

14. JavaScript I

罗翔宇

中国人民大学统计与大数据研究院

此课件内容及案例均来自于Eric Freeman,
Elisabeth Robson编著的《Head First JavaScript
程序设计》(O'Reilly, 袁国忠译)

什么是JavaScript

- ▶ 一款纯正的**Web编程语言**(标准的Web脚本语言),能够给网页添加行为。比如,能够与用户互动响应有趣的事件;从网上收集数据应用于网页中;在网页中绘制图形
- ▶ JavaScript是**最流行**的编程语言之一,所有现代的浏览器都支持它
- ▶ 使用范围不局限于浏览器,可以用于其他应用软件中
- ▶ 和Java除了名字外毫无关系(为了搭上Java这辆顺风车)
- ▶ 为什么要使用JavaScript?
 - ▶ 静态的网页已经没有市场价值,要给人留下印象,网页必须是动态、交互性的
 - ▶ 行业发展不再倾向于Flash创建动态页面

JavaScript工作原理

- ▶ 网页生态系统：
- ▶ HTML (Hypertext Markup Language, 超文本标记语言)来制定网页的内容和结构
- ▶ CSS (Cascading Style Sheets, 层叠样式表)制定网页的外观：颜色、字体、边框、边距等
- ▶ JavaScript能够在网页中进行编程，实现计算、响应、绘画、通信、提醒等
 - ▶ 检查用户的表单输入
 - ▶ 微博中提取并显示消息
 - ▶ 网页中运行游戏

JavaScript工作原理

- ▶ **编写**：直接在网页中添加JavaScript代码，或者将其放在独立的文件中，并在网页中包含该文件
- ▶ **加载**：在浏览器中输入网页的地址，遇到JavaScript代码后，将立即对其分析，为执行做好准备
- ▶ **执行**：浏览器在网页的整个生命周期内不断执行代码

网页基本框架

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Your HTML Page</title>
    <script>
      ← Your JavaScript code will typically go here.
    </script>
  </head>
  <body>
    ← Any web page content will go here.
  </body>
</html>
```

// 浏览器得知自己需要处理的内容是html

// 文档中html开始的部分, lang="en" 表示内容为英文

// 网页<head>元素

// <script>元素中编写JavaScript代码

也可以放在<body>元素中

// 网页<body>元素

将JavaScript代码加入网页

► 直接写入代码

The type attribute tells the browser you're writing JavaScript. The thing is, browsers assume you're using JavaScript if you leave it off. So, we recommend you leave it off, and so do the people who write the standards.

The `<script>` opening tag.

```
<script type="text/javascript">
```

Don't forget the right bracket on the opening tag.

