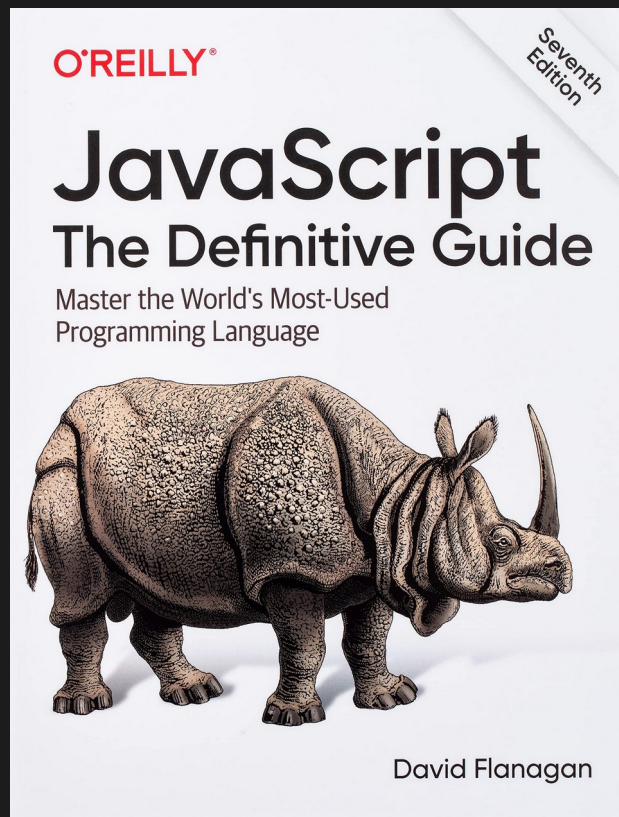


وینار کتاب خوانی JavaScript



JavaScript: The Definitive Guide, 7th Edition

By David Flanagan

جلسه اول

Email : Imanhpr1999@gmail.com

پیش گفتار

این کتاب و وبینار برای چه کسانی مناسب است؟

- برای کسانی که هیچ آشنایی با JavaScript ندارند و قصد یادگیری و آشنایی با این زبان را در پروژه های Back-End و Front-End دارند.
- برای کسانی که آشنایی قبلی با JavaScript دارند و قصد عمیق تر شدن در این زبان را دارند.
- هدف از این کتاب معرفی عمیق API موجود در جاوا اسکریپت برای برنامه نویسی Client-side و Server-side میباشد.
- پیشنهاد نویسنده بر این است از MDN و Node.js به عنوان داکيومنت استفاده شود.
- فایل های مربوط به هر جلسه را میتوانید از طریق Git دانلود کنید.
- ویدیو مربوط به هر جلسه در یوتیوب آپلود می شود.
- گاهی اوقات برای یادگیری و درک بهتر از منابع خارج از کتاب هم استفاده می شود.

سرفصل ها

- | | |
|--|---|
| ۱. مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript | ۹. کلاس‌ها - Classes |
| ۲. ساختار کلی زبان جاوا اسکریپت - Lexical Structure | ۱۰. ماژول ها - Modules |
| ۳. انواع داده و مقادیر و متغیر ها - Types, Values, and Variables | ۱۱. کتاب خانه‌های استاندارد جاوا اسکریپت - The JavaScript Standard Library |
| ۴. عبارات و عملگر ها - Expressions and Operators | ۱۲. Iterators and Generators |
| ۵. Statements | ۱۳. Asynchronous JavaScript |
| ۶. اشیاء - Objects | ۱۴. Metaprogramming |
| ۷. آرایه ها - Arrays | ۱۵. جاوا اسکریپت در محیط مرورگر ها - JavaScript in Web Browsers (Front-End) |
| ۸. توابع - Functions | ۱۶. برنامه نویسی سمت سرور - Server-Side JavaScript with Node |
| | ۱۷. JavaScript Tools and Extensions |

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- چرا Js یاد بگیریم ؟

(۱) جاوا اسکریپت زبان اصلی وب میباشد. بخش زیادی از وب سایت‌ها و تمام Browser های مدرن از Js پشتیبانی میکنند و این موضوع باعث شده Js تبدیل به پر استفاده ترین زبان برنامه نویسی تاریخ شود.

