

Chrome Packaged Apps are evolving to do more, work seamlessly offline, and give developers more control of their user interface. Join us for a dive into the developer preview of the next evolution of Chrome Apps and learn how you can start doing things you never thought possible using web technologies



About Erik Kay

Erik manages the Chrome Apps, Extensions and Native Client teams. [View full profile](#)



About Mihai Parparita

[View full profile](#)

Mihai is a Chrome Apps Tech Lead. He joined Google in 2004, and has previously worked on Google AdWords and Google Reader.



The Next Evolution of Chrome Apps

Erik Kay - Engineering Manager
Mihai Parparita - Software Engineer

Remember Google I/O 2011?



Ecosystem Is Thriving

Native Client

Amazing Games

Web Store

750 MILLION
APP INSTALLS



#io12

Apps Evolution



Breaking Out Of the Browser

- Launch from outside of the browser.
- First class OS windows (alt-tab, etc.)

Demo



Chrome OS

Extensions

☐ Developer mode

Apps IO 2012 1.1

Chrome Apps IO 2012 presentation.

☐ Allow in incognito [Reload \(Ctrl+R\)](#)☒ Enabled

MiniCodeEdit 0.1.11

A very small code editor.

☐ Allow in incognito [Reload \(Ctrl+R\)](#)☒ Enabled

Document viewing 2.2.31.4

View Word, Excel, and PowerPoint files on Chrome OS. Formats supported: .doc, .docx, .xls, .xlsx, .ppt, and .pptx files.

☐ Allow in incognito [Visit website](#)☒ Enabled

Games 1.1

Discover new games in the Chrome Web Store! This app is a quick shortcut to the Games category in the web store.

☐ Allow in incognito [Visit website](#)☒ Enabled

Google+ 1.0.1.424

Share the right things with the right people.

☐ Allow in incognito [Visit website](#)☒ Enabled

Music 1.1

☒ Enabled

ome OS

Extensions

☐ Developer mode



Apps IO 2012 1.1

Chrome Apps IO 2012 presentation.

☐ Allow in incognito [Reload \(Ctrl+R\)](#)

☒ Enabled

Search



Chrome



Chrome Web



Apps IO 2012



Files



Games



Gmail



Google Calen



Google Drive



Google Maps



Google Search



Google+



MiniCodeEdit



Music



Scratchpad



YouTube

ome OS. Formats supported: .doc, .docx, .xls,

This app is a quick shortcut to the Games

☒ Enabled

☒ Enabled

☒ Enabled

☒ Enabled

☒ Enabled

Enhanced User Interface

- Full control over multiple windows.
- Custom window frame without browser chrome.