```
alert("Hello world!");
```

Everything between the script tags must be valid JavaScript.

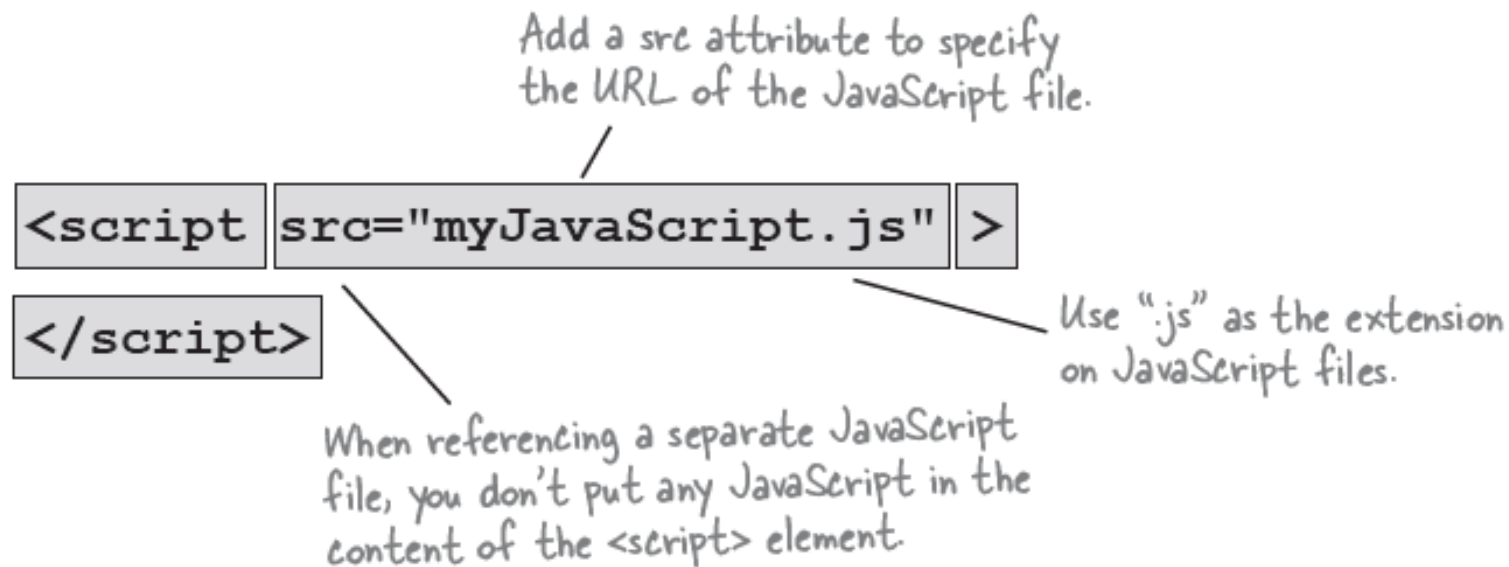
```
</script>
```

You must end the script with a closing `</script>` tag, always!

- `type="text/javascript"` 告诉浏览器你将用JavaScript；如果你不用`type`这一项，浏览器会默认你用的JavaScript
- 开始标签 `<script>`，结束标签 `</script>`
- `alert("Hello world!")` JavaScript 代码

将JavaScript代码加入网页

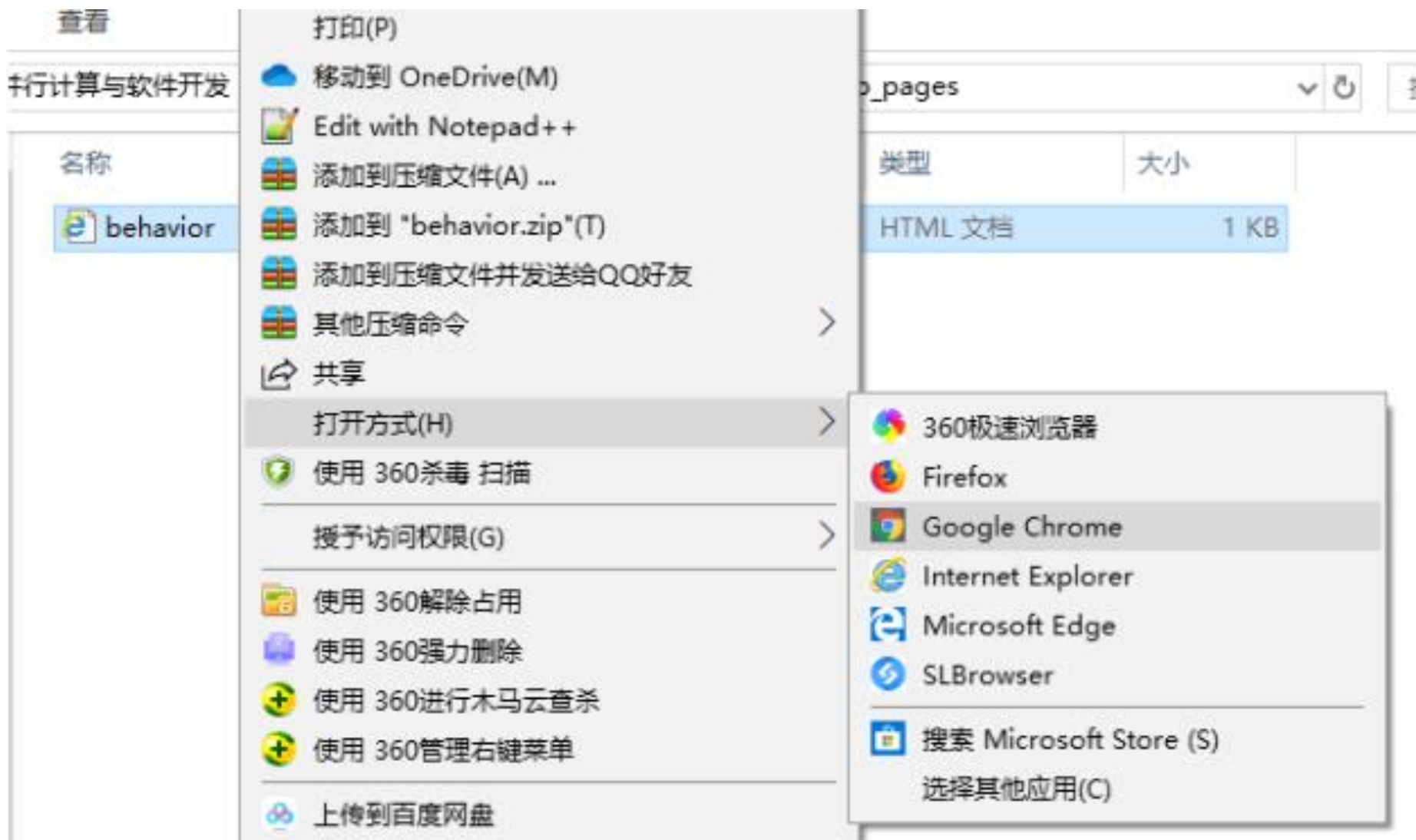
► 引用独立JavaScript(JS)文件



- 利用开始标签<script>中的src项引用JS文件，JS文件的后缀为.js
- 当通过这种方式引用JS文件的时候，不要在<script>和</script>中加入任何代码

使用JS的工具---Chrome浏览器

► 用Chrome打开behavior.html



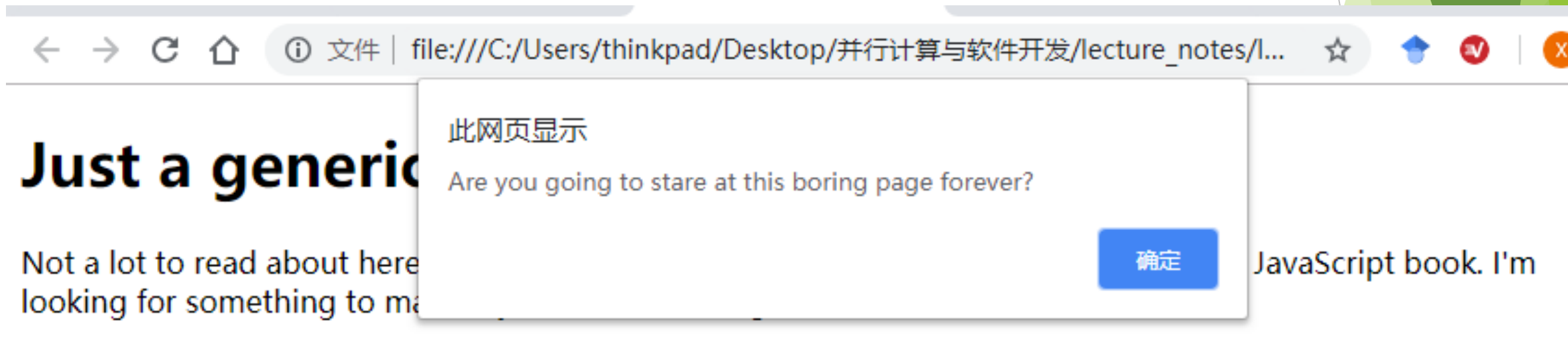
使用JS的工具---Chrome浏览器

- 打开后所看到的内容（静态的）

Just a generic heading

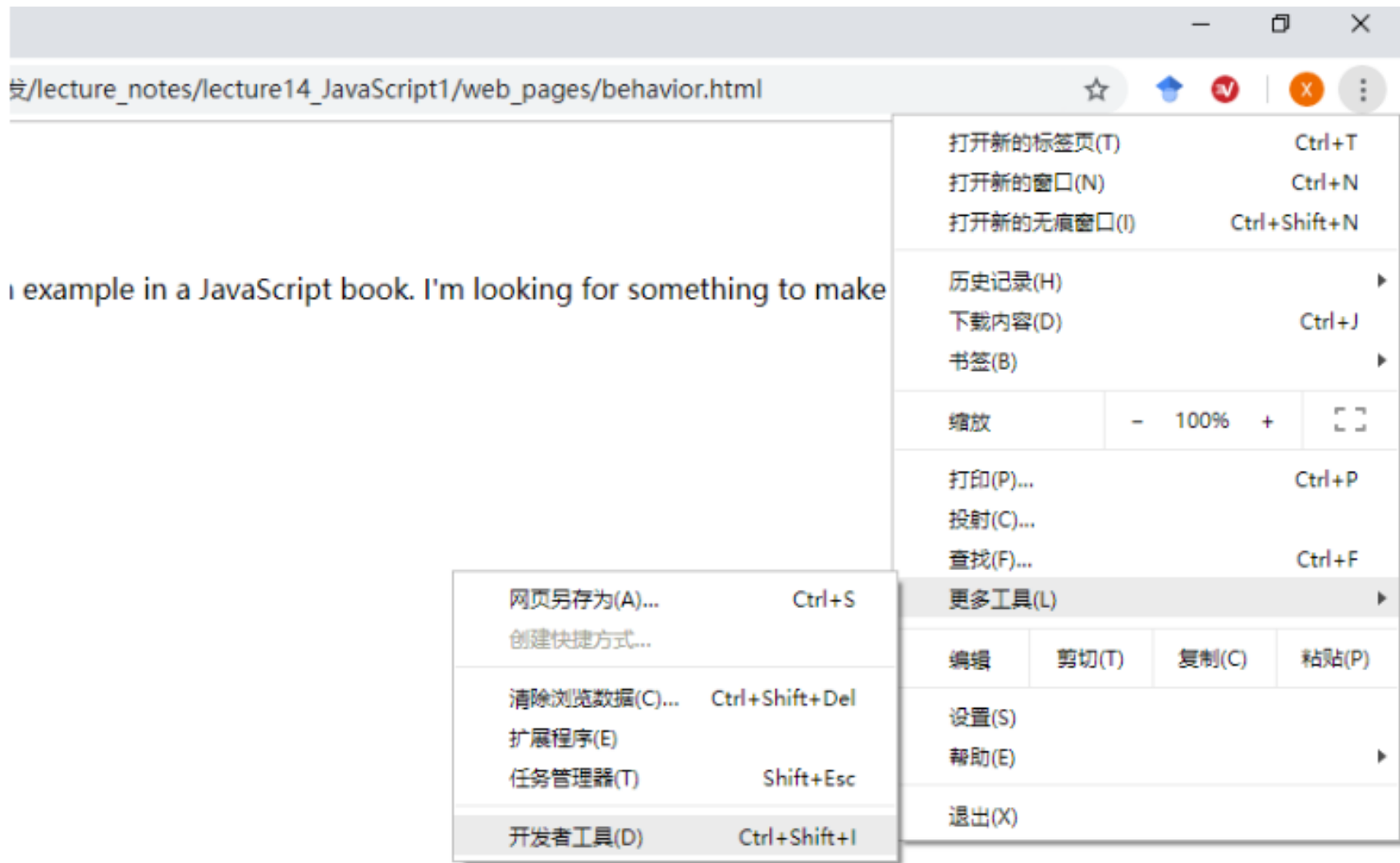
Not a lot to read about here. I'm just an obligatory paragraph living in an example in a JavaScript book. I'm looking for something to make my life more exciting.

- 5秒后，出现的情形（动态的消息提示）



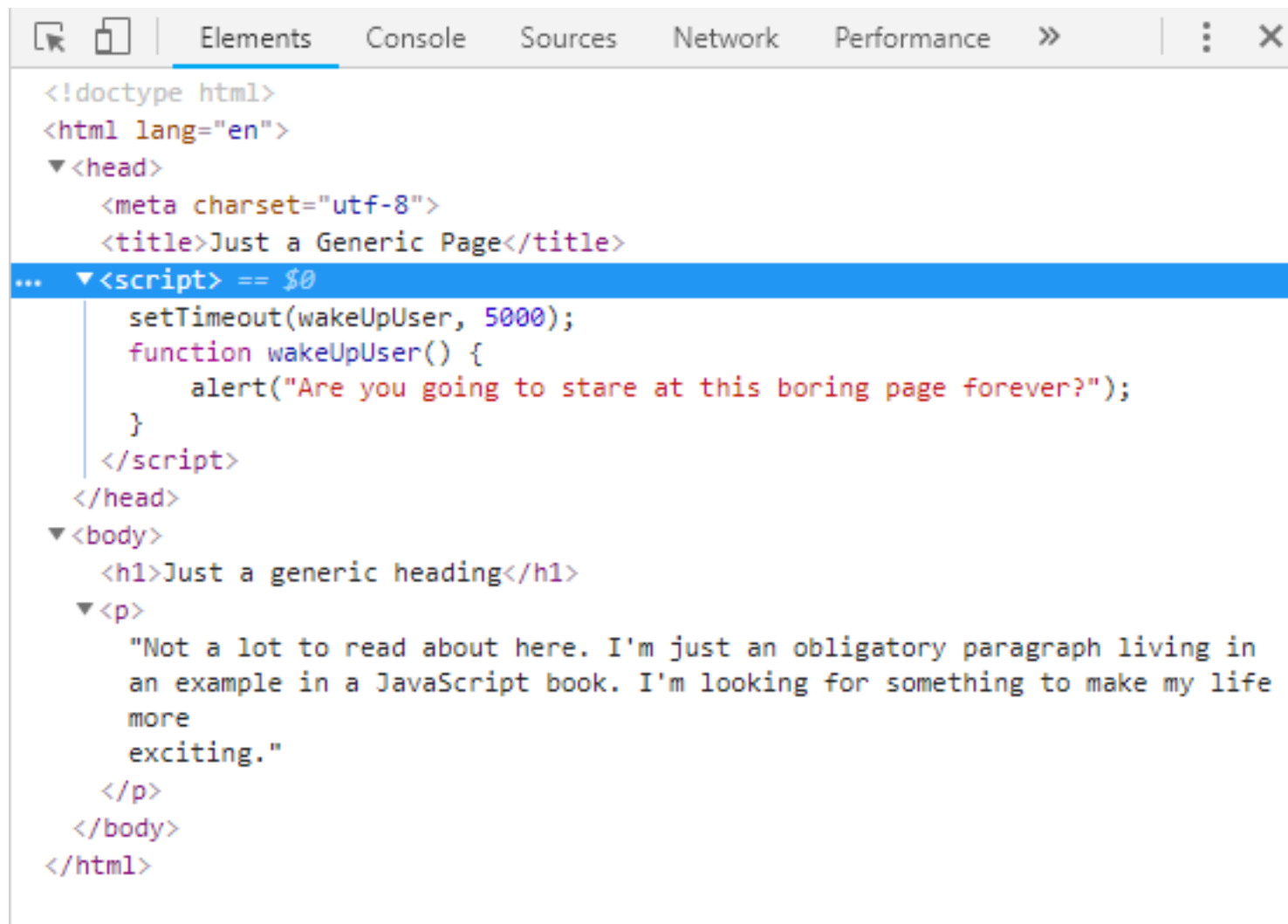
使用JS的工具---Chrome浏览器

► 背后的Html代码，以及其中的JS代码，如何查询？



使用JS的工具---Chrome浏览器

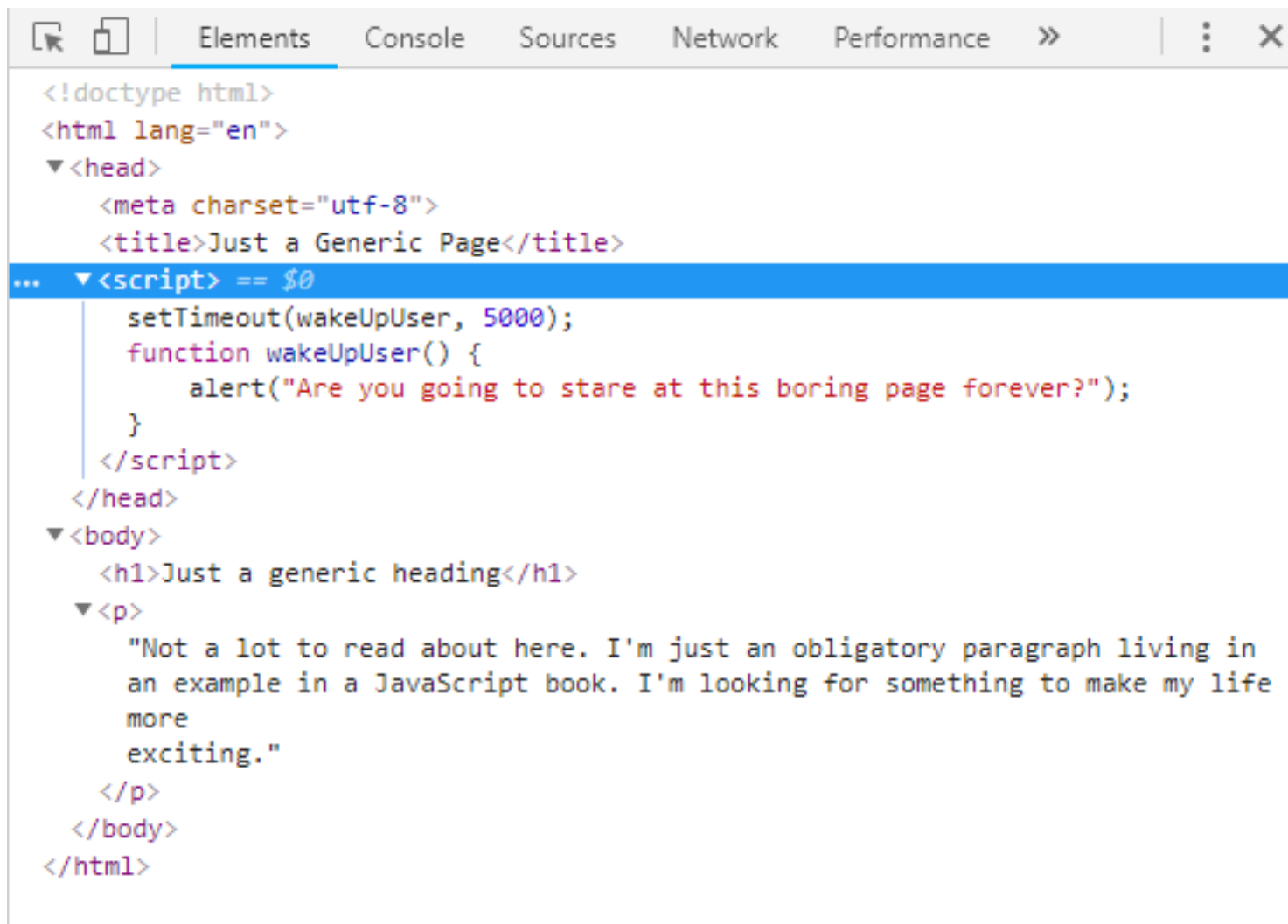
► 背后的Html代码，以及其中的JS代码，如何查询？



The screenshot shows the Chrome DevTools 'Elements' panel. The top toolbar includes icons for back, forward, and a search icon, followed by tabs for 'Elements', 'Console', 'Sources', 'Network', and 'Performance'. The 'Elements' tab is active, displaying the DOM tree and the corresponding HTML code. The DOM tree shows a root node with children: <html>, <head>, and <body>. The <head> node is expanded, showing <meta> and <title> elements. The <script> element is selected, and its code is displayed in the editor below. The code includes a setTimeout call and a function named wakeUpUser that shows an alert. The <body> node is also expanded, showing <h1> and <p> elements. The <p> element contains a long paragraph of text.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Just a Generic Page</title>
    ... <script> == $0
      setTimeout(wakeUpUser, 5000);
      function wakeUpUser() {
        alert("Are you going to stare at this boring page forever?");
      }
    </script>
  </head>
  <body>
    <h1>Just a generic heading</h1>
    <p>
      "Not a lot to read about here. I'm just an obligatory paragraph living in
      an example in a JavaScript book. I'm looking for something to make my life
      more
      exciting."
    </p>
  </body>
</html>
```

