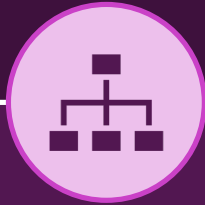


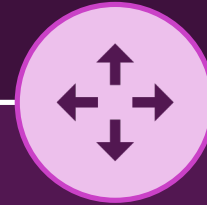
## Keep Your Control Structures Clean



Avoid Deep Nesting



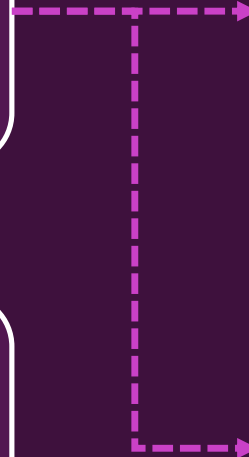
Prefer Positive Checks  
(*if isEmpty* vs *if isEmpty*)



Using Factory Functions &  
Polymorphism



Utilize Errors



## Use Guards & Fail Fast

```
if (email.includes('@')) {  
  // do stuff  
}
```



```
if (!email.includes('@')) {  
  return;  
}
```

Fail fast

Guard

// do stuff runs before main code runs

```
if (user.active) {  
  if (user.hasPurchases()) {  
    // do stuff  
  }  
}
```



```
if (!user.hasPurchases()) {  
  return;  
}
```

Fail fast

Guard

```
if (user.active) {  
  // do stuff  
}
```

## Embrace Errors & Error Handling

Throwing + handling errors can replace if statements and lead to more focused functions

Simple rule: If something is an error → Make it an error

```
if (!isEmail) {  
  return {code: 422, message: 'Invalid input'};  
}
```

```
if (!isEmail) {  
  const error = new Error('Invalid input');  
  error.code = 422;  
  throw error;  
}
```

Error Handling is  
“One Thing”!