**ESEMPI DI UTILIZZO**

Per utilizzare l'API, lo sviluppatore deve fornire e tenere traccia di una serie di informazioni chiave. Questi bit di informazioni vengono passati al costruttore [PaymentRequest](https://w3c.github.io/payment-request/#dom-paymentrequest) come argomenti e successivamente utilizzati per aggiornare la richiesta di pagamento visualizzata all'utente. Vale a dire, questi bit di informazione sono:

* Il *methodData*: Una sequenza di [PaymentMethodData](https://w3c.github.io/payment-request/" \l "dom-paymentmethoddata) che rappresenta i [metodi di pagamento](https://w3c.github.io/payment-request/#dfn-payment-methods) che il sito supporta;
* I *details*: i dettagli della transazione, come un [PaymentDetailsInit](https://w3c.github.io/payment-request/" \l "dom-paymentdetailsinit). Ciò include il costo totale e facoltativamente un elenco di beni o servizi acquistati, beni materiali e opzioni di spedizione. Inoltre, può facoltativamente includere "modificatori" su come vengono effettuati i pagamenti. Ad esempio, "se paghi con una carta di credito di tipo X, incorre in una tassa di elaborazione di $ 3,00".
* Le *options*: un elenco di [PaymentOptions](https://w3c.github.io/payment-request/#dom-paymentoptions), in quanto il sito ha bisogno di consegnare il bene o il servizio (ad esempio, per i beni fisici, il commerciante di solito avrà bisogno di un indirizzo fisico da spedire; per i beni digitali, di solito un'e-mail è sufficiente).

Una volta che il [PaymentRequest](https://w3c.github.io/payment-request/" \l "dom-paymentrequest) è stato costruito, viene presentato all'utente finale tramite il [show()](https://w3c.github.io/payment-request/#dom-paymentrequest-show)metodo. I [show()](https://w3c.github.io/payment-request/#dom-paymentrequest-show) ritornano una promessa che, una volta che l'utente conferma la richiesta di pagamento, si traduce in una [PaymentResponse](https://w3c.github.io/payment-request/" \l "dom-paymentresponse).

- payment request: uno sviluppatore crea una [PaymentRequest](https://w3c.github.io/payment-request/" \l "dom-paymentrequest) per effettuare una richiesta di pagamento. Questo è in genere associato all'utente che avvia un processo di pagamento. La [PaymentRequest](https://w3c.github.io/payment-request/" \l "dom-paymentrequest) consente agli sviluppatori di scambiare informazioni con l' [user agent](https://w3c.github.io/payment-request/#dfn-user-agents) mentre l'utente sta fornendo dati in input. Poiché la visualizzazione simultanea di più interfacce [PaymentRequest](https://w3c.github.io/payment-request/#dom-paymentrequest) potrebbe confondere l'utente, questa specifica limita lo [user agent](https://w3c.github.io/payment-request/#dfn-user-agents) a visualizzarne uno alla volta tramite il metodo [show()](https://w3c.github.io/payment-request/#dom-paymentrequest-show).

- payment methodData: un [PaymentMethodData](https://w3c.github.io/payment-request/" \l "dom-paymentmethoddata) viene utilizzato per indicare un insieme di [metodi di pagamento](https://w3c.github.io/payment-request/#dfn-payment-methods) supportati e qualsiasi dato specifico del [metodo di pagamento](https://w3c.github.io/payment-request/#dfn-payment-methods) associato per tali metodi.

- payment detailsInit: Il [PaymentDetailsInit](https://w3c.github.io/payment-request/" \l "dom-paymentdetailsinit) viene utilizzato nella costruzione della richiesta di pagamento.

- payment options: Il [PaymentOptions](https://w3c.github.io/payment-request/" \l "dom-paymentoptions) viene passato al costruttore [PaymentRequest](https://w3c.github.io/payment-request/#dom-paymentrequest) e fornisce informazioni sulle opzioni desiderate per la richiesta di pagamento.

- payment response: un [PaymentResponse](https://w3c.github.io/payment-request/" \l "dom-paymentresponse) viene restituito quando un utente ha selezionato un metodo di pagamento e approvato una richiesta di pagamento.

