بسمه تعالی

جزوه جاوا اسکریپت

تهیه کننده : جواد بدلی

آموزش JS—شیء Math

شیء Math، برای محاسبات ریاضی استفاده می شود.

round()
چگونه از تابع ()round برای گرد کردن اعداد استفاده کنیم.

random()
چگونه از تابع ()random برای برگرداندن یک عدد بین ۰ و ۱ استفاده کینم.

max()
برگداندن بزرگترین عدد

min()
برگرداندن کوچکترین عدد

شیء Math

شیء Math، برای محاصبات ریاضی استفاده می شود.

شیء Math شامل چندین خصوصیت و متد برای محاصبات ریاضی است.

نحوه استفاده از خصوصیت ها و متدهای شیء Math

```
var x=Math.PI;
var y=Math.sqrt(١٦);
```

توجه: Math یک سازنده (constructor) نیست. تمام خصوصیت ها و متدهای شیء Math را می توانید بدون ایجاد آن، صدا بزنید.

خصوصیت های شیء Math

JavaScript هشت خصوصیت برای شیء Math فراهم نموده است:

۱. E

۲. PI

۳. square root of ۲

٤. square root of ۲/۱

٥. natural log of ۲

٦. natural log of ۱۰

۷. base- ۲log of E

۸. base- ۱۰log of E

نحوه استفاده:

```
Math.E
Math.PI
Math.SQRT۲
Math.SQRT۱_۲
Math.LN۲
Math.LN۱۰
Math.LOG۲E
Math.LOG۱۰E
```

متد های شیء Math

علاوه بر خصوصیت هایی که برای شیء Math در دسترس است، چندین متد نیز برای آن وجود دارد.

در مثال زیر، از متد ()round برای گرد کردن عدد "٤.۷" به نزدیک ترین عدد صحیح استفاده شده است:

```
document.write(Math.round(٤.۷));
```

خروجی کد بالا:

٥

در مثال زیر، از متد ()random برای برگرداندن یک عدد، بین ۰ و ۱ استفاده شده است:

document.write(Math.random());

خروجی کد بالا:

۰.۶۹۸۲۱۲۲۰۸۷۸۱۵۳۰۶

در مثال زیر، از متد()floor و ()random برای برگرداندن یک عدد صحیح بین ۰ و ۱۰ استفاده شده است:

document.write(Math.floor(Math.random()*۱۱));

خروجی کد بالا:

۶

آموزش JS–تاریخ

از شیء Date برای کار کردن با تاریخ و زمان استفاده می شود.

برگرداندن تاریخ و زمان جاری
چگونه از متد ()Date برای چاپ کردن تاریخ جاری استفاده کنیم

getFullYear()
برگرداندن سال جاری با استفاده از متد ()getFullYear

getTime()
برگرداندن تعداد میلی ثانیه های گذشته از تاریخ ۱۹۷۰/۰۱/۰۱ تا امروز

setFullYear()
چگونه از متد ()setFullYear برای تنظیم یک تاریخ دلخواه استفاده کنیم

toUTCString()
چگونه با استفاده از متد ()toUTCString تاریخ جاری را به یک رشته تبدیل کنیم (بر طبق UTC)

getDay()
چگونه با استفاده از متد ()getDay و یک آرایه، روز جاری را چاپ کنیم

نمایش یک ساعت
چگونه یک ساعت (با ثانیه شمار) را روی صفحه وب نمایش دهیم

مرجع کامل شیء Date

برای مشاهده یک مرجع کامل از خصوصیت ها (Property) و متدهای (Method) مربوط به شیء Date، به لینک زیر مراجعه نمایید:

کلیه متدهای شیء Date

ایجاد شیء Date

شیء Date، برای کار کردن با تاریخ و زمان استفاده می شود.

شیء Date را می توان با استفاده از constructor یا سازنده ()Date ایجاد نمود.

چهار روش برای اعلان تاریخ وجود دارد:

```
new Date() // current date and time
new Date(milliseconds) //milliseconds since ۱۹۷۰/۰۱/۰۱
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

بیشتر پارامترهای بالا اختیاری هستند. در صورت مشخص نشدن، مقدار صفر ارسال خواهد شد.

زمانی که یک شیء Date ایجاد می شود، تعدادی متد (method) برای کار روی آن در دسترس قرار می گیرد. بیشتر این متدها برای تنظیم کردن (set) و یا گرفتن (get) سال، ماه، روز، ساعت، دقیقه، ثانیه و میلی ثانیه است (البته با توجه به تاریخ سیستم و یا زمان جهانی UTC).

تمام محاسبات براساس میلی ثانیه با زمان شروع ۰۰:۰۰:۰۰ ۱۹۷۰/۰۱/۰۱ و طبق زمان جهانی (UTC) می باشد. یک روز شامل ۸۶,۴۰۰,۰۰۰ میلی ثانیه است.

چند مثال برای اعلان تاریخ:

```
var today = new Date()
var d۱ = new Date("October ۱۳, ۱۹۷۵ ۱۱:۱۳:۰۰")
```

```
var d۲ = new Date(۷۹,۵,۲٤)
var d۳ = new Date(۷۹,۵,۲٤,۱۱,۳۳,۰)
```

تنظیم (Set) تاریخ

با استفاده از متدهای شیء Date به آسانی می توانید تاریخ را دستکاری کنید.

در مثال زیر، یک تاریخ مشخص (۲۰۱۰/۰۱/۱٤) برای شیء Date تنظیم شده است:

```
var myDate=new Date();
myDate.setFullYear(۲۰۱۰,۰,۱٤);
```

و در مثال زیر، تاریخ ۵ روز آینده، برای شیء Date تنظیم شده است:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+۵);
```

توجه: در مثال بالا، به ازای اضافه شدن ۵ روز به تاریخ، ممکن است علاوه بر شیفت داده شدن روز، ماه و یا سال نیز شیفت داده شوند.

