

آموزش کاربردی جاوا اسکریپت

Java Script

Applied JavaScript for Web developers

نویسنده :



احمد بادپی



انجمن علمی مهندسی فناوری اطلاعات
دانشگاه پیام نور مرکز آران و بیدگل

پیشگفتار

در سال های نه چندان دور، وب مکان بسیار سرد و کسل کننده ای بود. صفحات وبی که از متن های ساختمند HTML ایجاد شده بودند تنها به نمایش یکسری اطلاعات اکتفا می کردند. تنها راه تعامل کاربران با صفحات این بود که بر روی لینک کلیک کرده و منتظر بارگذاری یک صفحه جدید می ماندند. حتی فکر آن روزها هم عذاب آور است!

اما امروزه بسیاری از سایت ها همچون برنامه های کاربردی دسکتاب عمل کرده و می توانند بی درنگ به تمامی رویدادهای صفحه همچون کلیک پاسخ بدهند. و همه این ها به لطف موضوع جزو ای که هم اکنون در دست دارید است: جاواسکریپت!

جاواسکریپت زبانی است که با کمک آن می توانید صفحات HTML خود را لبریز از اینیشن ها، جلوه های بصری و ویژگی های تعاملی کنید. این زبان امکاناتی را فراهم می آورد تا صفحه وب شما بتواند به عملیات کاربری که در تعامل با آن هاست فورا پاسخ بدهد. مانند کلیک بر روی یک لینک، پر کردن یک فرم و حرکت نشانگر ماوس بر روی صفحه.

به کمک این زبان خواهید توانست به اجزا موجود در صفحه دسترسی داشته و حتی در صورت نیاز دست به تغییراتی در آن ها مطابق میل خود بزنید.

استفاده روز افرون از تکنولوژی هایی همچون Ajax و JQuery در طراحی صفحات وب که هسته اصلی آن ها را تشکیل می دهد به اهمیت یادگیری این زبان قدرتمند برای هر کس که دستی در توسعه وب دارد افروده است.

جزوه پیش رو که در راستای برگزاری دوره های آموزشی طراحی وب به همت انجمن علمی مهندسی فناوری اطلاعات دانشگاه پیام نور مرکز آران و بیدگل تهیه شده است شما را با بسیاری از ویژگی های این زبان محبوب و کاربردی آشنا خواهد کرد.

این جزو از روی یکی از مشهورترین و جامع ترین کتب آموزشی جاواسکریپت یعنی Professional Javascript for Web نوشته Nicholas C.Zakas Developers (نیکلاس سی زاکاس) و تجربیات شخصی تالیف و ترجمه شده است.

ذکر این نکته نیز ضروری است که ترجمه، تخصص اینجانب نبوده و هدف از تهیه جزو ای که هم اکنون در دستان شماست هم چیزی جز جامه عمل پوشاندن به حدیث شریف «زَكَاهُ الْعِلْمِ نَسْرُهُ» نبوده است. از این رو این جزو نیز خالی از اشکال و اشتباه نیست. لذا از تمامی خوانندگان تقاضا می شود به منظور گزارش اشکال و بیان نظرات، انتقادات و پیشنهادات خود با ایمیل اینجانب به آدرس ahmadbadpey@gmail.com مکاتبه فرمایند.

در پایان از تمامی دوستانم علی الخصوص حسین بیدی به خاطر ویرایش و صفحه آرایی و همچنین طراحی طرح روی جلد ، تمامی اعضای محترم انجمن علمی مهندسی فناوری اطلاعات دانشگاه پیام نور مرکز آران و بیدگل و تمامی دوستانی که با اعتماد به بند و حضور گرم خود در دوره های آموزشی برگزار شده موجبات دلگرمی و انگیزه برای تهیه این جزو را فراهم آورند صادقانه و صمیمانه تشکر می کنم.

احمد بادپی - فروردین ۱۳۹۱

مولای متقيان، علی(عليه السلام) می فرمایند:

إِذَا أَرْذَلَ اللَّهُ عَبْدًا حَظَرَ عَلَيْهِ الْعِلْمَ

هرگاه خداوند نخواهد بنده ای را به رؤالت برساند، اور از علم و دانش و یادگار فتن محروم می سازد.

فهرست

I	پیشگفتار
فصل ۱ : آشنایی با مفاهیم و اصطلاحات	
۱	تفاوت های جاوا اسکریپت و جاوا
۲	انواع زبان های برنامه نویسی تحت وب
۳	اجزا تشکیل دهنده جاوا اسکریپت
۴	۱۴؛ مدل شی گرای سند DOM
۵	۱۵؛ مدل شی گرای مرورگر BOM
۶	ویژگی های بنیادی جاوا اسکریپت
فصل ۲ : متغیرها و انواع داده ها	
۷	متغیرها در جاوا اسکریپت
۸	نامگذاری متغیرها
۹	کلمات کلیدی
۱۰	کلمات رزرو شده
۱۱	انواع داده های اصلی
۱۲	نوع داده Undefined
۱۳	نوع داده Null
۱۴	نوع داده Boolean
۱۵	نوع داده Number
۱۶	نوع داده String
۱۷	تبديل انواع
۱۸	تبديل به (شته)

۱۳..... تبدیل به عدد.....

۱۴..... استفاده از Type Casting برای تبدیل انواع.....

فصل ۳ : جاوا اسکریپت در مرورگرها.....

۱۵..... فایل های خارجی javascript.....

۱۶..... تفاوت های به کارگیری کدها به صورت درون خطی و خارجی.....

۱۷..... مکان قرار دادن تگ <script> در صفحه.....

۱۸..... مخفی کردن اسکریپت ها از مرورگرهای قدیمی.....

۱۹..... خطایابی.....

فصل ۴ : کار با آرایه ها در جاوا اسکریپت.....

۲۰..... ایجاد آرایه ها با استفاده از کلاس Array.....

۲۱..... بدست آوردن طول آرایه.....

۲۲..... تبدیل آرایه به رشته.....

۲۳..... تبدیل رشته به آرایه.....

۲۴..... اضافه کردن مقادیر جدید به آرایه ها.....

۲۵..... برگرداندن عناصر خاصی از آرایه.....

۲۶..... تبدیل آرایه ها به پشته و صف.....

۲۷..... مرتب سازی آرایه ها.....

۲۸..... حذف و درج در میانه های آرایه.....

فصل ۵ : کار با رشته ها در جاوا اسکریپت

۲۹..... ایجاد اشیا رشته ای (رشته) با استفاده از کلاس String.....

۳۰..... بدست آوردن کاراکتر موجود در یک موقعیت خاص.....

۳۱..... الحاق دو رشته.....

۳۲..... عملگر + برای الحاق (رشته ها).....

۳۳..... بدست آوردن موقعیت یک کاراکتر خاص در رشته.....

۳۵.....	مقایسه رشته ها
۳۶.....	جدا کردن زیر رشته ای از رشته دیگر
۳۷.....	toLowerCase() وtoUpperCase()

فصل ۶ : اشیای درونی (پیش ساخته)

۴۰.....	Math شی
۴۱.....	متدهای max() و min()
۴۲.....	دیگر توابع مفید
۴۳.....	کار با تاریخ و زمان در جاوا اسکریپت

فصل ۷ : BOM؛ مدل شی گرای مرورگر

۴۶.....	BOM چیست؟
۴۷.....	window شی
۴۸.....	دستکاری پنجره ها
۴۹.....	پیمايش و باز کردن پنجره های جدید
۵۰.....	پنجره های System Dialog
۵۱.....	خاصیت statusbar
۵۲.....	اجرای مکرر کدها از طریق متدهای Intervals و Timeouts
۵۳.....	history شی
۵۴.....	document شی
۵۵.....	location شی
۵۶.....	navigator شی
۵۷.....	screen شی

فصل ۸ : DOM؛ مدل شی گرای سند

۵۸.....	DOM چیست؟
۶۰.....	استفاده از DOM

۴۰	دسترسی به گره ها
۴۱	دسترسی به صفات عناصر
۴۱	دسترسی به گره های خاص
۴۱	getElementsByName()
۴۲	getElementsByName()
۴۲	getElementById()
۴۳	ایجاد و دستکاری گره ها
۴۳	ایجاد گره های جدید
۴۴	createElement() و createTextNode() .appendChild()
۴۵	insertBefore() replaceChild() .removeChild()
۴۵	createDocumentFragment()
۶۶	ویژگی های منحصر به فرد DOM برای HTML
۶۷	دستکاری قواعد سبک عناصر
۶۷	متدهای مربوطه به جداول
۶۹	متد ها و خاصیت های tbody
۶۹	متد ها و خاصیت های tr
۷۱	فصل ۹ : کار با فرمهای و عناصر فرم
۷۲	نوشتن اسکریپت ها برای دسترسی به عناصر فرم
۷۲	ایجاد ارجاع به عناصر مورد نظر
۷۲	دسترسی به عناصر داخل یک فرم
۷۳	ویژگی ها و خاصیت های عناصر form
۷۴	ارسال فرم بوسیله جاوااسکریپت
۷۵	ارسال form فقط یکبار !
۷۵	کار با کادرهای متنی

۷۶	بازیابی و تغییر مقدار یک textbox
۷۶	انتخاب متن های داخل کادرهای متنی
۷۶	رویداد های کادرهای متنی
۷۷	انتخاب فودکار متن درون کادرهای متنی
۷۷	چرخش Tab بین عناصر فرم به صورت خودکار
۷۸	محدود کردن کاراکتر های ورودی در یک textarea
۷۹	کار با combobox ها و listbox ها
۸۰	بازیابی/تغییر دادن option(ها) ای انتخاب شده
۸۱	اضافه کردن option ها
۸۱	حذف option ها

فصل ۱۰ : رویدادها در جاوا اسکریپت

۸۴	کنترل رویدادها
۸۴	انواع رویداد ها
۸۴	(رویدادهای mouse)
۸۵	ترتیب اجرایی رویدادها
۸۵	(رویدادهای صفحه کلید)
۸۶	ترتیب اجرایی رویداد های صفحه کلید
۸۶	دیگر رویداد ها
۸۷	event شی
۸۸	فواص و متدهای شی event

فصل ۱۱ : کار با کوکی ها

۹۰	ایجاد کوکی ها
۹۱	حذف کوکی ها
۹۱	بازیابی کوکی ها

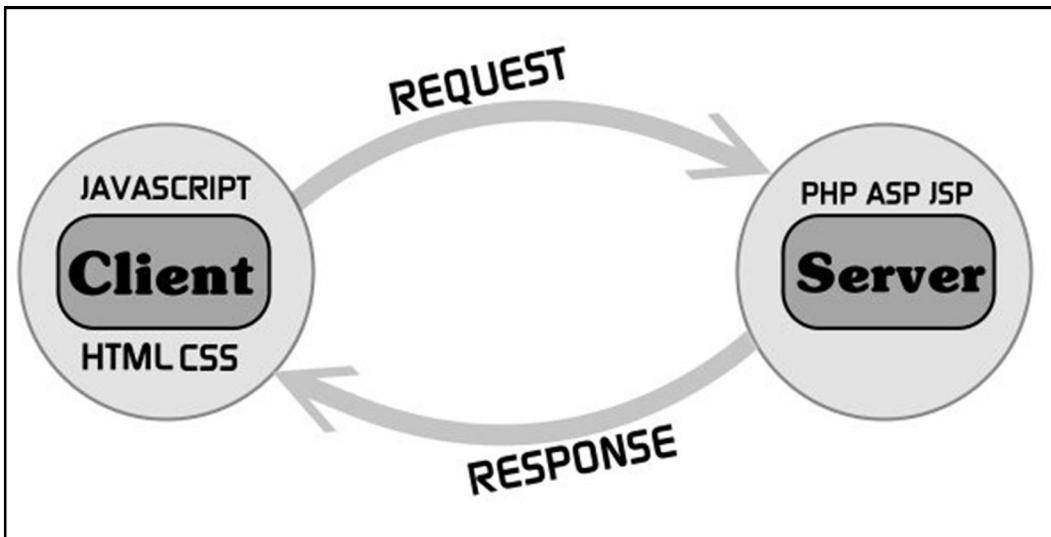
فصل یک

آشنایی با مفاهیم و اصطلاحات

این فصل اختصاص به بررسی مفاهیم و اصطلاحات رایج موجود در جاوا اسکریپت دارد. در این فصل ابتدا به انواع زبان های برنامه نویسی تحت وب خواهیم پرداخت و سپس تفاوت های اصلی زبان های جاوا اسکریپت و جاوا را شرح خواهیم داد. در ادامه نیز به هسته های تشکیل دهنده جاوا اسکریپت پرداخته و با کاربردهای هر یک از آن ها آشنا خواهیم شد.

انواع زبان های برنامه نویسی تحت وب

همانطور که می دانید کامپیوتر های موجود در شبکه اینترنت را به دو دسته اصلی تقسیم می کنند. کامپیوتر های کاربر^۱ و کامپیوتر های سرور^۲. زبان های برنامه نویسی تحت وب نیز به دو دسته تحت کاربر^۳ و تحت سرور^۴ تقسیم بندی می شوند.



زبان های تحت کاربر زبان هایی هستند که بوسیله مرورگر و فقط بر روی کامپیوترهای مشتری اجرا می شوند. در واقع برای اجرای این گونه زبان ها به سرورها نیازی نیست. زبان هایی همچون HTML، CSS و JAVASCRIPT از این دست هستند. از این زبان ها معمولاً به تنها برای ایجاد سایت های با محتوای ثابت که اصطلاحاً به آن ها سایت های ایستا^۵ می گویند استفاده می شود.

در مقابل این زبان ها، زبان های تحت سرور وجود دارند که برای اجرا نیاز به سرور ها داشته و می بایست برای اجرا حتماً بر روی سرور ها قرار بگیرند. اینگونه زبان ها امکان برقراری ارتباط با پایگاه داده^۶ را دارند. زبان هایی همچون PHP، ASP و JSP از این دست هستند. از این زبان ها برای ایجاد سایت های با محتوای پویا که اصطلاحاً به آن ها سایت های پویا^۷ گفته می شود استفاده می شود.

زبان JavaScript یکی از زبان های مهم برنامه نویسی وب و تحت کاربری باشد. این زبان اولین بار در سال ۱۹۹۵ ارائه شد و وظیفه آن تنها ارزش سنجی عناصر فرم بود.

¹ Client

² Server

³ Client side

⁴ Server side

⁵ Static

⁶ Database

⁷ Dynamic

تفاوت های جاوا اسکریپت و جاوا

این سوال که تفاوت زبان های جاوا اسکریپت و جاوا چیست همواره یکی از دغدغه های بسیاری از توسعه دهنده‌گان تازه کار وب به شمار می‌رود. جالب است بدانید صرفنظر از تشابه اسمی این دو زبان و تشابه نحو و دستورات آن‌ها با زبان C++, C++، تفاوت های بسیاری بین آن‌ها وجود دارد که در ادامه به برخی از آن‌ها اشاره می‌کنیم:

جاوا یک زبان برنامه نویسی کاملاً شی گرا^۱ (OOP) است که اولین بار توسط شرکت Sun MicroSystem به منظور خلق برنامه‌های کاربردی مستقل و قابل اجرا بر روی انواع سیستم‌های عامل ارائه شد. در حالی که جاوا اسکریپت به عنوان یک زبان شبیه شی گرا^۲ (LOO^۳) که اولین بار توسط شرکت NetScape ارائه شد، تنها یک فایل متنی ساده است که نمی‌توان از آن برای ایجاد برنامه‌های کاملاً مستقل استفاده کرد و برای اجرا می‌باشد در داخل صفحات HTML قرار گرفته و توسط مرورگرها تفسیر و اجرا شوند. در واقع کاربرد اصلی جاوا اسکریپت در صفحات وب بوده و از آن تنها به منظور افزودن قابلیت‌های تعاملی به صفحات وب استفاده می‌شود. البته نباید از ذکر این نکته نیز گذشت که در سال‌های اخیر امکان کاربرد برنامه‌های جاوا نیز در قالب Applet^۴ ها و صفحات JSP^۵ در وب فراهم شده است.

جاوا یک زبان کامپایلی است در حالی که جاوا اسکریپت همان طور که از اسمش پیداست یک زبان اسکریپتی (تفسیری) است. زبان‌های کامپایلی به زبان‌هایی گفته می‌شود که قبل از اجرا می‌باشد کامپایل شوند. زبان‌های اسکریپتی نیز به زبان‌هایی گفته می‌شود که مرحله کامپایل و اجرا آن‌ها جدا نبوده و در واقع کامپایل آن‌ها در زمان اجرا انجام می‌شود. وظیفه تفسیر برنامه‌های جاوا اسکریپت بر عهده مرورگر است. به برنامه‌هایی که به زبان‌های اسکریپتی نوشته می‌شوند اسکریپت می‌گویند.

از تفاوت‌های مهم دیگر این دو زبان می‌توان به سیک تعریف متغیرها در آن‌ها اشاره کرد. زبان‌های برنامه نویسی از لحاظ تعریف متغیرها به دو دسته زبان‌های Loosely Type و Strongly Type تقسیم می‌شوند. در زبان‌های با نوع قوی می‌باشد ابتدا نوع متغیرها را تعیین و سپس در برنامه از آن استفاده نمود. نوع این گونه متغیرها را نمی‌توان در طول اجرا برنامه تغییر داد و در صورتی که این متغیرها با عملگرهای مناسب خود به کار نروند نتایج نادرست به دست می‌آیند و یا خطایی به وقوع می‌پیوندد. زبان‌های C++ و java از این دست زبان‌ها هستند.

در مقابل در زبان‌های با نوع ضعیف نیازی به تعریف متغیرها و تعیین نوع داده آن‌ها نمی‌باشد. در این زبان‌ها تعیین نوع های داده به طور خودکار و بر حسب نیاز توسط خود زبان انجام می‌گیرد و بنابراین در طی فرآیند پردازش داده‌ها می‌توان در هر مرحله به راحتی نوع داده‌ها را بررسی و تغییر داد. زبان‌هایی همچون javascript و PHP از این دست هستند.

یادگیری جاوا اسکریپت بسیار ساده‌تر از جاوا است. این به این خاطر است که همه آنچه شما به عنوان یک توسعه دهنده وب برای یادگیری جاوا اسکریپت نیاز دارید در کی عمیق از HTML است. با این حال چنانچه در ک درستی از جاوا اسکریپت داشته باشید یادگیری جاوا نیز برایتان سهل و دلپذیر خواهد شد!

¹ Object Oriented Programming

² Like Object Oriented

³ Java Server Page

اجزا تشکیل دهنده جاوااسکریپت

۱ DOM: مدل شی گرای سند

DOM یکی از API ها (رابط برنامه نویسی) برای زبان های HTML و XML به شمار می رود. DOM تمام عناصر موجود در یک صفحه وب را به صورت درختی از گره ها^۳ نمایش می دهد و امکان کنترل آن ها برای توسعه دهندهان وب را فراهم می آورد. با استفاده از DOM می توان گره ها را به راحتی حذف، اضافه، جابجا و یا جایگزین کرد.

۲ BOM: مدل شی گرای مرورگر

یکی دیگر از API های ساخته شده برای HTML که به عنوان یکی از ویژگی های منحصر به فرد مرورگرهای IE و Netscape نیز شناخته می شود BOM است.

از BOM برای دسترسی و دستکاری ویژگی های پنجره یک مرورگر می توان استفاده کرد. توسعه دهندهان وب با استفاده از BOM می تواند کارهایی همچون جابجایی پنجره ها و تغییر متن موجود در نوار وضعیت مرورگر و دیگر کارهایی که ارتباط مستقیمی با محتوای تشکیل دهنده صفحه (سند) ندارند انجام دهند. معمولاً BOM با پنجره ها و فریم ها سرو کار داشته و می توان از طریق آن کارهای زیر را انجام داد:

- باز کردن پنجره های `popup`.
- توانایی باز کردن پنجره های جدید و تغییر اندازه و جابجایی و یا بستن آن ها.
- بدست آوردن اطلاعاتی از مرورگر و سیستم عامل کاربران همچون نوع، نسخه و...
- بدست آوردن اطلاعاتی در مورد سند و موقعیت صفحه ای که در مرورگر باز شده است.
- بدست آوردن اطلاعاتی در مورد `وضوح`^۴ صفحه نمایش کاربر.
- پشتیبانی از `cookie` ها.

به دلیل عدم وجود استانداردی واحد برای BOM، هر مرورگر ممکن است به صورتی متفاوت از آن پشتیبانی کند. مانند اشیا `window` و `navigator` که هر مرورگر متدها و خواص منحصر به فردی برای آن ها تعریف کرده است.

۳ ویژگی های بنیادی جاوااسکریپت

اینک به چند مفهوم اصلی در زبان javascript می پردازیم:

جاوا اسکریپت حساس به حروف^۶ است: یعنی همه چیز مانند نام متغیر ها ، نام توابع ، عملگر ها و هر چیز دیگری نسبت به حروف کوچک و بزرگ حساس است. به عنوان مثال متغیری با نام `Test` با متغیری با نام `test` متفاوت است.

¹ Document Object Model

² Application Programming Interface

³ Node

⁴ Browser Object Model

⁵ Resolution

⁶ Case Sensitive

آموزش کاربردی جاوا اسکریپت - احمد بادپی

متغیرها بدون نوع هستند: برخلاف زبان هایی همچون java و C ، متغیرها نوع خاصی نمی گیرند. در عوض هر متغیر می تواند با کلمه کلیدی var تعریف شده و مقداری را به عنوان مقدار اولیه پذیرد. در واقع متغیرها "مقدار گرا" هستند. یعنی در هنگامی که تعریف (مقداردهی) می شوند نوعشان نیز مشخص می گردد. این ویژگی امکان تغییر نوع داده ذخیره شده در یک متغیر در هر نقطه ای از برنامه را فراهم می کند.

قرار دادن (;) در انتهای هر دستور اختیاری است: دستورات در جاوا اسکریپت می توانند به ؛ ختم شوند یا نشوند. در صورت چشم پوشی از ؛ ، جاوا اسکریپت انتهای هر خط را به عنوان پایان دستور در نظر خواهد گرفت. با این حال روش صحیح ، استفاده از ؛ در انتهای دستورات است. چون بعضی از مرورگرها از روش اول پشتیبانی نمی کند و ممکن است در اجرای کدها دچار مشکل شوند.