使用JS的工具---Chrome浏览器

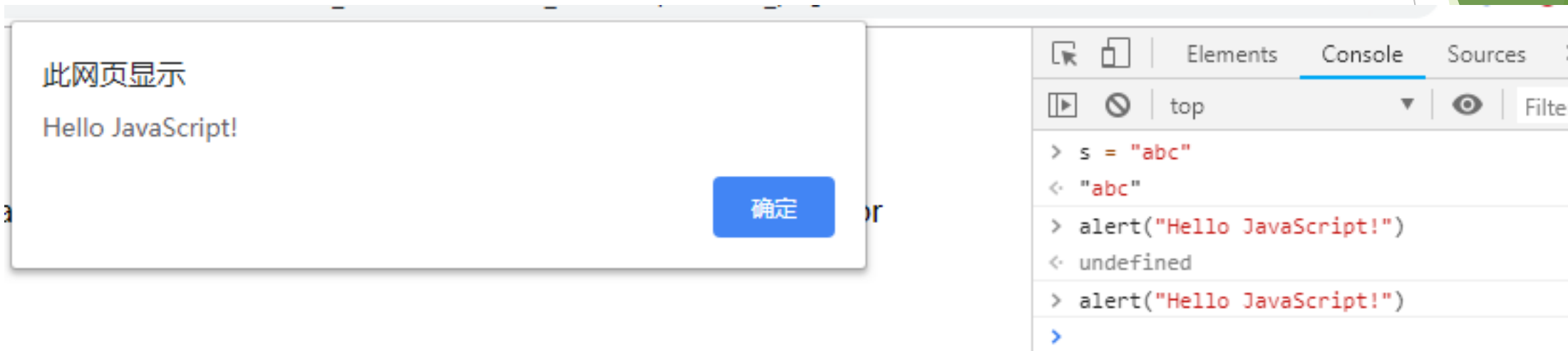
A screenshot of the Chrome DevTools interface. The 'Elements' panel is active, showing the DOM tree of a web page. The root element is an HTML document. The 'head' element is expanded, showing a 'meta' tag for charset and a 'title' tag. A script element is highlighted in blue, containing a 'setTimeout' call and a 'wakeUpUser' function. The 'body' element is also expanded, showing an 'h1' tag and a 'p' tag with a paragraph of text.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Just a Generic Page</title>
    ... <script> == $0
      setTimeout(wakeUpUser, 5000);
      function wakeUpUser() {
        alert("Are you going to stare at this boring page forever?");
      }
    </script>
  </head>
  <body>
    <h1>Just a generic heading</h1>
    <p>
      "Not a lot to read about here. I'm just an obligatory paragraph living in
      an example in a JavaScript book. I'm looking for something to make my life
      more
      exciting."
    </p>
  </body>
</html>
```

- ▶ `setTimeout(wakeUpUser, 5000)`表示等待5000毫秒后，执行`wakeUpUser`函数；`wakeUpUser`函数产生一个消息框，“Are you going to ... page forever?”

使用JS的工具---Chrome浏览器

- ▶ 在Console中，直接书写JS代码



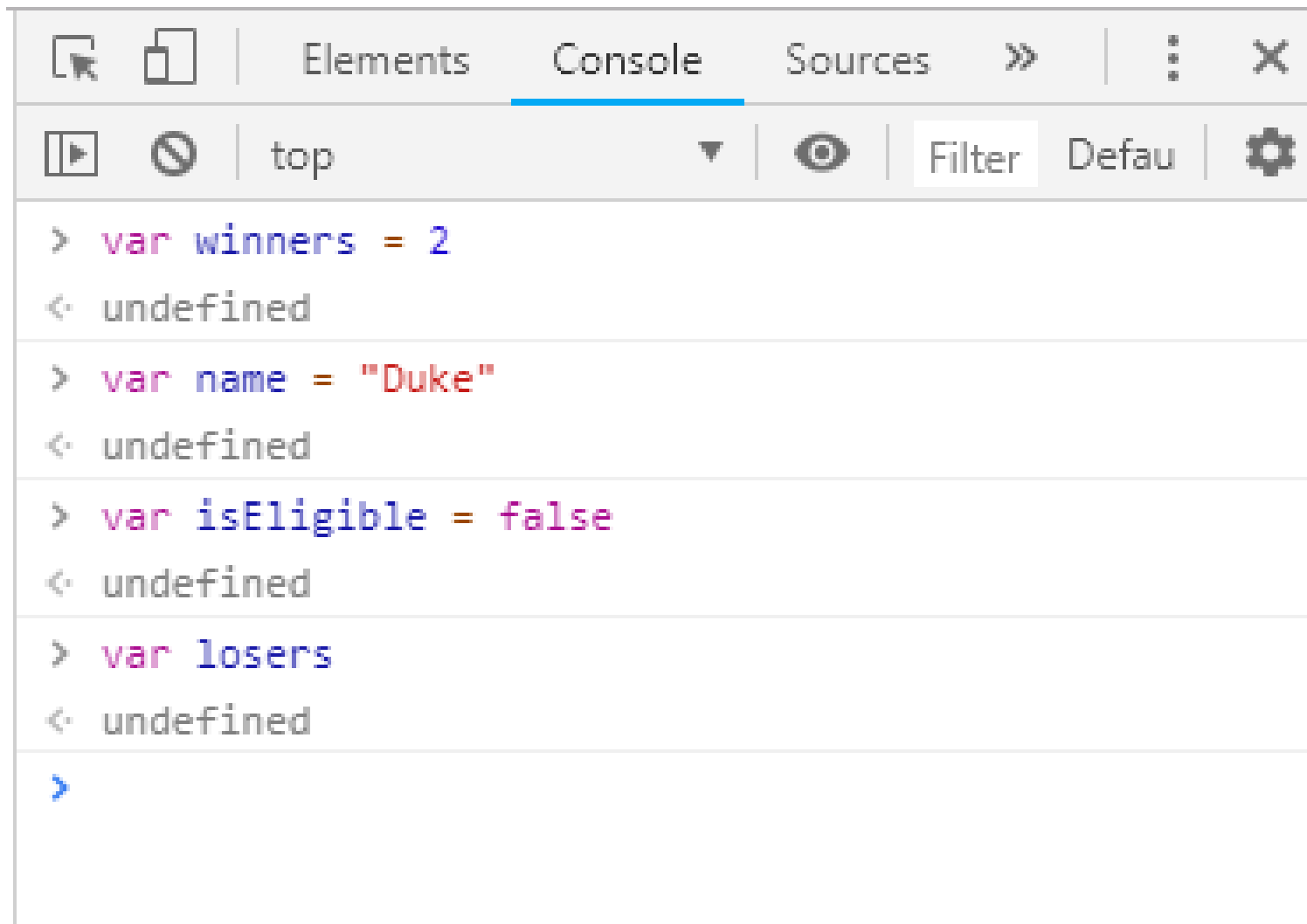
- ▶ 除了Chrome浏览器之外，还有各种网页制作软件，比如Dreamweaver



Dreamweaver

变量和值

► 声明变量(var)并赋值



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays four lines of JavaScript code, each followed by an 'undefined' return value. The code is as follows:

```
> var winners = 2  
< undefined  
  
> var name = "Duke"  
< undefined  
  
> var isEligible = false  
< undefined  
  
> var losers  
< undefined  
  
>
```

变量和值

- ▶ JS 区分大小写

- ▶ 在<script>中每条语句都以分号；结尾

```
var winners = 2;  
var name = "Duke";  
var isEligible = false;  
var losers;
```

- ▶ 单行注释以两个斜杠开头

```
// I'm a comment
```

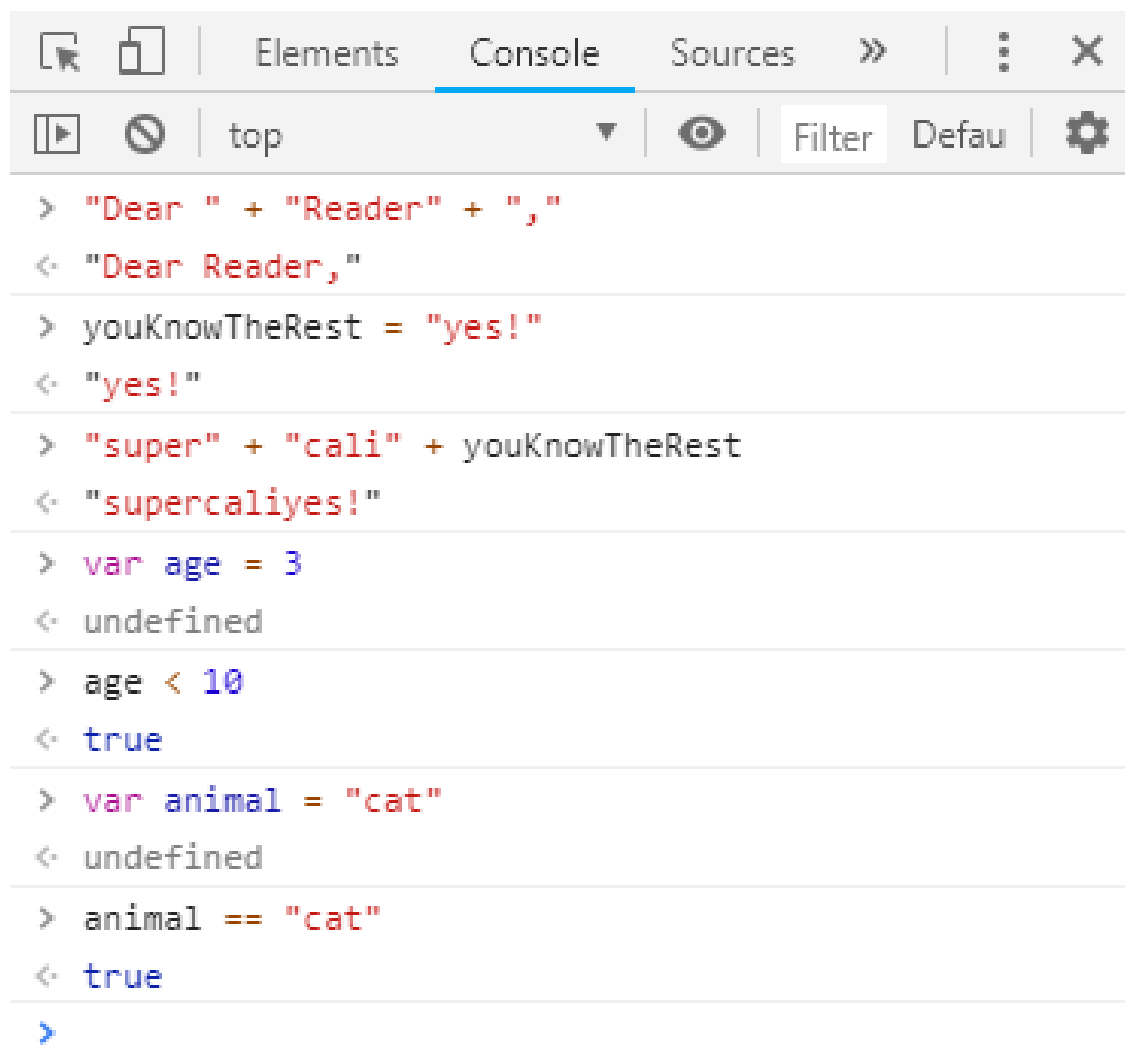
- ▶ 字符串用双引号/单引号括起