مقایسه دو تاریخ مختلف

می توان از شیء Date برای مقایسه دو تاریخ نیز استفاده نمود.

در مثال زیر، تاریخ جاری با تاریخ ۲۱۰۰/۰۱/۱٤ مقایسه شده است:

```
var x=new Date();
x.setFullYear(۲۱۰۰,۰,۱٤);
var today = new Date();

if (x>today)
 {
 alert("Today is before ۱٤th January ۲۱۰۰");
 }
else
```

```
{
alert("Today is after ١٤th January ٢١٠٠");
}
```

آموزش JS–آرایه ها

شیء آرایه یک نوع خاص از متغیر هاست که می تواند چندین داده را در قالب یک نام در خود ذخیره کند.تا

مثال -

ایجاد و مقداردهی یک آرایه:

مثال

```
var mycars = new Array();
mycars[٠] = "Saab";
mycars[١] = "Volvo";
mycars[٢] = "BMW";
```

مثال های بیشتری را در انتهای این مطلب خواهید دید.

آرایه چیست؟

آرایه یک متغیر خاص است که می توانید بیشتر از یک مقدار را در یک زمان در آن ذخیره نمایید.

اگر لیستی از آیتم ها داشته باشید (برای مثال، یک لیست از نام ماشین ها)، و بخواهید هر آیتم را در یک متغیر تنها ذخیره نمایید، می توان مانند زیر عمل نمود:

```
var car١="Saab";
var car٢="Volvo";
var car٣="BMW";
```

حالا:

- اگر لیست شما بیشتر از ۳ آیتم باشد مثلاً ۳۰۰تا چه کار می کنید.
- اگر در این لیست به دنبال یک ماشین خاص باشید چه کار می کنید.

در اینجا بهترین راه حل استفاده از آرایه ها است.

یک آرایه می تواند مقادیر متغیرها را تحت یک نام برای شما نگه دارد. و شما از طریق ایندکس آرایه می توانید به مقادیر دسترسی داشته باشید.

هر آیتم در آرایه ایندکس منحصر به فردی برای خود دارد که به راحتی از طریق ایندکس می توانید به مقادیر دسترسی پیدا کنید.

ایجاد آرایه

آرایه ها به سه روش ایجاد می شوند:

۱: با قاعده:

```
var myCars=new Array();
myCars[۰]="Saab";
myCars[۱]="Volvo";
myCars[۲]="BMW";
```

۲: خلاصه شده:

```
var myCars=new Array("Saab","Volvo","BMW");
```

۳: تحت الفظی:

```
var myCars=["Saab","Volvo","BMW"];
```

در کدهای بالا، یک شیء آرایه با نام myCars ایجاد می شود.

دسترسی به عناصر آرایه

هر آیتم در آرایه ایندکس منحصر به فردی برای خود دارد که به راحتی از طریق ایندکس می توانید به مقادیر دسترسی پیدا کنید.

کد زیر، مقدار اولین عنصر آرایه myCars را در متغیر name قرار می دهد:

```
var name=myCars[۰];
```

کد زیر، اولین عنصر آرایه myCars را تعریف می کند:

```
myCars[۰]="Opel";
```

توجه: ایندکس آرایه از صفر شروع می شود، یعنی اولین آیتم [۰] است، دومین آیتم [۱] و ...

ذخیره اشیاء مختلف در یک آرایه

تمام متغیرها، عناصر آرایه و توابع در JavaScript، شیء محسوب می شوند.

شما می توانید، انواع مختلف متغیرها، خروجی یک تابع و یا یک آرایه را در آرایه ای دیگر ذخیره نمایید:

```
myArray[۰]=Date.now;
myArray[۱]=myFunction;
myArray[۲]=myCars;
```

خصوصیت ها (Property) و متدهای (Method) مربوط به آرایه ها

در مثال زیر، خصوصیت length، تعداد عناصر آرایه myCars را در متغیر x قرار می دهد و تابع ()indexof، ایندکس عنصر مشخص شده را بر می گرداند:

```
var x=myCars.length          // the number of elements in myCars
var y=myCars.indexOf("Volvo")  // the index position of "Volvo"
```

ایجاد متدهای جدید

prototype، یک سازنده (constructor) عمومی در JavaScript است. از این طریق می توان، برای هر شیءای در JavaScript یک خصوصیت یا متد جدید ساخت.

مثال: ساخت یک متد جدید برای آرایه ها

```
Array.prototype.ucase=function()
{
 for (i=۰;i<this.length;i++)
 {this[i]=this[i].toUpperCase();}
}
```

در مثال بالا، یک متد جدید با نام ucase ساخته شده است که مقادیر عناصر آرایه را به حروف بزرگ تبدیل می کند

آموزش JS– شیء history

شیء window.history، شامل تاریخچه مرورگر است.

