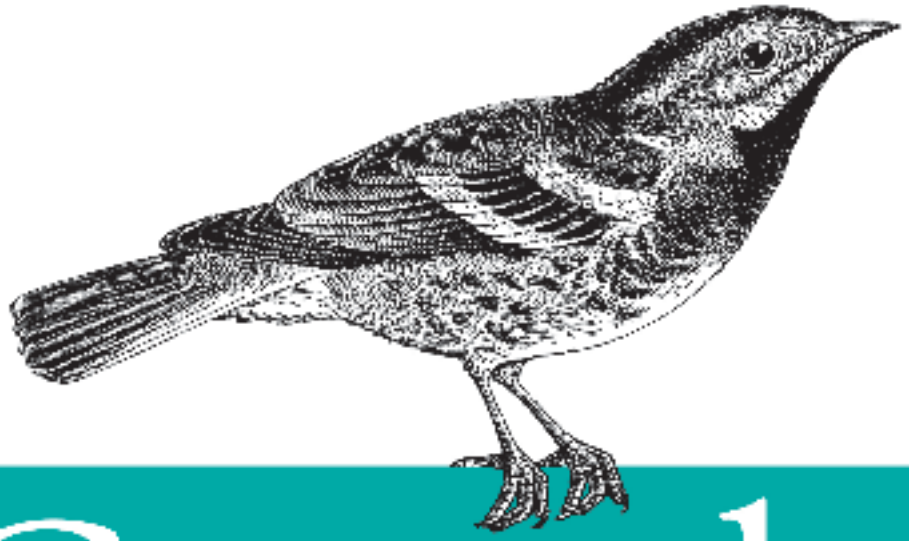
Adding Functionality to Your Google Apps



Google Script

Enterprise Application Essentials

O'REILLY®

James Ferreira

Google Script: Enterprise Application Essentials

How can you extend Google Apps to fit your organization's needs? This concise guide shows you how to use Google Script, the JavaScript based language that provides a complete web based development platform with no downloads, configuration, or compiling required. You'll learn how to add functionality to Gmail, spreadsheets, and other Google services, or build data driven apps that run from a spreadsheet, in a browser window, or within a Google Site.

If you have some Java experience, getting started with Google Script is easy. Through code examples and step-by-step instructions, you'll learn how to build applications that authenticate users, display custom data from a spreadsheet, send emails, and many more tasks.

- Learn Google Script's built-in debugger, script manager, and other features
- Create a user interface as a pop-up window, a web page, or a Google Sites gadget
- Use data objects and CSS to build effective product pages
- Automatically generate web forms from key values you specify in your Google Docs
- Create a database UI that works as a mobile app and Google Site gadget
- Use Google Docs and Gmail to create a document revision workflow

US $29.99 CAN $31.99
ISBN: 978-1-449-31852-9

Twitter: @oreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Google Script: Enterprise Application Essentials

James Ferreira

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Google Script: Enterprise Application Essentials
by James Ferreira

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Mary Treseler
Production Editor: Melanie Yarbrough
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Revision History for the First Edition:
2012-01-13 First release
See *http://oreilly.com/catalog/errata.csp?isbn=9781449318529* for release details.

ISBN: 978-1-449-31852-9

[LSI]

1327087181

Table of Contents

Preface

Introduction

If you are reading this book, there is a good chance you have heard of Google and its powerful office productivity suite, Google Apps. Google offers search, email, word processing, and hundreds of other cloud applications and services that can be available to the individual and can scale all the way up to massive corporations and governments. As one of Google's most popular services, Google Apps offers some of the best online office products available and is an excellent example of web-based applications that out perform legacy desktop software.

This book is about Google Apps Script, which is a service that runs from Google Apps, like Sites and Documents. Google Script is extremely powerful when automating many of the tasks required by day-to-day spreadsheet operations, but it also scales up to provide a complete application platform. If you are coming from a Microsoft Office direction, you can think of it as the macros for Google Documents, but unlike simple macros in MS Office, Google Script has a mature online editor with all the features one would expect in a development platform. Unleash Google Script's user interface capability and you can create entire data driven websites and applications that run across most modern browsers, including mobile.