```
"You rule!"  
'And so do you!'
```

- ▶ 空格无关紧要

```
x      =      2233;
```


变量和值

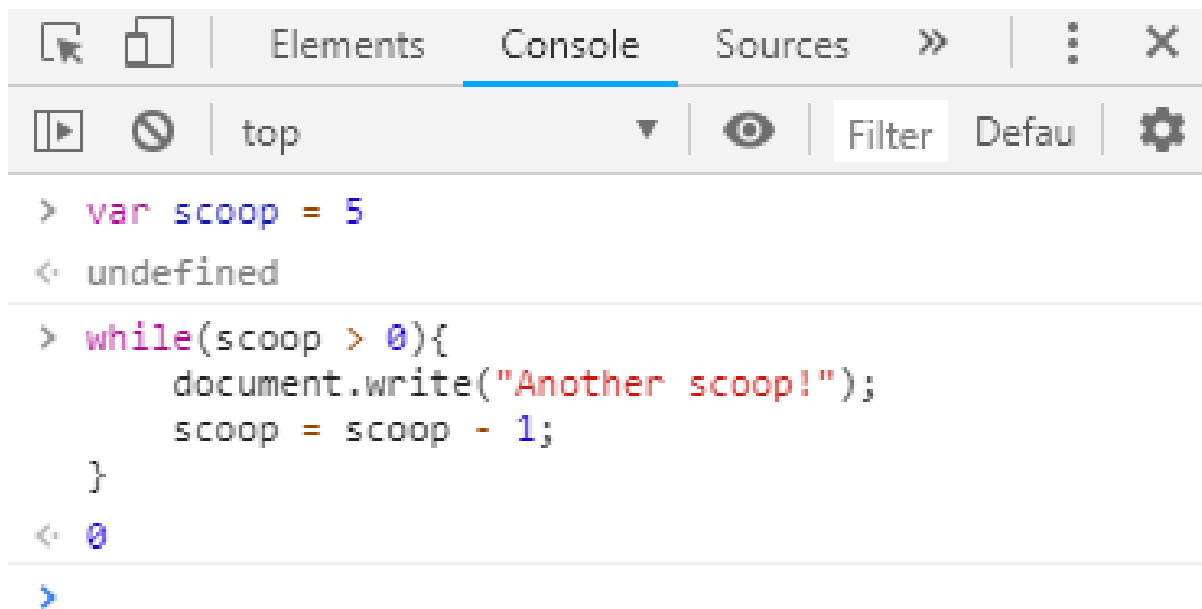


The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a series of JavaScript commands and their corresponding outputs. The commands include string concatenation, variable assignment, and logical operations. The outputs show the results of these operations, such as concatenated strings, variable values, and boolean results.

```
> "Dear " + "Reader" + ","  
< "Dear Reader,"  
  
> youKnowTheRest = "yes!"  
< "yes!"  
  
> "super" + "cali" + youKnowTheRest  
< "supercaliyes!"  
  
> var age = 3  
< undefined  
  
> age < 10  
< true  
  
> var animal = "cat"  
< undefined  
  
> animal == "cat"  
< true  
  
>
```

变量和值

- 在Console中输入下面代码，利用while循环



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following sequence of operations:

```
> var scoop = 5  
< undefined  
  
> while(scoop > 0){  
    document.write("Another scoop!");  
    scoop = scoop - 1;  
}  
< 0  
>
```

The code defines a variable `scoop` with the value 5, then enters a `while` loop that repeatedly writes "Another scoop!" to the document and decrements `scoop` until it reaches 0. The console output shows the return values of each statement: `undefined` for the variable assignment and `0` for the loop completion.

- 对应的网页就变成了

Another scoop!Another scoop!Another scoop!Another scoop!Another scoop!

变量和值

► if else

此网页显示

Ice cream is running low!

确定

```
Elements Console >>
top
> if(scoop > 5){
    alert("Eat faster");
}else{
    alert("Ice cream is running
low!");
}
```

与用户交流的方式

► 创建提醒框

- 使用函数`alert`, 并指定一个包含提醒消息的字符串, 浏览器就会在一个对话框中显示这条消息

► 直接写入文档

- 将网页视为一个文档, 使用`document.write`将任何HTML和内容写入网页

► 使用控制台

- 所有JS环境都包含控制台, 可将代码中的消息写入其中
- 要将消息写入控制台日志, 可使用函数`console.log`, 并传入要写入的字符串
- 可将`console.log`视为杰出的故障排除工具

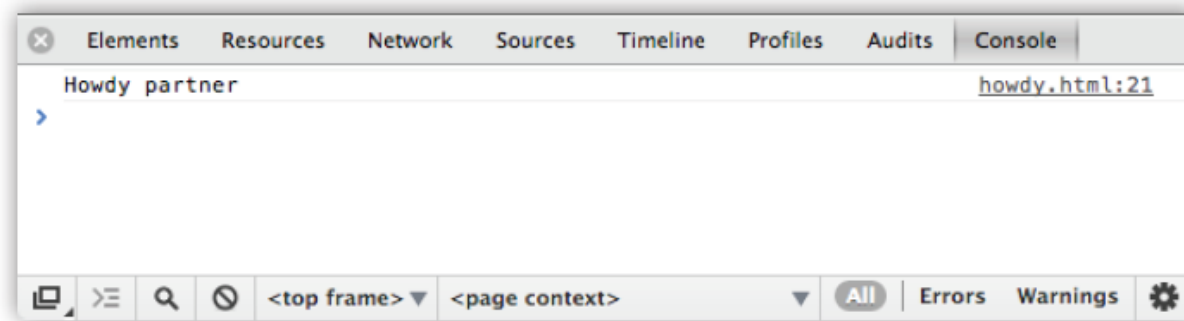
与用户交流的方式

► console.log 的使用方式

```
var message = "Howdy" + " " + "partner";  
console.log(message);
```

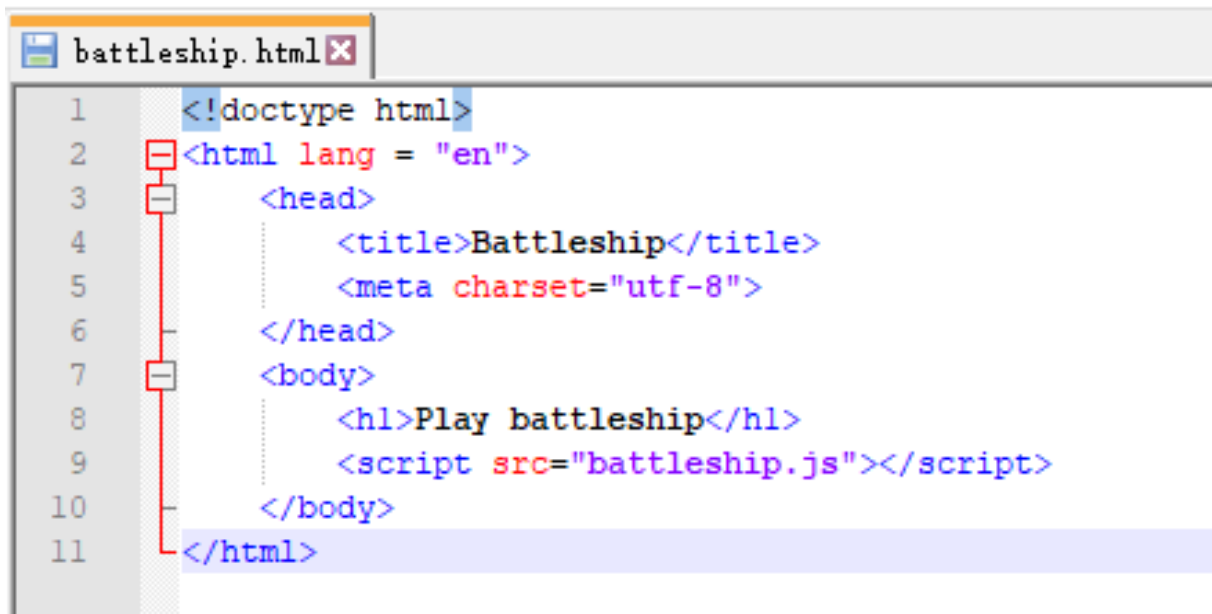
...and give it to console.log, and it will be shown in the browser's console, like this.

The console contains all the output logged by your code.



开发一款简单的战舰网页游戏

- ▶ 游戏说明，有格子0, 1, ..., 9，一艘战舰随机放置于连续的三个格子中，比如3, 4, 5，或者5, 6, 7
- ▶ 但是你不知道战舰的位置，需要不断猜测，直到将战舰的三段全部击中
- ▶ 首先，新建一个Html网页（可以通过notepad++ <https://notepad-plus-plus.org/downloads/v7.8.1/>），名字为Battleship.html



```
battleship.html
1 <!doctype html>
2 <html lang = "en">
3   <head>
4     <title>Battleship</title>
5     <meta charset="utf-8">
6   </head>
7   <body>
8     <h1>Play battleship</h1>
9     <script src="battleship.js"></script>
10  </body>
11 </html>
```

开发一款简单的战舰网页游戏



► 再书写JS代码，同样可以通过notepad++新建。

