آموزش
JavaScript

قسمت اول

مقدمه :

Script ها در واقع کدهایی هستند که درون فایل HTML از طریق برچسب <script> تعریف می‌شوند. یکی از پرکاربرد ترین زبانهای اسکریپت نویسی دنیا javascript است که در میلیونها صفحه web موجود در اینترنت بکار رفته است. Javascript برای توسعه طراحی صفحات، بررسی صحت اطلاعات و کنترل فرمها، تشخیص نوع مرورگر، ایجاد فایلهای مخفی جهت ردیابی فعالیت کاربر (cookie) و بسیاری از عملیات پیشرفته دیگر کاربرد دارد.

همچنین این زبان اسکریپت نویسی توسط همه مرورگرها منجمله اینترنت اکسپلورر ، نت اسکیپ ، فایر فاکس ، موزیلا ، اپرا و غیره پشتیبانی می‌شود.

پیش از یادگیری java script لازم است با HTML و XHTML آشنا گردید .

بطور خلاصه java script :

- ✓ برای متعامل کردن (Interactive) صفحات HTML بکار می رود و آنها را از حالت یکطرفه خارج می‌کند .
- ✓ یک زبان اسکریپتی سبک برای صفحات وب است که قدرت زیادی دارد و تقریبا هر خواسته‌ای را برآورده می‌کند .

- ✓ شامل یکسری کدهای اجرایی کامپیوتر است که درون HTML قرار می گیرد و زمان کمتری برای اجرا نیاز دارد پس بسیار سریع است .
- ✓ یک زبان تفسیری است یعنی بدون احتیاج به ترجمه و زمان بری کار می کند .
- ✓ هر فردی می تواند آنرا بطور رایگان درون صفحات خود بکار ببرد .

حال سوالی که مطرح می شود این است که آیا java script همان java است؟ در پاسخ به این سوال باید گفت که java و java script دو زبان برنامه نویسی کاملا متفاوت هستند که در سبک طراحی و استفاده هر کدام اختلاف زیادی وجود دارد .
در واقع جاوا که محصول کمپانی Sun Microsystems است و مانند زبانهای c , + +c از قدرت و انعطاف پذیری زیادی در برنامه نویسی برخوردار می باشد .

در مجموع کاربردهای java script به شرح زیر است :

- ✓ به عنوان یک ابزار برنامه نویسی برای طراحان HTML استفاده می شود . طراحان HTML به طور عادی برنامه نویس نیستند اما java script با قواعد ساده اش به گونه ای است که هر طراحی با اندک مطالعه و تمرین توانایی افزودن کدهای برنامه نویسی به صفحات خود را خواهد داشت
- ✓ با استفاده از java script می توان متن پویا و قابل تغییر در صفحات قرار داد . مثلا دستور زیر

document.write(“ <h1> “+name+” , </h1> “)

می تواند متغیر name را که مقدارش در هر بار تغییر می کند در صفحه چاپ نماید .

- ✓ Java script قدرت کنترل رخدادها را درون صفحات HTML دارا است. یک کد java script می تواند به گونه ای تنظیم شود که در هنگام یک عمل خاص مثل زدن یک دکمه ، باز شدن صفحه و یا کلیک کاربر روی یک المنت اجرا شود .
- ✓ Java script می تواند المنتهای HTML را بخواند یا بنویسد. در واقع توسط java script می توان محتوی یک المنت روی صفحه را خواند و تغییراتی روی آن به وجود آورد .
- ✓ Java script می تواند برای بررسی صحت اطلاعات وارد شده توسط کاربر بکار رود. در بسیاری از فرمهای ورود اطلاعات کاربر، کنترل صحت اطلاعاتی نظیر کدپستی ، پست الکترونیک email ، شماره کد ملی ، کارت اعتباری ، تاریخ و غیره که از قالب خاصی پیروی می کند یا کد کاربری و رمز عبور که بایستی از ترکیب حروف و ارقام باشد بسیار ضروری است و خطاهای برنامه را بشدت کاهش می دهد .
- ✓ java script قادر است نوع مرورگر بیننده را تشخیص دهد و متناسب با آن محتوی صفحه HTMLرا نمایش دهد در نتیجه صفحات HTML شما، سازگاری زیادی با مرورگرهای مختلف اینترنت خواهد داشت .

- ✓ java script می تواند برای ایجاد فایلهای ردیابی cookie درون سیستم کاربر بکار برده شود . این کار باعث می شود که صفحات اطلاعاتی را در کامپیوتر کاربر ذخیره کنند که برای ردگیری عملیات انجام شده وبرای بخاطر سپاری افراد بسیار مفید است و باعث می گردد به اصطلاح صفحات web هوشمندانه عمل نمایند .

چگونه java script را بکار بریم :

کدهای java script از طریق برچسب < script > در HTML و X HTML به صفحات افزوده می‌شود. به مثال زیر توجه کنید :

```
<html>
<body>
   <script  type = " text / java script " >
       document.write("hello")
   </script>
</body>
</html>
```

همانطور که مشاهده می شود در این کد دستور document.write که برای نوشتن روی صفحه جاری بکار می رود ، درون برچسب script قرار داده شده لذا مرورگر با رسیدن به این برچسب ،دستور فوق را اجرا نموده و پیغام hello روی صفحه ظاهر می گردد .

دقت کنید که در هنگام استفاده از برچسب < script > ، حتما صفت type با مقدار text / java script بکار برده شود تا مرورگر تشخیص دهد درون script چه نوع کدی برای اجرا وجود دارد .

چنانچه دستورات java script را درون برچسب < script > نگذارید مرورگر با این دستورات مانند متن ساده برخورد می کند و بجای اجرای دستور ، کل آن را چاپ می کند .

به جهت آنکه مرورگرهای قدیمی تر که نمی توانند کدهای java script را تشخیص دهند ، کدها را روی صفحه نمایش ندهند، بایدکل دستورات j script را درون برچسب comment (< - -// , - -!>) قرار داد :

```
<script  type = " text / java script " >
<!- -
    document.write("hello")
//- ->
</script>
```

دو علامت // که در ابتدای برچسب comment آمده در java script جهت درج توضیحات بکار می رود و به مرورگر می گوید این خط را ترجمه ننماید . نکته دیگری که در مورد java script لازم است بدانید این است که در این زبان احتیاجی به قرار دادن (;) در انتهای دستورات نیست منتهی بعضی کاربران براساس کار با زبانهای برنامه نویسی نظیر java و c + + اقدام به ختم دستورات با (;) می کنند . برای همین گذاردن (;) در انتهای دستوات j script اختیاری است و تأثیری در عمل دستورات ندارد . در صورتی که

بخواهید بیش از یک دستور را در یک خط بگذارید لازم است برای جداسازی دستورات از (;) استفاده کنید .

تمرین : نوشتن یک متن با قالب بندی خاص توسط java script .

```
<html>
<body>

<script type="text/javascript">
document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

نکته مهم : زبان java script یک زبان case – sensitive است یعنی در واقع بین حروف بزرگ و کوچک انگلیسی تفاوت قائل می شود بنابراین در تایپ دستورات باید دقت لازم را داشته باشید و حروف کوچک و بزرگ را رعایت کنید .

Java script را در کجا قرار دهیم ؟

با توجه به ترتیب تفسیر برچسبها در HTML مکان قرار گرفتن برچسب< script > برای تعیین زمان اجرایی کدها مهم است کدهای j script که در بدنه <body> استفاده می شود هنگامی اجرا خواهد شد که صفحه به مرورگر منتقل (load) گردد . اما کدهایی که در قسمت <head> قرار می گیرند تنها زمانی که فراخوانی (call) شوند ، اجرا خواهند شد .

در واقع اینکه کدهایj script بعد از load صفحه در مرورگر اجرا شوند ، خواسته بسیاری از طراحان نمی باشد چرا که اغلب طراحان بدنبال اجرای برخی دستورات یا در هنگام load صفحه می باشند و یا برخی از کدها باید توسط یک رخداد کاربر مثل کلیک روی یک المنت یا غیره صدا زده شوند تا حالت متعامل (interactive) به صفحه داده شود .

در حالتی که script در head صفحه واقع شود تنها زمانی اجرا می شود که فراخوانی شده یا رخدادی اتفاق بیفتد . در این حالت می توانید مطمئن باشید کهj script قبل از استفاده به صفحه مرورگر انتقال یافته است .

```
<head>
<script  type = " text / java script " >
        —
        —
        —
</script>  </head>
```

در حالتی که کدهای در jscript در قسمت body قرار گیرند در واقع هنگام بار شدن صفحه به مرورگر این کدها محتوی صفحه را تعیین می کنند به عنوان مثال فرض کنید صفحه ای می خواهد اطلاعات فردی

یک دانشجو را نمایش دهد . شما می توانید این صفحه را در HTML بسازید و بجای مقادیر مانند نام و نام خانوادگی ، آدرس و غیره با استفاده از j script داده های متغیر قرار دهید .