Enhanced User Interface

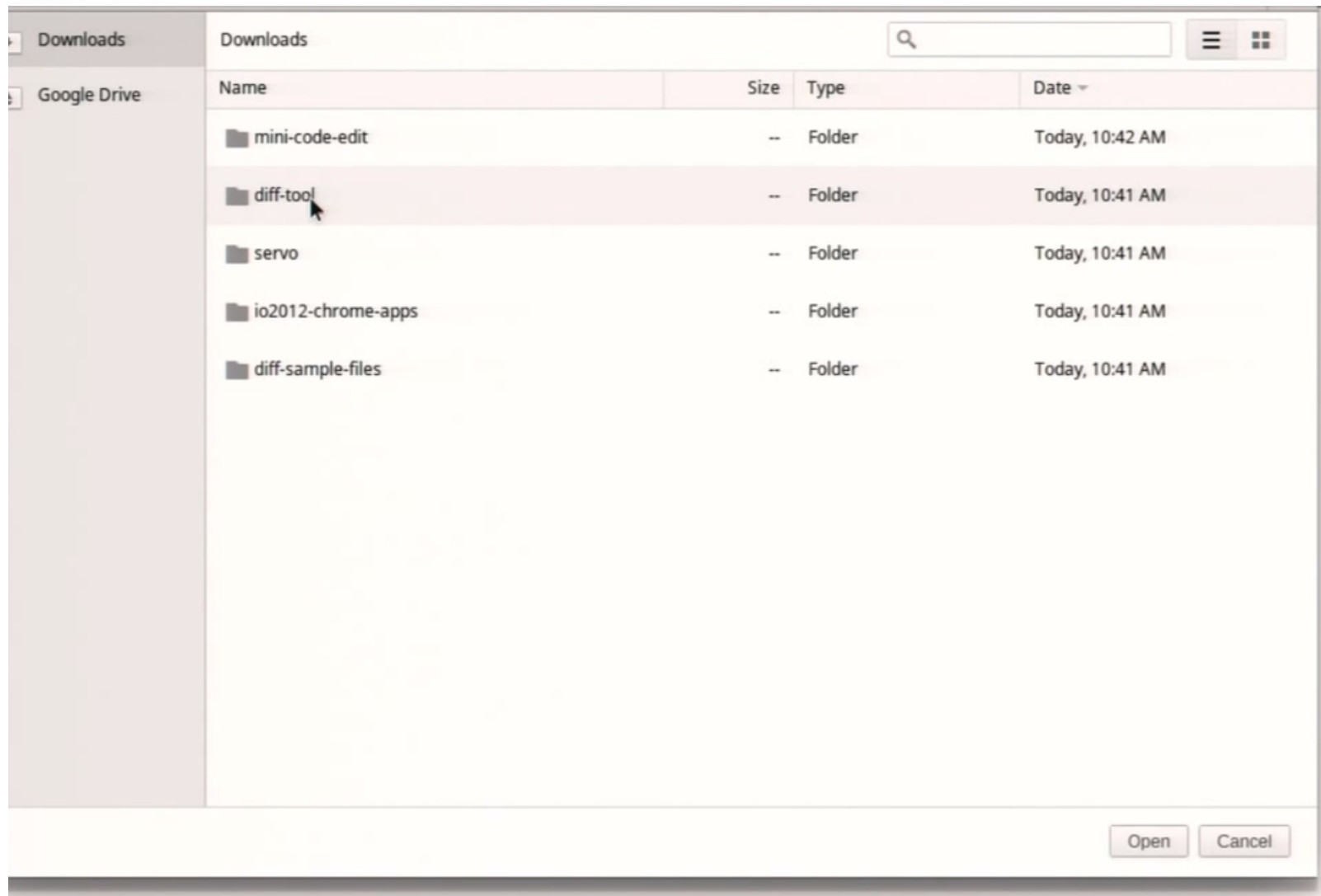
- Full control over multiple windows

Original	Bizarro World
x: 160	x: 421
y: 176	y: 176
width: 256	width: 256
height: 256	height: 281
Minimize me!	

Demo

Offline by Default

- Packaged app UI and logic is loaded and run locally.
- Enforced separation of client UI and data.
- APIs degrade gracefully when offline.
- Apps are launched from outside of the browser.



You are currently offline

Choose File

old-manifest.json

```
{
  "name": "Hello world",
  "version": "0.0.1",
  "app": {
    "launch": {
      "local_path": "main.html"
    }
  }
}
```



Choose File

new-manifest.json

Edit

```
1 {
2   "manifest_version": 2,
3   "name": "Hello world",
4   "version": "0.0.1",
5   "app": {
6     "background": {
7       "scripts": ["main.js"]
8     }
9   },
10  "permissions": [
11    "experimental"
12  ]
13 }
```

Choose URL

Choose File

diff-old

```
1 // Copyright (c) 2012 The Chromium Authors. All rights reserved.
2 // Use of this source code is governed by a BSD-style license
3 // found in the LICENSE file.
4
5 // File-level comment to appease parser. Eventually this will
6
7 [nodoc] namespace experimental.serial {
8
9     dictionary OpenInfo {
10         // The id of the opened connection.
11         long connectionId;
12     };
13
14     callback openCallback = void (OpenInfo openInfo);
15
16     callback closeCallback = void (boolean result);
17
18     dictionary ReadInfo {
19         // The number of bytes received, or a negative number
20         long bytesRead;
21
22         // The data received. warning: will probably become a
23         // appropriate binary-friendly type.
24         DOMString message;
25     };
26
27     callback readCallback = void (ReadInfo readInfo);
28
29     dictionary WriteInfo {
30         // The number of bytes written.
31         long bytesWritten;
32     };
33
34     callback writeCallback = void (WriteInfo writeInfo);
35
36     interface Functions {
```

Choose URL

Choose File

new-manifest.json

Edit

```
1 {
2   "manifest_version": 2,
3   "name": "Hello world",
4   "version": "0.0.1",
5   "app": {
6     "background": {
7       "scripts": ["main.js"]
8     }
9   },
10  "permissions": [
11    "experimental"
12  ]
13 }
```

Choose URL

Choose File

diff-old

Choose URL

Choose File

new-manifest.json

Edit

```
1 // Copyright (c) 2012 The Chromium Authors. All rights reserved.  
2 // Use of this source code is governed by a BSD-style license that can be  
3 // found in the LICENSE file.  
4  
5 // File-level comment to appear in the generated manifest.  
6  
7 [nodoc] namespace experiment {  
8  
9   dictionary OpenInfo {  
10     // The id of the opened connection.  
11     long connectionId;  
12   };  
13  
14   callback OpenCallback = void (OpenInfo info);  
15  
16   callback CloseCallback = void (boolean result);  
17  
18   dictionary ReadInfo {  
19     // The number of bytes received, or a negative number  
20     long bytesRead;  
21   };  
22 }
```

