



عملگرهای منطقی در جاوا اسکریپت

عملگرهای منطقی مختلفی در جاوا اسکریپت وجود دارد که در زیر در رابطه با آنها حرف خواهیم زد.

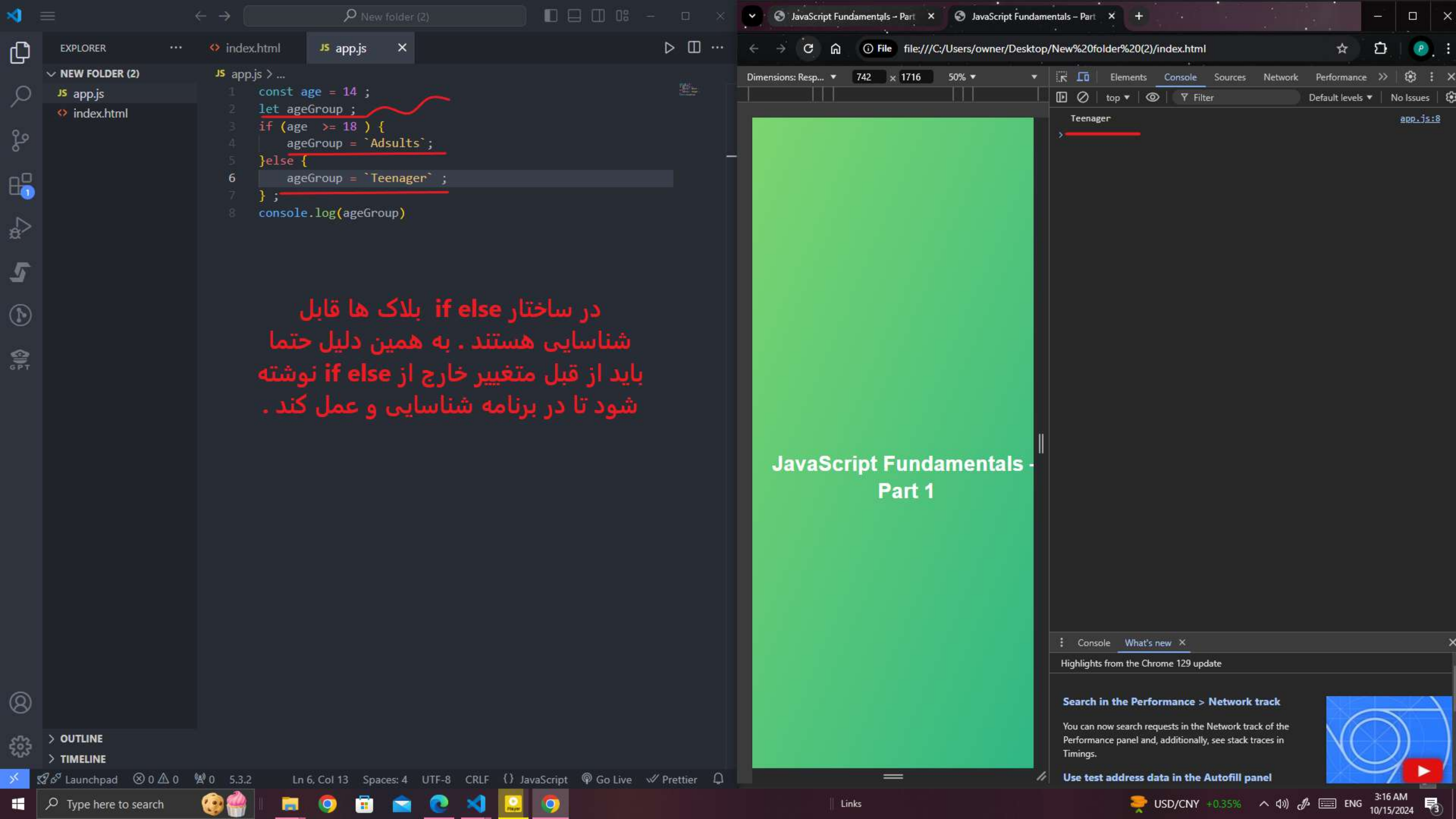
مثال	توضیحات	عملگر منطقی
		
$(10==20 \ \&\& \ 20==33) = \text{false}$	Logical AND	$\&\&$
$(10==20 \ \ 20==33) = \text{false}$	Logical OR	$ $
$!(10==20) = \text{true}$	Logical Not	$!$

عملگرهای مقایسه جاوا اسکریپت

عملگر مقایسه جاوا اسکریپت دو عملوند را با هم مقایسه می‌کند. عملگرهای مقایسه به شرح تصویر است:

مثال	توضیحات	عملگر مقایسه ای
$10 == 20 = \text{false}$	برابری	<code>==</code>
$10 === 20 = \text{false}$	یکسان (برابر و از یک نوع)	<code>===</code>
$10 != 20 = \text{true}$	غیر برابر	<code>!=</code>
$20 !== 20 = \text{false}$	یکسان نیست	<code>!==</code>
$20 > 10 = \text{true}$	بزرگتر از	<code>></code>
$20 >= 10 = \text{true}$	بزرگتر مساوی از	<code>>=</code>
$20 < 10 = \text{false}$	کوچکتر از	<code><</code>
$20 <= 10 = \text{false}$	کوچکتر مساوی از	<code><=</code>

نوعن عملگر حسابی	توضیحات	مثال
+	جمع	$10+20 = 30$
-	تفریق	$20-10 = 10$
*	ضرب	$10*20 = 200$
/	تقسیم	$20/10 = 2$
%	مدول (باقی مانده)	$20\%10 = 0$
++	افزایش	var a=10; a++; Now a = 11
--	کاهش	var a=10; a--; Now a = 9



در ساختار if else بلاک ها قابل
شناسایی هستند . به همین دلیل حتما
باید از قبل متغیر خارج از if else نوشته
شود تا در برنامه شناسایی و عمل کند .

JavaScript Fundamentals - Part 1

Console What's new X

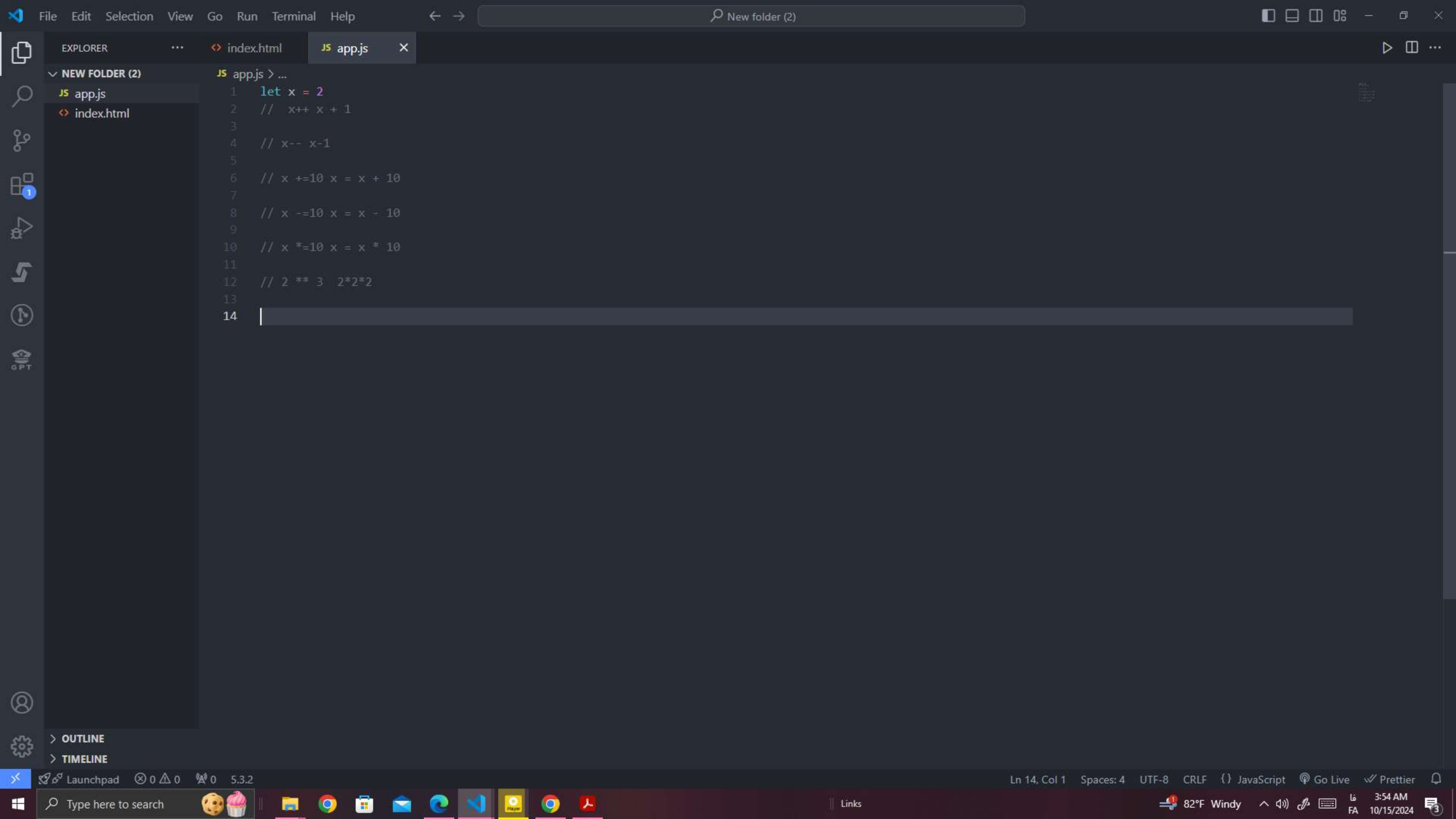
Highlights from the Chrome 129 update

Search in the Performance > Network track

You can now search requests in the Network track of the Performance panel and, additionally, see stack traces in Timings.