```
<body>
<script  type = " text / java script " >
        —
        —
        —
</script>
</body>
```

در برخی از موارد شما می توانید j script تماما هم در قسمت head و هم در داخل body به کار ببرید . در این صورت هم هر کـدام از اسـکریپتها بسته به محل قـرار گرفتن خود عمل خواهند نمود . .همچنین می توانید کدهای j script را خارج از فایل HTML استفاده کنید . در مواقعی که لازم است کد خاصی در صفحات متعدد اجرا شود نیازی نیست که کد j script را درون هر صفحه تکرار کنیم بلکه با قراردادن کد ها درون فایل متنی ساده با پسوند **.js** و استفاده از صفت src در برچسب <script> کدهای j script به فایل HTML الصاق خواهد شد .

به مثال زیر توجه کنید :

فایلی متنی با محتوی زیر به نام test.js بسازید:

```
document.write(" this is a external javascript " )
```

فایلی متنی با محتوی زیر به نام test.htm بسازید که حاوی کدهایhtml باشد:

```
<html>
<body>
<script src="test.js"> </script>
</body>
```

هنگامی که فایل test.htm به مرورگر بار می شود فایل test.js در محلی که توسط src مشخص شده باز شده و دستور درون آن اجرا می گردد در نتیجه پیغام this is a external javascript روی مرورگر دیده خواهد شد .

** تمرین : در این مثال اسکریپتی درون قسمت <head> قرارداده شده به گونه ای که می توانیم مطمئن باشیم اسکریپ قبل از صدازدن تابع تعریف شده به مرورگر منتقل شده ست :

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event")
}
</script>
</head>

<body onload="message()">

</body>
</html>
```

** تمرین : در این مثال اسکریپت درون بدنه HTML قرار داده شده و بعد از load صفحه نتیجه اجرای دستور به عنوان محتوی ظاهر می گردد .

```
<html>
<head>
</head>

<body>

<script type="text/javascript">
document.write("This message is written when the page
loads")
</script>

</body>
</html>
```

** تمرین : مثال زیر نحوه استفاده از یک فایل j script بیرونی را که به فایل html الحاق شده است نمایش می دهد .

```
<html>
<head>
</head>
<body>

<script src="xxx.js">
</script>

<p>
The actual script is in an external script file
called "xxx.js".
</p>

</body>
</html>
```

محتوی xxx.js به شرح زیر است :

```
document.write(' this script is external ! ')
```

متغیر ها در javascript :

یک متغیر فضایی از حافظه است که مقادیر را در خود ذخیره می کند . مقدار ذخیره شده در یک متغیر در طی اجرای دستورات اسکریپت در حال تغییر خواهد بود . در واقع متغیر فضایی از حافظه را در اختیار برنامه نویسان می گذارد تا با قرار دادن مقادیر دلخواه درون آن اهداف خود را پیاده سازی کند . دسترسی به متغیر توسط نام آن میسر خواهد بود .

برای نام گذاری متغیرها در java script دو قاعده بسیار مهم وجود دارد :

۱- نام متغیر case – sensitive است یعنی بین حروف کوچک و بزرگ تفاوت قائل می شود . به عنوان مثال متغیر A با متغیر a تفاوت دارد و java script هردو را متغیر جداگانه می داند .

۲- نام متغیر حتما باید با یک کاراکتر حرفی (a-z) یا علامت زیر خط (—) شروع شود و شروع نام متغیر با عدد اشتباه است . مثلا نام a1 یا a– صحیح است اما نام 1a اشتباه می باشد .

*** نحوه تعریف متغیر در jscript :**

برای تعریف متغیر ها درjava script به دو شکل زیر می توان عمل نمود :

Var نام متغیر = مقدار

نام متغیر = مقدار

به عنوان مثال :

```
Var strname = "ali"
Strname = "ali"
```

تعریف متغیر به هر کدام از صورتهای فوق تفاوتی ندارد و استفاده از کلمه var در ابتدای تعریف اختیاری است . البته وجود var به مرورگر اعلان می کند که متغیری در حافظه ایجاد نماید و این برای کار با انواع مرورگر ها معمولا مناسبتر ست .

برای مقدار دهی به متغیر ها همیشه باید نام متغیر در سمت چپ علامت انتساب (=) و مقدار آن که عددی یا حرفی است در سمت راست قرار گیرد ، توجه کنید که مقدار حرفی باید درون کوتیشن (") قرار گیرند .

***محدوده اعتبار یک متغیر :**

زمانی که یک متغیر را تعریف می کنید بسته به این که تعریف متغیر در کجا واقع شده ، اعتبار متغیر متفاوت است . مثلا متغیری که درون یک تابع (function) تعریف شود فقط درون آن قابل دسترسی است و هنگامی که مفسر دستورات تابع را اجرا نمود و از آن خارج شد ، آن تابع هم پاک می شود . به این متغیرهاکه محدود به ناحیه خاصی از کد مثل تابع هستند، متغیر محلی (local) گفته می شود . بطور کلی می توان متغیر محلی را با همان نام در توابع دیگر نیز تعریف کرد . به عنوان مثال اگر در تابعی از متغیری به نام Name استفاده می کنید به شرط آنکه Name درون تابع بصورت local باشد، می توانیددر تابع دیگری نیز از نام Name به عنوان متغیر استفاده کنید و این مسئله هیچ تداخلی را به وجود نمی آورد چرا که Name زمانی در حافظه شکل می گیرد که مفسر j script درون تابع باشد .

این موضوع برای متغیرهایی که خارج از تابع تعریف می شوند، فرق می کند . متغیرهایی که خارج از تابع تعریف می شوند به متغیرهای عمومی (global) معروف هستند و در تمام توابع و قسمتهای برنامه متغیر خواهند بود . این متغیرها از لحظه ای که تعریف می شوند تا زمانی که صفحه بسته می شود اعتبار دارد و تمامی توابع و دستورات می توانند از این متغیرها استفاده کنند .

قسمت دوم

ساختار شرط درjavascript :

جملات شرطی در java script برای پیاده سازی فعالیتهای مختلف بر اساس شروط متفاوت بکار می رود . به طور کلی یکی از مهمترین ساختارهای برنامه نویسی پیاده سازی شرطی است که این مهم از طریق جملات if…else در محیط java script امکان پذیر است . از java script به طور کلی چهار نوع ساختار تصمیم گیری وجود دارد که سه نوع آن توسط if و نوع چهارم توسط دستور switch قابل اجرا است :

✓ **جمله شرطی یکتا (if statement) :**

در هنگامی از ساختار if یکتا استفاده می شود که بخواهیم در صورت درست بودن شرطی کاری را انجام دهیم و به قولی شرط ما یک حالته باشد .

```
if   ( شرط )
{
   دستورات
}
```

✓ **جمله شرطی دو حالته (if…else statement) :**

هنگامی که در برنامه بخواهیم در صورت درست یا نادرست بودن یک شرط به تفکیک یک عمل خاص انجام پذیرد یعنی در صورتی که شرط درست باشد یک عمل خاص و در غیر اینصورت کار دیگری انجام بپذیرد از ساختار if…else به صورت زیر استفاده می گردد :

```
if   ( شرط )
     {
     دستورات حالت درست بودن
     }
else
     {
     دستورات حالت نادرست بودن
     }
```

✓ **جمله شرطی لانه ای (if…else if…else statement)** :

در حالتی که بخواهید بین چند بلوک برنامه یکی را براساس شروط انتخاب کنید می توانید با پیاده سازی شروط تو در تو و وابسته به هم این عمل رابراحتی انجام دهید . منتهی باید دقت کنید که چیدن شروط داخل هم با ظرافت خاصی صورت گیرد تا حالتهای مختلف را پوشش دهد و ساختار درست عمل نماید .

```
If
  {

      دستورات
  }
elseif
  {

      دستورات
  }
else
  {
      دستورات
  }
```

ساختار سوئیچ (switch statement) :

این ساختار امکان انتخاب یکی از بلوکهای دستورات برنامه را بر اساس مقادیر مختلف برای یک متغیر حالت خاص را ایجاد می کند . در واقع شما با استفاده از این ساختار قادر خواهید بود برای مقادیر متفاوت یک متغیر دستورات خاص را اجرا کنید که در قسمت بعدی به آن پرداخته می شود .

دقت کنید دستوراتی که بعد از if یا else قرار می گیرند حتما باید بین دو علامت {} واقع شوند وگرنه شرط if یا else اعمال نمی شود . همچنین چنانچه دستورif یا else را با حروف انگلیسی بزرگ بنویسید با خطا مواجه خواهید شد . حال به مثالهای زیر برای درک بهتر ساختارهای شرطی دقت کنید . (در مثالها حروف بزرگ و کوچک را رعایت کنید)

تمرین : if statement کد زیر را دریافت نموده و چنانچه قبل از ۱۰ صبح باشد پیام good morning را چاپ می کند

```
<script  type = " text / java script " >
var  d=new Date ( )
var time = diget Hours ( )
If (time <10 )
{
document.write ( "<b> good morning </b> ")
```

```
}
</script>
```

تمرین : if…else statement : در مثال زیر چنانچه ساعت کمتر از ۱۰ صبح باشد پیام good morning و در غیر اینصورت پیام good day ظاهر می گردد :

```
<script  type = " text / javascript " >
var  d=new Date ( )
var time = diget Hours ( )
if (time <10 )  {
document.write ( "good morning  ")
}
else
{
document.write ( "good day ")
}
</script>
```

تمرین : در مثال زیر یک نمونه کاربرد ساختار سوئیچ نمایش داده شده است . در این کد سیستم با شناسایی نام روز جاری پیام مناسبی صادر می کند . . برای روز یکشنبه Sunday مقدار ۰ ، روز دوشنبه Monday مقدار ۱ و روز سه شنبه thesday مقدار ۲ در نظر گرفته شده .