```
battleship.html x battleship.js x
1  var randomLoc = Math.floor(Math.random() * 8) //the first position of the enemy ship
2  var location1 = randomLoc;
3  var location2 = location1 + 1; //the second position of the enemy ship
4  var location3 = location2 + 1; //the third position of the enemy ship
5  var guess; // your guess
6  var hits = 0; // number of successful hits
7  var num_guess = 0; // number of your guesses
8  var isSunk = false; // whether the enemy ship sinks
9
10 while(isSunk == false){
11     guess = prompt("Ready, aim, fire! (enter a number from 0-9):");
12     if(guess < 0 || guess > 9){
13         alert("Please enter a valid cell number!");
14     }else{
15         num_guess = num_guess + 1;
16
17         if(guess == location1 || guess == location2 || guess == location3){
18             alert("HIT!");
19             hits = hits + 1;
20             if(hits == 3){
21                 isSunk = true;
22                 alert("You sank enemy battleship!");
23             }
24         }else{
25             alert("MISS");
26         }
27     }
28 }
```

prompt将用户
输入的值
传递给guess

开发一款简单的战舰网页游戏

► 打开网页，进行战斗！

 battleship	2020/5/18 17:27	HTML 文档	1 KB
 battleship	2020/5/18 17:42	JavaScript 文件	1 KB

► 进行击沉敌方军舰

此网页显示

Ready, aim, fire! (enter a number from 0-9):

此网页显示

MISS

此网页显示

Ready, aim, fire! (enter a number from 0-9):

此网页显示

Please enter a valid cell number!

开发一款简单的战舰网页游戏

► 进行击沉敌方军舰

此网页显示

Ready, aim, fire! (enter a number from 0-9):

确定

取消

此网页显示

Ready, aim, fire! (enter a number from 0-9):

确定

取消

此网页显示

Ready, aim, fire! (enter a number from 0-9):

确定

取消

此网页显示

MISS

确定

此网页显示

HIT!

确定

此网页显示

MISS

确定

开发一款简单的战舰网页游戏

► 进行击沉敌方军舰 (0, 1肯定对应战舰的前两部分)

此网页显示

Ready, aim, fire! (enter a number from 0-9):

确定 取消

此网页显示

Ready, aim, fire! (enter a number from 0-9):

确定 取消

此网页显示

HIT!

确定

此网页显示

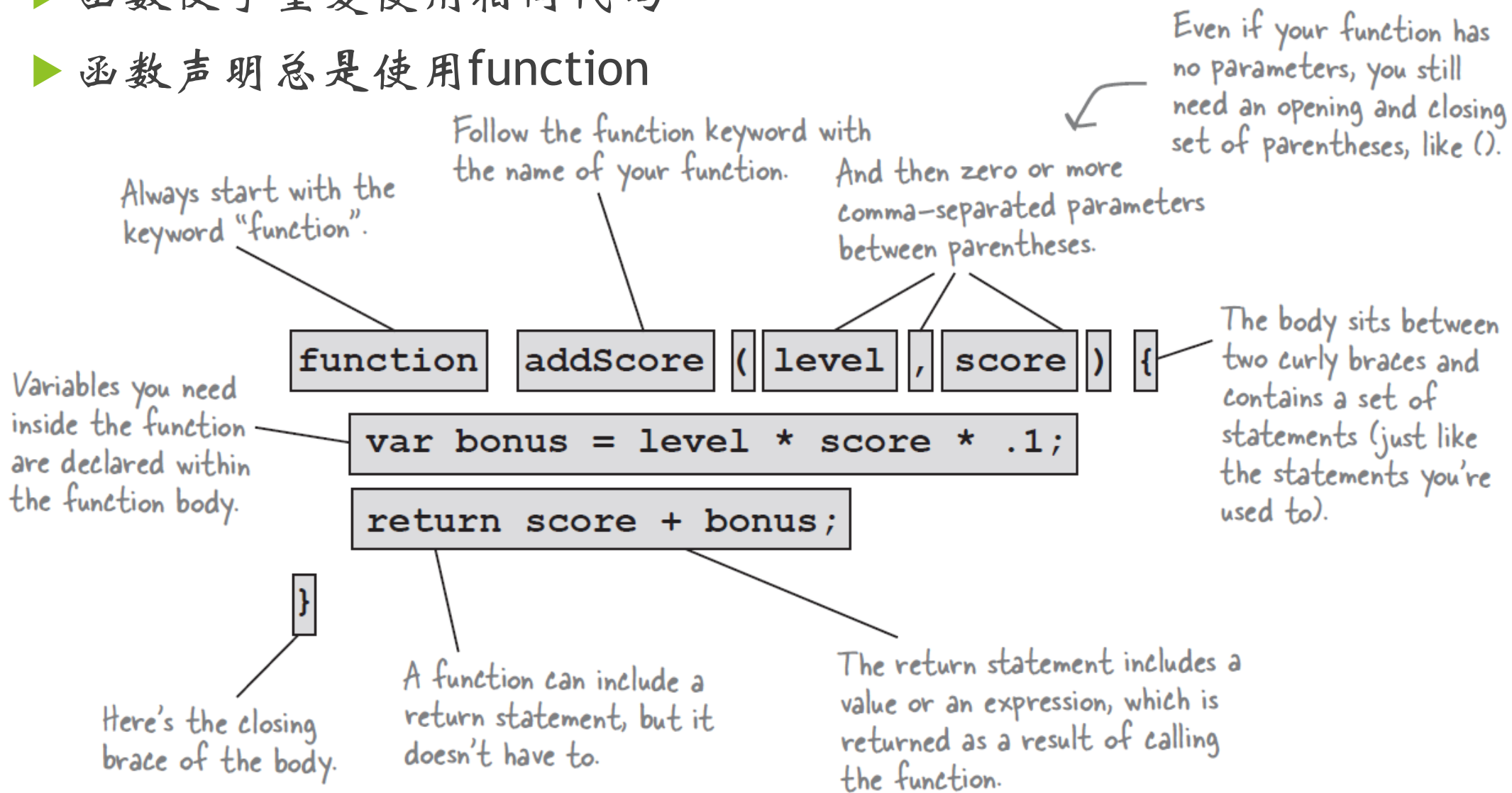
You sank enemy battleship!

确定

► 注：再次玩这个游戏敌方战舰的位置就会发生改变，因为战舰是随机放置。

函数介绍

- ▶ 函数便于重复使用相同代码
- ▶ 函数声明总是使用function



函数介绍

- ▶ 函数是按值传递的(pass-by-value)
- ▶ 函数可以没有返回值

```
function bark(name, weight) {  
  if (weight > 20) {  
    console.log(name + " says WOOF WOOF");  
  } else {  
    console.log(name + " says woof woof");  
  }  
}
```

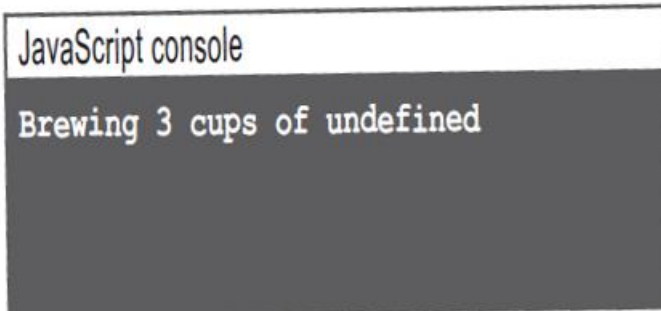
- ▶ 函数可以没有参数

```
function barkAtTheMoon() {  
  console.log("Woooooooooooooooooo!");  
}  
barkAtTheMoon();
```

函数介绍

- ▶ 传入参数缺失时，缺失参数的位置会被设置为undefined

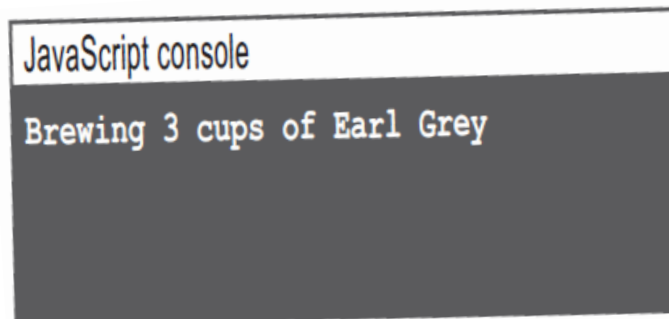
```
function makeTea(cups, tea) {  
    console.log("Brewing " + cups + " cups of " + tea);  
}  
makeTea(3);
```



JavaScript console
Brewing 3 cups of undefined

- ▶ 传入参数过多时，忽略掉多余的参数

```
function makeTea(cups, tea) {  
    console.log("Brewing " + cups + " cups of " + tea);  
}  
makeTea(3, "Earl Grey", "hey ma!", 42);
```



JavaScript console
Brewing 3 cups of Earl Grey

函数介绍

- ▶ 在函数外部声明的变量是全局变量
- ▶ 在函数中声明的变量是局部变量

```
var avatar;  
var levelThreshold = 1000;  
  
function getScore(points) {  
    var score;  
    var i = 0;  
    while (i < levelThreshold) {  
        //code here  
        i = i + 1;  
    }  
    return score;  
}
```

全局变量

局部变量

全局变量

函数介绍

- 使用未声明的变量时，它将被自动视为全局变量，即便在函数中首次使用它

```
function playTurn(player, location) {
```

```
  points = 0;
```

```
  if (location == 1) {
```

```
    points = points + 100;
```

```
  }
```

```
  return points;
```

```
}
```

We forgot to declare points with "var" before we used it. So points is automatically global.

函数介绍

- ▶ 局部变量和全局变量同名时，两者互不影响。
- ▶ 在函数中引用的都是局部变量

