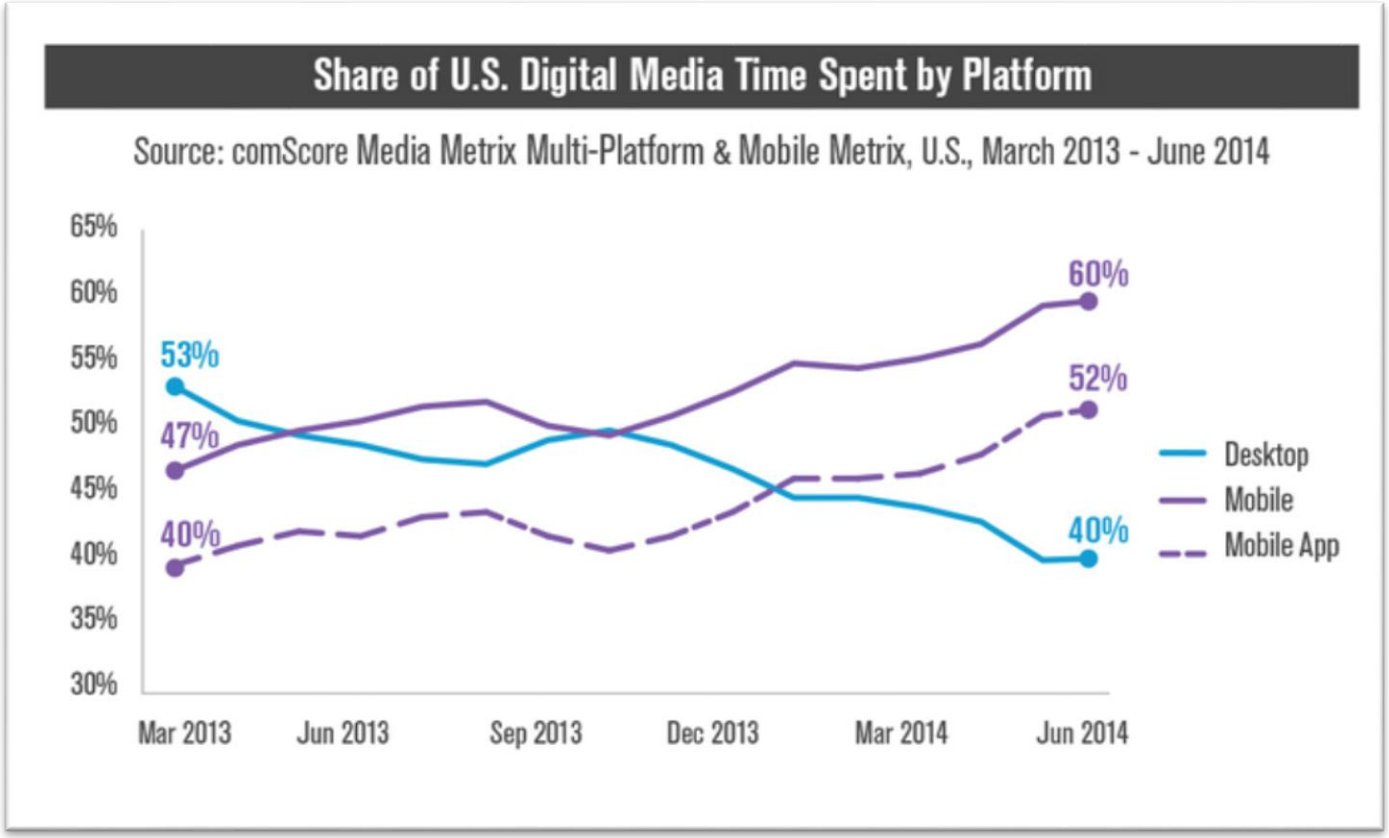
Beyond Mouse and Keyboard

Tilt-and-Tap

Introduction

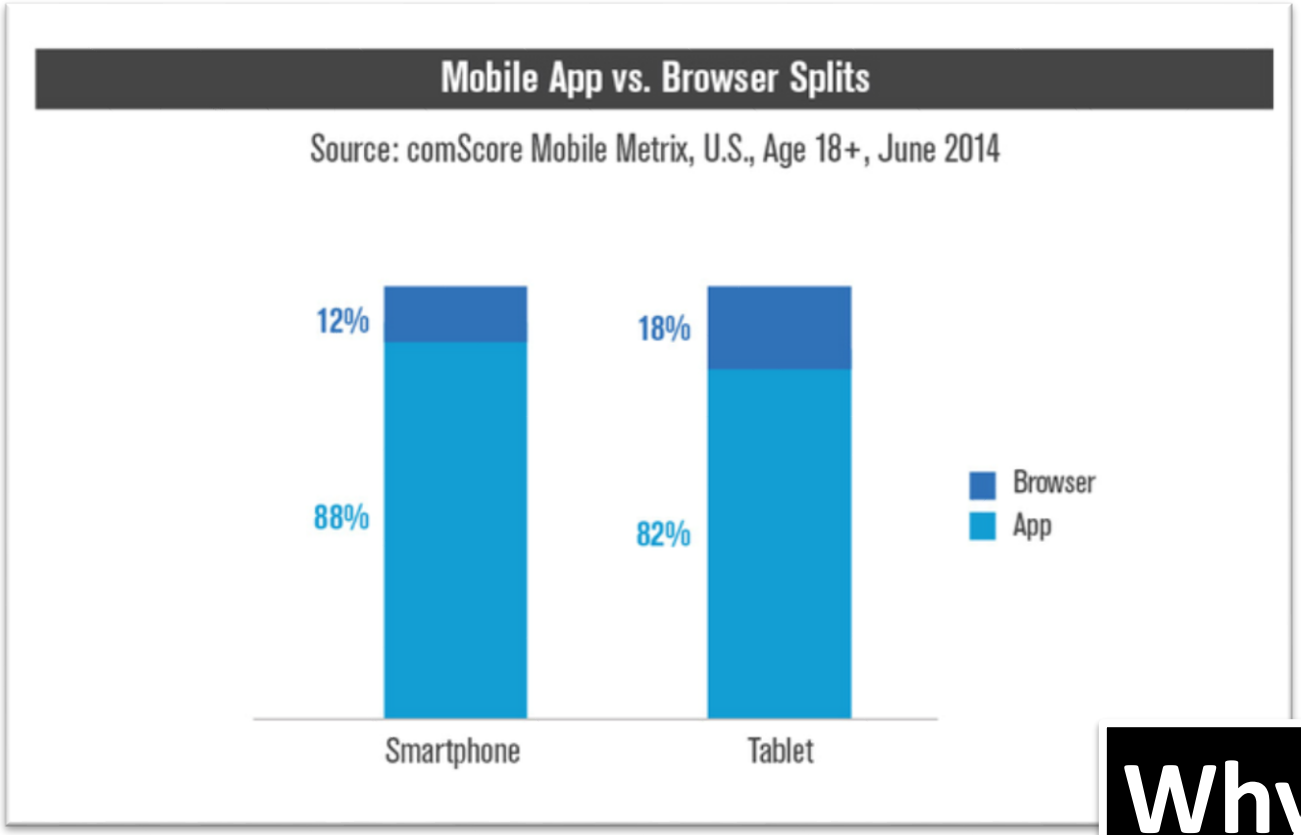
Mobile devices are now widely used for accessing digital media and services on the internet either through **mobile apps** or **web browsers**



http://techcrunch.com/2014/08/21/majority-of-digital-media-consumption-now-takes-place-in-mobile-apps/

Introduction

Mobile **apps** are currently clear ***winners*** over the **mobile web**



Why?

http://techcrunch.com/2014/08/21/majority-of-digital-media-consumption-now-takes-place-in-mobile-apps/

