



# HTML5 APIs

- YOU'VE NEVER HEARD OF -

- DREW MCLELLAN -

- PHP YORKSHIRE 2017 -



[flickr.com/photos/85520404@N03/9535499657](https://www.flickr.com/photos/85520404@N03/9535499657)

# Hello!

**HTML5... APIs?**

**Front end**

**Back end**

**HTML5 APIS**



**The web is  
changing rapidly**

# Ch-ch-ch-changes

- ▶ Screens are getting smaller, and bigger, and rounder, and wider, and taller, and wearable.
- ▶ Pointing devices are becoming meatier.
- ▶ We have access to many more hardware features.
- ▶ Web browsers are on the move.
- ▶ Power consumption has become a concern.

# Things we can do

- ▶ Access device sensors like the gyroscope, compass, light meter, GPS, camera, microphone.
- ▶ Control device outputs like the speaker, vibration motor and screen.
- ▶ Establish more app-like control of the environment our code is running in.

# Page Visibility

<https://www.w3.org/TR/page-visibility/>



# Page Visibility

Enables us to programmatically determine if a page is currently visible.

A page might be hidden if the window is minimised, if the page is in a background tab, or if the lock screen is shown.

(Plus a few transitional states.)

# Testing for visibility

The visibility of the document can be tested. You can add an event listener to be informed of when the visibility changes.

```
// Is the document visible?  
var visible = !document.hidden;
```

```
// Listen for changes  
document.addEventListener("visibilitychange", function(){  
  console.log('Visibility changed!');  
});
```

# When is it useful?

Stopping 'expensive' operations like animation.

Ensuring that the user sees important information like flash notifications or alerts.

Pausing media, where appropriate.



# Browser support

## Page Visibility - REC

JavaScript API for determining whether a document is visible on the display

Global 91.74%  
 unprefixed: 76.42%  
 U.K. 95.18%  
 unprefixed: 88.94%

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			51						
			52					4.4 	
		49	53	9.1		9.3		4.4.4 	
11	14	50	54	10	41	10	all	53	54
		51	55	TP	42				
		52	56		43				
		53	57						

Notes Known issues (1) Resources (8) Feedback

# Use it today!

There are lots of small cases where using this simple API will provide a better experience for your users.

# Device Orientation

<https://www.w3.org/TR/orientation-event/>

# Device Orientation

DOM events that provide information about the physical *orientation* and *motion* of a device.

This is mostly useful for mobile phones and tablets.

Enables us to write code that detects physical movements like rotation around a point and rate of rotation.

Not to be confused with screen orientation (portrait / landscape).

# Device Orientation

The API provides browser DOM events that we can attach listeners to.

The events are fired rapidly, so might need to be throttled (like we do with window scroll events).

```
// Test for support
if ('ondeviceorientation' in window) {
  // we have support for 'deviceorientation' events
};

// Listen for orientation changes
document.addEventListener("deviceorientation",
  function(event){
    console.log(event);
  });
```



# Device Orientation

Orientation values are reported as alpha, beta and gamma properties.

These are a series of rotations from a local coordinate frame.

They can be used to calculate compass headings with some crazy mathematics... which is all very usefully in the spec.

```
// Access orientation properties
function(event){
  var alpha = event.alpha;
  var beta  = event.beta;
  var gamma = event.gamma;
};

// A device flat on a horizontal surface
{
  alpha: 90,
  beta  : 0,
  gamma: 0
}

var compass_heading = (360 - alpha);
```

# Device Orientation

Orientation is expressed in a difference between the Earth frame and the device frame.

Here they are aligned.

This horrible image is from the spec, sorry.