2.1 The methodData argument

La sequenza methodData contiene [PaymentMethodData](https://w3c.github.io/payment-request/" \l "dom-paymentmethoddata) contenenti gli [identificativi del metodo di pagamento](https://w3c.github.io/payment-request/#dfn-payment-method-identifiers) per i [metodi di pagamento](https://w3c.github.io/payment-request/#dfn-payment-methods) accettati dal sito Web e qualsiasi dato specifico del [metodo di pagamento](https://w3c.github.io/payment-request/#dfn-payment-methods) associato .

EXAMPLE 1: The `methodData` argument

**const** methodData = [

{

supportedMethods: "basic-card",

data: {

supportedNetworks: ["visa", "mastercard"],

supportedTypes: ["debit", "credit"],

},

},

{

supportedMethods: "https://example.com/bobpay",

data: {

merchantIdentifier: "XXXX",

bobPaySpecificField: true,

},

},

];

2.2 L’argomento details

I details contengono informazioni sulla transazione che l'utente è invitato a completare.

EXAMPLE 2: The `details` argument

**const** details = {

id: "super-store-order-123-12312",

displayItems: [

{

label: "Sub-total",

amount: { currency: "USD", value: "55.00" },

},

{

label: "Sales Tax",

amount: { currency: "USD", value: "5.00" },

type: "tax"

},

],

total: {

label: "Total due",

*// The total is USD$65.00 here because we need to*

*// add shipping (below). The selected shipping*

*// costs USD$5.00.*

amount: { currency: "USD", value: "65.00" },

},

};

#### 2.2.1 Opzioni di spedizione

Qui vediamo un esempio di come aggiungere due opzioni di spedizione ai details.

EXAMPLE 3: Adding shipping options

**const** shippingOptions = [

{

id: "standard",

label: "🚛 Ground Shipping (2 days)",

amount: { currency: "USD", value: "5.00" },

selected: true,

},

{

id: "drone",

label: "🚀 Drone Express (2 hours)",

amount: { currency: "USD", value: "25.00" }

},

];

Object.assign(details, { shippingOptions });

#### 2.2.2 Modifiche condizionali alla richiesta di pagamento

Qui vediamo come aggiungere una tassa di elaborazione per l'utilizzo di una carta di credito. Si noti che richiede il ricalcolo del totale.

EXAMPLE 4: Modifying payment request based on card type

*// Credit card incurs a $3.00 processing fee.*

**const** creditCardFee = {

label: "Credit card processing fee",

amount: { currency: "USD", value: "3.00" },

};

*// Modifiers apply when the user chooses to pay with*

*// a credit card.*

**const** modifiers = [

{

additionalDisplayItems: [creditCardFee],

supportedMethods: "basic-card",

total: {

label: "Total due",

amount: { currency: "USD", value: "68.00" },

},

data: {

supportedTypes: "credit",

},

},

];

Object.assign(details, { modifiers });

2.3 L’argomento options

Options contiene informazioni che lo sviluppatore ha bisogno dall'utente per eseguire il pagamento.

EXAMPLE 5: The `options` argument

**const** options = {

requestPayerEmail: false,

requestPayerName: true,

requestPayerPhone: false,

requestShipping: true,

}

2.4 Costruire un PaymentRequest

Dopo aver raccolto tutti i bit di informazioni prerequisite, ora possiamo costruirne uno [PaymentRequest](https://w3c.github.io/payment-request/" \l "dom-paymentrequest) e richiedere che il browser lo presenti all'utente

EXAMPLE 6: Constructing a `PaymentRequest`

**async** **function** **doPaymentRequest**() {

**try** {

**const** request = **new** PaymentRequest(methodData, details, options);

*// See below for a detailed example of handling these events*

request.onshippingaddresschange = ev => ev.updateWith(details);

request.onshippingoptionchange = ev => ev.updateWith(details);

**const** response = **await** request.show();

**await** validateResponse(response);

} **catch** (err) {

*// AbortError, SecurityError*

console.error(err);

}

}

**async** **function** **validateResponse**(response) {

**try** {

**if** (**await** checkAllValuesAreGood(response)) {

**await** response.complete("success");

} **else** {

**await** response.complete("fail");

}

} **catch** (err) {

*// Something went wrong...*

**await** response.complete("fail");

}

}

doPaymentRequest();

### 2.5 Gestione degli eventi e aggiornamento della richiesta di pagamento

Prima che l'utente accetti di effettuare il pagamento, al sito viene data l'opportunità di aggiornare la richiesta di pagamento in risposta all'input dell'utente. Ciò può includere, ad esempio, la fornitura di ulteriori opzioni di spedizione (o la modifica dei costi), la rimozione di articoli che non possono essere spediti a un determinato indirizzo, ecc.