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
theDay=d.getDay()
switch (theDay)
{
case 5:
  document.write("<b>Finally Friday</b>")
  break
case 6:
  document.write("<b>Super Saturday</b>")
  break
case 0:
  document.write("<b>Sleepy Sunday</b>")
  break
default:
  document.write("<b>I'm really looking
forward to this weekend!</b>")
}
</script>

<p>This JavaScript will generate a
different greeting based on what day it is.
Note that Sunday=0, Monday=1, Tuesday=2,
etc.</p>

</body>
</html>
```

نکته مهم : در ساختار switch همانطور که مشاهده می شود در انتهای هر بلوک case یک دستور bteak در پایان دستورات case قرار نگیرد ، دستورات case بعدی بدون در نظر گرفتن مقدار آن اجرا می شوند . به مثال زیر توجه کنید :

```
Switch  (n)
{
    Case 1 :
          dacument.write("Case 1")
     Case 2 :
          dacument.write("Case 2")
}
```

در کد بالا به جهت آنکه در انتهای بلوک Case 1 دستور break نیامده چنانچه تعداد n=1 باشد آنگاه هم دستورات بلوک Case 1 اجرا می شود و هم دستور بلوک Case 2 که این اشتباه است . بنابرین باید دقت کنیم که درپایان دستورات یک بلوک و قبل از caseبعدی دستور break قرار گیرد .

البته این موضوع در برخی از موارد به کمک برنامه نویس می آید . در مواقعی که بخواهیم برای چند مقدار متفاوت یک متغیر ، دستورات یکسانی را بکار بریم کافی است بلوک های case مربوط به مقادیر همسان را بدون break پشت سرهم قرار دهیم . مثلا اگر بخواهیم برای مقادیر ۸و۹و۱۰ که مربوط به متغیر time می باشد عبارت Good morning روی صفحه چاپ شود و برای مقادیر ۱۳و۱۴و۱۵ عبارت Good evening : آنگاه به صورت زیر عمل می کنیم :

```
Switch (time)  {
      case  8 :
      case  9 :
      case  10 :
          document.write(“Good morning”)
          break
     case 13 :
     case 14 :
     case 15 :
          document.write(“Good evening”)
          break
}
```

نکته دیگری را در مورد switch باید بدانید این است که عبارت default در ساختار switch برای مواقعی کار برد دارد که بخواهیم برای سایر مقادیر احتمالی یک متغیر که درون case ها مشخص نشده اند برنامه ریزی کنیم مثلا اگر بخواهیم مثال قبل را طوری طرح کنیم که علاوه به موارد فوق به ازای سایر ساعات روز پیام good day . را چاپ نماید بصورت زیر عمل می کنیم :

```
Switch (time)  {
      case  8 :
      case  9 :
      case  10 :
              document.write(“Good morning”)
```

```
            break
    case 13 :
    case 14 :
    case 15 :
            document.write("Good evening")
            break
default :
            document.write("Good day")
}
```

عملگر ها در java script :

عملگر ها یا operators در هر زبان برنامه نویسی علائمی هستند که برای مفسر زبان به صورت خاصی تفسیر می گردند و در نتیجه باعث می شوند مفسر عملیات خاصی نظیر عملیات ریاضی ، مقایسه ای ، نسبت دهی و غیره را انجام دهد . به عبارت زیر توجه کنید :

5 + 2 = 7

در این عبارت به علامت + و == که به معنای جمع و حاصل عملیات می باشد operator یا عملگر و به مقادیر ۵ و ۲ و ۷ علوند یا operator گویند .

به طور کلی عملگر ها در java script به شش دسته زیر تقسیم می شوند :

✓ **عملگرهای محاسباتی Arithmetic perators** :

این عملگر ها برای انجام عملیات ریاضی و محاسباتی نظیر جمع ، تفریق ، ضرب ، تقسیم و افزایش یا کاهش مقدار متغیر ها بکار می روند و برای آنها باید از عملوند های عددی استفاده نمود ، جدول زیر انواع این عملگرها را با ذکر مثال شرح می دهد :

operator	شرح	مثال	نتیجه
+	جمع (Addition)	X=2 Y=2 X+y	4
_	تفریق (Subtraction)	X=5 Y=2	3
*	ضرب (Multiplication)	X=5 Y=4 X*y	20
/	تقسیم (Division)	15/5 5/2	3 2.5
%	باقی مانده تقسیم (Division reminders)	5%2 10%2 10%8	1 0 2
+ +	افزایش یک واحد به متغیر	X=5	X = 6

	(Increment)	X+ +	
_ _	کاهش یک واحد ار متغیر (Decrement)	X=5 X= - -	X = 4

✓ **عملگر انتسابی Assignment operators :**

این عملگر ها برای انتساب مقادیر کاربرد دارد . در این عملگر ها مقدار سمت راست عملگر با در نظر گرفتن نوع آن به متغیر سمت چپ منتقل می گردد ، لذا در سمت چپ عملگر همیشه باید از متغیر استفاده نمود و در سمت راست می توانید از متغیر یا مقدار ثابت استفاده کنید این عملگر با علوند عددی مورد استفاده قرار می گیرد. به جدول زیر توجه کنید :

operator	شرح	مثال	جمله مشابه
=	انتساب ساده	X=y	X=y
+ =	جمع دو مقدار به طوری که حاصل در متغیر سمت چپ قرار گیرد	X + =y	X=x+y
- =	تفریق دو مقدار به طوری که حاصل در متغیر سمت چپ قرار گیرد	x- =y	X=x-y
* =	ضرب دو مقدار به طوری که حاصل در متغیر سمت چپ قرار گیرد	X* = y	X=x*y
/ =	تقسیم دو مقدار به طوری که حاصل در متغیر سمت چپ قرار گیرد	x/ = y	X=x/y
% =	محاسبه باقی مانده تقسیم دو مقدار به طوری که حاصل در متغیر سمت چپ قرار گیرد	X% = y	X=x%y

✓ **عملگر مقایسه ای (comparison operators) :**

این عملگر ها همانطور که از نامش پیداست جهت مقایسه دو مقدار با متغیر بکار می رود و حاصل آن معمولا عبارت پولی (درست یا نادرست) خواهد بود و درون شرطها کاربرد دارد . توجه داشته باشید که برای این عملگر ها می توانید ار عملوند های عددی یا کاراکتری استفاده کنید . جدول زیر این عملگر ها را نمایش می دهد :

operator	شرح	مثال
= =	مقایسه برابری دو مقدار	X= = y (آیا x برابر y است ؟) 8 = = 5 مقدار false را بر می گرداند
= = =	مقایسه برابری دو مقدار هم از لحاظ نوع هم از لحاظ مقدار	X = 5 Y = "5" X = = y چون مقدار x با y برابر است حاصل true می باشد X= = = y مقدار x با y برابر است اما نوع آن فرق می کند پس حاصل false خواهد بود .
! =	نا مساوی	8 = !5 حاصل true است . 8 = ! 8 حاصل false است .
>	بزرگتر از	8 > 5 حاصل false است 5 >8 حاصل true است .
<	کوچکتر از	8>5 حاصل true است .
> =	بزرگتر مساوی	8 = >5 حاصل false است
< =	کوچکتر مساوی	8 = <5 حاصل true است .

✓ **عملگر های منطقی (logical operators) :**

این عملگر ها برای پیاده سازی جملات منطقی و ترکیبهای شرطی کاربرد دارد و شامل سه تر کیب شرطی and ، or و not است که به ترتیب با علائم && ، || و ! مشخص می شوند . جدول حاصل این سه ترکیب به صورت زیر است :

x	y	x&&y
F	F	F
F	T	F
T	F	F
T	T	T

X=6
Y=3
(x<10 && y>1) → true

x	y	x || y
F	F	F
F	T	T
T	F	T
T	T	T

x=6
y=3
(x = = 5 || y = = 5) → false

x	!x

x=6
y=3
! (x = = y) → true

F	T
T	F

✓ عملگر رشته ای (string operator) :

این عملگر برای پیوند در عبارت رشته ای به یکدیگر استفاده می شود و رشته دوم را به انتهای رشته اول وصل می کند . نکته ای که باید دقت کنید این است که این عملگر مشابه + است ولی با مقادیر رشته ای کار می کند .

به مثال زیر توجه کنید :

Txt1=”what a nice “
Txt 2 = “day”
Txt 3 = txt 1 + txt 2 → “ whate a nice day “

همانطور که می بینید متغیر txt 3 حاصل پیوند txt2 و txt 1 است و بین دو رشته متصل شده فاصله وجود نخواهد داشت . چنانچه مایلید بین دو رشته متصل شونده فاصله ای درج شوند یا باید در هنگام اتصال یک فاصله خالی درج کنید یا در انتهای رشته اول یک فاصله اضافی بگذارید :

Txt3=txt1+” “ +txt2 → “ whate a nice day “

✓ عملگر شرطی (conditional operator) :

این عملگر در واقع نوعی ساختار شرطی است بدین شکل که بر اساس حاصل شرط تعیین شده در عملگر یکی از مقادیر مشخص شده به متغیر منتسب می گردد . شکل استفاده از این عملگر به صورت زیر است :

نام متغیر = (شرط) ؟ مقدار اول : مقدار دوم

در این نوع عملگر اگر شرط درست باشد مقدار اول به متغیر منتسب می گردد و اگر شرط نادرست باشد مقدار دوم در متغیر قرار داده می شود و به نوعی این نوع عملگر شبیه دستور if … else عمل می نماید :

x = 5
y = 8
z = (x>y) ? 20:30 → z=30

قسمت سوم

کادر popup در javascript (popup boxes)

مفهوم popup در نرم افزار عبارت است پنجره کوچکی که زیر مجموعه پنجره اصلی محسوب می شود و در مواقعی مانند اخطار یا دریافت تأیید کاربر یا اعلام یک مطلب استفاده می گردد . در javascript سه نوع کادر popup وجود دارد :

- ✓ Alert box (کادر اخطار) :

یک کادر اخطار هنگامی استفاده می شود که می خواهیم مطمئن شویم اطلاعات به کاربر منتقل شده است و یا هنگامی که خطایی رخ می دهد یا برای انجام عملی باید کاربر را از مضرات آن مطلع سازیم . کادر اخطار شامل متنی است که به عنوان پیغام برای آن تأمین می کنید و یک دکمه ok بمنظور تأیید دریافت پیام اخطار و ناپدید شدن کادر از صفحه . دستور ایجاد کادر اخطار به فرم زیر است :