Use test address data in the Autofill panel





EXPLORER

index.html

JS app.js

NEW FOLDER (2)

JS app.js

index.html

JS app.js > ...

```
1 let x = 2
2 // x++ x + 1
3
4 // x-- x-1
5
6 // x +=10 x = x + 10
7
8 // x -=10 x = x - 10
9
10 // x *=10 x = x * 10
11
12 // 2 ** 3 2*2*2
13
14
```

OUTLINE

TIMELINE

FileEditSelectionViewGoRunTerminalHelp

EXPLORERNEW FOLDER (2)JS app.jsindex.html

JS app.js > ...
1 // type conversion
2
3 const brithYear = '2002';
4
5 console.log(Number(brithYear) , brithYear);
6
7 console.log(brithYear + 22) ;
8
9 console.log(Number(brithYear) + 22) ;
10
11 console.log(Number("this practice did it in 2024.")) ;
12
13 console.log(typeof(NaN)) ;
14
15 console.log(String(22) , 22) ;
16

Launchpad005.3.2

JavaScript Fundamentals – Part 1

220 x 1716

File file:///C:/Users/owner/Desktop/New%20Folder (2)/New folder (2) ...

ElementsConsoleSourcesNetwork

2002 '2002' app.js:3
200222 app.js:5
2024 app.js:7
NaN app.js:9
number app.js:11
22 22 app.js:13

در اینجا valu به صورت string نوشته شده است.

تابع Number تابعی است که string را به number به صورت دستی تغییر می دهد.

*** در اینجا با استفاده از تابع Number می خواهیم string را به number تبدیل کنیم.

NaN جز دسته number ها است و مقدار آن به صورت (not a number) است.

تابع String یک نوع تابع است که به صورت دستی number را به صورت string در می آید.

*** در اینجا می بینید که میزان را به صورت NaN نشان می دهد.

جمع به صورت number با استفاده از تابع Number

در اینجا جمع string در کنار number به صورت بالا هست. (خط 7)

به تغییر valu ها از یک حالت به حالتی دیگر که به صورت دستی انجام می شود را conversion می گویند. این تغییرات فقط برای این سه نوع valu صدق می کند.
string.1
number.2
boolean.3

75°F Partly cloudy9:58 AM10/15/2024

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)

- app.js
- index.html

index.html app.js

JS app.js > ...

```
1 // type coercion
2
3 console.log('10' + '13' + 3);
4
5 console.log(10 + 13 + 3);
6
7 console.log('33' - '10' - 3);
8
9 console.log('24' / '12');
10
11 console.log('12' * '2');
12
13 let n = '1' + 1;
14
15 n = n - 1;
16
17 console.log(n);
18
19 console.log(3 + 2 + 4 + '5');
20
21 console.log('25' - '12' - 4 + '6');
```

OUTLINE

TIMELINE

Ln 21, C

Launchpad 0 0 0 5.3.2

Type here to search

Links

76°F Partly cloudy 10:48 AM 10/15/2024

JavaScript Fundamentals - Part 1

file:///C:/Users/owner/D... 220 x 1716

Elements Console Sources

10133 app.js:3

26 app.js:5

20 app.js:7

2 app.js:9

24 app.js:11

10 app.js:17

95 app.js:19

96 app.js:21

JavaScript Fundamentals - Part 1

خروجی به صورت عدد

در این نوع تغییرات خود JS به صورت خودکار و با توجه به عملگرها valu ها را تغییر می دهد.

عملگر + همه ی valu ها را به string تبدیل می کند. جواب این خط می شود 10133 چون همه تبدیل به string شده است.

عملگرهای * / valu ها را به حالت number در می آورند. و مانند اعداد با آنها برخورد می کنند.

در اینجا در خط اول عملگر + است پس همه valu تبدیل به string شده و جواب 11 می شود.

در خط دوم عملگر - است و جواب 11 تبدیل به number شده و جواب 10 می شود.

در اینجا سه تا از اعداد به صورت number هستند ولی آخری به صورت string نوشته شده که نتیجه به صورت string است و جمع سه عدد اول به صورت عددی 9 و در کنار string جواب می شود 95

FileEditSelectionViewGoRunTerminalHelp

EXPLORER

NEW FOLDER (2)
JS app.js
index.html

index.htmlJS app.js

```
1 // falsly and turthly value
2
3 // false values : 0 , "" , undefined , NaN , null ;
4
5 console.log(Boolean(0));
6 console.log(Boolean(""));
7 console.log(Boolean(NaN));
8 console.log(Boolean(null));
9 console.log(Boolean(undefined));
10
11 console.log(Boolean(100)) ;
12 console.log(Boolean({}));
13
14
15
16
17
18
19
20
21
```

Ln 18, Col

JavaScript Fundamentals - Part

file:///C:/Users/owner/D...

220 x 1716

ElementsConsoleSources

Filter

No Issues

falseapp.js:5

falseapp.js:7

falseapp.js:9

falseapp.js:11

falseapp.js:13

trueapp.js:15

trueapp.js:17

در جاوا اسکریپت 5 نوع مقدار false داریم . که شامل موارد زیر است :

این 5 values همانطور که می بینید با استفاده از تابع boolean نشان داده شده است که مقدار آن false هستند .

بقیه value ها از جمله object خالی نیز جز دسته true ها هستند . همانطور که تابع boolean نشان داده است .

نکته : تابع Boolean نوع value را نشان می دهد که true هست یا false

در اینجا مقدار متغیر **empty** است پس در شرط مقدار **false** بوده و شرط اجرا نمی شود و **else** اجرا می شود .

```
1 // falsly and turthly value
2
3 // false values : 0 , "" , undefined , NaN , null ;
4
5 let age ;
6
7 if (age) {
8     console.log(`The Boolean is true. age is defin .`)
9 }else{
10     console.log(`The Boolean is false . age is undefind .`);
11 }
12
```

در اینجا نیز به دلیل 0 بودن مقدار متغیر **else** اجرا می شود .

```
13 let weight = 0 ;
14
15 if (weight) {
16     console.log(`The Boolean is true. weight is true .`)
17 } else {
18     console.log(`The Boolean is false .weight is false .`);
19 }
20
```

در اینجا 300 یک عدد هست که در **true** ها قرار دارد پس در اینجا **if** به دلیل **true** بودن مقدار اعمال می شود .

```
21 let height = 300 ;
22
23 if (height) {
24     console.log(`The Boolean is true. height is true .`)
25 } else {
26     console.log(`The Boolean is false .height is false .`);
27 }
28
```

نکته : تابع **if** همیشه مقدار های **true** را پشتیبانی می کند و در صورتی که **value** شرط **true** باشد به آن عمل می کند . پس یادمان باشد برای اجرا شدن شرط از اون 5 تا مقدار **false** استفاده نکنیم .

The Boolean is false . age is undefind . app_js:10
The Boolean is false .weight is false . app_js:18
The Boolean is true. height is true . app_js:24

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)
app.js
index.html

index.html app.js

```
1 const age = 18 ;  
2  
3 if (age === 18) {  
4     console.log(` I'm happy .`);  
5 }else{  
6     console.log(`I'm sad . `);  
7 } ;  
8  
9 const year = '2024' ;  
10  
11 if (year === 2024 ) {  
12     console.log(`good year`);  
13 }else if (year !== 2024){  
14     console.log(` bad year`);  
15 }  
16  
17 const IDNumber = '2525' ;  
18  
19 if (IDNumber == 2525){  
20     console.log(` this is mine .`);  
21 }else {  
22     console.log(`this is not mine `);  
23 }  
24  
25
```

=== این نوع عملگر معادله ها را با مقدار و نوع value یکسان نشان می دهد و مثلا value number با مقدار 18
!== این نوع عملگر همان عملگر بالا بوده با تفاوت اینکه به معنی (مساوی نیست) هست
== این نوع عملگر معادله ها را فقط با مقدار یکسان نشان می دهد و کاری به نوع value ندارد . مثلا 2525 در حالت string برابر هست با همین مقدار به صورت number
نکته : برای داشتن کد های تمیز و باگ کمتر خیلی بهتره از === استفاده کرد .