Introduction

App offers a better user experience, they are more *finger-friendly*

Responsive design should have solved this issue

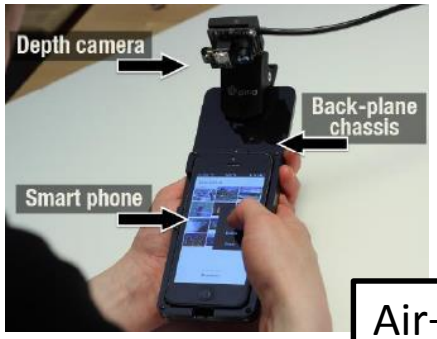
However, its main focus has very much been on **adapting content** and layout rather than **modes of interaction**

Responsive design is not the answer to all our questions

Introduction

Web

Apps



Air+

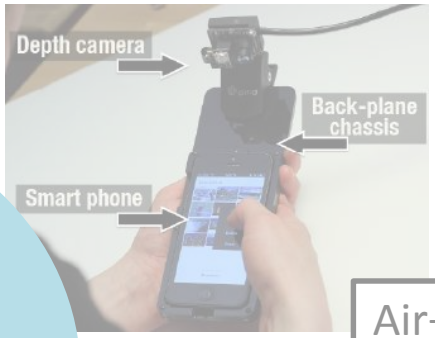
Motion Based Interactions

In air gestures

Introduction

Web

Apps



Air+

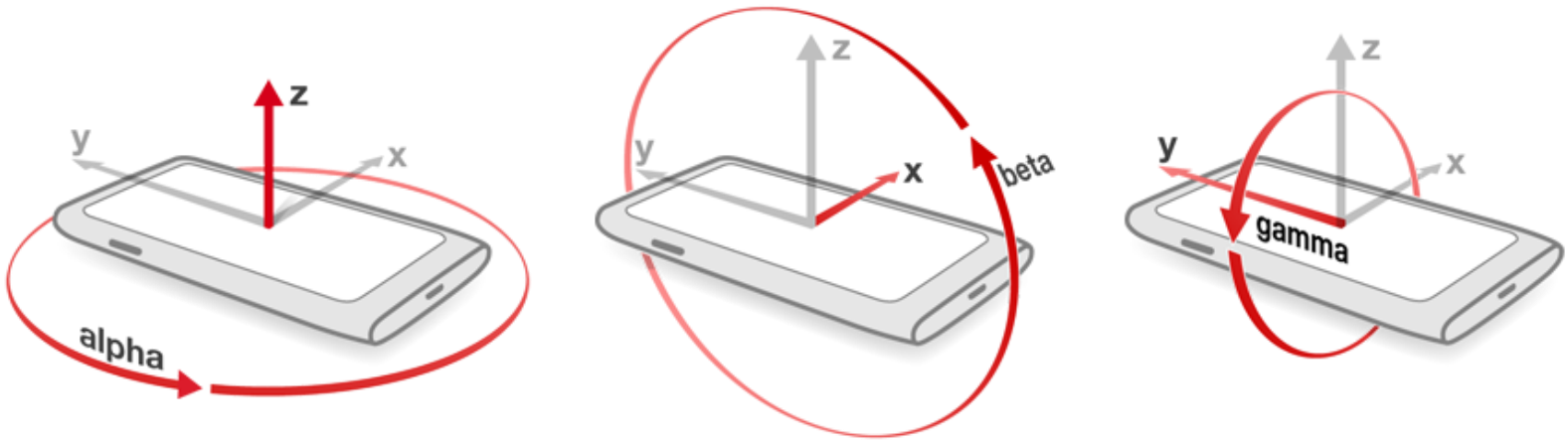
Motion Based Interactions

In air gestures

Motion Based Interactions

Motion based gestures (or *Tilting* gestures) are defined based on the **speed** and **orientation** of the device as measured by **accelerometer** and **gyroscope** sensors

Thanks to those sensors it is possible to recognize tilting gestures performed by the user

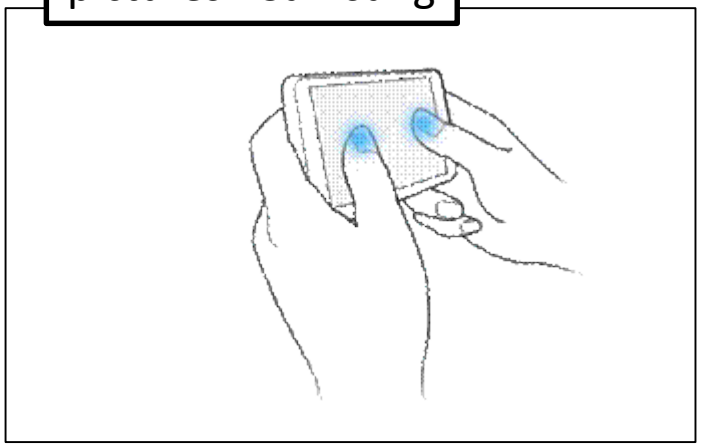


Motion Based Interactions

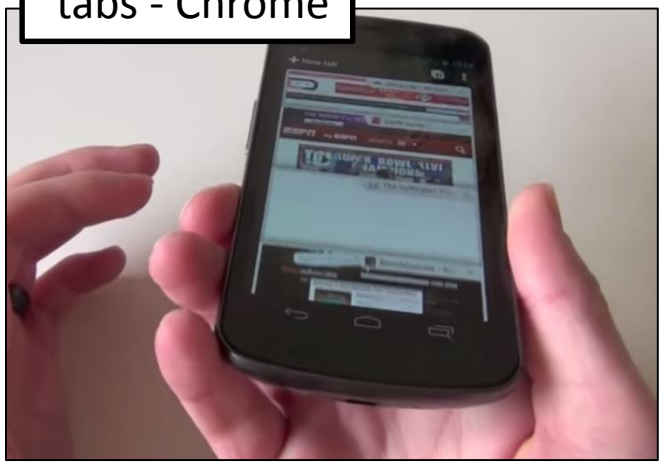
Previous works have studied such interactions in **several scenarios**: image galleries, menu selections, map browsing etc.

One of the advantages of such gestures is the possibility to interact with the device without changing the hand position

Zoom in and out pictures - Samsung



Scroll between tabs - Chrome



Tilt-and-Tap

Tilt-and-Tap is a **jQuery plugin**, its main goals is to apply these knowledge on web-based applications

Allowing developers to use such interaction in their web applications

Tilt-and-Tap offers **two** main type of tilting interactions: ***hard tilting*** and ***continuous tilting*** gestures

Tilt-and-Tap

Hard Tilting Gestures

A ***hard tilting*** interaction is a rapid movement from one position to another that corresponds to a discrete gesture

Some hard tilting gestures recognized by Tilt-and-Tap are the following:

- tiltUp
- tiltDown
- tiltRight
- tiltLeft
- tiltClock
- tiltCounterClock

tilt left gesture

Tilt-and-Tap

Hard Tilting Gestures

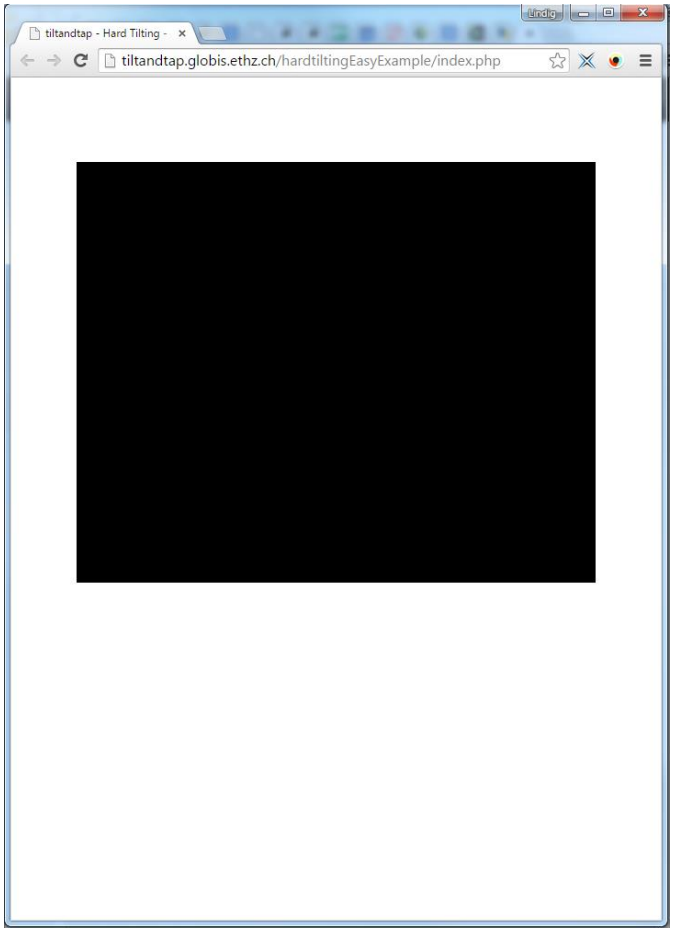
HTML

```
<div id="box" style="width: 250px; height: 250px; background-color: red;">
    <p> I am a Box </p>
</div>
```