درج توضیحات در جاوا اسکریپت: برای درج توضیحات در میان کدها می توان از روش های زبان های برنامه نویسی همچون C و C++ استفاده نمود یعنی از // برای توضیحات یک خطی یا /* */ برای توضیحات چند خطی:

```
//this is a single-line comment
/* this is a multiline
comment */
```



فصل دو

متغیرها و انواع داده‌ها

در این فصل ابتدا با روش تعریف متغیرها و قوانین نامگذاری آن‌ها در جاوا اسکریپت آشنا خواهیم شد و سپس انواع داده‌های موجود را بررسی خواهیم کرد. همچنین لیستی از کلمات کلیدی و رزرو شده که امکان استفاده از آن‌ها به عنوان نام متغیرها و توابع وجود ندارد را ارائه خواهیم کرد.

همچنین در این فصل به مبحث بسیار مهم تبدیل انواع پرداخته و روش‌های تبدیل انواع گوناگون به یکدیگر را بررسی خواهیم کرد.

متغیرها و انواع داده ها

متغیرها در جاوا اسکریپت

متغیر ها با کلمه var تعریف می شوند. مانند:

```
var test = 'ali' ;
```

۱

۲

در این مثال متغیری با نام test اعلام شده و مقدار اولیه 'ali' را می گیرد.

چون متغیرها بدون نوع هستند مفسر جاوا اسکریپت خود به خود نوع test را رشته در نظر می گیرد.

همچنین می توانیم دو یا چند متغیر را همزمان تعریف کنیم:

```
var test1='ali' , test2='salam' ;
```

باید توجه داشته باشیم متغیرهایی که با یک var تعریف می شود ممکن است نوع یکسانی نداشته باشند.

```
var test_1='ali' , age=25;
```

برخلاف جاوا در جاوا اسکریپت متغیرها می توانند مقدار اولیه نگیرند.

```
var test ;
```



برخلاف جاوا متغیرها می توانند در زمان های مختلف مقادیر متفاوتی داشته باشند. این یکی از امتیازات متغیرهای بدون نوع در زبان جاوا اسکریپت به شمار می رود. به عنوان مثال یک متغیر می تواند باید یک مقدار رشته ای مقداردهی اولیه شده و سپس در ادامه برنامه به یک مقدار عددی تغییر کند. به مثال زیر دقت کنید:

```
var test ="hi" ;
alert(test); // hi
test=55;
alert(test); // 55
```

نامگذاری متغیرها

نامگذاری متغیرها می بایست شرایط زیر را داشته باشد:

اولین کاراکتر متغیر می تواند یک حرف، یک underline (_) و یا یک علامت \$ باشد.

بقیه کاراکترها می توانند از \$، _ و یا هر حرف و عددی تشکیل شوند.

تمام متغیرهای زیر صحیح هستند:

```
var test ;
var $test ;
var $1 ;
var _$test2 ;
```

البته صحت نام یک متغیر از نظر نحوی، به این معنی نیست که می توانید از آن ها استفاده کنید. بهتر است در نامگذاری متغیرها از یکی از قراردادهای شناخته شده زیر تبعیت کنید:

آموزش کاربردی جاوا اسکریپت - احمد بادپی

- نشانه گذاری شتری^۱: در این قرارداد، حرف اول متغیر کوچک و حرف اول بقیه کلمات به صورت بزرگ نوشته می شود. به عنوان مثال:

```
var myTestValue = 0, mySecondTestValue = "hi";
```

- نشانه گذاری پاسکال: در این روش حرف اول متغیر و حرف اول بقیه کلمات به صورت بزرگ نوشته می شود. به عنوان مثال:

```
var MyTestValue = 0, MySecondTestValue = "hi";
```

- نشانه گذاری مجارتانی: در این روش از یک یا دنباله ای از پیشوندها قبل از نشانه گذاری پاسکال برای تعیین نوع یک متغیر استفاده می شود. به عنوان مثال کاراکتر i به معنی عدد صحیح و s به معنی رشته است. به عنوان مثال:

```
var iMyTestValue = 0, sMySecondTestValue = "hi";
```

جدول زیر لیستی از پیشوندهای موجود به منظور استفاده در نشانه گذاری مجارتانی را نشان می دهد. ما در این جزو از این پیشوندها برای نامگذاری متغیرها استفاده کرده ایم.

نمونه	پیشوند	نوع
aValues	a	آرایه
bFaound	b	بولین
fValue	f	عدد اعشاری
iValue	i	عدد صحیح
fnMethod	fn	تابع
oType	o	شی
sValue	s	رشته

یکی دیگر از امتیازات و یا شاید جذابیت های javascript (که در بسیاری از زبان های برنامه نویسی دیگر وجود ندارد) این است که لازم نیست متغیر ها را قبل از مقدار دهی، اعلان کنیم:

```
var sTest="hello";
sTest2=sTest + "world";
alert (sTest2); // hello world
```

در مثال فوق متغیر2 sTest2 قبل از مقدار دهی اعلان نشده است.

موقعی که مفسر به چنین متغیرهایی که بدون اعلان، مقدار دهی می شوند، برخورد می کند، یک متغیر سراسری به همان نام ایجاد کرده و مقداری را به آن اختصاص می دهد. با این وجود پیشنهاد می شود همیشه قبل از به کارگیری متغیرها آن ها را اعلان کنید.

^۱ Camel Notation

متغیرها و انواع داده ها

۱ کلمات کلیدی^۱

جاوا اسکریپت تعدادی از کلمات را به عنوان کلمات کلیدی می‌شناسد. این کلمات کلیدی معمولاً ابتدا یا انتهای دستورات را مشخص می‌کنند. کلمات کلیدی به عنوان کلمات رزرو شده شناخته می‌شوند و نمی‌توان از آن‌ها به عنوان نام متغیرها یا توابع استفاده نمود. در زیر لیست کاملی از این کلمات را مشاهده می‌کنید:

Break	else	new	var
Case	finally	return	void
Catch	for	switch	while
Continue	function	this	with
Default	if	throw	
Delete	in	try	
Do	instanceof	typeof	

اگر از یکی از کلمات فوق برای نامگذاری متغیرها یا توابع استفاده کنید با خطای Identifier expected روبرو خواهید شد.

۲ کلمات رزرو شده^۲

جاوا اسکریپت تعدادی از کلمات رزرو شده را نیز معرفی کرده است. کلمات رزرو شده نیز نمی‌توانند به عنوان نام متغیرها و توابع استفاده شوند. لیست کاملی از این کلمات را در زیر مشاهده می‌کنید:

Abstract	enum	int	short
Boolean	export	interface	static
Byte	extends	long	super
Char	final	native	synchronized
Class	float	package	throws
Const	goto	private	transient
Debugger	implements	protected	volatile
Double	import	public	

انواع داده‌های اصلی

در جاوا اسکریپت پنج نوع داده اصلی به شرح زیر وجود دارد:

undefined

null

boolean

number

string

از عملگر typeof برای تشخیص نوع یک مقدار استفاده می‌شود. این عملگر یک پارامتر که می‌تواند یک متغیر و یا یک مقدار باشد را دریافت کرده و نوع آن را بر می‌گرداند.

این عملگر یکی از پنج مقدار زیر را بر می‌گرداند:

undefined: اگر متغیر از نوع undefined است.

boolean: اگر متغیر از نوع boolean باشد.

¹ Keywords

² Reserved Words

آموزش کاربردی جاوا اسکریپت - احمد بادپی

- اگر متغیر از نوع Number باشد.
- اگر متغیر از String باشد.
- اگر متغیر یک ارجاع یا از نوع null باشد.

نوع داده Undefined

این نوع فقط شامل مقدار Undefined می شود. متغیری که اعلان شود ولی مقدار دهی اولیه نشود به صورت پیش فرض از نوع undefined خواهد بود.

```
var oTemp ;
alert (typeof oTemp) ; // outputs "Undefined"
```

به این نکته توجه داشته باشید که متغیری که اعلان می شود و مقدار نمی گیرد با متغیری که اصلاً اعلان هم نشده است کاملاً متفاوت است. هر چند که عملگر typeof بین این دو تفاوتی قائل نمی شود و برای هر دو متغیر، مقدار undefined را برمی گرداند، اگر چه فقط یکی از آن ها در مثال زیر (oTemp2) تعریف شده است.

```
var oTemp ;
alert (typeof oTemp) ; // outputs "undefined"
alert (typeof oTemp2) ; // outputs "undefined"
```

اگر از oTemp2 به وسیله‌ی هر عملگری به غیر از typeof استفاده کنید یک خطای خطا رخ خواهد داد:

```
//make sure this variable isn't defined
//var oTemp2;
//try outputting
alert(oTemp2 == undefined); //causes error
```

خروجی تابعی که در داخل بدنه خود مقداری را صراحتاً برنمی گرداند نیز مقدار undefined است:

```
function Testfunc () {
    // leave the function block
}
alert( TestFunc() == undefined ) ; //outputs "true"
```

نوع داده Null

دیگر نوع داده که فقط یک مقدار دارد، null است که مقدار ویژه null را می گیرد.

از نظر javascript نوع undefined یکی از مشتقات نوع null است و این دو معادل یکدیگرند:

```
alert(null == undefined) ; //outputs "true"
```

اگر چه این دو معادل یکدیگرند اما معانی متفاوتی دارند. تنها زمانی به یک متغیر نسبت داده می شود که آن متغیر اعلان شود ولی مقداردهی نشود. در حالی که یک متغیر زمانی از نوع null است که اشاره به شی ای داشته باشد که وجود ندارد.

متغیرها و انواع داده ها

نوع داده Boolean

نوع Boolean یکی از پر کاربردترین انواع داده در زبان های برنامه نویسی به شمار می رود و متغیری از این نوع فقط می تواند یکی از دو مقدار true یا false به عنوان مقدار بپذیرد. اگر چه بر خلاف زبان های برنامه نویسی متداول، در جاوا اسکریپت با 0 برابر نیست اما در صورت لزوم (و برای استفاده در عبارت های بولی) 0 به false تبدیل خواهد شد. به مثال های زیر توجه کنید:

```
var bFound = true;
var bLost = false;
```

نوع داده Number

این نوع نیز یکی از پر کاربردترین انواع است. از این نوع داده می توان برای نمایش اعداد صحیح ۸ بایتی و اعداد اعشاری ۱۶ بایتی استفاده کرد.

به عنوان مثال متغیر زیر از نوع صحیح است و مقدار اولیه‌ی ۵۵ را دارد:

```
var iNum = 55;
```

برای تعریف متغیرهای اعشاری به صورت زیر عمل می شود:

```
var fNum = 5.0;
```

نوع داده String

این نوع می تواند برای ذخیره صفر یا چندین کاراکتر به کار رود. هر کاراکتر در یک رشته موقعیتی دارد. موقعیت اولین کاراکتر صفر است.

برای تعریف یک متغیر رشته ای باید از (') یا (") استفاده کنیم، معمولاً برای تعریف یک کاراکتر از (') و برای تعریف یک رشته از (") استفاده می شود.

```
var sColor1 = "blue";
var sColor2 = 'blue';
```

تبدیل انواع

یکی از ویژگی های برجسته هر زبان برنامه نویسی قابلیت تبدیل انواع در آن است. جاوا اسکریپت نیز از این قاعده مستثنی نبوده و روش های ساده ای برای تبدیل انواع فراهم آورده است. اکثر انواع موجود در جاوا اسکریپت از متدهایی برای تبدیل به سایر انواع پشتیبانی کرده و توابعی سراسری برای تبدیل انواع به روش های پیچیده تر وجود دارد.

تبديل به رشته

یکی از جذابترین ویژگی هایی که جاوا اسکریپت در رابطه با انواع اصلی boolean ، string و numbers فراهم کرده است این است که آنها در اصل اشیای کاذب^۱ هستند، به این معنی که دارای خاصیت ها و متدهای مشترک و منحصر به فردی می باشند.

```
var sColor = "blue" ;
alert (sColor.length) ; //outputs "4"
```

سه نوع داده number ، boolean و string متدهی به نام toString() برای تبدیل به رشته دارند. این متده برای متغیرهای از نوع Boolean یکی از مقادیر رشته ای true و false را بسته به مقدار متغیر برمی گرداند:

```
var bFound = false;
alert(bFound.toString()) ; //outputs "false"
```

این متده برای متغیرهای از نوع number رشته ای حاوی آن عدد را برمی گرداند:

```
var iNum1 = 10;
var fNum2 = 10.0;
alert(iNum1.toString()) ; //outputs "10"
alert(fNum2.toString()) ; //outputs "10"
```

تبديل به عدد

جاوا اسکریپت دو متده برای تبدیل انواع غیر عددی به عددی فراهم کرده است:

- parseInt()
- parseFloat()

نکته: توجه کنید که حروف I و F باید به صورت حرف بزرگ نوشته شوند.

این متدها فقط بر روی رشته های حاوی عدد کار می کنند و بر روی بقیه انواع مقدار NaN را برمی گردانند. متده parseInt() از اولین کاراکتر رشته شروع می کند اگر عدد بود آن را برمی گرداند در غیر این صورت مقدار NaN را برمی گرداند. این روند تا آخرین کاراکتر ادامه پیدا می کند تا اینکه به کاراکتری غیر عددی برسد. به عنوان مثال این متده عبارت "123red" را به صورت 123 برمی گرداند.

```
var iNum1 = parseInt("1234blue") ; //returns 1234
var iNum3 = parseInt("22.5") ; //returns 22
var iNum4 = parseInt("blue") ; //returns NaN
```

متده parseFloat() نیز مثل parseInt() عمل کرده و از اولین کاراکتر شروع به جستجو می کند. البته در این متده اولین کاراکتر نقطه حساب نمی شود و آن را به همان صورت برمی گرداند.

¹ Pseudo-Objects

متغیرها و انواع داده ها

اگر دو کاراکتر نقطه در رشته وجود داشته باشند دومین نقطه به عنوان کاراکتر بی ارزش شناخته می شود و عملیات تبدیل متوقف می شود. مثال ها:

```
var fNum1 = parseFloat("1234blue");           //returns 1234.0
var fNum3 = parseFloat("22.5");                //returns 22.5
var fNum4 = parseFloat("22.34.5");             //returns 22.34
var fNum6 = parseFloat("blue");                //returns NaN
```

۱۲

۲

استفاده از Type Casting برای تبدیل انواع

در جاوااسکریپت امکان استفاده از روشی موسوم به Type Casting برای تبدیل انواع وجود دارد. سه نوع در جاوااسکریپت وجود دارد:

- Boolean()
- Number()
- String()

تابع () زمانی مقدار true را بر می گرداند که پارامتر دریافتی اش، رشته ای شامل حداقل یک کاراکتر، یک عدد غیر از صفر و یا یک شی باشد. مقدار false را نیز زمانی بر می گرداند که پارامتر دریافتی اش رشته ای تهی، عدد صفر یا یکی از مقادیر null و undefined باشد:

```
var b1 = Boolean("");                      //false - empty string
var b2 = Boolean("hi");                    //true - non-empty string
var b3 = Boolean(100);                     //true - non-zero number
var b4 = Boolean(null);                   //false - null
var b5 = Boolean(0);                      //false - zero
var b6 = Boolean(new Object());          //true - object
```

تابع () کاری شبیه Number() و parseFloat() را انجام می دهد اما تفاوت هایی هم دارد. اگر به یاد داشته باشید متدهای () و () آرگومان دریافتی را فقط تا اولین کاراکتر بی ارزش بر می گردانند. مثلاً رشته "4.5.6" به 4.5 تبدیل خواهد کرد. اما کاربرد متدهای Number() برای این رشته مقدار NaN را بر می گرداند زیرا این رشته در کل، از نظر متدهای Number()، امکان تبدیل به یک عدد را ندارد.

اگر رشته ای امکان تبدیل به یک عدد را داشته باشد متدهای Number() خود، برای استفاده از یکی از توابع parseInt() یا parseFloat() تصمیم می گیرد. در مثال زیر حاصل اجرای تابع () Number برای انواع داده ها را نشان می دهد:

Number(false)	0
Number(true)	1
Number(undefined)	NaN
Number(null)	0
Number("5.5")	5.5
Number("56")	56
Number("5.6.7")	NaN
Number(new Object())	NaN
Number(100)	100

ساده ترین تابع هم () است که همان چیزی را که می گیرد به عنوان رشته بر می گرداند:

```
var s1 = String(null); // "null"
```

فصل سه

جاوااسکریپت در مرورگرها

حال که تا حدودی با بسیاری از مفاهیم پایه جاوااسکریپت آشنا شدیم می خواهیم طریقه استفاده و قرار دادن آن ها در صفحه را بررسی کنیم. HTML برای استفاده از جاوااسکریپت در صفحات تگی به نام `<script>` را فراهم کرده که در ادامه با آن آشنا خواهیم شد.

جاوااسکریپت در مرورگرها

۱۸

۳

عموماً از این تگ در داخل تگ head صفحه استفاده می‌شود و می‌تواند یک ، دو یا سه صفت را بگیرد. صفت language که نوع زبان استفاده شده را تعیین می‌کند، صفت اختباری src که مکان یک فایل خارجی جاوااسکریپت را مشخص می‌کند و صفت type که نوع MIME TYPE یک فایل خارجی جاوااسکریپت را مشخص می‌کند و باید برابر عبارت javascript قرار داده شود. مقدار صفت language معمولاً برابر javascript یا یکی از نسخه‌های آن مثل text/javascript ۱.۳ تعیین می‌شود. (اگر از صفت javascript چشم پوشی شود ، مرورگرها آخرین نسخه موجود این زبان را در نظر می‌گیرند).

کدهای جاوااسکریپت در داخل تگ <script> نوشته می‌شوند اما در صورتی که هم‌زمان از صفت src نیز استفاده شود در این صورت معمولاً مرورگرها کدهای داخل تگ <script> را نادیده می‌گیرند. به مثال زیر دقت کنید:

```
<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript">
        var i = 0;
    </script>
    <script language="JavaScript"
src="../scripts/external.js"></script>
</head>
<body>
    <!-- body goes here -->
</body>
</html>
```

در این مثال هر دو نوع تعریف کدهای جاوااسکریپت در صفحه نشان داده شده است. تگ اسکریپت اول به صورت درون خطی^۱ به تعریف کدها پرداخته است و در تگ <script> دوم (به روش کدهای خارجی^۲) به یک فایل خارجی javascript اشاره شده است.

فایل‌های خارجی javascript

فایل‌های خارجی جاوااسکریپت فرمت بسیار ساده‌ای دارند. آن‌ها در واقع فایل‌های متین ساده حاوی کدهای جاوااسکریپت هستند. دقت کنید که در فایل‌های خارجی جاوااسکریپت از هیچ تگ script‌ی نباید استفاده شود. به عنوان مثال به تکه کد زیر دقت کنید:

¹ inline

² external

```

<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript">
        function sayHi() {
            alert("Hi");
        }
    </script>
</head>
<body>
    <!-- body goes here -->
</body>
</html>

```

می توانیم خود تابع sayhi() را در فایلی خارجی مثلا به نام external.js ذخیره کرده و آن را به صورت زیر در صفحه مورد نظر الحاق کنیم:

```

<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript" src="external.js"></script>
</head>
<body>
    <!-- body goes here -->
</body>
</html>

```

تفاوت های به کارگیری کدها به صورت درون خطی و خارجی
 چه موقع ما باید از روش درون خطی و چه موقع باید از روش خارجی برای به کارگیری کدهای جاوا اسکریپت استفاده کنیم؟ هر چند که قانون سفت و سختی برای استفاده از هر یک از روش های فوق وجود ندارد اما به دلایل زیر استفاده از روش درون خطی مناسب به نظر نمی رسد:

- امنیت:** هر کسی می تواند با باز کردن source صفحه شما، کدها را ببیند و چه بسا به حفره های امنیتی آن پی برد و از آن ها سوءاستفاده کند.
- ذخیره شدن در حافظه مرورگرها:** یکی از مزیت های استفاده از روش خارجی این است که فایل های خارجی جاوا اسکریپت پس از اولین بارگذاری در حافظه نهان مرورگر^۱ ذخیره شده و در دفعات بعد، از حافظه فراخوانی و استفاده خواهند شد.
- نگه داری کدها^۲:** چنانچه شما بخواهید از یک کد در چندین صفحه وب استفاده کنید مطمئنا استفاده از روش اول موجب افزایش کد نویسی و در نتیجه حجم صفحه خواهد شد اما می توانیم از روش دوم برای چندین فایل استفاده کنیم.

¹ cache

² Code Maintenance

مکان قرار دادن تگ <script> در صفحه

معمولًا کدها و توابع تعریفی بوسیله جاوااسکریپت باید در قسمت head صفحه قرار گیرد تا به موقع بارگذاری شده و برای استفاده در قسمت body صفحه آماده استفاده و فراخوانی باشند. معمولًا کدهایی که در آن ها، تابع از قبل تعریف شده صدا زده می شوند در قسمت body قرار می گیرند.

قراردادن تگ <script> در داخل قسمت body موجب اجراشدن کدهای داخل آن به مخصوص بارگذاری قسمتی از صفحه در مرورگر خواهد شد. برای مثال به تکه کد زیر دقت کنید:

```
<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript">
        function sayHi() {
            alert("Hi");
        }
    </script>
</head>
<body>
    <script language="JavaScript">
        sayHi();
    </script>
    <p>This is the first text the user will see.</p>
</body>
</html>
```

در کد فوق تابع sayHi() دقیقا قبل از نمایش هر گونه متنی در صفحه اجرا خواهد شد. به این معنی که پنجره alert قبل از متن This is the first text the user will see اجرا خواهد شد. این روش برای فراخوانی متدهای جاوااسکریپت اصلا پیشنهاد نمی شود و می بایست به جای آن از کنترلگرهای رویداد^۱ استفاده کرد. مثلا:

```
<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript">
        function sayHi() {
            alert("Hi");
        }
    </script>
</head>
<body>
    <input type="button" value="Call Function" onclick="sayHi()" />
</body>
</html>
```

¹ Event Handler

آموزش کاربردی جاوااسکریپت - احمد بادپی

در اینجا دکمه‌ای با استفاده از تگ input ایجاد شده است که در صورت کلیک بر روی آن تابع sayHi() فراخوانی می‌شود. صفت onclick در اینجا کنترلگر رویدادی را که باید به رویداد رخ داده پاسخ دهد، تعیین می‌کند.