JavaScript Fundamentals – Part

file:///C:/Users/owner/Desktop...

140

top

Filter Default levels

No Issues

I'm happy . app.js:4

bad year app.js:14

this is mine . app.js:20

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)

- app.js
- index.html

JS app.js > ...

```
1 const haveDegree = true ;
2 const canDrive = false ;
3
4 console.log(!haveDegree);
5
6 console.log(haveDegree && canDrive );
7
8 console.log(haveDegree || canDrive );
9
10
```

! به معنی نه هست و مقدار true یا false را برعکس می کند .

&& به معنی و (and) هست .

|| به معنی یا (or) هست .

OUTLINE

TIMELINE

JavaScript Fundamentals - Part

File file:///C:/Users/owner/Desktop...

140

top

Filter Default levels

No Issues

false	app.js:4
false	app.js:6
true	app.js:8

>

Ja
Fu
-
Pa

BASIC BOOLEAN LOGIC: THE AND, OR & NOT OPERATORS

A AND B

"Sarah has a driver's license
AND good vision"

Possible values

		A	
		TRUE	FALSE
B	TRUE	TRUE	FALSE
	FALSE	FALSE	FALSE

Results of operation, depending on 2 variables

👉
true when **ALL** are **true**

👉 No matter how many variables

A OR B

"Sarah has a driver's license
OR good vision"

		A	
		TRUE	FALSE
B	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE

👉
true when **ONE** is **true**

NOT A, NOT B



Inverts **true/false** value

👉 EXAMPLE:

A: Sarah has a driver's license

B: Sarah has good vision

👉 Boolean variables that can be either TRUE or FALSE

AN EXAMPLE 🧑💻

BOOLEAN VARIABLES

👉 A: Age is greater or equal 20

false

👉 B: Age is less than 30

true

age = 16

LET'S USE OPERATORS!

👉 !A

false

true

👉 A AND B

false

true

false

👉 A OR B

false

true

true

👉 !A AND B

true

true

true

👉 A OR !B

false

false

false

		A	
B	AND	TRUE	FALSE
	TRUE	TRUE	FALSE
	FALSE	FALSE	FALSE

		A	
B	OR	TRUE	FALSE
	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE

در اینجا دو مقدار یکی **true** و دیگری **false** در شرط با عملگر **&&** در کنار هم هستند .

```
1 const haveDegree = true ;
2 const canDrive = false ;
3
4 if (haveDegree && canDrive){
5     console.log(`You are lucky man .`) ;
6 }else{
7     console.log(`You must try .`);
8 }
9
10
11
12
13 if (haveDegree && !canDrive ){
14     console.log(`wow! you are great .`);
15 }else{
16     console.log(`Yaep!`);
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

true در کنار **false** در بین عملگر **&&** منجر به **false** می شوند . در این مثال نتیجه را مشاهده می کنید .

در اینجا با علامت **!** مقدار **canDrive** را از **false** به **true** تغییر دادیم . و الان هر دو متغیر ها **true** هستند .

true در کنار **true** بین عملگر **&&** منجر به **true** می شود . در این مثال نتیجه را مشاهده می کنید .

JavaScript Fundamentals - Part

file:///C:/Users/owner/Desktop...

168 x

Elements Console >> Filter

Default levels No Issues

You must try . app.js:7

wow! you are great . app.js:14

Ja Fu

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)

- app.js
- index.html

index.html

app.js

```
1 const haveDegree = true ;
2 const canDrive = false ;
3
4
5
6
7
8 if (haveDegree || canDrive) {
9     console.log(`You should learn other skills`);
10 } else {
11     console.log(`You need to get them .`);
12 }
13
14
15
16
17
18 if (!haveDegree || canDrive ){
19     console.log(`harry up !`);
20 }else{
21     console.log(`that is right .`);
22 }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

در اینجا دو مقدار **true** و **false** کنار هم قرار گرفته اند . و میان آنها عملگر **||** قرار دارد . در اینجا وقتی **||** وسط دو مقدار متفاوت **true** و **false** در نتیجه خروجی **true** است . در مثال هم مشاهده می کنید .

در اینجا **haveDegree** با **!** مقدارش از **true** به **false** تغییر کرده است . الان دو مقدار **false** کنار هم هستند . که نتیجه آن هم **false** است .

JavaScript Fundamentals - Part

File file:///C:/Users/owner/Desktop/...

168 x

Elements Console >> Filter

Default levels No Issues

You should learn other skills app.js:9
that is right . app.js:21

Ja
Fu
-

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)
app.js
index.html

index.html JS app.js

```
1 // switch statment
2 // we shuold to use it when we have too much condition . 😊
3
4 const day = "saturday";
5
6 switch (day) {
7
8   case "monday":
9     console.log(`I have to go to school.`);
10    break;
11
12   case "tuesday":
13   case "wendesday":
14     console.log(`I have to do my assignment.`);
15     break;
16
17   case "thursday":
18     console.log(`I have to clean my room.`);
19     break;
20
21   case "friday":
22     console.log(`I have to visit my grandparents.`);
23     break;
24
25   case "saturday":
26   case "sunday":
27     console.log(`I have to rest.`);
28     break;
29
30   default:
31     console.log(`Not in my plan.`);
32     break;
33
34 }
```

1/ باید متغییری بنویسیم که با توجه به آن شرط انجام شود . 😊

2/ switch را نوشته و جلوی آن متغییر مد نظر را که باید شرط با توجه به آن اعمال شود را داخل پرانتز می نویسیم .

3/ با نوشتن case و نوشتن value مورد نظر دو نقطه می گذاریم بعد event که مد نظرمون هست را می نویسیم .

4/ بانوشتن break یعنی شرص مورد نظر تمام شده و وارد case بعدی می شوید .

5/ در آخر از default در صورتی که هیچ شرطی اعمال نشد استفاده می کنیم .

switch کار با دستو آموزه J x +

File file:///C:/Users/owner/Desktop... 147 x

Elements Console >> Filter

Default levels No Issues

I have to rest. app.js:26

یادتون باشه همیشه حتما از break بلافاصله بعد از case هاتون استفاده نمایید، چون اگه یک کیس break نداشته باشه حتی اگه درست هم باشه بازم اجرا نمیشه و کیس بعدیش اجرا میشه.

همیشه کیس آخر چون آخرین کیس هست و بعدش کیس نیست که بخواد بررسی شه بهمین دلیل اختیاری هست که از break استفاده نمایید، چون آخرین کیس هست بصورت خودکار دیگه خودش حلقه رو متوقف میکنه.

اگه چند تا value یک event مشخص داشتن پس پشت سر هم می نویسم و بعد break می کنیم .

Ja Fu P

Launchpad 0 0 0 5.3.2

Type here to search

Links

66°F Sunny 6:48 AM 10/22/2024

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)
app.js
index.html

index.html app.js

1 // Conditional Operator
2 // we shuold to use it when we use value . 😊
3
4 const age = 13;
5
6 const drive = age >= 18 ? `drive a car 🚗` : `ride a bike 🚲` ;
7
8 console.log(drive);
9
10 let drive2 ;
11 if (age >= 18) {
12 drive2 = `drive a car 🚗`
13 } else {
14 drive2 = `ride a bike 🚲`
15 }
16
17 console.log(drive2);
18
19 console.log(`sara is \${age} years old so she can \${age >= 18 ? `drive a car 🚗` : `ride a bike 🚲`} `);
20
21

JS app.js > ...

71

File file:///C:/...
No Issues

ride a bike 🚲 app.js:8
ride a bike 🚲 app.js:17
sara is 13 years old so she can ride a bike 🚲 app.js:19
conditional operator
ها از نوع expressions ها هستند و می توانند value بسازند . در مثال ما value از نوع string ساختیم . و این فرق اساسی با if else هست .
!!!! در کد ها و مسائل بزرگتر باید از if else استفاده کرد و برای شرط های کوچک و یا در شرایط خاص باید از operator استفاده کرد .

Launchpad 0 0 5.3.2

Type here to search

Links

60°F Partly cloudy 8:49 AM 10/22/2024

در اینجا ؟ به معنی if بوده و :
به معنی else است .

if else از نوع statment ها هستند و نمی توانند هیچ value بسازند. به همین دلیل ما باید یک متغیر خارج از بلاک if else بسازیم تا دسترسی به اطلاعات پیدا کنیم .