```
alert ( "پیام اخطار" )
<script type = “text/ javascript >
    Alert ( “this is a alert box”)
</script>
```

Confirm box (کادر تأیید) :

کادر تأیید زمانی استفاده می شود که بخواهیم کاربر انجام یک مطلب یا ارسال اطلاعات را تأیید نماید . در کادر تأیید علاوه بر متن پیام تأیید و محو نمودن کادر و دکمه cancel برای انصراف و عدم تأیید وجود دارد . اگر کاربر ok را کلیک کند ، کادر مقدار true را بر می گرداند و در غیر اینصورت کادر مقدار false را برگشت می دهد .

دستور ایجاد یک کادر تأیید به شکل زیر است :

```
confirm ( " پیام ثانیه ")
```

دستور confirm باید به عنوان مقدار یک متغیر بکار برده شود تا بر اساس مقدار متغیر متوجه شویم کاربر ok یا cancel را انتخاب نموده به مثال زیر توجه کنید :

```
<script type = “text/ javascript >
    Var  A = confirm (“press a key “ )
    if  (A= = true)
          {
          =
       }
 </script>
```

کادر اعلان (prompt box) :

یک کادر اعلان زمانی استفاده می شود که کاربر باید قبل از ورود به یک صفحه مقداری را واردکند . یه کادر اعلان شامل پیام اعلان مقدار پیش فرض ورودی ، کادر محاوره ای برای ورود و مقدار ، دکمه ok و دکمه cancle خواهد بود . دستور ایجاد کادر اعلان به فرم زیر است :

prompt (" پیام اعلان " و " مقدار پیش فرض ")

کادر اعلان هم مانند کادر تأیید باید در انتساب به یک متغیر استفاده شود تا مقدار داده شده توسط کاربر به متغیر انتساب یابد . به مثال زیر توجه کنید :

```
<script type = “text/ javascript >
    Var A=prompt (“input a value” , “10” )
     if (A ! = null && A! = “  “ )
        {
        Document.write (“your input : “ + A )
        }
</script>
```

تمرین : در این مثال با نحوه ایجاد یک کادر اخطار زمانی که دکمه ای در صفحه را کلیک می کنیم آشنا می شوید . همانطور که دیده می شود تابع disp-alert ( ) در رویداد on click کلید توسط برچسب input تعریف شده در نتیجه با کلیک کردن دکمه روی صفحه تابع اجرا شده و دستور alert اجرا می شود :

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("I am an alert box!!")
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_alert()"
value="Display alert box">
</form>
</body>

</html>
```

تمرین : ایجاد یک کادر اخطار با پیام چند خطی همانطور که می دانید پیامهایی که در کادرهای popup قرار می گیرد در یک خط به نمایش در می آید و چنانچه مایلیدپیام یک کادر در چند خط نمایش داده شد و باید در پایان هر خط عبارت 1n را بعنوان علامت خط جدید قرار دهید به کد زیر توجه کنید :

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("Hello again! This is how we" + '\n'
+ "add line breaks to an alert box!")
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_alert()"
value="Display alert box">
</form>
</body>

</html>
```

تمرین : به نحوه نمایش یک کادر تأیید و تشخیص اینکه کاربر کدام کلید را فشرده است در این تمرین لحاظ شده است :

```
<html>
<head>
<script type="text/javascript">
function disp_confirm()
{
var name=confirm("Press a button")
if (name==true)
{
document.write("You pressed the OK button!")
}
else
{
document.write("You pressed the Cancel
button!")
}
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_confirm
()" value="Display a confirm box">
</form>
</body>

</html>
```

همانطور که مشاهده می شود تابع disp – confirm در رویداد onclick دکمه قرار گرفته و درون آن در یک دستور تعریف متغیر با استفاده از دستور confirm یک کادر تأیید ایجاد می شود آنگاه درون ساختار if اگر مقدار برگردانده شده از کادر تأیید به متغیر name برابر true باشد کاربر کلید ok را فشرده و در غیر اینصورت کاربر کلید cancle را فشرده است .

تمرین : در این کد طریقه ایجاد یک کادر اعلان و نمایش مقدار وارد شده توسط کاربر در کادر اعلان به نمایش گذاشته شده است :

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
var name=prompt("Please enter your name","")
if (name!=null && name!="")
{
document.write("Hello " + name + "! How are
you today?")
}
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_prompt
()" value="Display a prompt box">
</form>
</body>

</html>
```

همانطور که می بینید تابع disp – prompt که در رویداد onclick دکمه قرار گرفته بعد از کلیک دکمه اجرا می شود و درون آن کادر اعلان تعریف شده دردستور تعریف متغیر نمایش داده می شود و مقدار پیش فرض آن تهی است ، حال اگر کاربر مقدار جدیدی غیر از تهی وارد کند دکمه ok را بزند این مقدار در تابع توسط دستور document نمایش داده می شود .

توابع درjavascript (JS functions) :

توابع در واقع بلوکهایی از برنامه هستند که به طور جداگانه تحت عنوان یک نام تعریف شده اند و قابلیت استفاده مجدد از آنها در صفحات متعدد وجود دارد . شما در برنامه با یک طرح ریزی متناسب می توانید اعمال مختلف را در قالب تابع با نام های مختلف کنید و سپس انها را در هر کجا که می خواهید استفاده نمایید . توابع باعث می شود اسکریپها به صورت کنترل شده در صفحه اجرا شوند .

معمولا توابع در هنگامی که رویدادی در صفحه رخ می دهد یا آنها را صدا بزنیم اجرا می گردند . شما می توانید یک تابع درون صفحه را از هر کجا که مایلید call کنید و آن را اجرا نمایید . همچنین اگر تابع در یک فایل js. بیرونی که به صفحه ضمیمه می شود تعریف شده باشد می تواند در هر صفحه ای مورد استفاده قرار گیرد . تعریف توابع باید در قسمت <head> درون صفحه تعریف گردند تا در هنگام lood صفحه تعاریف معتبر گردد . به کد زیر توجه کنید :

```
<html>
<head>
<script type = “text/ javascript >
Function display message (  )
{
Alert ( “hello  world !” )
}
</script>
</head>
```

در این کد تابع( ) display message در قسمت head تعریف شده و در رویداد onclick دکمه درون فرم صدا زده می‌شود .

```
<body>
<form>
<input type = “ button” value = “ click me!” onclick = “display message (  ) “ >
</form>
</body>
</html>
```

در نتیجه هنگامی که کاربر دکمه فرم را کلیک کند تابعdisplay message اجرا شده و عبارت! hello world روی صفحه ظاهر می شود .

