

IBOOKS IN SCHOOLS

# Interactive activities



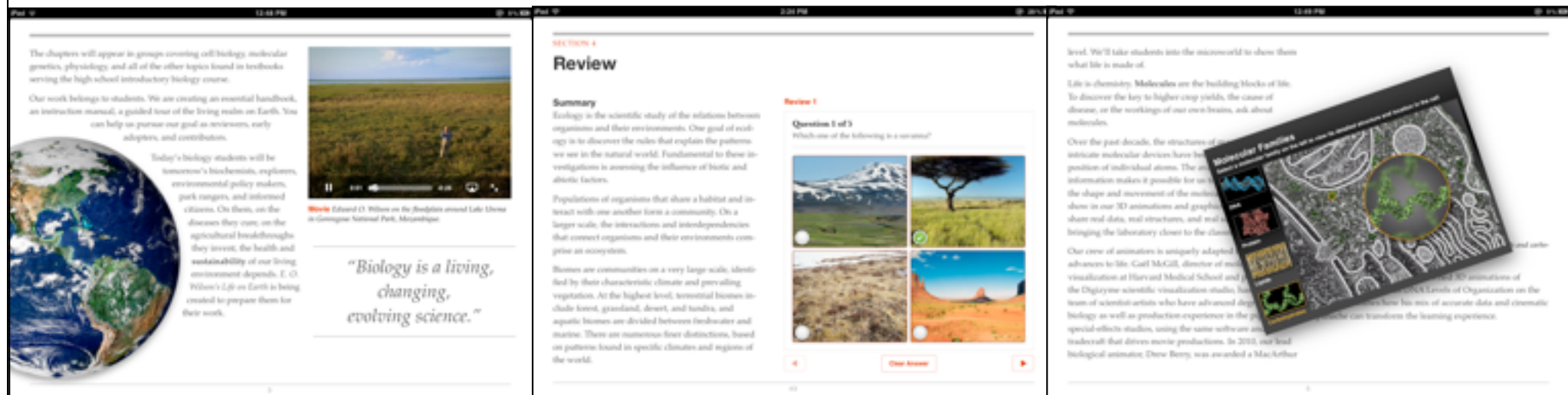
Marielle Lange



iBooks Author

## CHAPTER 1

# Interactive iBooks

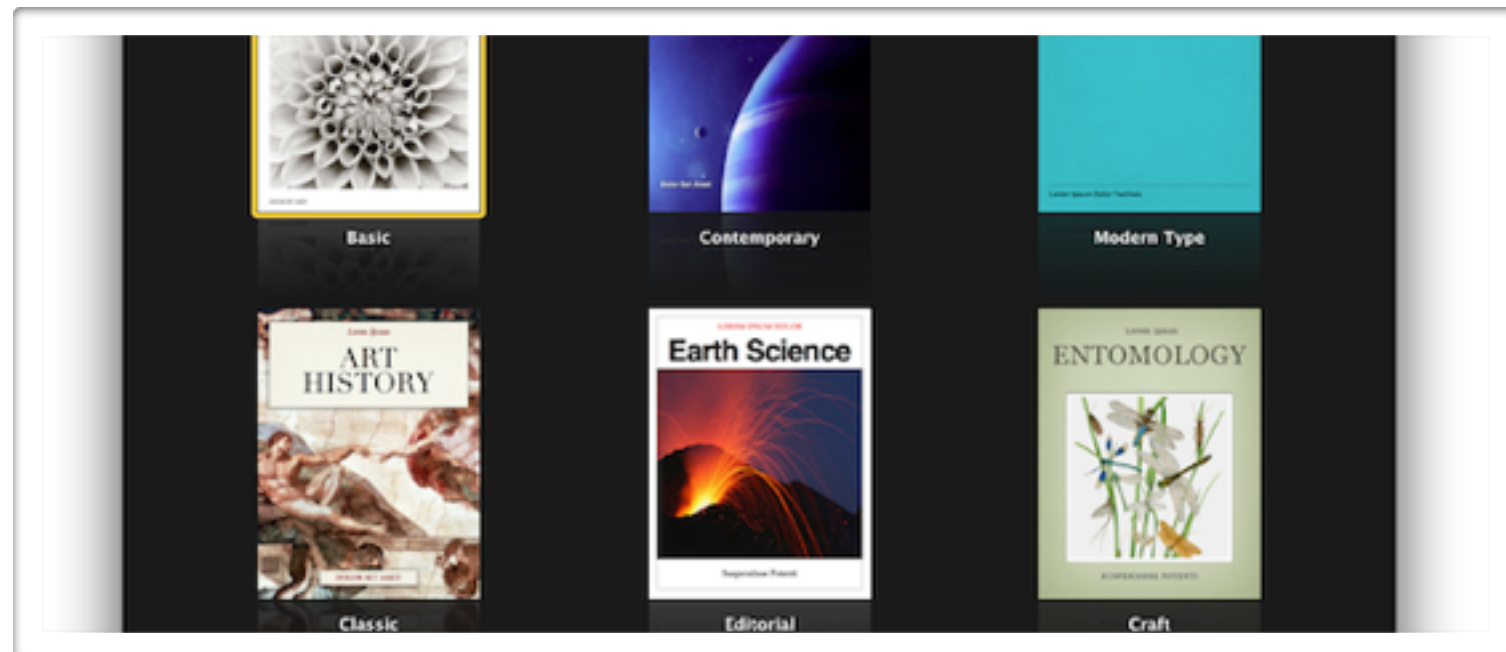


Interactivity in iBooks is driven exclusively by widgets. To add any in your book, click on the Widgets icon in the application toolbar. At the time of writing this tutorial, 7 widgets are available: Gallery, Media, Review (aka questionnaire), Keynote, Interactive Image, 3D and HTML. Tutorials for the first 6 are only a Google search away. Here we will focus on the less documented HTML widgets.

# Creating an iBook

## iBook Author

1. Install iBookAuthor
2. Select an appropriate template
3. Create a book with your chosen template



*Template Chooser View in iBook Author*

iBookAuthor is a mac only application that requires OSX 10.7.2 or later. If you are not too keen of upgrading your machine just yet, you can run the OS from an external disk (more info).