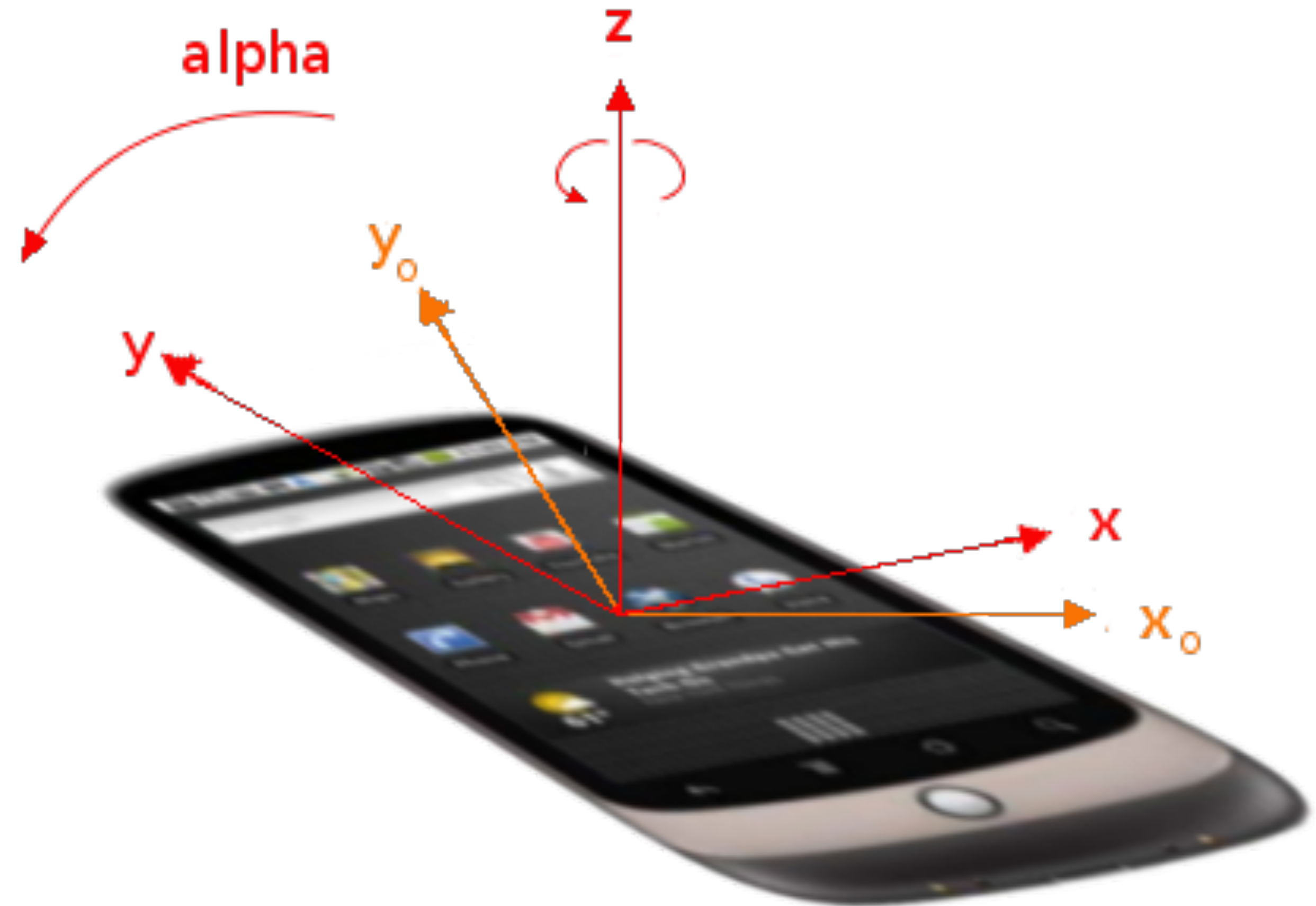


# Device Orientation

This marvellous work of art is showing the device rotated around the Z axis.

The value of Z remains the same, and X and Y change.

This results in a change to the alpha value.



# Device Orientation

The beta value changes with rotation around the X axis.



# Device Orientation

The gamma value changes with rotation around the Y axis.

You're probably best to just try it. It makes more sense in action.