Select URL

Please enter the URL

Submit

Cancel

diff-old

```

1 // Copyright (c) 2012 The Chromium Authors. All rights reserved.
2 // Use of this source code is governed by a BSD-style license
3 // found in the LICENSE file.
4
5 // File-level comment to appease parser. Eventually this will
6
7 [nodoc] namespace experimental.serial {
8
9     dictionary OpenInfo {
10         // The id of the opened connection.
11         long connectionId;
12     };
13
14     callback OpenCallback = void (OpenInfo openInfo);
15
16     callback CloseCallback = void (boolean result);
17
18     dictionary ReadInfo {
19         // The number of bytes received, or a negative number
20         long bytesRead;
21
22         // The data received.
23         // appropriate binary-friendly type.
24         DOMString message
25     };
26
27     callback ReadCallback = void (ReadInfo readInfo);
28
29     dictionary WriteInfo {
30         // The number of bytes written.
31         long bytesWritten;
32
33         // The data to be written.
34         DOMString message;
35     };
36
37     callback WriteCallback = void (WriteInfo writeInfo);
38
39     dictionary CloseInfo {
40         // The error code.
41         long errorCode;
42     };
43
44     callback CloseCallback2 = void (CloseInfo closeInfo);
45
46     dictionary ConnectInfo {
47         // The id of the connection.
48         long connectionId;
49         // The address of the connection.
50         DOMString address;
51         // The port of the connection.
52         long port;
53     };
54
55     callback ConnectCallback = void (ConnectInfo connectInfo);
56
57     dictionary ListenInfo {
58         // The address of the connection.
59         DOMString address;
60         // The port of the connection.
61         long port;
62     };
63
64     callback ListenCallback = void (ListenInfo listenInfo);
65
66     dictionary AcceptInfo {
67         // The id of the connection.
68         long connectionId;
69         // The address of the connection.
70         DOMString address;
71         // The port of the connection.
72         long port;
73     };
74
75     callback AcceptCallback = void (AcceptInfo acceptInfo);
76
77     dictionary DisconnectInfo {
78         // The id of the connection.
79         long connectionId;
80         // The error code.
81         long errorCode;
82     };
83
84     callback DisconnectCallback = void (DisconnectInfo disconnectInfo);
85
86     dictionary AbortInfo {
87         // The id of the connection.
88         long connectionId;
89         // The error code.
90         long errorCode;
91     };
92
93     callback AbortCallback = void (AbortInfo abortInfo);
94
95     dictionary CancelInfo {
96         // The id of the connection.
97         long connectionId;
98     };
99
100    callback CancelCallback = void (CancelInfo cancelInfo);
101
102    dictionary CancelInfo2 {
103        // The id of the connection.
104        long connectionId;
105        // The error code.
106        long errorCode;
107    };
108
109    callback CancelCallback2 = void (CancelInfo2 cancelInfo2);
110
111    dictionary CancelInfo3 {
112        // The id of the connection.
113        long connectionId;
114        // The error code.
115        long errorCode;
116        // The address of the connection.
117        DOMString address;
118        // The port of the connection.
119        long port;
120    };
121
122    callback CancelCallback3 = void (CancelInfo3 cancelInfo3);
123
124    dictionary CancelInfo4 {
125        // The id of the connection.
126        long connectionId;
127        // The error code.
128        long errorCode;
129        // The address of the connection.
130        DOMString address;
131        // The port of the connection.
132        long port;
133    };
134
135    callback CancelCallback4 = void (CancelInfo4 cancelInfo4);
136
137    dictionary CancelInfo5 {
138        // The id of the connection.
139        long connectionId;
140        // The error code.
141        long errorCode;
142        // The address of the connection.
143        DOMString address;
144        // The port of the connection.
145        long port;
146    };
147
148    callback CancelCallback5 = void (CancelInfo5 cancelInfo5);
149
150    dictionary CancelInfo6 {
151        // The id of the connection.
152        long connectionId;
153        // The error code.
154        long errorCode;
155        // The address of the connection.
156        DOMString address;
157        // The port of the connection.
158        long port;
159    };
160
161    callback CancelCallback6 = void (CancelInfo6 cancelInfo6);
162
163    dictionary CancelInfo7 {
164        // The id of the connection.
165        long connectionId;
166        // The error code.
167        long errorCode;
168        // The address of the connection.
169        DOMString address;
170        // The port of the connection.
171        long port;
172    };
173
174    callback CancelCallback7 = void (CancelInfo7 cancelInfo7);
175
176    dictionary CancelInfo8 {
177        // The id of the connection.
178        long connectionId;
179        // The error code.
180        long errorCode;
181        // The address of the connection.
182        DOMString address;
183        // The port of the connection.
184        long port;
185    };
186
187    callback CancelCallback8 = void (CancelInfo8 cancelInfo8);
188
189    dictionary CancelInfo9 {
190        // The id of the connection.
191        long connectionId;
192        // The error code.
193        long errorCode;
194        // The address of the connection.
195        DOMString address;
196        // The port of the connection.
197        long port;
198    };
199
200    callback CancelCallback9 = void (CancelInfo9 cancelInfo9);
201
202    dictionary CancelInfo10 {
203        // The id of the connection.
204        long connectionId;
205        // The error code.
206        long errorCode;
207        // The address of the connection.
208        DOMString address;
209        // The port of the connection.
210        long port;
211    };
212
213    callback CancelCallback10 = void (CancelInfo10 cancelInfo10);
214
215    dictionary CancelInfo11 {
216        // The id of the connection.
217        long connectionId;
218        // The error code.
219        long errorCode;
220        // The address of the connection.
221        DOMString address;
222        // The port of the connection.
223        long port;
224    };
225