نکته اینکه ازآنجایی که کدهای جاوااسکریپت به محض بارگذاری اجرا هم می‌شوند ممکن است در این صورت توابعی که از قبل وجود ندارند فراخوانی شوند که در این صورت یک خطا رخ خواهد داد. در مثال قبل با عوض کردن جای تگ‌های <script> یک خطا رخ خواهد داد:

```
<html>
<head>
    <title>Title of Page</title>
</head>
<body>
    <script language="JavaScript">
        sayHi();
    </script>
    <p>This is the first text the user will see.</p>
    <script language="JavaScript">
        function sayHi() {
            alert("Hi");
        }
    </script>
</body>
</html>
```

در صورت اجرای کد فوق یک خطا رخ خواهد داد زیرا تابع قبل از اینکه تعریف شود فراخوانی شده است. به دلیل بارگذاری کدها از بالا به پایین، تابع sayHi() که تگ <script> دوم ایجاد نشده است در دسترس نخواهد بود.

مخفي کردن اسکریپت‌ها از مرورگرهای قدیمی

هنوز کاربران زیادی وجود دارند که از مرورگرهایی استفاده می‌کنند که با جاوااسکریپت ناسازگار هستند. از آن مهمتر، تعدادی از کاربران گزینه پشتیبانی از جاوااسکریپت را در مرورگر خود غیر فعال کرده‌اند. از آنجایی که مرورگرهای قدیمی تگ <script> را نمی‌شناسند و نمی‌توانند آن را تفسیر نمایند در اکثر موارد به جای تفسیر اسکریپت، متن آن را در صفحه نمایش می‌دهند.

برای جلوگیری از این مشکل، می‌توان اسکریپت‌ها را در داخل توضیحات HTML قرار داد. با این کار مرورگرهای قدیمی آن را نادیده گرفته و نمایش نخواهند داد. از طرف دیگر مرورگرهای جدید می‌دانند که دستورات توضیحی که در بین دستورات آغازین و پایانی <script> منظور شده اند تنها برای مخفی کردن اسکریپت از دید مرورگرهای قدیمی تر است و لذا به تفسیر اسکریپت ادامه می‌دهند.

همان طور که می‌دانید برای نوشتن یک توضیح در سند HTML کافی است علامت --!> را در ابتداء و علامت <-- را در انتهای آن قرار دهید. به مثال زیر دقت کنید:

جاوااسکریپت در مرورگرها

۲۲

۳

```
<script language="JavaScript"><!-- hide from older browsers
    function sayHi() {
        alert("Hi");
    }
//-->
</script>
```

به دو کاراکتر / که در انتهای دستور فوق آمده دقت کنید. این دو / برای جلوگیری از این که مفسر مرورگرهای سازگار با جوااسکریپت، عبارت <-- را به عنوان یک دستور جوااسکریپت تفسیر نکند استفاده شده است. عدم استفاده از این دو / موجب ایجاد یک خطأ خواهد شد.

روش مخفی کردن اسکریپت ها از مرورگرهای ناسازگار با جوااسکریپت را فرا گرفتیم. اما چگونه می توان برای کاربرانی که از این مرورگرها استفاده می کنند نیز مطلب جایگزینی نمایش داد؟ برای این کار باید از تگی به نام <noscript> استفاده کنیم. مرورگرهای سازگار هر چیزی را که بین دستورات آغازین و پایانی <noscript> قرار داشته باشد ، نادیده می گیرند. از طرف دیگر مرورگرهای قدیمی این دستور را نمی شناسند و بنابراین آنرا نادیده گرفته و به سراغ دستورات بعدی (که توسط این دستور احاطه شده اند) می روند. به مثال زیر توجه کنید:

```
<html>
<head>
    <title>Title of Page</title>
    <script language="JavaScript">
        function sayHi() {
            alert("Hi");
        }
    </script>
</head>
<body>
    <script language="JavaScript">
        sayHi();
    </script>
    <noscript>
        <p>Your browser doesn't support JavaScript. If it did
support JavaScript, you would see this message: Hi!</p>
    </noscript>
        <p>This is the first text the user will see if JavaScript is
enabled. If
JavaScript is disabled this is the second text the user will see.</p>
    </body>
</html>
```

خطایابی

زمانی که مرورگرها قادر به اجرای دستور خاصی از کدهای جوااسکریپت نباشند، پیغامی مبنی بر رخداد یک خطأ را نمایش می دهند. ما برای رفع خطاهای این پیغام ها نیاز داریم اما متاسفانه در ک بسیاری از این پیغام ها در مرورگرهای مختلف دشوار است. اغلب برای مشاهده پیغام ها می بایست به console جوااسکریپت در مرورگرها مراجعه کنید:

آموزش کاربردی جاوااسکریپت - احمد بادپی

در مرورگر FireFox می توان از مسیر زیر به کنسول جاوااسکریپت دسترسی داشت

FireFox Menu > Web Developer > Error Console

این کار با استفاده از میانبر **ctrl+shift+j** نیز امکان پذیر است.



در مرورگر opera این کار از مسیر زیر

Tools > Advanced > Error Console

یا استفاده از میانبر **ctrl+shift+o** انجام می شود.



در مرورگر Google Chrome نیز می توان از مسیر

> Tools > Javascript Console

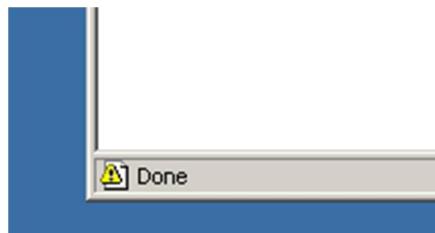
یا میانبر **j** **ctrl+shift+j** به این بخش دسترسی داشت.



در مرورگر Internet Explorer برای مشاهده پیغام مربوط به خطاهای رخ داده در صفحه می بایست بر روی آیکن

زردرنگ موجود در گوشه پایین سمت چپ مرورگر کلیک کرد تا پنجره ای حاوی متن و شماره خطی از برنامه که خطا

در آن رخ داده است باز شود.

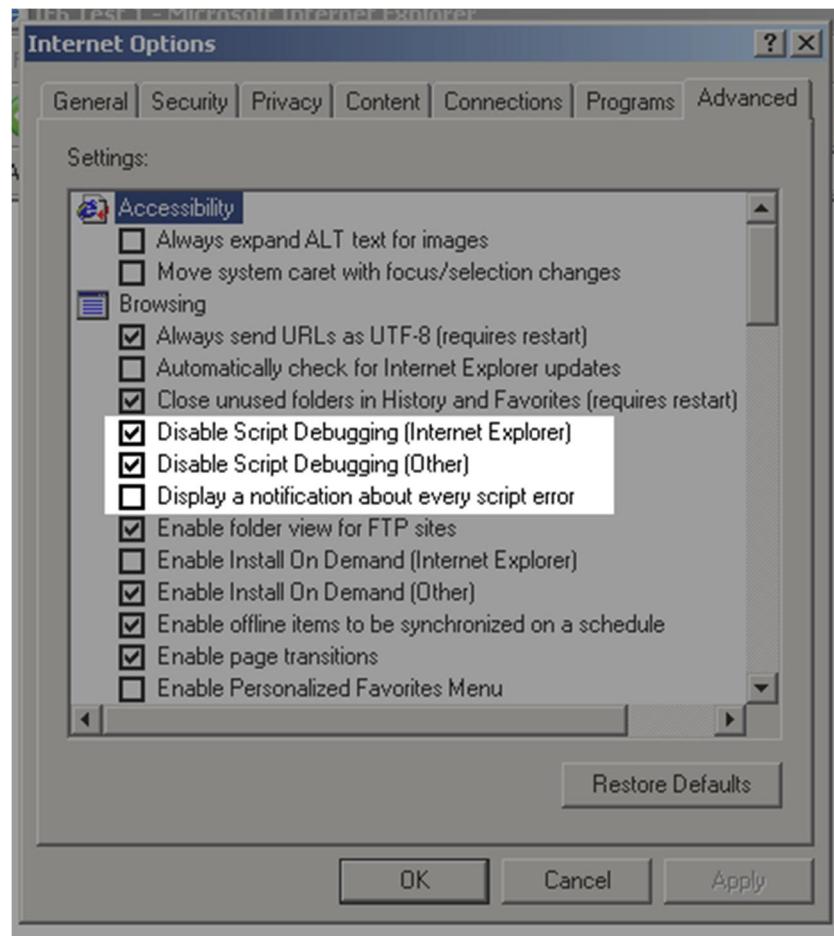


البته قبل از هر چیز می بایست از فعال بودن خطایابی و نمایش پیغام ها در این مرورگر اطمینان حاصل کنید. برای فعال سازی

این امکان در مسیر Tools > Internet Options > Advanced (Internet Explorer) گزینه **Display a notification about every script error** را تیک

بزنید:

جاوا اسکریپت در مرورگرها



پس از انجام موارد فوق، با هر بار رخداد یک خطأ، پنجره‌ای به صورت زیر نمایش داده خواهد شد:



فصل چهار

کار با آرایه ها در جاوا اسکریپت

آرایه ها در همه زبان های برنامه نویسی جز مهمترین ساختمان داده ها به شمار می روند. نقش آرایه ها در جاوا اسکریپت نیز برای ایجاد برنامه های انعطاف پذیر نیز انکار ناپذیر است. در این فصل ابتدا به بررسی روش های ساخت آرایه ها و ویژگی های اصلی آن پرداخته و در ادامه در مورد نحوه دستکاری آن ها همچون اضافه، حذف، انتخاب و مترب سازی عناصر آرایه پرداخته و به روش های تبدیل آرایه به رشته و بالعکس خواهیم پرداخت.

کار با آرایه ها در جاوا اسکریپت

۲۶

۴

ایجاد آرایه ها با استفاده از کلاس Array

در جاوا اسکریپت برخلاف جاوا، کلاس درون ساختی به نام `Array` وجود دارد که از آن برای ایجاد آرایه ها (که البته به عنوان یک شی در نظر گرفته می شوند) استفاده می شود. برای ایجاد یک شی از نوع آرایه از دستورات زیر استفاده می کنیم:

```
var aValues = new Array();
```

اگر از قبل تعداد عناصر آرایه مورد نظرتان را بدانید می توانید به شکل زیر عمل کنید:

```
var aValues = new Array(20);
```

برای مقداردهی خانه های آرایه به شکل زیر عمل می کنیم:

```
var aColors = new Array();
aColors[0] = "red";
aColors[1] = "green";
aColors[2] = "blue";
```

در آرایه بالا با هر بار اضافه کردن عنصر جدید به صورت خودکار به تعداد خانه های آن افزوده می شود.

اگر شما از قبل مقادیری که قرار است در آرایه قرار بگیرند را بدانید می توانید به صورت عمل کنید:

```
var aColors = new Array("red", "green", "blue");
```

برای دسترسی به عناصر آرایه به صورت زیر عمل می شود:

```
alert(aColors[1]); //outputs "green"
```

بدست آوردن طول آرایه

برای مشخص کردن تعداد عناصر موجود در آرایه از خاصیتی به نام `length`. استفاده می شود. این مقدار همیشه یک واحد بیشتر از موقعیت آخرین خانه آرایه است.

اگر در آرایه قبلی که سه عنصر داشت به یکباره موقعیت مثلا ۲۵ را پر کنیم طول آرایه چه خواهد بود؟

در این صورت جاوا اسکریپت خانه های از ۳ تا ۲۴ را با مقدار `null` پر خواهد کرد و طول آرایه هم برابر ۲۶ خواهد بود:

```
var aColors = new Array("red", "green", "blue");
alert(aColors.length); //outputs "3"
aColors[25] = "purple";
aColors.length; //outputs "26"
```

راه دیگر ایجاد یک آرایه استفاده از براکت ها `([])` و علامت ، بین هر عنصر از آرایه است به صورت زیر:

```
var aColors = ["red", "green", "blue"];
alert(aColors.length); //outputs "3"
aColors[25] = "purple";
alert(aColors.length); //outputs "26"
```



نکته: آرایه‌ها در جاوا اسکریپت می‌توانند حداقل 4294967295 خانه داشته باشند!

تبديل آرایه به رشته

آرایه‌ها از سه متدهای خاص برای خروجی عناصر خود به صورت رشته‌ای که با کاما از هم جدا شده اند پشتیبانی می‌کند:

```
var aColors = ["red", "green", "blue"];
alert(aColors.toString()); //outputs "red,green,blue"
alert(aColors.valueOf()); //outputs "red,green,blue"
alert(aColors.toLocaleString()); //outputs "red,green,blue"
```

می‌بینید که حاصل اجرای هر سه کد فوق یکسان است.

از متدهای به نام `join()` برای الحاق عناصر یک آرایه که البته به وسیله یک جداکننده^۱ از هم جدا شده اند استفاده می‌شود. این تابع یک آرگومان دارد که در واقع رشته‌ای است که بین هر یک از عناصر وجود دارد. به مثال‌های زیر دقت کنید:

```
var aColors = ["red", "green", "blue"];
alert(aColors.join(",")); //outputs "red,green,blue"
alert(aColors.join("-spring-")); //outputs "red-spring-green-spring-blue"
alert(aColors.join("] [")); //outputs "red] [green] [blue"
```

تبديل رشته به آرایه

سوالی که در اینجا پیش می‌آید این است که آیا اشیایی از نوع `string` را هم می‌توان به طریق مشابه به آرایه تبدیل کرد؟ جواب مثبت است!

شی `string` متدهایی دارد که یک آرگومان می‌گیرد که همانطور که حدس زدید جداکننده‌ی رشته برای تبدیل به آرایه را مشخص می‌کند.

حال اگر شما رشته‌ای دارید که با ، از هم جدا شده است می‌توانید به صورت زیر عمل کنید:

```
var sColors = "red,green,blue";
var aColors = sColors.split(",");
```

اگر هیچ جداکننده‌ای مشخص نشود، این تابع آرایه‌ای را بر می‌گرداند که هر عنصر آن شامل یکی از کاراکترهای رشته‌ی مورد نظر است. برای مثال:

```
var sColors = "green";
var aColors = sColors.split("");
alert(aColors.toString()); //outputs "g,r,e,e,n"
```

¹ separator

کار با آرایه ها در جاوا اسکریپت

۲۸

۴

اضافه کردن مقادیر جدید به آرایه ها

آرایه ها از متدهایی به نام concat() پشتیبانی می کنند. این تابع چندین آرگومان می گیرد و به آرایه جاری اضافه می کند و حاصل آن یک آرایه جدید خواهد بود. به مثالهای زیر دقت کنید:

```
var aColors = ["red", "green", "blue"];
var aColors2 = aColors.concat("yellow", "purple");
alert(aColors2.toString()); //outputs "red,green,blue,yellow,purple"
alert(aColors.toString()); //outputs "red,green,blue"
```

برگرداندن عناصر خاصی از آرایه

از تابعی به نام slice(). برای برگرداندن عناصر خاصی از آرایه استفاده می شود. این تابع دو آرگومان می گیرد و از خانه آرگومان اول تا قبل از آرگومان دوم را به آرایه جدیدی تبدیل می کند. اگر فقط آرگومان اول منظور گردد این تابع عناصر از آن آرگومان تا انتهای آرایه را بر می گرداند. به مثالهای زیر دقت کنید:

```
var aColors = ["red", "green", "blue", "yellow", "purple"];
var aColors2 = arr.slice(1);
var aColors3 = arr.slice(1, 4);
alert(aColors2.toString()); //outputs "green,blue,yellow,purple"
alert(aColors3.toString()); //outputs "green,blue,yellow"
```

در حالت کلی arr.slice(n,m) عناصر از خانه n تا m-1 را بر می گرداند.

تبدیل آرایه ها به پسته و صف

یکی از جذابترین ویژگی های آرایه در جاوا اسکریپت امکان تبدیل کردن آنها به دیگر ساختمان داده های رایج همچون stack و queue است.

اگر آرایه ای را به عنوان stack در نظر بگیریم می توانیم به راحتی از تابع push() و pop() برای اضافه و حذف عناصر از انتهای آرایه استفاده کنیم.

تابع push(). امکان اضافه کردن چندین عنصر به آرایه و تابع pop(). امکان حذف آخرین عنصر آرایه و برگرداندن آن به عنوان مقدار بازگشتی تابع را فراهم می کند. البته تابع pop(). عنصری را که بر می گرداند از آرایه حذف می کند. به مثالهای زیر دقت کنید:

```

var stack = new Array();
stack.push("red");
stack.push("green");
stack.push("yellow");
alert(stack.toString()); //outputs "red,green,yellow"
var vItem = stack.pop();
alert(vItem); //outputs "yellow"
alert(stack.toString()); //outputs "red,green"

```

۲۹

۴

جاوا اسکریپت توابع دیگری برای دستکاری عناصر ابتدایی آرایه فراهم می کند. تابعی به نام `shift()`. برای حذف و برگرداندن عنصر اول آرایه استفاده می شود. از طرف دیگر تابعی به نام `unshift()`. یک عنصر را به ابتدای آرایه اضافه کرده و بقیه عناصر را یک موقعیت به جلو جابجا می کند:

```

var aColors = ["red", "green", "yellow"];
var vItem = aColors.shift();
alert(aColors.toString()); //outputs "green,yellow"
alert(vItem); //outputs "red"
aColors.unshift("black");
alert(aColors.toString()); //outputs "black,green,yellow"

```

در شکل زیر نحوه عملکرد توابع فوق بر روی یک آرایه عددی نمایش داده شده است:



مرتب سازی آرایه ها

از دو تابع برای مرتب سازی^۱ عناصر آرایه استفاده می شود. تابعی به نام `reverse()` برای مرتب سازی عکس آرایه استفاده می شود. مثال:

```

var aColors = ["red", "green", "blue"];
aColors.reverse();
alert(aColors.toString()); //outputs "blue,green,red"

```

از طرف دیگر تابعی به نام `sort()`. عناصر آرایه را به صورت صعودی بر حسب مقادیرشان مرتب می کند. در این صورت عناصر آرایه بر حسب کدهای کاراکتری شان مرتب می شوند. مثال:

```

var aColors = ["red", "green", "blue", "yellow"];
aColors.sort();
alert(aColors.toString()); //outputs "blue,green,red,yellow"

```

^۱ ordering

کار با آرایه ها در جاوا اسکریپت

در صورتی که عناصر آرایه اعداد باشند نتیجه کمی عجیب و غریب است:

```
var aColors = [3, 32, 2, 5]
aColors.sort();
alert(aColors.toString()); //outputs "2,3,32,5"
```

۳۰

۴

همانطور که اشاره شد متدها sort() به صورت پیش فرض عناصر آرایه را به صورت الفبایی (بر حسب کدهای کاراکتری آنها) و به صورت صعودی مرتب می‌کند. چنانچه بخواهیم آرایه ای از اعداد را به صورت صعودی مرتب کنیم می‌بایست از یک تابع مقایسه‌ای که در قالب یک آرگومان برای این متدها فرستاده می‌شود استفاده کنیم. خود این تابع مقایسه‌ای همیشه دو آرگومان (به عنوان مثال a,b) گرفته، آنها را با هم مقایسه کرده و آرایه‌ی اصلی براساس مقادیر بازگشتی این تابع انجام می‌شود.

مقدار بازگشتی تابع مقایسه‌ای به یکی از سه صورت زیر است:

چنانچه می‌بایست b (آرگومان دوم) قبل از a (آرگومان اول) قرار بگیرد مقداری مثبت را برمی‌گرداند.

چنانچه می‌بایست مکان a و b تعییری نکند مقدار صفر را برمی‌گرداند.

چنانچه می‌بایست a قبل از b قرار بگیرد مقداری منفی را برمی‌گرداند.

به عنوان مثال در صورتی که تابع زیر را به عنوان آرگومان برای متدها sort() بفرستیم آرایه به صورت صعودی مرتب خواهد شد:

```
function Compare(a,b) {
    If(a>b) {
        Return 1;
    }else if (a<b) {
        Return -1;
    }else{
        Return 0;
    }
}
```

دقت داشته باشید که بدنه تابع فوق را می‌توان به شکل زیر کوتاهتر و ساده‌تر ترکرد ضمن اینکه همان عملکرد را خواهد داشت:

```
function Compare(a,b) {
    Return a-b;
}
```

برای مرتب سازی آرایه به صورت عددی نزولی کافی است جای 1 و -1 در تابع مقایسه‌ای را عوض کنید.

حذف و درج در میانه های آرایه

یکی از پیچیده ترین توابعی که در کار با آرایه ها مورد استفاده قرار می گیرد تابعی به نام `(splice)` است. هدف اصلی این تابع درج یکسری عناصر در میانه های آرایه است.

راه های گوناگونی برای این استفاده از این متدها در رابطه با آرایه و عمل درج پیشنهاد شده است:

عمل حذف: از این متدها برای حذف عناصری از میانه های آرایه می توان استفاده کرد. برای این کار از دو پارامتر برای این تابع استفاده می شود: موقعیت اولین عنصر و تعداد عناصر مورد نظر برای حذف. برای مثال `arr.splice(0, 2)` دو عنصر اول آرایه ای به نام `arr` را حذف می کند.

درج بدون حذف: شما می توانید از این تابع برای درج عناصر جدید با استفاده از سه پارامتر استفاده کنید: موقعیت شروع ، تعداد عناصر حذفی و عناصر جدید برای درج. شما می توانید هر تعداد پارامتر برای درج را به این تابع بدهید. برای مثال `arr.splice(2, 0, "red", "green")` عناصر `red` و `green` را از خانه دوم در آرایه درج می کند.

درج عنصر همراه با حذف: شما می توانید از این تابع برای درج عناصر جدید در یک موقعیت مشخص همزمان با عمل حذف و استفاده از سه پارامتر استفاده کنید: موقعیت شروع حذف ، تعداد عناصر حذفی و عناصر جدید درجی. به عنوان مثال `arr.splice(2, 1, "red", "green")` یک عنصر را از موقعیت ۲ حذف کرده و مقادیر `red` و `green` را از همان موقعیت (2) درج می کند.

