بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

JavaScript HTML DOM EventListener

مدرس : مهندس افشین رفوآ

دوره آموزش Jquery

دوره آموزش JavaScript

JavaScript HTML DOM EventListener

تابع addEventListener()

مثال:

یک event listener اضافه می کنیم که به محض کلیک کاربر روی دکمه، اجرا می شود.

این مثال به وسیله ی متد addEventListener()، یک رخداد click به المان دکمه الصاق (متصل) می کند.

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method to attach a click event to a button.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    document.getElementById("myBtn").addEventListener("click", displayDate);
    function displayDate() {
      document.getElementById("demo").innerHTML = Date();
    }
  </script>
</body>
```

</html>

متد addEventListener() یک مدیریت کننده ی رخداد (event handler) به المان مورد نظر ضمیمه می کند.

متد مزبور یک مدیریت کننده ی رخداد به المان مورد نظر، بدون بازنویسی (overwrite) مدیریت کننده های رخداد موجود، ضمیمه می کند.

می توان event handler های متعددی به تنها یک عنصر افزود.

می توان event handler های متعددی از نوع یکسان به یک المان اضافه کرد.

همچنین می توان event-handler ها را به هر شی DOM مانند شی window افزود.

این امکان وجود دارد که event listener های (گوش فراخوان رخداد) متعددی به هر شی DOM (و نه لزوما اشیا HTML) افزود.

متد addEventListener() همچنین مدیریت نحوه ی واکنش event به bubbling را به مراتب آسان تر می سازد. در bubbling رخداد مورد نظر ابتدا توسط درونی ترین المان ضبط و مدیریت شده، سپس به المان های خارجی انتقال داده (propagate) می شود.

در واقع این متد عملیات event حبابی را تسهیل کرده و نحوه ی اجرای event حبابی را مشخص می کند. Event حبابی مفهومی است که در آن یک event از یک tag درونی به یگ تگ بیرونی انتقال می یابد.

هنگامی که از متد addEventListener() استفاده می کنید، جاوا اسکریپت به منظور خوانایی بهتر از نشانه گذاری های HTML (HTML markup) جدا سازی شده و نیز به شما اجازه می دهد حتی زمانی که HTML markup را کنترل نمی کنید، گوش فراخوان رخداد (event listener) به عنصر مورد نظر اضافه کنید.

جهت حذف گوش فراخوان رخداد نیز کافی است از متد removeEventListener() استفاده کنید.

ساختار نگارشی

element.addEventListener(*event, function, useCapture*);

اولین پارامتر بیانگر نوع رخداد (event) می باشد مانند "click" یا "mousedown" .

دومین پارامتر، تابعی است که می خواهیم به هنگام اجرا شدن رخداد فراخوانده شود.

سومین پارامتر یک مقدار بولی است که مشخص می کند از event حبابی (event bubbling) استفاده شود یا event capturing (ضبط و مدیریت رخداد). پارامتر نیز اختیاری است.

Event capturing : در event capturing، رخداد ابتدا توسط بیرونی ترین المان ضبط (capture) شده، سپس به عناصر درونی انتقال داده می شود.

نکته: توجه داشته باشید که نباید از پیشوند " on " برای رخداد استفاده کنید. به عنوان مثال بجای "onclick" از "click" استفاده کنید.پ

افزودن Event Handler به یک المان

مثال:

این مثال هنگامی که کاربر روی یک المان مشخص کلیک می کند، "Hello World!" را نمایش می دهد:

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method to attach a click event to a button.</p>
  <button id="myBtn">Try it</button>
  <script>
    document.getElementById("myBtn").addEventListener("click", function () {
      alert("Hello World!");
    });
  </script>
</body>
</html>
```

یا

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method to execute a function when a user clicks on a button.</p>
```

```
  <button id="myBtn">Try it</button>
  <script>
    document.getElementById("myBtn").addEventListener("click", myFunction);
    function myFunction() {
      alert("Hello World!");
    }
  </script>
</body>
</html>
```

افزودن چندین event handler به یک المان

تابع addEventListener() به شما امکان می دهد بدون بازنویسی (overwrite) رخدادهای موجود چندین رخداد متعدد جدید به تنها یک المان اضافه کنید:

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method to add two click events to the same button.</p>
  <button id="myBtn">Try it</button>
  <script>
    var x = document.getElementById("myBtn");
    x.addEventListener("click", myFunction);
    x.addEventListener("click", someOtherFunction);
    function myFunction() {
      alert("Hello World!");
    }
    function someOtherFunction() {
      alert("This function was also executed!");
    }
  </script>
</body>
</html>
```

می توان رخدادهایی از نوع مختلف به المانی یکسان افزود:

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method to add many events on the same button.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    var x = document.getElementById("myBtn");
    x.addEventListener("mouseover", myFunction);
    x.addEventListener("click", mySecondFunction);
    x.addEventListener("mouseout", myThirdFunction);
    function myFunction() {
```

```
    document.getElementById("demo").innerHTML += "Moused over!<br>";
  }
  function mySecondFunction() {
    document.getElementById("demo").innerHTML += "Clicked!<br>";
  }
  function myThirdFunction() {
    document.getElementById("demo").innerHTML += "Moused out!<br>";
  }
</script>
</body>
</html>
```

افزودن یک event handler به شی Window

متد addEventListener() به شما امکان می دهد گوش فراخوان های رخداد متعددی به (هر) شی HTML DOM همچون المان های HTML، سند HTML ، شی window و یا هر شی دیگری (مانند xmlHttpRequest) که از رخدادها پشتیبانی می کنند، اضافه کنید.

مثال:

یک گوش فراخوان رخداد (event listener) تخصیص دهید که به محض اینکه کاربر اندازه ی پنجره را تغییر داد، اجرا و راه اندازی شود:

```
<!DOCTYPE html>
<html>
<body>
  <p>This example uses the addEventListener() method on the window object.</p>
  <p>Try resizing this browser window to trigger the "resize" event handler.</p>
  <p id="demo"></p>
  <script>
    window.addEventListener("resize", function () {
      document.getElementById("demo").innerHTML = Math.random();
    });
  </script>
</body>
</html>
```

ارسال پارامتر

به هنگام ارسال مقادیر پارامتر، از یک "anonymous function" (یا تابع نام گذاری نشده) استفاده کنید که تابع مشخص شده را با پارامترهای مورد نظر فراخوانی کند:

```
<!DOCTYPE html>
```

```
<html>
<body>
  <p>
    This example demonstrates how to pass parameter values when using the
    addEventListener() method.
  </p>
  <p>Click the button to perform a calculation.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    var p1 = 5;
    var p2 = 7;
    document.getElementById("myBtn").addEventListener("click", function () {
      myFunction(p1, p2);
    });
    function myFunction(a, b) {
      var result = a * b;
      document.getElementById("demo").innerHTML = result;
    }
  </script>
</body>
</html>
```

Event Handling یا Event Bubbling ؟

در کل دو روش انتقال رخداد (event propagation) در HTML DOM وجود دارد که به ترتیب عبارتند از:

1. Bubbling
2. Capturing

انتقال رخداد (event propagation) یک روش برای تعیین کردن ترتیب رخداد event ها می باشد. چنانچه یک المان <p> داخل المان دارید و کاربر روی المان <p> کلیک کند، رخداد "click" کدام المان باید اول اجرا و مدیریت شود؟

در *bubbling*، اول رخداد درونی ترین المان اجرا می شود و بعد رخداد المان بیرونی، به عبارتی دیگر رخداد click المان <p> پیش از رخداد click عنصر مدیریت و اجرا می شود.

در *capturing*، رخداد بیرونی ترین المان پیش از رخداد درونی ترین المان اجرا می شود، به عبارتی دیگر رخداد click عنصر ابتدا اجرا می شود و بعد رخداد click المان <p>.

می توان نوع propagation (انتقال) را با استفاده از "useCapture" به عنوان پارامتر ورودی در متد addEventListener() تعیین کرد:

addEventListener(*event*, *function*, *useCapture*);

مقدار پیش فرض false است که از نوع انتقال bubbling استفاده می کند، اما همین که مقدار را روی true تنظیم می کنید، رخداد از نوع انتقال (propagation) capturing بهره می گیرد.

مثال:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      background-color: coral;
      border: 1px solid;
      padding: 50px;
    }
  </style>
</head>
<body>
  <p>This example demonstrates the difference between bubbling and capturing when adding event listeners.</p>
  <div id="myDiv">
    <p id="myP">Click this paragraph, I am Bubbling.</p>
  </div><br>
  <div id="myDiv2">
    <p id="myP2">Click this paragraph, I am Capturing.</p>
  </div>
  <script>
    document.getElementById("myP").addEventListener("click", function () {
      alert("You clicked the P element!");
    }, false);
    document.getElementById("myDiv").addEventListener("click", function () {
      alert("You clicked the DIV element!");
    }, false);
    document.getElementById("myP2").addEventListener("click", function () {
      alert("You clicked the P element!");
    }, true);
    document.getElementById("myDiv2").addEventListener("click", function () {
      alert("You clicked the DIV element!");
    }, true);
  </script>
</body>
</html>
```

متد removeEventListener()

متد removeEventListener()، مدیریت کننده های رخدادی (event handler) که به وسیله ی تابع addEventListener() به المان ها متصل شده اند، حذف می کند:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #myDIV {
      background-color: coral;
      border: 1px solid;
      padding: 50px;
      color: white;
    }
  </style>
</head>
<body>
  <div id="myDIV">
    This div element has an onmousemove event handler that displays a random number every time you move your mouse inside this orange field.
    <p>Click the button to remove the DIV's event handler.</p>
    <button onclick="removeHandler()" id="myBtn">Try it</button>
  </div>
  <p id="demo"></p>
  <script>
    document.getElementById("myDIV").addEventListener("mousemove", myFunction);
    function myFunction() {
      document.getElementById("demo").innerHTML = Math.random();
    }
    function removeHandler() {
      document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
    }
  </script>
</body>
</html>
```

پشتیبانی مرورگرها از متدهای حذف و اضافه ی گوش فراخوان رخداد

Method					
addEventListener()	1.0	9.0	1.0	1.0	7.0
removeEventListener()	1.0	9.0	1.0	1.0	7.0

توجه: دو تابع ذکر شده در مرورگرهای IE ویرایش 8.0 و Opera ویرایش 7.0 و نسخه های پیشین این دو مرورگر پشتیبانی نمی شوند. برای ورژن های نام برده ی این دو مرورگر، به منظور متصل کردن و حذف event handler به المان مورد نظر، به ترتیب از دو متد attachEvent() و detachEvent() استفاده می کنیم.

element.attachEvent*(event, function);*

element.detachEvent*(event, function);*

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>The addEventListener() method is not supported Internet Explorer 8 and earlier versions.</p>
  <p>This example demonstrates a solution that will work for all browsers.</p>
  <button id="myBtn">Try it</button>
  <script>
    var x = document.getElementById("myBtn");
    if (x.addEventListener) {
      x.addEventListener("click", myFunction);
    } else if (x.attachEvent) {
      x.attachEvent("onclick", myFunction);
    }
    function myFunction() {
      alert("Hello World!");
    }
  </script>
</body>
</html>
```