React.js

Lecture 1

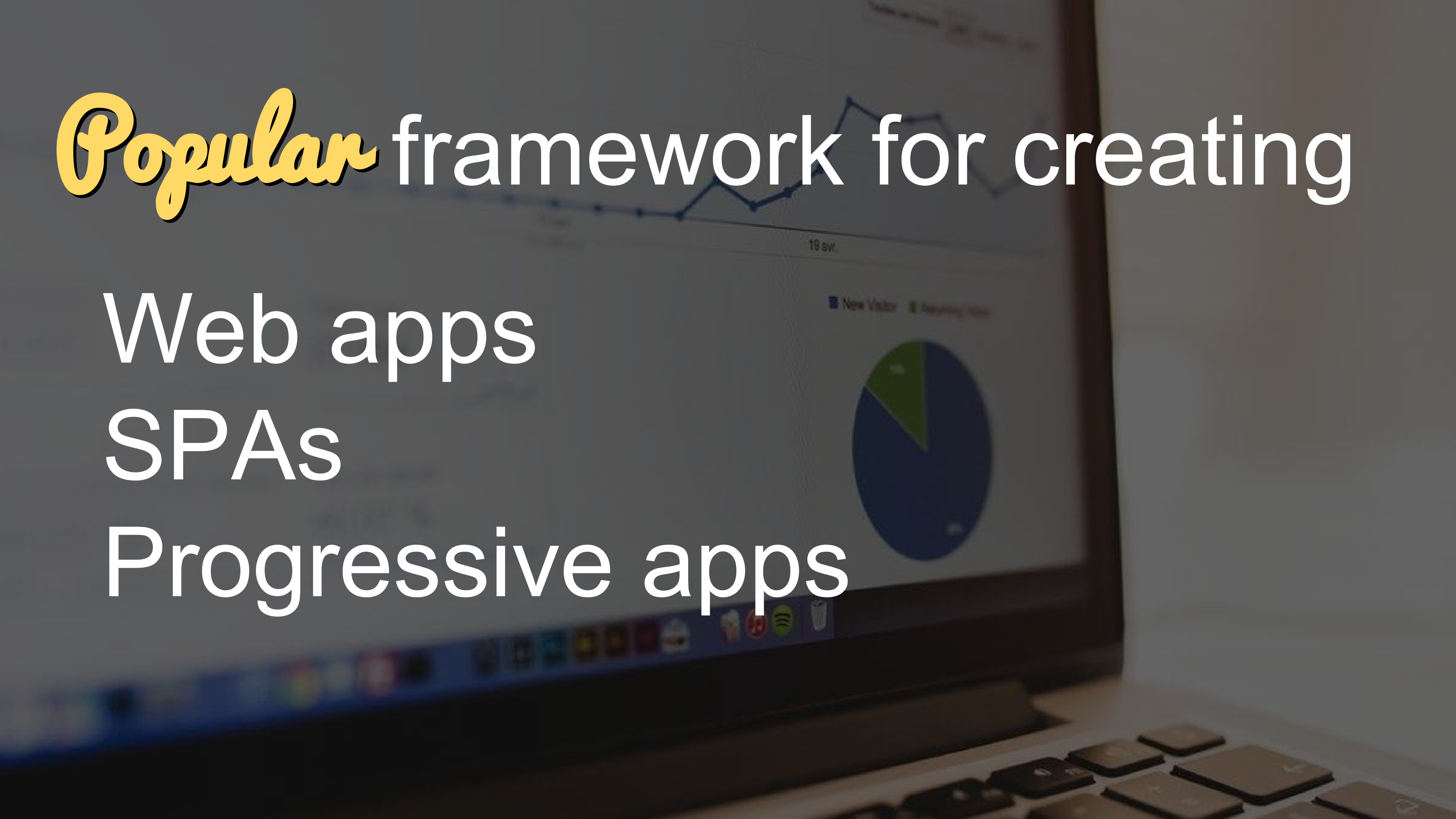
Picanteverde
(aka Alejandro Hernández)