فصل پنجم

کاربارشته ها در جاوا اسکریپت

این فصل به بررسی رشته ها در جاوا اسکریپت می پردازد. در این فصل ابتدا به روش های ایجاد رشته ها در جاوا اسکریپت پرداخته و سپس به توضیح روش های دستکاری آن ها همچون جدا کردن زیر رشته ها، الحاق و مقایسه رشته ها و... خواهیم پرداخت.

کار با رشته ها در جاوا اسکریپت

۳۴

۵

ایجاد اشیا رشته ای (رشته) با استفاده از کلاس String

از این کلاس برای ایجاد اشیا رشته ای (به اختصار رشته ها) استفاده می شود. دستور زیر متغیری حاوی رشته را تولید می کند:

```
var oStringObject = new String("hello world");
```

اشیای از نوع string خاصیتی به نام length. دارند که تعداد کاراکتر های رشته را برابر می گردانند. این شی از چندین متد نیز پشتیبانی می کند که در ادامه شرح خواهیم داد:

بدست آوردن کاراکتر موجود در یک موقعیت خاص

مثلا: عددی را به عنوان آرگومان می گیرد و کاراکتر نظیر آن در رشته اصلی را برمی گرداند.

```
var oStringObject = new String("hello world");
alert(oStringObject.charAt(1)); //outputs "e"
```

گرچنانچه می خواهید به جای خود کاراکتر کد کاراکتری آن را بدست آورید از متد charCodeAt(). استفاده کنید:

```
var oStringObject = new String("hello world");
alert(oStringObject.charCodeAt(1)); //outputs "101"
```

این دستور مقدار 101 که معادل کد کاراکتری حرف e است را برابر می گرداند.

الحق دو رشته

متد دیگر concat() است که برای الحق دو رشته استفاده می شود. برای مثال:

```
var oStringObject = new String("hello ");
var sResult = oStringObject.concat("world");
alert(sResult); //outputs "hello world"
alert(oStringObject); //outputs "hello "
```

به جای استفاده از متد concat() می توان از عملگر + نیز برای الحق دو رشته استفاده کرد.

عملگر + برای الحق رشته ها

از عملگر + در جاوا اسکریپت هم برای جمع اعداد و هر برای الحق رشته ها استفاده می شود. رفتار این عملگر براساس نوع عملوندها به صورت زیر تعیین می شود.

- اگر هر دو عملوند عددی باشند حاصل جمع آن ها محاسبه خواهد شد.
- اگر هر دو عملوند رشته ای باشند حاصل الحق رشته دوم به رشته اول خواهد بود.
- اگر یکی از عملوندها عددی و دیگری رشته ای باشد، عملوند عددی به رشته تبدیل شده و حاصل الحق آن دو خواهد بود.

به مثال زیر دقت کنید:

```
var result1 = 5+5;
alert(result1); // output 10
```

```
var result2 = 5+"5";
alert(result2); // output "55"
```

```
var result3 = "5"+5;
alert(result3); // output "55"
```

بدست آوردن موقعیت یک کاراکتر خاص در رشته

برای تشخیص اینکه یک کاراکتر خاص در یک رشته هست یا نه می توان از متدهای `.lastIndexOf()` و `.indexOf()` استفاده می شود.

هر دو این متدها موقعیت زیر رشته ای در رشته دیگر را برمی گرداند که البته در صورت پیدا نشدن مقدار 1- را برمی گردانند. تنها تفاوت این دو تابع در این است که `.indexOf()` جستجو را از ابتدای رشته (موقعیت 0) شروع می کند ولی دیگری جستجو را از انتهای رشته شروع می کند. برای مثال:

```
var oStringObject = new String("hello world");
alert(oStringObject.indexOf("o")); //outputs "4"
alert(oStringObject.lastIndexOf("o")); //outputs "7"
```

در صورتی که حرف O در عبارت بالا فقط یکبار تکرار می شد هر دو این متدها فقط یک مقدار را برمی گردانند. در متدهای `.lastIndexOf()` و `.indexOf()` می توان از یک آرگومان اختیاری دیگر نیز به منظور تعیین کاراکتری که عمل جستجو باید از آن شروع شود، استفاده کرد. به عنوان مثال در کد زیر عمل جستجو از چهارمین کاراکتر شروع می شود و در نتیجه مقدار 5 (یعنی موقعیت دومین a) برگردانده می شود:

```
Var example = "I am a javascript hacker";
Alert(example.indexOf('a', 3)); // output 5
```

مقایسه رشته ها

متدهای `localeCompare()` و `strcmp()` در زبان C++ است. این متدها برای مقایسه رشته ها مورد استفاده قرار می گیرد. این تابع یک آرگومان رشته ای می پذیرد و یکی از سه مقدار زیر را برمی گرداند:

۱. اگر شی رشته ای کوچکتر از آرگومان باشد 1- را برمی گرداند.
۲. اگر برابر باشند 0 را برمی گردانند.
۳. اگر شی رشته ای بزرگتر باشد مقدار 1 را برمی گرداند.

کار با رشته ها در جاوا اسکریپت

مثال ها:

۳۶

۵

```
var oStringObject = new String("yellow");
alert(oStringObject.localeCompare("brick")); //outputs "1"
alert(oStringObject.localeCompare("yellow")); //outputs "0"
alert(oStringObject.localeCompare ("zoo")); //outputs "-1"
```

جدا کردن زیر رشته ای از رشته دیگر

دوتابع برای جدا کردن زیر رشته ها از رشته اصلی وجود دارد: `slice()` و `substring()`.

هر دو این متدها یک یا دو آرگومان را می پذیرند که آرگومان اول محل شروع و آرگومان دوم محل پایان را تعیین می کند.
(البته خودآرگومان دوم جز زیر رشته نخواهد بود.)

اگر آرگومان دوم نادیده گرفته شود طول رشته درنظر گرفته خواهد شد.

چیزی که این دو متدهای گرداند زیر رشته حاصل است:

```
var oStringObject = new String("hello world");
alert(oStringObject.slice(3)); //outputs "lo world"
alert(oStringObject.substring(3)); //outputs "lo world"
alert(oStringObject.slice(3, 7)); //outputs "lo w"
alert(oStringObject.substring(3,7)); //outputs "lo w"
```

سوالی که در اینجا پیش می آید این است که چرا دقیقاً این دوتابع بکار را انجام می دهند؟ در حقیقت تفاوت آنها در کار با آرگومان های منفی است.

برای متدهای `slice()` و `substring()` آرگومان منفی با طول رشته جمع شده و حاصل آن به عنوان آرگومان اصلی درنظر گرفته می شود. در حالی که برای تابع `slice()` مقادیر منفی به عنوان صفر درنظر گرفته می شود.(درواقع نادیده گرفته می شوند.)

مثال ها:

```
var oStringObject= new String("hello world");
alert(oStringObject.slice(-3)); //outputs "rld"
alert(oStringObject.substring(-3)); //outputs "hello world"
alert(oStringObject.slice(3, -4)); //outputs "lo w"
alert(oStringObject.substring(3,-4)); //outputs "hel"
```

در خط دوم از کدبالا چون آرگومان منفی است طول رشته با `-3` جمع می شود که حاصل `8` است درواقع دستور زیر اجرا می شود:

```
oStringObject.slice(8);
```

که از خانه هشتم رشته تا انتهای آرایه را برمی گرداند. اما در خط سوم آرگومان منفی صفر درنظر گرفته می شود. یعنی:

```
oStringObject.substring(0);
```

در خط چهارم آرگومان دوم با طول رشته جمع شده و حاصل آن یعنی 8 به عنوان آرگومان دوم در نظر گرفته می شود. یعنی:

```
oStringObject.slice(3,8);
```

و در خط پنجم حاصل به صورت زیر محاسبه می شود:

```
oStringObject.substring(3,0);
```

 نکته: ترتیب آرگومان ها در متدهای slice() و substring(). اهمیتی ندارد. به عنوان مثال حاصل subString(6,3). با subString(3,6) تفاوتی نخواهد داشت. به هر حال آرگومان کوچکتر به عنوان حد ابتدایی و پارامتر بزرگتر به عنوان حد انتهایی در نظر گرفته می شود.

 نکته: در صورتی که آرگومان های ارسالی به متدهای slice() و substring(). یکسان باشند، مقدار null برگردانده می شود.

جاوا اسکریپت از متدهایی به نام substr() استفاده می کند. این متدها از آرگومان می گیرد که اولی موقعیت شروع و دومی طول زیر رشته ای که می خواهیم از رشته اصلی جدا کنیم خواهد بود. در صورتی که آرگومان دوم ذکر نشود عمل جدا کردن تا انتهای رشته خواهد بود. به مثال زیر توجه کنید:

```
var myString      =      "javascript";
var mySubStr     =      myString.substr(0,4);
alert(mySubStr);    //      output "java"
```

toLowerCase() و toUpperCase()

از توابعی همچون toUpperCase() و toLowerCase(). برای تبدیل حروف رشته به حروف بزرگ یا کوچک استفاده می شود که کار آن ها از روی اسمشار کاملا مشخص است:

```
var oStringObject= new String("Hello World");
alert(oStringObject.toLocaleUpperCase()); //outputs "HELLO WORLD"
alert(oStringObject.toUpperCase()); //outputs "HELLO WORLD"
alert(oStringObject.toLocaleLowerCase()); //outputs "hello world"
alert(oStringObject.toLowerCase()); //outputs "hello world"
```

فصل شش

اشیای درونی^۱ (پیش ساخته)

جاوا اسکریپت شامل تعدادی شی از پیش ساخته است که توسعه دهنده‌گان می‌توانند از آن‌ها در برنامه‌های خود استفاده کنند. در واقع کلاس‌هایی برای این اشیا نداریم و لازم نیست شی ای از روی آن‌ها ساخته شود.

اشیای درونی (پیش ساخته)

۱۰۶

Math شی

یکی از اشیای از پیش ساخته شده جاوااسکریپت است که برای انجام محاسبات عددی و عملیات مربوط به ریاضیات استفاده می شود. این شی شامل یکسری خاصیت و متدهای است که انجام محاسبات را آسان می کند.

.max() و .min()

از این توابع برای پیدا کردن کوچکترین و بزرگترین مقادیر از بین چند عدد استفاده می شود. این متدهای هر تعداد پارامتر را می توانند پذیرند:

```
var iMax = Math.max(3, 54, 32, 16);
alert(iMax); //outputs "54"
var iMin = Math.min(3, 54, 32, 16);
alert(iMin); //outputs "3"
```

این توابع برای جلوگیری از نوشتن برنامه های اضافی برای پیدا کردن `min` و `max` اعداد می تواند استفاده شود. یکی از متدهای `.abs()` است که قدر مطلق اعداد گرفته شده را برابر می گرداند.

گروهی دیگر از متدهای که برای گرد کردن اعداد اعشاری به صحیح مورد استفاده قرار می گیرند. این توابع شامل `.ceil()` و `.round()` و `.floor()` هستند.

تابع `.round()`: این تابع عدد گرفته شده را به عدد صحیح بالاتر گرد می کند اگر قسمت اعشاری از نصف بیشتر یا مساوی باشد و در غیر این صورت آن را به عدد صحیح پایین تر گرد می کند.

تابع `.ceil()`: این تابع بدون در نظر گرفتن قسمت اعشاری آن را به کوچکترین عدد صحیح بعدی گرد می کند.

تابع `.floor()`: این تابع بدون در نظر گرفتن قسمت اعشاری آن را به بزرگترین عدد صحیح قبلی گرد می کند.

به مثال های زیر توجه کنید:

```
alert(Math.ceil(25.5)); //outputs "26"
alert(Math.round(25.5)); //outputs "26"
alert(Math.floor(25.5)); //outputs "25"
```

گروه دیگری از متدهای کار با مقادیر توانی وجود دارد:

`.log()`: برای محاسبه لگاریتم طبیعی عدد گرفته شده به کار می رود.

`.pow()`: برای محاسبه توان یک عدد به کار می رود که دو آرگومان می گیرد:

```
var iNum = Math.pow(2, 10);
```

`.sqrt()`: جذر یک عدد را حساب می کند:

```
var iNum = Math.sqrt(4);
alert(iNum); //outputs "2"
```

شی `Math` شامل متدهای زیر نیز می باشد:

`acos(x)` , `asin(x)` , `atan(x)` , `atan2(x, y)` , `cos(x)` , `sin(x)` , `tan(x)`

یکی دیگر از متدهای مربوط به شی `Math` که کاربرد زیادی هم دارد `random()` است که برای تولید اعداد تصادفی بین ۰ و ۱ (البته نه خود ۰ و ۱) مورد استفاده قرار می‌گیرد.

البته برای تولید اعداد تصادفی در یک محدوده خاص از فرمول زیر استفاده می‌شود:

```
number = Math.floor(Math.random() * total_number_of_choices +
first_possible_value)
```

به عنوان مثال برای ایجاد مقادیر تصادفی بین ۱ و ۱۰ به صورت زیر عمل می‌شود:

```
var iNum = Math.floor(Math.random() * 10 + 1);
```

بهترین راه برای ایجاد مقادیر تصادفی استفاده از یک تابع است که به صورت زیر نوشته می‌شود:

```
function selectFrom(iFirstValue, iLastValue) {
    var iChoices = iLastValue - iFirstValue + 1;
    return Math.floor(Math.random() * iChoices + iFirstValue);
}
//select from between 2 and 10
var iNum = selectFrom(2, 10);
```

استفاده از این تابع برای انتخاب یک عنصر تصادفی از آرایه بسیار آسان است. برای مثال:

```
var aColors = ["red", "green", "blue", "yellow", "black", "purple",
"brown"];
var sColor = aColors[selectFrom(0, aColors.length-1)];
```

در اینجا آرگومان دوم تابع ، طول آرایه منهای ۱ است که در واقع موقعیت آخرین عنصر می‌باشد.

دیگر توابع مفید

از توابعی همچون `(URI)` `encodeURI()` و `encodeURIComponent()` برای کدگذاری آدرس های اینترنتی (URI ها) استفاده می‌شود. در حالت کلی و صحیح یک آدرس نباید شامل کarakترهای نامعتبر همچون space باشد. این تابع به شما در تبدیل کردن و encode کردن آدرس های اینترنتی نادرست و بی ارزش برای اینکه مرورگرها آنها را بفهمند استفاده می‌شود.

متدهای `encodeURI()` و `encodeURIComponent()` معمولاً برای آدرس های کامل (به عنوان مثال

<http://learningjquery.ir/illegal value.htm>) مورد استفاده قرار می‌گیرد در حالی که دیگری برای قسمتی از آدرس ها همچون illegal value.htm مورد استفاده قرار می‌گیرد.

تفاوت اصلی بین این دو تابع این است که تابع اول کarakتر های خاصی که به عنوان جزئی از آدرس هستند همچون (:) ، / ، ؟ و ... را encode نمی‌کند درحالی که تابع دوم تمام کarakتر های غیر استاندارد را encode خواهد کرد. برای مثال:

```
var sUri = "http://www.wrox.com/illegal value.htm#start";
alert(encodeURI(sUri));
alert(encodeURIComponent(sUri));
```

اشیای درونی (پیش ساخته)

۱۲

۶

حاصل اجرای کد بالا به صورت زیر خواهد شد:

```
http://www.wrox.com/illegal%20value.htm#start
http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.htm%23start
```

طبعیتاً دو تابع برای decode کردن آدرس های اینترنتی استفاده می شود همچون:

- decodeURI()
- decodeURIComponent()

به عنوان مثال:

```
var sUri = "http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.htm%23start";
alert(decodeURI(sUri));
alert(decodeURIComponent(sUri));
```

حاصل اجرای این کد به صورت زیر خواهد بود:

```
http%3A%2F%2Fwww.wrox.com%2Fillegal value.htm%23start
http://www.wrox.com/illegal value.htm#start
```

آخرین تابعی که به نظر قدرتمند می آید eval() است. این تابع که شبیه به مفسر جاواسکریپت کار می کند آرگومانی از نوع رشته می گیرد که در واقع یک برنامه به زبان جاواسکریپت است و این تابع آن را همانند سایر برنامه ها اجرا می کند. برای مثال:

```
eval("alert('hi'));
```

این تکه کد در حقیقت معادل دستور زیر است:

```
alert("hi");
```

موقعی که مفسر جاواسکریپت به این تابع می رسد آرگومان آن را به عنوان یک دستور خیلی ساده تفسیر کرده و اجرا می کند. این به این معنی است که شما می توانید از داخل آرگومان های این تابع به تمام متغیرهای خارج آن دسترسی داشته و از آن ها استفاده کنید :

```
var msg = "hello world";
eval("alert(msg)");
```

همچنین شما می توانید آرگومان تابع eval() را یک تابع تعریف کرده و سپس آن را خارج از تابع eval() صدا بزنید. برای مثال:

```
eval("function sayHi() { alert('hi'); }");
sayHi();
```

کار با تاریخ و زمان در جاوا اسکریپت

یکی از قابلیت های جالب جاوا اسکریپت، جمع آوری اطلاعات از سیستم کاربر و نمایش آنها در صفحات وب است. همانطور که می دانید HTML به تنهایی قادر به انجام چنین کاری نیست اما با کمک زبانهای دیگر تحت وب مانند Javascript، می تواند تا حدودی این مشکل را برطرف کند. شی هایی در جاوا اسکریپت وجود دارند که توسط متدهای مختلف، اطلاعات مورد نیاز را از سیستم گرفته و در اختیار کاربران قرار می دهند. یکی از این اشیاء Date می باشد که به کمک آن می توانیم تاریخ و زمان سیستم را هنگام اجرای کد دریافت کنیم، سپس آنرا نمایش دهیم و یا اینکه در یک متغیر ذخیره کنیم تا در صورت لزوم از آن بهره گیریم. برای ایجاد شی ای از این نوع می توان به شکل زیر عمل کرد:

```
var d = new Date();
```

شی Date() تعداد هزارم ثانیه های گذشته از ساعت 12:00:00 01/01/1970 روز 1970-01-01 تا زمان و تاریخ کنونی را در خود نگه داری می کند. این شی دارای متدهای نام valueOf() می باشد که این مقدار را برابر می گرداند. به عنوان مثال به کد زیر نگاه کنید:

```
<script type="text/javascript">
    var d=new Date();
    document.write(d.valueOf());
</script>
```

حاصل اجرای کد فوق می تواند عددی به شکل زیر باشد:

1269938333117

این شی دارای متدهایی است که از آن ها برای بدست آوردن جزئیات بیشتری از تاریخ و زمان استفاده نمود. بعضی از این متدها و خواص را در جدول زیر مشاهده می کنید:

اشیای درونی (پیش ساخته)

۱۴۶

توضیحات	نام متدها
روزی از ماه را برابر می گرداند که می تواند مقداری از ۱ تا ۳۱ باشد.	getDate()
ماهی از سال را برابر می گرداند که مقداری از ۰ تا ۱۱ می باشد	getMonth()
سال را در قالب عدد برابر می گرداند	getFullYear()
ساعتی از روز را برابر می گرداند که می تواند مقداری از ۰ تا ۲۳ باشد	getHours()
دقیقه را برابر می گرداند که مقداری از ۰ تا ۵۹ است.	getMinutes()
ثانیه را برابر می گرداند که مقداری از ۰ تا ۵۹ است	getSeconds()
روزی از هفته را برابر می گرداند که می تواند مقداری از ۰ تا ۶ باشد. (۶ به معنی Sunday)	getDay()
تعداد میلی ثانیه های گذشته از تاریخ 1970/1/1 را برابر می گرداند	getTime()
تعداد میلی ثانیه های گذشته از تاریخ 1970/1/1 را برابر می گرداند	valueOf()
رشته ای حاوی اطلاعاتی همچون مخفف نام روز جاری ، ماه جاری و ... را برابر می گرداند.	toString()

علاوه بر متدهای فوق ، شی Date از متدهای به نام getTimezoneOffset() که اختلاف بین زمان محلی و زمان واحد جهانی را بحسب دقیقه برابر می گرداند نیز پشتیانی می کند. به عنوان مثال این متدد مقدار ۲۱۰ را برای وقت محلی ایران بر می گرداند. (که همان اختلاف ۳:۳۰ دقیقه ای ساعت تهران نسبت به زمان واحد جهانی است.)

فصل هفت

BOM؛ مدل شے‌گرای مرورگر

BOM به عنوان یکی از اجزای اصلی و ابتدایی تشکیل دهنده جاوا اسکریپت نقش مهمی در تعامل کاربران با بخش‌های گوناگون مرورگرها همچون بخش نمایش سند، فریم‌ها، پنجره‌ها، تاریخچه^۱، مشخصات سیستم عامل و مرورگر و ... ایفا می‌کند. ما در این بخش ابتدا با BOM آشنا شده و سپس به بررسی بخش‌های تشکیل دهنده آن و هر یک از خصوصیات آن‌ها خواهیم پرداخت.