شیء window.history

شیء window.history را می توان بدون پیشوند window نیز نوشت.

بخاطر حفظ حریم کاربران، محدودیت هایی برای چگونگی دسترسی JavaScript به این شیء وجود دارد.

بعضی از متدهای شیء history:

- **history.back()**: همانند کلید back در مرورگر عمل می کند.
- **history.forward()**: همانند کلید forward در مرورگر عمل می کند.

متد history.back()

متد history.back()، با توجه به تاریخچه صفحات بازدید شده، مرورگر را به صفحه قبلی هدایت می کند.

این متد، مانند کلید back در مرورگر عمل می کند.

مثال

ایجاد کلید back در یک صفحه:

```
<html>
<head>
<script>
function goBack()
  {
  window.history.back()
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

خروجی کد بالا:

متد history.forward()

متد history.forward()، با توجه به تاریخچه صفحات بازدید شده، مرورگر را به صفحه بعدی هدایت می کند.

این متد، مانند کلید Forward در مرورگر عمل می کند.

مثال

ایجاد کلید forward در یک صفحه:

```
<html>
<head>
<script>
function goForward()
 {
 window.history.forward()
 }
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

آموزش JS—شیء screen

شیء window.screen، شامل اطلاعاتی درباره ی صفحه ی مونیتور بازدیدکننده است.

ابعاد صفحه ی مونیتور

شیء window.screen را می توان بدون پیشوند window نوشت.

بعضی از خصوصیت های شیء window.screen:

- **screen.availWidth**: عرض قابل دسترس صفحه مونیتور
- **screen.availHeight**: ارتفاع قابل دسترس صفحه مونیتور

عرض قابل دسترس صفحه مونیتور

خصوصیت screen.availWidth، عرض قابل دسترس صفحه ی مونیتور را به پیکسل برمی گرداند. (این اندازه شامل taskbar نمی شود)

مثال

مثال زیر، عرض صفحه مونیتورتان را برمی گرداند:

```
<script>

document.write("Available Width: " + screen.availWidth);

</script>
```

ارتفاع قابل دسترس صفحه مونیتور

خصوصیت screen.availHeight، ارتفاع قابل دسترس صفحه ی مونیتور را به پیکسل برمی گرداند. (این اندازه شامل taskbar نمی شود)

مثال

مثال زیر، ارتفاع صفحه مونیتورتان را برمی گرداند:

```
<script>

document.write("Available Height: " + screen.availHeight);

</script>
```

آموزش JS—شیء location

شیء window.location، می تواند برای برگرداندن آدرس صفحه جاری (URL) و یا هدایت کاربر به یک صفحه جدید استفاده شود.

شیء window.location

شیء window.location را می توان بدون پیشوند window نیز نوشت.

بعضی از خصوصیت های شیء location:

- **location.hostname**: نام دامین جاری را برمی گرداند.
- **location.pathname**: مسیر و نام صفحه جاری را برمی گرداند.
- **location.port**: شماره Port وب هاست را برمی گرداند. (۸۰ یا ٤٤٣)
- **location.protocol**: پروتکل مورد استفاده را برمی گرداند. (http یا https)

خصوصیت location.href

خصوصیت location.href، آدرس صفحه (URL) جاری را برمی گرداند.

مثال

مثال زیر، کل URL صفحه جاری را برمی گرداند:

```
<script>

document.write(location.href);

</script>
```

//برای مشاهده خروجی کد بالا، روی دکمه کلیک نمایید

خصوصیت location.pathname

خصوصیت location.pathname، مسیر و نام صفحه جاری، برروی سرور را برمی گرداند.

مثال

مثال زیر، مسیر و نام صفحه جاری را برمی گرداند:

```
<script>

document.write(location.pathname);

</script>
```

//برای مشاهده خروجی کد بالا، روی دکمه کلیک نمایید

متد location.assign()

متد location.assign()، یک صفحه جدید را بارگزاری می کند.

مثال

بارگزاری یک صفحه جدید:

```
<html>
<head>
<script>
function newDoc()
 {
 window.location.assign("http://www.beyamooz.com")
 }
</script>
</head>
<body>

<input type="button" value="Load new document" onclick="newDoc()">

</body>
</html>
```

آموزش JS—شیء Boolean

شیء Boolean برای تبدیل یک مقدار غیر Boolean به یک مقدار Boolean استفاده می شود (true یا false)

به ازای چه مقادیری شیء Boolean با true یا false تنظیم می شود.

ایجاد یک شیء Boolean

شیء Boolean تنها نماینده دو مقدار می تواند باشند: True یا False

قطعه کد زیر، یک شیء Boolean بنام myBoolean ایجاد می کند:

```
var myBoolean=new Boolean();
```

اگر زمان اعلان، شیء Boolean مقدار دهی نشود و یا با یکی از مقادیر زیر تنظیم شود:

- ۰
- ۰-
- null
- ""
- false
- undefined
- NaN

شیء Boolean با مقدار false تنظیم می شود و برای بقیه مقادیر (حتی اگر با رشته "false" مقدار دهی شود) با مقدار true تنظیم خواهد شد.