226    callback CancelCallback11 = void (CancelInfo11 cancelInfo11);
227
228    dictionary CancelInfo12 {
229        // The id of the connection.
230        long connectionId;
231        // The error code.
232        long errorCode;
233        // The address of the connection.
234        DOMString address;
235        // The port of the connection.
236        long port;
237    };
238
239    callback CancelCallback12 = void (CancelInfo12 cancelInfo12);
240
241    dictionary CancelInfo13 {
242        // The id of the connection.
243        long connectionId;
244        // The error code.
245        long errorCode;
246        // The address of the connection.
247        DOMString address;
248        // The port of the connection.
249        long port;
250    };
251
252    callback CancelCallback13 = void (CancelInfo13 cancelInfo13);
253
254    dictionary CancelInfo14 {
255        // The id of the connection.
256        long connectionId;
257        // The error code.
258        long errorCode;
259        // The address of the connection.
260        DOMString address;
261        // The port of the connection.
262        long port;
263    };
264
265    callback CancelCallback14 = void (CancelInfo14 cancelInfo14);
266
267    dictionary CancelInfo15 {
268        // The id of the connection.
269        long connectionId;
270        // The error code.
271        long errorCode;
272        // The address of the connection.
273        DOMString address;
274        // The port of the connection.
275        long port;
276    };
277
278    callback CancelCallback15 = void (CancelInfo15 cancelInfo15);
279
280    dictionary CancelInfo16 {
281        // The id of the connection.
282        long connectionId;
283        // The error code.
284        long errorCode;
285        // The address of the connection.
286        DOMString address;
287        // The port of the connection.
288        long port;
289    };
290
291    callback CancelCallback16 = void (CancelInfo16 cancelInfo16);
292
293    dictionary CancelInfo17 {
294        // The id of the connection.
295        long connectionId;
296        // The error code.
297        long errorCode;
298        // The address of the connection.
299        DOMString address;
300        // The port of the connection.
301        long port;
302    };
303
304    callback CancelCallback17 = void (CancelInfo17 cancelInfo17);
305
306    dictionary CancelInfo18 {
307        // The id of the connection.
308        long connectionId;
309        // The error code.
310        long errorCode;
311        // The address of the connection.
312        DOMString address;
313        // The port of the connection.
314        long port;
315    };
316
317    callback CancelCallback18 = void (CancelInfo18 cancelInfo18);
318
319    dictionary CancelInfo19 {
320        // The id of the connection.
321        long connectionId;
322        // The error code.
323        long errorCode;
324        // The address of the connection.
325        DOMString address;
326        // The port of the connection.
327        long port;
328    };
329
330    callback CancelCallback19 = void (CancelInfo19 cancelInfo19);
331
332    dictionary CancelInfo20 {
333        // The id of the connection.
334        long connectionId;
335        // The error code.
336        long errorCode;
337        // The address of the connection.
338        DOMString address;
339        // The port of the connection.
340        long port;
341    };
342
343    callback CancelCallback20 = void (CancelInfo20 cancelInfo20);
344
345    dictionary CancelInfo21 {
346        // The id of the connection.
347        long connectionId;
348        // The error code.
349        long errorCode;
350        // The address of the connection.
351        DOMString address;
352        // The port of the connection.
353        long port;
354    };
355
356    callback CancelCallback21 = void (CancelInfo21 cancelInfo21);
357
358    dictionary CancelInfo22 {
359        // The id of the connection.
360        long connectionId;
361        // The error code.
362        long errorCode;
363        // The address of the connection.
364        DOMString address;
365        // The port of the connection.
366        long port;
367    };
368
369    callback CancelCallback22 = void (CancelInfo22 cancelInfo22);
370
371    dictionary CancelInfo23 {
372        // The id of the connection.
373        long connectionId;
374        // The error code.
375        long errorCode;
376        // The address of the connection.
377        DOMString address;
378        // The port of the connection.
379        long port;
380    };
381
382    callback CancelCallback23 = void (CancelInfo23 cancelInfo23);
383
384    dictionary CancelInfo24 {
385        // The id of the connection.
386        long connectionId;
387        // The error code.
388        long errorCode;
389        // The address of the connection.
390        DOMString address;
391        // The port of the connection.
392        long port;
393    };
394
395    callback CancelCallback24 = void (CancelInfo24 cancelInfo24);
396
397    dictionary CancelInfo25 {
398        // The id of the connection.
399        long connectionId;
400        // The error code.
401        long errorCode;
402        // The address of the connection.
403        DOMString address;
404        // The port of the connection.
405        long port;
406    };
407
408    callback CancelCallback25 = void (CancelInfo25 cancelInfo25);
409
410    dictionary CancelInfo26 {
411        // The id of the connection.
412        long connectionId;
413        // The error code.
414        long errorCode;
415        // The address of the connection.
416        DOMString address;
417        // The port of the connection.
418        long port;
419    };
420
421    callback CancelCallback26 = void (CancelInfo26 cancelInfo26);
422
423    dictionary CancelInfo27 {
424        // The id of the connection.
425        long connectionId;
426        // The error code.
427        long errorCode;
428        // The address of the connection.
429        DOMString address;
430        // The port of the connection.
431        long port;
432    };
433
434    callback CancelCallback27 = void (CancelInfo27 cancelInfo27);
435
436    dictionary CancelInfo28 {
437        // The id of the connection.
438        long connectionId;
439        // The error code.
440        long errorCode;
441        // The address of the connection.
442        DOM
```