هیچ وقت نمی شود یک statment را داخل یک template literal نوشت چون هیچ value به ما نمی دهد . ولی conditional operator ها به دلیل اینکه value می سازند گزینه ای مناسبی هستند .
!!! در کل هر وقت نیاز به ساخت value در شرط شدیم از conditional operator ها استفاده می کنیم .

FUNCTIONS REVIEW: ANATOMY OF A FUNCTION

Function name

Parameters: placeholders to receive input values. Like local variables of a function

Function body: block of code that we want to reuse. Processes the function's input data

return statement to output a value from the function and terminate execution

```
function calcAge(birthYear, firstName) {  
  const age = 2037 - birthYear;  
  console.log(`${firstName} is ${age} years old`);  
  return age;  
}
```

```
const age = calcAge(1991, 'Jonas')
```

Calling, running or invoking the function, using ()

Variable to save returned value (function output)

Arguments: actual values of function parameters, to input data

File Edit Selection View Go Run Terminal Help

EXPLORER

NEW FOLDER (2)
JS app.js
index.html

JS app.js > ...
1 'use strict';
2
3 //Function Declaration ;
4
5 function callAge (brithYear) {
6 return 2023 - brithYear ;
7 }
8
9 const age1 = callAge(2002) ;
10
11 console.log(age1);
12
13
14 //function expression ;
15
16 const callAge2 = function (brithYear) {
17 return 2023 - brithYear ;
18 }
19 const age2 = callAge2(2002) ;
20 console.log(age2);
21

کلمه کلیدی که نتیجه تابع را مشخص می کند . لزوماً همه ی توابع این دستور را ندارند .
نحوه ی صدا زدن توابع می تواند به صورت تعریف یک متغیر باشد . به 2002 argument می گوئیم که مقدار پارامتر را مشخص می کند

پارامتر

نوعی تابع است که در آن اول function تعریف می شود و بعد با یک متغیر فراخوانی می شود . به این نوع توابع function declaretion می گویند .

نوعی تابع است که در آن اول از طریق یه متغیر تابع تعریف می شود و جلوی متغیر function را بدون نام تعریف می کنیم چون نام در متغیر نوشته شده است . و بعد از طریق یه متغیر دیگر call میکنیم تابع مد نظر را . به این نوع تابع function expression می گویند .

نکته : در تابع های expression تابع یک نوع value حساب می شود ولی نوع خاصی ندارد .
بهتر است از تابع expression استفاده شود زیرا نظم بهتری دارد ولی استفاده از هر دو در جای مناسب درست است .

فرق ای نوع تابع این است که در تابع declaration اگر اول تابع را call کنیم و بعد تعریف کنیم عمل می کند ولی در تابع expression اینگونه نیست و اول باید تابع تعریف شود و بعد call شود .

Ln 13, Col 1

function x JavaScript x

File file:///C:... 1716

Elements Console >> Filter

Default levels No Issues

21 app.js:11
21 app.js:19
>

Console What's new

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance

The new live metrics in the Performance panel now provide environment recommendations to help you better understand your users' experience.

Air: Severe 8:02 AM 10/27/2024

FileEditSelectionViewGoRunTerminalHelp

EXPLORER

NEW FOLDER (2)
JS app.js
index.html

JS app.js > ...
1 'use strict';
2
3 //Arrow Function ;
4
5 const callAge = brithYear => 2024 - brithYear ;
6
7 const age = callAge (2002);
8
9 console.log(age);
10
11 // *****//
12
13 const yearRetirement = (brithYear , fristName) =>{
14 const age = 2024 - brithYear ;
15 const retirement = 65 - age ;
16 return `\${fristName} retire untile \${retirement} years .`
17 }
18
19 console.log(yearRetirement(2002 , ' marjan'));
20

Arrow Function
1/ یک متغیر تعریف می کنیم .
2/ پارامتر مد نظر را می نویسیم .
3/ => جلوی پارامتر می نویسیم .
4/ بدنه تابع را می نویسیم .
5/ تابع را call می کنیم .

در صورت داشتن چند پارامتر
یک پرانتز باز می کنیم و داخل
آن پارامتر های مد نظر را می
نویسیم و بین آنها کاما می
گذاریم .

در صورت اینکه بدنه تابع
شامل چندین دستور باشد ،
جلوی => یک {} باز می
کنیم و دستور ها را می
نویسیم .

OUTLINE
TIMELINE

Launchpad 0 0 5.3.2

JavaScript

File file... Filter

71 x 1716

Elements Console

Default levels No Issues

22 app.js:9
marjan retire untile 43 years . app.js:19
>

Console What's new

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance

The new live metrics in the Performance panel now provide environment recommendations to help you better understand your users' experience.

Ln 22, Col 1

Type here to search

Links

Live 9:32 AM 10/27/2024

FUNCTIONS REVIEW: 3 DIFFERENT FUNCTION TYPES

👉 Function declaration

Function that can be used before it's declared



👉 Function expression

Essentially a function *value* stored in a variable



👉 Arrow function

Great for a quick one-line functions. Has no `this` keyword (more later...)



```
function calcAge(birthYear) {  
  return 2037 - birthYear;  
}
```

```
const calcAge = function (birthYear) {  
  return 2037 - birthYear;  
};
```

```
const calcAge = birthYear => 2037 - birthYear;
```



Three different ways of writing functions, but they all work in a similar way: receive **input** data, **transform** data, and then **output** data.

JS app.js > ...

```

3
4 const names = ['marjan', 'ali', 'maryam'];
5
6 const years = new Array (2002, 2010, 2023, );
7
8 console.log(names [0]);
9
10 console.log(names.length);
11
12 console.log(names[names.length-1]);
13
14
15
16
17

```

که در اینجا **names** میشه نام آرایه مون ،
بعدش یه علامت مساوی = میزاریم و بعدش
براکت باز] و در نهایت آیتم ها یا مقادیر آرایه
رو وارد میکنیم (آیتم ها با کاما , از همدیگه
جدا میشن) و در آخرم براکت بسته] میزاریم.

روش دیگه استفاده از کلمه کلیدی **new**
هست و سپس جلوی آن **array** را نوشته و
بعد پرانتز باز کرده و **value** ها را می
نویسیم .

⚠ کلا استفاده از **new** پیشنهاد نمیشه و باعث
میشه سرعت اجرای برنامه کاهش پیدا کنه و
از طرفی این روش خوانایی ضعیف تری
نسبت به روش قبلی داره.

در اینجا به **value** مد نظر در **array** دسترسی
پیدا می کنیم . به این گونه که اسم **array** را
نوشته و سپس براکت را باز می کنیم و شماره
value مد نظر را می نویسیم .

وقتی که می خواهیم بدونیم در هر **array** چند تا
value قرار دارد از **length** کنار اسم **array** می
نویسیم و تعداد **value** مشخص می شود .

برای پیدا کردن **value** اخر از این ترفند
استفاده کرد . بعد از اسم **array** براکت
باز کرده و داخل براکت
array-name.length-1 را می
نویسیم .

⚠⚠⚠ در شماره گذاری **value** ها در
array ، **value** اول صفر حساب می
شود و دوم یک یعنی شماره گذاری
value از صفر شروع می شود .

