# 😀课外补充 (主要查看MDN开发手册)

## 1.渲染页面: 浏览器的工作原理 --9

MDN链接: https://developer.mozilla.org/zh-CN/docs/Web/Performance/How\_browsers\_work

简书链接: https://www.jianshu.com/p/e6252dc9be32

## 2.如何要存储你的变量信息 --10

MDN链接: <https://developer.mozilla.org/zh-CN/docs/Learn/JavaScript/First_steps/Variables>

## 3.表达式与运算符 --14

# 2022.3.31 第1节 欢迎

## 1~5 介绍下开始课程前的准备工作

略过

# 2022.3.31 第2节:脚本语言基础

## 6.部分介绍

1. john老师简单介绍课程

## 7.世界你好！

### 7.1 ctrl+alt+j 打开控制台

略过

### 7.2 控制台输入简单的代码

略过

## 8.JavaScript简介

### ⭐8.1 什么是JavaScript？

1. 我们可以以许多不同的方式定义 JavaScript，但是我的定义是它是一种高级面向对象的多范式编程语言

a) "多范式" 是多种编程方法的意思,有哪些编程方法呢？

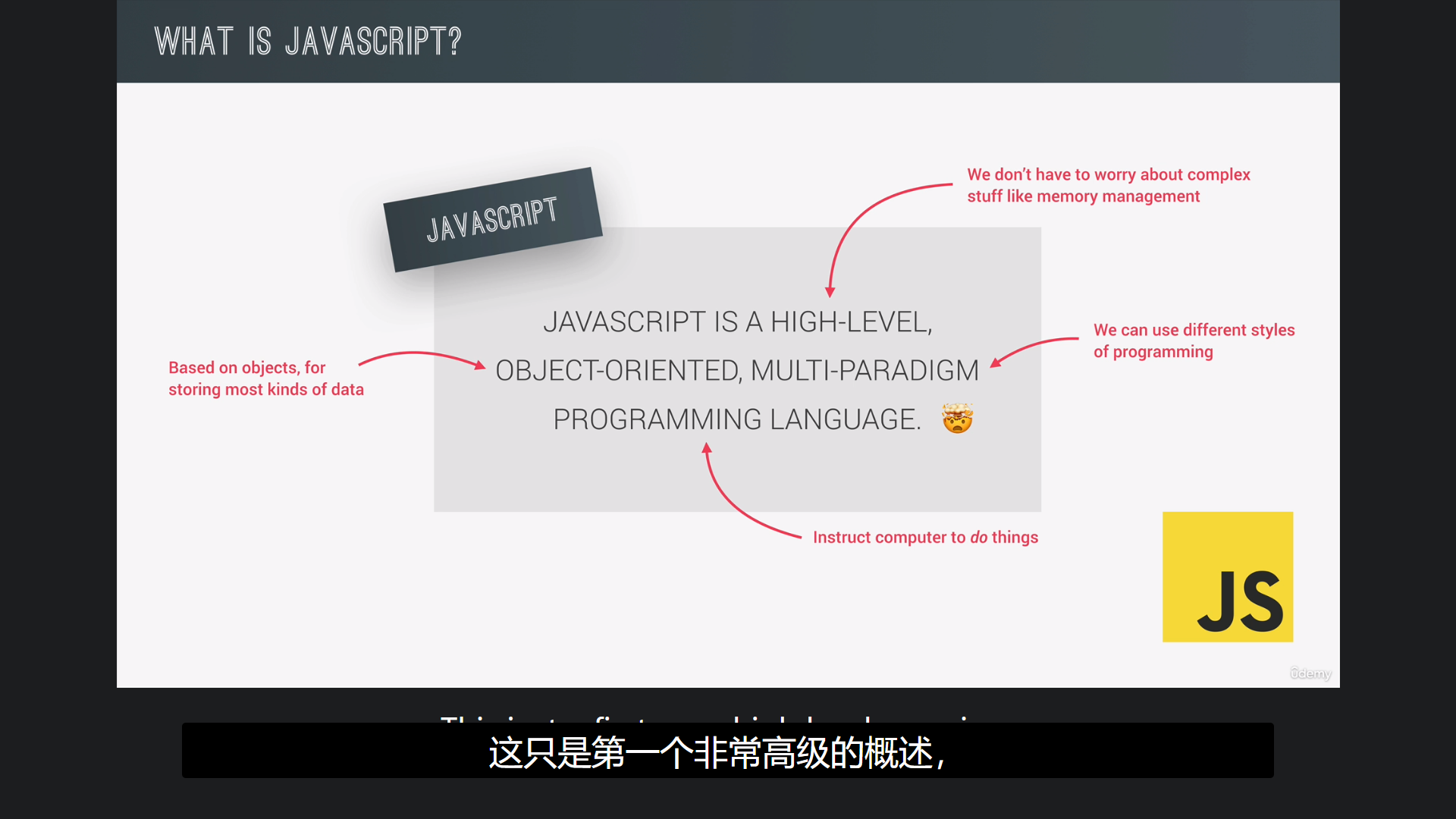
b) 目前说来,有面向过程,面向对象,泛型,函数式四种程序设计方法。”