¹ history

BOM؛ مدل شی گرای مرورگر

چیست؟ BOM

آشنایی با مفهوم BOM به منظور یادگیری و استفاده کارآمد از جاوااسکریپت بسیار اهمیت دارد. BOM اشیایی که با پنجره‌ی مرورگر ارتباط و تعامل مستقیم دارند را فراهم می‌کند. مانند شکل زیر:

۱۵

۷

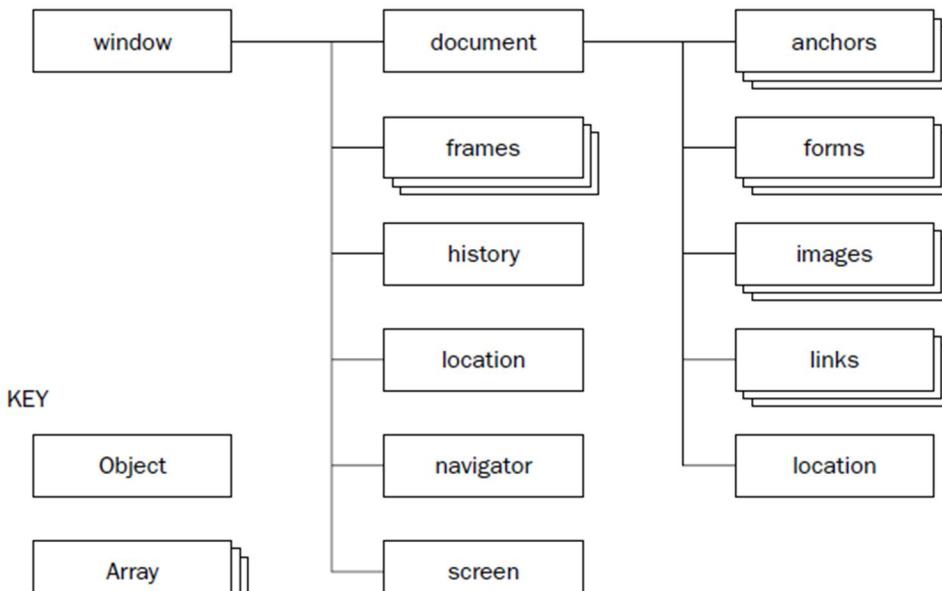


Figure 5-3

اکنون به بررسی هر یک از اجزای تشکیل دهنده BOM و خصوصیات و متدهای مربوط به آن‌ها می‌پردازیم:

window شی

شی window تمامی پنجره‌های مرورگر را شامل می‌شود اما نه لزوماً محتوایی که در آن نمایش داده می‌شود. از این شی برای جابجایی، تغییر اندازه و دیگر اثرات بر روی پنجره‌ها استفاده می‌کنیم.

دستکاری پنجره‌ها

شش متدهای دستکاری پنجره مرورگر برای شی window وجود دارد:

۱. moveBy(dx,dy)

پنجره را نسبت به موقعیت کنونی به اندازه x در جهت افقی و به اندازه y در جهت عمودی جابجا می‌کند. عدد های منفی هم برای x,y مجازند.

۲. moveTo(x,y)

گوشه بالای چپ مرورگر را به موقعیت y,x می‌برد. مقادیر منفی نیز مجاز هستند.

۳. resizeBy(w,h)

عرض پنجره مرورگر را به اندازه w و ارتفاع آنرا به اندازه h نسبت به size کنونی تغییر می‌دهد. مقادیر منفی نیز مجازند.

۴. resizeTo(w,h)

عرض مرورگر را به w و ارتفاع آن را به h تغییر می‌دهد. مقادیر منفی مجاز نیستند.

آموزش کاربردی جاوا اسکریپت - احمد بادپی

۵. scrollBy(dx,dy)

اسکرول افقی را به اندازه x و اسکرول عمودی را به اندازه y جابجا می کند.

۶. scrollTo(x,y)

این متده مختصات جدید اسکرول های افقی و عمودی را به ترتیب با استفاده از آرگومان های x و y مشخص می کند.

مثال ها:

`window.moveTo(10, 20)`

پنجره را نسبت به مکان فعلی 10px 10 پیکسل به سمت راست و 20px به سمت پایین جابجا می کند.

`window.resizeTo(150, 300)`

عرض پنجره را به 150px و ارتفاع آن را به 300px تغییر می دهد.

`window.resizeBy(150, 0)`

فقط 150px به عرض کنونی پنجره اضافه می کند.

`window.moveTo(0, 0)`

پنجره را به گوشه بالا و سمت چپ صفحه نمایش هدایت می کند.

پیمايش و باز کردن پنجره های جدید

برای باز کردن پنجره های جدید با استفاده از جاوا اسکریپت از متده open() استفاده می شود که چهار آرگومان می گیرد:

۱. آدرس صفحه

۲. نام صفحه

۳. رشته ای از ویژگی های

۴. و یک مقدار Boolean

عموماً فقط از سه آرگومان اول استفاده می شود. اگر پنجره ای از قبل با نامی که برای آرگومان دوم انتخاب کرده اید وجود داشته باشد صفحه در آن پنجره باز خواهد شد، در غیر این صورت در پنجره ای جدید باز می شود. اگر آرگومان سوم تعیین نشود پنجره با تنظیمات پنجره اصلی مرورگر باز خواهد شد.

ویژگی های آرگومان سوم مشخص می کند که پنجره ای جدید چه خصوصیاتی داشته باشد که در زیر بیان می کنیم:

خصوصیات با = مقدار دهی می شود و با ، از هم جدا می شود.

برخی از خصوصیات مجاز قابل استفاده عبارتند از:

left: فاصله از چپ top: فاصله از بالا width: عرض پنجره height: ارتفاع پنجره resizable: آیا پنجره قابل تغییر اندازه باشد یا خیر

BOM: مدل شی گرای مرورگر

۱۸

۷

- (Yes/No): مشخص می کند آیا پنجره نوار اسکرول داشته باشد یا خیر
- (Yes/No): آیا شامل نوار ابزار باشد
- (Yes/No): آیا نوار وضعیت داشته باشد
- (Yes/No): آیا نوار آدرس داشته باشد
- در رشتہ ای از خصوصیات نباید هیچ فضای خالی وجود داشته باشد.

متدهای window.open(). شی ای از نوع window را بر می گرداند که تمام متدها و خاصیت هایی که شی window دارد را دارد است. برای بستن پنجره از متدهای close() استفاده می شود. این متدهای فقط می توانند پنجره ای که توسط جاوا اسکریپت باز شده است را مستقیماً ببندند نه پنجره ای اصلی.

پنجره های System Dialog

شی window چندین تابع برای نمایش پیغام و گرفتن جواب از کاربران را دارد.

(alert()): این تابع یک آرگومان از نوع متن می گیرد و آن را در قالب یک پنجره کوچک که یک دکمه ok دارد نمایش می دهد:

```
<script type="text/javascript" >
    alert('Hello world');
</script>
```

از این پنجره معمولاً برای نمایش یک پیغام به صورت هشدار استفاده می شود.



(confirm()): این تابع هم مانند تابع بالا است. تنها تفاوت این دو وجود یک دکمه cancel در پنجره باز شونده است.

```
<script type="text/javascript" >
    confirm('Are you sure ? ');
</script>
```

در صورتی که کاربر دکمه Ok را بزند مقدار True و در صورت زدن دکمه cancel مقدار False را بر می گرداند.



(prompt()): پنجره ای این متدهای چهار قسمت دارد. دکمه های Ok، cancel، یک متن و یک کادر متنی برای وارد کردن یک رشتہ توسط کاربر.

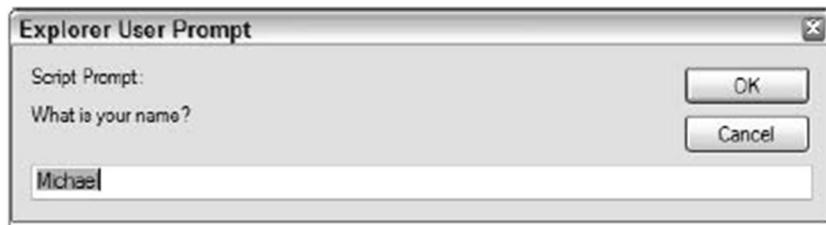
آموزش کاربردی جاوا اسکریپت - احمد بادپی

این متد دو آرگومان می گیرد:

۱. عنوان سوال یا متنی که به کاربر نشان داده می شود.
۲. مقدار پیش فرض برای کادر متنی

```
<script type="text/javascript" >
    Prompt('what is your name', 'Michael');
</script>
```

در صورتی که کاربر دکمه Ok را بزند تابع مقدار وارد شده در کادر متنی را برمی گرداند و در صورت زدن دکمهی Cancel مقدار Null را برمی گرداند.

**statusbar**

این قسمت پنجره فرآیند بارگذاری و پایان بارگذاری را به کاربر نشان می دهد. هر چند که می توانیم از دو خاصیت به نام های status و defaultStatus برای تغییر آن استفاده کنیم. همانطور که حدس زدید از خاصیت Status برای تغییر متن برای چند لحظه استفاده می شود در حالی که از defaultstatus برای تغییر Statusbar تا زمانی که کاربر در صفحه هست استفاده می شود.

برای تغییر لحظه ای نوار وضعیت مثلاً وقتی کاربر، ماوس را روی یک لینک قرار می دهد می توان از کد زیر استفاده نمود:

```
<a href="books.htm" onmouseover="window.status='Information on Wrox
books.'"
">Books</a>
```

اجرای مکرر کدها از طریق متدهای Intervals و Timeouts

از این دو تابع برای اجرای یک تکه کد بعد از بازه زمانی خاصی استفاده می شود.

setIntervals: کد گرفته شده را پس از عددی بر حسب میلی ثانیه اجرا می کند. در حالی که Intervals کد گرفته شده را مکرراً بعد از مدتی بر حسب میلی ثانیه چندین بار تکرار می کند. این متد دو آرگومان می گیرد:

۱. کدی که باید اجرا شود.
۲. مدت زمانی که باید بعد از آن کد اجرا شود.

آرگومان اولی هم می تواند به صورت یک رشته از کدها و هم نام یک تابع باشد. هر سه کد زیر بعد از یک ثانیه یک پنجره هشدار را نمایش می دهند:

BOM: مدل شی گرای مرورگر

```
<script type="text/javascript" >
    setTimeout("alert('Hello world!')", 1000);
</script>
```



Y

```
<script type="text/javascript" >
    setTimeout(function() { alert("Hello world!"); }, 1000);
</script>
```

```
<script type="text/javascript" >

    function sayHelloWorld() {
        alert("Hello world!");
    }
    setTimeout(sayHelloWorld, 1000);
</script>
```

برای جلوگیری از اجرای تابع setTimeout(). از متده استفاده می شود:

```
<script type="text/javascript" >
    var iTimeoutId = setTimeout("alert('Hello world!')", 1000);
    clearTimeout(iTimeoutId);
</script>
```

setIntervals: مانند تابع قبلی است جز اینکه کد گرفته شده را بعد از گذشت بازه‌ی زمانی مشخص تکرار می‌کند.

برای جلوگیری از اجرای این متده، از تابعی به نام clearInterval() استفاده می‌شود:

```
setInterval("alert('Hello world!') ", 1000);
-----
setInterval(function() { alert("Hello world!"); }, 1000);
-----
function sayHelloWorld() {
    alert("Hello world!");
}
setInterval(sayHelloWorld, 1000);
```

شی history

ممکن است بخواهیم به تاریخچه‌ی مرورگر دسترسی داشته باشیم. البته هیچ راهی برای دسترسی به آدرس صفحات که در وجود دارند، نیست. برای این کار از متدها و خاصیت‌های شی history window مربوط به شی استفاده می‌کنیم: go() متد. فقط یک پارامتر می‌گیرد: تعداد صفحاتی که باید به جلو یا به عقب پیمایش شوند. اگر عدد منفی باشد به صفحات قبل و اگر عدد مثبت باشد به صفحات جلو می‌رویم. برای مثال جهت رفتن به یک صفحه عقب از کد زیر استفاده می‌کنیم:

آموزش کاربردی جاوا اسکریپت - احمد بادپی

```
window.history.go(-1);
```

و برای رفتن به جلو:

```
window.history.go(+1);
```

همیچنین می توانیم از متدهای back() و forward() استفاده کنیم.

```
//go back one
history.back();
//go forward one
history.forward();
```

همچنین از خاصیت lenght برای تعداد صفحات موجود در history استفاده کنیم:

```
alert("There are currently " + history.length + " pages in history.");
```

document شی

این شی که تنها شی مشترک بین مدل های شی گرای BOM و DOM است دارای خصوصیاتی می باشد.
یکی از خاصیت های این شی URL است که برای تنظیم و دسترسی به آدرس کنونی صفحه استفاده می شود.

```
document.URL = "http://www.learningjquery.ir/";
```

شی document از خاصیتی به نام referrer. برای به دست آوردن آدرس صفحه ای که کاربر از آن به صفحه کنونی آمده است استفاده می کند.

این شی همچنین خاصیتی به نام title. دارد که از آن برای بدست آوردن و حتی تغییر عنوان صفحه استفاده می کند. می توان از این خاصیت خواندنی/نوشتگی برای تغییر عنوان صفحه به صورت پویا نیز استفاده کرد. به عنوان مثال دستور زیر عنوان صفحه جاری را به New Title Page تغییر می دهد:

```
Document.title = "New Title Page";
```

همچنین این شی دارای یکسری خصوصیات مجموعه ای برای دسترسی به انواع عناصر داخل صفحه می بازد.

برخی از خاصیت ها به شرح زیر است:

توضیحات	مجموعه
دسترسی به لینک های صفحه	anchors
دسترسی به تمامی عناصر embed صفحه	embeds
دسترسی به تمامی فرم های صفحه	forms
دسترسی به تمامی عناصر عکس صفحه	images
دسترسی به تمامی لینک های صفحه	links

BOM مدل شی گرای مرورگر

هر مجموعه می تواند بوسیله‌ی عدد یا نام index گذاری شوند. به این معنی که شما می توانید به صورت زیر به اولین عنصر عکس صفحه دسترسی داشته باشید:

```
Document.images[0];
Or
Document.images['image-name'] ;
```

۵۲

۷

با این روش ما می توانیم به آدرس آن‌ها هم دسترسی داشته باشیم، به صورت زیر:

```
document.images[0].src
```

از دیگر متدهای این شی می توان به () writeln() و write() برای چاپ یک متن اشاره کرد.

شی location

یکی دیگر از شی‌ها برای دسترسی به آدرس صفحه جاری، location است. ما توسط خاصیت location.href می توانیم برای تنظیم یا بدست آوردن URL استفاده کنیم:

```
location.href= "http://www.learningjquery.ir/";
```

متدهای assign() هم همین کار را می‌کند:

```
Location.assign("http:// www.learningjquery.ir")
```

از متدهای reload() برای بارگزاری مجدد صفحه استفاده می‌شود. ما می‌توانیم تعیین کنیم که بارگزاری مجدد از روی Cache یا Server باشد. این کار با یکی از آرگومان‌های false برای بارگزاری مجدد از Catch و true برای بارگزاری مجدد از Server استفاده می‌شود. چنانچه آرگومانی ذکر نشود به صورت پیش‌فرض false در نظر گرفته خواهد شد.

شی navigator

این شی ایکی از اشیای قدیمی مدل شی گرای BOM به شمار می‌رود. از این شی برای دسترسی و بدست آوردن اطلاعاتی در مورد نوع و نسخه مرورگر استفاده می‌شود. بعضی از خاصیت‌های آن به شرح زیر است:

آموزش کاربردی جاوا اسکریپت - احمد بادپی

خاصیت ها	توضیحات
appcodeName	رشته ای حاوی کد رشته ای مرورگر
appName	نام عمومی مرورگر
appMinotVersion	اطلاعات اضافی مرورگر
appVersion	نسخه مرورگر
browserLanguage	نوع زبان مرورگر یا سیستم عامل
cookieEnabled	مشخص می کند میکند آیا کوکی ها فعال هستند یا خیر
cpuClass	کلاس cpu را مشخص میکند
javaEnabled	فعال بودن جاوا
Language	زبان مرورگر را مشخص میکند
mimeType	آرایه ای از mimetype های ثبت شده در مرورگر
Platform	نوع platform که کامپیوتر کاربر بر روی آن قرار دارد را مشخص می کند.
userAgent	رشته ای حاوی اطلاعاتی درمورد نوع و نسخه مرورگر و سیستم عامل را نگه داری می کند که می توان از آن برای بازیابی مشخصات دقیق سیستم عامل و مرورگر کار استفاده کرد. قدار این خاصیت همیشه همراه با هر درخواست توسط مرورگرها فرستاده می شود.

جدول زیر مقدار خاصیت navigator.userAgent. شی را در چهار مرورگر مشهور و سیستم عامل ویندوز XP نمایش می

دهد:

Google Chrome 16	Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.77 Safari/535.7
Opera 11	Opera/9.80 (Windows NT 5.1; U; en) Presto/2.7.62 Version/11.00
Firefox 9.0	Mozilla/5.0 (Windows NT 5.1; rv:9.0) Gecko/20100101 Firefox/9.0
Internet Explorer 7.0	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.2;.NET CLR 2.0.50727;.NET CLR 3.0.4506.2152;.NET CLR 3.5.30729)

BOM؛ مدل شی گرای مرورگر

شی screen

از این شی برای دسترسی به اطلاعات مربوطه به صفحه نمایش کاربر استفاده می‌شود. این شی شامل خواص زیر است:

توضیحات	خاصیت
ارتفاع قابل دسترس از ویندوز	availHeight
عرض قابل دسترس از ویندوز	availWidth
تعداد بیت‌ها برای نمایش رنگ‌ها	colorDepth
ارتفاع صفحه	height
عرض صفحه	width

از دو خاصیت اول می‌توان برای بدست آوردن سایز جدید پنجره استفاده نمود. به طور مثال برای fullscreen کردن صفحه نمایش می‌توان از کد زیر استفاده نمود:

```
Window.moveTo(0,0);
Window.resizeTo(screen.availWidth,screen.availHeight);
```

فصل هشت

DOM؛ مدل شے‌گرای سند

در این فصل به بررسی DOM یکی دیگر از اجزای مهم تشکیل دهنده جاوااسکریپت می‌پردازیم. این DOM است که امکان دسترسی و دستکاری عناصر موجود در صفحه و قابلیت اضافه، حذف و جابجایی آن‌ها در جای جای صفحه را فراهم می‌آورد. در سرآغاز این فصل ابتدا به بررسی و تشریح DOM پرداخته و سپس روش‌هایی که برای دستکاری عناصر موجود در صفحه را فراهم کرده توضیح خواهیم داد.

مدل شی گرای سند DOM

چیست؟ DOM

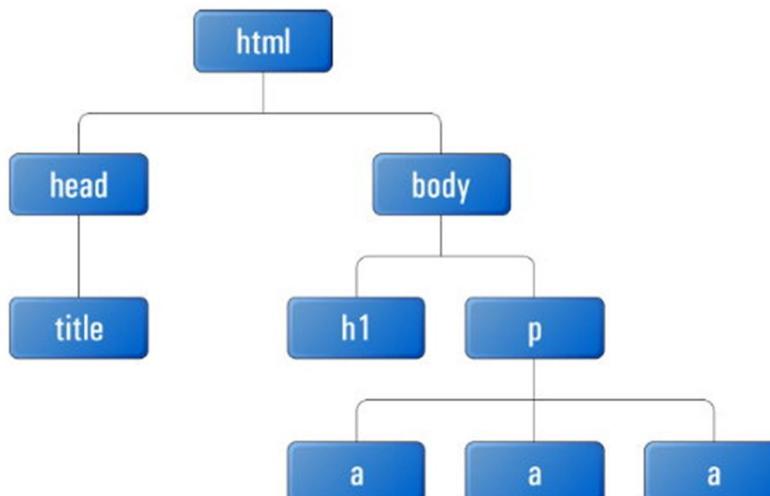
DOM به توسعه دهنده‌گان وب امکان دسترسی و دستکاری عناصر یک صفحه HTML را می‌دهد. این مدل عناصر موجود در یک صفحه HTML را به صورت درختی از گره‌ها ترسیم می‌کند. به تکه کد زیر دقت کنید:

```
<html>
<head>
    <title>DOMinating JavaScript</title>
</head>
<body>
    <h1>DOMinating JavaScript</h1>
    <p>
        If you need some help with your JavaScript, you might like
        to read articles from
        <a href="http://www.danwebb.net/" rel="external">DanWebb</a>
        <a href="http://www.quirksmode.org/" rel="external">PPK</a>
        and
        <a href="http://adactio.com/" rel="external">Jeremy
        Keith</a>
    </p>
</body>
</html>
```

۵۶

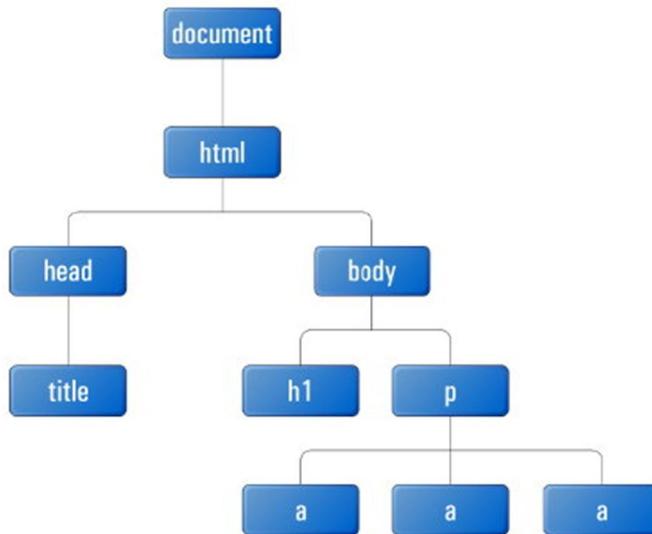
۸

این کد را می‌توان در قالب درخت زیر نمایش داد:



همانطور که می‌بینید می‌توان هر یک از عناصر موجود در صفحه را در قالب یک گره^۱ نمایش داده می‌شود. اما همیشه در DOM گرهی ویژه به نام document وجود دارد که در بالاترین سطح درخت قرار گرفته و سایر گره‌ها را شامل می‌شود. با این فرض درخت فوق به شکل زیر تبدیل خواهد شد:

^۱ node



در درخت بالا هر مستطیل به عنوان یک گره محسوب می شود. گره ها انواع مختلفی دارند که بعضی از آن ها به شرح زیر است:

: بالاترین گرهی که همه گره های دیگر به آن متصل هستند(فرزند آن هستند). به این نوع گره ، گره سند^۱ گفته می شود.

: گرهی که شامل یک عنصر از صفحه باشد. این گره شامل یک تگ آغازی و یک تگ پایانی مانند <tag></tag> یا </tag> است. این نوع گره تنها نوعی است که می تواند شامل فرزندان از انواع دیگر باشد. به این گره ها ، گره عنصری^۲ گفته می شود.