diff-new

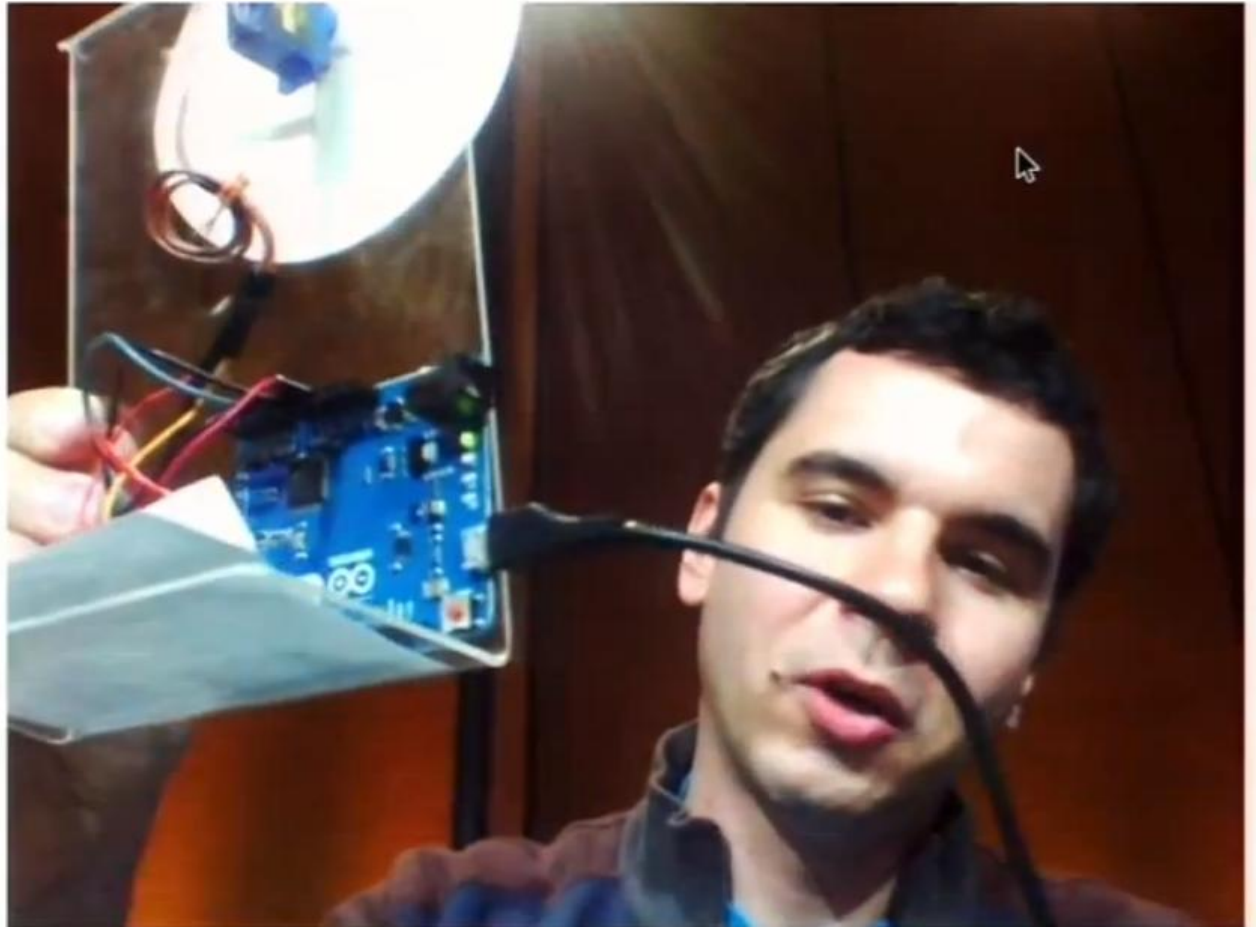
```

1 // Copyright (c) 2012 The Chromium Authors. All rights reserved.
2 // Use of this source code is governed by a BSD-style license
3 // found in the LICENSE file.
4
5 // File-level comment to appease parser. Eventually this will
6
7 namespace experimental.serial {
8
9     callback GetPortsCallback = void (DOMString[] ports);
10
11     dictionary OpenInfo {
12         // The id of the opened connection.
13         long connectionId;
14     };
15
16     callback OpenCallback = void (OpenInfo openInfo);
17
18     // Returns true if operation was successful.
19     callback CloseCallback = void (boolean result);
20
21     dictionary ReadInfo {
22         // The number of bytes received, or a negative number
23         long bytesRead;
24
25         // The data received.
26
27         ArrayBuffer data
28     };
29
30     callback ReadCallback = void (ReadInfo readInfo);
31
32     dictionary WriteInfo {
33         // The number of bytes written.
34
35         long bytesWritten;
36     };
37
38     callback WriteCallback = void (WriteInfo writeInfo);
39 }

