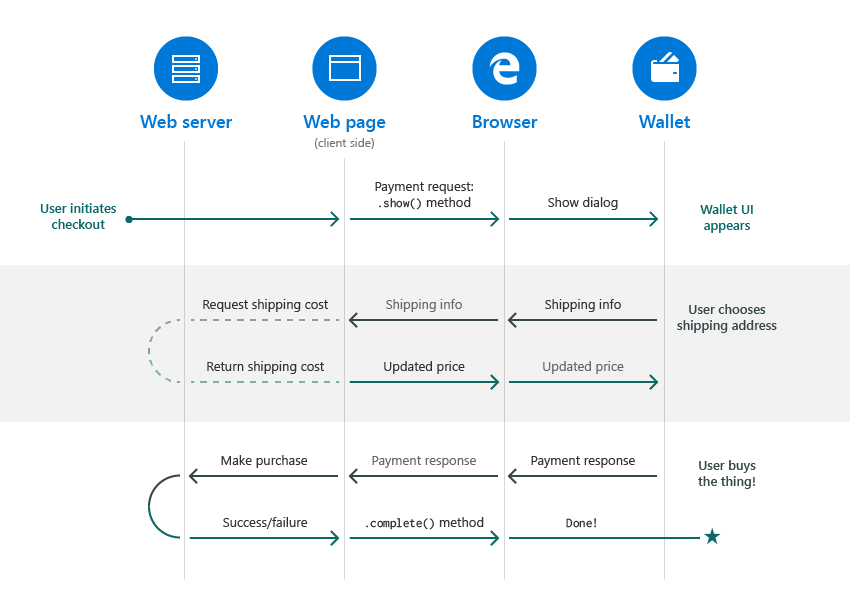
Microsoft e gli altri membri del [W3C Web Payments Working Group hanno](https://www.w3.org/Payments/WG/) progettato l'API con l'obiettivo di standardizzare le comunicazioni tra commercianti, browser e metodi di pagamento per fornire un'esperienza migliore agli utenti e un'unica API coerente per gli sviluppatori.

Con l'API di richiesta di pagamento, le informazioni di pagamento sono fornite dal portafoglio (una volta che l'utente ha concesso il consenso), anziché essere raccolte tramite un modulo di checkout nel sito web. Il browser media tutte le informazioni passate tra il portafoglio e il commerciante.



Come implementare l'API di richiesta di pagamento in un sito di esempio.

## Costruttore

Iniziamo con il costruttore: l'oggetto PaymentRequest è costruito passando i seguenti parametri:

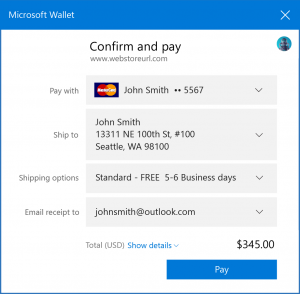
* **methodData**: una serie di dizionari che contiene gli [identificativi del metodo di pagamento](https://w3c.github.io/browser-payment-api/) e tutti i dati pertinenti; un identificativo del metodo di pagamento è una stringa che identifica un metodo di pagamento supportato
* **dettagli**: informazioni sulla transazione, come gli elementi pubblicitari in un ordine
* **opzioni**: informazioni aggiuntive che il Wallet potrebbe dover raccogliere

|  |
| --- |
| Var methodData = [ |
|  | { |
|  | supportedMethods : [ ' basic-card ' ], |
|  | dati : { |
|  | supportedNetworks : [ ' visa ' , ' mastercard ' , ' amex ' ], |
|  | supportedTypes : [ ' credit ' ] |
|  | } |
|  | } |
|  | ]; |
|  |  |
|  | dettagli var = { |
|  | displayItems : [ |
|  | { |
|  | etichetta : " Totale parziale " , |
|  | importo : {valuta : " USD " , valore : " 100,00 " } // US $ 100,00 |
|  | }, |
|  | { |
|  | etichetta : " Imposta sulle vendite " , |
|  | importo : {valuta : " USD " , valore : " 9,00 " } // US $ 9,00 |
|  | } |
|  | ], |
|  | totale : { |
|  | etichetta : " Totale dovuto " , |
|  | importo : {valuta : " USD " , valore : " 109,00 " } // US $ 109,00 |
|  | } |
|  | }; |
|  |  |
|  | var options = { |
|  | requestShipping : vero |
|  | }; |
|  |  |
|  | var payment = new PaymentRequest (methodData, dettagli, opzioni); |