The simplest way to obtain iBookAuthor is via the AppStore. Simply type iBooks Author in the

search book, then follow the usual steps to download and install the free application.

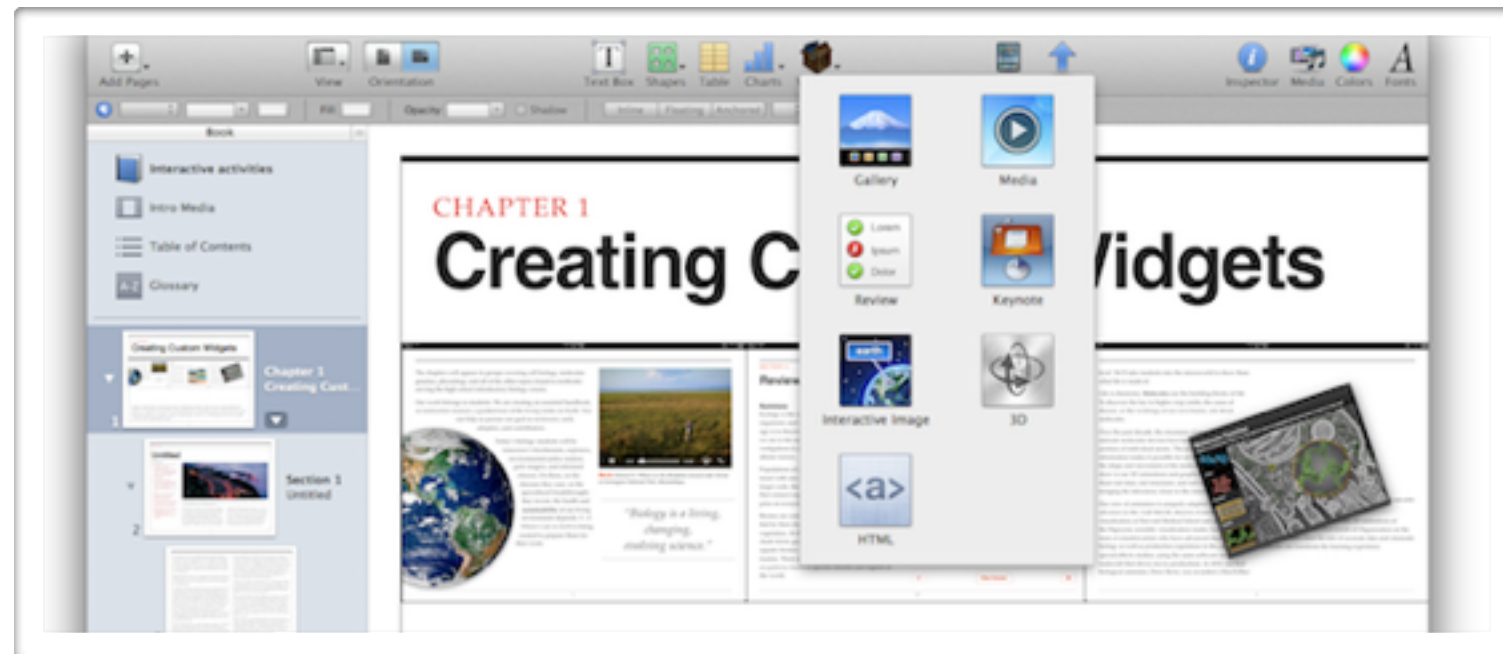
Once installed, launch the application and select one of the existing templates. We will go for "Earth Sciences" as it provides a horizontal layout.

# HTML Widgets

## Highly Interactive Activities

HTML widgets let you create completely custom and possibly highly interactive activities with any combination of:

1. HTML
2. HTML5
3. CSS3
4. JavaScript



*To add interactivity in your book, add a Widget. At the time of writing this tutorial, 7 widgets are available: Gallery, Media, Review (aka questionnaire), Keynote, Interactive Image, 3D and HTML.*

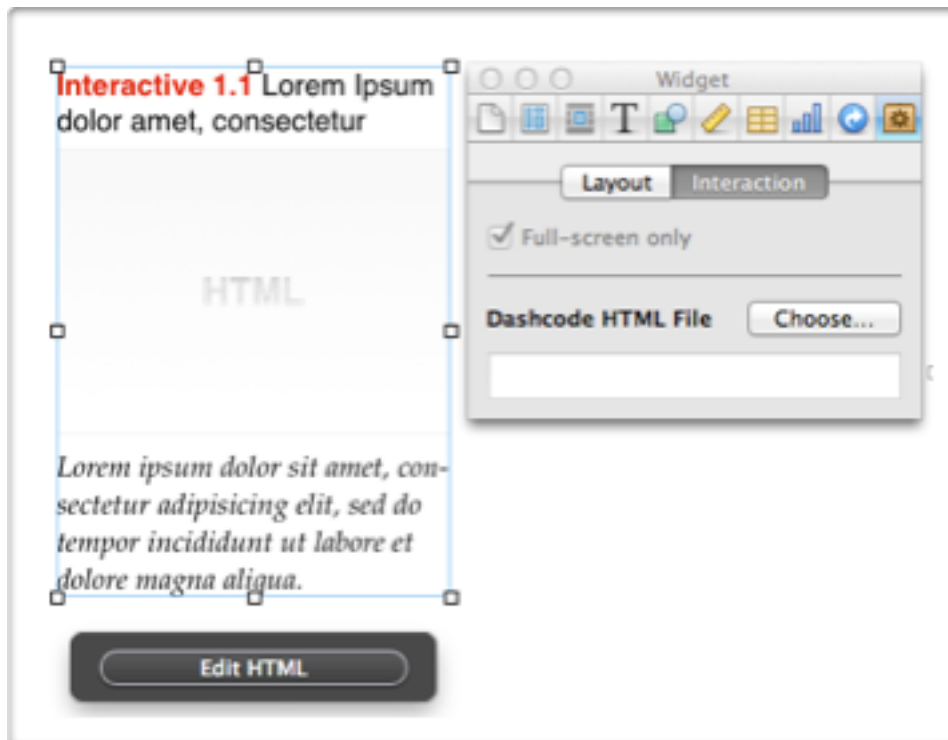
Specialist tutorials for the first 6 can easily be found with your favorite Search Engine.