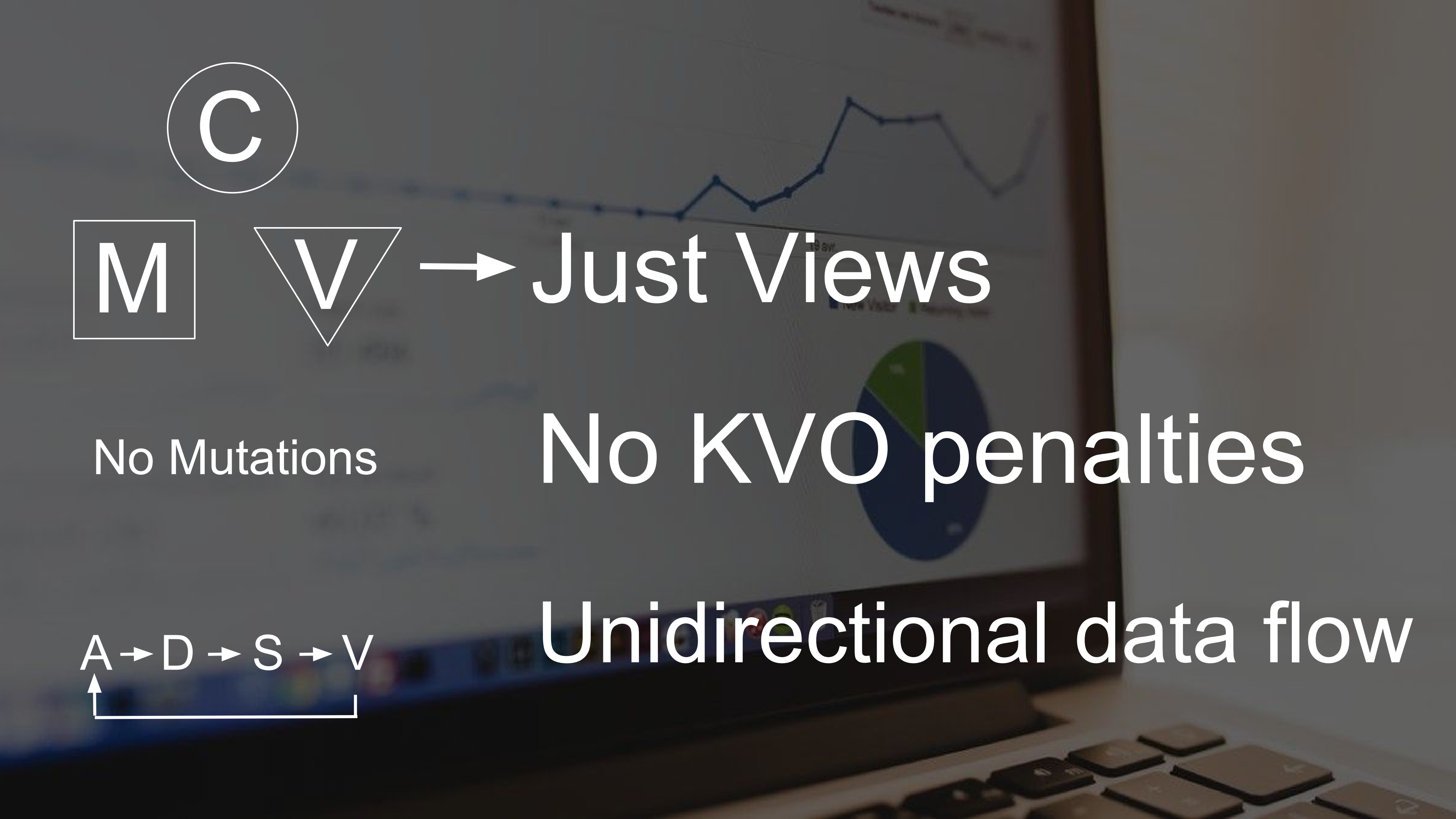
Popular framework for creating

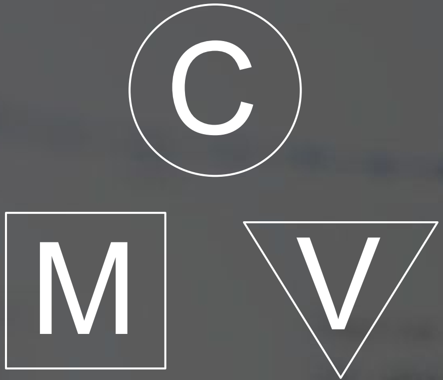
Web apps

SPAs

Progressive apps







Just Views

No Mutations

No KVO penalties

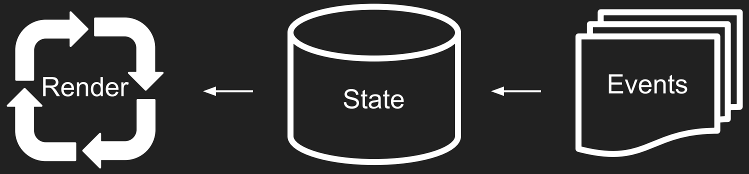
$$\begin{tikzcd} A \arrow[r] & D \arrow[r] & S \arrow[r] & V \arrow[lll, bend left] \end{tikzcd}$$

Unidirectional data flow

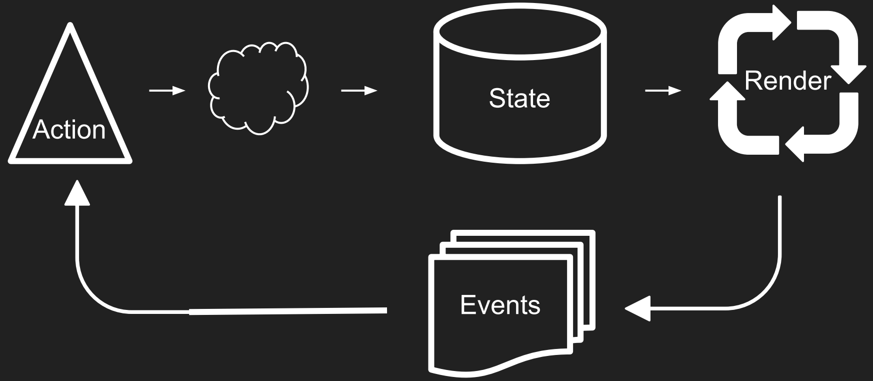
