# Hello Again

In this project, you will write your first Chrome Application. You will set it up the same way as the extension, but after you are done, you will launch it a little bit differently.

## Core Concepts

* Javascript functions
* HTML forms
* Installing a Chrome Application
* Finding and fixing mistakes with the Javascript console

## Getting Started

By now you should have already downloaded Chrome and gotten set up with a text editor (If not, see the "Hello World" packet). Create a new folder (call it whatever you want) for this packet, and as before copy each of the files below into that folder.

## Source Files

*manifest.json*

This manifest file is similar to that for Hello World, but now instead of a browser-action we are telling Chrome that this is an "app," and when Chrome launches the app, it should load "hello.html."

{  
 "name": "Hello World Again",  
 "description": "Say hello to whomever.",  
 "version": "1.0",  
 "manifest\_version": 2,  
  
 "app": {  
 "launch": {  
 "local\_path": "hello.html"  
 }  
 }  
}

*hello.html*

When the application loads, Chrome will display this HTML in the main browser window (not in a pop-up).

<!DOCTYPE html>  
<html>  
<head>  
 <!-- Tells the browser to run the code in hello.js. -->  
 <script src="hello.js" type="text/javascript"></script>  
</head>  
<body>  
 <!-- A big header at the top of the page, in black, that says "Hello, world!" -->  
 <h1 style="color:black" id='chatroom-title'>Hello, world!</h1>  
  
 <!-- This tag defines an HTML form, which can be used for getting user input. -->  
 <form id="chat-form">  
 <!-- A text box within the form. -->  
 <input type="text" name="message" id="message-input">  
 </form>  
</body>  
</html>

*hello.js*

The JavaScript file tells the browsers what actions to perform and when to perform them. You can think of this as the "brain" of the application.

The lines that begin with // are comments. They have no functional use, other than to inform the reader. You do not need to type in the comments.

// "document" is the variable that refers to the HTML structure.  
document.addEventListener('DOMContentLoaded', function() {  
  
 // This gives us a reference to the HTML element whose id is "chat-form."  
 var form = document.getElementById('chat-form');  
  
 // This gives us a reference to the element whose id is "message-input."  
 var messageInput = document.getElementById('message-input');  
  
 // This gives us a reference to the title.  
 var chatTitle = document.getElementById('chatroom-title');  
  
 // An "EventListener" calls the function passed in when the "submit" event  
 // is detected on the form.  
 form.addEventListener('submit', function(event) {  
 event.preventDefault();  
  
 // The variable "name" now refers to the text the user typed into the  
 // message input box in the form.  
 var name = messageInput.value;  
  
 // Here we change the text of the title of the page.  
 chatTitle.textContent = "Hello, " + name + "!";  
  
 // This line sets the value of the input box back to the empty string.  
 messageInput.value = "";  
 });  
  
// Don't forget to make sure your parentheses and brackets line up!  
});

## Running your Chrome Application

Add your application the same way as you did the Hello World extension from chrome://extensions in your Chrome browser (you'll still click "Load unpacked extension..." even though it is an application.). Again, try to fix any errors that you see pop up.

Once your application is installed, open a new tab in Chrome and select the "Hello Again" box. Try typing "D-code" in the input box and hit enter. If the top of the page now says "Hello, D-code!" and the input box is empty, congratulations! You've made your first Chrome Application!

## Fixing errors

There's a good change your application won't work the first time you try it. If this is the case, now you have to find and fix the mistakes.

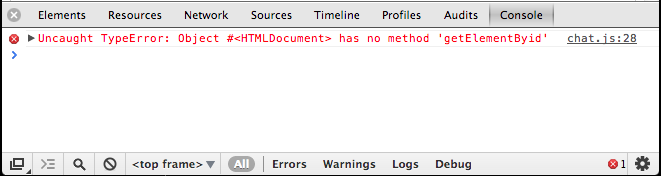
If there is something going wrong with your code, Chrome will try to tell you by printing our errors and warnings. If you know where to look, these can be very helpful.

### Javascript Console

Chrome has a Javascript console where errors and warnings are displayed. You can access it through the menu options:

1. Click "View" in the Chrome menu bar at the top.
2. Scroll down to "Developer" (with the arrow next to it), and another menu should open to the side.
3. Click "JavaScript Console." A window should appear at the bottom of your browser.

This is an example of what an error looks like the Javascript console:

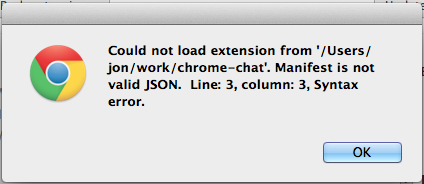


Javascript Console

The console is reporting that there is an error in chat.js on line 28. The error is that we are trying to call a function called getElementByid, but the correct name of the function is getElementById. Notice that the I didn't capitalize the 'i' in Id correctly. Javascript is sensitive to capitalization, and this is what is causing the error.

### Line numbers

An error in the console will contain a relevant line number and file name. Usually the mistake or typo will be on this line, but not always. Sometimes a mistake in an earlier piece of your code will trigger an error in a later part. For example, I received this error when trying to load my Chrome extension.



Manifest Error

and here is the beginning of my manifest:

{  
 "name": "My Application"  
 "description": "This app that has a bug in it.",  
 "version": "1.0",

The end of each definition in the manifest file needs to end with a comma. Notice that I forgot a comma on the end of line 2. This is the cause of the error, even though Chrome is reporting the error on line 3.

### SlimText warnings

The SlimText text editor will give you warnings are you are typing. These warnings show up to the left of your code.



Slimtext warning

In this example, SlimText is warning us that we have mispelled the world funtion on line 5.

SlimText will also apply colors to your code as you type it. You can use these colors as clues to find mistakes. Notice how SlimText applies a different color formatting to the world function on line 1 versus the incorrectly spelled functon on line 5.

### What does an error mean?

Over time, you will get better at understanding what the computer is trying to tell you in its error messages. In the beginning, however, it can be confusing and frustrating.

If you cannot understand the meaning if an error, copy and paste it into Google. Google can be your best friend when you are coding.

At this stage there is no problem you will encounter that hasn't already been solved by thousands of other people. The answers are all on the Internet. Keep googling!

### Common errors

Everything must be spelled exactly right, including capitalization.

All open parenthesis, quotation, or curly brace should have a corresponding closing parenthesis, quotation, or curly brace. Very often, "unbalanced" curly braces will cause errors that can be hard to track down.

### What if there are no errors?

Sometimes the computer will not generate any errors, but it still won't do what you want. If this happens, you can generate your own messages into the Javascript Console to help you figure out what is broken. To do this, use the console.log function.

For example:

console.log('Hello 1');  
  
document.addEventListener('DMContentLoaded', function() {  
 console.log('Hello 2')  
  
 // more setup goes here..  
});

When I run this, I will see the following output in my Javascript console:

Hello 1

Since I can see the Hello 1 line, I know that the first part of my Javascript code is running. But the Hello 2 line is never displayed in the console. This leads me to believe that there is something wrong on the line above it, on line 3.

Indeed, I have mispelled the event name, DOMContentLoaded. I have left out an O. This is the kind of mistake that won't generate an obvious error, so you need to use console.log to track it down.

## See what you can do!

Can you think of any features to add to this application?

* Add another text box to change the greeting from "Hello" to something else.
* Try changing the color of the h1 element. What other styles can be added to this element?