مروری بر مطالب گفته شده

در این بخش مروری خواهیم داشت بر مطالبی از جاوا اسکریپت که تا کنون آموختیم و نکات مهم را یادآوری خواهیم کرد .

ساختار کد

در انتهای جملات سمی کالن قرار می گیرد :

```
alert('Hello'); alert('World');
```

در اکثر اوقات ، نوشتن جملات در خطوط متعدد باعث قرار گیری ضمنی سمی‌کالن در انتهای هر خط می‌شود :

```
alert('Hello')
alert('World')
```

به این عمل ، automatic semicolon insertion می گویند . گاهی اوقات این فرآیند به درستی کار نمی‌کند :

```
alert("There will be an error after this message")
[1, 2].forEach(alert)
```

در اکثر راهنماهای جاوا اسکریپت ، پیشنهاد می‌شود که در انتهای خطوط از سمی‌کالن استفاده نمایید .

در انتهای بلاک های کد و ساختارهایی مانند حلقه ها ، نیازی به استفاده از سمی‌کالن نیست :

```
function f() {
  // no semicolon needed after function declaration
}
for(;;) {
  // no semicolon needed after the loop
}
```


. . . با این حال اگر در انتهای این ساختارها ، سمی‌کالن بگذاریم ، به خطا برنخورده و صرفا نادیده گرفته خواهند شد .

حالت سختگیرانه یا مدرن (Strict mode)

برای فعال سازی کامل ویژگی های مدرن جاوا اسکریپت ، باید در ابتدای اسکریپت یا ابتدای تعریف فانکشن و قبل از سایر خطوط از "use strict" استفاده نماییم .

بدون استفاده از این عبارت نیز اسکریپت ما همچنان کار خواهد کرد ولی ویژگی های آن رفتار قدیمی خواهند داشت. پیشنهاد ما این است که همیشه از use strict استفاده نمایید.

بعضی از ویژگی های مدرن مانند کلاس ها (در آموزش های بعد توضیح داده خواهد شد) ، به صورت ضمنی (خودکار) ، حالت مدرن را فعال می کنند.

متغیرها

با استفاده از کلمات زیر می‌توان متغیرها را تعریف کرد:

- let
- const : برای تعریف ثوابت به کار می‌رود.
- var : روش قدیمی تعریف متغیر

در نام یک متغیر تنها می‌توانید از موارد زیر استفاده نمایید:

- حروف و اعداد. ولی نام متغیر نباید با عدد شروع شود.
- کاراکترهای $ و _
- الفبا و علامت های غیر لاتین مجاز بوده ولی توصیه می‌شود که از آنها استفاده ننمایید.

در جاوا اسکریپت ، متغیرها داینامیک بوده و هر نوع داده‌ای را می‌توانند نگهداری کنند :

```
let x = 5;
x = "John";
```

در جاوا اسکریپت ۸ نوع داده وجود دارد :

- number : برای اعداد صحیح و اعشاری
- bigint : برای اعداد صحیح با طول دلخواه
- string : برای رشته ها
- boolean : برای مقادیر منطقی true/false
- null : نوع داده ای که فقط شامل null است و معنای آن "خالی" یا "عدم وجود" است.
- undefined : نوع داده ای که فقط شامل undefined است و معنای آن "مقدار دهی نشده" است.
- object و symbol : برای ساختارهای پیچیده و unique identifier ها که هنوز آنها را مطالعه نکرده ایم.

عملگر typeof نوع مقدار داده شده را بر می‌گرداند و به دو شکل می‌توان از آن استفاده نمود :

```
typeof null == "object" // error in the language

typeof function(){} == "function" // functions are treated specially
```

تعامل با کاربر

هنگامی که از جاوا اسکریپت در مرورگر استفاده می‌کنیم ، بوسیله فانکشن‌های زیر می‌توانیم با کاربر تعامل داشته باشیم :

- prompt(question, [default]) : سوالی پرسیده و اگر کاربر جوابی داده باشد آن را برگردانده و یا اگر کاربر دکمه cancel را زده باشد null برمی‌گرداند .
- confirm(question) : سوالی از کاربر پرسیده و دو گزینه در اختیار وی قرار می‌دهد . جواب کاربر با مقادیر true/false برگردانده می‌شود .
- alert(message) : پیغامی را نمایش می‌دهد .

همه فانکشن های فوق modal هستند و این یعنی اینکه آنها ،روند اجرا را متوقف کرده و منتظر پاسخ از سمت کاربر می‌مانند . مثال :

```
let userName = prompt("Your name?", "Alice");

let isTeaWanted = confirm("Do you want some tea?");

alert( "Visitor: " + userName ); // Alice

alert( "Tea wanted: " + isTeaWanted ); // true
```

عملگرها

جاوا اسکریپت از عملگرهای زیر پشتیبانی می‌کند :

عملگرهای حسابی

عملگرهای معمولی + ، - ، * و / . همچنین از % برای باقی‌مانده و از ** برای توان استفاده می‌کند .

