Dependency Injection in React

IoC Containers to the Rescue

GitHub Repo: https://github.com/Xipooo/ReactDIDemo

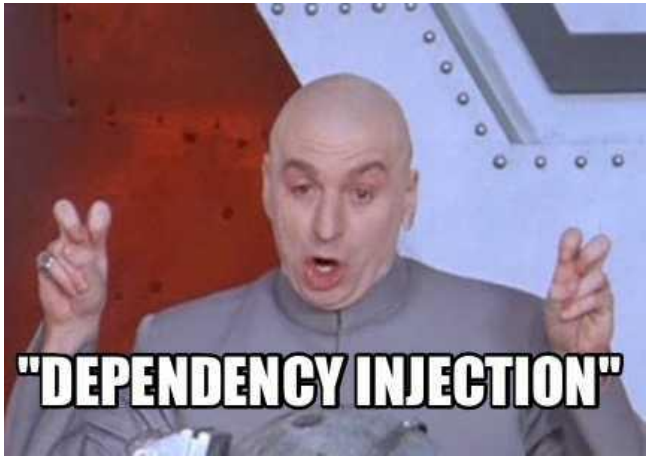
Who Am I?

- Steve Bishop
- Lead Enterprise Instructor @ Galvanize
- Over 20 years developing applications in JavaScript, Java, C#, VB, SQL, NoSQL
- Developed instructional programs ranging from beginner to advanced
- Over 9 Million YouTube views for my online courses - ProgrammingMadeEZ.com
- Former Golf Instructor
- Likes to argue (and listen)

@EzProgramming

@ProgrammingMadeEZ

What is Dependency Injection?



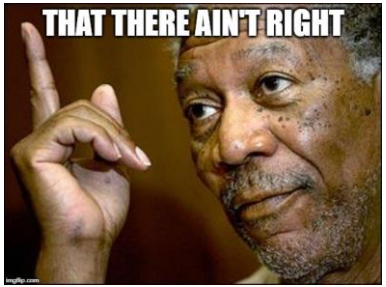
“A technique whereby one object supplies the dependencies of another object.” - Wikipedia

What is Dependency Injection?

Dependency Injection helps us adhere to the Single Responsibility Principle and Separation of Concerns by moving the instantiation of objects (or data) to another part of the application.

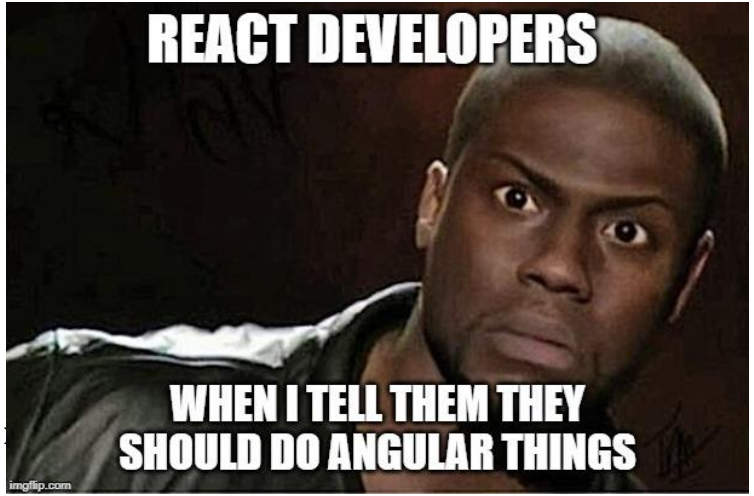
No process should be responsible for both the creation of its dependencies AND logic.

```
function add() {
        let a = 3;
        let b = 5;
        return a + b;
}
```




```
function add(a, b) {
        return a + b;
}
```

React Already Has DI



Using ***props*** is Dependency Injection.

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

Using ***context*** is Dependency Injection.

```
function Welcome() {
     const { message } = useContext(MessageContext);
     return <p>{ message }</p>
}
```

Inversion of Control Containers

- An IoC Container manages the creation and lifetime of objects.
- Classes are registered which maps an identifier to the class and determines the lifetime the objects created of that class.
- Objects are created and passed to the calling process through Constructor Injection, Property Injection, or Method Injection.
- Also, they're MAGIC!

Use The Force

CounterService.ts

```
@injectable()
export class CounterService {
  public count: number = 0;
  public increment(): void {
    this.count = this.count + 1;
  }
}
```

IoC.config.ts

```
let container = new Container();
container.bind(CounterService).to(CounterService).inSingletonScope();
export { container };
```

Counter.ts

```
class Counter {
  @resolve(CounterService)
  private _counterService: CounterService;
  onIncrement(){
    this._counterService.increment();
  }
}
```

LET’S CODE!!!

GitHub Repo:

https://github.com/Xipooo/ReactDIDemo