(۲) در ۱۰ سال گذشته Node.js باعث شده که شاهد استفاده Js برای برنامه نویسی Server-side هم باشیم.

- تعریف زبان Js (کتاب) :

JavaScript is a **high-level**, **dynamic**, **interpreted** programming language that is well-suited to **object-oriented** and **functional** programming styles.

- تعریف زبان Js (ویکی پدیا) :

JavaScript is a **high-level**, often **just-in-time compiled** language that conforms to the ECMAScript standard. It has **dynamic typing**, **prototype-based object-orientation**, and **first-class** functions. It is **multi-paradigm**, supporting **event-driven**, **functional**, and **imperative** programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM)

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- تاریخچه کوتاهی درمورد JS :

(۱) جاوا اسکریپت در سال‌های اولیه اینترنت برای مرورگر **Netscape** طراحی شده بود.

(۲) جاوا اسکریپت در حقیقت نام مرسوم و رایج برای پیاده‌سازی **ECMA Script** میباشد.

(۳) در این کتاب از مخفف **ES** برای اشاره به استاندارد های مختلف JS دارد به عنوان مثال ES5 و ES6

(۴) در این کتاب ما به بررسی جاوا اسکریپت ES5 و ورژن های بعدتر میپردازیم.

(۵) با انتشار **ES6** در سال 2015 با توجه به تغییرات زیادی که در زبان حاصل شد، این زبان از یک **scripting language** ساده تبدیل به یک زبان **general-purpose** قدرتمند برای پروژه های مقیاس پذیر شد.

(۶) بعد از انتشار **ES6** جاوا اسکریپت وارد چرخه انتشار سالانه شد و ما ورژن های جدید JS رو با نام های ES2016, ES2017, ES2018 و ... میشناسیم.

(۷) با تکامل زبان JS دیزاینر های این زبان تصمیم گرفتند که برخی از ضعف های زبان JS که قبل از ES5 وجود داشتند را فیکس کنند ولی با این حال قصد داشتند **Backward compatibility** را هم حفظ کنند و این کار بدون حذف کردن ویژگی‌های قدیمی غیرممکن بود. (با اضافه شدن **strict mode** برخی از این ضعف‌ها را بهبود بخشیدند)

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- انواع پلتفرم ها و هسته JS :

برای مفید بودن هر زبان برنامه نویسی نیاز به Standard Library یا پلتفرمی برای اعمال پایه‌ای مثل **I/O** دارد. با این حال در هسته اصلی جاوا اسکریپت برای کار با اعداد و متن ها الگوریتم هایی و ساختمان داده‌های مثل **arrays, sets, maps** و ... پیاده‌سازی شده‌اند اما نکته قابل توجه این است که هسته اصلی JS عملیات **I/O** را برای ما پیاده‌سازی نکرده و پیاده‌سازی این API ها به عهده **host environment** ها میباشد.

برای مثال **host environment** اورجینال جاوا اسکریپت مرورگر ها میباشد که باعث می‌شوند. مرورگر ها به عنوان مثال به جاوا اسکریپت اجازه میدهند تا از موس و کیبورد کاربر ورودی دریافت کند و در مرورگر با استفاده از کد های HTML و CSS به کاربر نمایش دهد.

و از سال 2010 Node.js به عنوان یک **host environment** دیگر در دسترس ما وجود دارد که به ما این قابلیت را میدهد که بجای استفاده از API هایی که Browser به ما میدهد با OS در ارتباط باشیم و کار هایی مثل نوشتن و خواندن فایل یا ایجاد کردن سرور HTTP بپردازیم.

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- رویکرد کلی کتاب در یادگیری مفاهیم زبان Js.