```
var beanCounter = 10;
```

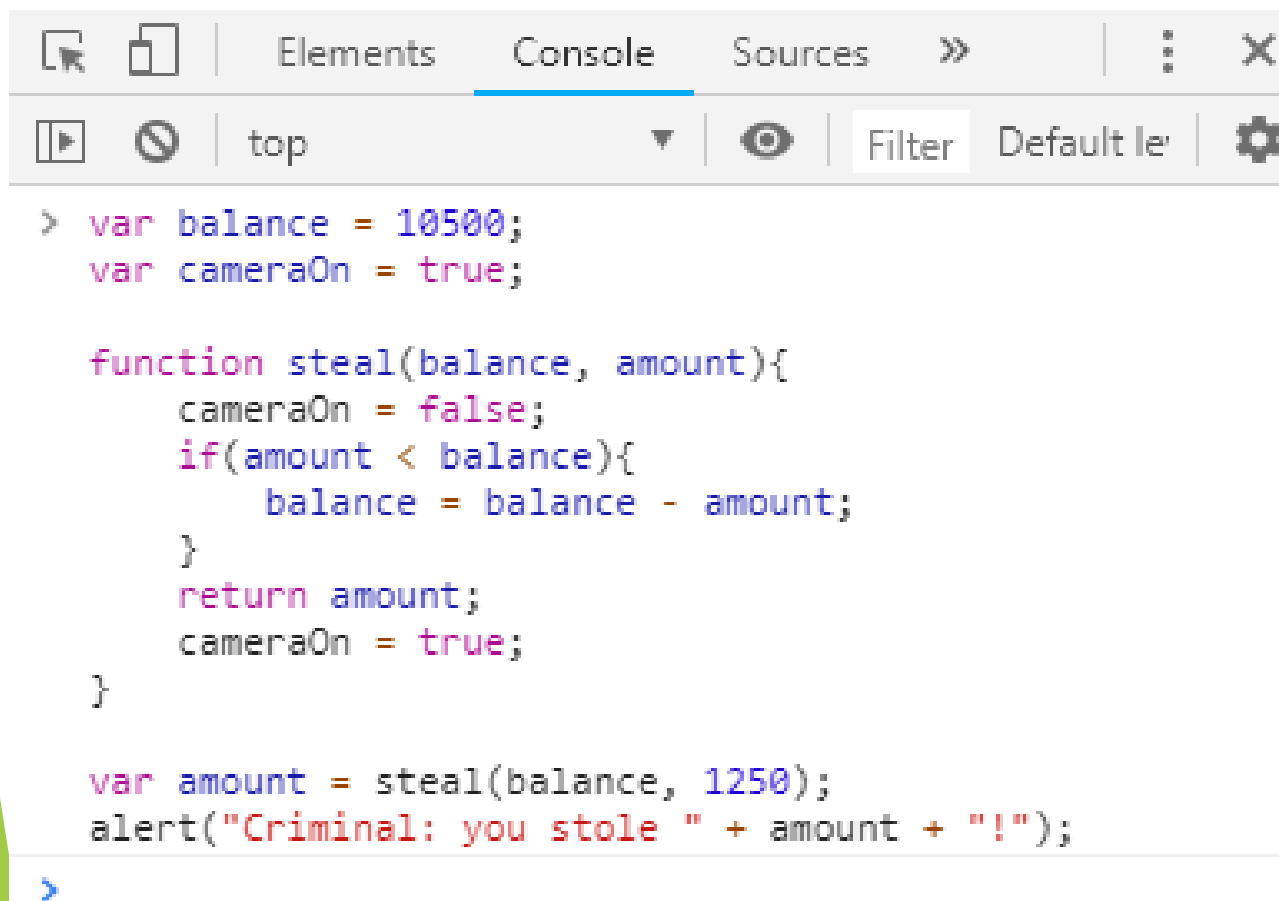
```
function getNumberOfItems(ordertype) {  
    var beanCounter = 0;  
    if (ordertype == "order") {  
        // do some stuff with beanCounter...  
    }  
    return beanCounter;  
}
```

← We've got a global
and a local!
←

例子：那个盗窃未遂案没必要调查

- ▶ 给成事不足的雷斯垂德警长打完电话，福尔摩斯在壁炉前坐下，接着看报纸。华生充满期待地看着他。
- ▶ 福尔摩斯头也不抬地说“干什么？”
- ▶ 华生问：“雷斯垂德怎么说？”
- ▶ “哦，他说他在银行账户中找到了可疑的流氓代码”
- ▶ “还有呢？”华生极力地掩饰自己的失望
- ▶ 福尔摩斯说：“雷斯垂德通过邮件把代码发给了我，我跟他这个案子不用查了。罪犯犯了致命的错误，根本不可能把钱偷走”
- ▶ “你是怎么知道的？”华生问。
- ▶ “要是你懂，这显而易见，”福尔摩斯气愤地说，“别再问我了，让我把这张报纸看完。”

例子：那个盗窃未遂案没必要调查



The image shows a web browser's developer console with the 'Console' tab selected. The code in the console is as follows:

```
> var balance = 10500;
   var cameraOn = true;

   function steal(balance, amount){
       cameraOn = false;
       if(amount < balance){
           balance = balance - amount;
       }
       return amount;
       cameraOn = true;
   }

   var amount = steal(balance, 1250);
   alert("Criminal: you stole " + amount + "!");

>
```

Below the code, an alert dialog box is visible with the text "Criminal: you stole 1250!" and a blue button labeled "确定" (Confirm).

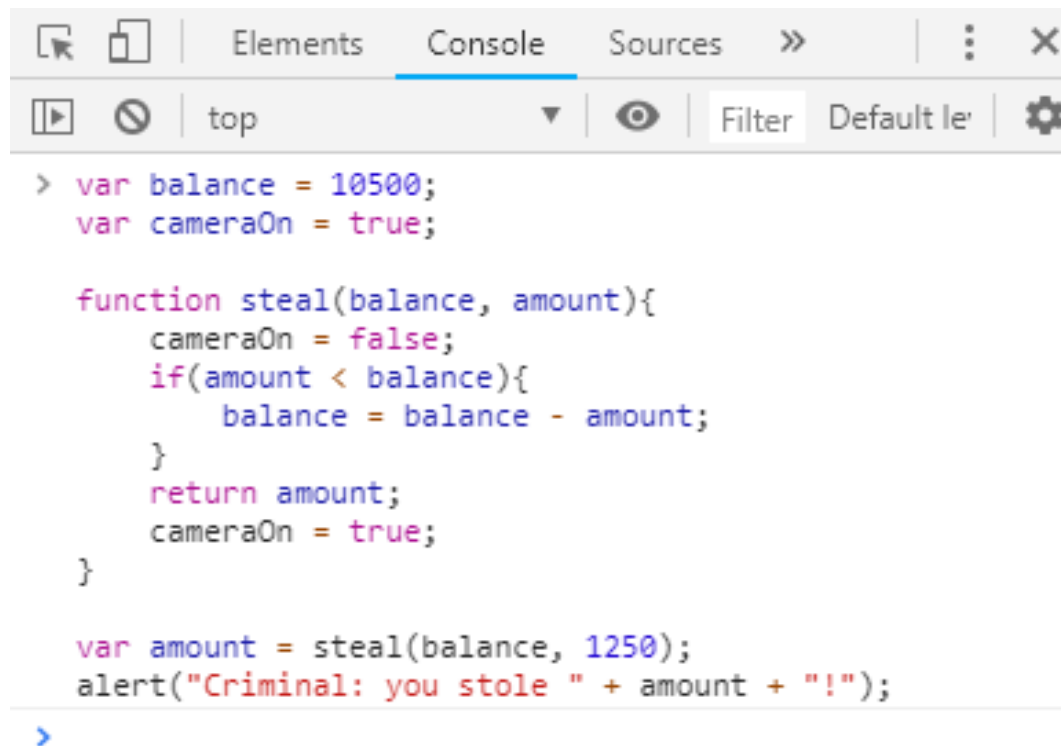
www.baidu.com 显示

Criminal: you stole 1250!

确定

例子：那个盗窃未遂案没必要调查

- ▶ balance是一个全局变量，在函数中只是传递了它的一个copy；函数中改变这个copy，并不会改变全局变量balance的值
- ▶ 罪犯不但没有得逞，还忘记了将监控重新打开，这是因为return语句后的所有代码会被忽略！



```
> var balance = 10500;
var cameraOn = true;

function steal(balance, amount){
  cameraOn = false;
  if(amount < balance){
    balance = balance - amount;
  }
  return amount;
  cameraOn = true;
}

var amount = steal(balance, 1250);
alert("Criminal: you stole " + amount + "!");
>
```

例子：那个盗窃未遂案没必要调查

► 改进版本

```
> var balance = 10500;
   var cameraOn = true;

   function steal(amount){
       cameraOn = false;
       if(amount < balance){
           balance = balance - amount;
       }
       cameraOn = true;
       return amount;
   }

   var amount = steal(1250);
   alert("Criminal: you stole " + amount + "!");
< undefined

> balance
< 9250
```

数组

- ▶ 是一种按顺序存储数据的数据结构，
- ▶ 可以存储数字、字符串、布尔值、其他数组、对象
- ▶ 包含一系列元素，每个元素都有索引
- ▶ 索引从0开始
- ▶ 内部的元素类型可以不一样