JavaScript

```
$("#box").tiltandtap({
    onTiltDown  : changeColor,
    onTiltUp    : changeColor
});

function changeColor ()
{
    var rcolor=Math.floor((Math.random() * 250) + 1);
    var gcolor=Math.floor((Math.random() * 250) + 1);
    var bcolor=Math.floor((Math.random() * 250) + 1);
    $("#box").css("background-color", "rgb(" + rcolor + "," + gcolor + "," + bcolor + ")");
}
```



Tilt-and-Tap

Hard Tilting Gestures

HTML

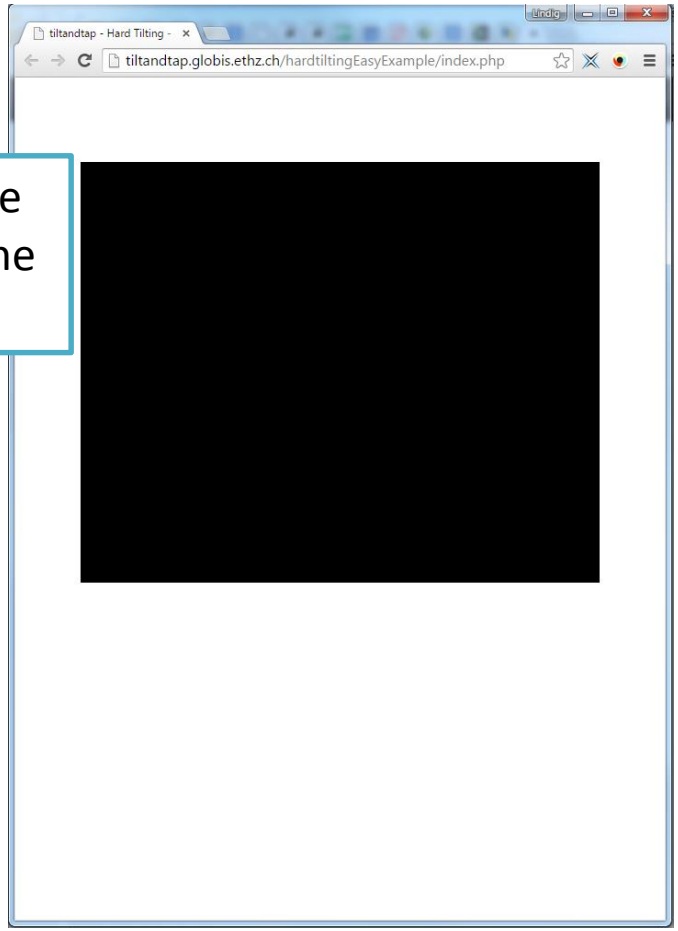
```
<div
    <
</div
```

The only thing the developer has to do it is to include the js in its web app. and define some callbacks for the desired gestures

JavaScript

```
$("#box").tiltandtap({
    onTiltDown  : changeColor,
    onTiltUp    : changeColor
});

function changeColor ()
{
    var rcolor=Math.floor((Math.random() * 250) + 1);
    var gcolor=Math.floor((Math.random() * 250) + 1);
    var bcolor=Math.floor((Math.random() * 250) + 1);
    $("#box").css("background-color", "rgb(" + rcolor + "," + gcolor + "," + bcolor + ")");
}
```



Tilt-and-Tap

Hard Tilting Gestures

To each tilting gestures the user can combine other possible interactions such as:
tap, **double tap** and **tap hold**

So, it is possible to modify the previous example: We now want to change the color of the box only if the user *holds tap on the box* ***and*** *tilt the device*

Tilt-and-Tap

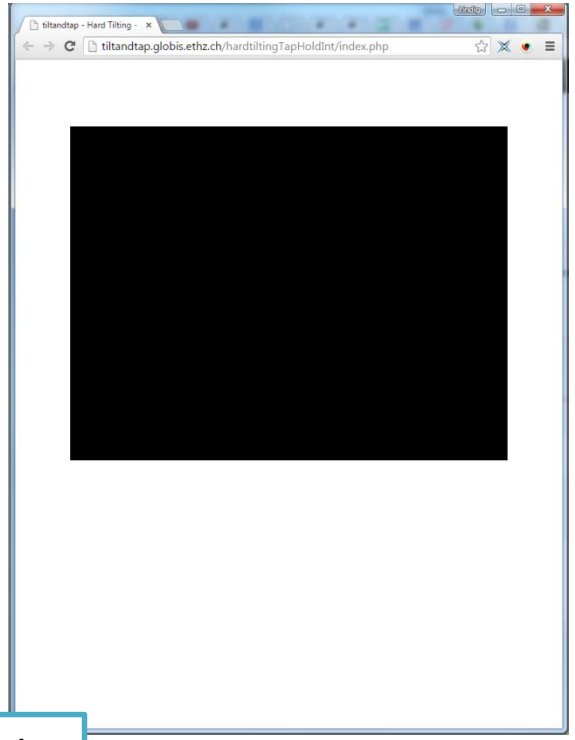
Hard Tilting Gestures

HTML

```
<div id="box" style="width: 250px; height: 250px; background-color: red;">
    <p> I am a Box </p>
</div>
```

JavaScript

```
$("#box").tiltandtap({
    tiltDown  : { onTiltDown: changeColor, interaction : {type: "press", element: "box"}},
    tiltUp    : { onTiltUp: changeColor, interaction : {type: "press", element: "box"}}
});
```



"type" indicates the type of interaction (press = hold tap) and element indicates the ID of the DOM element where the touch interaction is performed

Tilt-and-Tap

Hard Tilting Gestures

You can decide how fast the hard tilting gesture should be by setting the ***dimBuffer*** *Option*

In addition to these possible combinations, Tilt-and-Tap also offers the possibility to have some *user feedback* when a hard tilting gesture is performed

Currently, Tilt-and-Tap supports the following feedback:

- *Visual Feedback*
- *Vibration Feedback*
- *Sound Feedback*
- *And combinations of them*

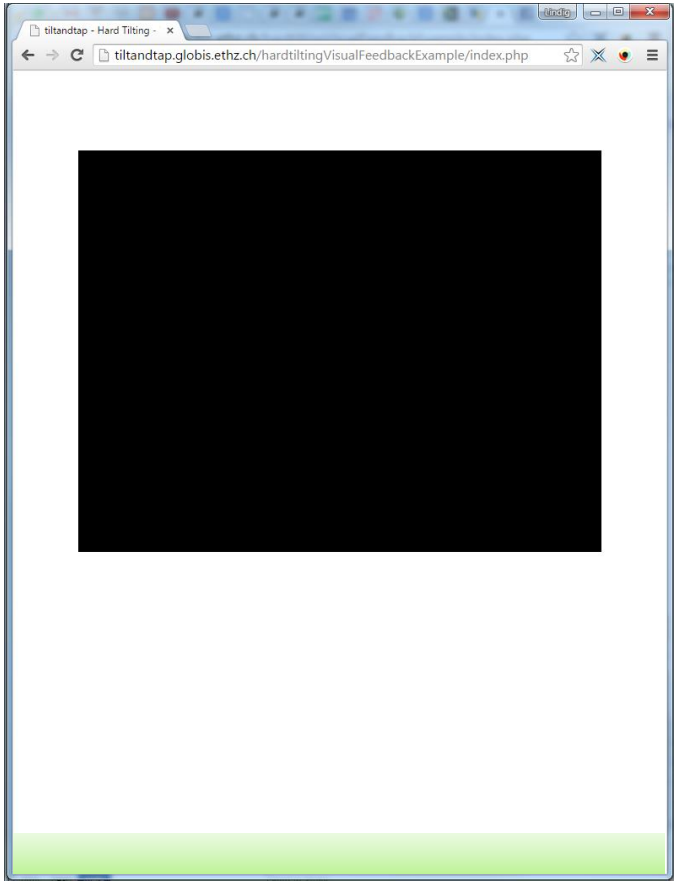
Let's modify the first example in order to have some feedback when a tilting gesture is performed