در این کتاب ما از مفاهیم پایه‌ای Low-level شروع میکنیم و بعد از ساخت پایه قوی و مناسب شروع به صحبت درمورد مفاهیم higher-level تر میکنیم.

و توجه داشته باشید که یادگیری یک زبان برنامه نویسی جدید به هیچ عنوان یک مسیر خطی نیست و ما گاهی مجبوریم در زمان سفر کنیم و به برخی از مفاهیم از قبل یا بعد رجوع کنیم.

در فصل اول کتاب ما یک نگاه کلی و سریع به برخی از ویژگی‌های کلی زبان Js میپردازیم تا در آینده درک این مفاهیم برای ما آسان‌تر باشد.

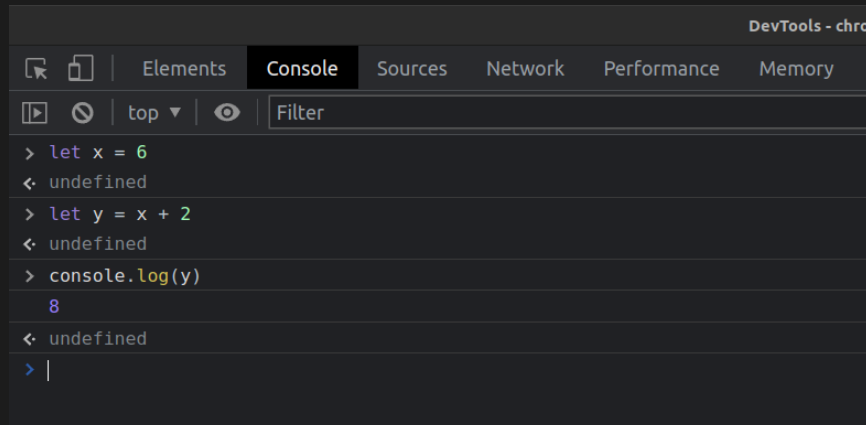
برای یادگیری یک زبان برنامه نویسی جدید قطعاً شما به یک محیط نیاز دارید تا مثال‌های منبع خود را تایپ کنید و تست کنید و برای Js شما نیاز به یک Interpreter دارید.

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

Exploring JavaScript 1.1

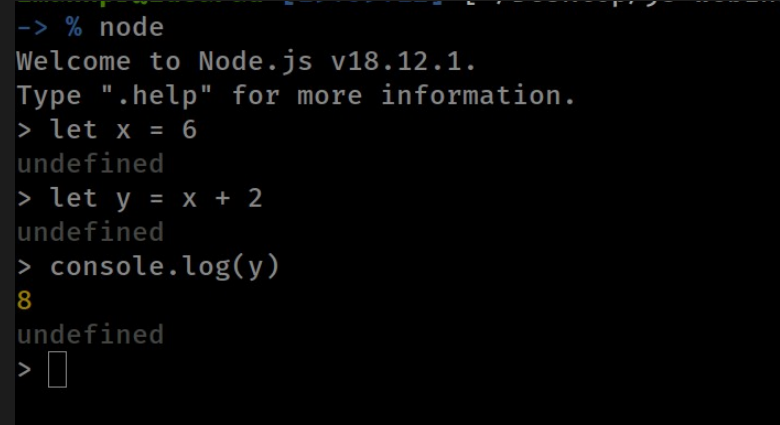
- راحت ترین روش برای تست کد های Js نوشتن کد در محیط Browser میباشد که شما میتوانید با استفاده از کلید های میانبر F12 , Ctrl-Shift-I و سپس Console Tab را انتخاب کنید.
- روش بعدی که برای تست کد های Js در دسترس میباشد استفاده از Node.js میباشد و برای این کار شما نیاز به نصب Node.js بر روی سیستم خودتون دارید.