نحوه تعریف یک تابع :

فرم کلی تعریف تابع در javascript به صورت زیر است :

```
Function  نام تابع ( ... و مقدار ۲ و مقدار ۱ )
    {
دستورات
    }
```

نام تابع از همان قوانین نامگذاری متغیر ها تبعیت می کند و باید دقت کنید نام متغیر و تابع نباید با دستورات استاندارد jscript مشابه باشد و نیز نام تابع به حروف بزرگ و کوچک حساس است . به عنوان مثال می توانیم ۲ تابع با نام STR و str داشته باشیم که برایjscript هر کدام تابع جداگانه ای خواهد بود . مقدار۱ ، مقدار۲ و ... پاراترهای ورودی تابع هستند که در واقع در غالب متغیر می باشد و برای ارسال مقادیری جهت بکارگیری در تابع کاربرد دارد . چنانچه بخواهیم تابعی را بدون پاراتر های ورودی بنویسیم به صورت زیر عمل می کنیم :

```
Function      نام تابع (  )
    {
دستورات
    }
```

نکته ای که باید در نظر بگیرید این است که توابع به شکلهای متفاوتی بکار می روند . بعضی توابع پاراتر ورودی دارند ولی چیزی را به برنامه صدا کننده بر نمی گردانند و فقط عملی را انجام می دهند ، برخی دیگر پاراترهای ورودی ندارند اما مقداری را براساس عملیات خود باید به برنامه اصلی برگردانند ، دسته سوم پاراتر ورودی ندارند اما مقداری را بر اساس عملیات خود باید به برنامه اصلی برگردانند ، دسته سوم پاراتر ورودی دارند و خروجی حاصل از عملیلت درون خود را به برنامه اصلی می فرستند و گروه چهارم نه پراتر ورودی دارند و نه خروجی و تنها برای انجام اعمال خاصی بکار می روند .
برای ارسال پاراتر ورودی به منابع دو مسأله باید رعایت شود . یکی اینکه نام متغیر متناظر با مقدار ارسالی در تعریف تابع تعیین شود و دوم اینکه در هنگام صدا زدن تابع مقدار ارسالی را تعیین کنیم . چنانچه برای متغیر های تعیین شده در تعریف تابع مقداری ارسال نکنیم یامقدار پاراترهای ورودی با متغیر های تعریف شده برابر نباشد با خطا مواجه می شویم . تابع زیر را در نظر بگیرید .

```
Function  prod (a,b)
  {
    X= a * b
  }
```

هنگامی که این تابع را صدا می زنیم باید دستور زیر استفاده گردد :

```
Prod ( 5, 6 )
```

و یا بجای مقادیر ثابت می توانیم از متغیر استفاده کنیم :

```
Prod ( i , j )
```

اما اگر بجای ۲ مقدار درونی ، یک متغیر مقدار تعریف کنیم و یا اصلا مقداری ندهیم با خطا مواجه می شویم .
جهت بازگردادندن مقداری از تابع به برنامه اصلی از دستور return در انتهای تابع استفاده می کنیم . مقداری که توسط return باز می گردد در نام تابع ذخیره می شود بنابراین باید نام تابع را مانند یک متغیر در انتساب بکار گیریم تا بتوانیم حاصل عملیات تابع رااستفاده کنیم . به مثال زیر توجه کنید :

```
Function  prod (a,b)
  {
    X= a * b
   return  x
  }
```

همانطور که می توانیم در این تابع حاصل ضرب دو مقدار ورودی در متغیر x ذخیره می شود و توسط return برمی گردد .
اگر بخواهیم تابع فوق را بکار ببریم باید از دستوری مشابه زیر استفاده کنیم :

```
y = prod  (5,4)
```

در نتیجه مقدار y=20 خواهد بود که حاصل عمل دستورات درون تابع است .

تمرین : نحوه صدا کردن یک تابع درjavascript :

```
<html>
<head>

<script type="text/javascript">
function myfunction()
{
alert("HELLO")
}
</script>

</head>
<body>

<form>
<input type="button"
onclick="myfunction()"
value="Call function">
</form>

<p>By pressing the button, a function will
be called. The function will alert a
message.</p>

</body>
</html>
```

در این مثال تابع ( ) my function برای ایجاد یک کادر اخطار استفاده شده است و درون صفحه ازطریق خاصیت onclick دکمه تعریف شده فراخوانی می شود .

تمرین : در این تمرین باروش ارسال متغیر به یک تابع و استفاده ار ان آشنا می شوید :

```
<html>
<head>

<script type="text/javascript">
function myfunction(txt)
{
alert(txt)
}
</script>

</head>
<body>

<form>
<input type="button"
onclick="myfunction('Hello')"
value="Call function">
</form>

<p>By pressing the button, a function with
an argument will be called. The function
will alert
this argument.</p>

</body>
</html>
```

در کد بالا پاراتر txt بعنوان متغیر ورودی تابع ( ) my function تعریف شده که مقدار آن در دستور alert مورد استفاده قرار می گیرد .

تمرین : این تمرین نمونه دیگری از روش ارسال پارامتر به تابع می باشد .

```
<html>
<head>
<script type="text/javascript">
function myfunction(txt)
{
alert(txt)
}
</script>
</head>

<body>
<form>
<input type="button"
onclick="myfunction('Good Morning!')"
value="In the Morning">

<input type="button"
onclick="myfunction('Good Evening!')"
value="In the Evening">
</form>

<p>
When you click on one of the buttons, a
function will be called. The function will
alert
the argument that is passed to it.
</p>

</body>
</html>
```

تمرین : کد زیر تابعی را تعریف می کند که مقداری را به برنامه اصلی باز خواهد گرداند .

در این کد تابع( ) my function بدون پاراتر ورودی تعریف شده است . و توسط دستور return رشته ای را بر می گرداند . درون script همانطور که مشاهده می شود نام تابع ( ) my function در دستورdocument.write قرار می گیرد تا مقدار حاصل از عمل آن روی صفحه نمایش داده شود . تمرین : در این مثال تابعی تعریف شده است که دو پارامتر ورودی دریافت می نماید و مقداری را به عنوان نتیجه بر می گرداند :

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3))
</script>
<p>The script in the body section calls a
function with two parameters (4 and 3).</p>
<p>The function will return the product of
these two parameters.</p>
</body>
</html>
```

همانطور که مشاهده می کنید تابع ( ) product با دو مقدار ورودی ۳ و ۴ درون دستور document.write بکار رفته تا در نتیجه عمل آن و اجرای دستور return ، مقدار حاصل ضرب دو عدد روی صفحه وب چاپ گردد .

حلقه ها در جاوااسکریپ (javascript loops) :

همانطور که می دانید مفهوم تکرار در برنامه سازی جهت تکرار دستورات به تعداد شخص یا تحت یک شرط خاص استفاده می شود که این مهم معمولا در زبانهای برنامه نویسی شامل ۲ یا ۳ ساختار می شود . در java script تکرار یا برای کنترل دفعات مشخص است یعنی تعدادتکرار دستورات را می دانیم که در این صورت از ساختار for .. loop استفاده می شود و یا تکرار براساس درستی یا نادرستی شرطی خاص انجام می شود که در این حالت باید از ساختار while … loop بهره ببریم بنا براین :

- ✓ **For** :در جاوا اسکریپ جهت تکرار یک بلوک از دستورات به مقدار مشخص بکار می رود
- ✓ **While**: در جاوا اسکریپ جهت تکرار یک بلوک از دستورات تا زمانی که شرط تعیین شده درست باشد ، استفاده می شود .

ساختار حلقه for (حلقه شمارشی)

همانطور که بیان شد حلقه for زمانی بکار می رود که بدانیم چند بار باید دستورات را تکرار کنیم . در بدنه حلقه for سه پارامتر را باید درست لحاظ کنیم :

- ✓ **مقدار اولیه** : این مقدار در ابتدای کار حلقه به متغیر شمارنده منتسب می شود و در واقع مشخص کننده نقطه شروع تکرار است .
- ✓ **شرط پایان حلقه** : در واقع انتهای شمارش تکرار حلقه مقدار ثانویه خواهد بود که حلقه باید تا رسیدن به ان ادامه یابد .
- ✓ **گام حلقه** : در واقع می تواندر jscript گام رسیدن مقدار اولیه به مقدار ثانویه تنظیم نمود که این عمل به صورت شمارش یکی یکی یا دوتا دوتا و غیره انجام شود .

نکته ای که باید در نظر بگیرید این است که اگر مقدار اولیه ار مقدار ثانویه کمتر باشد لزوما گام حلقه مثبت خواهد بود و در مواردی که مقدار اولیه بیشتر از مقدار ثانویه باشد ، باید گام حلقه را منفی در نظر بگیریم .

شکل کلی ساختار حلقه for در jscript بصورت زیر است :

```
for ( نام متغیر = مقدار اولیه ; متغیر ⟵ مقدار ثانویه ; متغیر = متغیر + گام حلقه )
    {
        دستورات
    }
```

به مثال زیر توجه کنید :

```
for ( i = 0  ;  i < = 10  ;  i + = 2 )
    {
    document.write("hello")
    }
```

در این مثال ۵ بار کلمه hello روی صفحه چاپ می شود و حلقه از ۰ شروع شده و با فاصله ۲ تایی به مقدار ۱۰ می رسد .

حال به مثال زیر دقت کنید :