Re-rendering on every change makes things simple.

Video games are constantly rendering the current state on the render loop.

Events modify the state and is reflected on the next rendering



Re-rendering on every change makes things simple.



React re-renders only on every state change

Events fires Actions and those modify the State

DEMO

Re-rendering Problems

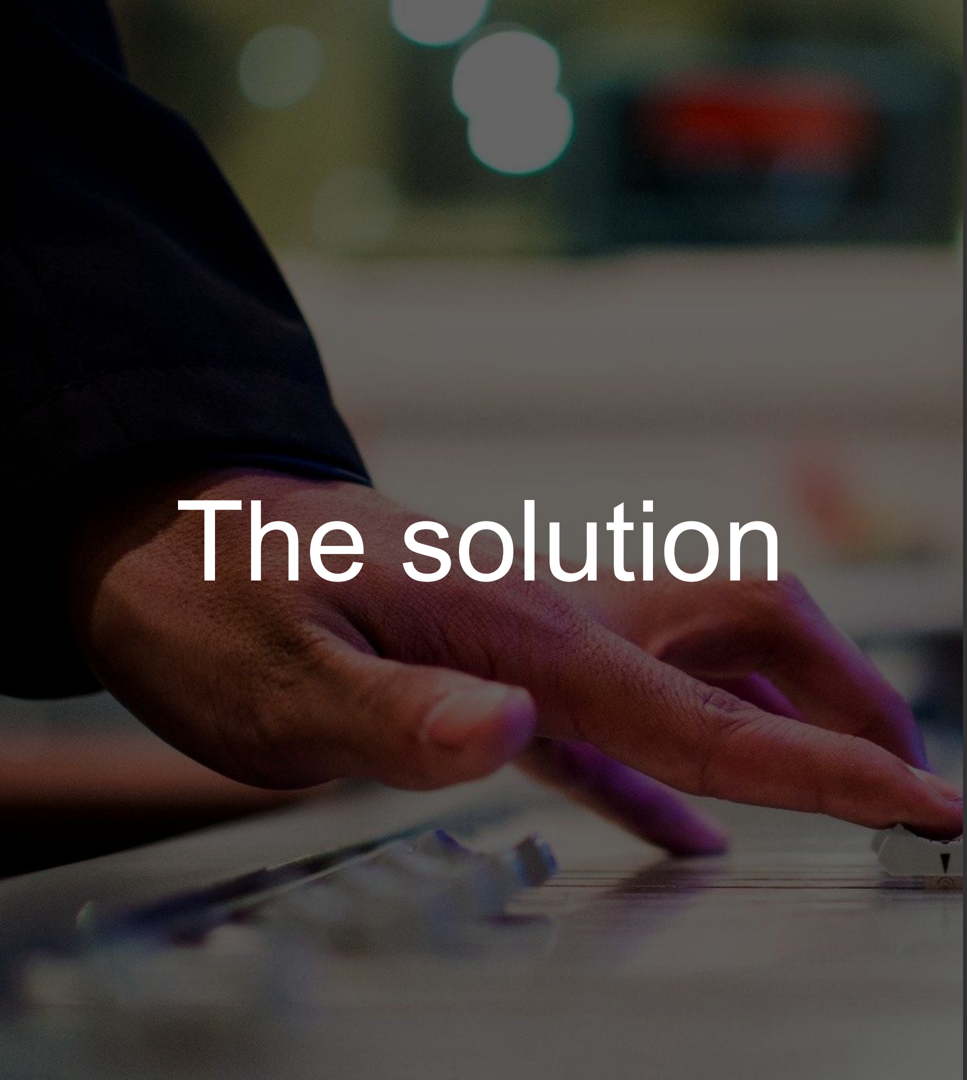
The way of the ‘90 (server side rendering)

