ES6基础

1. 新的变量声明方式 let / const

主要区别在于提供了块级作用域与不再具备变量提升

```
// let
let a = 0

// const
const b = 0
const c = []
```

使用let来声明一个值会被改变的变量，而使用const来声明一个值不会被改变的变量，也可以称之为常量。 当值为基础数据类型时，那么这里的值，就是指值本身。 而当值对应的为引用数据类型时，那么我这里说的值，则表示指向该对象的引用。 这里需要注意，正因为该值为一个引用，只需要保证引用不变就可以，我们仍然可以改变该引用所指向的对象。

2. 箭头函数的应用

```
// 当函数直接被return时，可以省略函数体的括号
const fn = (a, b)=> a + b
// 多行不能省略
const fn1 = ()=> {
  const a = 0
  const b = 1
  return a + b
}
// 箭头函数可以替换函数表达式，但是不能替换函数声明
```

关于箭头函数中的this

箭头函数没有this，如果使用了this，一定指向了外层的this；如果要使用函数内部的this，就一定不要使用箭头函数

3. 模板字符串

```
const a = 10
const b = 20
const string1 = `${a}+${b}=${a+b}`
// 模板字符串支持换行
```

4. 解析结构

一种全新的写法

```
// 首先有这么一个对象
const props = {
    className: 'tiger-button',
    loading: false,
    clicked: true,
    disabled: 'disabled'
}

// 当我们想要取得其中的2个值：loading与clicked时：
const { loading, clicked } = props;

// 给一个默认值，当props对象中找不到loading时，loading就等于该默认值
const { loading = false, clicked } = props;
```

数组也有属于自己的解析结构

```
// es6
const arr = [1, 2, 3];
const [a, b, c] = arr;
```

数组以序列号一一对应，这是一个有序的对应关系。 而对象根据属性名一一对应，这是一个无序的对应关系。根据这个特性，使用解析结构从对象中获取属性值更加具有可用性。

5. 函数默认参数

```
function add(x = 20, y = 30) {
    return x + y;
}

console.log(add());
```

6.展开运算符

在ES6中用…来表示展开运算符，它可以将数组方法或者对象进行展开

```
const arr1 = [1, 2, 3];
const arr2 = [...arr1, 10, 20, 30];
// 这样，arr2 就变成了[1, 2, 3, 10, 20, 30];

// 对对象也可以这么操作
const obj1 = {
  a: 1,
  b: 2,
```

```
  c: 3
}

const obj2 = {
  ...obj1,
  d: 4,
  e: 5,
  f: 6
}

// 结果类似于 const obj2 = Object.assign({}, obj1, {d: 4})
```

```
// 展开运算符还常常运用在解析结构之中，例如在Raect封装组件的时候常常不确定props到底还有多少数据会传进来，就会利用展开运算符来处理剩余的数据。

// 这种方式在react中十分常用
const props = {
  size: 1,
  src: 'xxxx',
  mode: 'si'
}

const { size, ...others } = props;

console.log(others)

// 然后再利用暂开运算符传递给下一个元素，再以后封装react组件时会大量使用到这种方式，正在学习react的同学一定要搞懂这种使用方式
<button {...others} size={size} />
```

```
展开运算符还用在函数的参数中，来表示函数的不定参。只有放在最后才能作为函数的不定参，否则会报错。

// 所有参数之和
const add = (a, b, ...more) => {
    return more.reduce((m, n) => m + n) + a + b
}

console.log(add(1, 23, 1, 2, 3, 4, 5)) // 39
```

7. 对象字面量与class

ES6针对对象字面量做了许多简化语法的处理。

当属性与值的变量同名时。

```
const name = 'Jane';
const age = 20

// es6
const person = {
  name,
  age
}

// es5
var person = {
  name: name,
  age: age
};
```

除了属性之外，对象字面量写法中的方法也可以有简写方式。

```
// es6
const person = {
  name,
  age,
  getName() { // 只要不使用箭头函数，this就还是我们熟悉的this
    return this.name
  }
}

// es5
var person = {
  name: name,
  age: age,
  getName: function getName() {
    return this.name;
  }
};
```

在对象字面量中可以使用中括号作为属性，表示属性也能是一个变量了。

```
const name = 'Jane';
const age = 20

const person = {
  [name]: true,
  [age]: true
}
```

在ant-design的源码实现中，就大量使用了这种方式来拼接当前元素的className，例如:

```
let alertCls = classNames(prefixCls, {
      [`${prefixCls}-${type}`]: true,
      [`${prefixCls}-close`]: !this.state.closing,
      [`${prefixCls}-with-description`]: !!description,
      [`${prefixCls}-no-icon`]: !showIcon,
      [`${prefixCls}-banner`]: !!banner,
 }, className);
```

class

ES6为创建对象提供了新的语法糖，这就是Class语法。

```
// ES5
// 构造函数
function Person(name, age) {
  this.name = name;
  this.age = age;
}

// 原型方法
Person.prototype.getName = function() {
  return this.name
}

// ES6
class Person {
  constructor(name, age) {  // 构造函数
    this.name = name;
    this.age = age;
  }

  getName() {  // 原型方法
    return this.name
  }
}
```

除此之外，还需要特别注意在实际使用中的几种写法方式的不同，在下面的例子注释中，说明了他们分别对应的ES5中的含义

```
class Person {
  constructor(name, age) {  // 构造函数
    this.name = name;
    this.age = age;
  }

  getName() {   // 这种写法表示将方法添加到原型中
    return this.name
  }
```

```
  static a = 20;  // 等同于 Person.a = 20

  c = 20;   // 表示在构造函数中添加属性 在构造函数中等同于 this.c = 20

// 箭头函数的写法表示在构造函数中添加方法，在构造函数中等同于this.getAge = function()
{}
  getAge = () => this.age
}
// 箭头函数需要注意的仍然是this的指向问题，因为箭头函数this指向不能被改变的特性，因此在
react组件中常常利用这个特性来在不同的组件进行传值会更加方便。
```

extends

相比ES5，ES6的继承就要简单很多

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  getName() {
    return this.name
  }
}

// Student类继承Person类
class Student extends Person {
  constructor(name, age, gender, classes) {
    super(name, age);
    this.gender = gender;
    this.classes = classes;
  }

  getGender() {
    return this.gender;
  }
}
```

只需要一个extends关键字，就可以实现继承了，不用像ES5那样去担心构造函数继承和原型继承，除此之外，我们还需要关注一个叫做super的方法。

在继承的构造函数中，我们必须如上面的例子那么调用一次super方法，它表示构造函数的继承，与ES5中利用call/apply继承构造函数是一样的功能。

```
// 构造函数中
// es6
super(name, age);
```

```
// es5
Person.call(this);
// super还可以直接调用父级的原型方法，super.getName
```

```
// 继承在react中有大量的使用场景，许多组件都利用继承来创建。

import React, { Component } from 'react';

class App extends Component {
  defaultProps = {}
  state = {}
  componentWillMount() {}
  componentDidMount() {}
  btnClick = e => {}
  render() {}
}
```

只要根据上面所学到的知识，明确的知道哪些属性方法是放在构造函数中，哪些属性方法是放到了原型中，那么在编写react组件的时候就要简单和清晰很多。