```
for ( i = 10  ;  i > = 0  ;  i - = 2 )
     {
    document.write("hello")
     }
```

در این کد هم ۵ بار کلمه hello روی صفحه چاپ می شود منتهی حلقه از مقدار ۱۰ شروع شده و با کاهش دو واحدی به صفر می رسد . در مورد حلقه for باید به این نکته توجه داشته باشید که چنانچه یکی از پاراترهای سه گانه for (مقدار

اولیه ، شرط یا گام حلقه) ذکر نشود با خطا مواجه می شوید ولی اگر پاراترها را درست تعیین کنید اما در تعیین شرط یا گام حلقه اشتباه کنید ، خطایی رخ نمی دهد فقط حلقه اجرا نخواهد شد .

تمرین : در این تمرین نحوه ایجاد یک ساختار در for برای تکرار دستورات نمایش داده شده است :

```
<html>
<body>

<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write("The number is " + i)
document.write("<br />")
}
</script>

<p>Explanation:</p>

<p>This for loop starts with i=0.</p>

<p>As long as <b>i</b> is less than, or
equal to 5, the loop will continue to
run.</p>

<p><b>i</b> will increase by 1 each time
the loop runs.</p>

</body>
</html>
```

همانطور که می بینید حلقه for با شمارنده i از مقدار ۰ شروع شده و یکی یکی به سمت ۵ حرکت می کند در نتیجه دستور درون بلوک حلقه ۵ بار چاپ می شود و اعداد ۰ تا ۵ روی صفحه دیده خواهند شد . کارکتر
 که بعد از دستور چاپ اعداد آمده باعث می شود هر عدد در یک خط جداگانه چاپ گردد .

حال با رستکاری حلقه و تغییر مقدار اولیه از ۰ به ۵ و تغییر شرط حلقه و گام آن طوری script را تغییر دهید که اعداد ۵ تا ۰ را چاپ نماید .

تمرین : تمرین زیر نحوه تغییر اندازه تیتر header در html را با کمک حلقه for نمایش می دهد و در نتیجه انواع تیتر ها با اندازه مختلف روی صفحه دیده خواهد شد .

در کد بالا حلقه for از مقدار ۱ تا ۶ با گام یک واحدی تعریف شده و درون آن رشته ای شکل می گیرد که در واقع برچسب تیتر در ht,l را تشکیل می دهد ، منتهی قسمت عددی برچسب با تغییر حلقه تعیین می شود . مثلا اگر i =2 باشد انگاه عبارت "< " + i +"h"> معادل <h2> خواهد بود .

ساختار حلقه while (حلقه شرطی) :

بر اساس آنچه تاکنون بیان شد حلقه while تکرار بلوک دستورات درون خود را تا زمانی که شرط مطرح شده درون آن true باشد اجرا می کند . حلقه while به دو صورت در جاوا اسکریپت مورد استفاده قرار می گیرد . یکی در قالبwhile و یکی در قالب do ... while . تفاوت حلقه while با do ... while در این است که دستورات درون while تنها در صورتی اجرا می شود که شرط حلقه true باشد اما دستورات درون ساختار do ... while چه شرط حلقه درست باشد چه نادرست ، حداقل یکبار اجرا می گردند . در ادامه این مطلب را بیشتر مورد بررسی قرار می دهیم .

ساختار حلقهwhile بطور کلی به فرم زیر است :

```
while ( شرط )
    {
      دستورات
    }
```

همانطور که می بینید در این ساختار بدلیل اینکه ابتدا شرط بررسی می شود سپس بلوک دستورات قرار دارد ، در نتیجه دستورات در صورتی اجرا می شوند که حتما شرط درست باشد در غیر اینصورت دستورات در درون while اجرا نخواهد شد به مثال زیر توجه کنید :

```
Var i = 0
While   ( i < 10 )
   {
      document.write("hello")
       i + +
    }
```

در مثال بالا کلمه hello ده بار روی صفحه چاپ می شود . چنانچه شرط را به $i > 10$ تغییر دهیم دستورات حلقه ابدا اجرا شود : نکته ای که باید دقت کنید این است که در ساختار while برخلاف for در ابتدای حلقه گام حلقه مشخص نمی شود و لازم است دستور افزایش یا کاهش متغیر حلقه (i + +) را حتما درون ساختار بنویسیم . چنانچه این دستور فراموش شود و شرط حلقه نیز درست باشد یک حلقه بی نهایت بوجود می آید که هیچگاه ختم نمی گردد لذا سیستم مشغول شده و اغلب دچار مشکل خواهید شد .

ساختار حلقه do ... while به شکل زیر است :

```
do
   {
      دستورات
    }
while ( شرط )
```

همچنان که پیداست بلوک دستورات قبل از شرط قرار گرفته است ، لذا ابتدا یکبار دستورات اجرا می شود آنگاه شرط کنترل شده و چنانچه درست باشد دستورات درون بلوک do while تکرار خواهد شد :

```
Var i = 0
do
   {
      document.write(“hello”)
       i + +
    }
While ( i < = 10 )
```

تمرین : مثال برای استفاده از حلقه while . در این مثال حلقهwhile پنج بار تکرار می شود و ۵ جمله روی صفحه ظاهر می گردد :

```
<html>
<body>

<script type="text/javascript">
i = 0
while (i <= 5)
{
document.write("The number is " + i)
document.write("<br>")
i++
}
</script>

<p>Explanation:</p>

<p><b>i</b>  equal to 0.</p>

<p>While <b>i</b> is less than , or equal
to, 5, the loop will continue to run.</p>

<p><b>i</b> will increase by 1 each time
the loop runs.</p>

</body>
</html>
```

تمرین : نمونه ای از پیاده سازی حلقه با ساختارdo while . این برنامه همان خروجی مثال قبل را خواهد داشت :

کنترل حلقه ها در javascript :

در ساختار حلقه ها مشاهده نمودید که کنترل پایان حلقه در بدنه آن لحاظ می شود و حلقه تکرار آن به صورت عادی تا زمان خاصی ادامه می یابد . اما دو نوع کنترل دیگر نیز برای حلقه ها وجود دارد که اجتناب ناپذیر است . یکی اینکه بخواهیم تحت شرایطی خاص تکرار حلقه ها را قبل از پایان دفعات تکرار یا رسیدن به شرط خاتمه حلقه متوقف کنیم و دیگر اینکه بخواهیم حلقه مرحله فعلی را رها کند و به مرحله بعد برود .

✓ **دستور break :**

در حالت اول برای ختم تکرار ساختار حلقه از دستور break استفاده می شود . این دستور باعث می شود کنترل مغز javascript دستورات درون بلوک را تکرار نکند و به بعد از ساختار حلقه مراجعه نماید به عنوان مثال کد زیر حلقه تکراری برای ده بار تکرار تعریف می کند و این حلقه ۳ بار بیشتر اجرا نمی شود چون با رسیدن مقدار متغیر حلقه به عدد۳ دستور break اجرا شده و حلقه خاتمه می یابد :

```
Var i = 0
For ( i = 0 ; i < = 10; i + + )
{
If ( i = = 3 ) { break }
      document.write(“hello”)
}
```

دستور continue :

دستور continue درون حلقه باعث می شود عملیات مرحله فعلی در مکان continue خاتمه یابد و حلقه از مرحله جدید شروع به اجرا نماید . به عنوان مثال برنامه زیر حلقه ای را تعریف می کند که قرار است اعداد از ۱ تا ۱۰ را روی صفحه نمایش دهد . چنانچه در قبال شرط(i = = 3) دستورcontinue قرار گیرد حلقه در مرحله سوم اجرا نشده و از مرحله چهارم شروع به کار می کند در نتیجه عدد ۳ روی صفحه چاپ نخواهد شد :

```
Var i = 0
For ( i = 0 ; i < = 10; i + + )
{
If ( i = = 3 ) { continue }
      document.write( I )
}
```

تمرین : کد زیر نمونه ای ار کاربرد دستور break و نحوه عمل آنرا نمایش می دهد :

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3){break}
document.write("The number is " + i)
document.write("<br />")
}
</script>
<p>Explanation: The loop will break when
i=3.</p>
</body>
</html>
```

تمرین : برنامه زیر مثالی از نحوه کار دستورcontinue درون یک حلقه را نشان می دهد :

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3){continue}
document.write("The number is " + i)
document.write("<br />")
}
</script>

<p>Explanation: The loop will break the
current loop and continue with the next
value when i=3.</p>

</body>
</html>
```

ساختار for … in :

ساختار for … in نوعی ساختار عملیاتی است برای حرکت بین خانه های یک آرایه یک محتوی یک موضوع . این ساختار را نباید با حلقه for اشتباه بگیرید . در واقع این ساختار باعث می شود دستورات درونش به ازاء هر مقدار درون یک آرایه یا موضوع ، یکبار اجرا شوند .

فرم کلی ساختار for … in به صورت زیر است :

for(مقدار in موضوع یا آرایه)

{

دستورات
}

به مثال زیر توجه کنید :