Re-rendering to DOM is expensive (slow reflows & re paints)

Flickering (user might notice re rendering)

Forms and User Inputs (might clear user fields on every render)

Scroll (return to the top on every render)



The solution

Virtual DOM

Virtual Event System

(synthetic events)

Virtual DOM

Virtual DOM elements don’t render html, they render a json representation of html.

```
{
  element: ‘div’,
  attrs: {
   class: ‘title’
  },
  childs: [
   ‘Hello World!’
  ]
}
```

On Every Update

- Render a new Virtual DOM tree
- Calculates the diffs with the previous render of VDOM tree
- Computes the minimum set of DOM mutations and put them in a queue
- Batch executes all DOM operations

Virtual DOM

```
{ el: ‘div’, attrs: { class: [ ‘online’, ‘web’ ] }, childs: [ ‘Alex’ ] }

{ el: ‘div’, attrs: { class: [ ‘online’, ‘web’ ] }, childs: [ ‘Pete’ ] }

{ el: ‘div’, attrs: { class: [ ‘online’, ‘web’ ] }, childs: [ ‘Max’ ] }
```

Virtual DOM

```
<div class= “online web”>Alex</div>

<div class= “online web”>Pete</div>

<div class= “online web”>Max</div>
```

Pete went idle on mobile

Max went offline

Rachel went online on web

Virtual DOM

```
{ el: ‘div’, attrs: { class: [ ‘online’, ‘web’ ] }, childs: [ ‘Alex’ ] }

{ el: ‘div’, attrs: { class: [ ‘idle’, ‘mobile’ ] }, childs: [ ‘Pete’ ] }

{ el: ‘div’, attrs: { class: [ ‘online’, ‘web’ ] }, childs: [ ‘Rachel’ ] }
```

Virtual DOM

```
<div class= “online web”>Alex</div>

<div class= “online web”>Pete</div>

<div class= “online web”>Max</div>
```

```
el.childNodes[1].className = ‘idle mobile’;
el.childNodes[2].textContent = ‘Rachel’;
```

Virtual DOM

Reconciliation process

Push DOM mutation into the library

Also with Virtual DOM

Declarative (the lib is in charge of creating and maintaining the DOM)

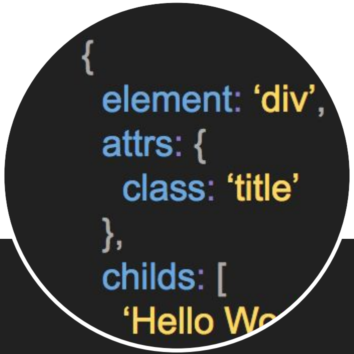
Synthetic Events (works consistently across browsers)

HTML5 events on IE8 (synthetic events)

It can run in node.js (render to string, SEO)

Server side rendering and client hooks (boot react on server rendered DOM)

