22.异步 async

学习要点：

1.async 语法

本节课我们来开始学习 ES8 新增的异步 async 方法。

一. async 语法

1. async 也是处理异步的，它是对 Promise 的一种扩展，让异步更加方便；
2. 优势：async 是基于 Promise 的，虽然是异步操作，但看上去像同步；
3. 首先，我们先来看下 async 的基本语法；

```
//创建一个 async 函数，执行异步操作
//await 关键字：等待异步执行完毕后回调；
let as = async () => {
    let result = await p;
    console.log(result);
};

//执行 async 函数
as();
```

PS：上面语法，用 ES5 过渡一下，帮助理解，具体如下：

```
async function as() {}
let as = async function () {}
let obj = {async as() {}};              //也支持对象方式
```

4. 如果有三个异步需要列队输出，我们用 async 语法来处理一下；

```
//多个异步，按输出顺序加载，没有 then，清晰很多
async function as() {
    let r1 = await p1,
        r2 = await p2,
        r3 = await p3;

    console.log(r1);
    console.log(r2);
    console.log(r3);
}

as();
```

PS：await 关键字只能在 async 函数内部，否则不可识别；

PS：从上面的例子中，能感受到语义和清晰度都得到了很大提升，更像同步代码；

```
//批量异步列队，类似 Promise.all()
async function as() {
    let all = [await p1, await p2, await p3];
    console.log(all);
}

as();
```

5. async 函数如果设置了返回值，这个值是 Promise 对象。

```
//返回值是 Promise 对象
//相当于 Promise.resolve()
async function as() {
    return 'hello, async!';
}

as().then(value => {
    console.log(value);
});
```

PS：如果 return await p；这种，会导致提前输出 pending 状态，还是需要 then；

```
async function as() {
    return await p1;
}

console.log(as());      //得到的是 Promise 对象的 pending 状态
as().then(value => {    //这里还是需要 then
    console.log(value);
});
```