دست خط یک برنامه نویس

شاید تا به حال در مورد دست خط برنامه نویسان شنیده باشید مواردی همچونن نام گذاری ، فاصله گذاری مناسب، طول مناسب برای هر خط کد و...

امروزه این موارد برای کد زدن در حوزه متفاوت بصورت استاندارد در آمده و جاوا اسکریپت و لایبراری های آن هم

از این استاندارد ها بهره مند شده اند و سه استاندارد پر طرفدار آن به شرح زیر است :

1.Google JavaScript Style Guide

2.Airbnb JavaScript Style Guide

3.JavaScript Standard Style Guide

انتخاب بهترین برند:

شاید پیش خودتون بگین که گوگل به دلیل اینکه برند شناخته تری است بهتره از آن استفاده کنیم ولی موارد بیشتری هستند که باید به آن دقت کنید مثلا اگر برای ری اکت قصد استفاده دارید استاندارد گوگل به درد شما نمیخورد چون ری اکت را ساپورت نمی کند، به جدول زیر دقت کنید شاید کمکتون کنه:

Rule(Rule Name)	Google	AirBnB	Standard
Semicolons (semi)	Required	Required	No
Trailing Commas (comma-dangle)	Required	Required	Not Allowed
Template Strings (prefer-template)	No Stance	Prefered	No Stance
Space Before Function Parentheses (space-before-function-paren)	No Space	No Space	Space Required
Import Extensions (import/extensions)	Allowed	Not Allowed	Allowed
Object Curly Spacing (object-curly-spacing)	No Space Allowed	Space Required	Space Required
Console Statements (no-console)	No Stance	None	No Stance
Arrow Functions Return Assignment (no-return-assign)	No Stance	No	No
React Prop Ordering (react/sort-prop-types)	N/A	No Stance	No Stance
React Prop Validation (react/prop-types)	N/A	Required	Not Required
Object Property Shorthand (object-shorthand)	No Stance	Prefer	No Stance
Object Destructuring (prefer-destructuring)	No Stance	Prefer	No Stance

*در حال حاضر پراستفاده ترین استاندارد برای کار با کتابخانه ری اکت جی اس استاندارد ایربی ان بی است.

آشنایی با استاندارد ایر بی ان بی برای ری اکت:

قوانین پایه:

- برای هر کامپوننت یک فایل مجزا در نظر بگیرید (البته برای کامپوننت های stateless میتونید استثنا قائل بشین).
- همیشه از سینتکس JSX استفاده کنید.
- از React.createElement استفاده نکنید مگر اینکه در یک فایلی که سینتکس JSX ندارد.

کجا از کلاس کامپوننت و کجا از فانکشن کامپوننت استفاده کنیم:

اگر استیت داخلی دارید یا از رفرنس استفاده میکنید بهتره از کلاس کامپوننت استفاده کنید البته ترجیحا از class . React.createClass به جای extends React.Component

```
// bad
const Listing = React.createClass({
  // ...
  render() {
    return <div>{this.state.hello}</div>;
  }
});

// good
class Listing extends React.Component {
  // ...
  render() {
    return <div>{this.state.hello}</div>;
  }
}
```

و اگر استیت یا رفرنس ندارید ترجیحا از فانکشن کامپوننت استفاده کنید البته Arrow function ننویسید

```
// bad
class Listing extends React.Component {
  render() {
    return <div>{this.props.hello}</div>;
  }
}

// bad (relying on function name inference is discouraged)
const Listing = ({ hello }) => (
  <div>{hello}</div>
);

// good
function Listing({ hello }) {
  return <div>{hello}</div>;
}
import Footer from './Footer/Footer';
```

از Mixins استفاده نکنید

با عث پیچیدگی آسیب زننده میشه و توضیحش خودش یک مقاله میشه ، اگر با Mixins آشنا نیستین از اینجا بخوانید:

همچنین اگر علاقه دارید بدونید چرا زیان آور است این مطلب رو بخونید:

نام گذاری:

- از پسوند JSX استفاده کنید.
- از مدل نام گذاری پاسکال برای نامگذاری فایلها استفاده کنید برا مثال.ReservationCard.jsx
- کامپوننت های ری اکت را پاسکال کیس نامگذاری کنید و اگر اینستنس ازشون میگیرید کمل کیس نامگذاری کنید.

```
// bad
import reservationCard from './ReservationCard';

// good
import ReservationCard from './ReservationCard';

// bad
const ReservationItem = <ReservationCard />;

// good
const reservationItem = <ReservationCard />;
```

- نام فایل و کامپوننت یکی باشد برای مثال فایل ReservationCard.jsx معادل کامپوننت ReservationCard باشد و اگر از پوشه برای دسته بندی کامپوننت ها استفاده می کنید برای کامپوننت روت از نام index.js استفاده کنید وپوشه را به عنوان نام کامپوننت استفاده کنید.

```
// bad
import Footer from './Footer/Footer';

// bad
import Footer from './Footer/index';

// good
import Footer from './Footer';
```

- برای کامپوننت های Higher-order از نام گذاری مرکب استفاده کنید در حقیقت فقط برای DisplayName برای مثال اگر نام Higher-order ما ()withFoo میباشد باید displayName آن. withFoo(Bar) این کمک می کند زمانیکه خطا رخ می دهد ما راحتتر بفهمیم چه اتفاقی افتاده.

```
// bad
export default function withFoo(WrappedComponent) {
  return function WithFoo(props) {
    return <WrappedComponent {...props} foo />;
  }
}

// good
export default function withFoo(WrappedComponent) {
  function WithFoo(props) {
    return <WrappedComponent {...props} foo />;
  }

  const wrappedComponentName = WrappedComponent.displayName
    || WrappedComponent.name
    || 'Component';

  WithFoo.displayName = `withFoo(${wrappedComponentName})`;
  return WithFoo;
}
```

- برای نام گذاری Props ها دقت کنید از نام های رزرو شده استفاده نکنید.

```
// bad
<MyComponent style="fancy" />

// bad
<MyComponent className="fancy" />

// good
<MyComponent variant="fancy" />
```

Declaration

هنگام تعریف کامپوننت با displaName نامگذاری نکنید:

```
// bad
export default React.createClass({
  displayName: 'ReservationCard',
  // stuff goes here
});

// good
export default class ReservationCard extends React.Component {
}
```

تراز کردن کدهای JSX بر اساس شکل زیر باشد:

```
// bad
<Foo superLongParam="bar"
     anotherSuperLongParam="baz" />

// good
<Foo
  superLongParam="bar"
  anotherSuperLongParam="baz"
/>

// if props fit in one line then keep it on the same line
<Foo bar="bar" />

// children get indented normally
<Foo
  superLongParam="bar"
  anotherSuperLongParam="baz"
>
  <Quux />
</Foo>
```

کوتیشن:

در سینتکس JSX همیشه از دبل کوتیشن استفاده کنید و برای جاوا اسکریپت سینگل کت

```
// bad
<Foo bar='bar' />

// good
<Foo bar="bar" />
```

فاصله گذاری:

همیشه برای اتربیوتهای کامپوننت و بستن تگ یک اسپیس فاصله بذارید نه کمتر نه بیشتر

```
// bad
<Foo/>

// very bad
<Foo             />

// bad
<Foo
 />

// good
<Foo />
```

در کد JSX برای مقدار داخل کرلی بریسس فاصله نگذارید

```
// bad
<Foo bar={ baz } />

// good
<Foo bar={baz} />
```

Props

پراپ ها را کمل کیس نام گذاری کنید

```
// bad
<Foo
  UserName="hello"
  phone_number={12345678}
/>

// good
<Foo
  userName="hello"
  phoneNumber={12345678}
/>
```