In addition to the integrated development environment (IDE), Google Script comes complete with a manager for organizing scripts, built-in debugging, auto code completion, timed event triggers, and automated revisioning to name a few features. What really caught this author's attention was that everything is web-based. There is no need to download and configure a code editor or transport development files from computer to computer, wasting time resynchronizing files and reconnecting libraries. Simply sign into your Google account and start creating. Google Scripts are written in the JavaScript language version 1.8/ECMA 262 (3rd Edition), so there is no need to compile the code, making application development very fast.

With its own set of libraries, Google Script can interact with most of the services provided by Google, making it the Swiss army knife behind the mainline products. Other application building methods for accessing Google products like App Engine and the gData APIs, offered in many different languages, all require a place for you to develop

and deploy your code. With Google Script, you are building the code into the existing Google platform, and that equates a robust experience where your products are inherent in Google's legendary 99.9 percent availability. Because there is no need to have anything more than a basic Internet connected browser, development on this platform is something anyone can get started with, and there is no upfront expense. Google Script is not locked inside Google where it can only talk to Google servers, rather it can communicate through JDBC, JSON, SOAP, and has a `urlFetch` method making it very versatile when communicating across the Web.

At Google I/O 2010, a new feature called UiApp was unveiled, giving Google Script programmers the ability to build custom user interfaces that can run inside a spreadsheet window as a Google Gadget or completely independent in a browser. Talk about earth shattering, a cloud programing platform that can access just about any web-based service and has the ability to create AJAX style web pages? That is noteworthy. One year later, in 2011, additional improvements were added, giving Google Script a drag-and-drop visual editor. This feature reduces the amount of code writing and makes creating an application more approachable for power users with limited coding experience. To date, Google Script is the only way to gain full access to Gmail at the message level.

This book will focus on teaching you how to build powerful web applications using Google Script. It is laid out in sections that explain how the different parts of Google Script work and puts all these together in a series of fully functional applications that you can put to work right away.

Who Should Read This Book

This book is perfect for anyone who wants to extend what can be done with Google Apps but is not ready to dive into the complicated world of the Google Web Tool Kit and Java APIs. You don't have to be a webmaster or programmer to grasp the concepts in this book. Google Script takes care of server configuration, gives you a place to save your projects and allows you to start developing immediately. This book is approachable by anyone with basic coding skills and a fundamental understanding of JavaScript. If you have never used JavaScript, I recommend having a copy of *Head First JavaScript* (O'Reilly) close at hand to help you through concepts like variables, arrays, and objects. All the application examples have highly detailed explanations, so if you are a Google Apps power user, you should not have difficulty grasping the content in this book and writing incredible applications using Google Script.

What You Will Need

You will need a web browser (I recommend Chrome) and any type of Google account. That's it! Google Script is a completely web-based solution that is free and ready for you to start programming today.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Google Script: Enterprise Application Essentials* by James Ferreira (O'Reilly). Copyright 2012 James Ferreira, 978-1-449-31852-9."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9781449318529

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

PART I

Understanding Google Script

CHAPTER 1

First Steps in Google Script

What is Google Script and why should you use it to build a web application? Simply put, Google Script is an easy way to figuratively glue Google and other web services together to form one powerful interactive web application. Just ahead, a more in-depth explanation of Google Script and how to use it to enhance existing Google Apps. You will also learn the basics of building an application. This first chapter should get your feet firmly planted on the ground floor of the Google Script development platform and demystify its usage.

Google Script Is...

Google Script is a coding and application development platform built into Google Apps, enabling you to add functionality to spreadsheets, Gmail, Sites, and other services from Google. For example, if your spreadsheet needs a menu item in the tool bar for creating a pivot table, you would write a Google Script that adds it to the menu and performs the task. Because Google Script serves as a backend to other Google services, you will need a spreadsheet or Site to hold the scripts you create. This does not mean that your script will be limited to the spreadsheet or site containing the script. On the contrary, a Google Script can run as a web service without the user ever knowing there is a spreadsheet involved in the interaction. This book will focus extensively on the concept of using Google Script to build applications that present themselves as web services running independently of other interfaces. You will learn how to use Google Script to build apps that run from a spreadsheet, in a browser window or within a Google Site, and from the user's perspective, they will appear to be complete applications such as you might expect when using a web service like Picasa or Amazon.