Tilt-and-Tap

Hard Tilting Gestures

JavaScript

```
$("body").tiltandtap({
    tiltUp {
        onTiltUp: changeColor ,
        visualFeedback: "green"
    }
});
```



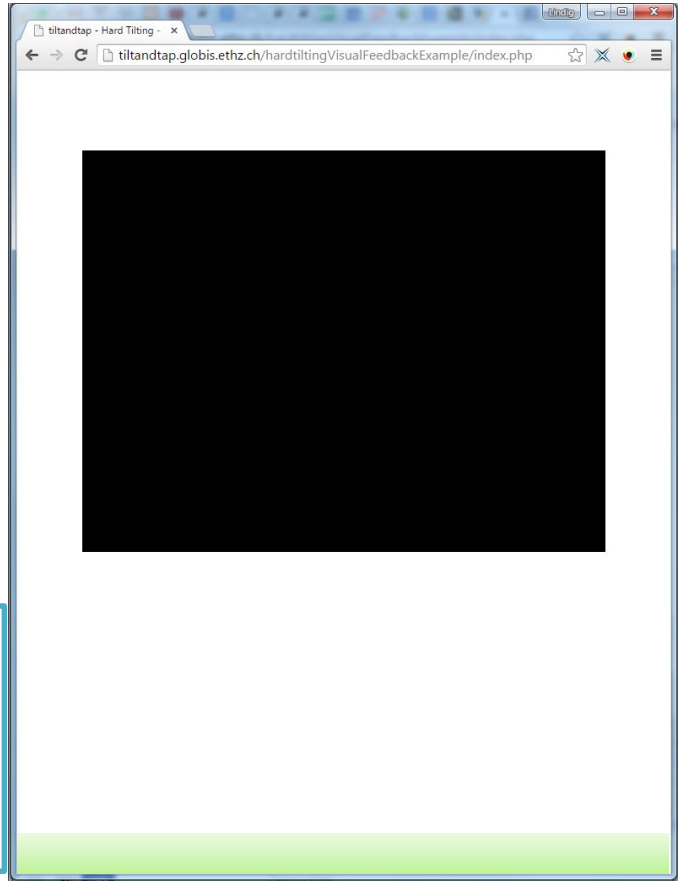
Tilt-and-Tap

Hard Tilting Gestures

JavaScript

```
$("body").tiltandtap({
    tiltUp {
        onTiltUp: changeColor,
        visualFeedback: "green"
    }
});
```

The user can indicate the color of the border. Since this feedback is specified for the tiltUp gesture, the green border will be displayed on the **bottom** side of the page. This div will automatically disappear after some seconds.



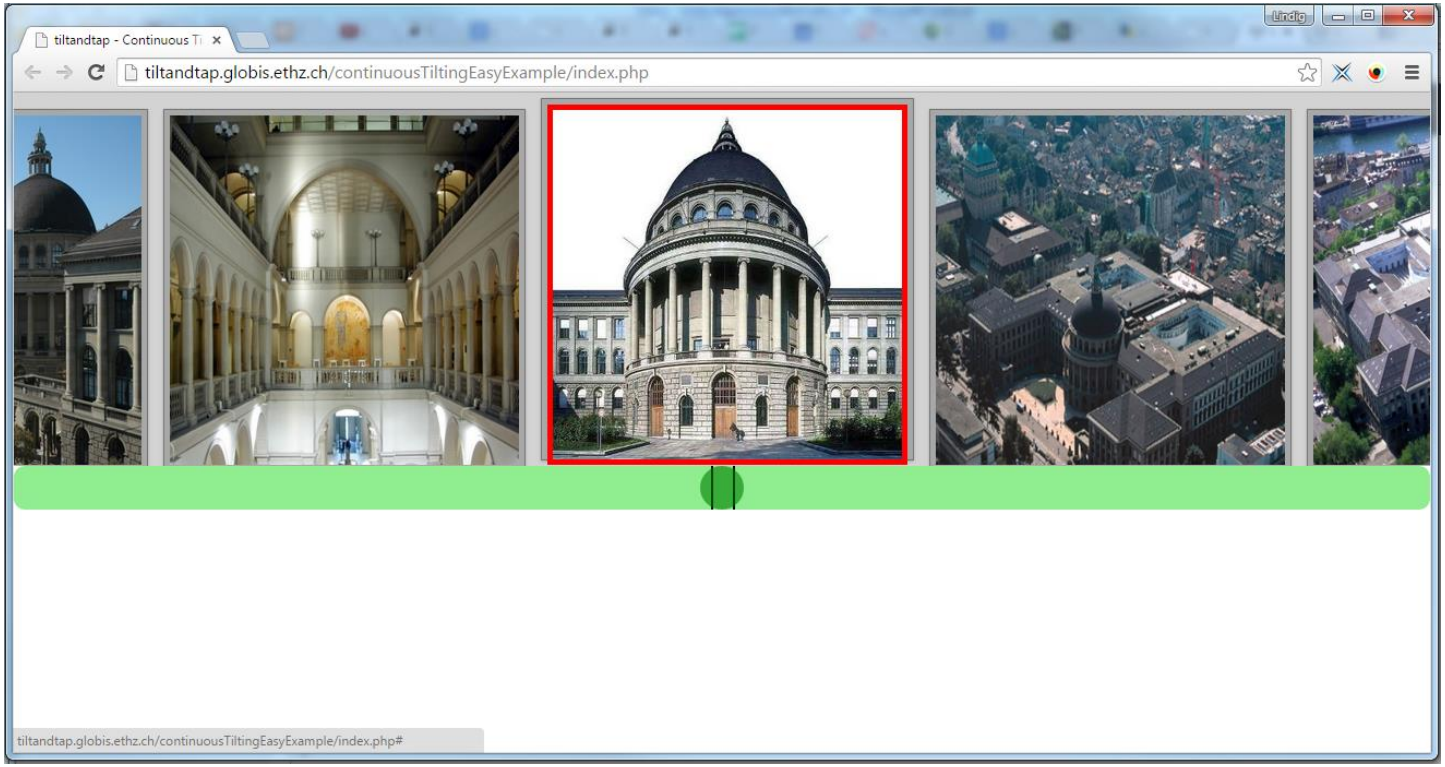
Tilt-and-Tap

Continuous Tilting Gestures

Let’s assume that we have a sequence of pictures in a web page

We want to browse them by moving the device to the left or to the right

We also want some sort of indicator that helps the user while is browsing the gallery