```
<html>
<body>
<script type = “text/javascript”>
 Var x
Var my cars = new array (  )
My cars [ 0 ] = “ saab”
My cars [1] = “Volvo”
My cars [2] = “bmv”
For (x in my cars )
{
      document.write(my cars [x] )
}
</script>
</body>
</html>
```

درون برچسب <script> سطر اول متغیر ط را تعریف می نماید . در سطر بعد آرایه my cars تعریف می گردد (در مورد آرایه ها در فصلهای بعدی صحبت بعمل خواهد امد) در سطر سوم تا پنجم خانه های ۰ تا ۲ آرایه مقدار دهی می شود و در نهایت ساختار for … in به ازاء هر x عضو آرایه یکبار دستورdocument.write را اجرا می کند . در نتیجه محتوی آرایه روی صفحه چاپ می گردد .

رویدادها در(js events) :

رویدادها ، عملیاتی هستند که توسط javascript قابل تشخیص می باشد . توسط رویدادها شما قابلیت برنامه ریزی عملیات مختلف از سوی کاربر را خواهید داشت . هر المنتی روی صفحه دارای تعدادی رویداد مختلف میباشد که می تواند توابع جاوا اسکریپت را برای انجام عمل خاصی فراخوانی کند . به عنوان مثال می توانید رویداد onclick یک دکمه رابا تابع خاصی برنامه ریزی کنید تا زمانی که کاربر روی دکمه کلیک می کند عملیات درون تابع اجرا شود . نمونه ای از رویدادها به شرح زیر می باشد :

- ✓ کلیک ماوس (mouse click)
- ✓ لود شدن صفحه یا تصویر (page or image load)
- ✓ قرار گرفتن نشانگر ماوس در مکانی خاص از صفحه (mouse over)
- ✓ انتخاب یک جعبه متن یا المنت از درون فرم (focus , blur)
- ✓ ارسال اطلاعات یک form (submit)
- ✓ فشردن یک کلید (keystroke)

کار رویدادها در ایجاد برنامه های هوشمند بسیار مفید است و در واقع می توانیم با بکارگیری مناسب رویدادها ، ضخامت متعادل و کاربر پسند به وجود آوریم . در ادامه دسته بندی مشخصی از انواع رویداد هایی که توابع javascript را فراخوانی می کنند ، ارائه می شود (برای دسترسی به مرجع کامل رویدادها به ضمائم کتاب مراجعه کنید)

*** رویداد onload و on unload :**

رویداد onload زمانی رخ می دهد که کاربر وارد صفحه ای می شود و قبل از هر اتفاقی صفحه load می شود لذا با بکار گیری onload می توان اعمالی را توسط javascript قبل از اجرای هر چیزی در صفحه انجام داد .

رویداد unonload :عکس رویداد onload است و زمانی اتفاق می افتد که کاربر صفحه را ترک می کند بنابراین می توان در هنگام بستن صفحه و قبل ار آنکه صفحه را ترک کنیم اعمالی را انجام دهیم .

در مثال زیر هنگامی که صفحه باز می شود و نیز در هنگام بسته شدن صفحه پیام ظاهر می گردد :

<body onload = " show alert ('welcom')" onunload = " showalert (' good bye')>

همانطور که می بینید دو رویداد onload و un onload درون برچسب body تعریف می شوند .

*** رویداد های onfocus و onblur و onchange :**

این رویدادها برای بررسی صحت اطلاعات درون المنتهای یک فرم کاربرد دارد . رویداد onfocus زمانی اتفاق می افتد که کنترل صفحه به یک المنت در فرم انتقال یابد . به عبارت دیگر می توانیم هنگامی که کاربر روی یک کنترل زوم می کند تا درون آن چیزی بنویسد یا مقدار آن را تغییر دهد قبل از عمل کاربر قابل برنامه ریزی است .

رویداد onblur : عکس رویداد onfocus است و زمانی اتفاق می افتد که کاربر کنترل را ترک می کند . به عنوان مثال فرض کنید که کاربر می خواهد درون یک جعبه متن داخل فرم اطلاعاتی را بنویسد آنگاه هنگامی که کاربر روی جعبه متن می رود (کلیک می کند یا با کلید tab جعبه متن را انتخاب می کند) رویداد onfocus رخ می دهد و هنگامی که کاربر از این جعبه متن به کنترل دیگری می رود رویداد on blur اتفاق خواهد افتاد .

رویداد onchange زمانی اتفاق می افتد که کاربر اطلاعات یا حالت یک کنترل راتغییر دهد . بنابرآنچه بیان شد ترتیب رویدادهای فوق را می توان بصورت زیر تصور نمود :

(رفتن به کنترل بعدی on blur) ⟵ (onchange) تغییر محتوی ⟵ onfocus) ⟵ انتخاب کنترل توسط کاربر)

رویداد on submit :

این رویدادها همانطور که از نامش پیداست زمانی اتفاق خواهد افتاد که شما دکمه submit فرم را زده اید و می خواهید اطلاعات درون فرم را به سرور انتقال دهید . در واقع با استفاده از onsubmit شما قادرید بعد ار پایان ورود اطلاعات به فرم و قبل از ارسال آن به server اعمالی را انجام دهید . این روش برای کنترل صحت محتوی فرم قبل از ارسال بسیار مفید است و می تواند باعث افزایش سرعت عمل سایت شما شود :

<form method = “post” action = “ xxx.htm”
Onsubmit = “return check form ( ) “ >

*** رویداد onmouseout و onmouseover :**

این رویدادها مربوط به تشخیص حرکت mouse روی المنتهای صفحه می باشد که می تواند در طراحی منوهای گرافیکی و یا سایر مقاصد جذاب کارآمد باشد . رویدادonmouseover زمانی اتفاق می افتد که نشانگر ماوس روی المنت قرار بگیرد و رویداد onmouseover در هنگامی فعال می شود که نشانگر ماوس از محدوده المنت خارج گردد .

به کد زیر توجه کنید . در این برچسب یک لینک تعریف شده که هنگامی ماوس روی آن می رود کادر اخطاری نمایش داده می شود :

…

کنترل خطا در جاوا اسکریپت (js cotching Errors) :

یکی از مشکلات عمده برنامه نویسان این است که درون برنامه ممکن است خطاهایی رخ دهد و چنانچه این خطا ها کنترل نگردد برنامه از مسیر عادی خارج شده و کاربر مجبور است همه چیز را از نوع شروع نماید . فرض کنید در حال ورود اطلاعات به فرمی هستید که در آن فیلدهای زیادی وجود دارد ، چنانچه در هنگام submit کردن با یک خطا روبرو شوید همه ورودیهای شما از بین می رود و شما مجبورید همه اطلاعات را دوباره وارد کنید . بسیاری از کاربران در برخورد با چنین حالتی صفحه و سایت را رها می کنند این مسئله جذابیت صفحه را کاهش می دهد . کادر خطای اینچنینی معمولا با پیام Runtime error همراه است و به کاربر دو گزینه debug و cancel را می دهد .

ساختار try … catch

به فرم زیر است :

```
Try {
      دستورات
    }
Catch (err)
    {
      دستورات کنترل خطا
    }
```

به طور کلی برای کنترل خطا سه نوع ساختار وجود دارد :

- ✓ **ساختار try … catch** : برای کنترل خطا های برنامه و دستورات
- ✓ **دستور throw** : برای ایجاد خطای مصنوعی و کنترل ورودیهای کاربران (که باید با try استفاده شود)
- ✓ **رویداد onerror** : مدل قدیمی تر کنترل خطا در jscript

دستوراتی که در قسمت try مشخص شده اند اجرا می گردند و در صورتـــی که این دستورات منجر به تولید خطا توسط javascript می گردد آنگاه دستورات درون بلوک catch اجرا شده و خطـــا را مدیریت می کنند و چنانچه خطایی تولید نشود ، دستورات درون catch اجرا نخواهد شد . در مثال زیر اسکریپی نوشته شده است که درون آن تابع ( ) message توسط رویداد on click فراخوانی می شود . درون تابع بجای دستور ( ) alert از دستور ( ) addalert استفاده شده که چون برای j script ناشناخته است منجر به بروز خطا می گردد :

```
<html>
<head>
<script type = “text/javascript”>
Function message ( )
{
Addalert ( “welcome guest”)
}
</script>
</head>
<body>
<input type = “buttom “ value = “click me “ onclick = “message (  ) “ />
</body>
</html>
```

با لود شدن صفحه و فشردن صفحه توسط کاربر خطای runtime rrror بروز می کند . حالا در کد زیر با استفاده از ساختار try…catch بروز خطا را کنترل می کنیم و مشاهده خواهید کر د که بجای پیام runtime rrror پیغام تعیین شده ما نمایش داده می شود :

```
<html>
<head>
<script type = “text/javascript”>
Var-txt = “  “
Function message ( )
{
Addalert ( “welcome guest”)
}
Catch (err)
{
Txt=”there was on error on this page.”1n”
Txt+ - “error description :” + err.deseription + “1n’
Txt + = “ click ok to continue”
Alert (txt)
}
}
</script> </head>
```

<body>
< input type = “buttom” value = “view message” onclick = “message ( ) “ / >
</body> </html>

در کد بالا err.description شرح خطایی که سیستم تولید نموده را نشان می دهد و برای نمایش شماره خطا می توانیم از errinamber استفاده کنیم . در هنگامی که دکمه را کلیک می کنیم تابع ( ) message اجرا شده و در نتیجه پیغام خطایی که در txtذخیره کرده ایم نمایش داده می شود .

تمرین: نوشتن کدی که مشابه مثال قبل خطای تولید شده را کنترل کند اما از طریق کادر تأیید با کاربر ارتباط برقرار کند . در این تمرین به جای دستور alert(txt) از دستور confirm (txt) درون یک شرط استفاده شده است تا در صورتی که کاربر دکمه cancle را بزند صفحه خاصی نمایش داده شود .

