

Веб-компоненты

Веб-компоненты — это набор различных технологий, позволяющих создавать многократно используемые настраиваемые элементы — с их функциональностью, изолированной от остального кода — и использовать их в своих веб-приложениях.

Концепции и использование Как разработчики, мы все знаем, что повторное использование кода — это хорошая идея. Традиционно это было не так просто для пользовательских структур разметки — подумайте о сложном HTML (и связанном с ним стиле и сценарии), который вам иногда приходилось писать для отображения пользовательских элементов управления пользовательского интерфейса, и о том, как их многократное использование может превратить вашу страницу в беспорядок, если вы не будете осторожны.

Веб-компоненты призваны решить такие проблемы — они состоят из трех основных технологий, которые можно использовать вместе для создания универсальных пользовательских элементов с инкапсулированной функциональностью, которые можно повторно использовать где угодно, не опасаясь коллизий кода.

Пользовательские элементы Набор API-интерфейсов JavaScript, которые позволяют вам определять пользовательские элементы и их поведение, которые затем можно использовать по желанию в вашем пользовательском интерфейсе.

Тень DOM Набор API-интерфейсов JavaScript для прикрепления инкапсулированного «теневое» дерева DOM к элементу, который отображается отдельно от основного документа DOM, и управления соответствующей функциональностью. Таким образом, вы можете сохранить свойства элемента в тайне, чтобы их можно было использовать в сценариях и стилях, не опасаясь столкновения с другими частями документа.

HTML-шаблоны Элементы `<template>` и `<slot>` позволяют писать шаблоны разметки, которые не отображаются на отображаемой странице. Затем их можно повторно использовать несколько раз в качестве основы структуры пользовательского элемента.

Базовый подход к реализации веб-компонента обычно выглядит примерно так:

Создайте класс, в котором вы указываете функциональность веб-компонента, используя синтаксис класса. Зарегистрируйте новый пользовательский элемент с помощью `CustomElementRegistry.define()` метода, передав ему имя определяемого элемента, класс или функцию, в которой указаны его функциональные возможности, и, необязательно, элемент, от которого он наследуется. При необходимости прикрепите теневую DOM к пользовательскому элементу с помощью `Element.attachShadow()` метода. Добавьте дочерние элементы, прослушатели событий и т. д. в теневой DOM, используя обычные методы DOM. При необходимости определите шаблон HTML с помощью `<template>` и `<slot>`. Снова используйте обычные методы DOM, чтобы клонировать шаблон и прикрепить его к теневой модели DOM. Используйте свой пользовательский элемент в любом месте на странице, как и любой обычный элемент HTML. Путеводители Использование пользовательских элементов Руководство, показывающее, как использовать функции пользовательских элементов для создания простых веб-компонентов, а также изучение обратных вызовов жизненного цикла и некоторых других более сложных функций.

Использование теневой DOM Руководство, в котором рассматриваются основы теневой DOM,

показывающие, как прикрепить теневой DOM к элементу, добавить его в дерево теневого DOM, стилизовать его и многое другое.

Использование шаблонов и слотов Руководство, показывающее, как определить многократно используемую структуру HTML с помощью элементов `<template>` и `<slot>`, а затем использовать эту структуру в веб-компонентах.

Ссылка Пользовательские элементы CustomElementRegistry Содержит функциональные возможности, связанные с пользовательскими элементами, в первую очередь CustomElementRegistry.define() метод, используемый для регистрации новых пользовательских элементов, чтобы их можно было затем использовать в вашем документе.

Window.customElements Возвращает ссылку на CustomElementRegistry объект.

Обратные вызовы жизненного цикла Специальные функции обратного вызова, определенные внутри определения класса пользовательского элемента, которые влияют на его поведение:

connectedCallback() Вызывается, когда пользовательский элемент впервые подключается к DOM документа.

disconnectedCallback() Вызывается, когда пользовательский элемент отключается от DOM документа.

adoptedCallback() Вызывается, когда пользовательский элемент перемещается в новый документ.

attributeChangedCallback() Вызывается при добавлении, удалении или изменении одного из атрибутов пользовательского элемента.

Расширения для создания пользовательских встроенных элементов Определены следующие расширения:

Глобальный isAttribute HTML Позволяет указать, что стандартный элемент HTML должен вести себя как зарегистрированный настраиваемый встроенный элемент.

