
Pressure.js

downloads 29k downloads 29k downloads 29k

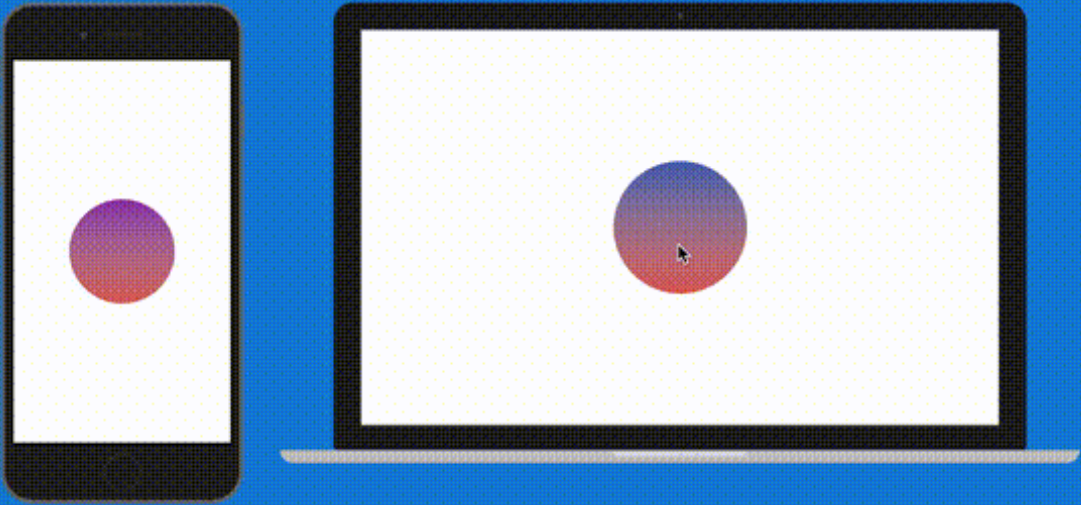
Pressure.js Examples Documentation Press View on GitHub

Pressure.js

current v2.0.0

Star 1,183 Tweet

Pressure is a JavaScript library for handling both Force Touch and 3D Touch on the web, bundled under one library with a simple API that makes working with them painless.



Pressure is a JavaScript library for handling both Force Touch and 3D Touch on the web, bundled under one library with a simple API that makes working with them painless.

Head over to the documentation for installation instructions, supported devices, and more details on pressure.js.

Install

Download pressure.min.js or pressure.js files from GitHub or install with npm or bower ##### npm

```
1 npm install pressure --save
```

bower

```
1 bower install pressure --save
```

Setup

Use pressure in the global space:

```
1 Pressure.set('#id-name', {
2   change: function(force){
3     this.innerHTML = force;
4   }
5 });
```

OR use it with browserify or CommonJS like setups:

```
1 var Pressure = require('pressure');
2
3 Pressure.set('#id-name', {
4   change: function(force){
5     this.innerHTML = force;
6   }
7 });
```

Usage

NOTE: the “this” keyword in each of the callback methods will be the element itself that has force applied to it

```
1 Pressure.set('#element', {
2   start: function(event){
3     // this is called on force start
4   },
5   end: function(){
6     // this is called on force end
7   },
8   startDeepPress: function(event){
9     // this is called on "force click" / "deep press", aka once the
10      force is greater than 0.5
11   },
12   endDeepPress: function(){
13     // this is called when the "force click" / "deep press" end
14   },
15   change: function(force, event){
16     // this is called every time there is a change in pressure
17     // force will always be a value from 0 to 1 on mobile and desktop
18   },
19   unsupported: function(){
```

```
19 // NOTE: this is only called if the polyfill option is disabled!
20 // this is called once there is a touch on the element and the
    device or browser does not support Force or 3D touch
21 }
22 });
```

jQuery Usage

NOTE: the “this” keyword in each of the callback methods will be the element itself that has force applied to it

```
1 $('#element').pressure({
2   start: function(event){
3     // this is called on force start
4   },
5   end: function(){
6     // this is called on force end
7   },
8   startDeepPress: function(event){
9     // this is called on "force click" / "deep press", aka once the
        force is greater than 0.5
10  },
11  endDeepPress: function(){
12    // this is called when the "force click" / "deep press" end
13  },
14  change: function(force, event){
15    // this is called every time there is a change in pressure
16    // force will always be a value from 0 to 1 on mobile and desktop
17  },
18  unsupported: function(){
19    // NOTE: this is only called if the polyfill option is disabled!
20    // this is called once there is a touch on the element and the
        device or browser does not support Force or 3D touch
21  }
22 });
```

Options

With Pressure, the third parameter is an optional object of options that can be passed in.