# When is it useful?

Good for creating 'physical world' interactions.

It's the same sensors that the Facebook mobile app uses for displaying panoramas.

Could be used for game control.

Makes physical gestures possible (e.g. shake to undo).

Align a map to match reality...

# Browser support

## DeviceOrientation & DeviceMotion events - CR

Global

2.07% + 89.34% = 91.41%

API for detecting orientation and motion events from the device running the browser.

Current aligned Usage relative Date relative [Show all](#)

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49					4.4	
		51	55			9.3		4.4.4	
<sup>1</sup> 11	14	52	56	10	43	10.2	all	53	56
	15	53	57	10.1	44				
		54	58	TP	45				
		55	59						

Notes

Known issues (2)

Resources (10)

Feedback

Partial support refers to the lack of compassneeds calibration event. Partial support also refers to the lack of devicemotion event support for Chrome 30- and Opera. Opera Mobile 14 lost the devicemotion event support. Firefox 3.6, 4 and 5 support the non-standard [MozOrientation](#) event.

# Browser support

Support for orientation is pretty wide in mobile browsers.

Missing support is often for the ever so exciting `compassneeds Calibration event`.



# Start experimenting!

Browser support is not too bad.

Could be interesting to use for prototypes and small projects.

There might be ways motion detection could be useful for applications other than just updating a view on the screen.

Lots of potential uses in mapping, gaming and health applications.

# Battery Status

<http://www.w3.org/TR/battery-status/>

# Battery Status

Enables us to programmatically monitor the status of the device's battery.

We can see if the battery is charging or discharging, how long it will take to charge or discharge, and what the current battery level is.

The interface is Promise-based.

# Battery status

The navigator object exposes a `getBattery` promise.

If the device has multiple batteries, the browser's `BatteryManager` interface exposes a unified view.

Battery level is between 0 and 1.

```
navigator.getBattery().then(function(battery) {  
  console.log(battery.level);  
  
  // Listen for updates  
  battery.addEventListener('levelchange', function(){  
    console.log(this.level);  
  });  
});
```

# Battery status

By checking if the battery is charging or discharging, we can then get the time left until that action completes.

If the battery is charging and we ask for the discharge time, it will be positive infinity which is useful to no one.

The charging and discharging times are in seconds.

```
navigator.getBattery().then(function(battery) {  
  if (battery.charging) {  
    console.log('%d mins until full',  
      Math.floor(battery.chargingTime/60));  
  } else {  
    console.log('%d mins until empty',  
      Math.floor(battery.dischargingTime/60));  
  }  
});
```

# When is it useful?

If a user's battery is low, you might scale back on any battery-intensive actions.

You might want to save the user's progress to the server or local storage if the battery is critically low.

You might perform network polls frequently when charging, but infrequently when discharging.

# Browser support

## Battery Status API - CR

Method to provide information about the battery status of the hosting device.

Global 62.96% + 9.75% = 72.72%  
unprefixed: 62.96% + 9.7% = 72.66%

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49					4.4	
		51	55			9.3		4.4.4	
11	14	52	56	10	43	10.2	all	53	56
	15	53	57	10.1	44				
		54	58	TP	45				
		55	59						

Notes Known issues (0) Resources (5) Feedback

MS Edge status: Under Consideration

# Use it when available

If the battery status is available, you can make use of it.

If not, just carry on with whatever you were doing before. Those with supporting devices will get the benefit, and that's all you can do.

Not just phones - laptops too!



# Vibration

<https://www.w3.org/TR/vibration/>

# Vibration

Gives us access to the vibration mechanism of the device.

That's usually a phone or perhaps a tablet.

Designed for simple tactile feedback only, nothing fancy.

# Vibration

Vibration time is set in milliseconds.

When an array is given, the even items are vibrations, the odd items are pauses. This enables more complex patterns.