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The following code has been executed:

```
> let x = 6
< undefined
> let y = x + 2
< undefined
> console.log(y)
8
< undefined
> |
```

Google Chrome



The screenshot shows a terminal window with the following text:

```
-> % node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
> let x = 6
undefined
> let y = x + 2
undefined
> console.log(y)
8
undefined
> |
```

Node.js

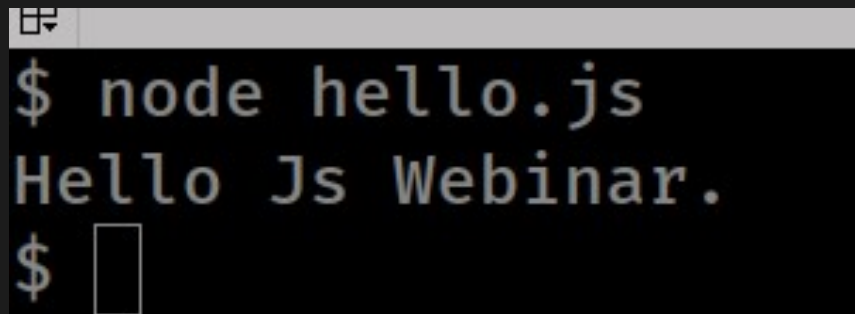
فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

1.2 Hello World

- وقتی که حاضر بودیم تکه کد های طولانی‌تر و پیچیده‌تر بنویسیم منطقی نیست که از محیط interactive استفاده کنیم و بهتر است از یک Code Editor استفاده کنیم و بعد از نوشتن کد برای اجرا میتوانیم کد را در محیط node کپی پیست کنیم یا در یک فایل با فرمت Js ذخیره کنیم و سپس با استفاده از Node اجرا کنیم.
- توجه اگر از Node به صورت Noninteractive استفاده کنید نتیجه کد ها به صورت اتوماتیک به نمایش در نمی آید و شما باید با استفاده از فانکشن console.log() نتیجه را در کنسول Browser و یا در Node نمایش دهید.

```
// hello.js  
console.log("Hello Js Webinar.");
```

A screenshot of a terminal window with a dark background. The prompt is a dollar sign '\$'. The command 'node hello.js' has been entered and executed. The output 'Hello Js Webinar.' is displayed on the next line. The prompt '\$' is shown again on the third line, followed by a small white rectangular cursor box.

```
$ node hello.js  
Hello Js Webinar.  
$
```

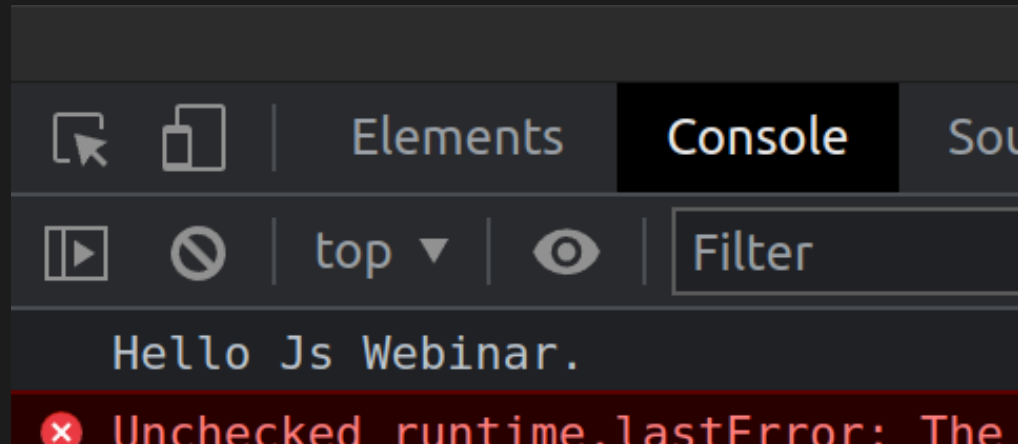
فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

1.2 Hello World

- در صورتی که دوست دارید نتیجه برنامه نوشته شده را در محیط Browser هم ببینید ابتدا یک فایل HTML ایجاد کرده و سپس کد زیر را در این فایل قرار دهید و بعد از باز کردن فایل HTML ایجاد شده در Browser وارد محیط Console شوید و خروجی دستور را مشاهده کنید

```
←!— index.html →  
<script src="hello.js"></script>
```



فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

1.3 A Tour of JavaScript

- در این بخش نگاهی سریع به جاوا اسکریپت با چند مثال میکنیم.