There are some real advantages to having your scripts (i.e. applications) stored in one of the Google Apps services. Primarily, security is already built-in, meaning you do not need to worry about implementing that component into your application as you would if it were running on a legacy web server needing patches and constant monitoring for malicious attacks. As part of Google Apps, Google Script also allows you the same collaborative development abilities that are part of the Apps suite. What is truly exciting

about Google Script is that it is a 100 percent web development environment that requires no transferring of files from computer to computer, backups, revision control, uploads to a production server, updating the development software, or many of the other tedious aspects of development that get in the way of actually writing applications. These parts are all built-in, allowing you to focus on creating products for your business, school, club, or anything that needs to run on the Web.

The UiApp service, which stands for User Interface App, was released in early 2010 as a way to allow developers to collect user input that could be sent back to a script for processing. UiApp uses the Google Web Toolkit (GWT) Widget set as the framework for building an interface. Widgets allow you to create such things as a text box and a Submit button but also more complex items like flex tables and list boxes. Everything you see in a Google Script UI are widgets cleverly arranged within a frame in the page. The only other elements—panels—are the containers that hold all your widgets, and that is truly all there is to the visual part of a Google Script UI. Part III lists every Google Script Widget, including example code. If you are familiar with GWT, you will be right at home creating UIs in Google Script. Never heard of GWT? Don't worry, this book will have you crafting widgets and transversing AJAX with little pain and only a mildly slopping learning curve.

What You Will Get From This Book

By the time you get to the back cover you will have learned all the necessary elements that go into building enterprise applications using Google Script. With this knowledge under your belt, you will be able to create your own applications and take full advantage of your Google hosted services. Your apps will have the ability to recognize and authenticate users and carry out tasks such as displaying custom data from a spreadsheet, data entry, sending emails, and so much more. Have a look at Part II to see the kinds of applications we will be building and let your imagination flow.

Getting Started

Enough preamble, let's dig in!

For the most part, we will be building our scripts in the Google Documents service. To get started with the examples in this chapter, load up Google Docs, *http:docs.google.com* or *http:docs.google.com/a/<your domain>* if you are using a Google Apps account. Create a new Spreadsheet.

In your Spreadsheet, click the Tools menu, then select Script editor, see Figure 1-1.

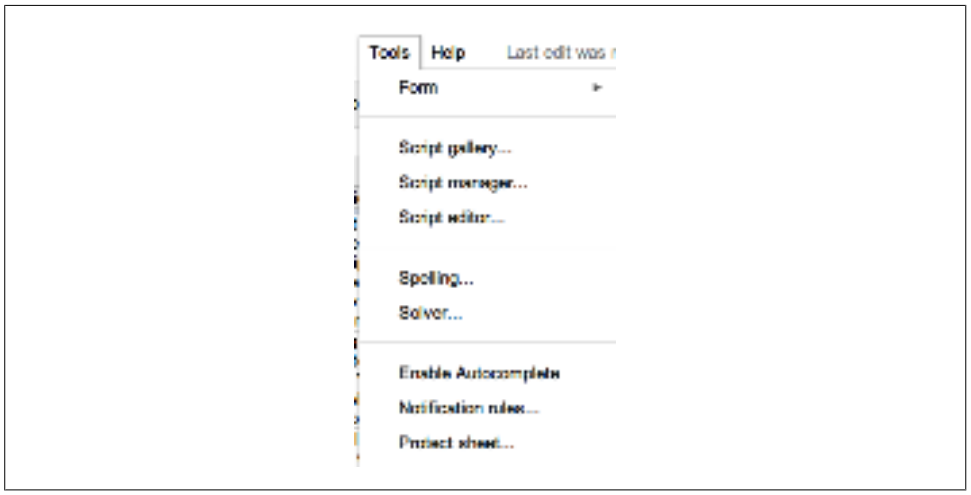


Figure 1-1. The tools menu has several options for managing scripts.

The Google Script editor will open as a new window, see Figure 1-2.