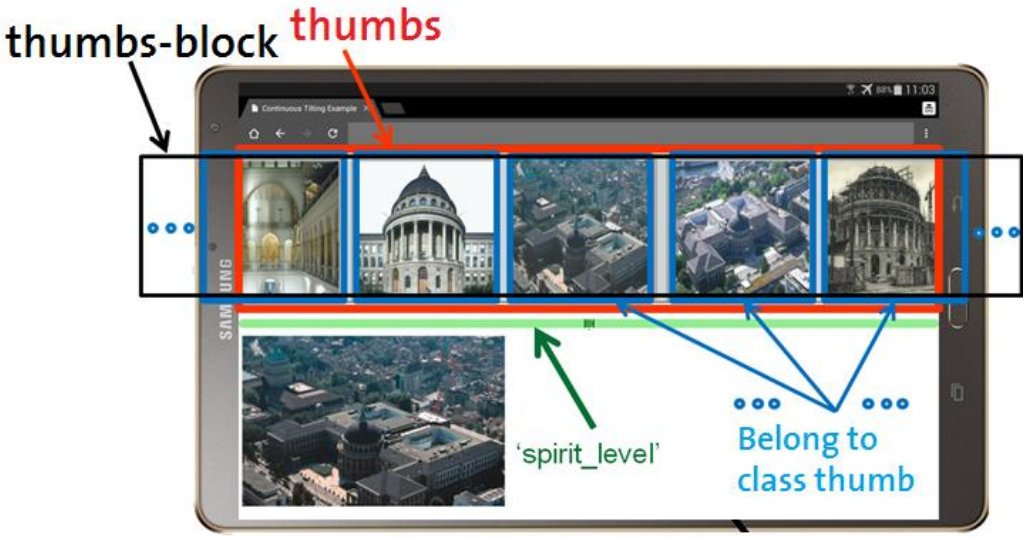


Tilt-and-Tap

Continuous Tilting Gestures

HTML

```
<div class="thumbs-block" id="thumbs-block">
    <div class="thumbs" id="thumbs">
        <a href="#" class="thumb" id="thumb1" ... </a>
        <a href="#" class="thumb" id="thumb2" ... </a>
    </div>
<div id="spirit_level">
    <div id="spirit_level_search"></div>
</div>
```



Tilt-and-Tap

Continuous Tilting Gestures

JavaScript

```
$("body").tiltandtap({
        elem_selection: {
            elem_selected: elem_selected
            ,indicator: {
                radius: "20px", color: "green", left: "50%", top: "0px"
                ,opacity: 0.6, visibility: true
                ,dimensionality: "1D", element: "spirit_level_search"
            }
            ,gallery: {
                classname: "gallery_class"
                ,innerid: "thumbs"
                ,outerid: "thumbs-block"
                ,sliding_enabled: true
            }
            ,speed: 1
            ,sensitivity: 0.5
            ,scrollingdirection: "physical"
        }
});


function elem_selected(cur_elem){
    selected_element = cur_elem;
    var el = cur_elem.getAttribute("id");
    $(".border").removeClass("border");
    $("#"+el).addClass("border");
}
```

Tilt-and-Tap

Continuous Tilting Gestures

JavaScript

```
$("body").tiltandtap({
        elem_selection: {
            elem_selected: elem_selected
            ,indicator: {
                radius: "20px", color: "green", left: "50%", top: "0px"
                ,opacity: 0.5, visibility: true
                ,dimensionality: "1D", element: "spirit_level_search"
            }
            ,speed: 1
            ,sensitivity: 0.5
            ,scrollingdirection: "physical"
        }
});


function elem_selected(cur_elem){
    selected_element = cur_elem;
    var el = cur_elem.getAttribute("id");
    $(".border").removeClass("border");
    $("#"+el).addClass("border");
}
```

Callback function called when an element is selected by the ball

Tilt-and-Tap

Continuous Tilting Gestures

JavaScript

```
$("body").tiltandtap({
        elem_selection: {
            elem_selected: elem_selected
            ,indicator: {
                radius: "20px", color: "green", left: "50%", top: "0px"
                ,opacity: 0.6, visibility: true
                ,dimensionality: "1D", element: "spirit_level_search"
            }
            ,gallery: {
                classname: "gallery_class"
            ,scrollingdirection: "physical"
        }
});


function elem_selected(cur_elem){
    selected_element = cur_elem;
    var el = cur_elem.getAttribute("id");
    $(".border").removeClass("border");
    $("#"+el).addClass("border");
}
```

Options of the ball/indicator

Tilt-and-Tap

Continuous Tilting Gestures

JavaScript

```
$("body").tiltandtap({
        elem_selection: {
            elem_selected: elem_selected
            ,indicator: {
                radius: "20px", color: "green", left: "50%", top: "0px"
                ,opacity: 0.6, visibility: true
                ,dimensionality: "1D", element: "spirit_level_search"
            }
            ,gallery: {
                classname: "gallery_class"
                ,innerid: "thumbs"
                ,outerid: "thumbs-block"
                ,sliding_enabled: true
            }
            ,speed: 1
        }
});

function
    selec
    var el = cur_elem.getAttribute("id");
    $(".border").removeClass("border");
    $("#"+el).addClass("border");
}
```

Options relative to the gallery: class of the pictures, the "wrapper" *innerid*, and the child div *outerid*

Tilt-and-Tap

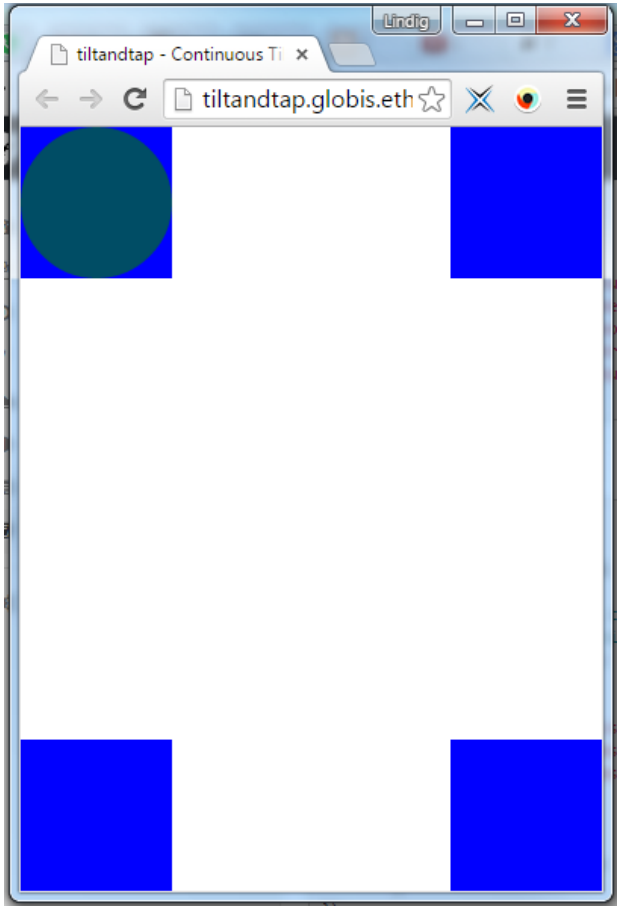
Continuous Tilting Gestures

Tilt-and-Tap offers also the possibility to use the continuous tilting interaction not only in one dimension, but also in two dimension

```
$("body").tiltandtap({
        elem_selection: {
            elem_selected: elsel
            ,indicator: {
                radius: "50px", color: "green", left: "0%", top: "0%"
                ,opacity: 0.6, visibility: true
                ,dimensionality: "2D", element: "thumbs-block"
            }
            ,gallery: {
                 classname: "divs"
});

function elsel(cur_elem){
       selected_element = cur_elem;
       var el = cur_elem.getAttribute("id");
       $(".border").removeClass("border");
       $("#"+el).addClass("border");
}
```

The only thing that has to be changed in order to allow such interaction is the *dimensionality feature*



Tilt-and-Tap

Wiki

Tilt-and-Tap offers several other features (and combination of them) for both continuous and hard tilting gestures

Such features are explained in the Tilt-and-Tap wiki, that you can find here

In the wiki is also available the code for each of the example we showed you today

Option Name	Description	Default Value	Possible Values
onTiltLeft	The sub-option onTiltLeft is a function which calls a developer-specified function when the framework detects a left tilting gesture	null	an existing function
thTiltLeft	Threshold for the left tilting	3.1	any number ***More information here***
audioFeedback	It indicates which audio file ("click.mp3" in the example below) should be played if the corresponding tilting gesture is performed	null	any existing path of an audio file
vibrationFeedback	It indicates for how long the mobile device should vibrate (if it can) as a vibrotactile feedback of the tilting gesture	null	number in milliseconds
visualFeedback	The sub-option visualFeedback shows the user graphically which gesture was performed by displaying on the corresponding border a color gradient with a thickness depending on how intense the gesture was performed.	null	any color (hex, rgb or text)
interaction.type	The hard tilting gesture can be combined with an additional interaction (which has to be done on beforehand) whose type can for example be a pressing on a specific element indicated by its DOM (Document Object Model) element's ID. This option indicates which interaction has to be performed.	null	"press"
interaction.element	It indicates where the touch interaction (if indicated) has to be performed.	null	a valid ID of an existing element in the DOM