Any ongoing vibration can be cancelled.

```
// Vibrate for 1000 ms  
navigator.vibrate(1000);
```

```
// Vibration to a pattern  
navigator.vibrate([150, 50, 150]);
```

```
// Cancel any vibrations  
navigator.vibrate(0);
```

# When is it useful?

**Providing tactile feedback for important actions.**

**Could be used as a rumble in games.**

**Create a cool Morse code device?**

# Browser support

## Vibration API - REC

Global 74.91%  
unprefixed: 74.86%

Method to access the vibration mechanism of the hosting device.

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49					4.4	
		51	55			9.3		4.4.4	
11	14	52	56	10	43	10.2	all	53	56
	15	53	57	10.1	44				
		54	58	TP	45				
		55	59						

Notes Known issues (0) Resources (9) Feedback

MS Edge status: Under Consideration

# Use it!

Works in most mobile browsers other than iOS Safari.

Should be safe to use as an extra where it is supported.

Don't design interactions that rely on it, and maybe check battery status too!

# Web Notifications

<http://www.w3.org/TR/notifications/>

# Web Notifications

Enable us to issue an alert to the user outside the context of the web page.

This is normally through the operating system's standard alerts mechanism.

Users must grant permission before notifications can be shown.



# Web Notifications

We can test for the Notification property of the window object to see if we have support.

Before sending a notification, we need to request permission. This call returns either 'granted', 'denied' or 'default'.

We can only send a notification when the result is 'granted'.

```
if ('Notification' in window) {  
    // Notifications are supported!  
}
```

```
Notification.requestPermission(function(status) {  
    if (status == 'granted') {  
        // We have permission to notify!  
    };  
});
```

# Web Notifications

The Notification constructor takes a title, and then an object containing options.

Basic options are 'body' for the message and 'icon' for an icon to show with the notification.

```
var notification = new Notification(  
    'Your life is in danger',  
    {  
        body: 'You forgot to take your pills',  
        icon: 'skull-and-crossbones.png'  
    }  
);
```

# Web Notifications

The 'tag' option acts like an ID for the notification.

If there are multiple instances of your code running (e.g. two browser windows) the tag prevents the notification being duplicated.

It can also be used to address the notification to cancel it.

```
var notification = new Notification(  
  'Your life is in danger', {  
    body: 'You forgot to take your pills',  
    icon: 'skull-and-crossbones.png',  
    tag: 'pills-warning',  
    lang: 'en-US',  
    dir: 'ltr'  
  })  
);
```

# What are they good for?

Notifying the user of background task completion, e.g. encoding has finished, upload has completed.

Notifying of incoming activity, e.g. a message has been received, a user has logged in.

# Browser support

## Web Notifications - LS

Global 38.52% + 5.4% = 43.92%  
 unprefixed: 38.52%

Method of alerting the user outside of a web page by displaying notifications (that do not require interaction by the user).

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49					4.4	
		51	55			9.3		4.4.4	
11	14	52	56	10	43	10.2	all	53	56
	15	53	57	10.1	44				
		54	58	TP	45				
		55	59						

Notes Known issues (3) Resources (10) Feedback

No notes

# Use them!

Pretty great support on desktop.

Judge carefully when to ask permission to display notifications. Do it before you need to send, but not before the user trusts you or they'll decline.

# Web MIDI

<https://www.w3.org/TR/webmidi/>

**MIDI?!**



# Musical Instrument Digital Interface

MIDI is a very well established protocol for sending event messages about musical notes, control signals and clock signals.

It's used by musical keyboards, synths, drum machines, digital control surfaces, theatre lighting and sound systems, and most importantly...



**Keytars!**

# Web MIDI

MIDI sends note-on and note-off events (with pitch and velocity), and change events for any number of other controls.

It's basically a well defined protocol for event based input and output for physical buttons and switches.

Which makes it quite exciting.

# Web MIDI

We first need to request access to MIDI devices.

This returns a promise, with a success and failure callback.

