به نام خدا

جاوا اسکریپت زبانی پرکاربرد و وسیع است که با بخشی از آن آشنا شدید. هدف از این نوشتار، مرور برخی از دستورات، اشیاء و توابع این زبان است که باید بلد باشید.

رشته (string) در جاوا اسکریپت

در این بخش به معرفی توابع و خواص string می پردازیم. رشته یا string به راحتی با نوشتن تعریف می‌شود:

```
var s = "Helli5" ;
```

اهم توابع و خواص string در زیر آمده است:

خاصیت length

این خاصیت، تعداد کاراکترها (حرف) عبارت متنی را بر می گرداند . فاصله بین حروف نیز یک کاراکتر محسوب می شود. شکل کلی استفاده از این خاصیت به شرح زیر است:

Syntax	str.length

مثال : در مثال زیر یک متغیر رشته ای را تعریف و سپس به وسیله دستور document.write ، طول آن را نشان داده ایم:

	مثال

کد	<script type="text/javascript">
	  var str = "Helli5" ;
	  document.write (str.length) ;
	</script>

خروجی	6

تابع charAt :

این تابع ، مقدار یک حرف (کاراکتر) در یک متغیر متنی ، که شماره آن را توسط خاصیت index تعیین می کنیم ، را بر می گرداند .

نکته ۱ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف v و شماره ۷ حرف c خواهد بود.

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

Syntax	str.charAt (Index) * Index = شماره حرف مورد نظر در متغیر

مثال : در مثال زیر یک متغیر رشته ای به نام matn را ایجاد و مقدار دهی کرده ایم . سپس حروف شماره ۳ و ۷ آن را به کمک تابع charAt نمایش داده ایم :

Example	

<script type ="text/javascript">
  var matn = "Helli5 School" ;
  document.write (matn.charAt (3) + "
") ;
  document.write (matn.charAt (7)) ;
</script>
	کد

l S	خروجی

تابع concat :

از این تابع برای چسباندن و اضافه کردن دو یا چند متغیر رشته ای به هم استفاده می شود . خروجی ، عبارت رشته ای جدیدی است شامل تمام متغیرهای رشته ای به هم اضافه شده .

راهنمایی : در syntax این تابع ، str نمایانگر اولین متغیر رشته ای بوده و string1 و string2 و ... سایر متغیر هایی هستند ، که می خواهیم به ترتیب به متغیر اول اضافه شوند . در این تابع باید حداقل دو متغیر برای اضافه شدن به هم تعیین شوند .

Syntax	str.concat (String1 , String2 , ...) * str = متغیر رشته ای اول * String1 = متغیر رشته ای دوم * String2 = متغیر ... هم یعنی می تواند ادامه داشته باشد و پس از آن چند رشته دیگر نیز بیاید و رشته ای شماره سوم که تعیین آن اختیاری است

مثال : در مثال زیر یک متغیر رشته ای به نام matn را ایجاد و مقدار دهی کرده ایم . سپس به وسیله تابع concat دو متغیر دیگر را به آن اضافه کرده و متن جدید را در خروجی چاپ کرده ایم :

Example

کد	

<script type ="text/javascript">
  var matn = "Helli5, " ;
  var str1 = "a School " ;
  var str2 = "for Talented Students!" ;
  document.write (matn.concat (str1 , str2)) ;
</script>	

خروجی	
Helli5, a School for Talented Students!	

تابع indexOf :

این تابع شماره مکان قرار گیری اولین نمونه یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای ، را بر می گرداند . در این تابع حرف یا کلمه مورد نظر توسط خاصیت searchvalue تعیین می شود . همچنین می توان مکان شروع جستجو در متغیر را نیز به وسیله خاصیت fromindex تعیین کرد . دراین صورت محل آغاز جستجو به جای اول متغیر ، از کاراکتر تعیین شده خواهد بود.

نکته ۱ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف v و شماره ۷ حرف c خواهد بود .

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

نکته ۳ : تابع indexOf به بزرگ یا کوچک بودن حروف حساس است .

نکته ۴ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی ۱- خواهد بود .