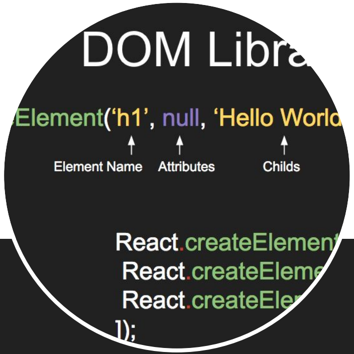
HOW?



```
{
  element: 'div',
  attrs: {
    class: 'title'
  },
  childs: [
    'Hello Wo
```

Virtual DOM

Generating representations of the object. Declarative



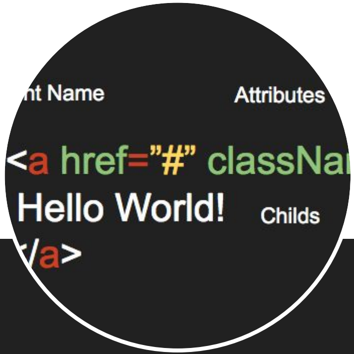
DOM Libra

```
Element('h1', null, 'Hello World
```

```
React.createElement
 React.createEleme
 React.createEle
]);
```

DOM Library

Interactions with the real DOM and libs for HTML, SVG, etc



```
<a href="#" classNa
Hello World!
</a>
```

JSX

Custom language to make our life easy when using the DOM Lib

BABEL

Transform

Convert JSX + ES2015 + ES2016 + ES2017 + custom plugins
To
ES5

DOM Library

```
ReactDOM.render(
  React.createElement(‘h1’, null, ‘Hello World!’),
  document.getElementById(‘content’)
);
```

DOM Library

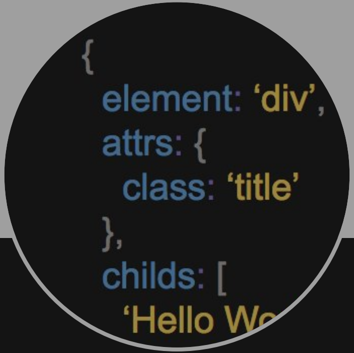
```
React.createElement('h1', null, 'Hello World!');
```

$$\begin{tikzcd}[column sep=small] \text{Element Name} & \text{Attributes} & \text{Childs} \end{tikzcd}$$

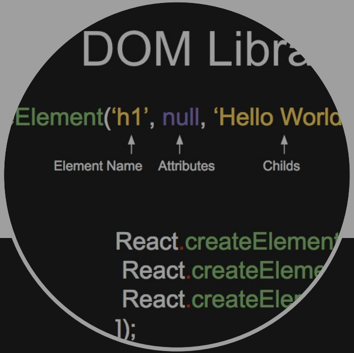
```
React.createElement(
  'a',
  {
    href: '#',
    className: 'link'
  },
  'Hello World!'
);
```

```
React.createElement('div', null, [
 React.createElement('h1', null, 'Hello'),
 React.createElement('h1', null, 'World!')
]);
```

That’s why...

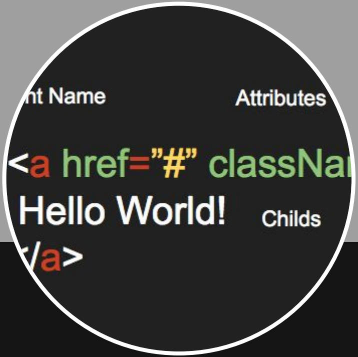


```
{
  element: ‘div’,
  attrs: {
    class: ‘title’
  },
  childs: [
    ‘Hello Wo
```



```
Element(‘h1’, null, ‘Hello World
```

```
React.createElement
 React.createEleme
 React.createEle
]);
```



```
<a href=”#” classNa
Hello World!
</a>
```

Virtual DOM

Generating representations of the object. Declarative

DOM Library

Interactions with the real DOM and libs for HTML, SVG, etc

JSX

Custom language to make our life easy when using the DOM Lib

Transform

Convert JSX + ES2015 + ES2016 + ES2017 + custom plugins
To
ES5

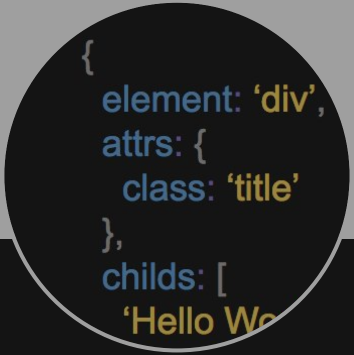
JSX

```
React.createElement(
  ‘a’,          Element Name
  {
    href: ‘#’,            Attributes
    className: ‘link’
  },
  ‘Hello World!’   Childs
);
```

```
Element Name            Attributes

  <a href=”#” className=”#”>
   Hello World!   Childs
  </a>
```