Elements Console >>

top Filter

Default levels No Issues

marjan	app.js:8
3	app.js:10
maryam	app.js:12

> |

Console What's new X

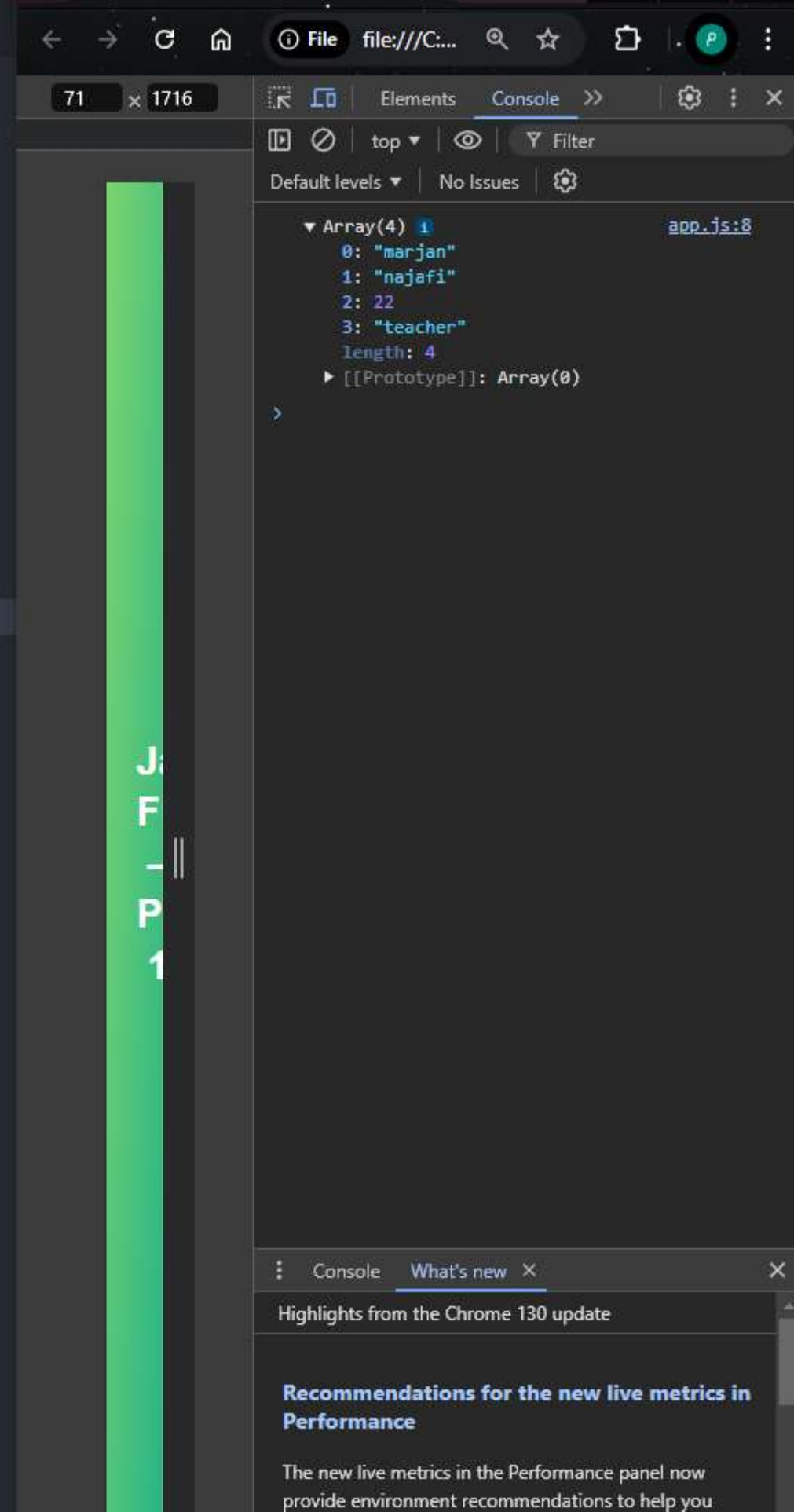
Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance


```
EXPLORER
...
index.html
JS app.js
NEW FOLDER (2)
JS app.js
index.html

JS app.js > ...
1 'use strict';
2 // Array ;
3
4 const fristName = 'marjan' ;
5
6 const worker = [fristName , 'najafi' , 2024-2002 , 'teacher'] ;
7
8 console.log(worker);
9
10
11
12
13
14
15
```

value ها در **array** می توانند از **type** های مختلفی باشند حتی می توانند **expression** باشند . الان در این مثال متغیر شامل یک **value** هست که در کنسول نشان داده شده است . همچنین تفریق یک نوع **expression** هست که در **array** محاسبه می شود و در کنسول مشاهده می کنید .



JS app.js > ...

```
1 'use strict';
2 // Array ;
3
4 const callAge = function (birthYear) {
5   return 2024 - birthYear ;
6 } ;
7
8 const years = [1990 , 2020 , 2015 , 1990 , 1987 , 2014 , 1995] ;
9
10 const age = [callAge(years[0]) , callAge(years[1]) , callAge(years[years.length-1])];
11
12 console.log( age );
13
14
```

آخرین value

1/ در اینجا ما به تابع تعریف می کنیم که سن افراد را حساب می کند .

2/ سپس متغیری تعریف می کنیم که در آن array از سال تولد افراد را شامل می شود .

3/ متغیر دیگری تعریف می کنیم که در آن array از سن ها را شامل می شود . در اینجا تابع را صدا می زنیم و بعد array که اسمش years هست را جا گذاری می کنیم .

نکته :

در array می توان call function ها را نوشت چون نتیجه انها value هست و خود عبارت نوعی expression است .

71 x 1716

Elements Console >>

top Filter

Default levels No Issues

(3) [34, 4, 29]

app.js:12

>

Console What's new

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance

The new live metrics in the Performance panel now provide environment recommendations to help you

JS app.js > ...

```
1 'use strict';
2 // Array ;
3
4 const friends = ['marjan' , 'sara' , 'zahra' , 'mina'] ;
5
6 friends.push('tina') ;
7
8 console.log(friends);
9
10 friends.unshift('sima') ;
11
12 console.log(friends);
13
14
```

به این array با دقت نگاه کنید .

متد push برای اضافه کردن value مد
نظر به آخر array می شود.

متد unshift برای اضافه کردن value
مد نظر به اول array استفاده می
شود .

متد ها نوعی تابع هستند . پس ما
می توانیم مانند تابع ها باهاشون
رفتار کنیم .

File file:///C:/Users/owner/Desktop/New%20folder...

71 x

Elements Console Sources Network >> Filter Default levels No Issues

(5) ['marjan', 'sara', 'zahra', 'mina', 'tina'] app.js:8

(6) ['sima', 'marjan', 'sara', 'zahra', 'mina', 'tina'] app.js:12

> |

Console What's new x

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance

The new live metrics in the Performance panel now

(2)

JS app.js > friends

```
1 'use strict';  
2 // Array ;  
3   
4 const friends = ['marjan', 'sara', 'zahra', 'mina', 'azadeh'] ;
```

به این array با دقت نگاه کنید .

```
6 friends.pop() ;  
7  
8 const popped = friends.pop() ;  
9
```



متد pop ، value مد نظر را از آخر array حذف می کند .




```
11 console.log(friends);  
12  
13 friends.shift() ;  
14  
15 console.log(friends);  
16  
17
```

همانند تابع های دیگر در متغییر تعریف کردیم و سپس تابع pop را call کردیم .

متد shift ، value مد نظر را از اول array حذف می کند .

71 x

Elements Console Sources Network >>   X

  top  Filter Default levels No Issues



▶ (3) ['marjan', 'sara', 'zahra'] app.js:11

▶ (2) ['sara', 'zahra'] app.js:15

>

Console What's new X

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance




```
JS app.js > ...
1 "use strict";
2 // Array ;
3
4 const friends = ["marjan", "sara", "zahra", "mina", "azadeh"];
5
6
7 console.log(friends.indexOf("zahra"));
8
9 console.log(friends.indexOf("ali"));
10
11 console.log(friends.includes('sara'));
12
13 console.log(friends.includes('mohamad'));
14
15
16
```