اگر مقدار یک پراپ true هست لازم نیست مقدار را اضافه کنید:

```
// bad
<img src="hello.jpg" />

// good
<img src="hello.jpg" alt="Me waving hello" />

// good
<img src="hello.jpg" alt="" />

// good
<img src="hello.jpg" role="presentation" />

// bad
<img src="hello.jpg" alt="Picture of me waving
hello" />

// good
<img src="hello.jpg" alt="Me waving hello" />
```

از [ARIA Role](#)های معتبر و non-abstract استفاده کنید:

```
// bad - not an ARIA role
<div role="datepicker" />

// bad - abstract ARIA role
<div role="range" />

// good
<div role="button" />
```

در المنتها از accessKey استفاده نکنید ، دسترسی را پیچیده می کند.

```
// bad
<div accessKey="h" />

// good
<div />
```

به هیچ عنوان از array index برای مقدار پراپ key استفاده نکنید و از یک مقدار یکتا استفاده کنید:

```
// bad
 {todos.map((todo, index) =>
   <Todo
     {...todo}
     key={index}
   />
 )}

 // good
 {todos.map(todo => (
   <Todo
     {...todo}
     key={todo.id}
   />
 ))}
```

همیشه برای پراپ هایی که اجباری نیستند مقدار پیش فرض مشخص کنید:

```
// bad
 function SFC({ foo, bar, children }) {
   return <div>{foo}{bar}{children}</div>;
 }
 SFC.propTypes = {
   foo: PropTypes.number.isRequired,
   bar: PropTypes.string,
   children: PropTypes.node,
 };

 // good
 function SFC({ foo, bar, children }) {
   return <div>{foo}{bar}{children}</div>;
 }
 SFC.propTypes = {
   foo: PropTypes.number.isRequired,
   bar: PropTypes.string,
   children: PropTypes.node,
 };
 SFC.defaultProps = {
   bar: '',
   children: null,
 };
```

در صورت امکان در HOC ها از پاس دادن پراپ های غیر ضروری اجتناب کنید این کار از تولید بسیاری از باگها جلوگیری میکنه:

```
// good
  render() {
    const { irrelevantProp, ...relevantProps  } = this.props;
    return <WrappedComponent {...relevantProps} />
  }

  // bad
  render() {
    const { irrelevantProp, ...relevantProps  } = this.props;
    return <WrappedComponent {...this.props} />
  }
```

رفرنس ها:

همیشه برای رفرنسها از callback استفاده کنید

```
// bad
<Foo
  ref="myRef"
/>

// good
<Foo
  ref={(ref) => { this.myRef = ref; }}
/>
```

پرانتزها:

در return اگر المنتها بیشتر از یک خطه حتما از پرانتز استفاده کنید.

```
// bad
render() {
  return <MyComponent variant="long body" foo="bar">
           <MyChild />
         </MyComponent>;
}

// good
render() {
  return (
    <MyComponent variant="long body" foo="bar">
      <MyChild />
    </MyComponent>
  );
}

// good, when single line
render() {
  const body = <div>hello</div>;
  return <MyComponent>{body}</MyComponent>;
}
```

المنتها:

همیشه المنمتهایی که children ندارند بصورت self-close نوشته شود:

```
// bad
<Foo variant="stuff"></Foo>

// good
<Foo variant="stuff" />
```

اگر المنتهای شما تعدادی پراپ دارد که چند خطی می شود حتما المنت را در یک خط جدید ببندید:

```
// bad
<Foo
  bar="bar"
  baz="baz" />

// good
<Foo
  bar="bar"
  baz="baz"
/>
```