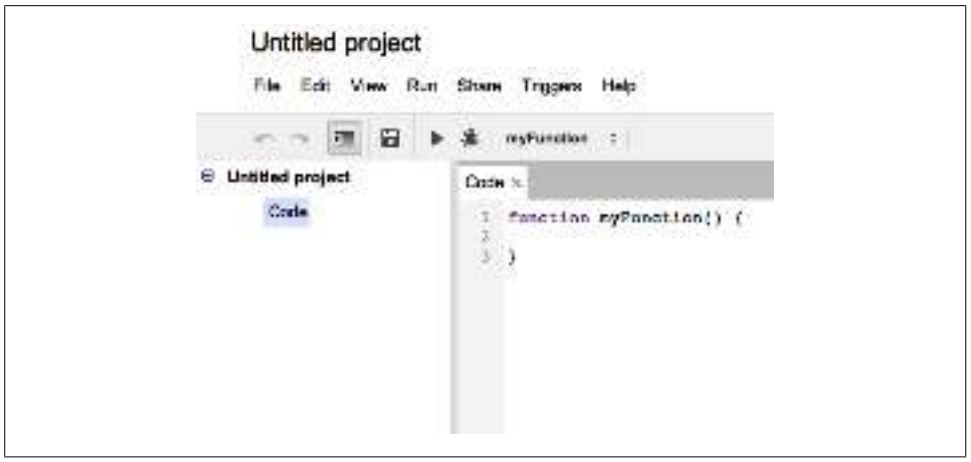


Figure 1-2. The Google Script editor is a full IDE running in the cloud.

Looking Around the Editor

Before writing your first script, let's take a look at some of the features in the Google Script editor. First off you will notice that it looks much like what you already know from Google Documents. Under the File menu are the typical Save, Delete, Rename, Open, Create New File or Project, etc. Also like many of the other Google Apps services, there is a Revisions feature that will allow you to turn back the clock to a point when your code *was* working. (Not that we ever need such features...) Seriously, we often go

down the wrong road during development, and revisions can save you hours of trying to get back to a known good point. When launched, a pop-up Revisions box will show what the code looked like in the version you selected.

In the File menu, there are two very important options: "Properties" and "Build a User Interface." Properties makes it possible to store a limited amount of information in Key:Value pairs for use by your script at runtime. Properties can be edited in the box that pops up after clicking the "Properties" option in the File menu, or by using the Properties Service right in your code. Many of the apps in this book will need to sign into non-Google services, and Script Properties is a great place to store something like a password. The Build a User Interface, or the GUI Builder, is one of the tools we will use to create a user interface. This will be covered throughout the book, so just note that this is where you launch it.

Nothing very exciting in the Edit menu, other than "Find Selection," which also incorporates Replace, a good way to globally change the name of a variable. Moving onto the View menu, there are some important points: Execution Transcript and Logs. When a script is run from the editor, the Execution Transcript will list each command as it is run. Using the Execution Transcript, you can see the order that the code is executed, which is helpful in debugging. "Logs" is used along with the Logger Service and allows the writing of information and other notes as a way to track information. This was very useful before the debugger was added and can still be a big help when testing code. I want to reiterate that these features only work from the Editor and will not be of much use debugging in the UiApp when it is run from the browser. Don't worry, there is a whole section to help you debug like a pro.

A quick example of using the Logger:

```
function myFunction(){
  Logger.log('A test of the Log');
}
```

Click Run, and then check in the Log under the View menu.

The Share menu is where access to the script is set and where you will find the Publish option that makes displaying a UI possible. The publishing feature is covered later in the chapter.

Triggers are the automation component that have the ability to run a script at specified times or in certain events, like the submission of a form or when the spreadsheet is edited. Triggers are very useful for tasks such as backing up information at 1 A.M. so you get credit for working hard while fast asleep.

That's about it for the menu. Figure 1-3 shows a few buttons that explain themselves and make for easier access to the most common features.

The Debug option next to Run will bring up a window at the bottom of the code window and show the values of your code as it is executed. It has features for setting break points, stepping in and over parts of code, and will make developing non-UI parts of

Figure 1-3. Various ways to access a script's code.

your code in each case. The user documentation on the Google Script website explains in detail on [usage of the debugger](#).