2. 以编程语言本身开始,编程语言基本上只是一个工具,允许我们编写指示计算机的代码来指示计算机做某事,这也是我们使用JavaScript的主要目标

3. JavaScript是一种高级语言,这意味着我们不必考虑大量复杂的东西,例如在运行或程序时管理计算机内存,所以我们在JavaScript中,我们不想担心所有这些小的细节都有很多所谓的抽象,这使得语言更容易写入和学习

4. JavaScript是面向对象的,这意味着语言主要基于用于存储大多数数据的对象的概念,我们将在整个课程中了解面向对象的编程

5. JavaScript也是一种多范式语言,这意味着它是如此灵活和多样化,我们可以使用各种不同的编程风格,例如命令式和声明式的编程,这些不同的样式只是构造我们代码的不同方式



指导计算机做事

我们可以使用不同的编程风格

我们不必担心像内存管理这样复杂的事情

基于对象，用于存储大多数类型的数据

JAVASCRIPT是一种高级语言

面向对象的多范式编程语言

什么是JAVASCRIPT？

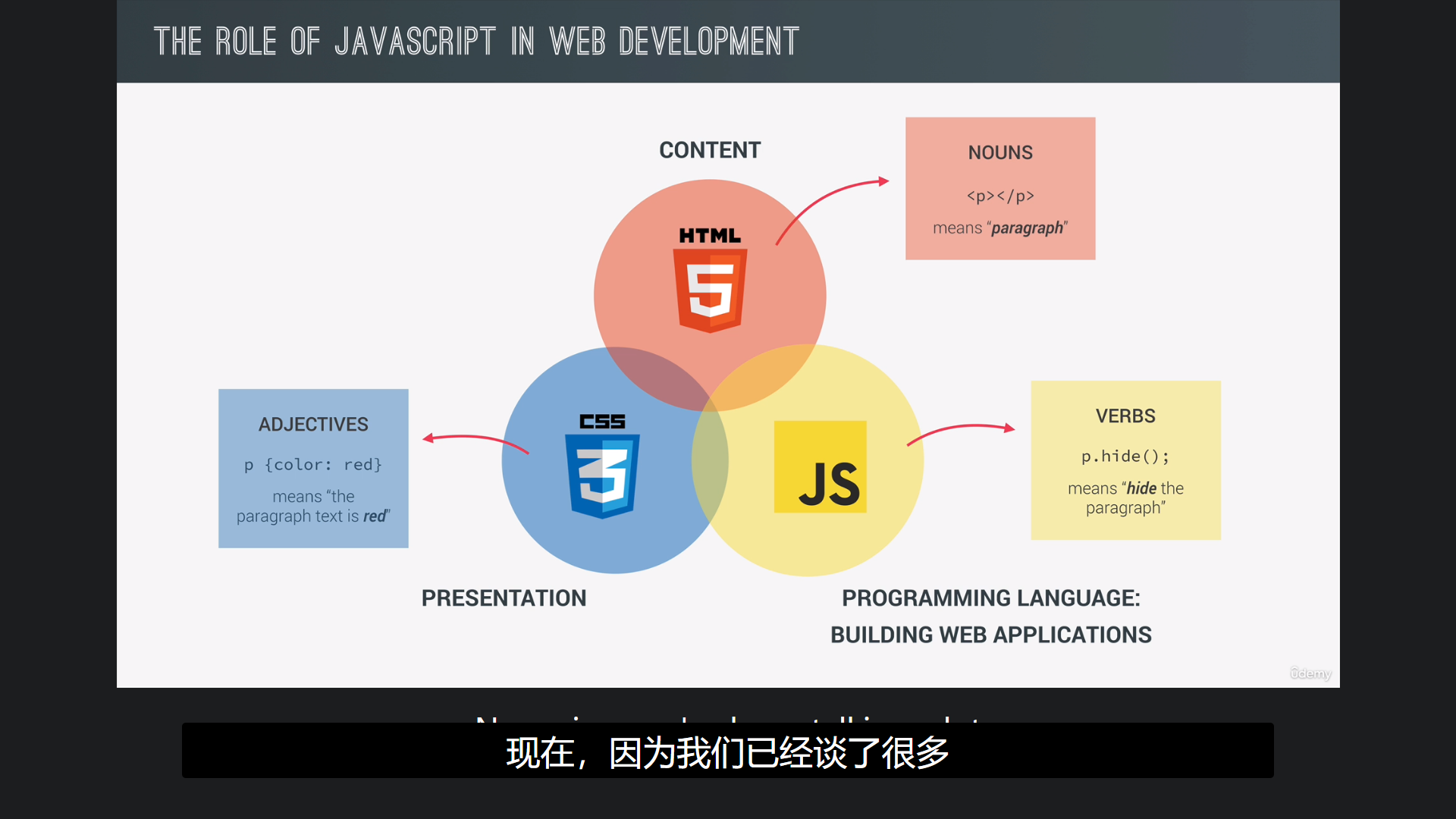
### ⭐8.2 JavaScript的作用是什么？

1. Web 的三大核心技术：HTML、CSS，当然还有 JavaScript。因此，这三种技术共同创造了美观、互动和动态的网站或 Web 应用程序。

2. 现在 HTML 负责页面的内容。因此，您在网页上看到的文本、图像、按钮和所有内容始终是用 HTML 编写的。

3. 然后 CSS 负责该内容的呈现。所以基本上是为了样式和在网页上布置元素。

4. 最后，JavaScript 是互联网的真正编程语言。它允许开发人员为任何网页添加动态和交互效果。我们还使用它来操作内容或 CSS，从远程服务器加载数据并真正在浏览器中构建整个应用程序，然后我们将其称为 Web 应用程序。