Syntax	str.indexOf (searchvalue , fromindex) * searchvalue = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند * fromindex = شماره مکان کاراکتری که می خواهیم عمل جستجو از آن آغاز شود را تعیین می کند

مثال : در مثال به وسیله تابع indexOf به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "world" در متغیر رشته ای مثال ، نتیجه خروجی ۱- بوده است . در آخر هم به جستجو حرف e در شرایطی که شروع جستجو از کاراکتر شماره ۵ تعیین شده ، پرداخته ایم :

Example	

<script type ="text/javascript">
  var matn = "Hello World!" ;
  document.write (matn.indexOf ("l") + "
") ;
  document.write (matn.indexOf ("World") + "
") ;
  document.write (matn.indexOf ("world") + "
") ;
  document.write (matn.indexOf ("l" , 5)) ;
</script>
	کد

2 6	خروجی

-1 9	

تابع lastIndexOf :

این تابع شماره مکان قرار گیری آخرین نمونه یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای ، را بر می گرداند . در این تابع حرف یا کلمه مورد نظر توسط خاصیت searchvalue تعیین می شود . همچنین می توان شماره مکان یک کاراکتر را نیز به وسیله خاصیت fromindex به تابع اعلام کرد ، که عملیات جستجو از سمت راست به چپ (بر عکس) ، براساس مکان آن کاراکتر صورت بگیرد .

نکته ۱ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف v و شماره ۷ حرف c خواهد بود .

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

نکته ۳ : تابع lastIndexOf به بزرگ یا کوچک بودن حروف حساس است .

نکته ۴ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی ۱- خواهد بود .

Syntax	str.lastIndexOf (searchvalue , fromindex) * searchvalue = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند * fromindex = شماره مکان کاراکتری که می خواهیم عمل جستجو از آن به صورت برعکس از راست به چپ آغاز شود را تعیین می کند .

مثال : در مثال به وسیله تابع lastIndexOf به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "world" در متغیر رشته ای مثال ، نتیجه خروجی ۱- بوده است . در آخر هم به جستجو حرف e در شرایطی که شروع جستجو از کاراکتر شماره ۵ تعیین شده ، پرداخته ایم :

Example	
<script type ="text/javascript"> var matn = "Hello World!" ; document.write (matn.lastIndexOf ("l") + " ") ; document.write (matn.lastIndexOf ("World") + " ") ; document.write (matn.lastIndexOf ("world") + " ") ; document.write (matn.lastIndexOf ("l" , 5)) ; </script>	کد
9 6 -1 3	خروجی

تابع match :

این تابع ، جهت جستجو برای وجود یا عدم وجود یک حرف یا کلمه خاص در یک مغیر رشته ای استفاده می شود . عملکرد این تابع بسیار شبیه تابع indexOf است ، با این تفاوت که به جای شماره مکان قرا گیری حرف یا کلمه مورد جستجو ، خود آن را بر می گرداند . کلمه یا حرف مورد نظر توسط خاصیت serchvalue تعیین می شود .

نکته ۱ : تابع match به بزرگ یا کوچک بودن حروف حساس است .

نکته ۲ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی null خواهد بود .

Syntax	str.match (searchvalue) * searchvalue = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند

مثال : در مثال به وسیله تابع match به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "world" در متغیر رشته ای مثال ، نتیجه خروجی null بوده است .

Example	

<script type ="text/javascript">
  var matn = "Hello World!" ;
  document.write (matn.match ("l") + "
") ;
  document.write (matn.match ("World") + "
") ;
  document.write (matn.match ("world") + "
") ;
</script>
	کد

l World null	خروجی

تابع replace :

از این تابع ، برای جستجو یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای و جایگزینی آن با یک مقدار جدید استفاده می شود . حرف یا کلمه مورد جستجو توسط خاصیت findstring و مقدار جایگزین توسط خاصیت newstring تعیین می شود.

Syntax	str.replace (findatring , newstring) * findstring = کلمه یا حرف مورد جستجو * newstring = مقدار جدیدی که می خواهیم جایگزین مقدار قبلی شود

این تابع می تواند با ۲ پارامتر اختیاری به شرح زیر استفاده شود :

پارامتر i : در صورت به کار بردن این پارامتر بعد از واژه مورد جستجو ، عملیات جستجو به بزرگ یا کوچک بودن حروف حساس نخواهد بود.

پارامتر g : در صورت به کار بردن این پارامتر بعد از واژه مورد جستجو ، کلیه موارد یافت شده ، با مقدار جدید جایگزین خواهند شد . در صورت به کار نبردن این پارامتر ، فقط اولین مورد یافت شده ، جایگزین می شود.