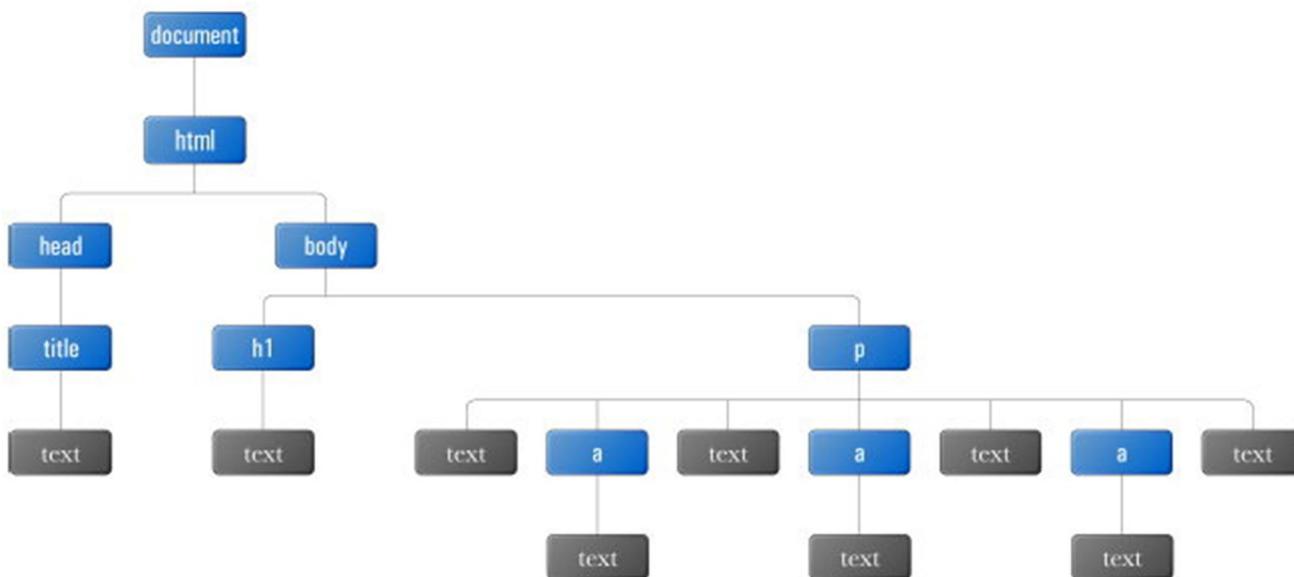
: این نوع گره ها به متن موجود در داخل یک تگ آغازی و تگ پایانی اشاره دارند. این نوع گره ها هم نمی توانند فرزند داشته باشند. به این نوع گره ها ، گره متنی^۳ می گویند. اگر گره های متنی را هم به مثالی که بررسی کردیم اضافه کنیم درخت ما به شکل زیر تبدیل خواهد شد:

¹ Document Node

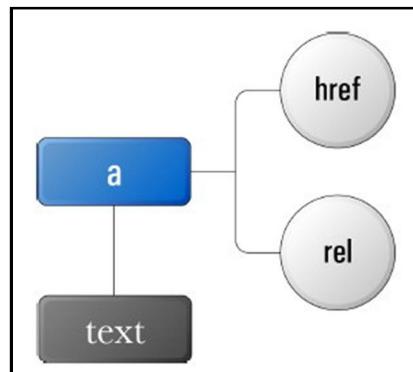
² Element Node

³ Text Node

مدل شی گرای سند DOM



: گره ای که به یک صفت از یک عنصر اشاره می کند و قادر فرزند می باشد. به این نوع گره ها ، گره صفتی^۱ گفته می شود. در درخت DOM معمولاً این گره ها را به صورت دایره ای و متصل به گره های عنصری نمایش می دهند. به عنوان مثال هر یک از عناصر لینکی که در مثال بالا مشاهده می شود دارای صفت های href و rel هستند که می توان آن ها را به صورت زیر نمایش داد:



: به گره های توضیحی اشاره می کند و قادر فرزند است. (در واقع به تگ comment صفحه اشاره می کند.) غالباً گرهی اصلی به عنوان راس این درخت وجود دارد که همان document است. گره ها از نظر جاوا اسکریپت به عنوان یک شی در نظر گرفته می شوند که این اشیا می توانند خصیت ها و متدهایی داشته باشند. بعضی از آن ها به شرح زیر هستند:

¹ Attribute Node

توضیحات

نوع / نوع بازگشتی

خاصیت / متند

نام گره را برابر می‌گرداند. این خاصیت بستگی به نوع گره دارد.	String	nodeName
مقدار گره را برابر می‌گرداند. این خاصیت بستگی به نوع گره دارد.	String	nodeValue
یکی از انواع گره را برابر می‌گرداند.	Number	nodeType
اشاره به شی document که گره جزئی از آن است دارد	Document	ownerDocument
اشاره به اولین گره از لیست گره‌ها دارد.	Node	firstChild
اشاره به آخرین گره از لیست گره‌ها دارد.	Node	lastChild
لیست (آرایه) ای از تمام گره‌های داخل یک گره	NodeList	childNodes
اشاره به گره همزاد (برادر) قبلی دارد. اگر همزاد قبلی وجود نداشت باشد مقدار null را برابر می‌گردد.	Node	previousSibling
اشاره به گره همزاد (برادر) بعدی دارد. اگر همزاد بعدی وجود نداشت باشد مقدار null را برابر می‌گردد.	Node	nextSibling
در صورتی که آرایه childNodes دارای یک یا بیشتر از یک عضو (گره) باشد True را برابر می‌گردد.	Boolean	hasChildNodes()
آرگومان node را به انتهای آرایه childNodes اضافه می‌کند.	Node	appendChild(node)
آرگومان node را از انتهای آرایه childNodes حذف می‌کند.	Node	removeChild(node)
در آرایه oldnode، childNodes را با newnode جایجا می‌کند.	Node	replaceChild (newnode, oldnode)
در آرایه newnode، childNodes را قبل از refnode قرار می‌دهد.	Node	insertBefore (newnode, refnode)

DOM مدل شی گرای سند

استفاده از DOM

دسترسی به گره ها

تکه کد زیر را در نظر بگیرید:

۶۰

۸

```
<html>
<head>
  <title>DOM Example</title>
</head>
<body>
  <p>Hello World!</p>
  <p>Isn't this exciting?</p>
  <p>You're learning to use the DOM!</p>
</body>
</html>
```

ولا برای دسترسی به عنصر HTML می توان از `document.documentElement` که یکی از خصیت های شی است استفاده کنیم. به صورت زیر:

```
var oHtml = document.documentElement;
```

حال می توانیم با استفاده از این متغیر به عناصر head و body به صورت زیر دسترسی داشته باشیم:

```
var oHead = oHtml.firstChild;
var oBody = oHtml.lastChild;
```

راه دیگر به صورت زیر است:

```
var oHead = oHtml.childNodes[0];
var oBody = oHtml.childNodes[1];
```

برای بدست آوردن تعداد فرزندان یک گره:

```
alert(oHtml.childNodes.length); //outputs "2"
```

می توانیم از متدهای موسوم به item(). برای دسترسی نیز استفاده کنیم:

```
var oHead = oHtml.childNodes.item(0);
var oBody = oHtml.childNodes.item(1);
```

همچنین از دستور `document.body` را برای دسترسی به عنصر body صفحه استفاده می کند.

```
var oBody = document.body;
```

می توانیم صحت رابطه های سه متغیر oHead ، oBody و oHtml را به صورت زیر نشان دهیم:

```
alert(oHead.parentNode == oHtml);           //outputs "true"
alert(oBody.parentNode == oHtml);           //outputs "true"
alert(oBody.previousSibling == oHead);     //outputs "true"
alert(oHead.nextSibling == oBody);          //outputs "true"
alert(oHead.ownerDocument == document);    //outputs "true"
```

۶۱

۸

دسترسی به صفات عناصر

DOM برای دسترسی و دستکاری صفات یک عنصر سه متد تعریف کرده است:

`getAttribute(name)`: مقدار صفتی به نام name را از عنصری خاص برمی گرداند.

`setAttribute(name,new Value)`: مقدار صفتی به نام name را برابر new Value قرار می دهد.

`removeAttribute(name)`: صفتی به نام name را از عنصری مشخص حذف می کند.

این متدها برای دسترسی و دستکاری مستقیم صفت های یک عنصر بسیار مناسب اند. بنابراین برای به دست آوردن مقدار صفت ID تگی مشخص می توان به صورت زیر عمل کرد:

```
var sId = oP.getAttribute("id");
```

و برای تغییر مقدار صفت ID به صورت زیر عمل می کنیم:

```
oP.setAttribute("id", "newId");
```

دسترسی به گره های خاص

تا به اینجا با دسترسی به گره های فرزند و پدری آشنا شدیم. اما اگر بخواهیم به یک گره خاص، آن هم در عمق یک درخت دسترسی داشته باشیم چه؟ برای سهولت این کار، DOM چندین متد برای دسترسی مستقیم به گره ها فراهم آورده است.

`getElementsByName()`

از این متد برای دسترسی به لیستی(آرایه) از عناصر خاص استفاده می شود.

```
var oImgs = document.getElementsByTagName("img");
```

دستور فوق لیستی از تمام عناصر img صفحه را در oImgs ذخیره می کند.

فرض کنید می خواهیم به اولین عنصر عکس اولین پاراگراف صفحه دسترسی داشته باشیم:

```
var oPs = document.getElementsByTagName("p");
var oImgsInP = oPs[0].getElementsByTagName("img");
```

می توانیم از دستور زیر برای دسترسی به تمام عناصر صفحه استفاده کنیم:

```
var oAllElements = document.getElementsByTagName("*");
```

DOM مدل شی گرای سند**getElementsByName()**

برای دسترسی به عناصری که صفت name آنها برابر با مقدار خاص است از این متده استفاده می کند. به مثال زیر توجه کنید:

```
<html>
<head>
    <title>DOM Example</title>
</head>
<body>
    <form method="post" action="dosomething.php">
        <fieldset>
            <legend>What color do you like?</legend>
            <input type="radio" name="radColor" value="red" /> Red<br />
            <input type="radio" name="radColor" value="green" /> Green<br />
            <input type="radio" name="radColor" value="blue" /> Blue<br />
        </fieldset>
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
```

۶۲

۸

این صفحه رنگ مورد علاقه کاربر را سوال می کند. اما می خواهیم فقط مقدار radiobutton ی که انتخاب شده است را پیدا کنیم. برای ایجاد ارجاع به عناصر radiobutton می توان از کد زیر استفاده نمود.

```
var oRadios = document.getElementsByName("radColor");
```

حال می توانید از همان روش قبلی برای به دست آوردن مقدار هر از radiobutton ها به روش زیر عمل کنید:

```
alert(oRadios[0].getAttribute("value")); //outputs "red"
```

getElementById()

از این متده برای دسترسی به عناصر به وسیله خاصیت ID آنها استفاده می شود. می دانیم که خاصیت ID باید یکتا باشد به این معنی که هیچ دو عنصری نمی توانند داخل یک صفحه، ID یکسانی داشته باشند. این سریعترین و رایجترین راه برای دسترسی به عنصری خاص از صفحه است. به کد زیر نگاه کنید:

```
<html>
<head>
    <title>DOM Example</title>
</head>
<body>
    <p>Hello World!</p>
    <div id="div1">This is my first layer</div>
</body>
</html>
```

چنانچه بخواهیم از متدهای getElementsByTagName() برای دسترسی به عنصر div این صفحه با شناسه div1 استفاده کنیم باید به صورت زیر عمل کنیم:

```
var oDivs = document.getElementsByTagName("div");
var oDiv1 = null;
for (var i=0; i < oDivs.length; i++) {
    if (oDivs[i].getAttribute("id") == "div1") {
        oDiv1 = oDivs[i];
        break;
    }
}
```

اما می توانیم همین کار را به صورت زیر و با استفاده از متدهای getElementById() انجام دهیم:

```
var oDiv1 = document.getElementById("div1");
```

می بینید که استفاده از حالت دوم بسیار ساده تر ، کوتاه تر و بهینه تر است.

ایجاد و دستکاری گره ها

می توانیم از DOM برای اضافه کردن، حذف کردن و جابجایی و دیگر دستکاری ها استفاده کنیم.

ایجاد گره های جدید

برای ایجاد گره های جدید (از انواع مختلف) از متدهای زیر استفاده می شود:

- : برای ایجاد یک صفت جدید با name گرفته شده به کار می رود
- : برای ایجاد یک توضیح
- : برای ایجاد یک عنصر جدید استفاده می شود.
- : ایجاد یک متن ساده با عنوان text

DOM؛ مدل شی گرای سند

createElement() و createTextNode() appendChild()

فرض کنید تکه کد زیر را داریم:

```
<html>
  <head>
    <title>createElement() Example</title>
  </head>
  <body>
  </body>
</html>
```

۶۴

۸

حال می خواهیم عبارت زیر را در این صفحه چاپ کنیم:

```
<p>Hello World !</p>
```

اولین کار ایجاد یک عنصر `<p>` است.

```
var oP = document.createElement("p");
```

حال یک متن ساده ایجاد می کنیم:

```
var oText = document.createTextNode("Hello World!");
```

حال باید متن را به عنصر `<p>`، الحاق کنیم. برای این کار از متده استفاده می کنیم. از این متده برای اضافه کردن یک فرزند به انتهای لیست فرزندان یک گره استفاده می شود.

```
oP.appendChild(oText);
```

پاراگرافی که را ایجاد کرده ایم باید به صفحه و قسمت `body` و یا یکی از زیر مجموعه های آن الحاق کنیم. به این شکل:

```
oP.appendChild(oText);
```

 نکته: گره های عنصری از خاصیتی خواندنی/نوشتگی به نام `innerHTML`. برای بازیابی و تغییر محتوای یک عنصر پشتیبانی می کنند. این خاصیت جز استانداردهای W3C نمی باشد اما تمامی مرورگرها از آن پشتیبانی می کنند. فرض کنید تگ زیر را در صفحه داریم:

```
<p id="intro">Hello World!</p>
```

اجرای دستور زیر موجب نمایش پیغام Hello World! یعنی همان محتوای تگ `<p>` خواهد شد:

```
txt=document.getElementById('intro').innerHTML;
alert(txt);// Hello World!
```

آموزش کاربردی جاوا اسکریپت - احمد بادپی

با نسبت دادن مقداری جدید به این خاصیت با استفاده از علامت انتساب می توانیم محتوای یک تگ را تغییر دهیم:

```
Var myPara=document.getElementById('intro');
myPara.innerHTML="Click <a href=' #'>Here</a>";
```

تگ `<p>` به صورت زیر درخواهد آمد. زیرا خاصیت `innerHTML` محتوای تگ را به صورت کامل با مقدار جدید جایگزین می کند.

```
<p id="intro">Click <a href=' #'>Here</a></p>
```

`insertBefore()` و `replaceChild()` و `removeChild()`

طبعتا وقتی می توانیم گرهی را اضافه کنیم امکان حذف آن ها نیز وجود خواهد داشت. برای حذف گره ها از متده استفاده می کنیم. این متده یک آرگومان می گیرد که در واقع گرهی است که باید حذف شود. به شکل زیر:

```
var oP = document.body.getElementsByTagName("p") [0];
document.body.removeChild(oP);
```

برای جابجایی گره ها از متده `replaceChild()` استفاده می شود. از این تابع به صورت زیر استفاده می شود:

```
var oNewP = document.createElement("p");
var oText = document.createTextNode("Hello Universe! ");
oNewP.appendChild(oText);
var oOldP = document.body.getElementsByTagName("p") [0];
oOldP.parentNode.replaceChild(oNewP, oOldP);
```

برای اضافه کردن یک عنصر به قبل از عنصر دیگری از `insertBefore()` استفاده می شود. این متده دو آرگومان می پذیرد و آرگومان اول را قبل از آرگومان دوم قرار می دهد.

`createDocumentFragment()`

به محض اینکه تعدادی گره جدید به سند اضافه می کنیم صفحه برای نمایش تغییرات، به روزرسانی می شود. این رفتار برای تعداد تغییرات کم مناسب است. اما هنگامی که تغییرات زیاد باشد و صفحه بخواهد این رفتار را برای تک تک تغییرات تکرار کند ممکن است موجب کندی در عملکرد مرورگر شود.

برای رفع این مشکل می توانیم از یک تکه برنامه¹ استفاده کنیم. می توانیم تمام گره های جدید را به تکه برنامه اضافه کرده و سپس آن را در صفحه اصلی قرار دهیم. فرض کنید می خواهیم چندین پاراگراف را در صفحه ایجاد کنیم. در صورت استفاده از روش های قبلی این امر موجب رفرش هر باره صفحه خواهد شد.

اما بهتر است به روش زیر عمل کنیم:

¹ `documentFragment`

مدل شی گرای سند DOM

```

var arrText = ["first", "second", "third", "fourth", "fifth",
"sixth"];
var oFragment = document.createDocumentFragment();
for (var i=0; i < arrText.length; i++) {
    var oP = document.createElement("p");
    var oText = document.createTextNode(arrText[i]);
    oP.appendChild(oText);
    oFragment.appendChild(oP);
}
document.body.appendChild(oFragment);

```

۶۶

۸

ویژگی های منحصر به فرد DOM برای HTML

یکی از ویژگی های DOM این است که امکان تنظیم و دستکاری صفات مربوط به عناصر HTML را فراهم می‌آورد. از جمله این ویژگی ها می‌توان به در نظر گرفتن صفات عناصر به عنوان خاصیت های هر شی اشاره کرد که برای این کار متدها و خاصیت هایی ارائه شده است.

می‌توانیم به صفات عناصر به عنوان خاصیت های آن دسترسی داشته باشیم. فرض کنید عنصر زیر را داریم:

```

```

برای دسترسی و تنظیم src و border می‌توانیم از متدهای getAttribute() و setAttribute() استفاده کنیم:

```

alert(oImg.getAttribute("src"));
alert(oImg.getAttribute("border"));
oImg.setAttribute("src", "mypicture2.jpg");
oImg.setAttribute("border", "1");

```

می‌توانیم از نام صفات هم به عنوان خاصیت هر یک از اشیاء، برای بازیابی و تغییر مقدار صفات استفاده کنیم:

```

alert(oImg.src);
alert(oImg.border);
oImg.src = "mypicture2.jpg";
oImg.border = "1";

```

 نکته: برخلاف بسیاری از صفات تگ ها، نمی‌توانیم از خود صفت class به عنوان یک خاصیت استفاده کنیم. چون این کلمه جز کلمات رزرو شده است و باید به جای آن از کلمه className استفاده کنیم.

دستکاری قواعد سبک عناصر

گره های عنصری از شی ای به نام style برای دسترسی به قواعد سبک تعریف شده برای تگ ها پشتیبانی می کنند. سینتکس کلی استفاده از این شی به صورت زیر است:

```
document.getElementById('id').style.property="newValue";
```

به عنوان مثال فراخوانی دستور زیر موجب تغییر رنگ متن عنصری با id intro به رنگ قرمز خواهد شد:

```
document.getElementById('intro').style.color="red";
```

در جاوا اسکریپت به منظور دسترسی به آن دسته از قواعدی که شامل کarakتر - هستند می بایست ابتدا کarakتر - را حذف کرده و اولین حرف کلمه های بعد از آن را به صورت بزرگ بنویسیم. به عنوان مثال، کد زیر موجب تغییر رنگ پس زمینه عنصر به آبی خواهد شد:

```
document.getElementById('intro').style.backgroundColor="blue";
```

به عنوان مثالی دیگر برای دسترسی به قواعدی همچون padding-bottom و border-top-width می بایست به ترتیب از نام های paddingBottom و borderTopWidth استفاده نمود.

متدهای مربوطه به جداول

فرض کنید که می خواهیم جدول زیر را به صورت پویا و با استفاده از جاوا اسکریپت ایجاد کنیم:

```
<table border="1" width="100%>
  <tbody>
    <tr>
      <td>Cell 1,1</td>
      <td>Cell 2,1</td>
    </tr>
    <tr>
      <td>Cell 1,2</td>
      <td>Cell 2,2</td>
    </tr>
  </tbody>
</table>
```

اگر برای ایجاد این جدول بخواهیم از متدهای رایج DOM استفاده کنیم کد ما به صورت ذیل بسیار طولانی و گاهی اوقات سردرگم کننده خواهد شد:

مدل شی گرای سند DOM

۶۸

۸

```
//create the table
var oTable = document.createElement("table");
oTable.setAttribute("border", "1");
oTable.setAttribute("width", "100%");

//create the tbody
var oTBody = document.createElement("tbody");
oTable.appendChild(oTBody);

//create the first row
var oTR1 = document.createElement("tr");
oTBody.appendChild(oTR1);
var oTD11 = document.createElement("td");
oTD11.appendChild(document.createTextNode("Cell 1,1"));
oTR1.appendChild(oTD11);
var oTD21 = document.createElement("td");
oTD21.appendChild(document.createTextNode("Cell 2,1"));
oTR1.appendChild(oTD21);

//create the second row
var oTR2 = document.createElement("tr");
oTBody.appendChild(oTR2);
var oTD12 = document.createElement("td");
oTD12.appendChild(document.createTextNode("Cell 1,2"));
oTR2.appendChild(oTD12);
var oTD22 = document.createElement("td");
oTD22.appendChild(document.createTextNode("Cell 2,2"));
oTR2.appendChild(oTD22);

//add the table to the document body
document.body.appendChild(oTable);
```

برای آسانی اینکار DOM یکسری خصیت‌ها و متدهای منحصر به فردی برای عناصر اصلی جداول همچون table, tody, tody، tbody، tr ایجاد کرده است.