What is not that well documented yet at the moment are HTML widgets. They let you create completely custom inter actives. I will focus on this here.

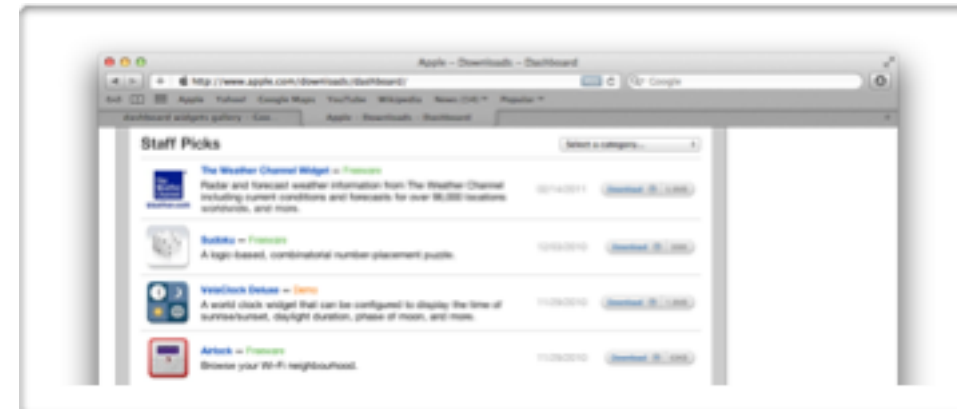
HTML widgets truly open up for a world of opportunities. They let you create completely custom and possibly highly interactive activities. The limit is mostly what can be achieved with any combination of HTML, HTML5, CSS3, and JavaScript.

## Dashcode HTML File

If you select Widget, you will be prompted for a “Dashcode HTML File”. It can be a bit confusing.



What you are really asked to select is a dashboard widget. That is a collection of files bundled as a widget, with extension, .wdgt. This is the exact same type of widgets as you would use in the Dashboard mode that has been available for quite some time on Apple computers.



*Dashboard Widgets*

There are three ways to create widgets.

1. Use special authoring tools that will let you generate widget content. The widget content is then often limited to animations but you can often get away without having to do any coding.
2. Write the widget from scratch using Dashcode.
3. Adapt a widget template.

### Create Simple Animations

Sometimes, all you want is a good animation to illustrate a point. Many tools have appeared in the recent years that let you create such animations without the need for any coding skill.

Sencha Animator - [How to Embed Interactive CSS3 Animations in an iBook](#).

Tumult Hype - [Easy HTML5 Animations in iBooks using Tumult Hype and iBooks Author](#)

Adobe Edge (provided you don't use SVG) - [Create HTML5 widget for iBook Author with Adobe Edge](#) (youtube video)

### Create A Widget With Dashcode (complex)

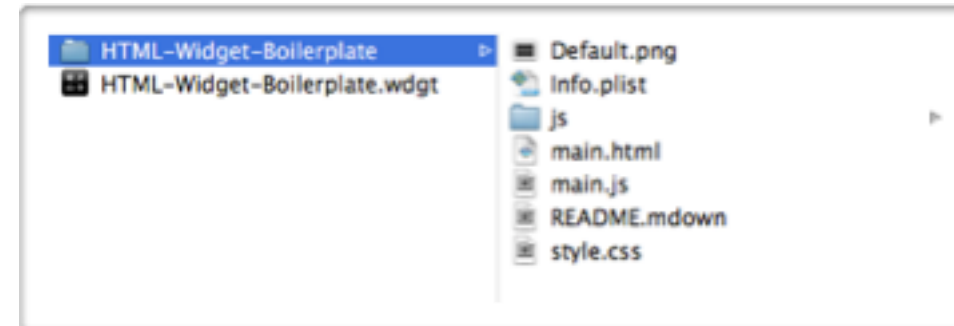
If you check Apple's official documentation on [HTML Widgets](#), you will see Dashcode mentioned as an editor that you can use to write widgets. A problem then is that Dashcode is made available with the Xcode developers tools. If you search the AppStore for Dashcode, you will get no result. You need to download Dashcode from the developers websites and access to this requires an Apple Developer account (which is free). More information on how to obtain, install and run Dashcode can be found in Apple's official [Dashcode User Guide](#).

### Adapt A Widget Template (simple)

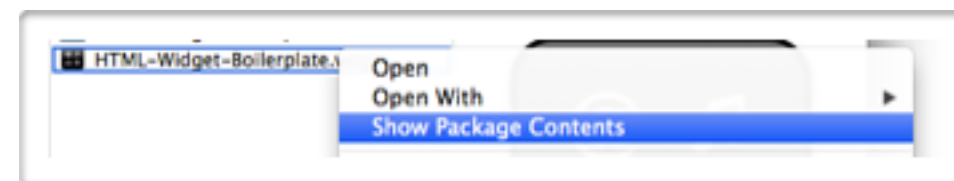
Whether you do or don't already have some knowledge of html and javascript, the quickest and easiest way to get started is from a widget boiler plate.

You will find one such template kindly provided by Trevor Burnham on github, under [iBooks-HTML-Widget-Boilerplate](#).

What is really handy to be aware of is that the widget is only a way to bundle a couple of files. It really is a folder with files. If you add a .wdgt extension to any folder, it will change its appearance to a widget bundle. Conversely, if you remove the '.wdgt' extension from any dashboard widget (you may have to use the get info dialog for this), the file will transform into a folder.



To access the folder content without having to touch the extension, right-click on the file and then select 'Show Package Contents'. The folder content will show up on the Finder.

