

变量：var，变量是存储信息的容器。

```
var a=function(){console.log(this)}; a()
```

函数：function,函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

方便复用

参数：在调用函数时，您可以向其传递值，这些值被称为参数。

return：终止函数的执行，并返回一个指定的值给函数调用者。

数组：[],使用单独的变量名来存储一系列的值。 插入数据，push,pop, shift ,unshift,splice, concat, map

事件：e.preventDefault();取消事件的默认动作。

html: div ul li form input 超文本语言（英语：HyperText Markup Language，简称：**HTML**）是一种用于创建网页的标记语言。

命名规范：

运算符：

JavaScript 算术运算符

y=5

+ 加 x=y+2 x=7

- 减 x=y-2 x=3

* 乘 x=y*2 x=10

/ 除 x=y/2 x=2.5

% 求余数 (保留整数) x=y%2 x=1

++ 累加 x=++y x=6

-- 递减 x=--y x=4

JavaScript 赋值运算符：

x =10

= x=y x=5

+= x+=y x=x+y x=15

-= x-=y x=x-y x=5

= x=y x=x*y x=50

/= x/=y x=x/y x=2

%= x%=y x=x%y x=0

1、小括号运算符：()

- 1、小括号运算符，用来控制表达式中的运算优先级 a + (b * c)
- 2、函数声明时参数表 function init(arg1,arg2){alert(arg1,arg2)}
- 3、作为函数或对象方法的调用运算符 init(1,2)

2、点运算符：.

- 1、表示算术中的小数点（浮点数），如 2.5
- 2、取对象属性、方法，如 [].push(2)

对象：{}, JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。

```
var x = {a:1,b:2}
```

```
var x = new array()
```

```
var x = object();
```

```
var oReq = new XMLHttpRequest();
```

```
oReq.open('get','comment2.json');
```

```
oReq.send();
```

```
oReq.onreadystatechange = function(){
```

```
    if(){
```

```
        oReq.text
```

```
    }
```

```
}
```

```
img.onload
```

eval,将字符串转成json

{键名:键值}

{key:value}

```
var ReactClass = {  
  createClass: function (spec) {  
  }  
};
```

ReactClass.createClass

```
var React = {
```

```
  createClass: ReactClass.createClass
```

```
};
```

React.createClass

var CommentBox = React.createClass({

```
  render: function() {
```

```
    return (
```

```
      <div className="commentBox">
```

```
        111111
```

```
      </div>
```

```
    );
```

```
  }
```

```
});
```

```
var argument = {
```

```
  render: CommentBoxRender
```

```
}
```

```
var CommentBoxRender = function() {
```

```
  return (
```

```
    <div className="commentBox">
```

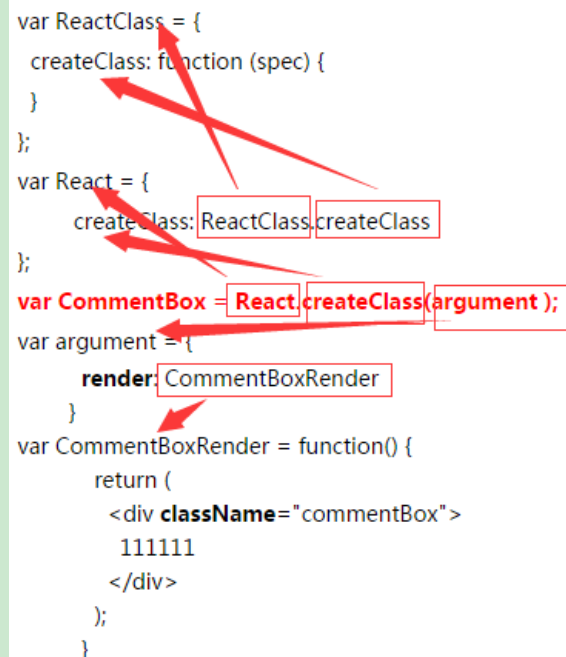
```
      111111
```

```
    </div>
```

```
  );
```

```
}
```

```
var ReactClass = {  
  createClass: function (spec) {  
  }  
};  
var React = {  
  createClass: ReactClass.createClass  
};  
var CommentBox = React.createClass(argument);  
var argument = {  
  render: CommentBoxRender  
}  
var CommentBoxRender = function() {  
  return (  
    <div className="commentBox">  
      111111  
    </div>  
  );  
}
```



注意：

- 1、react中html标签不用引号包围；
- 2、react中html标签class需要写成className

webpack-dev-server --inline 启动服务器

es6

class

```
class Comment extends React.Component{
  render(){
    return (
      <li className="commentAuthor">{this.props.author} : {this.props.children}</li>
    );
  }
}
```

import import xxx from './xxx.js'; es6标准定义了模块化，es5时代模块化框架seajs，requirejs

export export default xxx

XMLHttpRequest对象

```
var myFormData = new FormData(),
    oReq = new XMLHttpRequest(),
    srv = 'comment2.json'
;
myFormData.append('aa','1');
oReq.open("get", srv);
oReq.send(myFormData);
oReq.onreadystatechange = function (){
  if(oReq.readyState == 4){
    if(oReq.status == 200){
      var r = eval('(' + oReq.responseText + ')');
      this.setState({data:r});
    }else{alert('服务器错误' + oReq.status)}
  }
}.bind(this)
```

将我们写好的js，css静态文件打包并压缩 **webpack webpack.module1.pro.config.js -p**

正确打包方法：webpack --config webpack.module1.pro.config.js -p

this 函数运行时，自动生成的一个内部对象，只能在函数内部使用。this指的是，调用函数的那个对象：`function a()`

`{alert(this)};a();`全局调用，this代表全局对象

```
var React = {
  createClass: function (spec) {
    spec.render.call(window)
  }
};
var CommentBox = React.createClass({
  render: function() {
    console.log(this)
  }
});
```

类方法的this `confirm(e) {} confirm = (e) => {}`

bind，XMLHttpRequest对象回调方法内部this改变

getInitialState和constructor：是在组件生命周期中仅执行一次，设置组件的初始化状态。该方法是在React源码中做了封装的。es5使用getInitialState,es6使用原生的构造函数constructor代替getInitialState

```
constructor () {
  super();
  this.state = {
    data:[]
  }
}
```

```
}
```

componentDidMount:在页面dom元素已经渲染完成后调用一次

These methods are called when an instance of a component is being created and inserted into the DOM:

```
constructor()  
componentWillMount()  
render()  
componentDidMount()
```

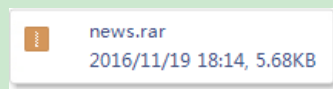
state和props: 通过state变化来重新绘制页面元素，父元素通过props值的变化来重新绘制子元素

就是将组件看成是一个状态机，一开始有一个初始状态，然后用户互动，导致状态变化，从而触发重新渲染 UI

getInitialState 方法用于定义初始状态，也就是一个对象，这个对象可以通过 this.state 属性读取。当用户点击组件，导致状态变化，this.setState 方法就修改状态值，每次修改以后，自动调用 this.render 方法，再次渲染组件。

由于 this.props 和 this.state 都用于描述组件的特性，可能会产生混淆。一个简单的区分方法是，this.props 表示那些一旦定义，就不再改变的特性，而 this.state 是会随着用户互动而产生变化的特性。

ref属性:真实dom节点



课上代码：