```


New APIs

- System
- Shared Data
- Services



The Programming Model

- Packaged apps.

```
"manifest_version": 2,  
"name": "Hello World",  
"version": "1.0",  
"app": {  
  "background": {  
    "scripts": ["main.js"]  
  }  
},  
"permissions": [  
  "experimental"  
]
```







e: manifest.json Mode: JavaScript (JSON)

```
chrome.experimental.app.onLaunched.addListener(function() {  
  chrome.appWindow.create('window.html');  
});
```

New Open Save

```
1 <!DOCTYPE html>  
2 <html>  
3   <body>  
4     <h1>Hello World!</h1>  
5   </body>  
6 </html>
```

Google Drive

Name	Size	Type	Date ▾
 helloworld	--	Folder	Today, 1:52 PM
 mini-code-edit	--	Folder	Today, 10:42 AM
 diff-tool	--	Folder	Today, 10:41 AM
 servo	--	Folder	Today, 10:41 AM
 io2012-chrome-apps	--	Folder	Today, 10:41 AM
 diff-sample-files	--	Folder	Today, 10:41 AM

ome OS

Extensions

☒ Developer mode

Load unpacked extension...

Pack extension...

Update extensions now



Apps IO 2012 1.1

☒ Enabled

Search



Chrome



Chrome Web Store



Apps IO 2012



Diff Tool



Files



Games



Gmail



Google Calendar



Google Drive



Google Maps



Google Search



Google+



Hello World



MiniCodeEdit



Music



Scratchpad

012-chrome-apps
[presentation.html](#)
(inactive)

☒ Enabled

-tool
[\(Inactive\)](#)
[Reload \(Ctrl+R\)](#)

☒ Enabled

loworld
[\(Inactive\)](#)

☒ Enabled

me OS

Extensions

☒ Developer mode

Load unpacked extension...

Reload extension...

Update extensions now

ons

S

Hello World!

☒ Enabled



☒ Enabled



☒ Enabled



ID: elimdkmemkngoeiajbjhnjaennggpjn

Loaded from: /home/chronos/user/Downloads/helloworld

Inspect views: [window.html](#) [_generated background page.html](#)
[_generated background page.html \(Incognito\) \(Inactive\)](#)

☐ Allow in incognito [Reload \(Ctrl+R\)](#)

