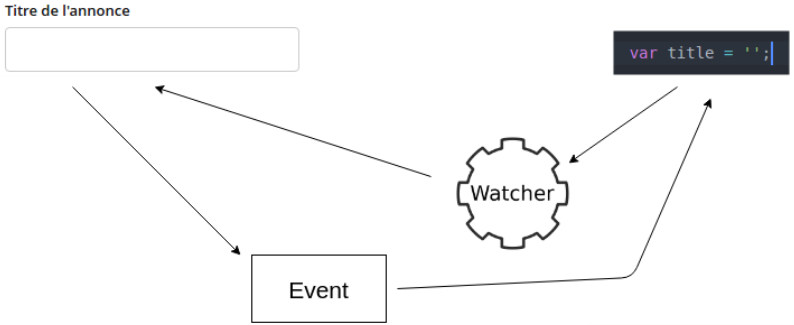
- Focus on the UI, not a Framework
- One-way reactive data flow (no two-way data binding)
- Virtual DOM

# Data Binding

2 ways data binding




1 way data binding

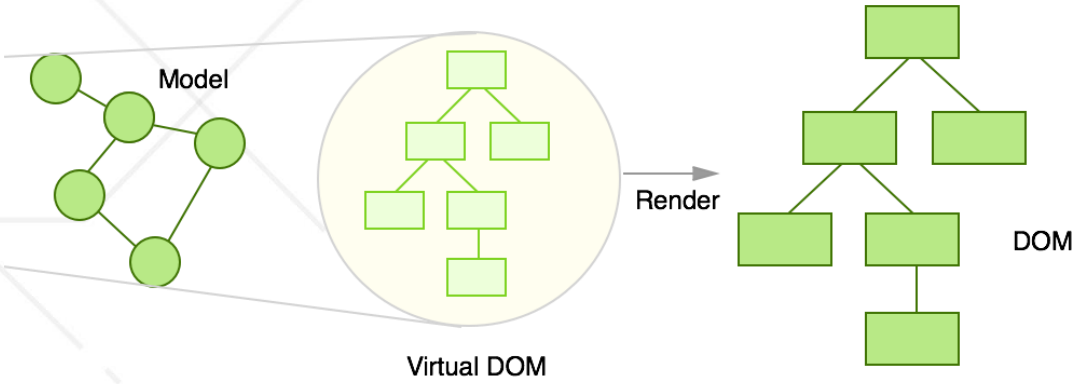


# Virtual DOM

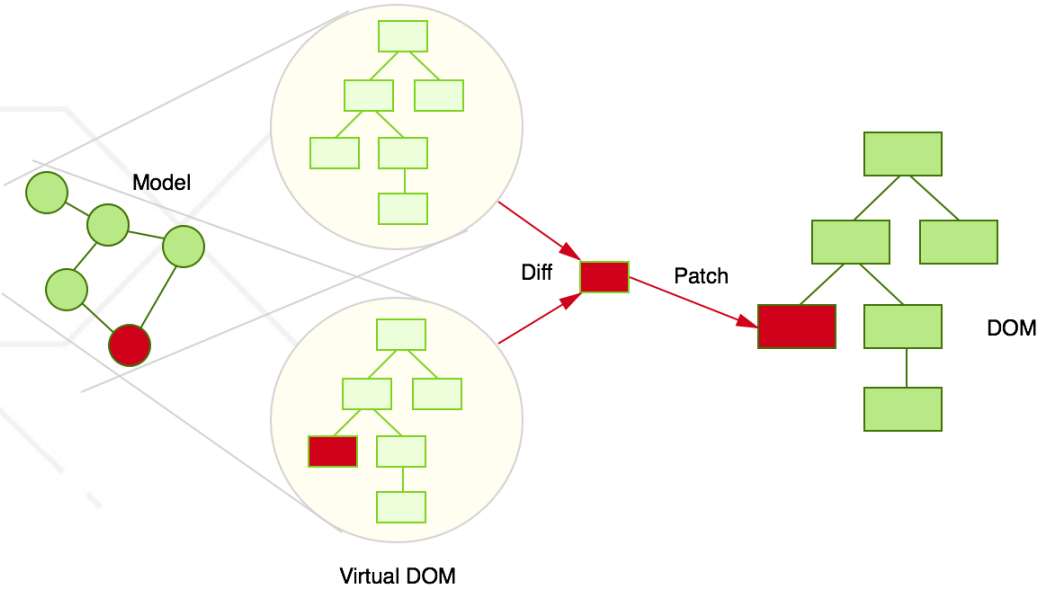
Keep track of state in DOM is hard.

