JS Code Smell in HC APM Mobile

1. **Calculated Data Should be Immutable**

```
componentWillReceiveProps(nextProps) {
  if (nextProps.location.query) {
    this.state.list.query = {...this.state.list.query, ...nextProps.location.query}
    this.setState(this.state)
  }
}
```

Problem:
Here state is altered directy which violet the rule that state should be treated as immutable in React. Generally, objects should be immutable because immutable objects are simpler to construct, test and use, cause less side effects.
Solution:
Use Immutable.js to handle deep nested state objects.

2. **Avoid Magic String**

```
renderButtons() {
  const done = []
  done.push(<Button key='play' onClick={this.ev
  done.push(<Button key='delete' type="warn" on

  switch (this.state.status) {
    case 'init':
      return <Button key='record' type="primary
    case 'recording':
    case 'uploading':
    case 'playing':
      return <Button key='stop' type="default"
    case 'done':
      return done
  }
}
```

Problem:
Magic strings like these are spread through out the code which is difficult to maintain.
Solution:
Use object types or const and symbols

3. **Multi-Responsibility Class**

```
5   import Query from './Query'
6
7   class APMList extends Query {
8
```

Problem:
Here the class query is responsible for sending queries, pagination etc which is bad. Because it violates the single responsibility rule.
Solution:
Write loosely coupled code.
Use composition instead of inheritance for example.

4. **Network reqeuest in UI Component**

```
woPickedup(tab) {
  this.setState({tab, loading: true, pickedPage: 0})
  history.replace(route_urls.wx.repairList + '?tab=' + tab)
  rest.list(urls.woPickedup, {status : tab, page: 0})
  .then(res => {
    this.state.request.data = res.data
    this.state.hasMore = res.data.length == 10
    this.state.loading = false
    this.setState(this.state)
  })
}
```

UI component should not care about how request are sent out. Network requests are burried deep down in ui component which is difficult to reuse.
Solution:
Seperate out network request into a seperate request module. Use redux and redux saga to dispatch action.

5. **Avoid Unnecessary Render**

Problem:
A small change in the parent state will cause all its child to rerender itself which causes a lot of unnecessary updates.
Solution:
Implement function shouldComponentUpdate to avoid unnecessary updates.

6. **Use CSS in an Elegant Way**

```
<PullToRefresh
  onRefresh={
    (resolve, reject) => {
      this.refresh().then(resolve).catch(reject)
    }
  }
  loadingHeight={70}
  loaderLoadingIcon={
    <Svg classnames={'pulsesvg loadingsvg'} svg={pulse} />
  }
  loaderDefaultIcon={(progress) => {
    return (
      <Svg classnames={'pulsesvg'} svg={pulse} />
    )
  }}
>
```

Problem:
Class name confliction is difficult to avoid and handle.
Solution:
Use css module

7. **Avoid promise**

```
return rest.get(urls.assets + '/' + id).then(device => {

  dispatch({
    type: types.device.one.set,
    device
  })
```

Problem:
Promise and callback is not clear and causes nested code which is bad.
Solution:
Use async and await or generator instead.

8. **Hard code text**.

```
query: util.date.month,
itemContent: [
  { key: 'type', label: '警报类型', type: 'alertType' },
  { key: 'alertTimeStr', label: '报警时间' },
  { key: 'hospitalName', label: '所在医院' },
  { key: 'currentValue', label: '当前值' },
  { key: 'alertMsg', label: '报警摘要' }
]
```

Problem:
Currently all the text are hard coded in the code and does not support i18n which is difficult to maintain.
Solution:
Use i18n.

9. **Too Large Class**

```
412 ▼ function mapStateToProps(state) {
413     return {
414       preload: util.preload(state)
415     }
416   }
417
418   export default connect(mapStateToProps)(GeneralForm)
419
```

Problem:
We've got some super class like Form.js which may be difficult to understand, read and maintain in the future.
Solution:
Consider restructure and break it into smaller classes.

10. **Use default value instead of shortcut**

```
<div>
  <CellsTitle>{title ? title : '当前步骤'}</CellsTitle>
  <Form>
```

Problem:
Shortcut default value in function is unclear and difficult to maintain.
Solution:
Write it as js function default parameter value.