Option Name	Description	Default Value	Possible Values
elem_selection	Inside the elem_selection option, the developer can specify with the sub-option elem_selected what shold happen when an element is selected with the ball. In that case, the selected element is returned to the developer	null	an existing function
speed	By speed, we indicate how fast the gallery of elements (if available) and the ball should slide depending on the tilt angle of the mobile device	null	[0, Number.MAX_VALUE]
sensitivity	The majority of movement sensors built in mobile devices return also noise values. This has an effect to the framework and makes the ball (and also the gallery of elements) move although the mobile device does not. By specifying a sensitivity value, this unwanted behavior can be prevented	0.5	[0, Number.MAX_VALUE]
scrollingdirection	**The value "physical" (for Android devices) causes the ball (also the gallery) to obey the laws of physics. That means if the mobile device is continuously tilted to left, the gallery will slide to left, while the viewport remains fixed. The value "counterphysical" (for Android devices) reverses the sliding direction of the ball and the gallery. For Apple products if you want the "physical" behavior you should use the "counterphysical" keyword and vice versa. More details here.**	"physical"	"physical", "counterphysical"
gallery.classname	Common class of elements to be selected/browsed by the ball	null	any existing class name
gallery.innerid	Div that contains all the elements (see Figure)	null	a valid id of an existing element in the DOM
gallery.outerid	Showed viewport (see Figure)	null	a valid id of an existing element in the DOM
indicator	This option can be used to personalize the ball appearance and its behaviours . All the options are defined in the next table	//	//

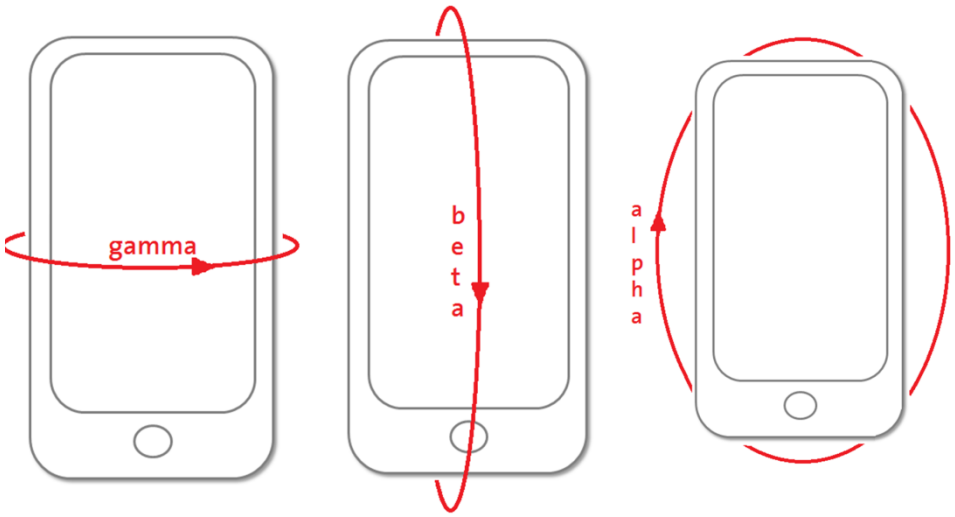
Tilt-and-Tap

HTML5 Events

Tilt-and-Tap makes use of two HTML5 Events:
DeviceOrientationChange
DeviceMotionChange

DeviceOrientationChange

Gives us the possibility to access to the orientation of the device by returning each *X (where X depends on the current device/browser)* milliseconds **three angles**



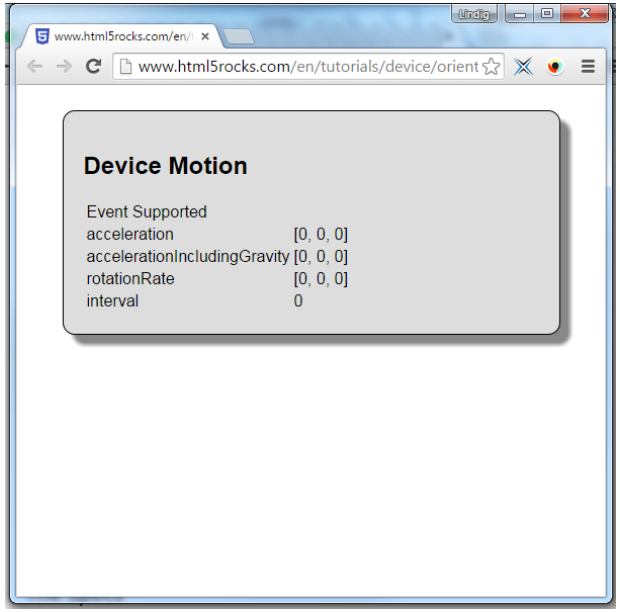
Tilt-and-Tap

HTML5 Events

Tilt-and-Tap makes use of two HTML5 Events:
DeviceOrientationChange
DeviceMotionChange

DeviceMotionChange

Gives us other information such as the acceleration of the device not only its direction
Moreover, it also indicates the interval of the device/browser.
The interval indicates how many milliseconds these data are returned.



Tilt-and-Tap

Hard Tilting - Implementation

To implement the hard tilting recognition, Tilt-and-Tap makes use of the *rotationRate* data returned by the **DeviceMotionChange** event

rotationRate indicates the rate of change of the device, and it is calculated by using the three angles returned by the **DeviceOrientationChange** event and the **acceleration** of the device

Similar to **DeviceOrientationChange**, it gives three values alpha, beta and gamma, however, in this case, representing ***degree per second***

Tilt-and-Tap

Hard Tilting - Implementation

We have noticed that every time a user performed a tilting gesture in some direction, the opposite tilting gesture was also performed

Caused by a ***recoil*** factor

For this reason we came up with the following idea to implement the hard tilting recognition

Tilt-and-Tap

Hard Tilting - Implementation

We have noticed that every time a user performed a tilting gesture in some direction, the opposite tilting gesture was also performed

Caused by a ***recoil*** factor

For this reason we came up with the following idea to implement the hard tilting recognition

Let’s assume that the interval of the device is equal to 50 milliseconds

Tilt-and-Tap

Hard Tilting - Implementation

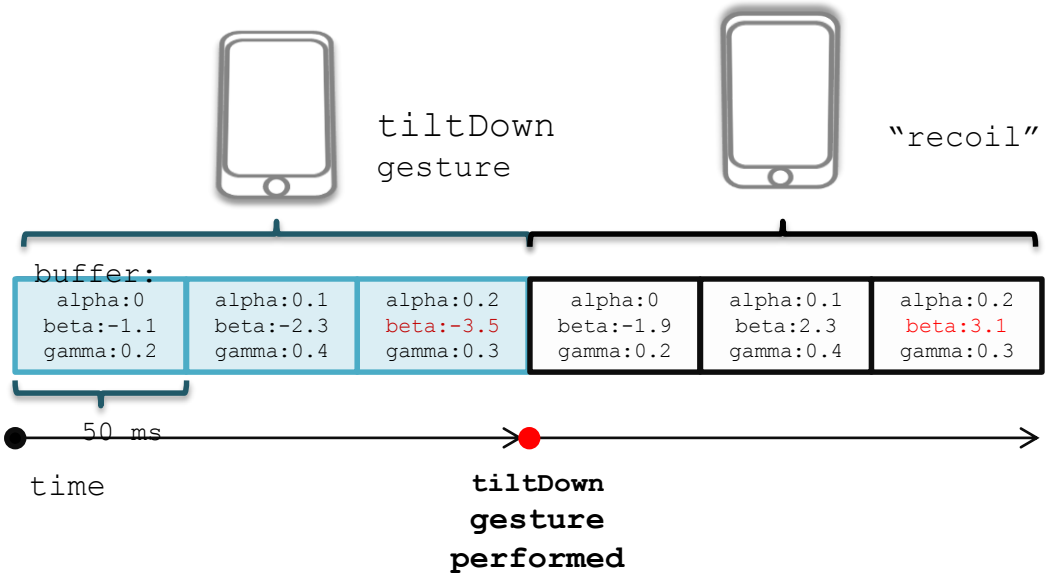
1. When the framework is initialised, it has an empty buffer of size three.