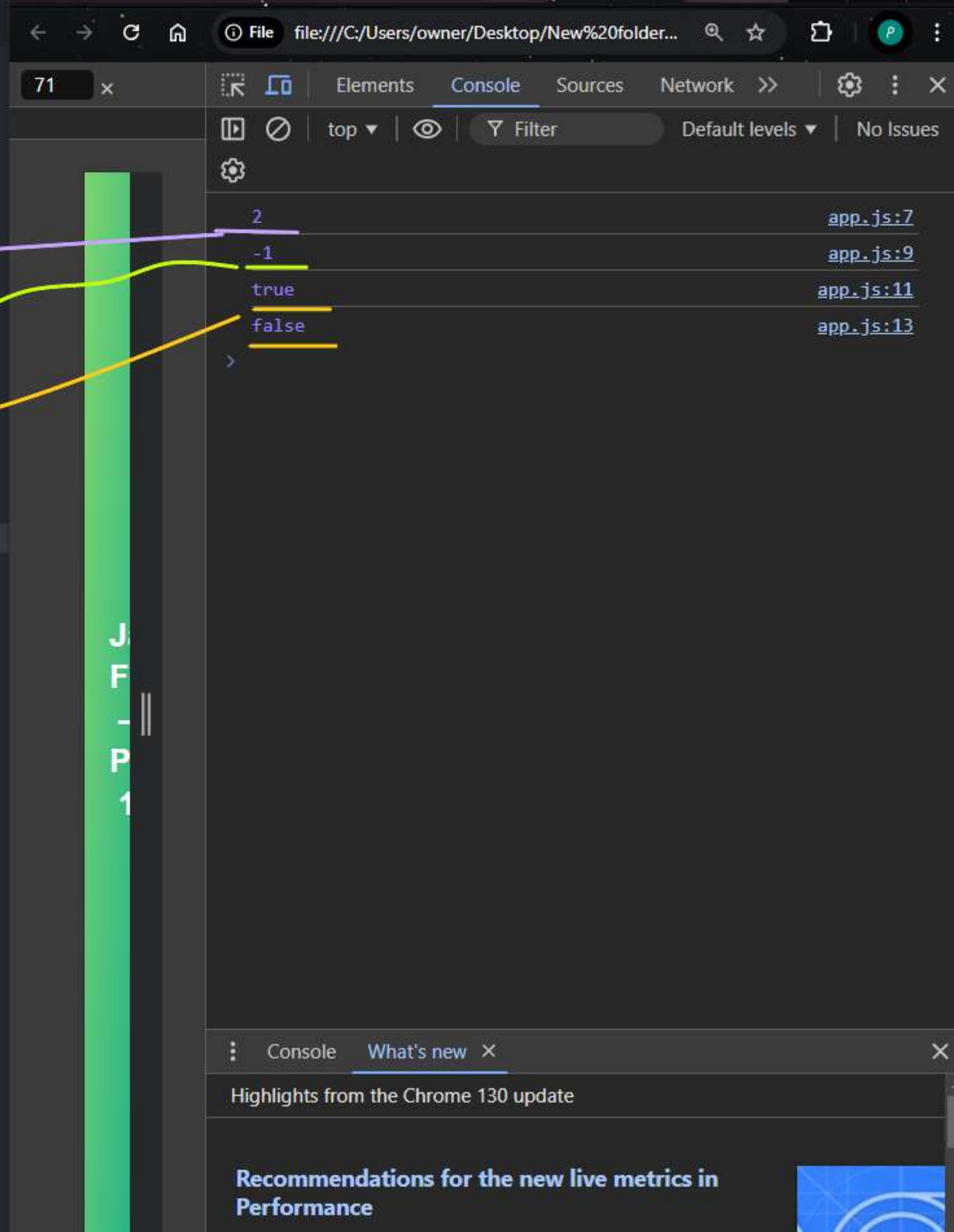
اگر value مد نظر در array نباشد
اون وقت مقدار ان -1 می شود .

متد indexOf :

متدی است که در آن برای جستجو value مد نظر استفاده می شود . و نشان می دهد value مد نظر چندمین value در array است .

متد includes :

متدی است که در آن برای جستجو value مد نظر و در صورت بودن در array با true و نبودن با false نشان می دهد . در این متد باید نوع value منطبق باشد .




```
1 "use strict";
```

```
2 // Object ;
```

```
3
```

```
4 const personInfo = {
```

```
5   fristName : 'marjan',
```

```
6   lastName : 'najafi',
```

```
7   job : 'developer',
```

```
8   age : 2023-2002 ,
```

```
9   friends : ['mina' , 'zahra' , 'tina' , 'parisa']
```

```
10 };
```

```
11
```

```
12 console.log(personInfo);
```

```
13
```

```
14
```

1/ در مرحله ی اول یک **object-name** انتخاب می کنیم . و {} را باز می کنیم .

2/ در مرحله ی دوم برای **key** مد نظر اسم لنتخاب می

کنیم و جلوی **key** یه : می گذاریم .

3/ در مرحله ی سوم **value** مد نظر را نوشته و برای جدا

کردن با قسمت بعد یک کاما (,) می گذاریم .

● تفاوت اشیا با سایر انواع داده در جاوا اسکریپت

تفاوت اصلی بین اشیا (Objects) و دیگر دیتاتایپ ها در جاوا اسکریپت این است که اشیا قادر به نگهداری داده های مختلف مانند اعداد، رشته ها، آرایه ها، توابع و اشیاء دیگر هستند، در حالی که دیتاتایپ های دیگر محدود به یک نوع خاص از داده هستند. برای مثال یک متغیر از نوع **Number** تنها می تواند یک عدد را نگهداری کند و یک متغیر از نوع **String** تنها می تواند یک رشته را نگهداری کند.

به طور کلی از اشیا برای ذخیره سازی داده های مرتبط به یکدیگر و ساختاردهی منطقی تر کدها استفاده می شود. اشیا این امکان را فراهم می کنند تا داده های مختلف با هر نوع ویژگی هایی در یک **Object** ذخیره شوند.

● انواع object در جاوا اسکریپت

به صورت کلی تمامی داده های اولیه از جمله متغیرهای بولی، متغیرهای عددی، متغیرهای رشته ای و ... یک شی در جاوا اسکریپت محسوب می شوند. متغیر تاریخ همواره یک شی است، متغیرهای ریاضی یا همان **Math** در جاوا اسکریپت یک شی به شمار می آیند، **regular expression** همواره یک شی هستند، آرایه ها شی محسوب می شوند، توابع یک شی در جاوا اسکریپت هستند، خود **Object** ها یک شی در جاوا اسکریپت محسوب می شوند.

▼ Object 1

age: 21

► friends: (4) ['mina', 'zahra', 'tina', 'parisa']

fristName: "marjan"

job: "developer"

lastName: "najafi"

► [[Prototype]]: Object

>


```

JS app.js > ...
1  "use strict";
2  // Object ;
3
4  const personInfo = {
5      fristName : 'marjan' ,
6      lastName : 'najafi',
7      job : 'developer',
8      age : 2023-2002 ,
9      friends : ['mina' , 'zahra' , 'tina' , 'parisa']
10 };

```

```

11
12 1 const info = personInfo.job ;
13
14 2 const info2 = personInfo['fristName'] ;
15
16 console.log(info , info2);
17
18

```

۲ روش برای فراخوانی یا نمایش خصوصیات اشیاء داریم :

روش اول : اسم **object** را می نویسیم و بعد یک نقطه (.) می گذاریم و اسم **key** مد نظر را می نویسیم .

روش دوم : اسم **object** رو مینویسیم بعدش یه [" "] میزاریم و داخلش اسم **key** رو وارد میکنیم.

71



Filter Default levels

No Issues

developer marjan app.js:16



Console What's new


Highlights from the Chrome 130 up...

```
js app.js > ...
1  "use strict";
2  // Object ;
3
4  const personInfo = {
5    fristName : 'marjan' ,
6    lastName : 'najafi',
7    job : 'developer',
8    age : 2023-2002 ,
9    friends : ['mina' , 'zahra' , 'tina' , 'parisa']
10 };
11
12 const keyName = 'Name' ;
13 const fristName = personInfo['frist' + keyName] ;
14 const lastName = personInfo['last' + keyName] ;
15
16 console.log(fristName , lastName);
17
18
19
20
21
```


فرق روش نقطه با براکت در این است که :
در براکت شما می توانید **expression** بنویسید ولی در
روش نقطه این مورد امکان پذیر نیست .
ولی در کل بهتر است از روش نقطه بریم مگر در شرایط
خاص

71 x



top 

Filter Default levels 

No Issues 

marjan najafi [app.js:16](#)

>

Console What's new x x

Highlights from the Chrome 130 up...

Recommendations for the new I

JS app.js > [🔍] personInfo > 🔑 friends

```
1 "use strict";
2 // Object ;
3
4 const personInfo = {
5   fristName : 'marjan' ,
6   lastName : 'najafi',
7   job : 'developer',
8   age : 2023-2002 ,
9   ⚡ friends : ['mina' , 'zahra' , 'tina' ]
10 };
11
12 personInfo.location = 'Iran' ; 1
13
14 personInfo['gender'] = 'female' ; 2
15
16 console.log(personInfo);
```

برای اضافه کردن value به object از دو روش زیر می توان استفاده کرد :

1/ نوشتن object-name و بعد نقطه سپس نوشتن key جدید و بعد مساوی نوشتن value

2/ نوشتن object-name سپس نوشتن key در '' و بعد از مساوی نوشتن value

71 x

Elements Console >> ⚙️ ⋮ ✕

top ▾ 🔍 Filter

Default levels ▾ No Issues ⚙️

app.js:16

```
{fristName: 'marjan', lastName: 'najaf
▼ i', job: 'developer', age: 21, friends:
  Array(3), ...} i
  age: 21
  ► friends: (3) ['mina', 'zahra', 'tina']
  fristName: "marjan"
  gender: "female"
  job: "developer"
  lastName: "najafi"
  location: "Iran"
  ► [[Prototype]]: Object
```

⋮ Console What's new ✕

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance

The new live metrics in the Performance panel now provide environment recommendations to help you

JS app.js > ...

```
1  "use strict";
2  // Object ;
3
4  const personInfo = {
5    fristName: "marjan",
6    lastName: "najafi",
7    job: "developer",
8    brithYear: 2002,
9    friends: ["mina", "zahra", "tina"],
10   location: "Iran",
11   age: function () {
12     return 2024 - this.brithYear ;
13   }
14 };
15 console.log(personInfo.age());
16
```

در اینجا کلمه کلیدی **this** باعث فراخوانی **function** شده . **this** داره به قسمتی از کد اشاره می کنه باید در تابع اجرا بشه . (یعنی پراپرتی **brithYear**)

● در **object** ها می توان **function** نیز به صورت بالا نوشت . اول نوشتن **key** و بعد از : تابع مد نظر را می نویسیم .