Nello snippet precedente, stiamo consentendo agli utenti di pagare con qualsiasi carta di debito o di credito appartenente alle reti Visa, MasterCard o Amex. L'oggetto dettagli contiene l'importo totale parziale, l'imposta sulle vendite e il totale dovuto. Questi dettagli verranno mostrati all'utente nel portafoglio. Tieni presente che l'API non aggiunge elementi o calcola l'imposta sulle vendite: spetta al commerciante fornire le informazioni corrette. In questo esempio, stiamo vendendo un bene fisico, quindi chiediamo al portafoglio di ritirare l'indirizzo di spedizione del cliente.

## Visualizzazione dell'interfaccia utente, elaborazione del pagamento e visualizzazione dei risultati

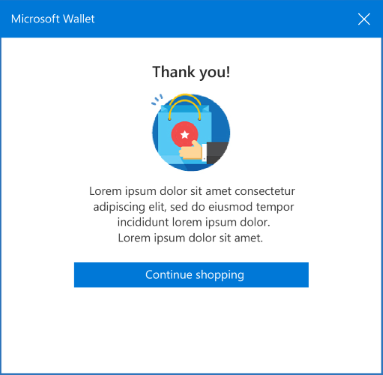
Una volta creato l'oggetto PaymentRequest, è possibile attivare il browser per visualizzare il wallet con request.show ().

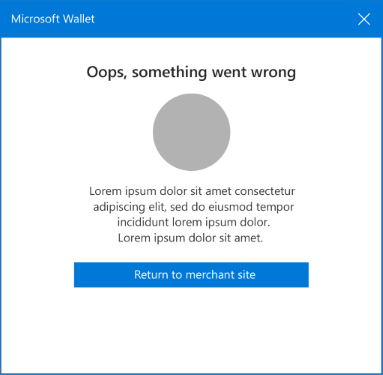


I clienti possono quindi selezionare le informazioni di pagamento, l'indirizzo di spedizione e altri campi appropriati, quindi fare clic su Paga quando è pronto. A questo punto, gli utenti dovranno verificare la loro identità. In caso di esito positivo, ciò soddisferà la promessa request.show () e restituirà al sito Web tutte le informazioni che il cliente ha fornito al Portafoglio. Per il metodo di pagamento con carta di base, l'oggetto risultato conterrà il nome del titolare della carta, il numero della carta, il mese di scadenza e altri campi pertinenti. Il commerciante può quindi utilizzare queste informazioni per elaborare la transazione sul back-end.

Dopo che la risposta è tornata dal server, è possibile utilizzare result.complete ('successo') per visualizzare la schermata di successo nel Portafoglio e result.complete ('fail') per indicare una transazione fallita.

|  |
| --- |
| // Mostra l'interfaccia utente nativa |
|  | pagamento . mostra () |
|  | // Quando la promessa è soddisfatta, passa i risultati al tuo server per l'elaborazione |
|  | . try ( risultato => { |
|  | processo di ritorno (risultato). poi ( risposta => { |
|  | if ( response . status === 200 ) { |
|  | // Mostra che la transazione ha avuto successo nell'interfaccia utente |
|  |  |
|  | restituire il risultato . completo ( ' successo ' ); |
|  | } else { |
|  | // Mostra nell'interfaccia utente nativa che la transazione ha avuto esito negativo |
|  | restituire il risultato . completo ( ' fallire ' ); |
|  | } |
|  | }). catch (( err ) => { |
|  | console . Errore ( ' richiesta respinta User ' , err . messaggio ) |
|  | }); |
|  | }); |





## Ascoltando gli eventi

Il prezzo potrebbe cambiare in base all'indirizzo di spedizione e alle opzioni di spedizione selezionate dal cliente. È possibile ascoltare tali modifiche con gli eventi shippingaddresschange e shippingoptionchange per ricalcolare di conseguenza i prezzi.