متدها و خصیت‌های منحصر به فرد جدول به شرح زیر می‌باشد:

- | | |
|--|-------------------------------------|
| : اشاره به عنصر caption جدول دارد. (البته اگر وجود داشته باشد.) | <input checked="" type="checkbox"/> |
| : مجموعه (آرایه) ای از عناصر tbody | <input checked="" type="checkbox"/> |
| : اشاره به عنصر tfoot جدول | <input checked="" type="checkbox"/> |
| : اشاره به عنصر thead جدول | <input checked="" type="checkbox"/> |
| : مجموعه ای از تمام ردیف‌های جدول | <input checked="" type="checkbox"/> |
| : ایجاد و قرار دادن یک عنصر جدید thead در جدول createThead() | <input checked="" type="checkbox"/> |
| : ایجاد و قرار دادن یک عنصر جدید tfoot در جدول createTfoot() | <input checked="" type="checkbox"/> |
| : ایجاد و قرار دادن یک عنصر جدید caption در جدول createCaption() | <input checked="" type="checkbox"/> |
| : حذف عنصر thead از جدول deleteThead() | <input checked="" type="checkbox"/> |

آموزش کاربردی جاوا اسکریپت - احمد بادپی

: حذف عنصر tfoot از جدول	<input checked="" type="checkbox"/>
: حذف عنصر Caption از جدول	<input checked="" type="checkbox"/>
: حذف ردیفی از جدول که در موقعیت position قرار دارد	<input checked="" type="checkbox"/>
: قرار دادن ردیفی در موقعیت position insertRow(position)	<input checked="" type="checkbox"/>

۶۹

۸

متدها و خصیصت‌های tbody

: مجموعه از ردیف‌ها در عنصر Rows	<input checked="" type="checkbox"/>
: حذف ردیفی در موقعیت position deleteRow(position)	<input checked="" type="checkbox"/>
: قرار دادن ردیفی در موقعیت position مجموعه‌ای از ردیف‌ها insertRow(position)	<input checked="" type="checkbox"/>

متدها و خصیصت‌های tr

: مجموعه‌ای از سلول‌ها در یک ردیف Cells	<input checked="" type="checkbox"/>
: حذف سلولی در موقعیت position deleteCell(position)	<input checked="" type="checkbox"/>
: قرار دادن سلولی در موقعیت position مجموعه‌ای از سلول‌ها insertCell(position)	<input checked="" type="checkbox"/>

برای ایجاد جدول قبلی کد ما به صورت زیر خواهد بود:

```
//create the table
var oTable = document.createElement("table");
oTable.setAttribute("border", "1");
oTable.setAttribute("width", "100%");
//create the tbody
var oTBody = document.createElement("tbody");
oTable.appendChild(oTBody);
//create the first row
oTBody.insertRow(0);
oTBody.rows[0].insertCell(0);
oTBody.rows[0].cells[0].appendChild(document.createTextNode("Cell
1,1"));
oTBody.rows[0].insertCell(1);
oTBody.rows[0].cells[1].appendChild(document.createTextNode("Cell
2,1"));
//create the second row
oTBody.insertRow(1);
oTBody.rows[1].insertCell(0);
oTBody.rows[1].cells[0].appendChild(document.createTextNode("Cell
1,2"));
oTBody.rows[1].insertCell(1);
oTBody.rows[1].cells[1].appendChild(document.createTextNode("Cell
2,2"));
//add the table to the document body
document.body.appendChild(oTable);
```

فصل نه

کار با فرمها و عناصر فرم

در صفحات وب فرم‌ها تنها عناصری هستند که کاربران می‌توانند به صورت مستقیم یکسری اطلاعات را در آن‌ها وارد نمایند.

برای ایجاد یک فرم از تگ form و برای ایجاد عناصر آن از تگ‌هایی همچون input ، select ، textarea و... استفاده می‌شود که مرورگرها بوسیله آن‌ها قادر به نمایش فیلد‌های یک خطی ، چند خطی ، منوهای بازشو ، دکمه‌ها و... هستند. در این فصل به بررسی روش‌های کار بر روی فرم‌ها از طریق جاوااسکریپت پرداخته و به نحوه اعتبار سنجی^۱ داده‌های وارد شده در یک فرم خواهیم پرداخت.

¹ Form validation

کار با فرمها و عناصر فرم

اساس یک عنصر فرم در صفحه

یک فرم در صفحه بوسیله تگ form که دارای صفت های زیر می باشد ایجاد می شود:

- :method**: مشخص می کند که مرورگر از چه روشی برای ارسال داده های فرم استفاده کند که می تواند یکی از دو مقدار GET و POST را بپذیرد.
- :action**: فرم ها پس از ارسال باید به یک صفحه پردازشگر که البته به یکی از زبان های تحت سور^۱ نوشته می شوند هدایت شوند. این صفت آدرس صفحه پردازشگر فرم را مشخص می کند.
- :enctype**: نوع رمز گذاری^۲ داده های فرم را هنگام ارسال مشخص می کند که در حالت پیش فرض برابر application/x-url-encoded است. اما در حالتی که داخل فرم خود عنصری از نوع file که کاربران را قادر به آپلود فایل ها می کند وجود داشته باشد باید آن را برابر multipart/form-data قرار دهیم.
- :accept**: لیستی از MIME type فایل هایی که کاربر مجاز به آپلود آن ها می باشد را تعیین می کند.
- :accept-charset**: لیستی از مجموعه کاراکترهایی را که سرور باید در هنگام دریافت اطلاعات استفاده کند، مشخص می کند.

۷۲

۹

نوشتن اسکریپت ها برای دسترسی به عناصر فرم

کدنویسی برای عناصر فرم نسبت به سایر عناصر کمی متفاوت است.

ایجاد ارجاع^۳ به عناصر مورد نظر

قبل از سرو کار داشتن با عناصر form باید ارجاعی به فرم مورد نظر خود در صفحه ایجاد کنیم. این کار از چندین روش انجام می شود.

روش اول، استفاده از متد getElementById(). است که از ID فرم برای دسترسی به آن استفاده می کند.
روش دوم استفاده از آرایه `forms` است که به عنوان یکی از خصیت های شی document در DOM معرفی شده است.
برای این کار می توان از اندیس عددی که بستگی به مکان فرم مورد نظر در صفحه دارد استفاده کرد. به عنوان مثال:

```
var oForm = document.forms[0] ; // get the first form
var oOtherForm = document.forms["formZ"] ; // get the form whose name
is "formZ"
```

دسترسی به عناصر داخل یک فرم

هر عنصر داخل یک فرم مثل یک دکمه، یک فیلد یک خطی و... با استفاده از آرایه ای به نام `elements` که یکی از خصیت های یک شی از نوع فرم می باشد قابل دسترسی است.

می توانید از این آرایه و با استفاده از اندیس عددی یا اسمی مورد نظر به عناصر مختلف فرم دسترسی داشته باشید.

¹ Server side

² Encoding

³ reference

```
var oFirstField = oForm.elements[0] ; // get the first form field
var oTextbox1 = oForm.elements["textbox1"] ; // get the field with the
                                         "textbox1"
```

خط اول از کد بالا متغیری را تعریف می کند که به اولین عنصر از فرمی به نام oForm اشاره می کند.

خط دوم نیز متغیری را تعریف می کند که به عنصری به نام 1 textbox از فرمی به نام oForm اشاره می کند.

یک روش دیگر (که اصطلاحا به آن روش میانبر می گویند) برای دسترسی به عناصری که نام مشخصی دارند استفاده می شود به شکل زیر است:

```
var oTextbox1 = oForm.textbox1; //get the field with the name
                               "textbox1"
```

کد بالا متغیری تعریف می کند که به عنصری با شناسه (یا ID) textbox1 از فرمی به نام oForm اشاره می کند.

اگر اسم عنصر مورد نظر دارای چند space باشد باید در اطراف آن از براکت ([]) استفاده کنیم:

```
var oTextbox1 = oForm.textbox1; //get the field with the name
                               "textbox1"
```

ویژگی ها و خصیت های عناصر form

تمامی عناصر فرم (به جز عنصری از نوع hidden) شامل یکسری خواص و رویدادهای مشترکی هستند که در زیر بیان می کنیم:

- خاصیت disabled:** از این خصیت هم برای تشخیص اینکه کدام عنصر در حالت غیر فعال قرار دارد و هم برای فعال یا غیر فعال کردن یک عنصر از قبل فعال استفاده می شود.
 - خاصیت form:** اشاره به فرمی دارد که عنصر مورد نظر ما، داخل آن قرار دارد.
 - متد focus:** این متده موجب می شود تمرکز^۱ صفحه بر روی عنصر مورد نظر قرار گیرد.
 - متد blur:** این متده عکس متده بالا است و موجب می شود تمرکز صفحه از روی عنصر مورد نظر خارج شود.
 - رویداد blur:** این رویداد موقعی که تمرکز صفحه از روی عنصر مورد نظر خارج شود رخ می دهد.
 - رویداد focus:** عکس رویداد بالا عمل می کند و موقعی که تمرکز بر روی عنصر مورد نظر قرار بگیرد رخ می دهد.
- برای مثال:

^۱ focus

کار با فرمها و عناصر فرم

VI

۹

```
var oField1 = oForm.elements[0];
var oField2 = oForm.elements[1];

//set the first field to be disabled
oField1.disabled = true;

//set the focus to the second field
oField2.focus();

//is the form property equal to oForm?
alert(oField1.form == oForm); //outputs "true"
```

نکته: عناصر از نوع `hidden` فقط از خاصیت `form` که در بالا ذکر شد پشتیبانی می کند.



ارسال فرم بواسیله جاوااسکریپت

در HTML فرستادن فرم از طریق یک دکمه از نوع `submit` یا عکسی که در نقش دکمه `submit` عمل می کند انجام می شود.

مثال:

```
<input type="submit" value="Submit" />
<input type="image" src="submit.gif" />
```

در صورت کلیک بر روی هر یک از دکمه های بالا فرم به صورت معمولی ارسال می شود.

چنانچه دکمه `Enter` صفحه کلید را نیز فشار دهید مرورگر فرم را مثل حالتی که دکمه کلیک می شود ارسال می کند.

برای تست ارسال شدن فرم می توانید از کد ساده زیر در تگ آغازین فرم مورد نظرتان استفاده کنید:

```
<form method="post" action="javascript:alert('Submitted')">
```

اگر می خواهید که از هیچ یک از روش های فوق استفاده نکنید می توانید از متدهی `submit()` به نام `submit()` استفاده کنید.

این متدهی از تعریفات DOM برای یک عنصر `form` است و می تواند هر جایی از صفحه استفاده شود. برای این کار اولاً باید ارجاعی به فرم مورد نظر ایجاد کرد (طبق روش هایی که قبلاً ذکر شد):

```
oForm = document.getElementById("form1");
oForm = document.forms["form1"];
oForm = document.forms[0];
```

بعد از این کار می توانید به راحتی از این متدهی استفاده کنید:

```
oForm.submit();
```

ارسال form فقط یکبار !

یکی از مشکلاتی که توسعه دهندگان در فرم ها با آن روبرو هستند این است که بسیاری از کاربران برای اطمینان از اینکه فرم به درستی ارسال شود چندین بار بر روی دکمه submit کلیک می کنند. مشکلی که در اینجا هست این است که به ازای هر بار کلیک کاربر بر روی دکمه یک درخواست اضافی به سرور ارسال می شود.

راه حل این مشکل بسیار ساده است: بعد از اینکه کاربر دکمه را کلیک کرد، آن را غیر فعال^۱ می کنیم.

برای انجام این کار می توان به جای استفاده از دکمه submit معمولی زیر:

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

از کد زیر استفاده کرد:

```
<input type="button" value="Submit" onclick="this.disabled=true;
this.form.submit()" />
```

موقعی که این دکمه کلیک می شود اولاً خود دکمه غیر فعال می شود و سپس فرمی را که جزوی از آن است، ارسال می کند. توجه کنید که در اینجا کلمه کلیدی this به دکمه اشاره دارد و form به فرم دربرگیرنده دکمه اشاره می کند.

همانطور که یک فرم را می توانیم بوسیله متدها submit() ارسال کنیم می توانیم آن را به وسیله متدهای به نام reset() نیز (پاک سازی) کنیم:

```
<input type="button" value="Reset" onclick="document.forms[0].reset()"
/>
```

کار با کادرهای متنی

دو نوع کادر متنی در HTML مورد استفاده قرار می گیرد.

یک خطی:

```
<input type="text"/>
```

و چند خطی:

```
<textarea>Content</textarea>
```

برای درست کردن یک کادر متنی یک خطی می بایست صفت type عنصر input را برابر text قرار دهیم. صفت size طول کادر را بر حسب تعداد کاراکترها مشخص می کند. مقدار صفت value نیز مقدار پیشفرض موجود داخل آن را مشخص می کند. صفت maxlength حداکثر تعداد کاراکترهایی که بتوان در کادر وارد کرد را مشخص می کند.

```
<input type="text" size="25" maxlength="50" value="initial value" />
```

¹ disabled

کار با فرمها و عناصر فرم

عنصر `textarea` برای ایجاد کادرهای متنی چند خطی مورد استفاده قرار می‌گیرد. از صفت‌های `rows` و `cols` برای مشخص کردن طول و عرض `textarea` استفاده می‌شود.

```
<textarea rows="25" cols="5">initial value</textarea>
```

برخلاف `input` این عنصر امکان مشخص کردن حداکثر تعداد کاراکتر‌های ورودی را ندارد.

79

9

بازیابی و تغییر مقدار یک `textbox`

اگر چه هر دو عنصر بالا تفاوت‌هایی دارند اما هر دوی آن‌ها از خاصیتی به نام `value` برای بازیابی مقدار وارد شده در آن‌ها پشتیبانی می‌کنند.

به عنوان مثال برای بازیابی مقدار وارد شده در فیلدی با شناسه (یا ID) `txt1` می‌توان به صورت زیر عمل کرد:

```
var oTextbox1 = document.getElementById("txt1");
```

چون مقداری که خاصیت `value` برمی‌گرداند یک رشته ساده است می‌توان از تمامی متدها و خواصی که قبل از رشته ها اشاره کردیم استفاده کرد.

```
alert ('oTextbox1.length');
```

از این خاصیت برای قراردادن مقادیر جدید در کادرهای متنی نیز می‌توان استفاده کرد. به عنوان مثال با دستور زیر می‌توان مقادیر جدیدی را به `oTextbox1` (که در بالا ذکر شد) اضافه کنیم:

```
oTextbox1.value='first textbox';
```

انتخاب متن‌های داخل کادرهای متنی

هر دو نوع فیلد بالا از متندی به نام `select()`. برای انتخاب تمامی متن داخل آن‌ها پشتیبانی می‌کنند. برای این کار اولاً تمرکز صفحه باید بر روی آن قرار گیرد. برای اطمینان از این امر باید همیشه قبل از متندی `select()` از متندی `focus()` استفاده نمایید. (البته این کار در تمامی مرورگرها الزامی نیست اما بهتر است همیشه انجام شود.)

به عنوان مثال برای انتخاب تمامی متن موجود در کادر متنی فوق:

```
oTextbox1.focus();
oTextbox1.select();
```

رویدادهای کادرهای متنی

هر دو نوع فیلد بالا علاوه بر پشتیبانی از رویدادهای `blur` و `focus` از دو رویداد جدید به نام‌های `change` و `select` نیز پشتیبانی می‌کنند.

:change این رویداد وقتی رخ می‌دهد که کاربر بعد از تغییر متن داخل کادرهای متنی، آن‌ها از حالت تمرکز صفحه خارج کند.

select: این رویداد وقتی رخ می دهد که یک یا چند کاراکتر از رشته های داخل یک کادر متنی چه به صورت دستی یا توسط متدهای select(). انتخاب شوند.

تفاوت رویداد های change و blur این است که رویداد blur تنها زمانی رخ می دهد که تمرکز صفحه از عنصر مورد نظر خارج شود و رویداد change نیز وقتی رخ می دهد که علاوه بر تغییر متن داخل textareaها، تمرکز صفحه نیز از آن ها خارج می شود.

اگر متن داخل کادر متنی ثابت باشد و فقط تمرکز صفحه از عنصر برود blur رخ می دهد اما اگر متن هم تغییر کرده باشد ابتدا رویداد change و به دنبال آن blur رخ خواهد داد.

انتخاب خودکار متن درون کادرهای متنی

برای انتخاب خودکار متن درون یک کادر متنی هنگامی که تمرکز صفحه بر روی آن ها قرار می گیرد می توان به راحتی از دستور `this.select()` در رویداد onFocus عنصر مورد نظر استفاده نمود.

به عنوان مثال:

```
<input type="text" onfocus="this.select(); ">
<textarea onfocus="this.select()"></textarea>
```

چرخش Tab بین عناصر فرم به صورت خودکار

بعد از تکمیل کادر های متنی که تعداد کاراکترهای مشخصی را می پذیرند می توانید کنترل (تمرکز) صفحه را به دیگر عناصر صفحه منتقل کنید.

برای این کار می توانیم از صفت maxlength در تگ های `<input />` استفاده کنیم:

```
<input type="text" maxlength="4" />
```

کاری که باید در اینجا انجام دهیم تشخیص وارد شدن حداکثر کاراکتر ها و فراخوانی متدهای focus() و test(). برای عنصر فرم بعدی است. برای این کار از تابعی به نام test استفاده می کنیم:

کار با فرمها و عناصر فرم

۹۸

```

function test(oTextbox) {
    var oForm = oTextbox.form;

    //make sure the textbox is not the last field in the form
    if (oForm.elements[oForm.elements.length-1] != oTextbox
        && oTextbox.value.length == oTextbox.getAttribute("maxlength")) {
        for (var i=0; i < oForm.elements.length; i++) {
            if (oForm.elements[i] == oTextbox) {
                for(var j=i+1; j < oForm.elements.length; j++) {
                    if (oForm.elements[j].type != "hidden") {
                        oForm.elements[j].focus();
                        return;
                    }
                }
            }
        }
    }
}

```

تابعی که ما نوشتمیم باید بعد از هر بار وارد کردن کاراکتر داخل textbox فراخوانی می شود. برای اینکار از رویداد onKeyUp استفاده خواهیم کرد به صورت زیر:

```
<input type='text' maxlength='4' onKeyUp='test(this)' />
```

محدود کردن کاراکتر های ورودی در یک textarea

اگر چه یک textfield دارای صفتی به نام maxlength برای محدود کردن کاراکتر های ورودی است اما یک textarea فاقد این صفت است. اما ما می توانیم توسط یک کد ساده javascript اینکار را انجام دهیم. برای این کار ابتدا تابعی به نام isNotMax() تعریف خواهیم کرد. به صورت زیر:

```

function isNotMax(oTextbox) {
    Return oTextbox.value.length != oTextarea.getAttribute('maxlength');
}

```

همانطور که می بینید این تابع بسیار ساده است. فقط تعداد کاراکترهای وارد شده در کادر متنی را با صفت maxlength عنصر .getattribute() برای استاندارد است اما می توانیم توسط متدهای مورد نظر مقایسه می کند و در صورتی که برابر نباشد true و در غیر اینصورت false را برمی گرداند. توجه داشته باشید صفت maxlength برای textarea صفتی غیر استاندارد است اما می توانیم توسط متدهای مقدار آن را بدست آوریم.

در مرحله بعد باید این تابع را در رویداد onKeyPress عنصرمان فراخوانی کنیم. این رویداد قبل از وارد کردن هر کاراکتر را خواهد داد که دقیقاً زمانی است که باید به حداکثر رسیدن تعداد کاراکتر های ورودی را چک کنیم. چیزی مانند کد زیر:

```
<textarea rows='10' cols='25' maxlength='150' onKeyPress='return  
isNotMax(this)'></textarea>
```

توجه کنید که مقدار برگشتی از تابع به کنترل کننده رویداد onKeyPress فرستاده می شود. البته این شیوه از راه های قدیمی کنترل رفتار پیش فرض یک رویداد است.

موقعی که تعداد کاراکتر های ورودی از MAX کمتر باشد تابع مقدار true به معنی ادامه رفتار عادی رویداد را برمی گرداند در غیر این صورت موجب جلوگیری از رفتار عادی رویداد و در نتیجه کاراکترهای بیش از حد مجاز خواهد شد.

کار با combobox ها و listbox ها

عنصر combobox ها و listbox ها در HTML بوسیله تگی به نام select ایجاد می شوند که به صورت پیش فرض مرورگرها این عنصر را به صورت combobox نشان می دهند.

```
<select name="selAge" id="selAge">  
    <option value="1">18-21</option>  
    <option value="2">22-25</option>  
    <option value="3">26-29</option>  
    <option value="4">30-35</option>  
    <option value="5">Over 35</option>  
</select>
```

مقدار صفت value آیتمی را که توسط کاربر انتخاب می شود به سرور فرستاده می شود. برای نشان دادن یک listbox فقط کافی است صفتی به نام size را با مقداری که مشخص کننده ای تعداد آیتم های قابل نمایش به صورت پیش فرض است به تگ select اضافه کنید. به عنوان مثال کد زیر listbox ای با 5 آیتم نمایشی بصورت پیش فرض را نمایش می دهد:

```
<select name="selAge" id="selAge" size="3">  
    <option value="1">18-21</option>  
    <option value="2">22-25</option>  
    <option value="3">26-29</option>  
    <option value="4">30-35</option>  
    <option value="5">Over 35</option>  
</select>
```

برای دسترسی به هر دو نوع عنصر فوق می توان طبق قواعدی که قبلاً گفتم عمل کنید:

```
oListbox = document.getElementById("selAge");
```

برای تمامی عناصر select آرایه ای به نام options که هر خانه آن اشاره به option از آن عنصر دارد تعریف کرده است.

کار با فرمها و عناصر فرم

می توانیم برای نمایش متن (عنوان) هر option و مقدار صفت value آن ها از روش های قبلی استفاده کنیم. مثلا:

```
alert(oListbox.options[1].text); //output display text
alert(oListbox.options[1].value); //output value
```

علاوه بر این، هر option دارای خاصیتی به نام index است که در واقع موقعیت آن را در آرایه options مشخص می کند.

```
alert(oListbox.options[1].index); //outputs "1"
```

البته چون options یک آرایه است می توانیم از خاصیتی به نام length برای مشخص کردن تعداد کل عناصر select استفاده کنیم.

```
alert("There are " + oListbox.options.length + " in the list.");
```

اما حال از کجا بفهمیم که کدام option (آیتم) توسط کاربر انتخاب شده است؟

بازیابی/تغییر دادن option(ها)ی انتخاب شده

عنصر select دارای خاصیتی به نام selectedIndex است که آیتم انتخاب شده را نگه داری می کند و در صورتی که هیچ آیتمی انتخاب نشده باشد مقدار 1- را برمی گرداند.

```
alert("The index of the selected option is " +
oListbox.selectedIndex);
```

اما همانطور که می دانید با اضافه کردن صفتی مانند 'select' به عنصر multiple='multiple' امکان انتخاب بیش از یک آیتم در آن واحد امکان پذیر است. در این صورت خاصیت selectedIndex حاوی اولین عنصر انتخاب شده از list خواهد بود و این کمکی به ما نمی کند. چون index تمام های انتخاب شده احتیاج داریم. برای این کار نیاز به یک تابع داریم.