تعریف کلمه کلیدی **this** :

کلمه کلیدی **this** که معمولاً توی توابع استفاده میشه، به قسمتی از کد اشاره می کنه که داره تابع رو اجرا می کنه. مقدار **this** یک آبجکت هست.

فرض کنیم یک تابع توی جاوااسکریپت داره اجرا میشه. دونستن اینکه یک تابع چه جوری نوشته شده و یا توی کدوم قسمت از کد تعریف شده، روی مقدار **this** تاثیر نداره. فقط نحوه ی فراخوانی اون تابع هست که مقدار **this** رو تعیین می کنه. برای اینکه مقدار **this** رو متوجه بشیم، کافیه فقط بدونیم این تابع به چه صورت تعریف و فراخوانی شده.




```
1  "use strict";
2  // Object ;
3
4  const personInfo = {
5    brithYear: 2023,
6
7    calcage: function () {
8      this.age = 2024 - this.brithYear;
9      return this.age;
10   }
11 }
12
13
14 personInfo.calcage();
15
16 console.log(personInfo.age);
17
18
```

در صورتی که متد call نشود ،
پراپرتی جدید ساخته شده کار
نمی کند .

در جاوا اسکریپت، نوشتن یک تابع در یک شیء (object) به عنوان یک متد (method) شناخته می شود. به طور کلی، متدها توابعی هستند که به یک شیء خاص تعلق دارند و می توانند به ویژگی های آن شیء دسترسی پیدا کنند.

متد (Method):

- اگر یک تابع داخل یک شیء تعریف شده باشد، آن تابع به عنوان متد آن شیء شناخته می شود.
- متدها معمولاً برای انجام عملیات خاصی بر روی داده های موجود در شیء استفاده می شوند.

استفاده از this:

- کلمه کلیدی this در داخل یک متد به object که متد در آن تعریف شده است اشاره دارد.
- به این ترتیب، شما می توانید به ویژگی های آن object از داخل متد دسترسی پیدا کنید.

می توانید با استفاده از this یک کلید و مقدار جدید به یک شیء (object) اضافه کنید. در واقع، this به شما اجازه می دهد به خود شیء دسترسی پیدا کنید و می توانید ویژگی های جدید را به آن اضافه کنید.
در اینجا در متد calcage یک key جدید با استفاده از this می سازیم که اسم آن age است و سپس value مورد نظر را جلوی آن می نویسیم.

71 x

top

Filter

Default levels No Issues

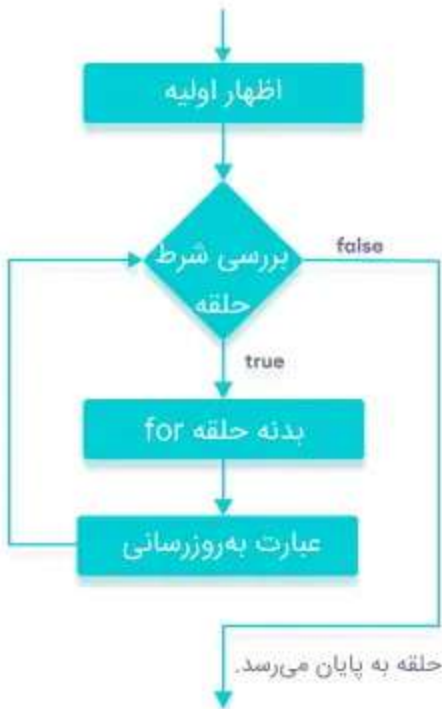
1 app.js:16

What's new

Highlights from the Chrome...

Recommendations for the Performance





FileEditSelectionViewGoRun...New folder (2)

index.htmlJS app.js

JS app.js > ...
1 "use strict";
2 // Object ;
3
4 for (let rep = 1; rep <= 10; rep++) {
5 console.log(`this is a practice of for loop \${rep}`);
6 }

1/در مرحله اول باید مقدار دهی اولیه را مشخص کنیم که حلقه از کجا باید شروع بشود.

2/در مرحله ی دوم شرطی را می نویسیم که نقطه پایان هم حساب می شود. به این صورت که اگر rep کوچکتر تا مساوی 10 باشد حلقه ادامه پیدا کند و بیشتر از آن حلقه به پایان برسد.

3/مرحله ی سوم اظهار نهایی است که نشان می دهد در هر پرش حلقه چه مقدار به آن باید اضافه شود.

71

ElementsConsole

topFilter

Default levelsNo Issues

this is a practice of for loop 1 app.js:5
this is a practice of for loop 2 app.js:5
this is a practice of for loop 3 app.js:5
this is a practice of for loop 4 app.js:5
this is a practice of for loop 5 app.js:5
this is a practice of for loop 6 app.js:5
this is a practice of for loop 7 app.js:5
this is a practice of for loop 8 app.js:5
this is a practice of for loop 9 app.js:5
this is a practice of for loop 10 app.js:5

هر حلقه دارای 3 بخش هست :
initialization (مقداردهی اولیه)
condition (شرط)
final expression (اظهار نهایی)؛ همچنین از آن با عنوان «incrementation» به معنی «افزایش» هم یاد می شود.

For
برای ایجاد حلقه در یک تکه کد به همراه ارزیابی یک شرط استفاده می شود.

Launchpad0005.3.2Ln 6, Col 2Spaces: 2UTF-8CRLFJavaScriptPort : 5500Prettier62°F Mostly sunny5:04 AM11/8/2024

JS app.js > ...

```
1 "use strict";
2 // Loop;
3
4 //break;
5
6 const personInfo = ['marjan' , 'najafi' , 2002-2024 , 'developer' , true] ;
7 const types = [] ;
8
9 for (let i = 0; i< personInfo.length; i++) {
10   if(typeof personInfo[i] === 'number' ) break ;
11   types.push(typeof personInfo[i]);
12 };
13
14 console.log(types);
15
```

وقتی به index 2 می رسد چون value عدد
است array شکسته می شود و فقط مقادیر
قبل که string هستند نمایش داده می شوند .

در اینجا شرط این است که در صورتی
که array مد نظر به number typeof
رسید حلقه شکسته شود و دیگه ادامه
پیدا نکند .

دستور break باعث می شود اجرا کدها یا دستورهای درون حلقه متوقف شود.

71

x

🔍

📄

Elements

Console >>

⚙️

⋮

✕

📄

🔍

top ▾

👁️

🔍 Filter

Default levels ▾

No Issues

⚙️

▼ (2) ['string', 'string'] ⓘ
0: "string"
1: "string"
length: 2
▶ [[Prototype]]: Array(0)

app.js:14

>

⋮

Console

What's new ✕

✕

Highlights from the Chrome 130 update

Recommendations for the new live metrics in
Performance

JS app.js > ...

```
1 "use strict";
2 // Loop;
3
4 //break;
5
6 const personInfo = ['marjan' , 'najafi' , 2002-2024 , 'developer' , true] ;
7 const types = [] ;
8
9 for (let i = 0; i< personInfo.length; i++) {
10   if(typeof personInfo[i] !== 'string' ) continue ;
11   types.push(typeof personInfo[i]);
12 };
13
14 console.log( types);
15
```

فقط این سه تا value که string هستند در
کنسول چاپ می شوند و مطابق شرط
هستند .

در اینجا مطابق شرط در صورتی
که array ، string نباشد از آن
می گذرد و وارد index بعد می
شود تا حلقه به پایان برسد .

دستور continue حلقه جاری را متوقف می کند و با مقدار بعدی، اجرای حلقه را از سر می گیرد. در واقع این دستور اجرای دستورات فرایند تکرار حلقه ی جاری را متوقف ساخته و اجرای حلقه را دوباره با تکرار (iteration) بعدی ادامه می دهد.