مثال ۱ : در مثال زیر یک متغیر رشته ای به نام ex را ایجاد و مقدار دهی کرده ایم . سپس با استفاده از تابع replace کلمه Java را با C# عوض کرده و خروجی جدید را بر روی صفحه نشان داده ایم . به دلیل عدم به کار بردن پارامتر g فقط اولین مورد کلمه Java با C# عوض شده و مرد دوم بدون تغییر باقی مانده است :

Example	
<script type ="text/javascript"> var ex = "Java is a powerful programing language . Java support Object Oriented Programming completely " ; document.write (ex.replace ("Java" , "C#")); </script>	کد
C# is a powerful programing language . Java support Object Oriented Programming completely	خروجی

مثال ۲ : استفاده از پارامتر g

مثال قبل را با اضافه کردن پارامتر g بازنویسی کرده ایم . در این حالت می بینیم که هر دو مورد کلمه Java با C# عوض شده است

Example	
<script type ="text/javascript"> var ex = "Java is a powerful programing language . Java support Object Oriented Programming completely " ; document.write (ex.replace (/Java/g , "C#")); </script>	کد
C# is a powerful programing language . C# support Object Oriented Programming completely	خروجی

مثال ۳ : استفاده از پارامتر i و g به صورت همزمان

مثال قبل را با استفاده همزمان از در پارامتر i و g بازنویسی کرده ایم . در این حالت می بینیم که هر دو مورد کلمه Java با C# عوض شده و همچنین این تابع به بزرگی و کوچکی حروف حساس نیست . این بار در کد دستوری مثال کلمه جاوا را با حروف کوچک و به صورت java نوشته ایم .

Example	
<script type ="text/javascript"> var ex = "Java is a powerful programing language . Java support Object Oriented Programming completely " ; document.write (ex.replace (/java/ig , "C#")); </script>	کد
C# is a powerful programing language . C# suport Object Oriented Programming completely	خروجی

تابع search :

این تابع ، جهت جستجو برای وجود یا عدم وجود یک حرف یا کلمه خاص در یک مغیر رشته ای استفاده می شود . عملکرد این تابع بسیار شبیه تابع match است ، با این تفاوت که این تابع را می توان با پارامتر i به کار برد . به کار بردن پارامتر i با این تابع باعث عدم حساسیت آن ، به بزرگ یا کوچک بودن حروف می شود .

نکته ۱ : کلمه یا حرف مورد نظر توسط خاصیت searchstring تعیین می شود .

نکته ۲ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود داشته باشد ، این تابع شماره اولین کاراکتر آن در طول متغیر رشته ای را برمی گرداند و در صورت عدم وجود حرف یا کلمه مقدار بازگشتی ۱- خواهد بود .

Syntax	str.search (searchstring) * searchstring = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند

مثال : در مثال به وسیله تابع search به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . توضیحات مثال :

۱. در دستور اول شماره مکان قرار گیری اولین مورد حرف e در متغیر برگشت داده شده است.
۲. در دستور دوم شماره مکان قرار گیری اولین کاراکتر کلمه World در متغیر برگشت داده شده است.
۳. در دستور سوم به دلیل حساسیت تابع به بزرگ یا کوچک بودن حروف ، کلمه world در متغیر پیدا نشده و مقدار خروجی ۱- بوده است.
۴. در دستور چهارم به دلیل به کار بردن تابع با پارامتر i ، حساسیت آن نسبت بت بزرگ یا کوچک بودن حروف از بین رفته و شماره مکان قرار گیری اولین کاراکتر کلمه World بر گشت داده شده است.

Example	
<script type ="text/javascript"> var matn = "Hello World" ; document.write (matn.search ("l") + " ") ; document.write (matn.search ("World") + " ") ; document.write (matn.search ("world") + " ") ; document.write (matn.search (/world/i)) ; </script>	کد
2 6 -1 6	خروجی

تابع slice :

از این تابع برای برش و جدا کردن بخشی از متن یک متغیر رشته ای و سپس ذخیره آن در یک متغیر جدید یا چاپ در خروجی استفاده می شود .

شماره کاراکتری که می خواهیم عمل برش از آن آغاز شود را توسط خاصیت start و شماره کاراکتری که می خواهیم عملیات برش در آنجا پایان پذیرد ، را توسط خاصیت end تعیین می کنیم . مجموعه کاراکترهایی که بین این دو مقدار باشند ، به عنوان خروجی نمایش داده می شود .