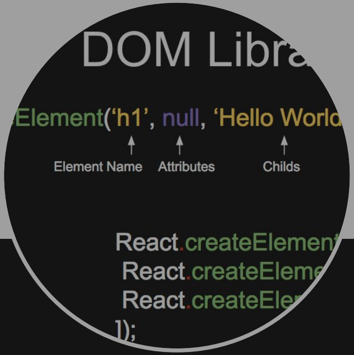
So we need...



```
{
  element: 'div',
  attrs: {
    class: 'title'
  },
  childs: [
    'Hello Wo
```

Virtual DOM

Generating representations of the object. Declarative



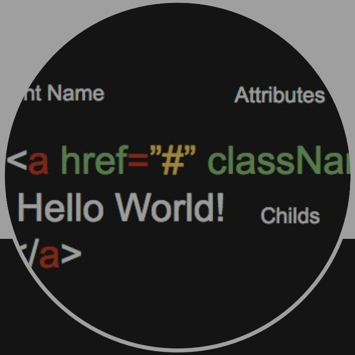
DOM Libra

```
Element('h1', null, 'Hello Worl
        ↑      ↑        ↑
Element Name Attributes Childs

        React.createElement
         React.createEleme
         React.createEle
        ]);
```

DOM Library

Interactions with the real DOM and libs for HTML, SVG, etc



```
nt Name           Attributes
<a href="#" classNa
Hello World!   Childs
/a>
```

JSX

Custom language to make our life easy when using the DOM Lib

Transform

Convert JSX + ES2015 + ES2016 + ES2017 + custom plugins
To
ES5

BABEL

Brings a whole new world +ES6

ES6

modules

proper tail calls

method definitions

Generators

promises

destructuring

fat arrow

symbol

for...of

Set

classes

proxy

let

Object.assign

const

Map

spread operator

template strings

WeakMap

https://kangax.github.io/compat-table/es6/
http://exploringjs.com/es6/

ES6 - modules

CommonJS

```
var mod1 = require(‘./mod1’);
module.exports = mod1;
```

- Compact syntax
- Designed for synchronous loading
- node.js

AMD

```
define([‘mod1’], function(mod1){
  return mod1;
});
```

- Complicated syntax
- Works without compilation
- Designed for asynchronous loading
- require.js

ES6 modules

```
import { mod1 } from ‘./mod1’;
export mod1;
```

- Compact syntax
- Support for synch and asynch
- Default and named exports

ES6 - modules

```
export var value1 = 31;

export var value2 = 42;

export { value2 as val2 };

export default function mod1(){};
```

```
import { value1 } from ‘./mod1’;

import { value2 as val2} from ‘./mod1’;

import mod1 from ‘./mod1’;

import {default as Mod1} from ‘./mod1’;

export mod1 from ‘./mod1’;
```

ES6 - let & const

```
var x = 3;
function func(randomize) {
  if (randomize) {
    var x = Math.random();
    return x;
  }
  return x;
}
func(false); // undefined
```

```
let x = 3;
function func(randomize) {
  if (randomize) {
    let x = Math.random();
    return x;
  }
  return x;
}
func(false); // 3
```

ES6 - Object.assign

```
var source = { b: 1};
var target = _.extend({a: 2} ,
source);
//target = { b:1, a:2 }
```

```
let source = { b: 1};
let target = Object.assign({a: 2},
source);
//target = { b:1, a:2 }
```

ES6 - method definitions

```
{
  method: function(arg){
    return arg + 7;
  }
}
```

```
{
  method(arg){
    return arg + 7;
  }
}
```

ES6 - destructuring

```
var obj = {
  first: 'Jane',
  last: 'Doe'
};
var first = obj.first;
var last = obj.last;
```

```
const obj = {
  first: 'Jane',
  last: 'Doe'
};
const { fisrt } = obj;
//first= 'Jane'
const { fisrt, last } = obj;
//first= 'Jane'; last= 'Doe'
const { first: f, last: l} = obj;
// f = 'Jane'; l = 'Doe'
```