EXAMPLE 7: Registering event handlers

**const** request = **new** PaymentRequest(methodData, details, options);

*// Async update to details*

request.onshippingaddresschange = ev => {

ev.updateWith(checkShipping(request));

};

*// Sync update to the total*

request.onshippingoptionchange = ev => {

**const** shippingOption = shippingOptions.find(

option => option.id === request.id

);

**const** newTotal = {

currency: "USD",

label: "Total due",

value: calculateNewTotal(shippingOption),

};

ev.updateWith({ ...details, total: newTotal });

};

**async** **function** **checkShipping**(request) {

**try** {

**const** json = request.shippingAddress.toJSON();

**await** ensureCanShipTo(json);

**const** { shippingOptions, total } = **await** calculateShipping(json);

**return** { ...details, shippingOptions, total };

} **catch** (err) {

**return** { ...details, error: `Sorry! we can't ship to your address.` };

}

}

### 2.6 Segnalazione di errori a grana fine

Uno sviluppatore può utilizzare il [shippingAddressErrors](https://w3c.github.io/payment-request/" \l "dom-paymentdetailsupdate-shippingaddresserrors), membro del [PaymentDetailsUpdate](https://w3c.github.io/payment-request/" \l "dom-paymentdetailsupdate), per indicare che ci sono errori di convalida con attributi specifici di a [PaymentAddress](https://w3c.github.io/payment-request/" \l "dom-paymentaddress). Il [shippingAddressErrors](https://w3c.github.io/payment-request/" \l "dom-paymentdetailsupdate-shippingaddresserrors) è un [AddressErrorFields](https://w3c.github.io/payment-request/" \l "dom-addresserrorfields), i cui membri delimitano in modo specifico i campi di un [indirizzo fisico](https://w3c.github.io/payment-request/#dfn-physical-address) errati e forniscono anche messaggi di errore utili per la visualizzazione all'utente finale.

EXAMPLE 8

request.onshippingaddresschange = ev => {

ev.updateWith(validateAddress(request.shippingAddress));

};

**function** **validateAddress**(shippingAddress) {

**const** error = "Can't ship to this address.";

**const** shippingAddressErrors = {

cityError: "FarmVille is not a real place.",

postalCodeError: "Unknown postal code for your country.",

};

*// Empty shippingOptions implies that we can't ship*

*// to this address.*

**const** shippingOptions = [];

**return** { error, shippingAddressErrors, shippingOptions };

}

### 2.7 Postare la risposta di pagamento su un server

È previsto che i dati in a [PaymentResponse](https://w3c.github.io/payment-request/" \l "dom-paymentresponse) verranno reindirizzati a un server per l'elaborazione. Per rendere questo più semplice possibile, [PaymentResponse](https://w3c.github.io/payment-request/" \l "dom-paymentresponse) fornisce un [toJSON()](https://w3c.github.io/payment-request/" \l "dom-paymentresponse-tojson)metodo che serializza l'oggetto direttamente in JSON. Ciò rende banale il POST del JSON risultante su un server che utilizza l' [API Fetch](https://fetch.spec.whatwg.org/) :

EXAMPLE 9: POSTing with `fetch()`

**async** **function** **doPaymentRequest**() {

**const** payRequest = **new** PaymentRequest(methodData, details, options);

**const** payResponse = **await** payRequest.show();

**let** result = "";

**try** {

**const** httpResponse = **await** fetch("/process-payment", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: payResponse.toJSON(),

});

result = httpResponse.ok ? "success" : "fail";

} **catch** (err) {

console.error(err);

result = "fail";

}

**await** payResponse.complete(result);

}

doPaymentRequest();