现在，我们还可以使用名词、形容词和动词的类比来使这种角色分离更容易理解。

1. 个类比中，HTML 代表名词。说P元素是一个段落。所以段落是这里的名词。

2. CSS 就是形容词，因为它描述了名词。就像这段 CSS 说段落文本是红色的一样。所以在这里，红色是描述名词的形容词

3. 最后，JavaScript 当然是动词，就像说隐藏段落一样。所以在这里我们正在做某事，所以我们有一个动词

内容

动词

名词

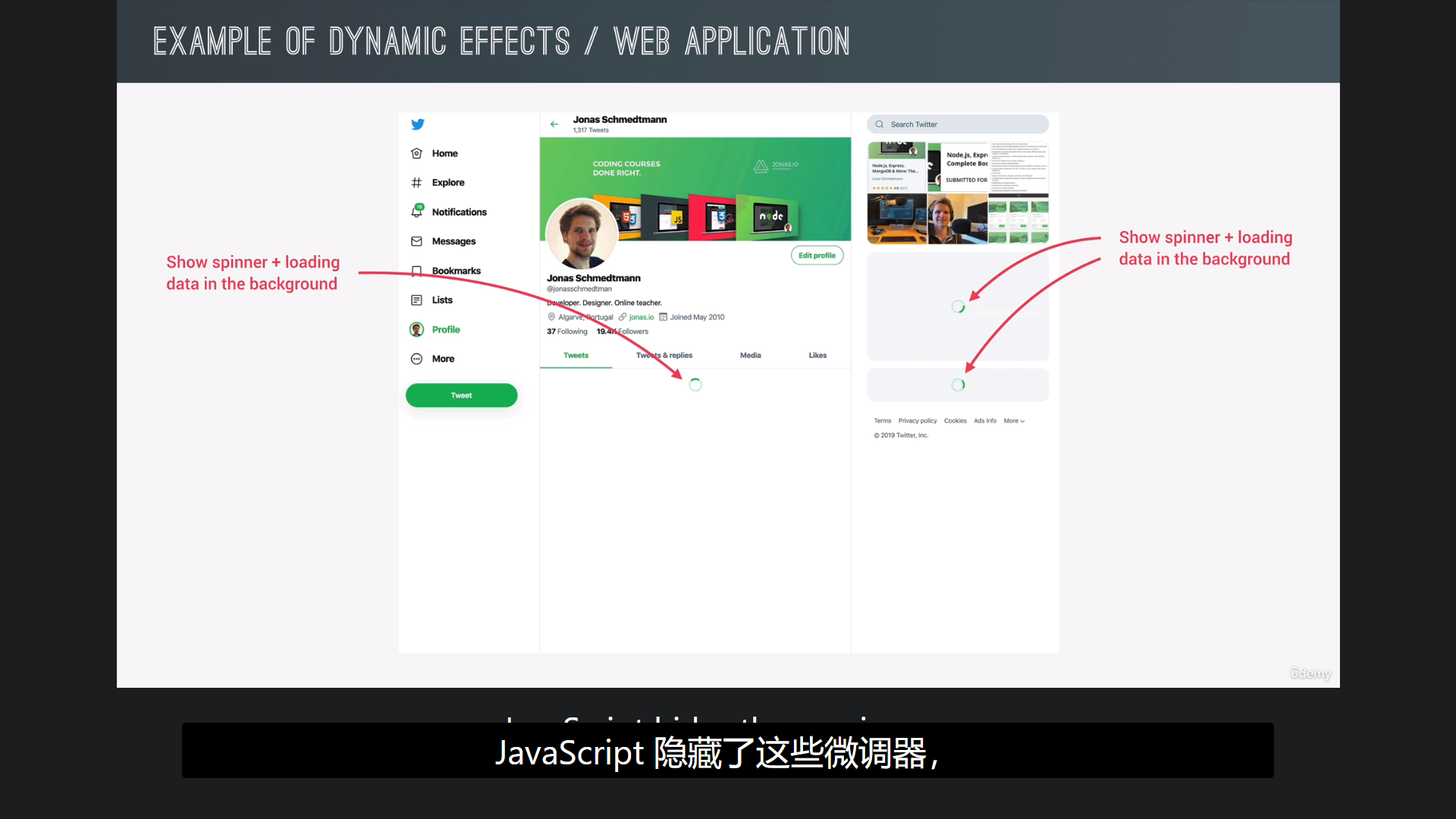
形容词

编程语言：构建WEB应用程序

演示

JAVASCRIPT在WEB开发中的作用

### 8.3 以实际案例展示JavaScript的功能



在后台显示微调器加载数据

动态效果/WEB应用程序示例



加载后显示数据

在鼠标悬停时显示用户信息

加载数据后显示推文

显示推文框

点击后

动态效果/WEB应用程序示例

### ⭐8.4 JavaScript其他很酷的地方

1. 在浏览器中构建整个web应用程序,这些应用程序感觉就像我们每天在计算机和手机上使用的应用程序一样,我们可以说,JavaScript是现代web开发的源泉,实际上是整个现代web本身首先成为可能的原因

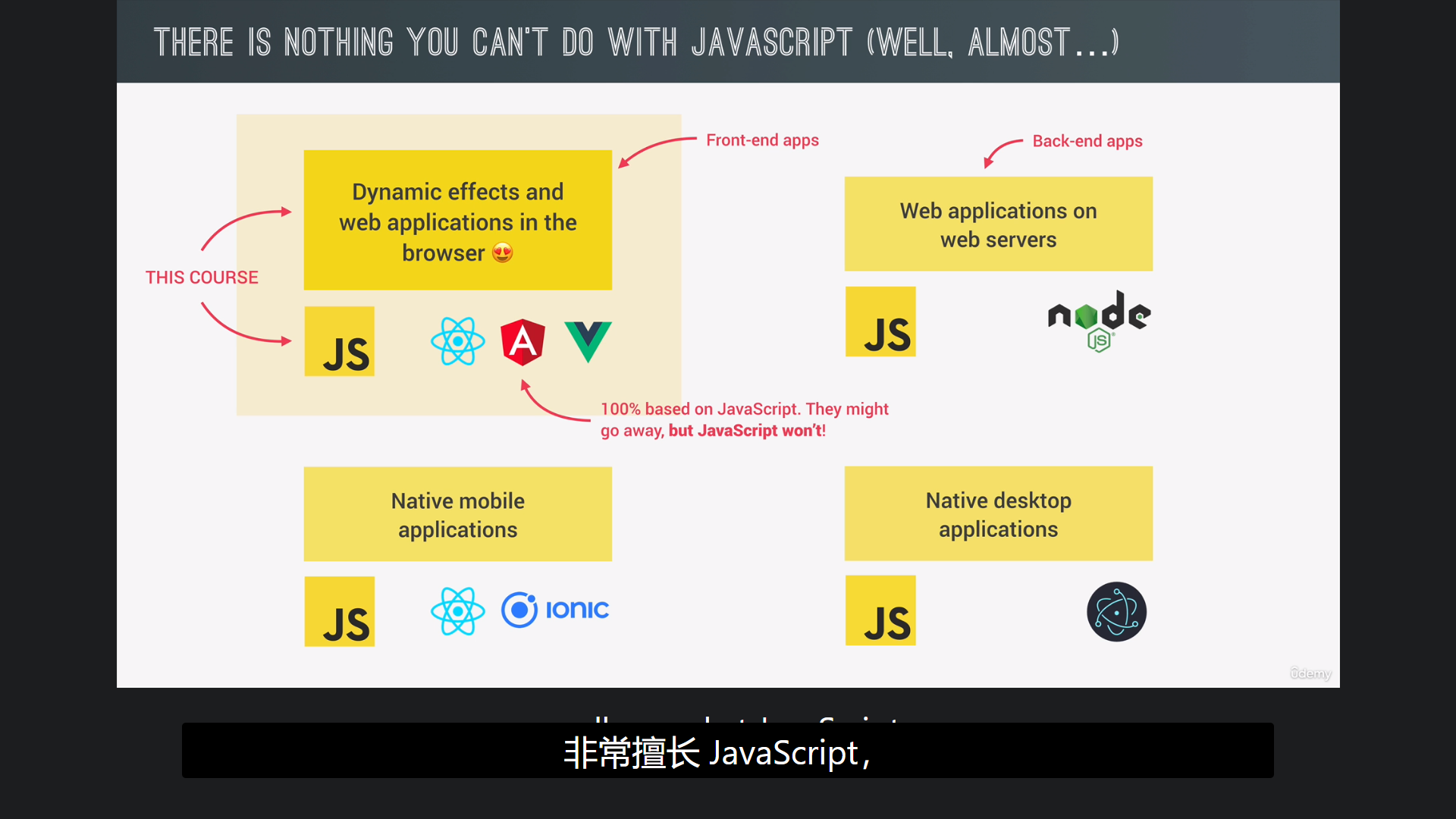
2. 现代这些框架和库仍然100% 基于 JavaScript,这意味着你应该在学习和使用这些框架之前变得非常非常擅长JavaScript,而不是在你只知道10行JavaScript之后就跳入你的第一个框架

3. 现在JavaScript语言和web浏览器实际上是两个不同的东西,这意味着JavaScript也可以在Web浏览器之外运行,例如可以使用一个非常流行的技术Node.js在Web服务器上使用JavaScript,它根本不需要任何浏览器

4. 这使我们能够创建所谓的后端应用程序,这些应用程序只是在Web服务器上运行并与数据库等进行交互的应用程序,另一方面,当我们在浏览器中使用JavaScript时,我们创建了所谓的前端应用程序

5. 我们实际上还可以使用JavaScript来构建原生移动应用程序,以及适用于任何手机和计算机操作系统的原生桌面应用程序,这是因为React Native等现代工具、Ionic框架和Electron等工具