```
<html>
<head>
<script type="text/javascript">
var txt=""
function message()
{
try
   {
   adddlert("Welcome guest!")
   }
catch(err)
   {
     txt="There was an error on this page.\n\n"
     txt+="Click OK to continue viewing this page,\n"
     txt+="or Cancel to return to the home page.\n\n"
     if(!confirm(txt))
         {

document.location.href="http://www.w3schools.com/"
         }
   }
}
</script>
</head>


<body>
<input type="button" value="View message"
onclick="message()" />
</body>

</html>
```

در بسیاری از مواقع برنامه نویسان مایلند مقدار وارده توسط کاربر را کنترل کنند یا اینکه خطایی احتمالی که در اثر ورود اطلاعات غیر همنوع توسط کاربر به وجود می اید باعث اشکال در برنامه نشود . به عنوان مثال اگر کاربر به جای اینکه عددی

را وارد کند ،حرف وارد نماید آنگاه برنامه در محاسبات ریاضی دچار اشکال خواهد شد و یا گاهی ممکن است بخواهیم محدوده خاصی را برای کاربر در نظر بگیریم که چنانچه از آن خارج شد برنامه خطای مناسبی صادر کند. برای این مواقع از دستور throw استفاده می کنیم تا خطایی ایجاد نماییم . برای روشن شدن موضوع به ذکر این مثال می پردازیم . در نظر بگیرید مایلید برنامه ای بنویسید که در آن کاربر عددی بین ۰ تا ۱۰ را وارد نماید . حال می خواهیم طوری برنامه را طرح کنیم که اگر عدد وارده کوچکتر از ۰ باشد پیغام the number is too low و چنانچه عدد از ۱۰ بزرگتر باشد برنامه پیام the number is too high را صادر نماید .

فرم دستور throw به شکل زیر ست که باید با حروف کوچک نوشته شود :

throw (توضیح)

با استفاده از throw جاوا اسکریپ یک توضیح را در نظر می گیرد تا اگر خطایی اتفاق افتاد ، با استفاده از این توضیح درون ساختار catch پیام مناسبی صادر نماید و یا اعمالی را انجام دهد . به مثال زیر توجه کنید :

```
<html>
<body>
<script type = “text/javascript”>
var x = prompt ( “enter a number between 0 and 10 : “ , “   “ )
try
    {
      if ( x>10 )
         throw “Err 1 “
      else if (x<0)
        Throw “ Err 2 “
      document . write ( “you enter correct number ! “ )
}
catch (err)
{
if (err = = “Err1”
     Alert ( “Error ! the value is too high “ )
if ( err = = “ Err 2 “ )
    alert ( “Error ! the value is too low “ )
}
</script> </body> </html>
```

همانطور که می بینید دستور throw خطای ساختگی ایجاد می کند و کنترل برنامه را به درون catch می فرستد تا متناسب با توضیحی که مشخص نموده عملیات انجام شود . در این برنامه متغیر x توسط دستور prompt مقداری را دریافت می کند . سپس درون ساختار try اگر مقدار x از ۱۰ بیشتر باشد دستور throw توضیح ”Err 1“ خواهد بود که در

این صورت پیام مناسب صادر می شود حال اگر بخواهیم برنامه بالا را طوری تغییر دهیم که علاوه بر محدوده ورودی نوع آن را نیز تشخیص دهد و در صورتی که کاربر مقدار غیر عددی وارد کرد نیز پیام خطا صادر کند به شکل زیر عمل می کند :

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","")
try
{
if(x>10)
  throw "Err1"
else if(x<0)
  throw "Err2"
else if(isNaN(x))
  throw "Err3"
}
catch(er)
{
if(er=="Err1")
  alert("Error! The value is too high")
if(er == "Err2")
  alert("Error! The value is too low")
if(er == "Err3")
  alert("Error! The value is not a number")
}
</script>
</body>
</html>
```

یکی دیگر از روشهای کنترل خطا استفاده از رویداد onerror می باشد . این رویداد درون <script> تعریف می گردد و فرم کلی تعریف آن به صورت زیر است :

onerror = نام تابع کنترل خطا

=

function نام تابع کنترل خطا (شماره خط و آدرس صفحه و پیام خط)

{

دستورات کنترل خطا

}

دستور Onerror معمولا بعد از برچسب <script> قرار می گیرد و مشخص کننده تابع کنترل خطا است . تابع کنترل خطا یک تابع معمولی است که سه پاراتر ورودی برای آن وجود دارد . پاراتر اول پیام خطا است که همان err.description می باشد و شرح خطای ایجاد شده در سیستم را درون خود نگهداری می نماید . پاراتر دوم آدرس url صفحه ای است که خطا در آن اتفاق افتاده و پاراتر سوم شماره خطی از صفحه است که خطا در آن رخ داده است . در نتیجه در هنگام بروز خطا برنامه نویس به محتوی خطا آدرس صفحه و شماره خط برنامه دسترسی دارد . که راهنمای خوبی برای اشکالزدایی برنامه محسوب می گردد .

در مثال زیر روش استفاده از رویداد onerror و تعریف تابع کنترل خطا به خوبی نشان داده شده است :

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr
var txt=""

function handleErr(msg,url,l)
{
txt="There was an error on this page.\n\n"
txt+="Error: " + msg + "\n"
txt+="URL: " + url + "\n"
txt+="Line: " + l + "\n\n"
txt+="Click OK to continue.\n\n"
alert(txt)
return true
}

function message()
{
adddlert("Welcome guest!")
}
</script>
</head>

<bodv>
<input type="button" value="View message"
onclick="message()" />
</body>

</html>
```

در کد بالا تابع ( ) handle Err به عنوان تابع کنترل خطا تعریف شده که سه پاراتر ورودی msg (پیام خطا) ، url (آدرس صفحه) و L شماره خط را دارا است . چون در بدنه اصلی اسکریپ به جای دستور alert از دستور addalert استفاده شده لذا خطا رخ می دهد در نتیجه تابع ( ) handle Err اجرا شده و پیام خطا ، آدرس صفحه و شماره خطی که در آن خطا رخ داده را نمایش خواهد داد .

کاراکتر های ویژه در جاوا اسکریپت (js special characters)

در جاوا اسکریپت امکان افزودن کاراکترهای ویژه به متن وجود دارد منتهی این عمل با استفاده از کاراکتر / (back slash) امکان پذیر خواهد بود . به متن زیر توجه کنید :

My name is “behzad”.

اگر بخواهیم این متن را در صفحه چاپ کنیم نمی توانیم عبارت زیر را بکار بریم :

document . write (“My name is “behzad”.”)

چون کاراکتر " برای سیستم مفهوم دارد لذا باید آن را به صورت "/ بکار بریم . بنابراین برای چاپ متن اول باید دستور زیررا بکار برد .

document . write (“My name is /“behzad /”.”)

منظور از کاراکتر های ویژه در واقع کاراکترهایی مانند خط جدید (new line) ، گیومه (apostrophes) ، کوتیشن (quotes) و سایر کاراکترهایی است که برای jscript معنا و مفهوم خاصی دارند .

مثلا اگر بخواهیم در جمله ای از یک علامت & استفاده کنیم باید آنرا به صورت &/ در رشته متنی قرار دهیم والا چاپ نخواهد شد .

document . write (“you /& me are walking.”) → you & me are walking .

جدول زیر برخی کاراکتر ویژه را نمایش می دهد :

خروجی	علامت
نمایش یک کوتیشن جداگانه درون رشته متنی	/'
نمایش یک زوج کوتیشن درون رشته متنی	/"
نمایش علامت & ampersand	/&
نمایش علامت / back slash	//
ایجاد خط جدید new line	/n
درج یک علامت carriage enter (مشابه زدن کلید enter)	/r
درج یک کاراکتر tab در رشته متنی	/t
درج یک کاراکتر back space (مشابه زدن کلید bks pace)	/b

چند نکته ای در رابطه با برنامه نویسی توسط javascript :

- ✓ کدها و دستورات javascript به حروف بزرگ و کوچک حساس هستند پس در هنگام نوشتن کد دقت کافی را بنمائیم .
- ✓ در jscript برای خواناتر شدن دستورات ، فضای اضافی بین دستورات می توان قرار داد که jscript آن را در نظر نمی گیرد به عنوان مثال دو دستور زیر یکسان هستند :

name=”hege”
name = “hege”

به این فضای خالی (white space) گفته می شود .

- ✓ در jscript یک رشته متنی را می توان در چند سطر نوشت منتهی باید در انتهای هر سطر علامت / قرار گیرد :

Document . write (“hello /
World ! “)

- ✓ توضیحات در jscript به دو صورت قابل افزودن به متن کد می باشد : یکی اینکه در ابتدای خط توضیح دو علامت / قرار دهیم :

// this is comment

دوم اینکه متن توضیح را درون */ و /* بگذاریم :

/* this is comment */