Three Ways to Create a UI

There are three ways to create and display a user interface (UI) in Google Script. The first way is in a spreadsheet as a pop-up window, the second, as a web page and the third as a gadget in a Google's web page.

As you work through the chapters, please note that the use of the code in each type of UI is the same and will only be described once as it is first introduced. It would be a good idea to go through all the different UI types to avoid confusion about a certain topic and to gain an understanding of why and why a certain UI type would work better for your application.

Hello UiApp Spreadsheet Integrated

Now that you know your way around the Editor, it is time to write your first script. Keep in mind that all Google Scripts are written entirely in JavaScript and there is no HTML needed to generate a UI. The first type of UI is called "Integrated" because it is going to display as a pop-up window from a spreadsheet. The term "integrated" comes from requiring a spreadsheet to display the UI, but this does not mean that any certain type of UI is more or less integrated than another. It is simply to give you a reference of what we are discussing because the code to display each type differs slightly.

Open the Editor, click File, and select New. In the new script, remove all of the boilerplate code, and add the following code:

```
function helloWorld() {
  var ssapp = SpreadsheetApp.getActiveSpreadsheet();
  var application = UiApp.createApplication().setTitle('Your Title');
```

```
  //TODO add your code here
  mydoc.show(application);
}
```

Click Save, and name your script "Hello World Integrated". Now click Run. Switch your browser window to the Spreadsheet view, and you will see an empty UI window with the title "Hello World" at the top.

The core of all Integrated UiApps and the components to make the UI display:

```
UiApp.createApplication();
show(application);
```

Diving into the code

All Google Scripts start with a function; when using the integrated UI, you can name your function almost anything you like:

```
function <name>()
```

The function names `doGet`, `doPost`, `onEdit`, `onInstall`, and `onOpen` are special reserved functions and should not be used as names for custom functions you create that are not performing these specific operations.

Here we use the Google Script Spreadsheets Service to create an object called "mydoc" that represents the current spreadsheet:

```
var mydoc = SpreadsheetApp.getActiveSpreadsheet();
```

The following line creates the UiApp object, which contains all the methods for creating UIs:

```
var app = UiApp.createApplication();
```

When using a spreadsheet, you need to insert the script into it by using the `show` method:

```
mydoc.show(application);
```

Time to get something for you to look at. Find the line starting with `//TODO` and replace it with the following code block:

```
var helloWorldLabel = app.createLabel('My first Google Script UI');
app.add(helloWorldLabel);
```

To display information on the UI, you will need a widget. In this case, a simple label will be created, which is a widget that only takes text in its argument and shows inline. Labels can be styled with CSS to give you all sorts of creative options. To make the Label widget appear in the UI, it needs to be added to the UiApp object by calling the `add` method and using the variable name in the methods argument.

Run the script and switch your browser to the window containing the spreadsheet. Figure 1-4 shows the UI displayed in a spreadsheet. There are endless possibilities for why you might want to pop up an interface for the user: data entry, choosing information from another service like Contacts, or running a script requesting additional information are just a few examples.

