Was ist ein Ereignis-Listener?

Was ist ein Ereignis-Listener?

Ein Ereignis-Listener ist eine Funktion in JavaScript, die auf das Eintreten eines Ereignisses wartet und dann darauf reagiert. JavaScript ist eine Programmiersprache, die Entwickler verwenden, um interaktive Webseiten zu erstellen. Mit der Ereignis-Listener-Funktion von JavaScript können Sie benutzerdefinierte Antworten auf Ereignisse wie Mausklicks, Tastaturklicks und Fenstergrößenänderung erstellen. Das Programmierparadigma des Wartens und Reagierens auf Ereignisse in Echtzeit wird als Ereignis-Handhabung bezeichnet.

Lesen Sie mehr über JavaScript“

Wie lautet die Syntax der Ereignis-Listener-Funktion?

Die Ereignis-Listener-Funktion weist ähnliche Eigenschaften wie andere JavaScript-Funktionen auf. Wenn sie aktiviert ist, ergreift sie die notwendigen Maßnahmen, um das Ereignis zu verarbeiten. Die Ereignis-Listener-Funktion kann beispielsweise den angezeigten Text ändern, Informationen aus Registrierungsformularen sammeln oder Daten in Datenbanken speichern.

Syntax des Ereignis-Listeners

Eine Ereignis-Listener-Funktion folgt der richtigen JavaScript-Syntax, wie in diesem nächsten Beispiel.

```
function RespondMouseClick() {

    document.getElementById("textdisplay1").innerHTML += "MouseClick happened" ;

}
```

Dieses Beispiel zeigt die *RespondMouseClick*-Ereignis-Listener-Funktion. Es ist üblich, den Funktionsnamen so zu schreiben, dass er den Zweck des Ereignis-Listeners widerspiegelt. In der Funktion schreiben Sie Codes, um bestimmte Aktionen durchzuführen, wenn das Ereignis eintritt. In diesem Beispiel fügt die Funktion den Text *MouseClick happened* an das HTML Element *textdisplay1* an.

Syntax des Ereignis-Handhabers

Alternativ können Sie eine Ereigns-Handhaber-Funktion verwenden, um auf das aktivierte Ereignis zu reagieren, wie im folgenden Beispiel.

```
} else {
console.log ("full screen error");
}
}
```

So können Sie mehrere Arten von Ereignissen eines bestimmten Elements mit einer Funktion verwalten.

Sie könnten beispielsweise einen Ereignis-Listener registrieren, um alle Arten von Blockchain-Ereignissen in ereignisbasierten Anwendungen zu verarbeiten. Weitere Informationen finden Sie unter Erstellen einer ereignisbasierten Anwendung mit Amazon Managed Blockchain.

Wie fügt man einen Ereignis-Listener hinzu?

Ein Ereignis-Listener wird erst aktiviert, nachdem Sie ihn zum jeweiligen JavaScript-Element hinzugefügt haben. Um das zu tun, können Sie eine Syntax wie diese verwenden:

- *element.addEventListener(event, listener);*
- *element.addEventListener(event, listener, useCapture);*
- *element.addEventListener(event, listener, options);*

Entwickler können beispielsweise die folgende Funktion aufrufen, um den Klick-Ereignis-Listener an ein Schaltflächenelement zu binden.

```
btn.addEventListener("click", RespondMouseClick);
```

Sie können einem bestimmten Ereignis-Objekt auch mehrere Ereignis-Listener hinzufügen, ohne bestehende Ereignis-Handhaber zu überschreiben.

Amazon Web Services (AWS) ermöglicht es Entwicklern beispielsweise, mehrere Callbacks für das *AWS.Request*-Objekt zu verketten. Weitere Informationen finden Sie in der Anleitung zur Verwendung eines Ereignis-Listeners für Anforderungsobjekte in AWS.

Parameter für das Hinzufügen von Ereignissen

Hier ist eine Erklärung der Parameter in der vorherigen Syntax:

- *Der useCapture-Parameter* ist ein optionaler Parameter, der den Modus der Ereignisübertragung angibt. Er akzeptiert boolesche Werte, wobei *„wahr“ die Erfassung aktiviert*, während *„falsch“ das Bubbling* aktiviert. Der Standardwert dieses Parameters ist auf *falsch* gesetzt.
- *Der Optionsparameter* besteht aus mehreren optionalen Werten, einschließlich Erfassungsmodus und Abweisungssignalen, die das Verhalten des Listeners darstellen.