ES6 - fat arrow functions

```
function(arg1, arg2){
  return arg1 + arg2;
};
```

```
(arg1, arg2) => {
  return arg1 + arg2;
}
```

ES6 - fat arrow functions

```
function(arg1){
  return arg1 + 7;
};
```

```
arg1 => {
  return arg1 + 7;
}
```

ES6 - fat arrow functions

```
function(arg1){
  return arg1 + 7;
};
```

```
arg1 => arg1 + 7
```

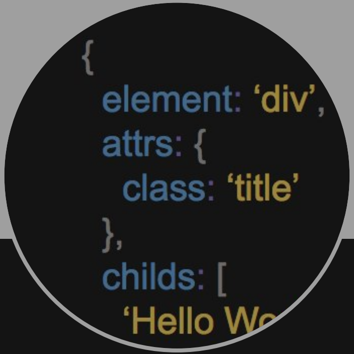
ES6 - fat arrow functions

```
function(arg1, arg2){
  return arg1 + arg2;
}.bind(this);

var that = this;
function(arg1, arg2){
  console.log(that);
  return arg1 + arg2;
};
```

```
(arg1, arg2) => {
  console.log(this);
  return arg1 + arg2;
}
```

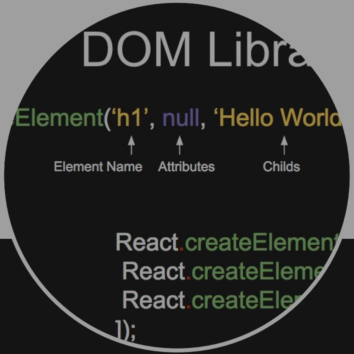
Back to ...



```
{
  element: 'div',
  attrs: {
    class: 'title'
  },
  childs: [
    'Hello Wo
```

Virtual DOM

Generating representations of the object. Declarative



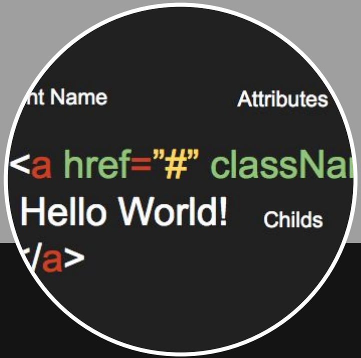
DOM Libra

```
Element('h1', null, 'Hello Worl
```

```
React.createElement
 React.createEleme
 React.createEle
]);
```

DOM Library

Interactions with the real DOM and libs for HTML, SVG, etc



```
<a href="#" classNa
Hello World!
</a>
```

JSX

Custom language to make our life easy when using the DOM Lib

Transform

Convert JSX + ES2015 + ES2016 + ES2017 + custom plugins
To
ES5

React createClass

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

let HelloWorld = React.createClass({
  render(){
    return (<h1>Hello world!</h1>);
  }
});

ReactDOM.render(<HelloWorld />, document.
getElementById('start'));
```

React compositions

```
let Item = React.createClass({
  render(){
    return (<div>This is an Item</div>);
  }
});
```

```
let List = React.createClass({
  render(){
    return (
      <div>
        This is a list with Items
        <div>
          <Item />
          <Item />
          <Item />
        </div>
      </div>
    );
  }
});
```

React properties

```
let List = React.createClass({
  render(){
    return (
      <div>
        This is a list with Items
        <div>
          <Item content="Item1"/>
          <Item content="Item2"/>
          <Item content="Item3"/>
        </div>
      </div>
    );
  }
});
```

```
let Item = React.createClass({
  render(){
    return (<div>{this.props.content}</div>);
  }
});
```

React JSX Syntax