|  |
| --- |
| pagamento . addEventListener ( " shippingaddresschange " , function ( changeEvent ) { |
|  | // Elabora la modifica dell'indirizzo di spedizione |
|  | }); |
|  |  |
|  | pagamento . addEventListener ( " shippingoptionchange " , function ( changeEvent ) { |
|  | // Modifica delle opzioni di spedizione del processo (ad esempio "spedizione in giornata") |
|  | }); |

## Rilevamento di funzionalità

I siti possono essere rilevati per l'API di richiesta di pagamento, inoltrare l'utente a un'esperienza legacy, basata su moduli, se non è disponibile.

|  |  |
| --- | --- |
|  | // Inoltra utente alla verifica basata su modulo se PaymentRequest non è disponibile |
|  | se ( ! window . PaymentRequest ) { |
|  | finestra . posizione . href = ' / form-based-checkout ' ; |
|  | ritorno ; |
|  | } |

Ecco un esempio di implementazione minima di questo codice:

|  |  |
| --- | --- |
|  | function startPaymentRequestCheckout () { |
|  | var methodData = [ |
|  | { |
|  | supportedMethods : [ ' basic-card ' ], |
|  | dati : { |
|  | supportedNetworks : [ ' visa ' , ' mastercard ' , ' amex ' ], |
|  | supportedTypes : [ ' credit ' ] |
|  | } |
|  | } |
|  | ]; |
|  |  |
|  | dettagli var = { |
|  | displayItems : [ |
|  | { |
|  | etichetta : " Totale parziale " , |
|  | importo : {valuta : " USD " , valore : " 100,00 " }, // US $ 100,00 |
|  | }, |
|  | { |
|  | etichetta : " Imposta sulle vendite " , |
|  | importo : {valuta : " USD " , valore : " 9,00 " }, // US $ 9,00 |
|  | } |
|  | ], |
|  | totale : { |
|  | etichetta : " Totale dovuto " , |
|  | importo : {valuta : " USD " , valore : " 109,00 " }, // US $ 109,00 |
|  | } |
|  | }; |
|  |  |
|  | var options = { |
|  | requestShipping : vero |
|  | }; |
|  |  |
|  | var request = new PaymentRequest (methodData, dettagli, opzioni); |
|  |  |
|  | // Mostra l'interfaccia utente nativa |
|  | richiesta . mostra () |
|  | // Quando la promessa è soddisfatta, passa i risultati al tuo server per l'elaborazione |
|  | . allora ( risultato => { |
|  | processo di ritorno (risultato). poi ( risposta => { |
|  | if ( response . status === 200 ) { |
|  | // Mostra che la transazione ha avuto successo nell'interfaccia utente |
|  | restituire il risultato . completo ( ' successo ' ); |
|  | } else { |
|  | // Mostra nell'interfaccia utente nativa che la transazione ha avuto esito negativo |
|  | restituire il risultato . completo ( ' fallire ' ); |
|  | } |
|  | }) |
|  | }); |
|  |  |
|  | richiesta . addEventListener ( " shippingaddresschange " , function ( changeEvent ) { |
|  | // Elabora la modifica dell'indirizzo di spedizione |
|  | }); |
|  |  |
|  | richiesta . addEventListener ( " onshippingoptionchange " , function ( changeEvent ) { |
|  | // Modifica delle opzioni di spedizione del processo (ad esempio "spedizione in giornata") |
|  | }); |
|  | } |
|  | documento . getElementById ( ' #checkout ' ). addEventListener ( ' click ' , startPaymentRequestCheckout) |

# Conclusioni

L'API di richiesta di pagamento fornisce un potente strumento per i commercianti per migliorare la conversione del checkout sul Web e per offrire ai clienti un'esperienza di acquisto più piacevole e conveniente. Questa API è un ottimo esempio della potenza e della flessibilità della piattaforma web, ed è sulla strada per un'ampia interoperabilità, con Chrome per Android che supporta l'API a partire da Chrome 54.