// Anything following double slashes is an English-language comment.

// Read the comments carefully: they explain the JavaScript code.

// A variable is a symbolic name for a value.

// Variables are declared with the let keyword:

let x;

// Declare a variable named x.

// Values can be assigned to variables with an = sign

x = 0;

// Now the variable x has the value 0

x // => 0: A variable evaluates to its value.

// JavaScript supports several types of values

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

// JavaScript supports several types of values

x = 1;

// Numbers.

x = 0.01;

// Numbers can be integers or reals.

x = "hello world";

// Strings of text in quotation marks.

x = 'JavaScript';

// Single quote marks also delimit strings.

x = true;

// A Boolean value.

x = false;

// The other Boolean value.

x = null;

// Null is a special value that means "no value."

x = undefined;

// Undefined is another special value like null.

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- دو **DataType** خیلی مهم در جاوا اسکریپت **Objects** و **Arrays** میباشد. این دو موضوعات درس ما در فصل 6 و 7 میباشد. اما با توجه به مهم بودن این دو شما بارها این دو را در فصل های قبل مشاهده خواهید کرد

```
// JavaScript's most important datatype is the object.
```

```
// An object is a collection of name/value pairs, or a string to value map.
```

```
let book = { // Objects are enclosed in curly braces.  
  topic: "JavaScript", // The property "topic" has value "JavaScript."  
  Edition: 7 // The property "edition" has value 7  
};
```

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

// Access the properties of an object with . or []:

book.topic // => "JavaScript"

book["edition"] // => 7: another way to access property values.

book.author = "Flanagan"; // Create new properties by assignment.

book.contents = {}; // {} is an empty object with no properties.

// Conditionally access properties with ?. (ES2020):

book.contents?.ch01?.sect1

// => undefined: book.contents has no ch01 property.

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

// JavaScript also supports arrays (numerically indexed lists) of values:

```
let primes = [2, 3, 5, 7];           // An array of 4 values, delimited with [ and ].
primes[0]                             // => 2: the first element (index 0) of the array.
primes.length                         // => 4: how many elements in the array.
primes[primes.length-1]              // => 7: the last element of the array.
Primes[4] = 9;                       // Add a new element by assignment.
Primes[4] = 11;                      // Or alter an existing element by assignment.
let empty = [];                     // [] is an empty array with no elements.
empty.length                         // => 0
```

// Arrays and objects can hold other arrays and objects:

```
let points = [                       // An array with 2 elements.
  {x: 0, y: 0},                     // Each element is an object.
  {x: 1, y: 1}
];
let data = {                         // An object with 2 properties
  trial1: [[1,2], [3,4]],           // The value of each property is an array
  trial2: [[2,3], [4,5]]           // The elements of the arrays are arrays.
};
```

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- An **expression** is a phrase of JavaScript that can be **evaluated** to produce a value
- For example, the use of **.** And **[]** to refer to the value of an object property or array element is an **expression**

// Operators act on values (the operands) to produce a new value.

// Arithmetic operators are some of the simplest:

3 + 2

// => 5: addition

3 - 2

// => 1: subtraction

3 * 2

// => 6: multiplication

3 / 2

// => 1.5: division

points[1].x - points[0].x

// => 1: more complicated operands also work

"3" + "2"

// => "32": + adds numbers, concatenates strings

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

// JavaScript defines some shorthand arithmetic operators

let count = 0;

count++;

count--;

count += 2;

count *= 3;

count

// Define a variable

// Increment the variable

// Decrement the variable

// Add 2: same as count = count + 2;

// Multiply by 3: same as count = count * 3;

// => 6: variable names are expressions, too.