6. 精通JavaScript之后,基本上没有什么不可以构建的！！！



前端应用

这门课

后端应用

本机桌面应用程序

100%基于Javascript。它们可能会消失，但JavaScript不会！

本地移动应用程序

Web服务器上的Web应用程序

浏览器中的动态效果和web应用程序

使用JAVASCRIPT没有什么是做不到的（好吧，几乎…）

## 9. 链接JavaScript文件

### 9.1 js中分号

代表一行代码的结束

### 9.2 js单独提取出来

略过

## 10.值和变量

### 10.1 谈论值的价值观

1. 一个值基本上就是一段数据,所以它们时我们在编程中拥有的最基本的信息单元

a) 信息单元：信息内涵的基本组成成分,在信息描述中一般通过语义上的基本信息单位进行,如采用单元词表来展示信息的基本组成成分

### ⭐10.2 将值存储到变量中

1. 我们声明一个变量,所以这实际上会在你的计算机内存中创建一个真正的变量,我们会将值存储在该变量中

2. 可以想象一个变量,就像一个盒子,所以在显示世界中,一个盒子可以装一些物体,比如一本书,然后我们可以在盒子上写一个标签找到该对象,变量实际上以完全相同的方式工作

3. 我们说，变量是用来存储数值的，那么有一个重要的概念需要区分。变量不是数值本身，它们仅仅是一个用于存储数值的容器。你可以把变量想象成一个个用来装东西的纸箱子。**(重点)**

### ⭐10.3 变量存储值的作用

1. 我们可以多次使用这个变量名在不同的地方,当我们想要修改全部这些值,只需在一个地方也是就初始化变量值的地方修改即可,相反我们必须手动将任何地方的值更改

2. 这是关于变量要记住的重要事情之一

### 10.4 变量的约定和规则

1. firstName驼峰命名推荐

2. first\_name Ruby类似其语言中使用

3. 命名不能以数字开头、关键字也不可以 (name是个例外)