نکته ۱ : تعیین نقطه پایانی اختیاری است و می تواند تعیین نشود . در صورت عدم تعیین آن ، نقطه پایان برش متغیر ، انتهای آن در نظر گرفته می شود .

نکته ۲ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف V و شماره ۷ حرف C خواهد بود .

نکته ۳ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

Syntax	str.slice (start , end) * start = شماره مکان کاراکتر آغاز قسمت برش * end = شماره مکان کاراکتر انتهای قسمت برش

مثال : در مثال زیر در دو حالت به برش متغیر matn پرداخته ایم . در حالت اول نقطه شروع کارکتر ۳ و نقطه پایان کاراکتر ۱۱ در نظر گرفته شده است . در حالت دوم نقطه شروع کاراکتر ۳ تعیین شده و نقطه پایان مشخص نشده است و در حالت دوم عمل برش تا انتهای متغیر انجام شده است :

	Example

کد	<script type ="text/javascript">
	  var matn = "Hello World!" ;
	  document.write (matn.slice (3 , 11) + "
	 ") ;
	  document.write (matn.slice (3));
	</script>

خروجی	lo Wo lo World!

تابع split :

از این تابع برای تقسیم کردن یک متغیر رشته ای به آرایه ای از متغیرهای رشته ای استفاده می شود . نتیجه خروجی ، شامل قطعات تقسیم شده متغیر رشته ای است که با کاما از هم جدا شده اند .

در این تابع توسط خاصیت seprator ، کاراکتر یا کلمه ای که می خواهیم عمل تقسیم شدن بر مبنای آن صورت بگیرد را تعیین کرده و توسط خاصیت number ، تعداد دفعات تقسیم متغیر رشته ای به قطعات کوچکتر را تعیین می کنیم . برای مثال اگر عدد ۳ وارد شود ، متغیر متنی به ۳ قطعه تقسیم می شود . تعیین این مقدار اختیاری است و می تواند تعیین نشود .

نکته ۱ : اگر خاصیت seprator ، " " تعیین شود ، عمل تقسیم بر حسب کلمات موجود در یک متغیر صورت می گیرد .

نکته ۲ : اگر خاصیت seprator ، "" تعیین شود ، عمل تقسیم بر حسب کلمات حروف در یک متغیر صورت می گیرد .

Syntax	str.split (seprator , number) * seprator = کارکتر یا کلمه ای که می خواهیم عمل تقسیم بر حسب آن انجام شود . * end = تعداد دفعات تقسیم متغیر رشته ای به قطعات کوچکتر را تعیین می کند

مثال : در مثال زیر در چهار حالت به برش متغیر matn توسط تابع split پرداخته ایم . در حالت اول ، تقسیم بر مبنای حرف e ، در حالت دوم تقسیم بر مبنای "" ، در حالت سوم تقسیم بر مبنای " " و تعداد دفعات تقسیم هم ۴ بار در نظر گرفته شده است . حالت اول در آخر هم تقسیم بر حسب حروف متغیر تعیین شده است .

Example	
<script type ="text/javascript"> var matn = "Hello World!" document.write (matn.split ("e") + " ") ; document.write (matn.split ("") + " "); document.write (matn.split ("" , 4) + " "); document.write (matn.split (" ")); </script>	کد
H,llo World! H,e,l,l,o, ,W,o,r,l,d,! H,e,l,l Hello,World!	خروجی

تابع substr :

از این متغیر برای برش و جدا کردن تعداد حروف معینی از یک متغیر متنی ، از یک نقطه مشخص در طول آن استفاده می شود . در این تابع ، نقطه شروع عمل برش را بر حسب شماره یک کاراکتر توسط خاصیت start و تعداد کاراکترهایی که می خواهیم از نقطه شروع برش داده شوند را توسط خاصیت length تعیین می کنیم .

Syntax	str.substr (start , length) * start = شماره کاراکتر آغاز نقطه برش در طول متغیر * length = تعداد کاراکترهایی که می خواهیم از نقطه شروع برش داده شوند

نکته ۱ : تعیین تعداد کاراکترهای مورد نظر برای عملیات برش اختیاری بوده و می تواند تعیین نشود . در صورت عدم تعیین آن ، انتهای متغیر به عنوان نقطه پایان برش در نظر گرفته می شود .