آموزش JS– شیء navigator

شیء window.navigator شامل اطلاعاتی درباره مرورگر بازدیدکننده است.

شیء window.navigator

شیء window.navigator را می توان بدون پیشوند window نیز نوشت.

مثال

```
<div id="example"></div>

<script>

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";

document.getElementById("example").innerHTML=txt;

</script>
```

اخطار !!!

اطلاعاتی که از شیء navigator دریافت می شود، اغلب گمراه کننده است و نباید برای یافتن نسخه مرورگر استفاده شود.

دلیل:

- داده های navigator می تواند توسط مالک مرورگر تغییر داده شود.
- بعضی مرورگرها برای دور زدن تست سایت، در تشخیص هویت خودشان خلل ایجاد می کنند.

- مرورگرهایی که قبل از یک سیستم عامل جدید آمده باشند، نمی توانند اطلاعات آن سیستم عامل را گزارش دهند.

تشخیص نوع مرورگر

به خاطر اینکه اطلاعات شیء navigator در مورد نوع مرورگر گمراه کننده است، می توانید از یک سری اشیاء مختلف برای پیدا کردن نوع مرورگر استفاده نمایید.

چونکه مرورگرهای مختلف، اشیاء مختلفی را پشتیبانی می کنند، می توانید از این نوع اشیاء برای پیدا کردن نوع مرورگر استفاده نمایید. برای مثال، چونکه فقط مرورگر Opera خصوصیت "window.opera" را پشتیبانی می کند، می توانید از آن برای مشخص کردن مرورگر Opera استفاده نمایید:

Example: if (window.opera) {...some action...}

آموزش JS—شیء screen

شیء window.screen، شامل اطلاعاتی درباره ی صفحه ی مونیتور بازدیدکننده است.

ابعاد صفحه ی مونیتور

شیء window.screen را می توان بدون پیشوند window نوشت.

بعضی از خصوصیت های شیء window.screen:

- **screen.availWidth**: عرض قابل دسترس صفحه مونیتور
- **screen.availHeight**: ارتفاع قابل دسترس صفحه مونیتور

عرض قابل دسترس صفحه مونیتور

خصوصیت screen.availWidth، عرض قابل دسترس صفحه ی مونیتور را به پیکسل برمی گرداند. (این اندازه شامل taskbar نمی شود)

مثال

مثال زیر، عرض صفحه مونیتورتان را برمی گرداند:

```
<script>

document.write("Available Width: " + screen.availWidth);

</script>
```

ارتفاع قابل دسترس صفحه مونیتور

خصوصیت screen.availHeight، ارتفاع قابل دسترس صفحه ی مونیتور را به پیکسل برمی گرداند. (این اندازه شامل taskbar نمی شود)

مثال

مثال زیر، ارتفاع صفحه مونیتورتان را برمی گرداند:

```
<script>

document.write("Available Height: " + screen.availHeight);

</script>
```

آموزش JS–متغییر cookie

cookieها معمولاً برای تشخیص هویت کاربر، استفاده می شوند.

---

cookie چیست؟

cookie متغیری است که در کامپیوتر بازدیدکنند ذخیره می شود و هر زمان که مرورگرِ کامپیوتر مذکور، درخواستی را به سرور ارسال کند، cookie نیز ارسال خواهد شد. با JavaScript هم می توان cookieها را ایجاد و هم بازیابی کرد.

چند مثال از کاربرد cookieها:

- **ذخیره نام در cookie**: اولین باری که بازدیدکننده به صفحه وب شما مراجعه می کند، نامش را وارد می کند. نام کاربر در یک cookie ذخیره می شود. در دفعات بعدی مراجعه به صفحه مذکور، یک پیغام خوش آمدگویی شبیه: "آقا/خانم x به وب سایت Beyamooz خوش آمدید." دریافت می کند. (نام کاربر از cookie دریافت می شود)
- **ذخیره تاریخ در cookie**: اولین باری که بازدیدکننده به صفحه وب شما مراجعه می کند، تاریخ جاری در یک cookie ذخیره می شود. در دفعات بعدی مراجعه کاربر، تاریخ ذخیره شده در cookie بازیافت می شود و یک پیغام شبیه: "آخرین بازدید شما در تاریخ دوشنبه ۲۱ اردیبهشت ۱۳۹۲ بوده است" نمایش داده می شود.

ایجاد و ذخیره یک cookie

در این مثال، یک cookie ایجاد خواهد شد و نام بازدیدکننده در آن ذخیره می شود.

در اولین باری که بازدیدکننده به صفحه وب شما مراجعه می کند، نام وی سوال می شود. نام بازدیدکننده در یک cookie ذخیره می شود و در دفعات بعدی که به صفحه وبِ مذکور مراجعه می کند، یک پیغام خوش آمدگویی دریافت خواهد کرد.

بنابراین ابتدا باید، یک تابع برای ذخیره نام بازدیدکننده ایجاد کنیم:

```
function setCookie(c_name,value,exdays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate() + exdays);
var c_value=escape(value) + ((exdays==null) ? "" : ";
expires="+exdate.toUTCString());
document.cookie=c_name + "=" + c_value;
}
```

پارامترهای ورودی تابع بالا:

۱. **c_name**: نام cookie

۲. **value**: مقدار cookie

۳. **exdays**: تعداد روز مانده تا منقضی شدن cookie

در تابع بالا، ابتدا برای بدست آوردن تاریخ انقضای cookie، پارامتر exdays را با تاریخ جاری جمع می کنیم و در متغیر exdate قرار می دهیم.