4. 变量名只能包含数字字母、下划线或美元赋值

5. name是关键字但可以用于变量命名,但是不推荐,因为在某种情况下,这会产生一些问题 **(重点)**

6. 全部是大写字母的变量名表示JavaScript中的特定用例,这是面对对象的编程,代表一个常数

7. 变量名要描述存储的值的含义

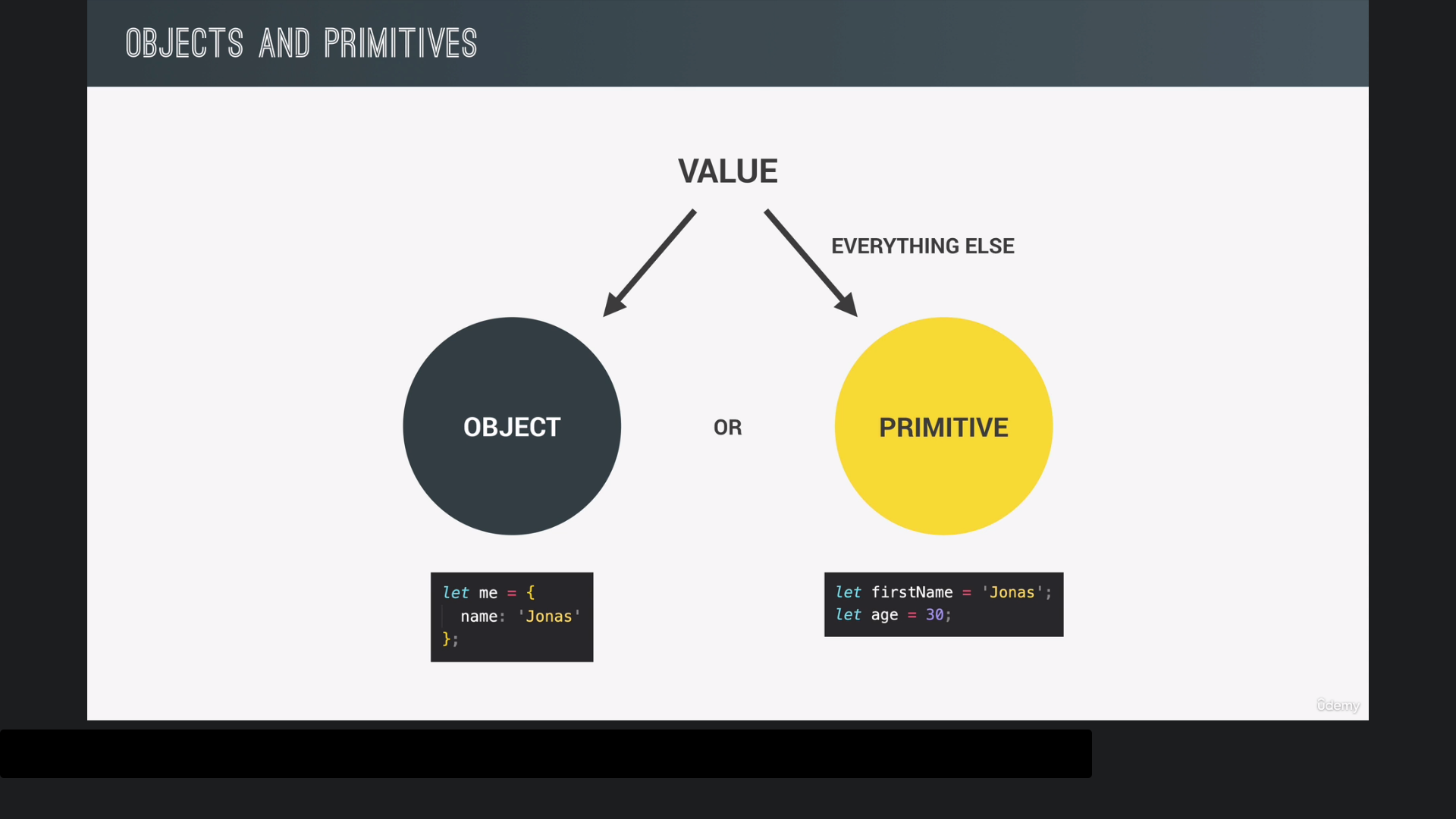
## 11.练习作业

## 12.数据类型

1. JavaScript的简单类型包括 数字、字符串、布尔值、null值、undefined值。 其他的值都是对象。对象是可变的键控集合。数组、函数、正则表达式都是对象

### 12.1 对象和原始值

1. 在JavaScript中,每个值要么是一个看起来像这样的对象,要么是一个原始值,因此当一个值不是对象时,它只是一个原始数据类型 **(重点)**



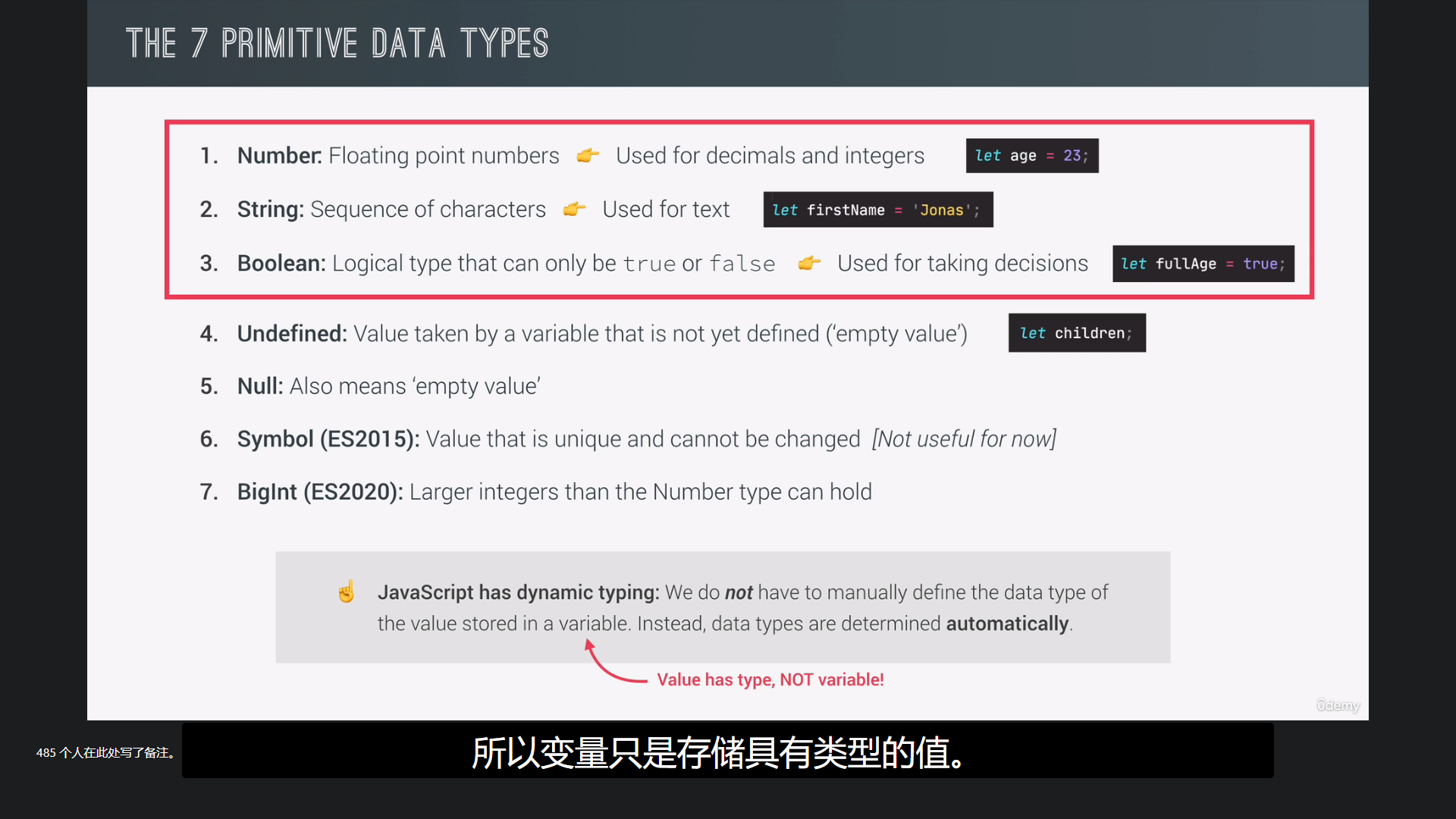
原始的

其他一切

对象和原语

### 12.2 基本数据类型

1. JavaScript脚本会在将值存储到变量时自动确认其数据类型,值和变量之间的区别非常重要,因为在JavaScript中,具有类型的是值,而不是变量 **(重点)**



浮点数

由尚未定义的变量获取的值（空值）

值具有类型，而不是变量！

JavaScript具有动态类型：我们不必手动定义存储在变量中的值的数据类型。相反，数据类型是自动确定的

大于数字类型所能容纳的整数

唯一且无法更改的值暂时无效

Null：也意味着空值

用于做决定

只能为true或false的逻辑类型

用于文本

字符序列

用于小数和整数

基本数据类型

### 12.3 typeof特殊运算符

1. typeof 操作符返回一个字符串，表示未经计算的操作数的类型。

### 12.4 动态数据类型的含义

1. 动态类型只是意味着我们可以轻松地更改由变量保存的值的类型,这意味着我们let variable后,只需variable = '自定义动态类型',就可以动态修改变量的类型了. **(重点)**

2. JavaScript 是一种弱类型或者说动态语言。这意味着你不用提前声明变量的类型，在程序运行过程中，类型会被自动确定。这也意味着你可以使用同一个变量保存不同类型的数据：

### 12.5 未定义和空值

1. 全局属性undefined表示原始值undefined。它是一个JavaScript的 原始数据类型 。

2. 值 null 特指对象的值未设置。它是 JavaScript 基本类型 之一，在布尔运算中被认为是falsy。

## 13.let、const和var

1. ES6中引入了let和const,因为它们是现代JavaScript,而var关键字是声明变量的旧方式

2. 我们了解它们有何不同以及在哪种情况下使用哪一种

3. let是块作用域而var是函数作用域对你来说真的没有用,对吧？因为你甚至不知道块或函数到底是什么？ **(重点)**

### 13.1 let

1. let关键字来声明可以在以后更改的常量,所以基本上在我们的程序执行期间,这正是我们这里所做的

2. JavaScript世界中经常听到的一个术语,当我们需要改变一个变量时,这是使用let的完美用例,这也适用于我们想要声明变量的情况

### ⭐13.2 const

1. const关键字声明未来任何时候都不应该改变的变量,因此const变量中的值不能更改

2. 用const创建的变量是不可变的这一事实也意味着我们不能声明空的const变量 **(重点)**

3. 为初始化值控制台报错：未捕获的语法错误:const声明中缺少初始值设定项 Uncaught SyntaxError: Missing initializer in const declaration

4. 这意味着我们使用const必须设定一个初始值

### 13.3 var

1. var基本上是ES6之前定义变量的就方法,它的工作原理实际上与let几乎相同,所以我们可以写var job等于程序员,然后就像用let一样,我们可以稍后改变变量的值

### ⭐13.4 没有声明变量

1. 我们没有使用let、const或var,但它仍然有效,然而这实际上是一个非常糟糕的想法,因为这不会在当前所谓的范围内创建变量,相反,JavaScript将在全局对象上创建一个属性**(重点)**

2. 稍后你会明白所有这些意味着什么,但重要的是你应该始终正确地声明变量,好吗？永远不要只写这样的变量而不真正声明它

## 14.基本运算符

1. 运算符基本上允许我们转换值或组合多个值,并真正对值进行各种工作,并且有许多类别的运算符,例如数学运算符、比较运算符、逻辑运算符、赋值运算符等等

### 14.1 数学运算符

    const now = 2037;

    const ageJonas = now - 1991;

    const ageSarah = now - 2018;

    console.log(ageJonas, ageSarah);

    console.log(ageJonas \* 2, ageJonas / 10, 2 \*\* 3);

    // 2 \*\* 3 means 2 to the power of 3 = 2 \* 2 \* 2

    const firstName = 'Jonas';

    const lastName = 'Schmedtmann';

    console.log(firstName + ' ' + lastName);