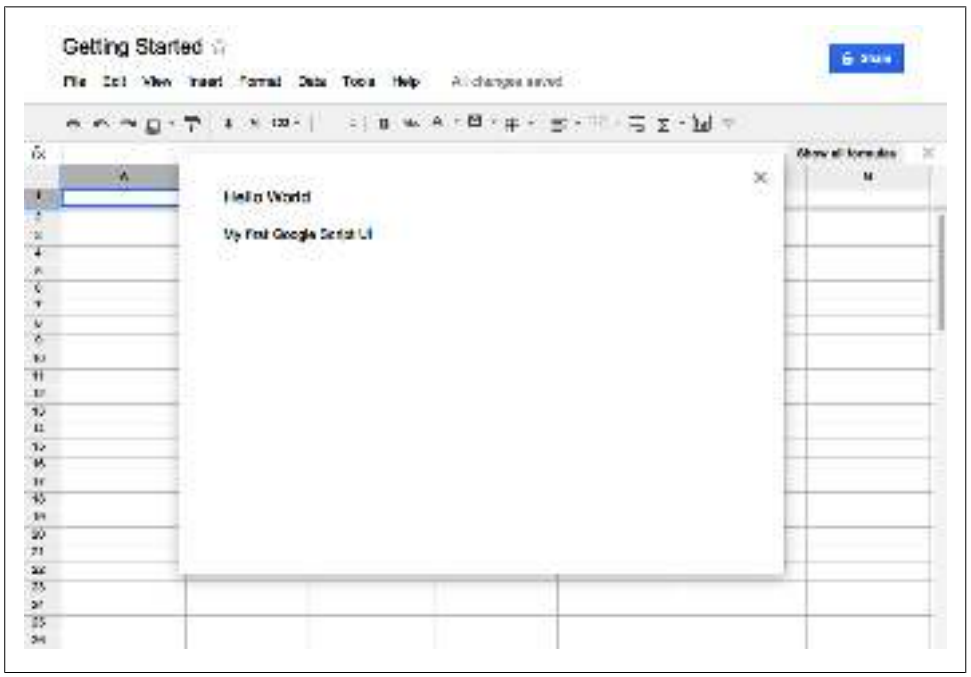


Figure 1-4. UI running inside a spreadsheet

Integrated Versus Standalone

One of the most exciting features of the UI Service is that it can run all by itself without the need to open a spreadsheet. This is accomplished by publishing the script, which creates an access point through a special Google URL. There are several options when publishing, such as restricting access to anyone but you or opening up the UI so that anyone visiting the URL can run it. Publishing does not, however, allow visitor access to your code; that is controlled by the sharing in the spreadsheet or site. This means you can create your application and the code will stay safely secured.

One important thing to remember is that a script running as a published web service will always run under the creator's account and will have access to the services to which he has granted access. Therefore, if your script lists all the emails in your Inbox and you make it public, anyone visiting the URL will see your Inbox not rather than his own.

Running the script as your account can be a benefit because you can set the spreadsheet sharing to limit access and then control what a UI viewer sees about the data while still allowing input into the spreadsheet. There will be more on this concept later when we start putting together real-world apps in Part II.

One limitation to having the script run as you, the creator, is that you will not be able to directly access a user's account from the built-in classes. For example, if your application needs to access the user's Contacts service, it will not work that way. You don't have that user to user access in Google Apps, so it does not work in Google Script either. Later in the book we will cover how to handle this problem using two-legged OAuth.

In the Integrated UI spreadsheet version of a Google Script, your users have the same access as the spreadsheet, so unlike the standalone UI, where a script runs under your account, the Integrated UI will run as the user's account.

While I present these differences of the two UiApp styles as hurdles, there are some very good reasons to have access restricted in this way. Fortunately, these security features don't limit us in building apps, but they add certain complexities that need to be considered.

Creating a Standalone UI

The second type of UI application is referred to as standalone because the UI is accessed from a special URL hosted on Google's cloud. There is no need for a spreadsheet to use the standalone UI but you can build this type of UI in a spreadsheet or in any of the Google services where the script editor is available. The URL can be made public allowing DNS mapping to your domain. For example, *http://Your_Great_App.domain.com*.

Open the script editor, select File➡New. Replace the `myFunction` code with:

```
function doGet(e) {
  var application = UiApp.createApplication();
  //Your Code
  return application;
}
```

The standalone UiApp will be the most commonly used style in this book because it has the ability to run as a gadget in a Sites page or as its own independent page.

The first difference from the Integrated UI version is that we must have a `doGet` function for the UiApp window manager to grab when the standalone URL is loaded in a

browser. This is analogous to the entry point you might use in GWT. `doGet` is the starting point for loading visible elements in the standalone UI.

The visual part of the UI is created using the UiApp class that will create an object for display. You don't need to have a widget to present visual information to the user if your script's purpose is to simply perform a task when the URL is loaded. This is accomplished by passing values in the URL parameters and will be covered in Chapter 8. Even if you are only collecting information and don't have content to display, it is good practice to put a message of some kind on the page for the user:

```
var application = UiApp.createApplication()
```

To get the UI to display a widget on the page you must return the UiApp instance:

```
return application;
```

Think of it this way, when you load the UiApp's URL, a Google server hosting your code looks for and runs the `doGet` function. If there is no return value for `doGet`, you will not see anything on the page. If you have not yet guessed, all Google Scripts run on the server side and the interaction you see is accomplished through remote procedure calls (RPC).