71 x

Elements Console >> ⚙️ ⋮ ✕

top ▾ 🔍 Filter

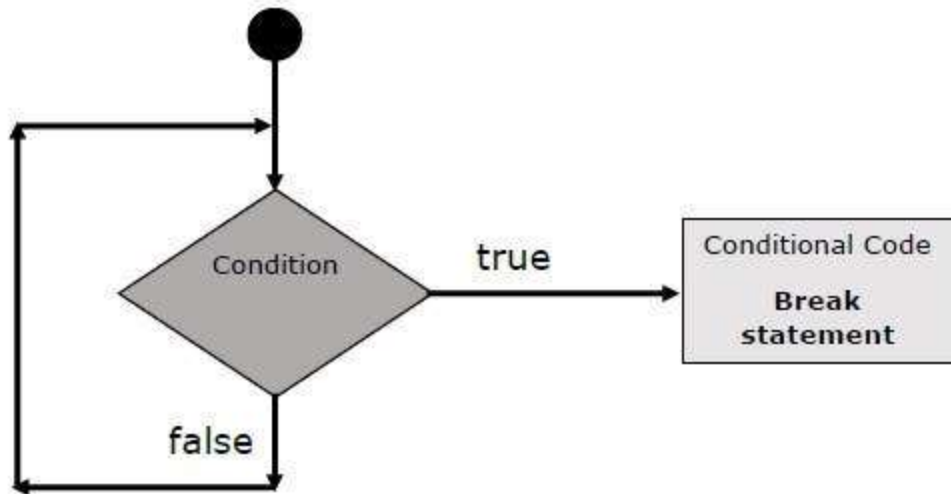
Default levels ▾ | No Issues | ⚙️

app.js:14
▶ (3) ['string', 'string', 'string']

Console What's new ✕

Highlights from the Chrome 130 update

Recommendations for the new live metrics in Performance



1/ در این مرحله اولین شرط را برای حلقه مشخص می کنید .

```
1 "use strict";
2 // While Loop ;
```

```
4 let condition = true;
```

2/ در این مرحله شرط اگر **true** باشد حلقه ادامه پیدا می کند (به صورت بی نهایت) و برنامه از حلقه خارج نمی شود .

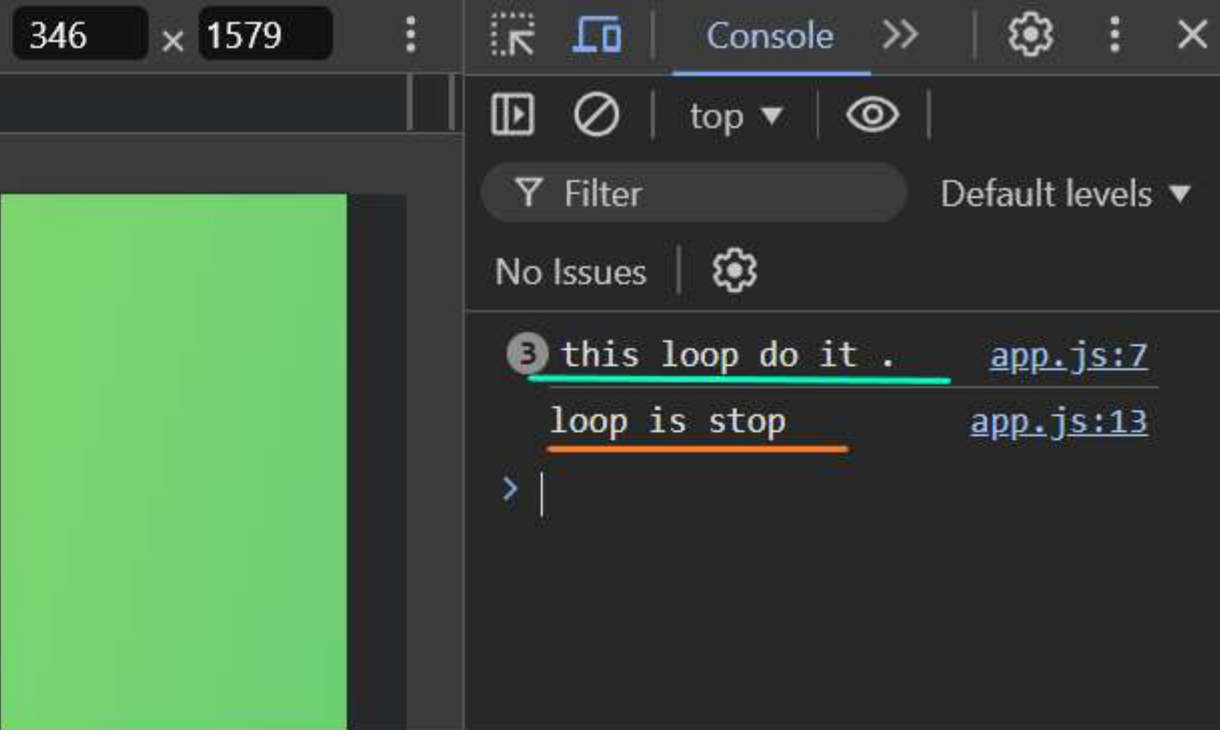
```
6 while (condition) {
7   console.log(`this loop do it .`);
```

```
9   let userInput = prompt(`Are you prefer to continue ?`);
```

```
11  if (userInput.toLocaleLowerCase() === "no") {
12    condition = false;
13    console.log(`loop is stop `);
14  }
```

3/ در این مرحله یک شرطی درست می کنیم بشود از حلقه خارج شد . در **prompt** اگر کاربر "no" را وارد کند، شرط **condition** به **false** تغییر می کند و حلقه متوقف می شود.

```
15
16
17 }
```

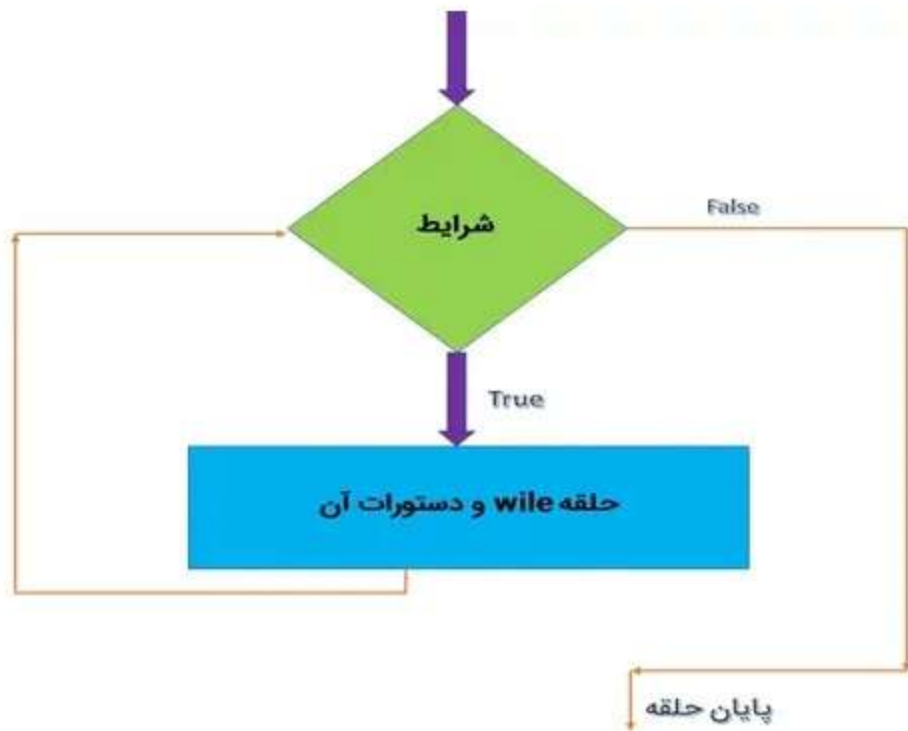


در جاوااسکریپت، می توانید از یک حلقه **while** بدون شمارنده استفاده کنید. این نوع حلقه ها معمولاً برای شرایطی که نمی دانیم چند بار باید اجرا شوند مناسب هستند و به ورودی کاربر یا شرایط دیگر بستگی دارند.

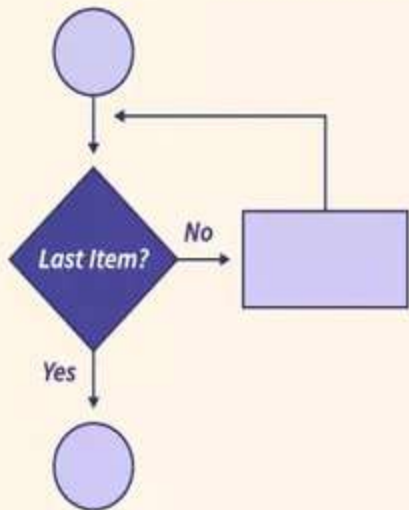
تفاوت حلقه **for** و **while** :

تفاوت حلقه **while** و حلقه **for** این است که به وسیله حلقه **for** در طول بررسی یک دنباله یک سری دستورات هم اجرا می شود، اما در حلقه **while** تا زمانی که یک شرط صدق می کند یک سری دستورات اجرا می شود.

حلقه **while** نوعی **loop** است که در آن قبل از شروع حلقه شرط مشخص می شود . سپس درون حلقه اگر شرط **true** باشد ادامه پیدا می کند و در صورت **false** شدن متوقف می شود .



For Loop



VS

While Loop