### 14.2 赋值运算符

    let x = 10 + 5; // 15

    x += 10; // x = x + 10 = 25

    x \*= 4; // x = x \* 4 = 100

    x++; // x = x + 1

    x--;

    x--;

    console.log(x);

### 14.3 比较运算符

    console.log(ageJonas > ageSarah); // >, <, >=, <=

    console.log(ageSarah >= 18);

    const isFullAge = ageSarah >= 18;

    console.log(now - 1991 > now - 2018);

## 15.运算符优先级

    const now = 2037;

    const ageJonas = now - 1991;

    const ageSarah = now - 2018;

    console.log(now - 1991 > now - 2018);

    let x, y;

    x = y = 25 - 10 - 5; // x = y = 10, x = 10

    console.log(x, y); // 10 10

    const averageAge = (ageJonas + ageSarah) / 2;

    console.log(ageJonas, ageSarah, averageAge);

## 16.编码挑战1

Mark 和 John 正在尝试比较他们的 BMI（身体质量指数），该指数使用以下公式计算：BMI = 质量 / 身高 \*\* 2 = 质量 /（身高 \* 身高）。 （质量单位为公斤，高度单位为米）。

1. 将 Mark 和 John 的质量和高度存储在变量中

2.使用公式计算他们的BMI（你甚至可以实现两个版本）

3. 创建一个布尔变量“markHigherBMI”，其中包含有关 Mark 的 BMI 是否高于 John 的信息。

测试数据 1：标记重 78 kg，高 1.69 m。 John 体重 92 kg，身高 1.95 m。

测试数据 2：标记重 95 kg，高 1.88 m。 约翰体重 85 kg，身高 1.76 m。

    const massMark = 95;

    const heightMark = 1.88;

    const massJohn = 85;

    const heightJohn = 1.76;

    const BMIMark = massMark / heightMark \*\* 2;

    const BMIJohn = massJohn / (heightJohn \* heightJohn);

    const markHigherBMI = BMIMark > BMIJohn;

    console.log(BMIMark, BMIJohn, markHigherBMI);

## 17. 字符串和模板文字

1. 模板字面量 是允许嵌入表达式的字符串字面量。你可以使用多行字符串和字符串插值功能。它们在ES2015规范的先前版本中被称为“模板字符串”。

2. 字面量表示如何表达这个值,一般除去表达式,给变量赋值时,等号右边都可以认为是字面量

  const firstName = 'Jonas';

  const job = 'teacher';

  const birthYear = 1991;

  const year = 2037;

### 17.1描述对比

  const jonas = 'I"m ' + firstName + ', a ' + (year - birthYear) + ' years old ' + job +

'!';

  console.log(jonas);

  const jonasNew = `I'm ${firstName}, a ${year - birthYear} year old ${job}!`;

  console.log(jonasNew);

  console.log(`Just a regular string...`);

### ⭐17.1 描述问题对比

  console.log('String with \n\

  multiple \n\

  lines');

  console.log(`String with

  multiple

  lines`);

### 18.决策: if / else 语句

    const age = 15;

    if (age >= 18) {

    console.log('Sarah can start driving license 🚗');

    } else {

    const yearsLeft = 18 - age;

    console.log(`Sarah is too young. Wait another ${yearsLeft} years :)`);

    }

    const birthYear = 2012;

    let century;

    if (birthYear <= 2000) {

    century = 20;

    } else {

    century = 21;

    }

    console.log(century);

## 19.编码挑战2

使用 Challenge #1 中的 BMI 示例和您已经编写的代码，并对其进行改进：

1. 在控制台打印一个漂亮的输出，说明谁的 BMI 更高。 消息可以是“Mark 的 BMI 比 John 的高！” 或“约翰的 BMI 比马克的高！”

2. 使用模板文字将 BMI 值包含在输出中。 示例：“Mark 的 BMI (28.3) 高于 John 的 (23.9)！”

提示：使用 if/else 语句😉

祝你好运😀

    const massMark = 78;

    const heightMark = 1.69;

    const massJohn = 92;

    const heightJohn = 1.95;

    // const massMark = 95;

    // const heightMark = 1.88;

    // const massJohn = 85;

    // const heightJohn = 1.76;

    const BMIMark = massMark / heightMark \*\* 2;

    const BMIJohn = massJohn / (heightJohn \* heightJohn);

    console.log(BMIMark, BMIJohn);

    if (BMIMark > BMIJohn) {

    console.log(`Mark's BMI (${BMIMark}) is higher than John's (${BMIJohn})!`)

    } else {

    console.log(`John's BMI (${BMIJohn}) is higher than Marks's (${BMIMark})!`)

    }

## 20.类型转换和强制转换

1. 类型强制是从一种数据类型到另一种数据类型的值的自动或隐式转换（例如字符串到数字）。类型转换类似于类型强制，因为它们都将值从一种数据类型转换为另一种数据类型，但有一个关键区别——类型强制是隐式的，而类型转换可以是隐式的或显式的。

**(重点)**

2. 类型转换（或类型转换）意味着将数据从一种数据类型传输到另一种数据类型。当编译器自动分配数据类型时会发生隐式转换，但源代码也可以显式要求进行转换。例如，给定指令5+2.0，浮点数2.0被隐式类型转换为整数，但给定指令Number("0x11")，字符串“0x11”被显式类型转换为数字 17。**(重点)**

### 21.1 类型转换

1. 类型转换是我们手动从一种类型转换为另一种类型

    const inputYear = '1991';

    console.log(Number(inputYear), inputYear);

    console.log(Number(inputYear) + 18);

    console.log(Number('Jonas'));

    console.log(typeof NaN); (重点)

    console.log(String(23), 23);

### ⭐21.2 强制转换

1. 类型强制是指JavaScript在后台自动为我们转换类型

2. 字符串类型值相减会返回数字类型,相加会返回字符串类型 **(重点)**

    // type coercion

    console.log('I am ' + 23 + ' years old');

    console.log('23' - '10' - 3);

    console.log('23' / '2');

    let n = '1' + 1; // '11' (重点)

    n = n - 1;

    console.log(n);

## 21.真值和假值

### 21.1 实践布尔值转换采用隐式

1. 在实践中,到布尔值的转换始终是隐式的,而不是显式的,或者换句话说,始终是JavaScript在后台自动执行的类型强制转换

### ⭐22.2 JavaScript究竟什么时候会对布尔值进类型强制呢？

1. 使用逻辑运算符时,(&& || !== ===)

2. 在逻辑上下文中(if else if(0) if(null) if(exoression--表达式))

### 21.3 Boolean()函数转换五种为假值的值

// 5 falsy values: 0, '', undefined, null, NaN (重点)

    console.log(Boolean(0)); // false

    console.log(Boolean(undefined)); // false

    console.log(Boolean('Jonas')); // true

    console.log(Boolean({})); // true (重点)

    console.log(Boolean('')); // false

### ⭐21.4 逻辑上下文中使用表单式判断

    const money = 100;

    if (money) {

    console.log("Don't spend it all ;)"); // Don't spend it all ;) ;