```
<Item content={strValue}/> //ok
<Item content="tempFile{intVal}"/> // not ok
{['hello', <span>Wolrd</span>, '!!']}
{[1,2,3,4].map(num=><Item key={num} content={'Item' + num} />)}
```

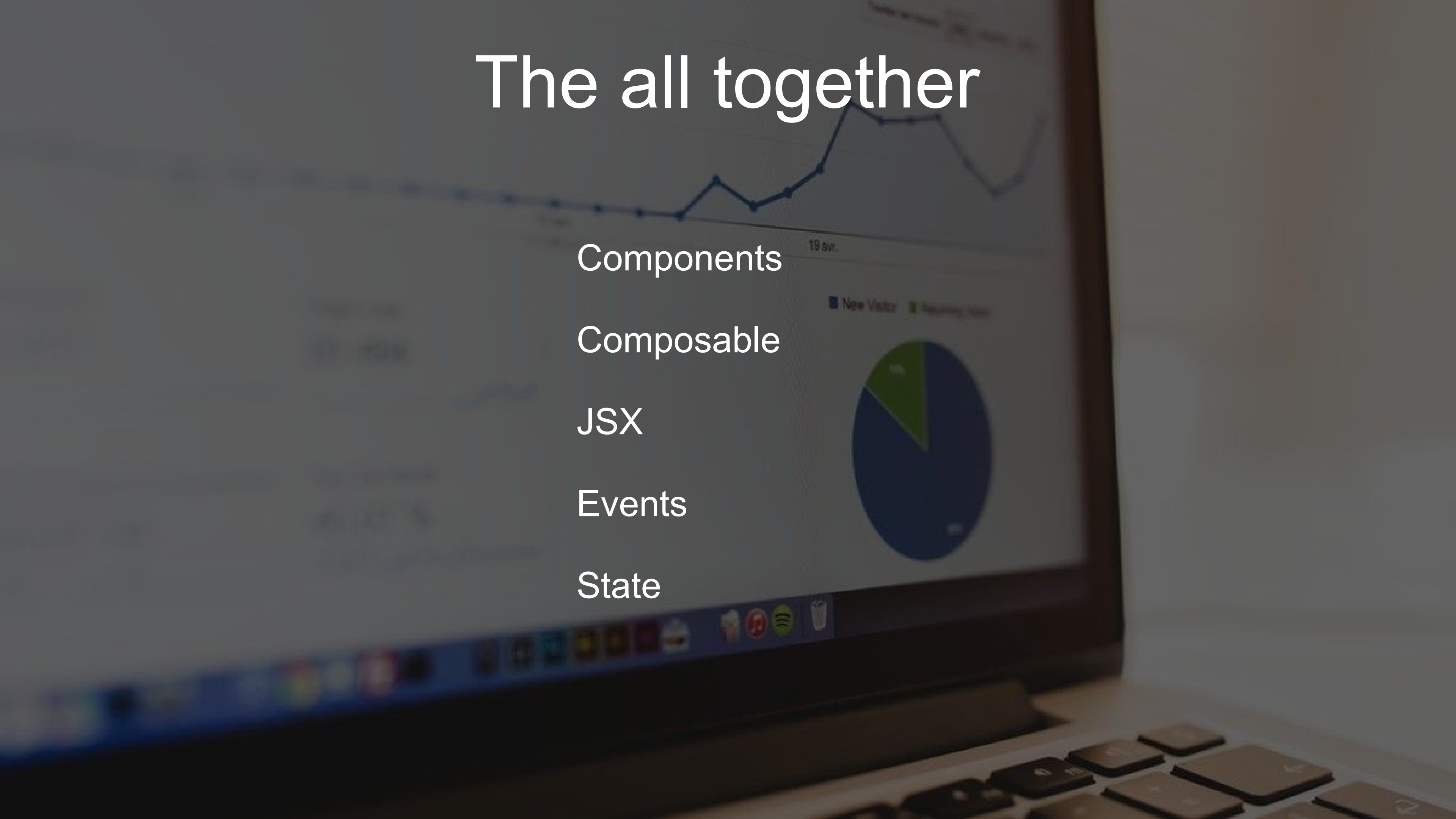
https://facebook.github.io/react/docs/jsx-gotchas.html

React Events

```
let Clickable = React.createClass({
  handleClick(e){
     console.log('Clicked!');
  },
  render(){
    return (
      <h1
        onClick={this.handleClick}
      >click me!</h1>);
  }
});
```

React State

```
let Clickable = React.createClass({
  getInitialState(e){
    return { count: 0 };
  },
  handleClick(e){
     this.setState({
       count: this.state.count + 1
     });
  },
  render(){
    return (<h1 onClick={this.handleClick}>
      You clicked {this.state.count} times!
    </h1>);
  }
});
```



The all together

Components

Composable

JSX

Events

State