سپس نام cookie، مقدار cookie و تاریخ منقضی شدن cookie را در شیء document.cookie ذخیره می کنیم.

برگرداندن مقدار cookie

در ادامه برای برگرداندن مقدار cookie، تابع زیر را ایجاد می کنیم:

```
function getCookie(c_name)
{
var c_value = document.cookie;
var c_start = c_value.indexOf(" " + c_name + "=");
if (c_start == -١)
  {
  c_start = c_value.indexOf(c_name + "=");
  }
if (c_start == -١)
  {
  c_value = null;
  }
else
  {
  c_start = c_value.indexOf("=", c_start) + ١;
  var c_end = c_value.indexOf(";", c_start);
    if (c_end == -١)
    {
    c_end = c_value.length;
    }
  c_value = unescape(c_value.substring(c_start,c_end));
  }
```

```
return c_value;
}
```

در کد بالا، با استفاده از متد ()indexOf داخل رشته cookie، نام cookie مورد نظر را جستجو می کنیم. (c_name: نام cookie)

در متد ()indexof اولی، اگر مقدار (c_name + "=" + " ") در رشته cookie یافت نشد، در متد ()indexof داخل if مقدار (c_name + "=") جستجو می شود (بدون " ") بدین ترتیب در نهایت مکان شروعِ نام cookie در متغیر c_start ذخیره می شود.

در ادامه، در صورتی که مقدار متغیر c_start مخالف "۱-" باشد، محل شروع و پایان مقدار cookie را بدست می آوریم و با استفاده از متد ()substring آنرا برمی گردانیم.

چک کردن مقدار cookie

در نهایت، باید تابعی ایجاد نمایید که در صورت تنظیم cookie، یک پیغام خوش آمدگویی نمایش دهد و اگر cookie تنظیم نشده بود پنجره Prompt ظاهر شود و نام کاربر سوال شود و با صدا زدن تابع ()setCookie نام کاربر در cookie ذخیره شود.

```
function checkCookie()
{
var username=getCookie("username");
 if (username!=null && username!="")
 {
 alert("Welcome again " + username);
 }
else
 {
 username=prompt("Please enter your name:","");
 if (username!=null && username!="")
  {
  setCookie("username",username,٣٦٥);
  }
```

```
  }
}
```

همه توابع بالا در یک مثال:

مثال

```
<!DOCTYPE html>
<html>
<head>
<script>
function getCookie(c_name)
{
var c_value = document.cookie;
var c_start = c_value.indexOf(" " + c_name + "=");
if (c_start == -١)
{
c_start = c_value.indexOf(c_name + "=");
}
if (c_start == -١)
{
c_value = null;
}
else
{
c_start = c_value.indexOf("=", c_start) + ١;
var c_end = c_value.indexOf(";", c_start);
if (c_end == -١)
{
c_end = c_value.length;
}
c_value = unescape(c_value.substring(c_start,c_end));
}
return c_value;
```

```
}

function setCookie(c_name,value,exdays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate() + exdays);
var c_value=escape(value) + ((exdays==null) ? "" : ";
expires="+exdate.toUTCString());
document.cookie=c_name + "=" + c_value;
}

function checkCookie()
{
var username=getCookie("username");
if (username!=null && username!="")
 {
 alert("Welcome again " + username);
 }
else
 {
 username=prompt("Please enter your name:","");
 if (username!=null && username!="")
  {
  setCookie("username",username,٣٦٥);
  }
 }
}
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

در مثال بالا، زمانی که صفحه بارگذاری می شود، تابع ()checkCookie اجرا می شود.

آموزش JS–زمانبندی رویدادها

در JavaScript، می توان یک تابع را در فاصله های زمانی معین اجرا کرد.

این قابلیت زمانبدی رویدادها (Timing Events) نامیده می شود.

زمانبدی رویدادها (Timing Events) در JavaScript

در JavaScript، این امکان وجود دارد که یک قطعه کد را در فاصله های زمانی معیین (time-intervals) اجرا کنید. به این امکان "زمانبندی رویدادها" یا (Timing Events) می گویند.

دو متد (method) ساده برای زمانبدی رویدادها وجود دارد:

- **()setInterval**: در فاصله های زمانی معیین، یک تابع را بارها و بارها اجرا می کند.
- **()setTimeout**: بعد از گذشت یک فاصله زمانی معیین، یک تابع را یکبار اجرا می کند.

توجه: متدهای ()setInterval و ()setTimeout هر دو جزء متدهای شیء window در مدل DOM هستند.

متد ()setInterval

متد ()setInterval، در فاصله های زمانی معین، یک تابع مشخص را بارها و بارها اجرا می کند.

نحوه استفاده:

window.setInterval("*javascript function*",*milliseconds*);

متد ()window.setInterval را می توان بدون پیشوند window نیز نوشت.

پارامتر اول متد ()setInterval باید یک تابع باشد.

پارامتر دوم، فاصله های زمانی بین هر بار اجرای تابع را مشخص می کند. (برحسب میلی ثانیه)

توجه: هر ۱۰۰۰ میلی ثانیه معادل ۱ ثانیه است.

مثال

هر ۳ ثانیه یکبار، پیغام "Hello" نمایش داده می شود:

```
setInterval(function(){alert("Hello")},۳۰۰۰);
```

این مثال، فقط طرز کار متد ()setInterval را نشان می دهد و اینکه بخواهید هر ۳ ثانیه یکبار یک پیغام را نمایش دهید خیلی واقعی نیست.

در مثال زیر، متد ()setInterval برای نمایش یک ساعت دیجیتال، تابع ()myTimer را هر ۱ ثانیه یکبار اجرا می کند:

مثال

نمایش زمان جاری:

```
var myVar=setInterval(function(){myTimer()},۱۰۰۰);

function myTimer()
{
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
```

چگونه اجرا را متوقف کنیم؟

متد ()clearInterval، برای توقف اجرای تابع مشخص شده در ()setInterval استفاده می شود.

نحوه استفاده:

```
window.clearInterval(intervalVariable)
```

متد ()window.clearInterval را می توان بدون پیشوند window نیز نوشت.

برای اینکه بتوان از متد clearInterval() استفاده کرد، باید زمان ایجاد متد setInterval()، از یک متغیر عمومی استفاده کرد:

myVar=setInterval("*javascript function*",*milliseconds*);

سپس قادر خواهید بود برای توقف اجرای تابع، از متد clearInterval() استفاده کنید.

مثال

مثال زیر مانند قبل است، اما یک دکمه "Stop time" به آن اضافه شده است:

```
<p id="demo"></p>
<button onclick="myStopFunction()">Stop time</button>

<script>
var myVar=setInterval(function(){myTimer()},۱۰۰۰);
function myTimer()
{
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
function myStopFunction()
{
clearInterval(myVar);
}
</script>
```

متد setTimeout()

متد setTimeout()، بعد از گذشت یک فاصله زمانی معیین، یک تابع را یکبار اجرا می کند

نحوه استفاده:

window.setTimeout("*javascript function*",*milliseconds*);

متد ()window.setTimeout را می توان بدون پیشوند window نیز نوشت.

پارامتر اول متد ()setTimeout باید یک تابع باشد.

پارامتر دوم، نشان می دهد که از حالا چند میلی ثانیه دیگر می خواهید پارامتر اول، اجرا شود. (برحسب میلی ثانیه)

مثال

بعد از گذشت ۳ ثانیه پیغام "Hello" ظاهر می شود:

```
setTimeout(function(){alert("Hello")},۳۰۰۰);
```

چگونه اجرا را متوقف کنیم؟

متد ()clearTimeout، برای توقف اجرای تابع مشخص شده در ()setTimeout استفاده می شود.

نحوه استفاده:

```
window.clearTimeout(timeoutVariable)
```

متد ()window.clearTimeout را می توان بدون پیشوند window نیز نوشت.

برای اینکه بتوان از متد ()clearTimeout استفاده کرد، باید زمان ایجاد متد ()setTimeout، از یک متغیر عمومی استفاده کرد:

```
myVar=setTimeout("javascript function",milliseconds);
```

سپس، اگر تابع قبلاً اجرا نشده باشد، قادر خواهید بود برای توقف اجرای تابع، از متد ()setTimeout استفاده کنید.

مثال

مثال زیر مانند قبل است، اما یک دکمه "Stop time" به آن اضافه شده است:

```
var myVar;

function myFunction()
{
```

```
myVar=setTimeout(function(){alert("Hello")},۳۰۰۰);
}

function myStopFunction()
{
clearTimeout(myVar);
}
```

آموزش JS–مدیریت خطاها

دستور **try**، به شما اجازه می دهد تا، خطاهای یک بلاک از دستورات را تست کنید.

دستور **catch**، خطاهای اتفاق افتاده در قسمت try را بررسی می کند.

دستور **throw**، به شما اجازه می دهد تا خطاهای سفارشی ایجاد کنید.

دلیل اتفاقِ خطاها!

زمانی که موتور JavaScript کدها را اجرا می کند، ممکن است خطاهای زیر اتفاق بیافتد:

۱. خطای املای دستورات (syntax error)، که معمولاً توسط برنامه نویس اتفاق می افتد.

۲. خطای گم شدگی خصوصیت (missing feature)، که ممکن است بخاطر تفاوت های بین مرورگرها و یا خطای املایی باشد.

۳. خطای ورودی اشتباه (wrong input)، که می تواند توسط یک کاربر باشد.

و البته این خطاها می تواند چیزهای غیرقابل پیشبینی دیگری نیز باشد.

پرتاب خطا (Throw Error)

زمانی که خطایی اتفاق می افتد، به طور معمول، اجرای دستورات متوقف شده و یک پیغام خطا تولید می شود.

اصطلاح تکنیکی برای این عمل، عبارت throw است (به معنی پرتاب کردن، به اصطلاح یک پیغام خطا به سمت catch پرتاب می شود)

try و catch در JavaScript

عبارت try، به شما اجازه می دهد تا، خطاهای یک بلاک از دستورات را تست کنید.

اگر خطایی در قسمت try اتفاق بیافتد، دستورات تعریف شده در قسمت catch اجرا می شوند. (catch به معنی گرفتن، به اصطلاح خطاهای پرتاب شده از قسمت try در قسمت catch گرفته می شوند)

try و catch ها همیشه جفت جفت می آیند.

نحوه استفاده