    } else {

    console.log('You should get a job!');

    }

    let height = 0;

    if (height) {

    console.log('YAY! Height is defined');

    } else {

    console.log('Height is UNDEFINED'); // Height is UNDEFINED

    }

### 🎨 21.4 什么是真值什么是假值？

1. 在 JavaScript 中，truthy（真值）指的是在布尔值上下文中，转换后的值为真的值。所有值都是真值，除非它们被定义为 假值（即除 false、0、-0、0n、""、null、undefined 和 NaN 以外皆为真值）。

2. 这句话有点拗口，其实就是说，除了上边所说的8种值是falsey，其余都是真值

3. false、0、-0、0n、""、null、undefined 和 NaN 这几种都是假值

### 🎨21.5 !!双重非运算符

1. 双重非运算符：显式地将任意值强制转换为其对应的布尔值, 其实就是Boolean函数的另一个表达方式

    console.log(!!'124'); // true



## 22.等式运算符==和===

1. 全等运算符 (===) 会检查它的两个操作数是否相等，并且返回一个布尔值结果。与相等运算符不同，全等运算符总是认为不同类型的操作数是不同的。

2. 全等运算符与相等运算符（==）最显著的区别是，如果操作数的类型不同，== 运算符会在比较之前尝试将它们转换为相同的类型。

3. === 全等运算法, ==相等运算法

    const age = '18';

    if (age === 18) console.log('You just became an adult :D (strict)');

    if (age == 18) console.log('You just became an adult :D (loose)');

    const favourite = Number(prompt("What's your favourite number?")); (重点)

    console.log(favourite);

    console.log(typeof favourite);

    if (favourite === 23) { // 22 === 23 -> FALSE

    console.log('Cool! 23 is an amzaing number!')

    } else if (favourite === 7) {

    console.log('7 is also a cool number')

    } else if (favourite === 9) {

    console.log('9 is also a cool number')

    } else {

    console.log('Number is not 23 or 7 or 9')

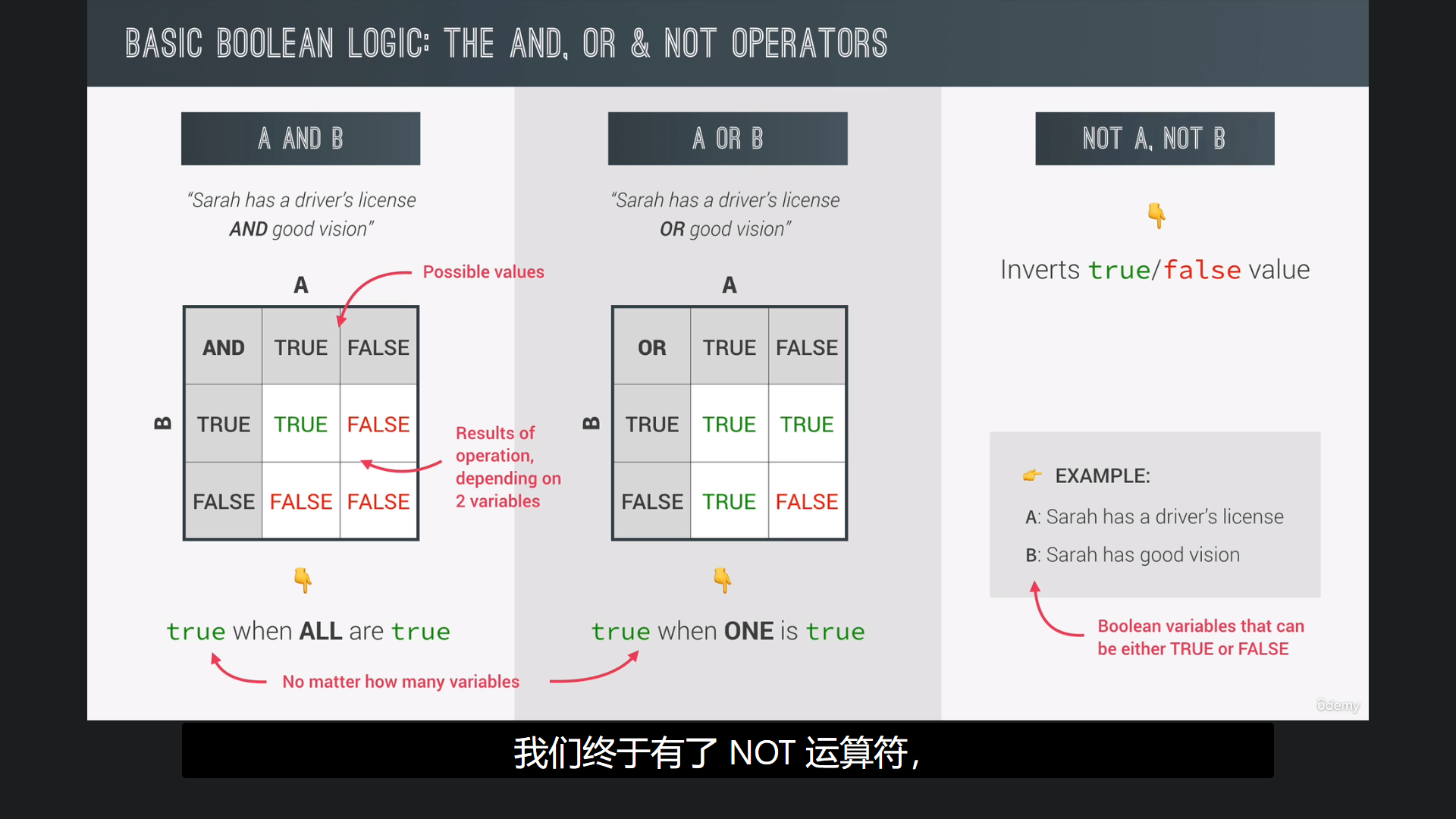
    }

    if (favourite !== 23) console.log('Why not 23?');

## 23.布尔逻辑

1. &&优先级大于|| **(重点)**

2. 布尔逻辑中先执行！非运算符

当一切都是真的时候才是真的

当一个人是真的时，他才是真的

不管有多少变量

布尔变量可以是真的，也可以是假的

结果活动

取决于2个变量

可能值

反转真/假值

莎拉有dnvers驾照

还是良好的视力

莎拉有德尼弗的驾照和良好的视力

基本布尔逻辑：AND。还是不是运营商

## 24.逻辑运算符

    const hasDriversLicense = true; // A

    const hasGoodVision = true; // B

    console.log(hasDriversLicense && hasGoodVision);

    console.log(hasDriversLicense || hasGoodVision);

    console.log(!hasDriversLicense);

    // if (hasDriversLicense && hasGoodVision) {

    //   console.log('Sarah is able to drive!');

    // } else {

    //   console.log('Someone else should drive...');