نکته ۲ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف **V** و شماره ۷ حرف **C** خواهد بود .

نکته ۳ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

مثال : در مثال زیر در ۲ حالت به برش متغیر متن پرداخته ایم . در حالت اول عملیات برش را از کاراکتر ۴ و به تعداد ۶ حرف انجام داده ایم . در حالت دوم عملیات برش را از کاراکتر ۴ انجام داده ایم ، ولی تعداد کاراکتر معینی را در نظر نگرفته ایم . در این حالت انتهای متغیر به عنوان نقطه پایان عملیات برش در نظر گرفته شده است :

Example	
<script type ="text/javascript"> var matn = "Hello World!" ; document.write (matn.substr (4 , 3) + " ") ; document.write (matn.substr (4)) ; </script>	کد
o W o World!	خروجی

تابع substring :

از این تابع برای برش و جدا کردن بخشی از یک متغیر رشته ای ، بین دو نقطه مشخص استفاده می شود . در این تابع ، نقطه شروع را توسط خاصیت start و بر حسب شماره یک کاراکتر و نقطه پایان را نیز با خاصیت stop آنهم بر حسب شماره مکان یک کاراکتر دیگر ، در طول متغیر متنی تعیین می کنیم .

Syntax	str.substr (start , stop) * start = شماره کاراکتر آغاز نقطه برش در طول متغیر * length = شماره کاراکتر نقطه پایان عملیات برش

نکته : تعیین نقطه پایان برای عملیات برش اختیاری بوده و می تواند تعیین نشود . در صورت عدم تعیین آن ، انتهای متغیر به عنوان نقطه پایان برش در نظر گرفته می شود .

مثال : در مثال زیر در ۲ حالت به برش متغیر متن پرداخته ایم . در حالت اول عملیات برش را بین کاراکترهای ۴ و ۹ تعیین کرده ایم . در حالت دوم عملیات برش را از کاراکتر ۴ آغاز کرده ایم ، ولی نقطه پایان آن را در نظر نگرفته ایم . در این حالت انتهای متغیر به عنوان نقطه پایان عملیات برش در نظر گرفته شده است :

Example

کد	

<script type ="text/javascript">
  var matn = "Hello World!" ;
  document.write (matn.substrnig (4 , 7) + "
") ;
  document.write (matn.substrnig (4)) ;
</script>	

خروجی	
o W o World!	

تابع toLowerCase :

از این تابع ، برای نمایش تمام متن یک متغیر رشته ای با حروف کوچک استفاده می شود . چنانچه در متغیر رشته ای مورد استفاده ، حرفی به شکل بزرگ باشد ، به صورت اتوماتیک به شکل کوچک آن تبدیل می شود .

Syntax	str.toLowerCase ( )

مثال : در مثال زیر یک متغیر رشته ای را با تابع toLowerCase با حروف کوچک نمایش داده ایم :

Example	

کد	<script type ="text/javascript">
	  var matn = "Hello World!" ;
	  document.write (matn.toLowerCase ( )) ;
	</script>

خروجی	hello world!

تابع toUpperCase :

از این تابع ، برای نمایش تمام متن یک متغیر رشته ای با حروف بزرگ استفاده می شود . چنانچه در متغیر رشته ای مورد استفاده ، حرفی به شکل کوچک باشد ، به صورت اتوماتیک به شکل بزرگ آن تبدیل می شود .

Syntax	str.toUpperCase ( )

مثال : در مثال زیر یک متغیر رشته ای را با تابع toUpperCase با حروف بزرگ نمایش داده ایم :

Example

کد	خروجی
<script type ="text/javascript"> var matn = "Hello World!" ; document.write (matn.toUpperCase ( )) ; </script>	HELLO WORLD!

آرایه در جاوا اسکریپت

مفهوم آرایه : آرایه مجموعه ای از متغیرهایی از یک نوع داده ای با نام یکسان است ، که هر کدام از اعضای آن توسط یک **شمارنده (اندیس)** ، از یکدیگر متمایز می شوند .

شمارنده هر عضو آرایه در یک براکت در مقابل نام آن تعیین شده ، که برای مقدار دهی و دسترسی به هر عضو آرایه از اندیس آن استفاده می شود .