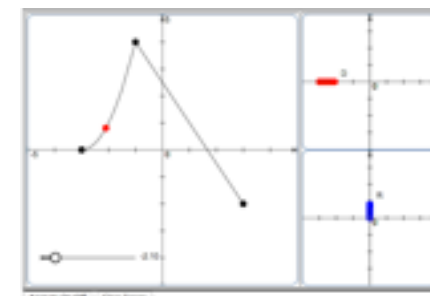


### Basic Requirements

A widget requires at least 3 files.

1. An image, 'Default.png' that provides a preview of the widget, as a thumbnail picture.
2. An html file, '[widget].html' that specifies the content of the interactive widget.
3. A plist file, 'Info.plist' to list the editable properties of the

#### Interactive 1.2 jsxGraph Test



*User contributed code on  
jsxGraph google group*

widget. This defines the source, appearance, and behaviour of the widget.



## What can Widgets do?

HTML widgets execute within a Webkit view. They can display almost any content that you could run in a safari browser. Almost as some webkit features have been deliberately turned off (and others added on) to adapt to the iBook context.

Early experiments suggest that widgets can use pretty much any javascript plugin or library out there.

### Gallery 1.1 Examples of reusable jQuery plugins



*Interactive charts*



And this really is what makes iBooks all of a sudden way more interesting, compared to traditional eBooks. There are a large number of quality of re-usable plugins or javascript libraries that allow you to create complex interactives, with limited programming knowledge.

Gallery 1 shows some example of plugins that could be of interest for illustrating learning content. I haven't checked them all and cannot guarantee that they all work within a widget. But this gives an idea of the scope of what can be done with javascript.

## Plugins

Plugins are stand-alone visual components that you can reuse across projects. The best ones don't require any technical knowledge to be re-used. All you need to do is add the required files in a separate folder, refer to them from your main html file and do little more than specify your media assets, in a way that is very similar to the way you would create an image gallery in iBooksAuthor, except that it will all be in text format.

For instance, this slideshow presentation:



Would require no more than this code:



```

<div id="slideshow">
  <ul class="slides">
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </ul>
  <span class="arrow previous"></span>
  <span class="arrow next"></span>
</div>

<script src="js/jquery.min.js"></script>
<script src="js/slideshow_plugin.js"></script>

```

Plugins can easily be found using any search engine. Adding jQuery typically helps. Search for ‘best plugins for [subject]’ for articles that propose a round up for plugins on a given theme. For instance, [180 Awesome jQuery Slider and Effects Roundup](#).

### *Libraries*

Libraries are of more use to the coders among you. They provide abstract away functionality. A familiar one is jQuery, a library that facilitates the manipulation of html documents. Other libraries, like jQuery mobile facilitate the creation of visual interface elements. Still other facilitate the creation of games or complex graphics.

Like plugins, they are typically quite easy to find. For instance, an extensive list of [javascript game libraries](#).

## Step by step guide to your first widget

*(coming soon)*

# Saving Widget Data on the iPad

## localStorage

LocalStorage let you store data for any iBook.

1. Set data with  
`localStorage.userName = 'John Doe'`
2. Read data with var  
`userName = localStorage.userName.`



*Complex interactions could be interrupted at any moment ([image source](#)).*

With complex widgets, it may be desirable to store usage data. This allows the user to stop in the middle of an activity and resume it later.

The easiest way to achieve this is with localStorage.

## LocalStorage

The data persist beyond the current session. The user can close the book, turn off the device and the data will still be available the next time the user opens the book. You can check this with the simple demonstration provided in the interactive on the right. (The code for all widgets is provided at the end of the section.)

### localStorage.wdgt



Local Storage Demonstration

Please enter your name

John Doe (last saved: John Doe)

Save Data

*Writing data to a permanent store, using localStorage*

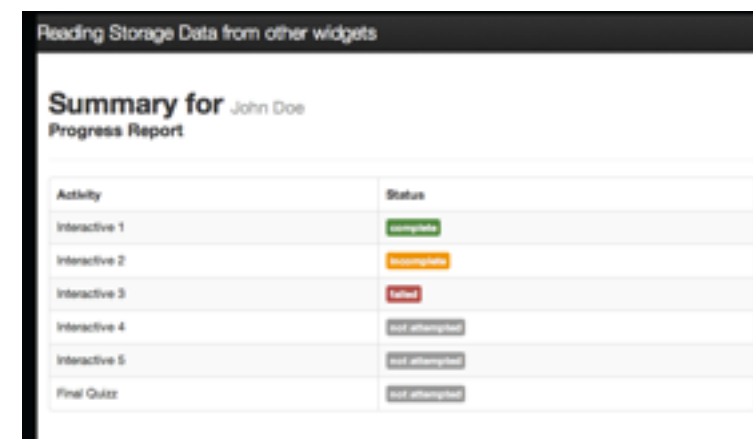
## Cross Widgets Data

Of interest, the storage is per book rather than by widget. This makes it easy to control how information specified in one widget can be used in another.

For instance, you could record activity completion in each interactive and present at the end of each chapter, a table that lists the activity completed and the ones yet to do.

Or you could present a summary view for all activities, with the question and the student answer, for the teacher to rapidly review and check for possible misunderstandings.

### sharingDataAcrossWidgets.wdgt



Reading Storage Data from other widgets

Summary for John Doe  
Progress Report

Activity	Status
Interactive 1	complete
Interactive 2	incomplete
Interactive 3	failed
Interactive 4	not attempted
Interactive 5	not attempted
Final Quiz	not attempted

*Reading data stored into localStorage in another widget.*

## Source Code And Documentation

Navigate to <https://github.com/widged/iwidgets-for-learning>

The widgets are in the widgets folder.

Click on the zip icon to download the project, uncompress the archive.