این تابع در طول آیتم های یک listbox چرخش کرده و مقدار خاصیتی به نام selected که مشخص کننده انتخاب شدن یا نشدن آیتم است را بررسی کرده و آن option index را به آرایه ای اضافه می کند. خاصیت selected فقط می تواند یکی از مقادیر true (انتخاب شده) یا fasle (انتخاب نشده) را در بر داشته باشد.

```
function getSelectedIndexes (oListbox) {
    var arrIndexes = new Array;
    for (var i=0; i < oListbox.options.length; i++) {
        if (oListbox.options[i].selected) {
            arrIndexes.push(i);
        }
    }
    return arrIndexes;
};
```

از این تابع می توان هم برای بدست آوردن آیتم های انتخاب شده و هم تعداد آن ها استفاده کرد.

اضافه کردن option ها

می توانیم از طریق جاوا اسکریپت ، آیتم های جدیدی به list ها اضافه کنیم. برای این کار تابعی با سه آرگومان زیر طراحی می کنیم:

list که می خواهیم روی آن کار کنیم، نام آیتمی که می خواهیم اضافه کنیم و مقداری که می خواهیم اضافه کنیم.

بعد توسط متدهای قبلی DOM یک عنصر option جدید ایجاد کرده و آن را به عنصر select اضافه می کنیم:

```
function add (oListbox, sName, sValue) {
    var oOption = document.createElement("option");
    oOption.appendChild(document.createTextNode(sName));
    if (arguments.length == 3) {
        oOption.setAttribute("value", sValue);
    }
    oListbox.appendChild(oOption);
}
```

چون صفت value برای یک option اختیاری است می توانیم در صورتی که value برای تابع فرستاده شده است آن را به arguments.length==3 فرستاده شده یا نه از دستور option اضافه کنیم. برای چک کردن اینکه value استفاده می کنیم.

حذف option ها

جاوا اسکریپت علاوه بر امکان اضافه کردن option ها ، امکان حذف آن ها را نیز فراهم می کند.

یکی از روش های قدیمی برای این کار استفاده از آرایه options و قراردادن مقدار null برای عنصری از آن که می خواهیم حذف کنیم است:

```
oListbox.options[1] = null;
```

روش بهتر و جدیدتر استفاده از متدهای remove() و index(). است که آرگومان remove() عنصر مورد نظر برای حذف را می پذیرد:

```
var oListbox = document.getElementById("selListbox");
oListbox.remove(0); //remove the first option
```

می توان همانند روش اضافه کردن option ها تابعی برای حذف آن ها از list ها استفاده کرد:

```
function del (oListbox, iIndex) {
    oListbox.remove(iIndex);
}
```

کار با فرمها و عناصر فرم

چنانچه بخواهید هر یک از option های موجود در یک listbox را حذف کنید می توانید متدها remove() را برای هر کدام از آن ها فراخوانی کنید.

```
function clear (oListbox) {  
    for (var i=oListbox.options.length-1; i >= 0; i--) {  
        del(oListbox, i);  
    }  
}
```

کد بالا برای حذف، آیتم ها را بر عکس طی می کند. این کار الزامی است چرا که با هر بار حذف شدن یک آیتم از لیست خاصیت index هر option شماره گذاری مجدد می شود. به این دلیل بهتر است همیشه اول عنصری با بزرگترین index و سپس عناصر با index کوچکتر تر حذف شوند.

فصل ۵

رویدادهای خواهد و اسکریپت

تعاملات جاوا اسکریپت با HTML از طریق رخداد رویدادهایی که به واسطه دستکاری هایی که کاربر یا مرورگر بر روی صفحه انجام می دهد، امکان پذیر می شود. رویدادها و چگونگی تشخیص و کنترل آن ها یکی از مباحث مهم جاوا اسکریپت به شمار می رود. ما در این فصل ابتدا با مفهوم رویدادها آشنا شده و سپس به روش های کنترل و پاسخگویی به آن ها خواهیم پرداخت. سپس با انواع رویدادها آشنا شده و در انتهای با شی event برای تشخیص خصوصیات رویدادهای رخداده استفاده خواهیم کرد.

کنترل رویدادها

موقعی که صفحه بارگذاری می شود رویدادی رخ داده است، موقعی که کاربر بر روی دکمه ای کلیک می کند، باز هم رویدادی رخ داده است. توسعه دهنده این رویدادها برای اجرای کدهایی که به رویدادها پاسخ می دهند استفاده کنند. مثلا دکمه ای موجب بستن پنجره شود، پیغامی را به کاربر نمایش دهد، داده ها را اعتبارسنجی کند و...^۱

رویدادها در واقع عملیات خاصی هستند که یا توسط کاربر یا توسط خود مرورگر انجام می شوند. این رویداد ها نام هایی همچون mouseover ، click ، load و... دارند. اصطلاحاً به تابعی که در پاسخ به یک رویداد فراخوانی می شود کنترلگر رویداد^۱ می گویند. به عنوان مثال تابعی که برای پاسخ به رویداد click صدا زده می شود کنترلگر onclick نامیده می شود. برای مشخص کردن کنترلگرهای حادثه به دو روش می توان عمل کرد: از طریق جاوا اسکریپت یا از طریق HTML. برای مشخص کردن یک کنترلگر از طریق جاوا اسکریپت ابتدا باید به شی مورد نظر ارجاعی ایجاد کرده و سپس تابعی را به کنترلگر حادثه آن (که به صورت یک خاصیت برای آن تعریف شده است) منتسب کنیم. برای مثال:

```
var oDiv = document.getElementById('div1');
oDiv.onclick= function (){
    alert('I Was Clicked !!!!');
}
```

دقت کنید که در این روش باید تمامی حروف نام کنترلگر رویداد به صورت کوچک نوشته شود. در روش دوم می توانیم یک صفت کنترلگر رویداد، که اسکریپتی را به عنوان مقدار می پذیرد در تگ مربوطه قرار دهیم. به صورت زیر:

```
<div onclick='alert("I Was Clicked !!!")'></div>
```

در این روش نام کنترلگر حادثه می تواند به هر شکلی نوشته شود. در نتیجه onclick معادل است با: Click یا .ONCLICK

انواع رویداد ها

رویدادهایی که در مرورگر رخ می دهند معمولاً به چند دسته زیر تقسیم بندی می شوند:

- ☒ رویدادهای mouse که وقتی کاربر از طریق ماوس خود کارهایی را انجام می دهد، رخ می دهند.
- ☒ رویدادهای keyboard که وقتی کاربر دکمه ای از صفحه کلید را فشار می دهد رخ می دهند.
- ☒ رویدادهای HTML که موقعی که تغییراتی در پنجره مرورگر انجام می شوند رخ می دهند.
- ☒ رویدادهای تغییر که زمانی که تغییراتی در ساختار DOM صفحه انجام می شود رخ می دهند.

رویدادهای mouse

رایج ترین رویدادهایی هستند که رخ می دهند و به شرح زیر می باشند:

^۱ Event Handler

آموزش کاربردی جاوا اسکریپت - احمد بادپی

- click: موقعی که کاربر دکمه چپ ماوس را فشار می دهد. (نه دکمه راست). هنگامی که تمرکز صفحه بر روی یک دکمه باشد و کاربر کلید enter را هم بزند این رویداد رخ می دهد.
 - dblclick: موقعی که کاربر دو بار دکمه چپ ماوس را کلیک می کند رخ می دهد.
 - mousedown: موقعی که کاربر هر دکمه ای از ماوس را فشار دهد رخ می دهد.
 - mouseout: موقعی رخ میدهد که نشانگر موس بر روی عنصر است و کاربر آن را به بیرون از محدوده عنصر هدایت میکند.
 - mouseover: موقعی رخ می دهد که نشانگر موس از خارج از عنصر بر روی ان هدایت می شود.
 - mouseup: موقعی رخ می دهد که هر دکمه ای از ماوس رها می شود.
 - mousemove: مکررا هنگامی که نشانگر موس بر روی عنصری است رخ می دهد.
- تمامی عناصر موجود در یک صفحه از رویدادهای فوق به خوبی پشتیبانی می کنند.

ترتیب اجرایی رویدادها

قبل از رخداد رویداد click همیشه ابتدا رویداد mousedown و در پی آن mouseup و آخر سر click رخ می دهد.
در هنگام اجرای رویداد dblclick رویدادهای زیر به ترتیب اجرا می شوند:

- | | |
|-----------|----|
| mousedown | .۱ |
| mouseup | .۲ |
| click | .۳ |
| mousedown | .۴ |
| mouseup | .۵ |
| click | .۶ |
| dblclick | .۷ |

هنگام جا به جا شدن نشانگر ماوس از یک عنصر بر روی عنصر دیگر، ابتدا رویداد mouseout سپس رویداد mousemove برای عنصر بین این دو و آخر سر، رویداد mouseover رخ می دهد.

رویدادهای صفحه کلید

رویدادهای صفحه کلید به واسطه عملیاتی که کاربر بر روی صفحه کلید انجام می دهد رخ می دهند. رویدادهای صفحه کلید به شرح زیر می باشند:

- keydown: هنگامی که کلیدی از صفحه کلید زده می شود رخ می دهد. این رویداد مکررا زمانی که دکمه ای پایین نگه داشته شود نیز رخ می دهد.(این رویداد همیشه قبل از keypress رخ می دهد.)
- keypress: هنگامی که کلیدی از صفحه کلید زده می شود و به موجب آن یک کاراکتر برگردانده می شود رخ می دهد. این رویداد مکررا زمانی که کاربر دکمه ای را پایین نگه می دارد نیز رخ می دهد.(کلیدهای ctrl, alt, enter shift موجب روخداد این رویداد نمی شوند.)
- keyup: هنگامی رخ می دهد که دکمه ای که پایین بوده است رها شود.

رویدادها در جاوا اسکریپت

ترتیب اجرایی رویداد های صفحه کلید

موقعی که کاربر یک کلید کاراکتری را در صفحه کلید فشار می دهد رویدا های زیر به ترتیب اجرا می شوند:

۱. keydown
۲. keypress
۳. keyup

اگر کلیدی غیر کاراکتری مثل shift فشار داده شود رویدا های زیر به ترتیب اجرا می شوند:

۱. keydown
۲. keyup

اگر کاربر کلیدی کاراکتری را فشارداده و پایین نگه دارد رویدادهای keydown و keypress مکررا یکی پس از دیگری رخ می دهند تا زمانی که کلید رها شود.

اگر کاربر کلیدی غیر کاراکتری را فشارداده و پایین نگه دارد فقط رویداد keydown مکررا اجرا می شود.

دیگر رویداد ها

از دیگر رویدادهایی که ممکن است در صفحه و بر روی بعضی از عناصر رخ دهد می توان به موارد زیر اشاره نمود:

load: موقعی رخ می دهد که صفحه به طور کامل بارگذاری شود یا اینکه یک عنصر img یا object به طور کامل بارگذاری شوند. برای فعال کردن کنترلگرهای رویداد onload برای صفحه آن را در دستور <body> قرار می دهیم. برای مثال در عبارت زیر از این رویداد استفاده کرده ایم تا پس از خاتمه بارگذاری صفحه پیغام loading complete نمایش داده شود:

```
<body onload='alert("loading complete !!!")'></body>
```

این رویداد برای تگ هایی همچون body، img، embed و script قابل استفاده است.

unload: هنگامی رخ می دهد که کاربر صفحه بار شده جاری را بیندد. این می تواند به موجب زدن دکمه X (close) پنجه ره یا وارد کردن یک آدرس جدید در نوار آدرس مرورگر باشد.

abort: این رویداد برای یک object هنگامی که کاربر قبل از بارگذاری کامل آن ، عمل بارگذاری را متوقف کند رخ می دهد.

error: این رویداد برای یک صفحه هنگامی که در آن یک خطأ رخ می دهد، برای یک عکس هنگامی که نتواند بارگذاری شود و برای یک عنصر object هنگامی که نتواند بارگذاری شود رخ می دهد.

select: این رویداد هنگامی رخ می دهد که کاربر یک یا چند کاراکتر را از داخل یک ناحیه متنی (منظور تگ های select و textarea و input) انتخاب کند رخ می دهد.

change: بر روی یک ناحیه متنی، هنگامی که مقدار داخل ناحیه متنی تغییر کرده و در ادامه تمرکز صفحه از روی عنصر خارج شود و برای یک عنصر select هنگامی که مقدار آن تغییر می کند رخ می دهد.

submit: برای عنصر form، هنگامی که دکمه submit مربوط به فرم کلیک می شود رخ می دهد.

reset: برای عنصر form، هنگامی که دکمه reset مربوط به فرم کلیک می شود رخ می دهد.

focus: برای یک عنصر زمانی که تمرکز صفحه بر روی آن قرار می گیرد رخ می دهد.

blur: برای یک عنصر زمانی که تمرکز صفحه را از دست می دهد رخ می دهد.

شی event

شی event که در نسخه 1.2 و بالاتر جاوا اسکریپت در دسترس قرار گرفته است، شی خاصی است که به همراه هر رویداد برای کنترلگر آن رویداد فرستاده می شود. در واقع کنترلگر رویداد می تواند آن را به عنوان یکی از پارامترها دریافت کند و خصیت های شی event اطلاعاتی را در مورد آن رویداد در دسترس برنامه نویسان قرار می دهد.

بعضی از اطلاعاتی که این شی در اختیار قرار می دهد به شرح زیر است:

- شی ای که موجب رخداد رویداد شده است.
- اطلاعاتی در مورد نشانگر ماوس درهنگام رخداد رویداد
- اطلاعاتی در مورد صفحه کلید در هنگام رخداد رویداد

برای دسترسی به این شی می توان به چندین طریق عمل کرد:

در Internet Explorer، این شی به عنوان یکی از خواص شی window قابل دسترسی است. این بدین معنی است که یک کنترلگر رویداد به طریق زیر می تواند به شی event دسترسی داشته باشد:

```
oDiv.onclick = function () {
    var oEvent = window.event;
}
```

اگر چه این شی به عنوان یکی از خواص window شناخته می شود اما فقط زمانی در دسترس است که رویدادی رخداده باشد. بعد از اینکه کنترلگر رویداد به طور کامل اجرا شد، شی event نیز از بین خواهد رفت.

اما در استانداردهای DOM می توان از روش دسترسی به آرگومان تابع برای دسترسی به شی event استفاده کنیم. به عنوان مثال:

```
oDiv.onclick = function () {
    var oEvent = arguments[0];
}
```

البته می توان نامی برای این آرگومان مشخص کرد و از آن برای دسترسی استفاده نمود:

```
oDiv.onclick = function (oEvent) {
    ...
}
```

رویدادها در جاوا اسکریپت

خواص و متدهای شی event

این شی شامل خواص و متدهایی است که در ادامه بررسی خواهیم کرد:

- : اگر دکمه shiftkey زده باشد true و در غیر این صورت false را برمی گرداند.
- : اگر دکمه altkey زده باشد true و در غیر این صورت false را برمی گرداند.
- : اگر دکمه ctrlkey زده باشد true و در غیر این صورت false را برمی گرداند.
- : مشخص می کند که کدام یک از دکمه های ماوس زده شده اند. مقادیری که این خاصیت برمی گرداند به شرح زیر است:

- ❖ ۰: هیچ دکمه ای زده نشده است
- ❖ ۱: دکمه چپ زده شده است.
- ❖ ۲: دکمه راست زده شده است.
- ❖ ۳: دکمه چپ و راست با هم زده شده اند.
- ❖ ۴: دکمه وسط زده شده است.
- ❖ ۵: دکمه های چپ و وسط با هم زده شده اند.
- ❖ ۶: دکمه های راست و وسط با هم زده شده اند.
- ❖ ۷: هر سه دکمه با هم زده شده اند.

- : مختصات افقی و عمودی ماوس را نسبت به گوشه چپ و بالای صفحه در هنگام رخداد رویداد را مشخص می کند.

- : نوع رویدادی که رخداده است را برمی گرداند مثلا click و mouseover و ...
- : عددی است که مشخص می کند کدام دکمه از صفحه کلید فشار داده شده است. (در مرورگرهای همچون Opera, Firefox, ...)

- : شی ای را که در معرض حادثه قرار گرفته است را مشخص می کند (مانند یک سند یا یک پیوند).

فصل یازده

کارباکوکی‌ها

کوکی‌ها در واقع متغیرهایی هستند که در قالب یک فایل متنی ساده بر روی کامپیوتر کاربر ذخیره می‌شوند و در هر بار درخواست صفحه جدید از سرور با همان کامپیوتر، این فایل هم برای سرور فرستاده می‌شود. می‌توانیم از کوکی‌ها برای ذخیره یکسری اطلاعات خاص کاربران صفحات استفاده کنیم و در صورت نیاز آن‌ها را در صفحات دیگر مورد استفاده قرار دهیم. در این فصل ابتدا پس از بررسی روش ایجاد کوکی‌ها با روش‌های دسترسی و بازیابی مقادیر ذخیره شده در آن‌ها آشنا خواهیم شد.

کار با کوکی ها

ایجاد کوکی ها

برای ایجاد کوکی ها در جاوااسکریپت از خاصیت `document.cookie`. شی به شکل زیر استفاده می کنیم:

```
document.cookie="name=value ; expires=Date ; path = path ;
domain=domain";
```

۹۰

۱۱

و برای بازیابی تمامی کوکی های از قبل ایجاد شده به شکل زیر عمل خواهیم کرد:

```
var x = document.cookie;
```

همانطور که در دستور ابتدایی می بینید برای ایجاد کوکی می بایست رشته ای حاوی یکسری خواص و مقادیرشان را در قالب جفت های `name=value` (که با ; از هم جدا شده اند) به خاصیت `cookie`. نسبت دهیم. در جدول زیر هر یک از این قسمت ها را شرح می دهیم.

مثال	توضیحات	خاصیت
<code>name=ali</code>	این دستور نام و مقدار کوکی را مشخص می کند.	<code>name=value</code>
<code>expires=13/06/2003 00:00:00</code>	این خاصیت اختیاری زمان انقضای کوکی را مشخص میکند. مقداری که به این خاصیت داده می شود می بایست تاریخی به فرمت بازگشتی از متدهای <code>Date</code> و <code>toGMTString()</code> باشد. در صورتی که این خاصیت مشخص نشود هنگامی که کاربر پنجره مرورگر را بیند کوکی نیز از بین خواهد رفت.	<code>expires=date</code>
<code>path=/tutorials/</code>	این خاصیت اختیاری نام مسیری از سایت را که می تواند به کوکی دسترسی داشته باشد را مشخص می کند.	<code>path=path</code>
<code>domain = mysite.com</code>	این خاصیت اختیاری نام سایتی که می تواند از کوکی استفاده کند را مشخص می کند.	<code>domain=domain</code>

در مثال زیر یک کوکی با نام `username` و با مقدار `ali` که در تاریخ 15/02/2010 از بین می روید ایجاد می شود :

```
document.cookie = " username = ali ; expires = 15/02/2010 00:00:00 ";
```

در مثال زیر یک کوکی با نام `myCookie` و با مقدار `this is my cookie` ایجاد شده است:

```
document.cookie = "myCookie=" + escape("This is my Cookie");
```

نکته: در کد فوق تابع `escape()` یک رشته را دریافت کرده و تمامی کاراکترهای نامعتبر آن را به کد معادلش تبدیل می کند. قبل از کد معادل یک علامت % قرار می گیرد. به عنوان مثال این تابع کاراکتر space را به کد 20% تبدیل می کند. این تابع معادل تابع `encodeURIComponent()` است.



حذف کوکی ها

برای حذف یک کوکی می توان از تابعی که زمان انقضای کوکی را به یک ثانیه قبل تنظیم می کند استفاده کنیم. این تابع به صورت زیر است:

```
91
11
function delete_cookie ( cookie_name )
{
var cookie_date = new Date ( ) ; // current date & time
    cookie_date.setTime ( cookie_date.getTime() - 1 );
    document.cookie = cookie_name + "="; expires=" +
cookie_date.toGMTString();
}
```

حال کافی است برای حذف یک کوکی نام آن را برای تابع فوق بفرستیم. دستور زیر کوکی با نام username را حذف می کند:

```
delete_cookie ("username") ;
```

بازیابی کوکی ها

حال که با ایجاد و حذف کدن کوکی ها آشنا شدیم نحوه بازیابی (دسترسی) به آنها را بیان می کنیم. برای بازیابی کوکی هایی که قبلا ایجاد شده اند باز هم از خاصیت `document.cookie` به صورت زیر استفاده می کنیم:

```
var x = document.cookie;
```

این دستور لیستی (رشته) از جفت های name=value تمامی کوکی های قابل دسترس برای سند جاری را که با ; از هم جدا شده اند بر می گرداند. به عنوان مثال متغیر x می توانید حاوی رشته ای به صورت زیر باشد:

```
"username=ali; password=abc123"
```

در این مثال دو کوکی از قبل ایجاد شده است: یکی با نام `username` و مقدار `ali` و دومی با نام `password` با مقدار `.abc123`

اکنون x یک متغیر رشته ای ساده است که می توانیم برای دسترسی به هر یک از کوکی ها و مقدارشان ابتدا x را بوسیله متدهای `split` به آرایه ای تبدیل کرده و بوسیله متدهای خاص آرایه به آن ها دسترسی داشته باشیم. به عنوان مثال برای چاپ مقدار کوکی های فوق می توان به صورت زیر عمل کرد:

```
var allCookie      =      document.cookie;
Var cookieParts   =      allCookie.split(";");
Var fistCookie    =      cookieParts[0];
Var secondCookie  =      cookieParts[1];

Var nameOfFirstCookie =      fistCookie.split("=")[0];
Var valueOfFirstCookie =      fistCookie.split("=")[1];

Var nameOfSecondCookie =      fistCookie.split("=")[0];
Var valueOfSecondCookie =      fistCookie.split("=")[1];
```