برای تعریف یک آرایه ، از واژه کلیدی new Array به شکل کلی زیر استفاده می شود :

var نام آرایه = new Array ( ) ;

مثال : var Cars = new Array ( ) ;

نکات مهم

نکته ۱ : می توان تعداد اعضای یک آرایه را در زمان تعریف ، در پرانتز جلوی واژه کلیدی new Array ، تعیین کرد :

مثال : آرایه زیر ۴ عضو دارد :

var Cars = new Array (4) ;

نکته ۲ : شماره گذاری اندیس اعضای یک آرایه از صفر شروع شده و برای هر عضو شمارنده یک واحد افزایش می یابد .

مثال : آرایه ای که در مثال قبل ایجاد کردیم ، ۴ عضو زیر را داراست :

Cars [0] , Cars [1] , Cars [2] , Cars [3]

نکته ۳ : برای مقدار دهی یا دسترسی به هر عضو آرایه ، از نام آرایه به همراه شمارنده یا اندیس عضو مورد نظر در براکت مقابل نام آن ، به شکل کلی زیر استفاده می شود :

نام آرایه [اندیس عضو مورد نظر] = مقدار مورد نظر ;

مثال : Cars [0] = "Ford" ;

Cars [1] = "Nissan" ;

مثال : در مثال زیر یک آرایه در یک اسکریپت تعریف شده و اعضای آن مقدار دهی شده اند . در پایان نیز عضو اول و دوم آرایه در خروجی بر روی صفحه چاپ شده اند :

Example

کد	<script type="text/javascript" > var Cars = new Array (3) ; Cars [0] = "Ford" ; Cars [1] = "Nissan" ; Cars [2] = "Mazda" ; document.write (Cars [0]) ; document.write (Cars [1]) ; </script >
خروجی	Ford Nissan

روش های مقدار دهی کلی اعضای یک آرایه

تمام یا بخش هایی از اعصای یک آرایه را می توان در هنگام تعریف و یا بعد از آن مقدار دهی کرد . به طور کلی ۲ روش برای مقدار دهی اعضای یک آرایه وجود دارد :

روش اول : در روش اول ، هر یک از اعضای آرایه را به صورت تکی مقدار دهی می کنیم . در مثال زیر یک ابتدا آرایه ۴ عضوی تعریف شده و سپس مقدار دهی شده است :

Example	
کد	<script type="text/javascript" > var Cars = new Array (4) ; Cars [0] = "Ford" ; Cars [1] = "Nissan" ; Cars [2] = "Mazda" ; Cars [3] = "Volvo" ; </script >

نکته : در صورتی که یک اندیس بیش از مقادیر تعریف شده به آرایه اضافه شود، مشکلی در درج عنصر به وجود نخواهد آمد، ولی تمامی عناصر مابین آنها undefined خواهد بود (در واقع به هدر رفته است). به مثال زیر توجه کنید:

Example	
کد	<script type="text/javascript" > var Cars = new Array (3) ; Cars [0] = "Ford" ; Cars [1] = "Nissan" ; Cars [2] = "Mazda" ; Cars [100] = "Volvo" ; document.write (Cars [100] + " ") ; document.write (Cars [99] + " ") ; document.write (Cars.length) ; </script >

خروجی	Volvo undefined 101

روش دوم : در روش دوم ، مقادیر مورد نظر برای تمام یا تعدادی از اعضای آرایه را در پرانتز جلوی واژه کلیدی new Array تعیین کرده و هر مقدار را با کاما از هم جدا می کنیم . در این حالت تعداد اعضای آرایه به طور اتوماتیک بر حسب تعداد مقادیر ورودی تعیین می شود . در مثال زیر ، آرایه تعریف ۴ عضو خواهد شد . اعضای این آرایه در مرحله تعریف آرایه تعیین شده اند :

Example	
< script type="text/javascript" > var Cars = new Array ("Ford" , "Nissan" , "Mazda" , "Volvo") ; < /script >	کد

روش سوم : در روش سوم ، مقادیر مورد نظر را در براکت تعیین کرده و هر مقدار را با کاما از هم جدا می کنیم . در این حالت تعداد اعضای آرایه به طور اتوماتیک بر حسب تعداد مقادیر ورودی تعیین می شود . در مثال زیر ، آرایه تعریف ۴ عضو خواهد شد . اعضای این آرایه در مرحله تعریف آرایه تعیین شده اند :

Example	
< script type="text/javascript" > var Cars = ["Ford" , "Nissan" , "Mazda" , "Volvo"] ; < /script >	کد

اهم توابع و خواص Array در زیر آمده است:

خاصیت length

این خاصیت، تعداد اعضای آرایه را بر می گرداند . شکل کلی استفاده از این خاصیت به شرح زیر است:

Syntax	a.length

مثال : در مثال زیر یک آرایه را تعریف و سپس به وسیله دستور document.write ، طول آن را نشان داده ایم:

مثال	
<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"]; document.write (a.length) ; </script>	کد

خروجی	4

تابع toString :

این تابع ، اعضای آرایه را با کاما (ویرگول) به هم چسبانده و به عنوان یک رشته بر می‌گرداند.