// Equality and relational operators test whether two values are equal,
// unequal, less than, greater than, and so on. They evaluate to true or false.

let x = 2, y = 3;

x === y

x !== y

x < y

x <= y

x > y

x >= y

"two" === "three"

"two" > "three"

false === (x > y)

// These = signs are assignment, not equality tests

// => false: equality

// => true: inequality

// => true: less-than

// => true: less-than or equal

// => false: greater-than

// => false: greater-than or equal

// => false: the two strings are different

// => true: "tw" is alphabetically greater than "th"

// => true: false is equal to false

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

// Logical operators combine or invert boolean values

(x === 2) && (y === 3) // => true: both comparisons are true. && is AND

(x > 3) || (y < 3) // => false: neither comparison is true. || is OR

!(x === y) // => true: ! inverts a boolean value

- If JavaScript **expressions** are like phrases, then JavaScript **statements** are like full sentences.
- **Expression** is something that computes a value but doesn't do anything.
- **Statements**, on the other hand, don't have a value, but they do alter the state.

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- A **function** is a named and **parameterized** block of JavaScript code that you define once, and can then **invoke over and over again**.

```
// Functions are parameterized blocks of JavaScript code that we can invoke.  
function plus1(x) {  
    return x + 1;  
}  
plus1(y)  
let square = function(x) {  
    return x * x;  
};  
square(plus1(y))
```

// Define a function named "plus1" with parameter "x"
// Return a value one larger than the value passed in
// Functions are enclosed in curly braces
// => 4: y is 3, so this invocation returns 3+1
// Functions are values and can be assigned to vars
// Compute the function's value
// Semicolon marks the end of the assignment.
// => 16: invoke two functions in one expression

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- In **ES6** and later, there is a shorthand syntax for defining functions. This concise syntax uses **=>** to separate the argument list from the function body, so functions defined this way are known as **arrow functions**.

```
const plus1 = x => x + 1;    // The input x maps to the output x + 1
const square = x => x * x;    // The input x maps to the output x * x
plus1(y)                     // => 4: function invocation is the same
square(plus1(y))              // => 16
```

```
// When functions are assigned to the properties of an object, we call
// them "methods." All JavaScript objects (including arrays) have methods:
let a = [];                  // Create an empty array
a.push(1,2,3);                // The push() method adds elements to an array
a.reverse();                  // Another method: reverse the order of elements
```

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

- JavaScript supports an **object-oriented** programming style, but it is significantly different than “**classical**” object-oriented programming languages

```
class Point {  
    constructor(x, y) {  
        this.x = x;  
        this.y = y;  
    }  
    Distance() {  
        return Math.sqrt(  
            this.x * this.x +  
            this.y * this.y  
        );  
    }  
}
```

```
// By convention, class names are capitalized.  
// Constructor function to initialize new instances.  
// This keyword is the new object being initialized.  
// Store function arguments as object properties.  
// No return is necessary in constructor functions.  
// Method to compute distance from origin to point.  
// Return the square root of  $x^2 + y^2$ .  
// this refers to the Point object on which  
// the distance method is invoked.
```

```
// Use the Point() constructor function with "new" to create Point objects  
let p = new Point(1, 1); // The geometric point (1,1).
```

```
// Now use a method of the Point object p  
p.distance() // => Math.SQRT2
```

فصل اول

مقدمه‌ای برا جاوا اسکریپت - Introduction to JavaScript

تمرین کلاسی :

سعی کنید بخش 1.4 Example: Character Frequency Histograms را به صورت خودخوان مطالعه کنید.

(به گفته نویسنده در تکه کد مثال این بخش از مباحث پیشرفته JS استفاده شده و از شما انتظاری نمیرود که کامل متوجه این مفاهیم شوید.)

The End