To display some text on the page, create a label and add it to the `application` object in the same way you did for the Integrated UI type:

```
var label = app.createLabel('Nothing but Web');
app.add(label);
```

That is all there is to making a basic UI page. Click Save, and name your script **`Stand-alone Web Service`**.

Publishing a script

To make the UI available, it will need to be published to the Web. This is Google's way of saying that a special URL has been created and the UI will be served from there. It is not public unless you make it so.

Click the Share menu and select "Publish as service..." as shown in Figure 1-5.

Figure 1-6 shows the "Publish as service..." option where you can choose the level of access you would like visitors to have. The options differ depending on whether you are using a Google or Google Apps account. The last setting—"Allow anyone to invoke the service"—will give access to any visitor that has a Google account. When you check this option, an additional choice will appear allowing you to select "Anonymous access," meaning the script is fully open to the Web and no sign-in is required.

The "yada yada" Google refers to is a serious warning. When you build a standalone application, it will run as you and it will have access to anything you have given it permission to see. For example, be careful not to publish the contents of your email Inbox to everyone on the Web.

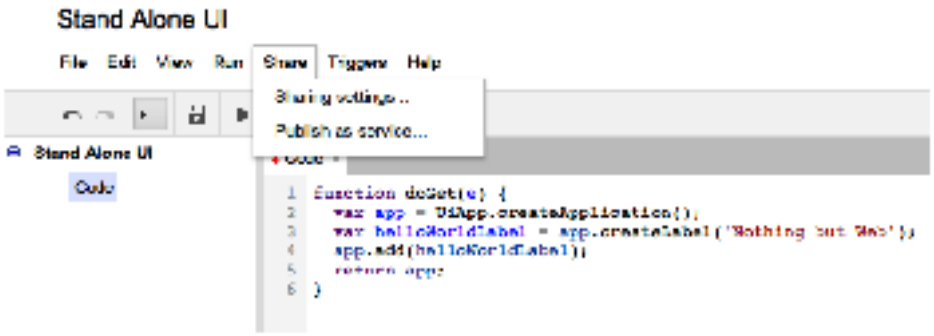


Figure 1-5. The Share menu has an option to allow others to add your script.

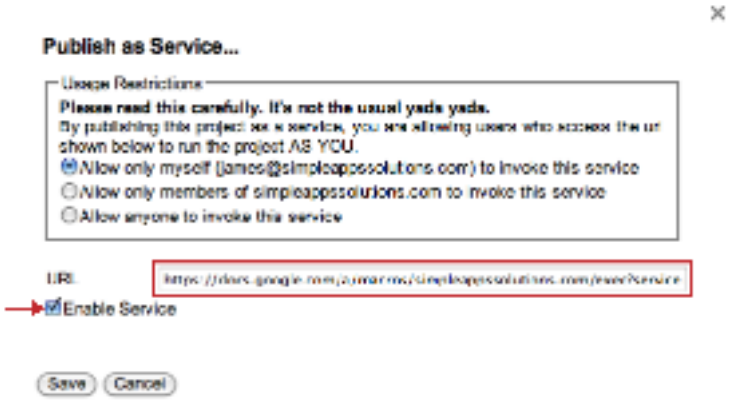


Figure 1-6. Settings on the publish page allow a number of sharing options.

Once you have chosen the level of access, check the Enable Service box and copy the URL in the box. This is the special URL for the page where this UI is being hosted. If you forget it, simply open this dialog and copy again. Click Save and open a new tab in your browser. Paste the URL in the address bar and load the page. You should now see your UI displaying the text from the label created earlier.

Congratulations! You have just created your first Google Script UI Web service. Not much of a "service" yet, but as you can see, it takes very little effort to get an application published out to a web site to use. What you might have missed is everything that's going on here, which is a good thing. Because, you do not need to create an HTML page, or figure out how to FTP to a web server somewhere to upload files. To that extent, you also didn't need to purchase and install a web server or buy a domain. Google Script gives you the ability to write your application entirely in JavaScript and then takes care of the rest of the details. I don't want to say it gets easier from here, but this is the foundation; after this, the functionality of providing a service has more to do with