## Code for the widgets in this section

### *"Local Storage"*

```
<script language="javascript" type="text/javascript" src="js/jquery.min.js"></script>

<div id="activity">
  <h1>Summary for <span id="userName"></h1>
  <h2>List of Activity completed</h2><br/>
  Activity 1: <span id="statusActivity1"></span>
</div>

<script language="javascript" type="text/javascript">
var $name = $('span#userName');
$name.append(localStorage.studentName != undefined ? localStorage.studentName : 'current user');

var $activity1 = $('span#statusActivity1');
$activity1.append(localStorage.completedActivity1);

</script>
```

### *"Cross Widgets Data"*

```
<script language="javascript" type="text/javascript" src="js/jquery.min.js"></script>

<div id="activity">
  Please enter your name<br/>
  <input type="text" id="studentName" size="40" value="John Doe" /> <br/>
</div>

<script language="javascript" type="text/javascript">
var $activity = $('div#activity');
var $input = $('input#studentName');

if(localStorage.studentName) {
  $activity.append('<span style="font-size: 0.8em">(already saved: ' +
    localStorage.studentName + "</span><br /><br />");
}

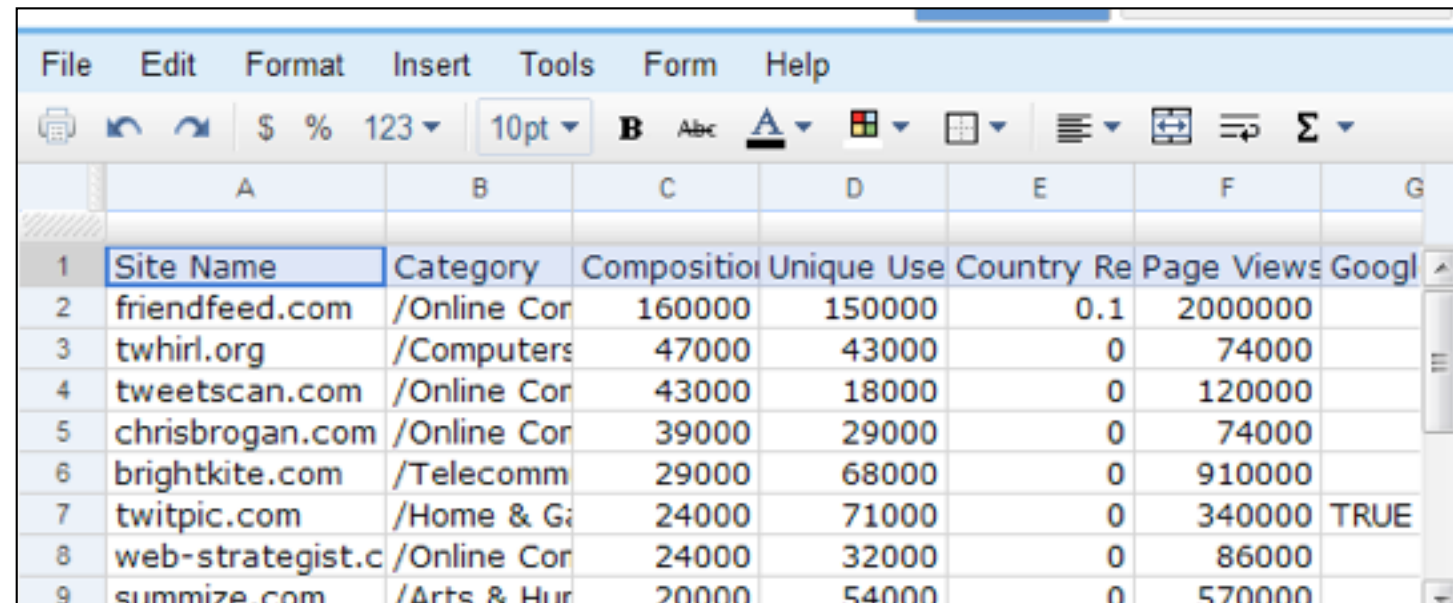
$activity.append('<button>Save Data</button> ').click(function () {
  var studentName = $input.val();
  localStorage.studentName=studentName;
  localStorage.completedActivity1=true;
});

</script>
```

# Saving Data Remotely

## Remote Data Storage

1. Low tech Google Spreadsheet read and write (no server setup required).
2. High tech Google Spreadsheet storage (server-based setup, PHP and Zend).
3. Learning Management System integration.



	A	B	C	D	E	F	G
1	Site Name	Category	Composition	Unique Use	Country Re	Page Views	Googl
2	friendfeed.com	/Online Cor	160000	150000	0.1	2000000	
3	twirl.org	/Computers	47000	43000	0	74000	
4	tweetscan.com	/Online Cor	43000	18000	0	120000	
5	chrisbrogan.com	/Online Cor	39000	29000	0	74000	
6	brightkite.com	/Telecomm	29000	68000	0	910000	
7	twitpic.com	/Home & G	24000	71000	0	340000	TRUE
8	web-strategist.c	/Online Cor	24000	32000	0	86000	
9	summize.com	/Arts & Hur	20000	54000	0	570000	

*Benefits of being able to access activity data across students.*

Without an easy way to save data, there is no good way for teachers to become aware of how much or how little a student understand.

Homework booklets could easily be collected and marked after class-time. This is not really a practical option with iPads, especially if the

school adopts a BYOD (Bring Your Own Device) setup.

At least three types of scenarios need to be considered.

First is the lone innovator. A teacher who has decided to take the plunge and invite the students



to bring their own devices but doesn't really have technical support. In particular, no access to any server. For that teacher, we introduce a low tech solution using google spreadsheets to read and write activity data. The type of functionality available is very much alike the one you would get from using Google Forms to create interactive questionnaires.

Second is a teacher investing in devices as part of a school-wide effort (whether it is a pilot limited to a few classes or truly school-wide; whether they provide them or require them). It is a planned effort, with budget for infrastructure, technical interventions and support. Server-based solutions are generally more appropriate as they provide more reliability, security, and control.

The third and last one is where the school had already made important investments in technology over the past years and would like the devices to integrate as seamlessly as possible in their existing infrastructure.

In what follows, we will cover only the first scenario. We will propose a solution that any teacher can implement, provided there is an online connection and access to google documents (<http://docs.google.com/>) is not blocked.