The DOM API is slow.
(Try to re-render the whole DOM on every change)

# Virtual DOM



# Virtual DOM



# React Virtual Dom

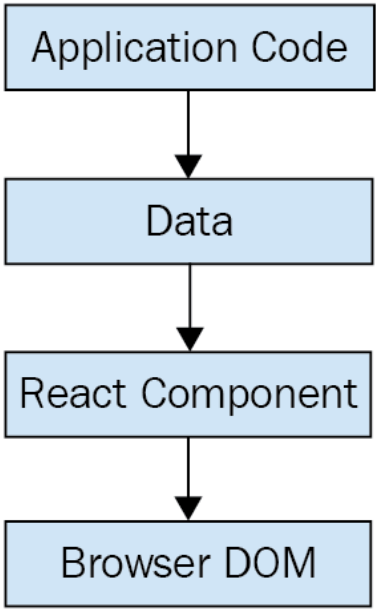
- React has something called the virtual DOM, which is used to keep a representation of the real DOM elements in memory.
- It does this so that each time we re-render a component, it can compare the new content to the content that's already displayed on the page.
- Based on the difference, the virtual DOM can execute the steps necessary to make the changes.

When you read about React, you'll often see words such as **diffing** and **patching**.

- **Diffing** means comparing old content with new content to figure out what's changed.
- **Patching** means executing the necessary DOM operations to render the new content.

# The React Way ?

- This is literally all there is to React—the core concept.
  - We have some application logic that generates some data.
  - We want to render this data to the UI, so we pass it to a React component, which handles the job of getting the HTML into the page.

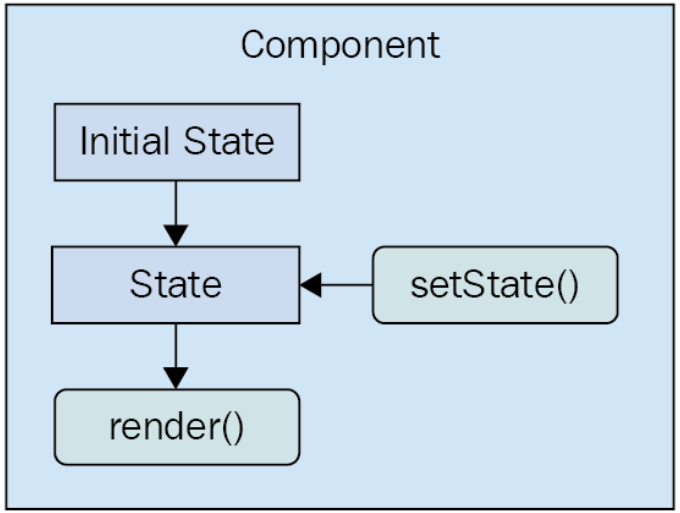


# What is JSX?

- JSX is React’s optional extension to the JavaScript syntax used for writing declarative XML-style syntax inside JavaScript code.
- For web projects, React’s JSX provides a set of XML tags that are similar to HTML.
- When transpiled (converted to plain JavaScript, so the browser or server can interpret the code), the XML is transformed into a function call to the React Library.

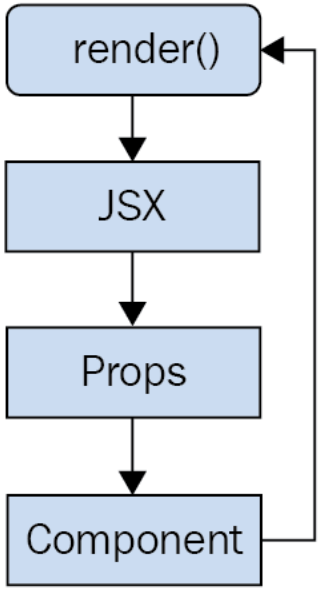
- The use of JSX is optional. However, embracing it has the following benefits:
  - XML is great for representing UIs in element trees with attributes.
  - It’s more concise and easier to visualize the structure of your application.
  - It’s plain JavaScript. It doesn't alter the language semantics.

# What is component state?




- React components declare the structure of UI elements using JSX. But, components need data if they are to be useful.
- State is the dynamic part of a React component. You can declare the initial state of a component, which changes over time.
- The state of a component is something that either the component itself can set, or other pieces of code, outside of the component.
- Imagine that you're rendering a component where a piece of its state is initialized to an empty array. Later on, this array is populated with data. This is called **a change in state**, and whenever you tell a React component to change its state, the component will automatically re-render itself.

# What are component properties?




- Properties are used to pass data into your React components. Instead of calling a method with new state as the argument, properties are passed only when the component is rendered. That is, you pass property values to JSX elements
- Properties are different than state because they don't change after the initial render of the component. If a property value has changed, and you want to re-render the component, then we have to re-render the JSX that was used to render it in the first place. The React internals take care of making sure this is done efficiently.

# Hello React!

- Create-react-app gives us a fully functioning React application in a single command. Let's create a hello app with react

```
$npx create-react-app helloworld

Success! Created helloworld at "/learn/react/helloworld"


We suggest that you begin by typing:

  cd helloworld
  npm start
```

# Hello React!

- npm start

```
Compiled successfully!

You can now view helloworld in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.0.0.34:3000/
```

- Let’s understand the code which got generated.


For more Information or set up an appointment kindly contact us today.