Code sample references work by Stuart Memo on [sitepoint.com](https://www.sitepoint.com)

```
if (navigator.requestMIDIAccess) {  
  // We have MIDI support!  
}
```

```
if (navigator.requestMIDIAccess) {  
  navigator.requestMIDIAccess()  
    .then(success, failure);  
}
```

# Web MIDI

If we have access to MIDI, our success callback gets a `MIDIAccess` object.

From this we can get all the different MIDI inputs we have access to, using an iterator.

This code loops through the inputs adding an event listener for the `onmidimessage` event.

```
function failure() {  
  // MIDI access denied :(  
}  
  
function success(midi) {  
  var inputs = midi.inputs.values();  
  
  for (var input = inputs.next();  
       input && !input.done;  
       input = inputs.next()) {  
  
    input.value.onmidimessage = messageReceived;  
  }  
}
```

# Web MIDI

Now we can receive MIDI messages! They look weird.

The format is event code, note number, velocity.

144 is note on.

128 is note off.

```
function messageReceived(message) {  
  console.log(message.data);  
}
```

```
[144, 61, 95]
```

```
[128, 61, 0]
```

```
[eventCode, note, velocity]
```



**Demo!**

# When is it useful?

Simple integration between physical devices and the browser.

There are *lots* of MIDI devices and most are very robust. Designed to be hit with sticks etc.

Perfect for children's games, controls for those with disabilities, kiosk applications, keytars.



# When is it useful?

You can also play notes out, enabling you to play instruments, control theatre lighting, sound effects, video playback.

It will not give you any musical talent. Sorry.

# Browser support

## Web MIDI API 📄 - WD

Global

56.66%

The Web MIDI API specification defines a means for web developers to enumerate, manipulate and access MIDI devices

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49					4.4	
		51	55			9.3		4.4.4	
11	14	52	56	10	43	10.2	all	53	56
	15	53	57	10.1	44				
		54	58	TP	45				
		55	59						

Notes

Known issues (0)

Resources (5)

Feedback

MS Edge status: Not currently planned

# Play with it

Could be fun for hack projects, and controlled environments.

Might not quite be ready for the open web until all computers ship with keytars.

Keytars!

# Payment Request

<http://www.w3.org/TR/payment-request/>

# Payment Request

Enables us to collect payment method (card number, token), shipping, and contact information for a transaction directly from the browser.

Saves the user needing to re-enter common personal information, especially on mobile.

Provides a neat 'saved card' UI without the site needing to save anything.

# What it isn't

The Payment Request API isn't a payment gateway. It doesn't take payments.

It doesn't integrate with payment gateways either.

It simply gathers the payment details from the user and hands them to your application.

# Payment Request

We can test for the availability by looking for `PaymentRequest` in `window`.

The request starts with a new `PaymentRequest` object, which takes arguments for supported payment methods, details of the transaction, and some options.

```
if (window.PaymentRequest) {  
    // Payments are supported!  
}
```

```
var pr = new PaymentRequest( methodData,  
                             transactionDetails,  
                             options );
```

# Payment methods

This is used to describe which payment methods you can accept. This will depend on your payment provider.

At the moment, Payment Request works with standard card payments and AndroidPay.

The data property contains any information specific to that payment method.

```
var methodData = [  
  {  
    supportedMethods: ['visa', 'mastercard', 'amex'],  
  },  
  {  
    supportedMethods: ['https://andriod.pay/pay'],  
    data: {  
      marchantID: '1234'  
    }  
  }  
];
```



# Transaction details

Here we provide the specifics of the transaction; how much to charge, the currency and so on.

We can also provide line items, which are displayed for the user.

```
var transactionDetails = {
  total: {
    label: 'Total',
    amount: { currency: 'GBP', value: '99.00' }
  },
  displayItems: [
    {
      label: 'Subtotal',
      amount: { currency: 'GBP', value: '99.00' }
    }
  ],
};
```

# Options

Lastly, we can set a small number of boolean options.

```
var options = {  
    requestShipping: true,  
    requestPayerEmail: true,  
    requestPayerPhone: false,  
    requestPayerName: true,  
};
```

# Payment Request

The `show()` method shows the browser payment UI and kicks off the process from the user's point of view.