## Google Forms for Storage

We are taking advantage of the fact that if you create a google form and make it public access, then you can post data to that form with a GET request of the form

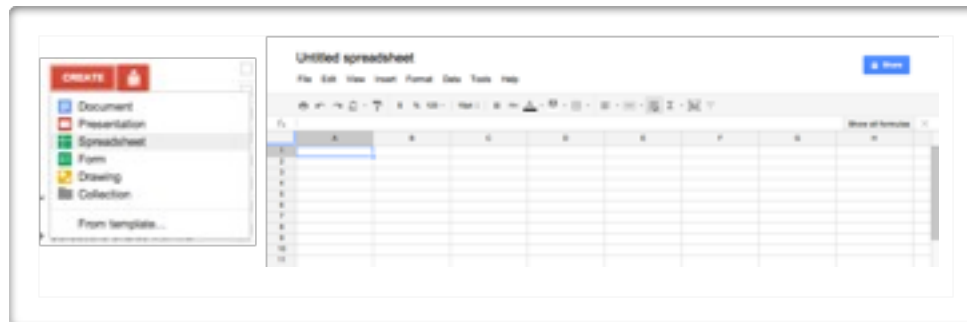
[https://spreadsheets.google.com/formResponse?formkey=\[yourKey\]&pageNumber=0&backupCache=&\[fields and their values\]](https://spreadsheets.google.com/formResponse?formkey=[yourKey]&pageNumber=0&backupCache=&[fields and their values]).

### Step By Step

Sign in your Google account if not already done and navigate to docs.google.com

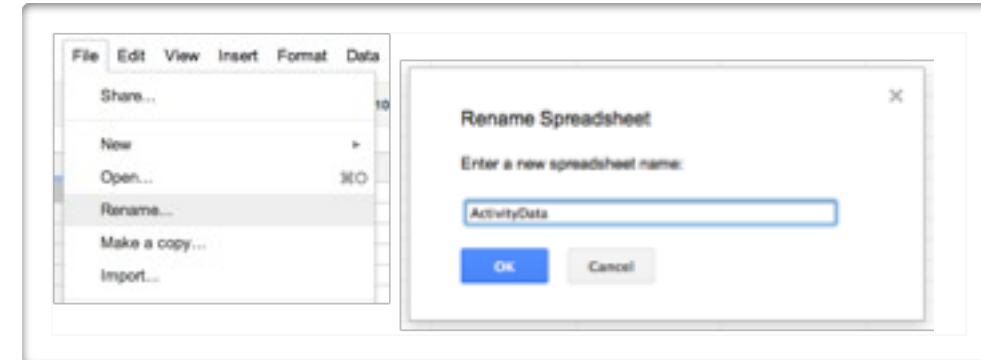


Click on “CREATE” and select spreadsheet.

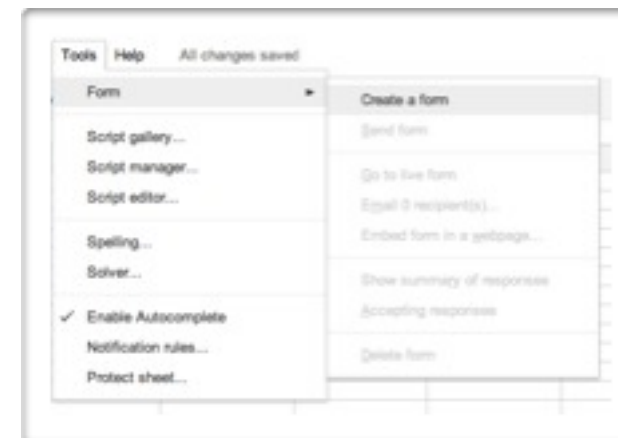


Immediately rename the spreadsheet to something convenient.

Here, we use “activityData”.



Now create a Form



Add Items until you have all the fields that you require. By default, all question types should be text. In this example, we define classUid (class unique id), userId (user unique id), activityUrl (activity url, will start with “x-ibooks-th” if ran from an iBook), activityUid (activity unique id), and activityData (activity data concatenated in one string).

Untick the option “Require [user] sign-in to view this form” to make it possible for anybody to access the form.

Before leaving the page, take note of the public url for that form

and write down the formKey part, we will need it later. For this example, the formKey is “dF9zaXhaVTPbHZCNkJEcUZkYUxqVUE6MQ”.

Now, you can post data to the form from javascript using an HTTP Request. The parameters entry.[#].single represent the column numbers of the spreadsheet that stores the data for that form.

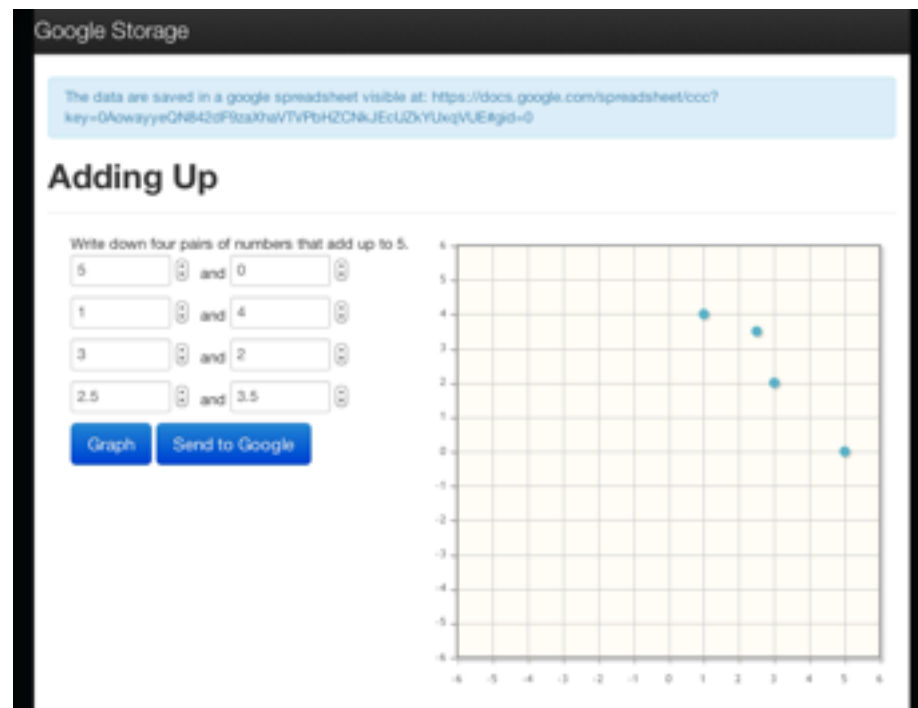
```
var plugin = {
  postData: function(data) {
    var param = {
      "formkey": dF9zaXhaVTPbHZCNkJEcUZkYUxqVUE6MQ,
      "pageNumber": 0,
      "backupCache": "",
      "entry.0.single": data.classId,
      "entry.1.single": data.user,
      "entry.2.single": data.url,
      "entry.3.single": data.uid,
      "entry.4.single": data.data,
    };

    var postUrl = "https://spreadsheets.google.com/formResponse" + "?" + $.param(param);
    var jqxhr = $.get(postUrl, function() {
      // alert("success");
    })
    .complete(function(data) { plugin.gsheetComplete(data); });

    gsheetsComplete: function(data) {
      console.log("data saved", data);
    }
  }
}
```