2. Every 50ms, the API returns an object containing the three values (alpha, beta, gamma) for the rotationRate of the device and we store it in the first free position in the buffer.

3. When the buffer is full, we check the last object stored:
   (a) If any of the sensor values in the object are bigger than the corresponding threshold, we save this information and go to step four.
   (b) If none of the values is larger than the threshold, we clear the buffer and start again from step one.

4. We call the function defined by the developer for the tilting gesture recognised.

5. We clear the buffer and discard the next three sensor readings to cater for recoil.



We save the first data in order to keep track of the motion history this allow the plugin to be more flexible

The developer can set the dimension of the two buffers

Tilt-and-Tap

Continuous Tilting - Implementation

To implement the continuous tilting interaction, Tilt-and-Tap makes use of the *accelerationIncludingGravity* data returned by the **DeviceMotionChange** event

The *accelerationIncludingGravity* object provides values for the device acceleration and direction in 3D space

Tilt-and-Tap

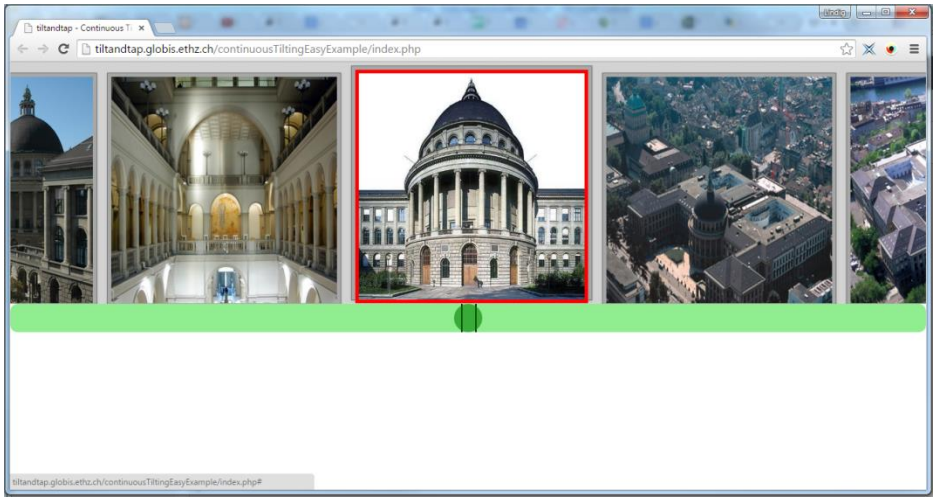
Continuous Tilting - Implementation

Our approach is a visual approach and it involves the use of a ball that, like a cursor, indicates the current position in the page

So the ball has ***two main roles***: it gives feedback to the user and its position and *velocity* will indicate how and where to move the target elements

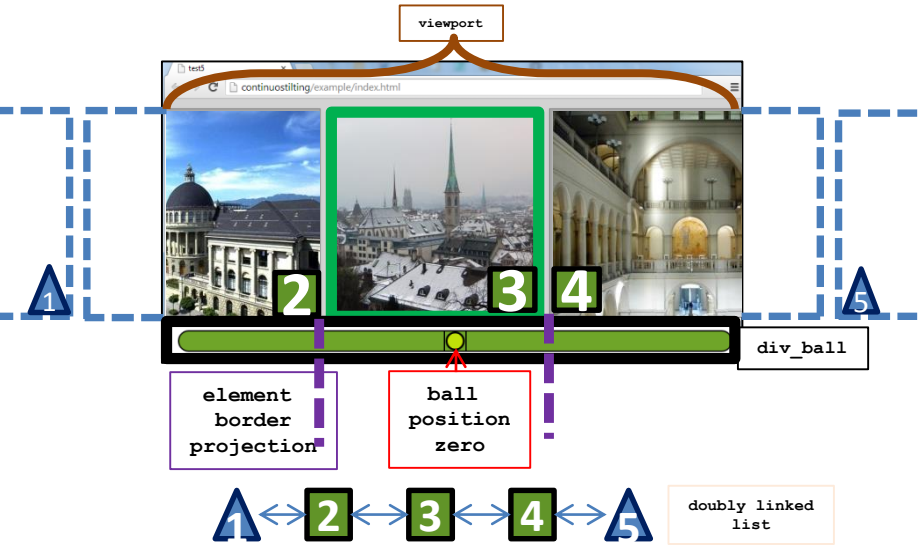
By default the ball will be visible, but the developer can also decide to hide it in the page

There are several ways to implement such interaction, we propose two ways



Tilt-and-Tap

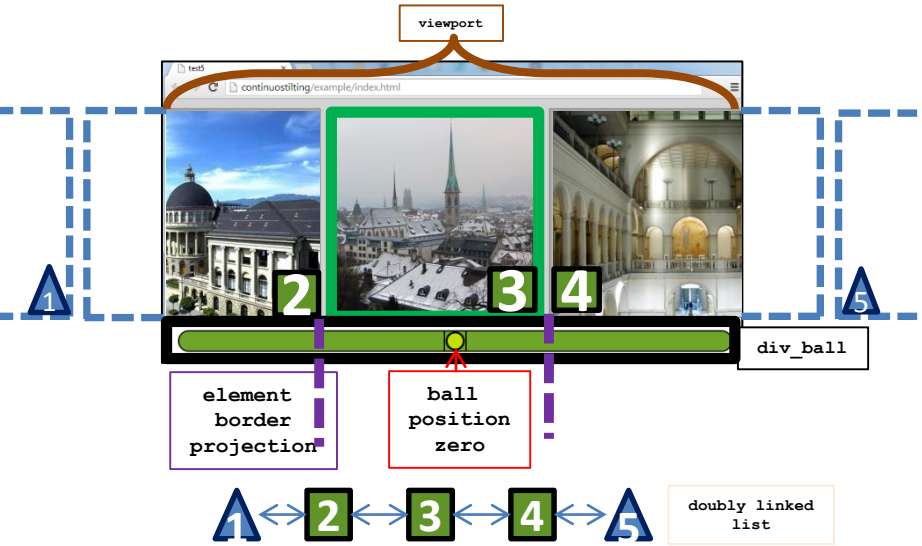
Continuous Tilting - Implementation



1) When the page load we create a doubly linked list of all the element with the class indicated by the user, this list is ordered depending on the position of the element
   I. This list also maintains information about elements that are inside or outside the viewport
2) Every 50 milliseconds we move the ball and the viewport
   I. We update the list ***O(2)***
   II. We check if the ball falls inside a different element ***O(s)***

Tilt-and-Tap

Continuous Tilting - Implementation



1) When the page load we create a doubly linked list of all the element with the class indicated by the user, this list is ordered depending on the position of the element
   I. This list also maintains information about elements that are inside or outside the viewport
2) Every 50 milliseconds we move the ball and the viewport
   I. We update the list ***O(2)***
   II. We check if the ball falls inside a different element ***O(s)***

O(2) -> it has to check only if the first element viewed in the viewport is now (if the direction of the motion interaction *goes* to the left), and if the last element is not visible anymore