It returns a promise, with a callback that contains the payment response.

We call the `complete()` method to let the browser know the result so it can update the UI for the user.

```
var pr = new PaymentRequest(methodData,  
                             transactionDetails,  
                             options);  
  
pr.show().then(function(paymentResponse) {  
    // send the card details to your gateway  
    // and then:  
    paymentResponse.complete('success');  
});
```

# Shipping events

An event is fired when the user changes their shipping address.

We can add a listener for this, and update the transaction details at that point if required.

This helps with shipping costs that change based on location.


```
pr.addListener('shippingaddresschange', function(e){  
  
  transactionDetails.displayItems.push({  
    label: 'Shipping',  
    amount: { currency: 'GBP', value: '10.00' }  
  });  
  
  transactionDetails.total.amount.value = '109.00';  
  
  e.updateWith(Promise.resolve(transactionDetails));  
});
```

Pay now

### Your payment

Order summary	Subtotal	£99.00	▶
	<b>Total</b>	<b>GBP £99.00</b>	▶

Delivery address ▶

Payment Visa ••••4242  ▶  
D MCLELLAN

Contact info Drew McLellan ▶



Pay Cancel



https://a91d247a.ngrok.io/pay x



Secure https://a91d247a.ngrok.io/payment.html



Pay now

← Delivery address

**Drew McLellan**

edgeofmyseat.com, Spike Island, 133 Cumberland Road, Bristol, BS1 6UX, ..

Add an address

Cancel



https://a91d247a.ngrok.io/pay x



Secure https://a91d247a.ngrok.io/payment.html




Pay now

### Your payment

Order summary	Subtotal	£99.00	
	Shipping	£10.00	▶
	<b>Total</b>	<b>GBP £109.00</b>	

Delivery address Drew McLellan  
edgeofmyseat.com, Spike Island, 133 Cum... ▶

Payment Visa •••• 4242  ▶  
D MCLELLAN

Contact info Drew McLellan ▶



Pay

Cancel



https://a91d247a.ngrok.io/pay x



Secure https://a91d247a.ngrok.io/payment.html



Pay now

← Enter the CVC for Visa ••••4242

Once you've confirm, your card details will be shared with this site



Confirm

Cancel





# When is it useful?

**Provides a very native feeling payment UI for mobile, where filling out forms can be very tedious.**

**Enables you to offer those users the convenience of using a saved card, without saving cards.**

# Browser support

## Payment Request API 📄 - WD

Global 26.57%  
U.K. 16.53%

Payment Request is a new API for the open web that makes checkout flows easier, faster and consistent on shopping sites.

Current aligned Usage relative Date relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			1 55				9.3	4.4	
		51	1 56	4 10	1 43	4 10.2		4.4.4	
11	2 14	52	1 57	4 10.1	1 44	4 10.3	all	56	57
	3 15	53	1 58	4 TP	1 45				
		54	1 59		1 46				
		55	1 60						

Notes Known issues (0) Resources (9) Feedback

# Start experimenting!

Payment Request is very new and support is experimental. The API is well designed and shouldn't change to drastically.

It will provide a major advantage for mobile users, so it's worth knowing. Kick the tyres and give feedback.

**Phew.**

# HTML5 APIs

Page Visibility

Device Orientation

Battery Status

Vibration

Web Notifications

Web MIDI

Payment Request

Ambient Light

Geolocation

Web Audio

Web Share

Screen Orientation

# HTML5 APIs

Clipboard

Speech synthesis

Speech detection

Media capture streams

Proximity

Network information

File & File System

Drag and drop

Fullscreen

Web workers



**Thanks!**

@drewm

[speakerdeck.com/drewm](https://speakerdeck.com/drewm)