## Google Storage Widget

`googleStorage.wdgt`



*In this widget, a student is asked to provide data that match a condition (two numbers that add up to 5). A graph is provided on the right, to encourage him to detect incoherence and rethink his answers. When confident enough, he can send the data to the teacher.*

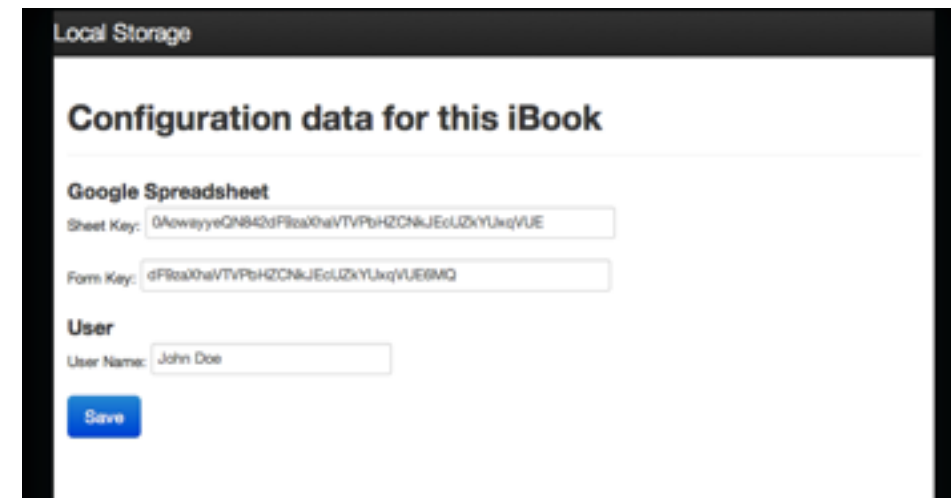
### Source Code And Documentation

Navigate to <https://github.com/widged/iwidgets-for-learning>

The widget file is in the widgets folder, “`googleStorage.wdgt`”.

Click on the zip icon to download the project, uncompress the archive.

`initializeGoogleStorage.wdgt`



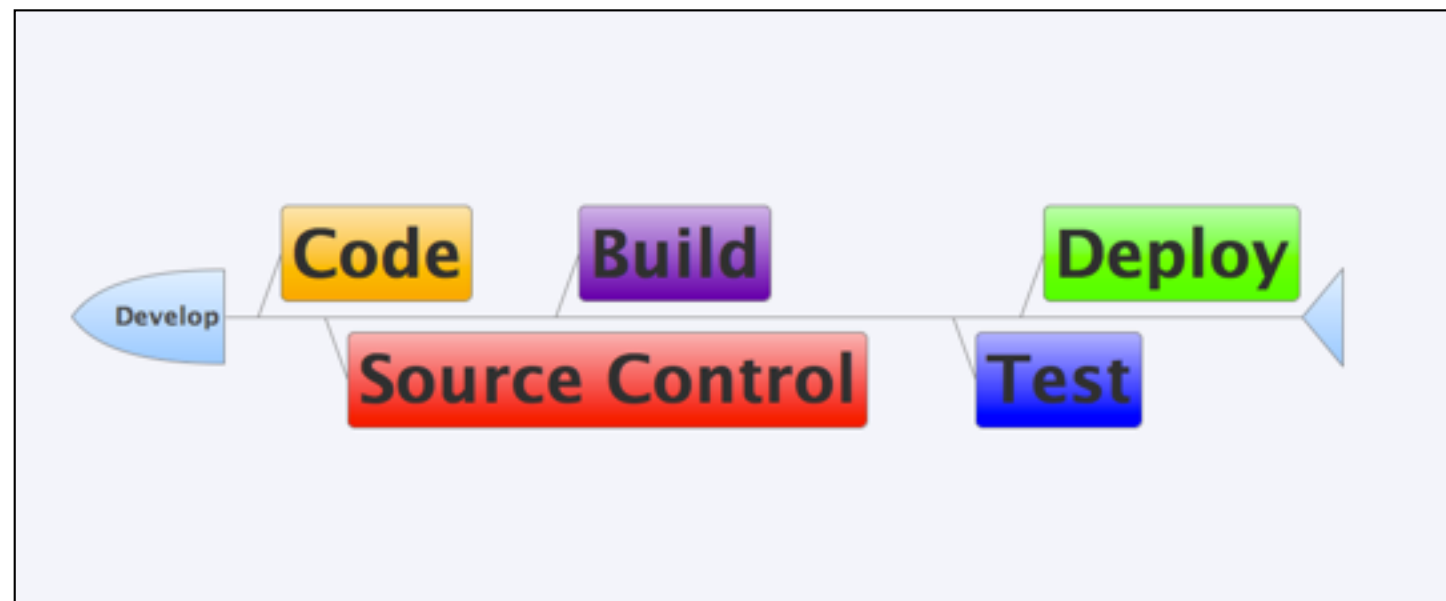
*Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do tempor incididunt ut labore et dolore magna aliqua.*

For the data to be sent to a Google Spreadsheet, the widget requires a Google formkey to have been stored in `localStorage`. Use the `initialiazeGoogleStorage` for this purpose.