    // }

    const isTired = false; // C

    console.log(hasDriversLicense && hasGoodVision && isTired);

    if (hasDriversLicense && hasGoodVision && !isTired) {

    console.log('Sarah is able to drive!');

    } else {

    console.log('Someone else should drive...');

    }

## 25.编码挑战3

有两个体操队，海豚队和考拉队。他们相互竞争 3 次。平均分最高的获胜者赢得奖杯！

1. 使用下面的测试数据计算每支球队的平均分

2. 比较队伍的平均分来决定比赛的获胜者，并打印到控制台。不要忘记可能会有平局，因此也要对此进行测试（平局意味着他们的平均得分相同）。

3. BONUS 1：包括最低分数为 100 的要求。在此规则下，只有当一个团队的得分高于另一团队并且同时得分至少为 100 分时，它才会获胜。提示：使用逻辑运算符来测试最低分数，以及多个 else-if 块😉

4. 奖励 2：最低分数也适用于平局！因此，只有当两支球队得分相同且得分大于或等于 100 分时，才会发生平局。否则，没有球队赢得奖杯。

测试数据：海豚得分 96、108 和 89。考拉得分 88、91 和 110

测试数据奖励 1：海豚得分 97、112 和 101。考拉得分 109、95 和 123

测试数据奖励 2：海豚得分 97、112 和 101。考拉得分 109、95 和 106

    // const scoreDolphins = (96 + 108 + 89) / 3;

    // const scoreKoalas = (88 + 91 + 110) / 3;

    // console.log(scoreDolphins, scoreKoalas);

    // if (scoreDolphins > scoreKoalas) {

    //   console.log('Dolphins win the trophy 🏆');

    // } else if (scoreKoalas > scoreDolphins) {

    //   console.log('Koalas win the trophy 🏆');

    // } else if (scoreDolphins === scoreKoalas) {

    //   console.log('Both win the trophy!');

    // }

    // BONUS 1

    const scoreDolphins = (97 + 112 + 80) / 3;

    const scoreKoalas = (109 + 95 + 50) / 3;

    console.log(scoreDolphins, scoreKoalas);

    if (scoreDolphins > scoreKoalas && scoreDolphins >= 100) {

    console.log('Dolphins win the trophy 🏆');

    } else if (scoreKoalas > scoreDolphins && scoreKoalas >= 100) {

    console.log('Koalas win the trophy 🏆');

    } else if (scoreDolphins === scoreKoalas && scoreDolphins >= 100 && scoreKoalas >= 100) {

    console.log('Both win the trophy!');

    } else {

    console.log('No one wins the trophy 😭');

    }

## 26.switch语句

1. switch语句评估一个表达式,将表达式的值与case子句匹配,并执行与该情况相关联的语句

2. switch()括号中的表达式用于匹配 expression 的 case 子句。如果 expression 与给定的 valueN 相匹配，则执行该 case 子句中的语句直到该 switch 语句结束或遇到一个 break

3. 一个 default 子句；如果给定，这条子句会在 expression 的值与任一 case 语句均不匹配时执行

4. switch匹配表达式是严格运算符===

5. 如果有相同case但分别输出不同内容,程序会匹配第一个case子句

    const day = 'friday';

    switch (day) {

    case 'monday': // day === 'monday'

        console.log('Plan course structure');

        console.log('Go to coding meetup');

        break;

    case 'tuesday':

        console.log('Prepare theory videos');

        break;

    case 'wednesday': (重点)

    case 'thursday':

        console.log('Write code examples');

        break;

    case 'friday':

        console.log('Record videos');

        break;

    case 'saturday':

    case 'sunday':

        console.log('Enjoy the weekend :D');

        break;

    default:

        console.log('Not a valid day!');

    }

    if (day === 'monday') {

    console.log('Plan course structure');

    console.log('Go to coding meetup');

    } else if (day === 'tuesday') {

    console.log('Prepare theory videos');

    } else if (day === 'wednesday' || day === 'thursday') { (重点)

    console.log('Write code examples');

    } else if (day === 'friday') {

    console.log('Record videos');

    } else if (day === 'saturday' || day === 'sunday') {

    console.log('Enjoy the weekend :D');

    } else {

    console.log('Not a valid day!');

    }

## 27.语句和表达式

### ⭐27.1 语句和表达式的区别

1. 本质上,表达式是一段产生值的代码,列如3+4会产生7数字,true && !false 会产生true布尔值

2. 语句就像一段更大的代码,它被执行并且自身不会产生任何值,比如我们使用if else语句,声明就像一个完整的句子,而表达式就像组成句子的单词

3. 只要某事以分号结尾,那就是一个声明,就像一个完整的句子

### 27.2表达式示例

    3 + 4

    1991

    true && false && !false

### ⭐27.3语句示例

1. 基本上我们将整个编程编写为一系列动作,这些动作是陈述,因此让我们以if else语句为例,如果满足条件我们创建一个名

为xx的字符串,代码本身在这里并不重要,所以if else实际上是一个语句,switch语句也是一样的

2. 这里的这个陈述并没有真正产生价值,我们只是简单地声明了一个变量,我们在满足的情况下声明了这个变量,但不产生值,现在字符串本身又是一个表达式,概念是模糊的(不必追踪细节),我们只需要区分语句和表达式就可以了

    if (23 > 10) {

    const str = '23 is bigger';

    }

### ⭐27.4 模板文字示例

1. 在模板文字中，我们只能插入表达式，但不能插入语句 **(重点)**

    const me = 'Jonas';

    console.log(`I'm ${2037 - 1991} years old ${me}`);

## 28.三元运算符

### ⭐28.1 三元运算符与else if对比示例

1. 我们不使用语句,而是使用此条件运算符,条件运算符也成为三元运算符,因为它分为三个部分,与其他运营商不同,列如加号

运算符只有两部分

2. 三元运算符分为三个部分,1:判断情况 2: 符合情况输出值 3:不符合情况输出值

3. 条件运算符实际上是一个运算符,运算符总产生一个值,换句话说,运算符是一个表达式 **(重点)**

4. 使用三元运算符对基本上有条件地声明变量非常有用

  // The Conditional (Ternary) Operator

    const age = 23;

    // age >= 18 ? console.log('I like to drink wine 🍷') : console.log('I like to drink water 💧');

    const drink = age >= 18 ? 'wine 🍷' : 'water 💧';

    console.log(drink);

    let drink2;

    if (age >= 18) {

    drink2 = 'wine 🍷';

    } else {

    drink2 = 'water 💧';

    }

    console.log(drink2);

### ⭐28.2模板文字中使用三元运算符

    console.log(`I like to drink ${age >= 18 ? 'wine 🍷' : 'water 💧'}`);

### ⭐28.3三元运算符不是if else的替代品

1. 牢记运算符和表达式的概念

2. 需要根据条件执行更大的代码块时, 三元运算符是行不通的 **(重点)**

3. 快速做出决定时, 在 JavaScript 真正需要表达式的地方尤其如此，就像在这个模板文字中一样, 三元运算符是完美的 **(重点)**

### 🎨 28.4 表达式和运算符