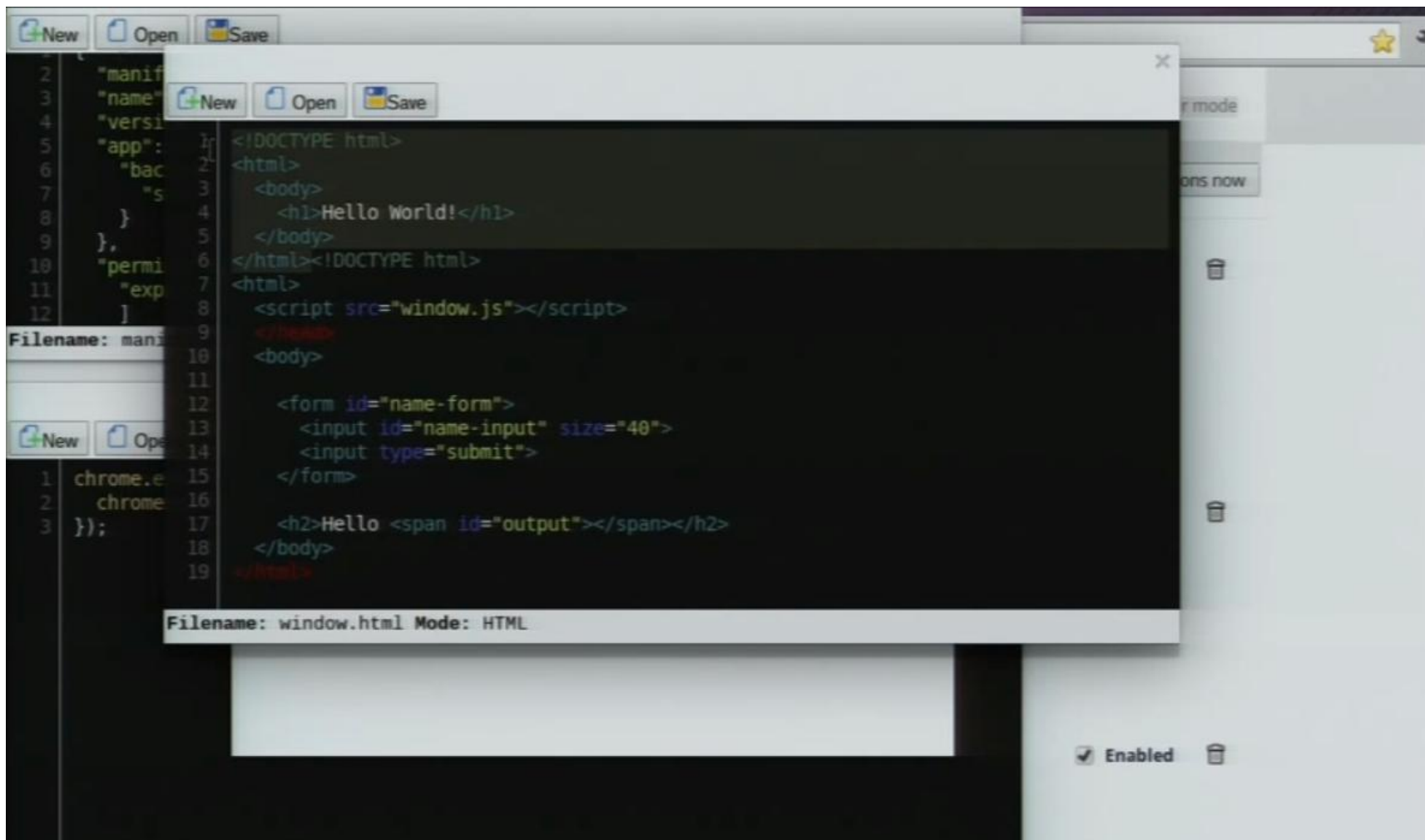
The Programming Model

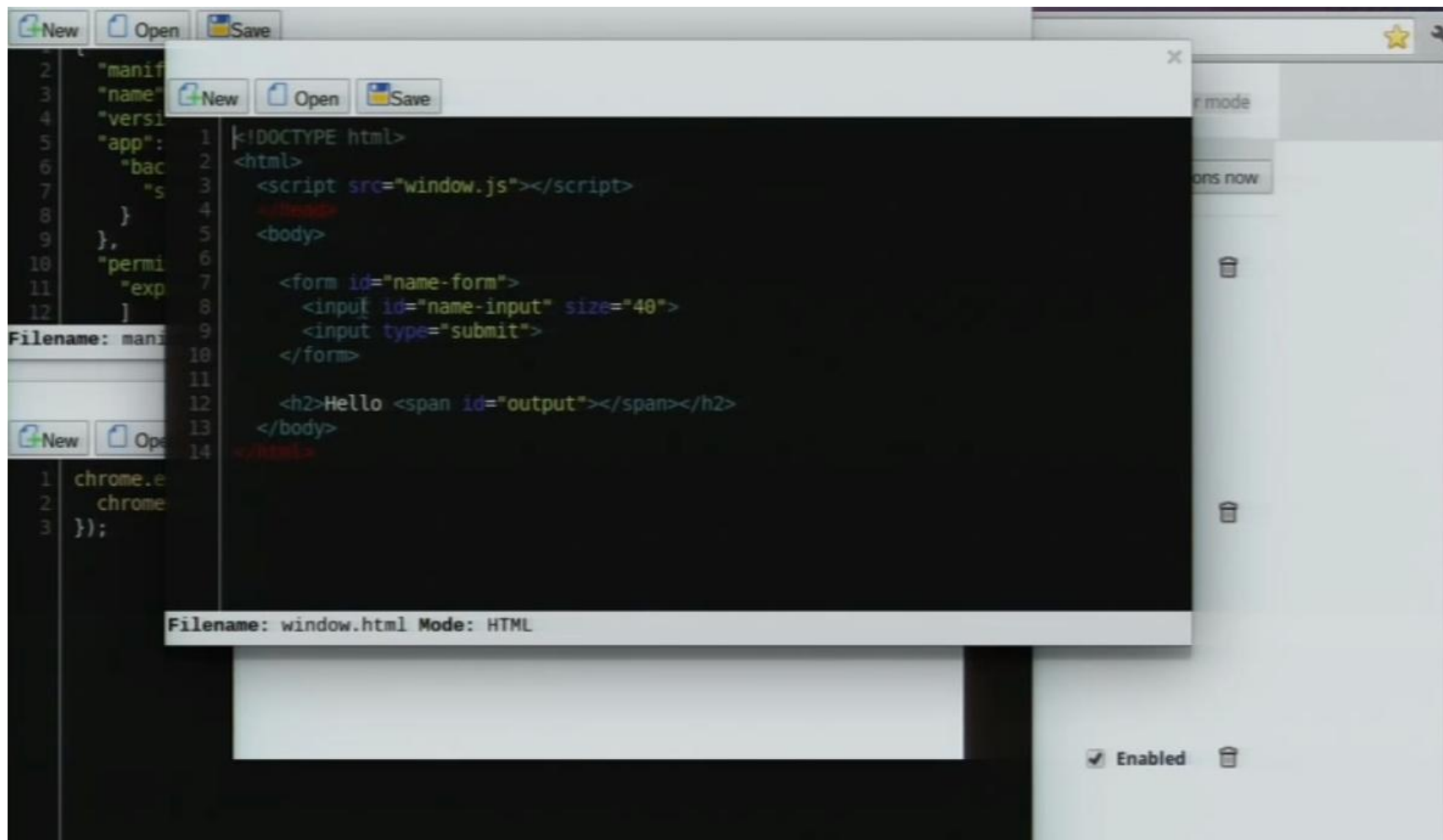
- Packaged apps.
- Background page as the hub.
- App lifetime controlled by runtime; event-driven.
- "Single-page", no navigation.
- Some web features deprecated.