# Development Workflow

## Setup

1. Debug on the device with Firebug Lite.
2. Best Practices
  - Module Pattern
  - Source control
  - Documentation
3. Share your code.



*Id malesuada lectus. Suspendisse potenti. Etiam felis nisl, cursus bibendum tempus nec. Aliquam at turpis tellus. Id malesuada lectus. Suspendisse potenti.*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer id dui sed odio imperdiet feugiat et nec ipsum. Ut rutrum massa non ligula facilisis in ulla mcorper purus dapibus. Quisque nec leo enim. Morbi in nunc nec purus ulla mcorper lacinia. Morbi tincidunt odio sit amet dolor pharetra dignissim. Nullam volutpat, ante a frin gilla

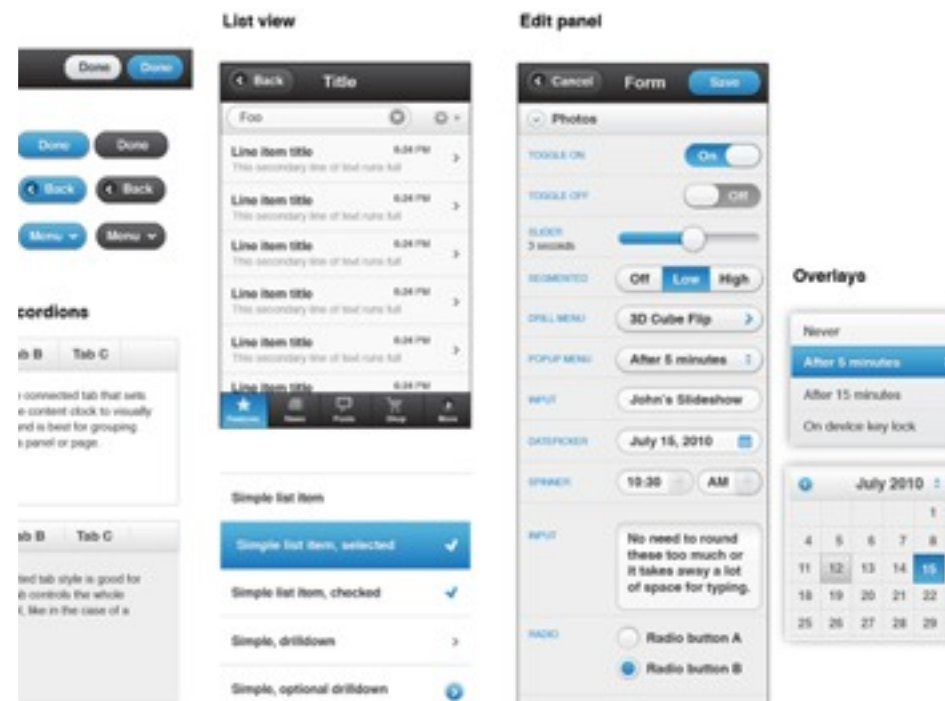
imp erdiet, dui neque laoreet metus, eu adipiscing erat arcu sit amet metus. Maecenas eu lorem nisi, id luctus nunc. Nam id risus velit. Sed faucibus, sem vel male suada blandit, quam tortor convallis odio, quis bibendum lorem felis quis mauris. Quis que euismod bibendum sag ittis. Suspe ndisse pell entesque libero et urna.

## Learn to Code?

### Use a Mobile UI

Numerous libraries and frameworks exist to let you create a visual interface that works well on devices, where the interaction is by finger touch rather than mouseclick.

#### Gallery 1.2 Mobile UI Libraries



jQuery Mobile - <http://jquerymobile.com/>

## Debug on the iPad

### Embedding Firebug Lite

A problem with developing for the iPad is that the testing and development environments can be quite different. The code may work perfectly when on the personal computer and then fail once loaded on the iPad. It can be a real challenge to figure out why the code breaks on the iPad.

Then sometimes you want to debug some javascript code that is completely specific to the iBook environment. The widget instance, for instance. It contains functionality specific to a dashboard widget.

It would be really handy if you could run firebug on your iPad or access the console information. Why not?

Not necessarily widely known, there is a version of Firebug that you can embed in a webpage. It is written entirely in Javascript and can run in any webkit environment. It's called Firebug Lite and you can get it from the firebug website -

<http://getfirebug.com/firebuglite>

You only need to ensure that you run firebug lite with a chrome set to div instead of iframe as iBooks don't allow you to embed content in iFrames. I couldn't figure out how to do this using the script options.

#### Interactive 1.3 Firebug Lite



Widget Running Firebug Lite on the iPad



You can force the div mode within the code. Locate the bit of code below. Add a line with type = "div" before the if.

```
// create the Chrome as a "div" (windowless mode)
if (type == "div")
{
    createChromeDiv();
    return;
}
```

### firebugEmbed.wdgt



*Firebug running in windowless mode, on the iPad*

## Debug Remotely

Another category of solutions is remote debugging. They can be, however, a bit more complicated to implement for the non-technical user as they require some server or websocket setup.

For instance, jsconsole

(<http://www.jsconsole.com/remote-debugging.html>).

## IDEs

### Dashcode

### Sublime Text

```
ln -s /Applications/Sublime\ Text\
2.app/Contents/SharedSupport/bin/subl /usr/bin
git config --global core.editor "subl -w"
```

### Mate

```
ln -s
/Applications/Mate.app/Contents/SharedSupport/bin/mate
/usr/bin
git config --global core.editor "mate -w"
```

### Aptana

## Licenses

```
curl -d{ "copyright": "[name], [url]", "url": "[url]", "theme": "flesch"
} http://[alias].mit-license.org
```

## Source Control

### Git

```
git config glogal
```

### Github

<http://pages.github.com/>

## Automated Builds

ANT

Share your widgets

Share your code

jsdo.it

github