از arrow function برای ارسال متغیرها استفاده کنید:

```
function ItemList(props) {
  return (
    <ul>
      {props.items.map((item, index) => (
        <Item
          key={item.key}
          onClick={() => doSomethingWith(item.name, index)}
        />
      ))}
    </ul>
  );
}
```

event handlerها را در متد constructor بایند کنید:

```
// bad
class extends React.Component {
  onClickDiv() {
    // do stuff
  }

  render() {
    return <div onClick={this.onClickDiv.bind(this)} />;
  }
}

// good
class extends React.Component {
  constructor(props) {
    super(props);

    this.onClickDiv = this.onClickDiv.bind(this);
  }

  onClickDiv() {
    // do stuff
  }

  render() {
    return <div onClick={this.onClickDiv} />;
  }
}
```

در کامپوننت های ری اکت برای نامگذاری متدها از آندرلاین در شروع استفاده نکنید:

```
// bad
React.createClass({
  _onClickSubmit() {
    // do stuff
  },

  // other stuff
});

// good
class extends React.Component {
  onClickSubmit() {
    // do stuff
  }

  // other stuff
}
```

مطمئن شوید متد render کلمه کلیدی return را داشته باشد:

مرتب سازی:

مرتب سازی برای: class extends React.Component

1. optional static methods
2. constructor
3. getChildContext
4. componentWillMount
5. componentDidMount
6. componentWillReceiveProps
7. shouldComponentUpdate
8. componentWillUpdate
9. componentDidUpdate
10. componentWillUnmount
11. *clickHandlers / eventHandlers* : onClickSubmit() , onChangeDescription()
12. *getter methods for render* : getSelectReason() , getFooterContent()
13. *optional render methods* : renderNavigation() , renderProfilePicture()
14. render

نحوه تعریف propTypes, defaultProps, contextTypes و...

```
import React from 'react';
import PropTypes from 'prop-types';

const propTypes = {
  id: PropTypes.number.isRequired,
  url: PropTypes.string.isRequired,
  text: PropTypes.string,
};

const defaultProps = {
  text: 'Hello World',
};

class Link extends React.Component {
  static methodsAreOk() {
    return true;
  }

  render() {
    return <a href={this.props.url} data-
id={this.props.id}>{this.props.text}</a
>;
  }
}

Link.propTypes = propTypes;
Link.defaultProps = defaultProps;

export default Link;
```

مرتب سازی برای: React.createClass

1. displayName
2. propTypes
3. contextTypes
4. childContextTypes
5. mixins
6. statics
7. defaultProps

8. getDefaultProps
9. getInitialState
10. getChildContext
11. componentWillMount
12. componentDidMount
13. componentWillReceiveProps
14. shouldComponentUpdate
15. componentWillUpdate
16. componentDidUpdate
17. componentWillUnmount
18. *clickHandlers / eventHandlers* : onClickSubmit() , onChangeDescription()
19. *getter methods for render* : getSelectReason() , getFooterContent()
20. *optional render methods* : renderNavigation() , renderProfilePicture()
21. render

از isMounted استفاده نکنید رسما داره منسوخ میشه.

تقریبا مواردی که خود Airbnb گفته بود رو مطرح کردیم و در پایان شما میتونید پکیجهای مربوطه رو که با eslintپیاده شده رو روی کد بیس خودتون پیاده کنید.

با این کار بصورت اتوماتیک اگر هریک از موارد رو رعایت نکنید بهتون خطا یا اخطار میده

https://www.npmjs.com/package/eslint

https://www.npmjs.com/package/eslint-config-airbnb

https://www.npmjs.com/package/eslint-plugin-jsx-a11y

https://www.npmjs.com/package/eslint-plugin-import

https://www.npmjs.com/package/eslint-plugin-react

در روت پروژه خود یک فایل با نام eslintrc. بسازید و مقادریر زیر را اضافه کنید

```
{
 "extends": "airbnb",
 "env": {
  "browser": true
 }
}
```

با ارزوی موفقیت و پیشرفت روز افزون برای شما عزیزان

حسین اشرفی پور

Refrences:

React Design Patterns and Best Practices - Michele Bertoli

https://airbnb.io/