عملگر دوتایی + می‌تواند رشته ها را نیز تجمیع کند و اگر یکی از عملوندها رشته باشد ، عملوند بعدی را به رشته تبدیل می‌کند :

```
alert( '1' + 2 ); // '12', string

alert( 1 + '2' ); // '12', string
```

عملگر انتساب

روش آن به شکل a=b یا در ترکیب با سایر عملگرها به شکل a*=2 است .

عملگر بیتی

این عملگرها در پایین‌ترین سطح و در سطح بیت با اعداد صحیح 32 بیتی کار می‌کنند .

عملگر سه تایی

به شکل cond ? resultA : resultB نوشته می شود.

عملگرهای منطقی

عملگر منطقی AND (&&) و عملگر منطقی OR (||) به شکل اتصال کوتاه کار می‌کنند و هرجا که متوقف شوند مقدار محاسبه شده را برمی‌گردانند .

عملگر منطقی NOT(!) ، عملوند خود را به مقدار boolean تبدیل کرده و عکس آن را برمی‌گردانند .

عمگر Nullish

عملگر ?? راهی برای تشخیص متغیرهای مقدار دهی شده از لیستی از متغیرهاست . نتیجه عبارت a??b مقدار a خواهد شد مگراینکه مقدار آن null یا undefined باشد که در این صورت مقدار b برگردانده می‌شود .

عملگرهای مقایسه ای

مقایسه برابری داده های از نوع متفاوت باعث تبدیل آنها به عدد می‌شود (به جز null و undefined که فقط با یکدیگر برابر هستند . بنابراین عبارات زیر برابر هستند :

```
alert( 0 == false ); // true
alert( 0 == '' ); // true
```

باقی مقایسه ها نیز به عدد تبدیل می شوند .

عملگر === این عمل تبدیل کردن را انجام نمی‌دهد . بوسیله این عملگر ، داده های از نوع مختلف با یکدیگر متفاوت اند حتی اگر مقدار یکسانی داشته باشند .

عملگرهای بزرگتر و کوچکتر ، رشته ها را کاراکتر به کاراکتر مقایسه می‌کنند . باقی نوع ها به عدد تبدیل می‌شوند .

باقی عملگرها

علاوه بر عملگرهای فوق ، عملکرهای دیگری نیز وجود دارند . مانند عملگر کاما .

حلقه ها (loops)

در مورد سه نوع حلقه صحبت کردیم :

```
// 1
while (condition) {}
// 2
do {} while (condition);
// 3
for(let i = 0; i < 10; i++) {}
```


- متغیرهایی که داخل حلقه تعریف می‌شود ، فقط در داخل حلقه قابل دسترس است . همچنین می توانیم از let استفاده نکرده و از متغییری که قبلا تعریف شده بود ، استفاده کنیم .
- از کلمات break یا loop برای خروج از حلقه یا iteration جاری استفاده می‌کنیم . از label برای خروج از حلقه های تودرتو استفاده می‌کنیم .

Switch

ساختار switch می‌تواند جایگزین چندین if شود و از عملگر === برای مقایسه استفاده می‌کند. مثال :

```
let age = prompt('Your age?', 18);

switch (age) {
  case 18:
    alert("Won't work"); // the result of prompt is a string, not a number
    break;
  case "18":
    alert("This works!");
    break;
  default:
    alert("Any value not equal to one above");
}
```


فانکشن ها

گفتیم که در جاوا اسکریپت به سه روش می‌توان فانکشن ایجاد نمود :

۱. Function Declaration : فانکشن در بدنه اصلی اسکریپت نوشته می‌شود.

```
function sum(a, b) {
  let result = a + b;
  return result;
}
```

۲. Function Expression : فانکشن در متن یک عبارت نوشته می‌شود.

```
let sum = function(a, b) {
  let result = a + b;
  return result;
};
```

۳. Arrow Function :

```
// expression at the right side
let sum = (a, b) => a + b;
// or multi-line syntax with { ... }, need return here:
let sum = (a, b) => {
  // ...
  return a + b;
}
// without arguments
```

```
let sayHi = () => alert("Hello");
// with a single argument
let double = n => n * 2;
```

- فانکشن ها ممکن است متغیر محلی داشته باشند . این متغیرها در بدنه فانکشن تعریف می‌شوند و تنها درون همان فانکشن قابل استفاده‌اند .
- پارامترها می‌توانند مقدار پیش فرض داشته باشند :

```
function sum(a = 1, b = 2) {...}
```

- فانکشن‌ها همیشه مقداری را برمی‌گردانند . اگر در بدنه فانکشن کلمه return استفاده نشود ، مقدار undefined برگردانده می‌شود .

راهی طولانی در پیش داریم :)

تا اینجا فقط راجع بخشی از ویژگی های پایه ای جاوا اسکریپت صحبت کردیم . در ادامه آموزش و بخش های بعد ، راجع به ویژگی‌های دیگر جاوا اسکریپت صحبت خواهیم کرد .