```
try
 {
 //Run some code here
 }
catch(err)
 {
 //Handle errors here
 }
```

مثال

در مثال زیر، در قسمت try، عمداً یک خطای املایی ایجاد کرده ایم.

قسمت catch خطایی که در try اتفاق افتاده را می گیرد و پیغام مناسب را چاپ می کند:

مثال

```
<!DOCTYPE html>
<html>
<head>
<script>
var txt="";
function message()
{
try
```

```
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.message + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>

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

</html>
```

دستور throw

دستور **throw**، به شما اجازه می دهد تا خطاهای سفارشی ایجاد کنید.

اصطلاح تکنیکی صحیح برای ایجاد خطا: عبارت **"throw an exception"** است.

دستور throw باید همراه try و catch استفاده شود.

نحوه استفاده

throw *exception*

exception می تواند یک رشته، یک عدد، یک داده Boolean و یا یک Object باشد.

مثال

این مثال، مقدار ورودی متغیر را تست می کند. اگر مقدار اشتباه باشد، یک خطا (exception) به سمت catch پرتاب (throw) می شود. قسمت catch خطا را می گیرد و پیغام مناسب را چاپ می کند.

مثال

```
<script>
function myFunction()
{
try
  {
  var x=document.getElementById("demo").value;
  if(x=="")    throw "empty";
  if(isNaN(x)) throw "not a number";
  if(x>١٠)     throw "too high";
  if(x<٥)      throw "too low";
  }
catch(err)
  {
  var y=document.getElementById("mess");
  y.innerHTML="Error: " + err + ".";
  }
}
</script>

<h١>My First JavaScript</h١>
<p>Please input a number between ٥ and ١٠:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="mess"></p>
```

توجه داشته باشید که در مثال بالا اگر در تابع getElementById مشکلی رخ دهد، خطای مناسب با آن پرتاب (throw) خواهد شد.

آموزش JS–اعتبارسنجی فرم

اعتبارسنجی (Validation) فرم ها در JavaScript

از JavaScript می توان برای اعتبار سنجی فرم ها، قبل از اینکه به سرور ارسال شود استفاده نمود.

فرم های داده ای که توسط JavaScript اعتبارسنجی می شوند به صورت معمول می تواند شامل موارد زیر باشد:

- آیا فیلدهای الزامی (ستاره دار) پر شده است؟
- آیا کاربر آدرس ایمیل را به فرمت صحیح وارد کرده است؟
- آیا کاربر، داده صحیح را وارد کرده است؟
- آیا کاربر در یک فیلد عددی، فقط کاراکترهای عددی را وارد کرده است؟

فیلدهای الزامی (Required)

تابع زیر، خالی بودن فیلد را چک می کند. اگر خالی باشد، پیغام "First name must be filled out" ظاهر می شود و فرم به سرور ارسال نمی شود:

```
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
  {
  alert("First name must be filled out");
  return false;
  }
}
```

تابع بالا، زمانی فراخوانی می شود که، رویداد "onsubmit" فرم اتفاق بیافتد:

مثال

```
<form name="myForm" action="js_validation.php" onsubmit="return
validateForm()" method="post">
First name: <input type="text" name="fname">
```

```
<input type="submit" value="Submit">
</form>
```

اعتبار سنجی ایمیل (E-mail Validation)

تابع زیر، فرمت معمول آدرس ایمیل را چک می کند.

فرمت صحیح آدرس ایمیل شامل موارد زیر است:

١. باید شامل یک علامت "@" باشد.

٢. باید حداقل یک نقطه (.) داشته باشد باشد.

٣. علامت "@" نباید اولین کاراکتر باشد.

٤. آخرین نقطه، باید بعد از علامت "@" ظاهر شود.

٥. بعد از آخرین نقطه باید حداقل ٢ کاراکتر بیاید.

```
function validateForm()
{
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<١ || dotpos<atpos+٢ || dotpos+٢<=x.length)
 {
 alert("Not a valid e-mail address");
 return false;
 }
}
```

تابع بالا، زمانی فراخوانی می شود که، رویداد "onsubmit" فرم اتفاق بیافتد:

مثال

```
<form name="myForm" action="js_validation.php" onsubmit="return
validateForm();" method="post">
```

Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>

متفرقه JS–رسم خط

برای رسم خط در جاوا اسکریپت ما ابتدا نیاز داریم که تگ <canvas> را بشناسیم.

همچنین ما باید دستورات زیر را بیاموزیم:

۱. document.getElementById()

۲. getContext("۲d")

۳. beginPath()

۴. moveTo()

۵. lineTo()

۶. stroke()

در ادامه به توضیح هریک از این دستورات می پردازیم.

معرفی تگ <canvas>

برای رسم گرافیک در وب ما نیاز به تگ <canvas> داریم.

با استفاده از تگ <canvas> ما قادر خواهیم بود انواع خط ها و دایره ها و بیضی ها و حتی انیمیشن ها را رسم کنیم .

کلمه ی canvas به معنی پرده نقاشی یا بوم نقاشی است و همانطور که از اسم آن بر می آید در آن می توان هر چیزی را رسم کرد .

تگ <canvas>وقتی که در ابتدا ایجاد می شود خالی است و چیزی درون آن نیست .

مثال ۱ : رسم خط با استفاده از <canvas> و جاوا اسکریپت