Вариант «есть» Document.createElement() метода Позволяет создать экземпляр стандартного HTML-элемента, который ведет себя как заданный зарегистрированный настраиваемый встроенный элемент.

Псевдоклассы CSS Псевдоклассы, относящиеся конкретно к пользовательским элементам:

:defined Соответствует любому определенному элементу, включая встроенные элементы и пользовательские элементы, определенные с помощью CustomElementRegistry.define().

:host Выбирает теневой хост теневого DOM, содержащего CSS, внутри которого он используется.

:host() Выбирает теневой хост теневого DOM, содержащего CSS, внутри которого он используется (так что вы можете выбрать пользовательский элемент внутри его теневого DOM), но только если селектор, указанный в качестве параметра функции, соответствует теневому хосту.

`:host-context()` Выбирает теневого хост теневого DOM, содержащего CSS, внутри которого он используется (так что вы можете выбрать пользовательский элемент внутри его теневого DOM) — но только если селектор, указанный в качестве параметра функции, соответствует предку(ам) теневого хоста в поместите его внутри иерархии DOM.

Псевдоэлементы CSS Псевдоэлементы, относящиеся конкретно к пользовательским элементам:

`::part` Представляет любой элемент в теновом дереве, который имеет соответствующий `part` атрибут.

Тень DOM `ShadowRoot` Представляет корневой узел теневого поддерева DOM.

Element расширения Расширения интерфейса Element, связанные с теновым DOM:

Метод `Element.attachShadow()` прикрепляет теновое дерево DOM к указанному элементу. Свойство `Element.shadowRoot` возвращает теновой корень, прикрепленный к указанному элементу, или `null` если теновой корень не прикреплен. Соответствующие Node дополнения Дополнения к Node интерфейсу, относящиеся к теновому DOM:

Метод `Node.getRootNode()` возвращает корень объекта контекста, который необязательно включает теновой корень, если он доступен. Свойство `Node.isConnected` возвращает логическое значение, указывающее, подключен ли узел (прямо или косвенно) к объекту контекста, например, к объекту `Document` в случае обычного DOM или `ShadowRoot` в случае теневого DOM. Event расширения Расширения интерфейса Event, связанные с теновым DOM:

`Event.composed` Возвращает значение `true`, если событие будет распространяться через границу теневого DOM в стандартный DOM, в противном случае `false`.

`Event.composedPath` Возвращает путь события (объекты, для которых будут вызываться прослушатели). Это не включает узлы в теновых деревьях, если теновой корень был создан с `ShadowRoot.mode` закрытым.

HTML-шаблоны `<template>` Содержит фрагмент HTML, который не отображается при первоначальной загрузке содержащего документа, но может отображаться во время выполнения с помощью JavaScript, который в основном используется в качестве основы для структур пользовательских элементов. Связанный интерфейс DOM — `HTMLTemplateElement`.

`<slot>` Заполнитель внутри веб-компонента, который можно заполнить собственной разметкой, что позволяет создавать отдельные деревья DOM и представлять их вместе. Связанный интерфейс DOM — `HTMLSlotElement`.

Глобальный `slot` атрибут HTML Назначает элементу слот в теновом дереве теневого DOM.

`Element.assignedSlot` Атрибут только для чтения, который возвращает ссылку на элемент, `<slot>` в который вставлен этот элемент.

`Text.assignedSlot` Доступный только для чтения атрибут, возвращающий ссылку на элемент, `<slot>` в который вставлен этот текстовый узел.

Element расширения Расширения интерфейса Element, связанные со слотами:

`Element.slot` Возвращает имя теневого слота DOM, прикрепленного к элементу.

Псевдоэлементы CSS Псевдоэлементы, относящиеся конкретно к слотам:

::slotted Соответствует любому содержимому, которое вставляется в слот.

событие slotchange_ Запускается для HTMLSlotElementэкземпляра (<slot>элемента), когда узлы, содержащиеся в этом слоте, изменяются.

Примеры Мы создаем ряд примеров в нашем репозитории GitHub с примерами веб-компонентов . Со временем будет добавлено больше.

From:
<https://book51.ru/> - **book51.ru**

Permanent link:
https://book51.ru/doku.php?id=software:development:web:docs:web:api:web_components

Last update: **2023/08/22 21:58**