Syntax	a.toString( )

مثال : در مثال زیر یک آرایه را تعریف و سپس به وسیله دستور document.write ، تبدیل شده‌ی آن به رشته را نشان داده ایم:

مثال	
<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"]; document.write (a.toString()) ; </script>	کد
Banana,Orange,Apple,Mango	خروجی

تابع join :

این تابع ، اعضای آرایه را با یک رشته دلخواه به هم چسبانده و به عنوان یک رشته بر می‌گرداند.

Syntax	a.join( )

مثال : در مثال زیر یک آرایه را تعریف و سپس به وسیله دستور document.write ، join شده‌ی آن به رشته را نشان داده ایم:

مثال	
<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"]; document.write (a.join(" and ")) ; </script>	کد
Banana and Orange and Apple and Mango	خروجی

تابع push :

این تابع ، یک عضو به انتهای آرایه اضافه می‌کند.

Syntax	a.push(x) * x = عضو جدید که باید به انتهای آرایه اضافه شود را مشخص می کند

مثال : در مثال زیر یک آرایه را تعریف کرده و سپس یک عضو به انتهای آن اضافه نموده ایم:

	مثال
کد	<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"]; a.push("Kiwi") ; document.write (a.toString()) ; </script>
خروجی	Banana,Orange,Apple,Mango,Kiwi

نکته : اثر این دستور دقیقاً مثل آن است که بنویسیم ;"a[a.length]="Kiwi، زیرا به این ترتیب به انتهای آرایه عضو جدیدی اضافه می‌شود.

تابع pop :

این تابع ، یک عضو از انتهای آرایه حذف کرده و آن را بر می‌گرداند.

Syntax	a.pop()

مثال : در مثال زیر یک آرایه را تعریف کرده و سپس یک عضو از انتهای آن برداشته و آن را چاپ نموده ایم:

	مثال
کد	<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"]; document.write (a.pop() + " ") ;

document.write (a.length) ; </script>	
Mango 3	خروجی

تابع sort :

این تابع ، آرایه را به صورت صعودی مرتب می‌کند.

Syntax	a.sort()

مثال : در مثال زیر یک آرایه را تعریف کرده و سپس اعضای آن را مرتب و چاپ نموده ایم:

مثال	

<script type="text/javascript">
var a = ["Banana", "Orange", "Apple", "Mango"];
a.sort();
document.write (a.toString()) ;
</script>
	کد

Apple,Banana,Mango,Orange	خروجی

تابع reverse :

این تابع ، ترتیب عناصر آرایه را معکوس می‌کند.

Syntax	a.reverse()

مثال : در مثال زیر یک آرایه را تعریف کرده و سپس اعضای آن را معکوس و چاپ نموده ایم:

مثال	
<script type="text/javascript"> var a = ["Banana", "Orange", "Apple", "Mango"];	کد

a.reverse(); document.write (a.toString()) ; </script>	
Mango,Apple,Orange,Banana	خروجی

نکته : می توان با استفاده از دستور sort و reverse پشت سر هم، آرایه را به‌صورت نزولی مرتب کرد.

تابع slice :

این تابع ، زیر مجموعه ای از عناصر آرایه را بر می‌گرداند.

Syntax	a.slice(i, j) * i = اندیس عضوی که زیر مجموعه از آن شروع می شود را مشخص می کند * j = اندیس عضوی که زیر مجموعه به آن ختم می شود را مشخص می کند

مثال : در مثال زیر یک آرایه را تعریف کرده و سپس زیر مجموعه ای از اعضای آن را انتخاب و چاپ نموده ایم:

مثال	

<script type="text/javascript">
var a = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var b = a.slice(1,4);
document.write (b.toString()) ;
</script>
	کد

Orange,Lemon,Apple	خروجی