Polyfill Support

Using the “polyfill” keyword, you can disable polyfill support for the element. The polyfill is enabled by default and is useful if the device or browser does not support pressure, it will fall back to using

time. For example instead of force from 0 to 1, it counts up from 0 to 1 over the course of one second, as long as you are holding the element. Try some of the examples on the main page on a device that does not support pressure and see for yourself how it works.

```
1 Pressure.set('#example', {
2   change: function(force, event){
3     this.innerHTML = force;
4   },
5   unsupported: function(){
6     alert("Oh no, this device does not support pressure.");
7   }
8 }, {polyfill: false});
```

Polyfill Speed Up

If you are using the polyfill (on by default), you can see the “polyfillSpeedUp” speed to determine how fast the polyfill takes to go from 0 to 1. The value is an integer in milliseconds and the default is 1000 (1 second).

```
1 Pressure.set('#example', {
2   change: function(force, event){
3     this.innerHTML = force;
4   }
5 }, {polyfillSpeedUp: 5000});
6 // takes 5 seconds to go from a force value of 0 to 1
7 // only on devices that do not support pressure
```

Polyfill Speed Down

If you are using the polyfill (on by default), you can see the “polyfillSpeedDown” speed to determine how fast the polyfill takes to go from 1 to 0 when you let go. The value is an integer in milliseconds and the default is 0 (aka off).

```
1 Pressure.set('#example', {
2   change: function(force, event){
3     this.innerHTML = force;
4   }
5 }, {polyfillSpeedDown: 2000});
6 // takes 2 seconds to go from a force value of 1 to 0
7 // only on devices that do not support pressure
```

Only run on Force Touch trackpads (mouse)

Set the option `only` to the type you want it to run on 'mouse', 'touch', or 'pointer'. The names are the types of events that pressure will respond to.

```
1 Pressure.set('#example',{
2   change: function(force, event){
3     console.log(force);
4   },
5 }, {only: 'mouse'});
```

Only run on 3D Touch (touch)

```
1 Pressure.set('#example',{
2   change: function(force, event){
3     console.log(force);
4   },
5 }, {only: 'touch'});
```

Only run on Pointer Supported Devices (pointer)

```
1 Pressure.set('#example',{
2   change: function(force, event){
3     console.log(force);
4   },
5 }, {only: 'pointer'});
```

Change the `preventSelect` option

The `preventDefault` option is “true” by default and it prevents the default actions that happen on 3D “peel and pop” actions and the Force “define word” actions as well as other defaults. To allow the defaults to run set `preventDefault` to “false”

```
1 Pressure.set('#example',{
2   change: function(force, event){
3     console.log(force);
4   },
5 }, {preventSelect: false});
```

Helpers

Config

You can use `Pressure.config()` to set default configurations for site wide setup. All of the configurations are the same as the options listed above.

Heads Up: If you have a config set, you can always override the config on individual Pressure elements by passing in any of the options listed above to a specific Pressure block.

When using the jQuery Pressure library, use `$.pressureConfig()` rather than `Pressure.map()`

```
1 // These are the default configs set by Pressure
2 Pressure.config({
3   polyfill: true,
4   polyfillSpeedUp: 1000,
5   polyfillSpeedDown: 0,
6   preventDefault: true,
7   only: null
8 });
```

Map

You can use `Pressure.map()` to map a value from one range of values to another. It takes 5 params: `Pressure.map(inputValue, inputValueMin, inputValueMax, mapToMin, mapToMax)`; Here is a good write up on how this works in the Processing framework: Map Function.

When using the jQuery Pressure library, use `$.pressureMap()` rather than `Pressure.map()`

```
1 Pressure.set('#element', {
2   change: function(force, event){
3     // this takes the force, given that the force can range from 0 to
4     // 1, and maps that force value on a 100 to 200 range
5     this.style.width = Pressure.map(force, 0, 1, 100, 200);
6   }
7 });
```