```
var c=document.getElementById("cvs");
var context=c.getContext("٢d");
context.beginPath();
context.moveTo(٠,٠);
context.lineTo(٣٠٠,١٥٠);
context.stroke();
```

توجه : فعلا لازم نیست که دستورات بالا را کاملا بلد باشید . در ادامه ی این مطلب تمامی این دستورات شرح داده خواهند شد .

وقتی مثال بالا را مشاهده کردید درون ویرایشگر کد های زیر را خواهید دید :

کدهای مثال ۱

```
١ <html>
٢ <body>
٣ <canvas id="cvs" width="٣٠٠px" height="١٥٠px" style="border:١px solid
#DFDFDF">
٤ </cancas>
٥  <script>
٦       var c=document.getElementById("cvs");
٧       var context=c.getContext("٢d");
٨       context.beginPath();
٩       context.moveTo(٠,٠);
١٠      context.lineTo(٣٠٠,١٥٠);
١١      context.stroke();
١٢  </script>
١٣ </body>
```

١٤ </html>

تشریح کد ها :

در خط ١ یک سند HTML تعریف شده است .

در خط ٢ بدنه ی HTML تعریف شده است .

در خط ٣ همانطور که مشاهده می کنید ما یک تگ <canvas> را تعریف کرده ایم . و ID آن را cvs گذاشته ایم . و در ادامه عرض <canvas> را برابر با ٣٠٠ پیکسل و ارتفاع

آن را برابر با ١٥٠ پیکسل قرار داده ایم . بعد از این ما یک استایل درون خطی تعریف کرده ایم . در درون این استایل با استفاده از border یک کادر با اندازه یک پیکسل و با رنگ تیره تعریف کرده ایم . توجه کنید که #DFDFDF یک عدد هگزادسیمال است که بیان گر یک رنگ است .

در خط ٤ تگ <canvas> را بسته ایم .

در خط ٥ یک تگ <script> ایجاد می کنیم و در ادامه دستورات رسم خط را وارد می کنیم .

در خط ٦ یک متغیر به نام c ایجاد کرده ایم و آن را برابر با دستور document.getElementById("cvs") قرار می دهیم . ممکن است در آموزش های مقدماتی این دستور را قبلا مطالعه کرده باشید اما در اینجا توضیح مختصری درباره ی آن می دهیم . دستور document.getElementById() برای ارتباط برقرار کردن با یک عنصر HTML است . اگر بخواهیم آن را تفسیر کنیم مثل این است که شما به رایانه بگویید : عنصر HTML که ID آن را دارد پیدا کن و با آن ارتباط برقرار کن و در متغیر c ذخیره کن . حال عنصر مورد نظر با ID مشخص شده است .

در خط ٧ ابتدا باید بفهمیم که دستور c.getContext("d") چه کاری را انجام می دهد .این دستور متدهای لازم برای رسم گرافیک درون تگ <canvas> را فراهم می کند . در این خط ما یک متغیر به نام context را تعریف کردیم و آن را برابر با c.getContext("d") قرار داده ایم . اگر شما برنامه نویسی شئ گرا را به طور عمیق مطالعه نکرده اید لازم نیست که در فهم خط ٧ تلاش زیادی بکنید ، فقط کافیست که بدانید برای رسم در تگ <canvas> همیشه باید این خط را به کد های خود اضاف کنید .

در خط ٨ از دستور beginPath() استفاده کرده ایم ، از این دستور برای ایجاد یک رسم جدید استفاده می شود .

در خط ۹ از دستور moveTo(x,y) استفاده کرده ایم . همانطور که می دانید برای رسم یک خط باید دو نقطه داشته باشیم و آن دو نقطه را به هم وصل کنیم تا یک خط به وجود آید . این دستور نقطه ی شروع رسم خط را مشخص می کند . به عبارت دیگر شما به رایانه می گویید که به نقطه ی (x,y) برو تا رسم را از این نقطه شروع کنیم .

در خط ۱۰ از دستور lineTo(x,y) استفاده کرده ایم . این دستور نقطه ی دوم را مشخص کرده و از نقطه ی قبلی به این نقطه ی جدید خطی را رسم می کند .

در خط ۱۱ از دستور stroke() استفاده کرده ایم . با اجرای این دستور خط مورد نظر به تگ <canvas> اضافه می شود . و در آخر در خطوط ۱۲ و ۱۳ و ۱٤ تگ های باز شده را می بندیم .

دستور	عملکرد
document.getElementById()	برای ارتباط با یک عنصر به وسیله ID استفاده می شود
getContext("d۲")	ایجاد متد های لازم برای برای رسم گرافیک
beginPath()	برای ایجاد یک رسم جدید استفاده می شود
moveTo()	نقطه ی شروع رسم را معین می کند
lineTo()	نقطه پایانی رسم خط را مشخص می کند
stroke()	خط رسم شده را به محیط گرافیکی اضافه می کند

جدول خلاصه دستورات به کار برده شده در این مثال :