```
var flavors = ["vanilla", "butterscotch", "lavender", "chocolate", "cookie dough"];
```

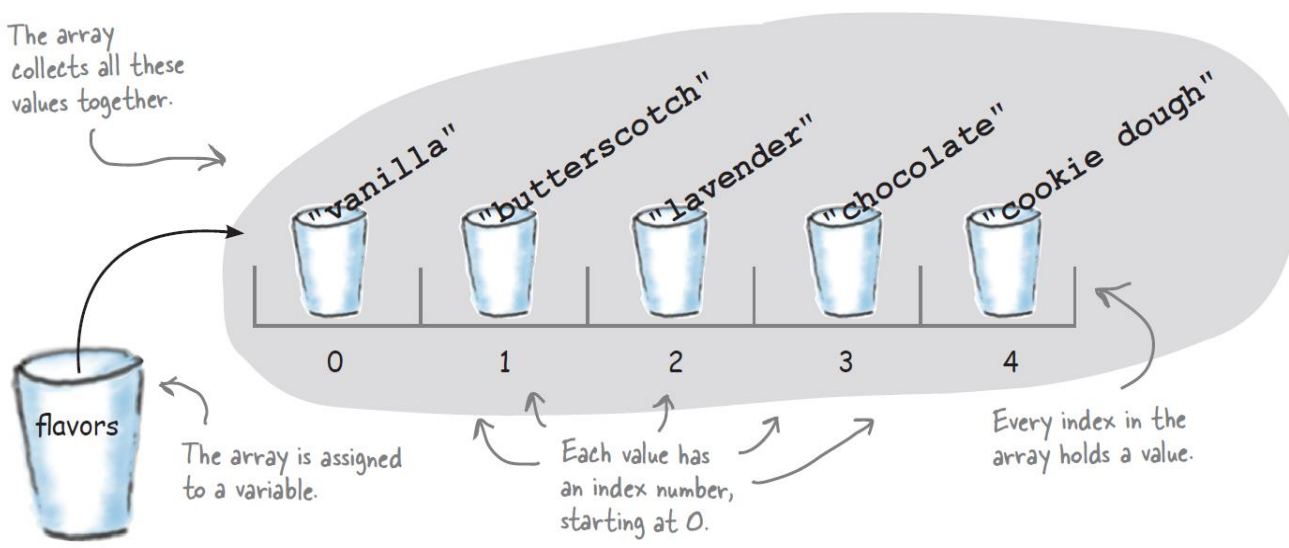
Let's assign the array to a variable named flavors.

To begin the array, use the [character...

and then list each item of the array...

... and end the array with the] character.

The array collects all these values together.



The array is assigned to a variable.

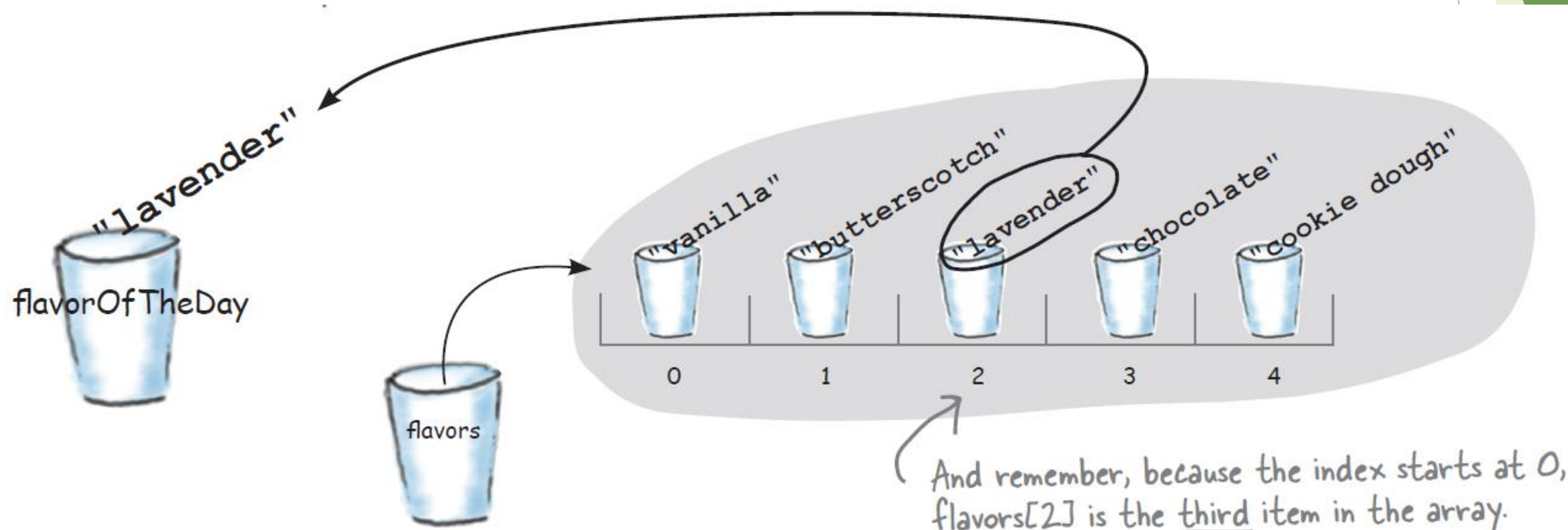
Each value has an index number, starting at 0.

Every index in the array holds a value.

数组

► 访问数组

```
var flavorOfTheDay = flavors[2];
```



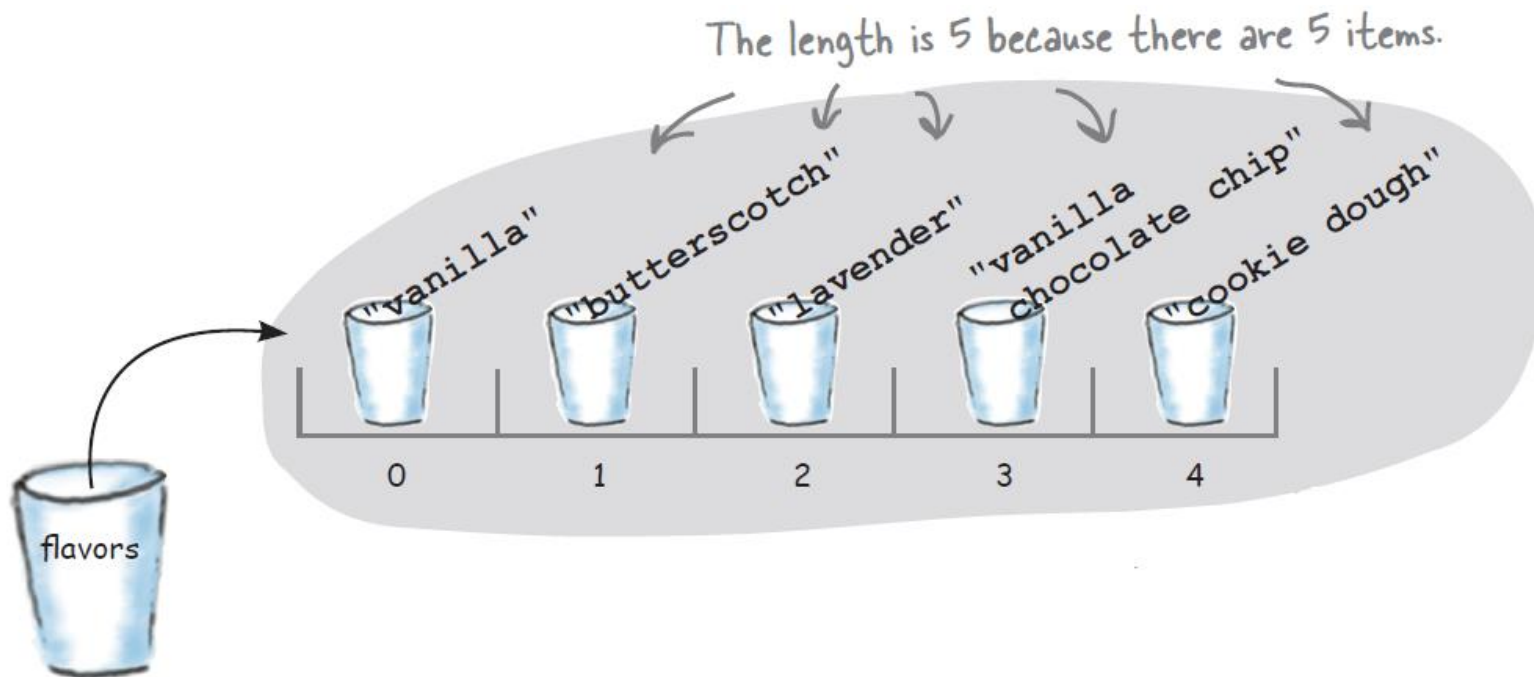
数组

► 访问数组

```
flavors[3] = "vanilla chocolate chip";
```

► 确定数组长度

```
var numFlavors = flavors.length;
```



数组

► 如何遍历数组

```
var scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69,  
              34, 55, 51, 52, 44, 51, 69, 64, 66, 55, 52, 61,  
              46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44];
```

```
var output;
```

```
Ⓐ var i = 0;           ↙ First we INITIALIZED a counter.  
  while Ⓑ i < scores.length) {   ↘ Then we tested that counter in a CONDITIONAL expression.  
    output = "Bubble solution #" + i + " score: " + scores[i];  
    console.log(output);  
    Ⓒ i = i + 1;  
  }
```

```
for (Ⓐ var i = 0; Ⓑ i < scores.length; Ⓒ i = i + 1) {  
  output = "Bubble solution #" + i + " score: " + scores[i];  
  console.log(output);  
}
```


数组

► 创建空数组并添加元素

► 方法一

```
var genres = [];  
  
genres[0] = "Rockabilly";  
genres[1] = "Ambient";
```

► 方法二

```
genres.push("Rockabilly");  
genres.push("Ambient");
```