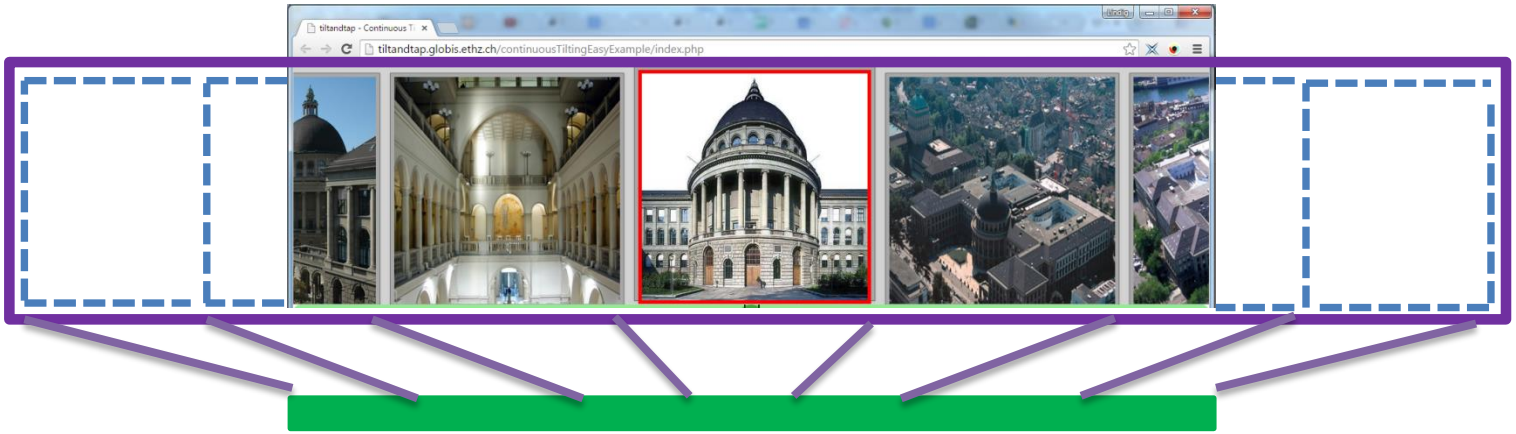
O(s) -> where s are the elements that are now visible in the viewport

Tilt-and-Tap

Continuous Tilting - Implementation

The other approach we developed divide the div of the ball in *n* segments *(when the page is loaded)* where *n* are the elements that have *class* indicated by the developer

Each segment corresponds to an element and when the ball falls inside that segment we select the corresponding element



Tilt-and-Tap

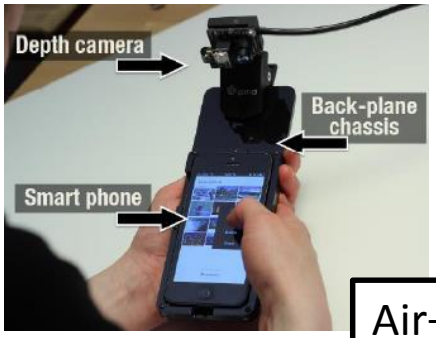
Portability and Performance Issues

Do you remember?

Web

Apps

Motion Based Interactions



Air+

In air gestures

Tilt-and-Tap

Portability Issues

The main the reason why the web is not used as apps are used to *innovate* regards its

But avoid problems by just forgetting about the web, it is not a solution, this is the reason why Tilt-and-Tap is developed by using only JavaScript / jQuery

… with some challenges

Tilt-and-Tap

Portability Issues

The two events: **DeviceOrientationChange** and **DeviceMotionChange** are still work in progress and not all the browsers implement them in the same way

Tilt-and-Tap

Portability Issues

On caniuse.com it is possible to see which browser supports these two events, but sometimes this information are wrong or old

For this reason we have tested the two events by our own and we came up with the following table

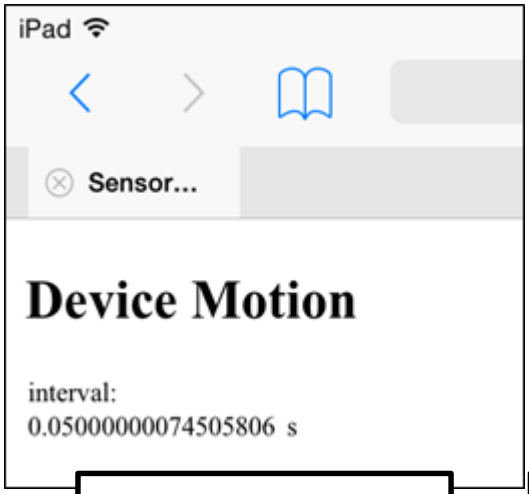
API	Mobile Browser	Version	Android						iOS	Windows 8.1
			Samsung S2	Samsung 8.4	Nexus 5	Nexus 7	HTC M8	LG Watch W100	IPad	Nokia Lumnia 925
DeviceOrientation - DeviceMotionChange	Chrome	40	Yes	Yes	Yes	~	Yes	/	Yes	/
	Firefox	35	Yes	Yes	Yes	~	Yes	/	/	/
	Safari	9.1	/	/	/	/	/	/	Yes	/
	Internet Explorer	11	/	/	/	/	/	/	/	No
	WIB	1.0beta 19	/	/	/	/	/	Yes	/	/

Tilt-and-Tap

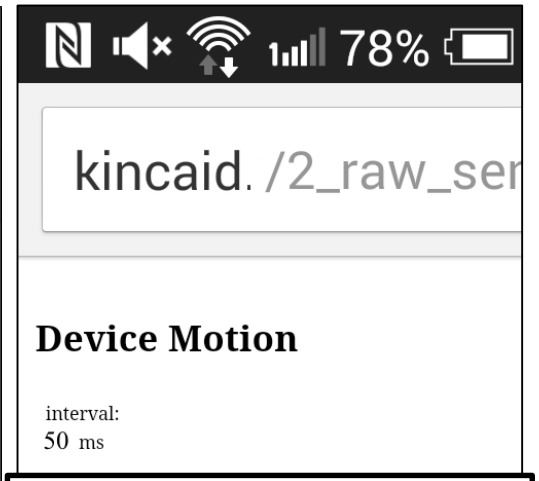
Portability Issues

We then compared the data returned by **DeviceOrientationChange** and **DeviveMotionChange** for three different browsers and devices to determine the main differences between browsers and platforms

1) The interval data given by the **DeviceMotionChange** event, should be returned in *milliseconds* but, while this is true on most browsers on Android devices, this is not true on Apple devices



Safari on iPad Air



Chrome on Samsung Tab

Tilt-and-Tap

Portability Issues

We then compared the data returned by **DeviceOrientationChange** and **DeviveMotionChange** for three different browsers and devices to determine the main differences between browsers and platforms

2) The interval data returned by Firefox is 100ms, in Safari and Chrome on Apple products is 50ms in Chrome and other browsers on Android devices is ~ 16ms

Tilt-and-Tap

Portability Issues

We then compared the data returned by **DeviceOrientationChange** and **DeviveMotionChange** for three different browsers and devices to determine the main differences between browsers and platforms

3) The range of the beta value for DeviceOrientationChange in Firefox and Chrome on Android devices is [-90,90], but [-180,180] on Safari and Chrome on iOS. However, the range of the alpha value for DeviceOrientationChange in Firefox and Chrome on Android devices is [-180,180], while it is [-90,90] in Safari and Chrome on iOS

4) The value **accelerationIncludingGravity.x** in Safari is positive if the device is tilted to the right and negative if tilted to the left. In all the other browsers, it is the opposite.

Tilt-and-Tap

Portability Issues

We solved the first problem by the use of buffers

We solved the last 3 problems by using a *switch* that depending on the device connected will accordingly change thresholds and the buffer dimension

Tilt-and-Tap

Performance Issues

Performance issues are crucial in such applications where functions are called several time in a seconds

For this reason it is necessary to write carefully avoiding *false friends ($(".selector")),* and extensive external plugins

Tilt-and-Tap

Links

https://eday.inf.ethz.ch/lindad/tiltandtap/wikis/home

https://eday.inf.ethz.ch/lindad/tiltandtap/wikis/thresholds

https://eday.inf.ethz.ch/lindad/tiltandtap/wikis/options

http://www.html5rocks.com/en/tutorials/device/orientation/