Wie entfernt man einen Ereignis-Listener?

Ereignis-Listener bleiben aktiv, bis Sie sie aus den zugehörigen JavaScript-Elementen entfernen. Sie können dazu die folgende Syntax verwenden.

element.removeEventListener(type, listener, useCapture);

Die Parameter zum Entfernen von Ereignis-Listenern ähneln denen, die Sie zum Hinzufügen von Ereignis-Listenern verwenden. Wenn Sie einen Ereignis-Listener entfernen, müssen Sie dieselben Parameter für*Typ, Listener*und *useCapture* angeben. Wenn Sie dies nicht tun, bleibt der Ereignis-Listener aktiv und wird weiterhin für zukünftige Ereignisse ausgelöst.

Sie können beispielsweise ein Ereignis mit dem folgenden Code hinzufügen.

button.addEventListener("click", RespondMouseClick, true);

Beim Anwenden des folgenden Codes kann der Ereignis-Listener jedoch nicht entfernt werden. Das liegt daran, dass sich der *useCapture*-Wert von dem unterscheidet, der für das Schaltflächenobjekt registriert wurde.

button.removeEventListener("click", RespondMouseClick, false);

Um das Ereignis erfolgreich zu entfernen und zu verhindern, dass es ausgelöst wird, können Sie den folgenden Code verwenden.

button.removeEventListener("click", RespondMouseClick, true);

Wie funktionieren verschachtelte Ereignis-Listener-Funktionen?

untergeordneten Elements ist.

* *

* *

* *

* *

* *

Komplexe Webanwendungen können mehrere übergeordnete und untergeordnete Ebenen mit entsprechenden Ereignis-Listener-Funktionen haben. Wenn ein bestimmtes Ereignis eintritt, löst es Ereignis-Listener auf verschiedenen Ebenen in einer bestimmten Reihenfolge aus. Wenn Sie beispielsweise auf eine untergeordnete Schaltfläche klicken, wird das Ereignis an alle Handhaber weitergegeben, die einen Mausklick erfassen.

Ereignisse können sich in zwei Modi ausbreiten - Bubbling und Erfassung.

Ereignis-Bubbling

Bubbling ist der Standardmodus der JavaScript-Ereignisbehandlung. Er überträgt das Ereignis von der innersten Schicht zur äußersten Schicht.

Zum Beispiel könnte ein Benutzer mit dem Mauszeiger über ein Textfeld im untergeordneten Abschnitt fahren. Dann könnte die Anwendung das Ereignis in der folgenden Reihenfolge weitergeben:

1. Der Ereignis-Listener im untergeordneten Element verarbeitet das Maus-Hover-Ereignis.
2. Dann verarbeitet der übergeordnete Ereignis-Listener das Ereignis und übergibt die Steuerung an den Ereignis-Listener des Dokuments.

Verwenden Sie die folgende Syntax, um Ereignis-Bubbling einzurichten:

- *element.addEventListener(event, listener, [false]);*
- *element.addEventListener(event, listener);*

Ereignisserfassung

Die Ereigniserfassung ist ein spezieller Ereignisbehandlungsmodus in JavaScript, bei dem sich das Ereignis von der äußersten Schicht nach innen ausbreitet. Sobald das Ereignis das

1. Der Dokument-Ereignis-Listener verarbeitet das *Mausklick-Ereignis*, gefolgt vom übergeordneten Ereignis-Handhaber.
2. Das Ereignis erreicht das Zielelement, das die Schaltfläche ist. Der Ereignis-Listener der Schaltfläche verarbeitet das Ereignis.
3. Die Ereignishandhabung wechselt vom Erfassungs- in den Bubbling-Modus.
4. Dasselbe Mausklickereignis löst den Ereignis-Handhaber beim übergeordneten Objekt aus, bevor er im Dokument endet.

Wie man zwischen Blubbling und Erfassung wählt

Bubbling und Erfassung ermöglichen es Ihnen, mit Ereignissen anders umzugehen. Bubbling wird selten verwendet, da die Erfassung ausreicht, um die meisten Ereignisse in Anwendungen zu verarbeiten.