Demo

The Security Model

- Process isolation.
- Sandboxing.
- Permissions model.





The Security Model

- Process isolation.
- Sandboxing.
- Permissions model.
- Content Security Policy (CSP).
- Storage isolation.
- Explicit shared data APIs.
- No extension APIs.

Demo

<browser>

```
<browser src="http://news.google.com/" width="750" height="300">
```

+You Search Images Maps Play YouTube News Gmail Documents Calendar More -

Google chrome.apps.io.2012@gmail.com

Press Esc to exit full screen mode.

News - U.S. edition - Modern -

Sports


 **As Anthony Davis prepares to head to New Orleans Hornets, guessing game begins ...**

New York Daily News - 1 hour ago

After Anthony Davis goes to the Hornets with the No. 1 pick in Thursday's NBA draft, it's all one big guessing game, except that everyone knows by now that the Rockets are trying to move up and send multiple picks to the Magic for Dwight Howard.

Highly Cited: [Anthony Davis Trademarks His Brow](#) CNBC.com - by Darren Rovell (in 139,069 Google+ circles)

Related [NBA Draft](#) » [ANTHONY DAVIS](#) »



Take it For a Spin

Demo

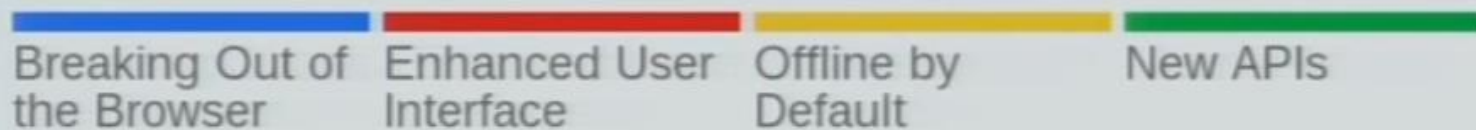


New Open Save

```
43 };
44 }
45
46 function openSelectedPort() {
47     var portPicker = document.getElementById('port-picker');
48     var selectedPort = portPicker.options[portPicker.selectedIndex].value;
49     chrome.experimental.serial.open(selectedPort, onOpen);
50 }
51
52 onload = function() {
53     var tv = document.getElementById('tv');
54     navigator.webkitGetUserMedia(
55         {video: true},
56         function(stream) {
57             tv.classList.add('working');
58             document.getElementById('camera-output').src =
59                 webkitURL.createObjectURL(stream);
60         },
61         function() {
62             tv.classList.add('broken');
63         });
64
65     document.getElementById('position-input').onchange = function() {
66         setPosition(parseInt(this.value, 10));
67     };
68
69     chrome.experimental.serial.getPorts(function(ports) {
70         buildPortPicker(ports)
71         openSelectedPort();
72     });
73 };
74
```

Filename: servo.js Mode: JavaScript

Apps Evolved



- Available for testing on Canary
- System Applications working group



Check Out More Demos

- Media player - Sencha
- Photobooth - Kendo UI
- Text Editor - AngularJS (Google)
- "Johnny" - Google
- github.com/GoogleChrome

Thank You!

Try out the developer preview and
send us feedback.

developer.chrome.com/apps

chromium-apps@chromium.org

[#chromium-apps](#) (freenode)