Wenn Sie sich zwischen Bubbling und Erfassung entscheiden, sollten Sie den Ablauf der Ereignisübertragung und dessen Abstimmung mit der Programmierlogik Ihrer Anwendung berücksichtigen.

Nehmen wir zum Beispiel ein übergeordnetes Formular, das aus zwei untergeordneten Elementen besteht. Das erste Element erfordert eine sofortige Aktualisierung, wenn am zweiten Element ein Ereignis eintritt. In diesem Fall sollten Sie den Erfassungsmodus verwenden. Es stellt sicher, dass der übergeordnete Ereignis-Listener das Ereignis verarbeitet und das erste Element aktualisiert. Dann übergibt es die Kontrolle an den Ereignis-Listener am zweiten untergeordneten Element.

Wie stoppt man die Übertragung von Ereignissen in verschachtelten Ereignis-Listener-Funktionen?

Ereignisse werden in einer verschachtelten Listener-Anordnung so lange weitergegeben, bis sie das Endziel erreichen. Sie müssen bestimmte Methoden anwenden, um die weitere Ausbreitung des Ereignisses zu verhindern.

Die folgende Methode stoppt das Ereignis am Ereignis-Listener.

event.stopPropagation();

Wenn Sie *StopPropagation* beispielsweise auf der untergeordneten Schaltfläche aufrufen, wird das Mausklickereignis nicht auf die übergeordnete Ebene und auf die Dokumentebene

Objekt verschiedene Typen von Ereignis-Listenern registriert sind, werden diese trotz des *StopPropagation*-Aufrufs weiter ausgelöst.

Um alle Ereignisse zu beenden, die sich auf ein bestimmtes Objekt beziehen, können Sie die *StopImmediatePropagation-Methode* wie folgt verwenden.

```
event.stopImmediatePropagation();
```

Wenn ein Ereignis-Listener *StopImmediatePropagation* aufruft, werden keine anderen mit dem Objekt verknüpften Ereignis-Listener ausgelöst.

Wie kann AWS Ihre JavaScript-Anforderungen unterstützen?

Amazon Web Services (AWS) bietet das AWS-SDK für JavaScript an, sodass Sie Services in Ihren Anwendungen mit Bibliotheken und APIs problemlos verwenden können.

Sie können das SDK für die Entwicklung von serverseitigen, Web- und mobilen Webanwendungen verwenden. Das SDK unterstützt JavaScript-Runtime, Node.JS und React Native sowie Cross-Laufzeit. So können Entwickler dasselbe Client-Servicepaket auf verschiedenen Plattformen ausführen.

Hier sind weitere Vorteile der Verwendung des SDK:

- Das SDK ist in TypeScript geschrieben. Dies bietet Vorteile wie statische Typüberprüfung und Unterstützung für Klassen und Module.
- Das SDK bietet einen Middleware-Stack, mit dem Sie benutzerdefinierte Aktionen einführen können.
- Das SDK hat eine modulare Architektur. So können Sie nur die Pakete verwenden, die zur Optimierung der Anwendungsleistung erforderlich sind.

Beginnen Sie mit JavaScript-Anwendungen, indem Sie sich noch heute für ein AWS-Konto registrieren.

Nächste Schritte in AWS

Kostenfreier Einstieg | Kontakt

Zusätzliche produktbezogene Ressourcen ansehen

Entwickler-Tools-Services ausprobieren »

Registrieren Sie sich und erhalten Sie ein kostenloses Konto

Sie erhalten sofort Zugriff auf das kostenlose Kontingent von AWS.

Registrieren »

Mit der Entwicklung in der Konsole starten

Starten Sie mit der Entwicklung in der AWS-Managementkonsole.

Anmelden »

Bei der Konsole anmelden

Mehr über AWS erfahren

- Was ist AWS?
- Was ist Cloud Computing?
- Inklusion, Vielfalt und Gleichstellung bei AWS
- Was ist DevOps?
- Was ist ein Container?
- Was ist ein Data Lake?
- AWS Cloud-Sicherheit
- Neuerungen
- Blogs
- Pressemitteilungen

Ressourcen für AWS

- Erste Schritte
- Schulung und Zertifizierung
- AWS-Lösungsbibliothek
- Architekturzentrum
- Häufig gestellte Fragen zu Produkt und Technik
- Berichte von Analysten
- AWS-Partner

Kostenfreier Einstieg

Kontakt