sebpiq/WebPd

Run your Pure Data patches on the web.

1. JavaScript 96.8%
2. Pure Data 2.3%

JavaScript Pure Data

sebpiq authored Apr 30, 2015

WebPd

WebPd is a **100% JavaScript Pure Data runtime** using **Web Audio API** to play audio in the browser. It aims at allowing a subset of Pure Data programming language to run in the browser without plugins and with best possible performance.

WebPd should be supported by all browsers supporting Web Audio API.

The following documentation is only about **WebPd**. To learn more about Pure Data, and start creating Pd patches which you can later use with **WebPd**, please refer to the official Pure Data documentation.

Quick start

The following instructions provide a quick start for people with JavaScript knowledge. If you have no experience with JS, you might want to read the step-by-step guide instead.

1. Grab the latest version of WebPd from here.

2. Add WebPd to your Web page, and load a patch by calling Pd.loadPatch.

```
<!doctype HTML>
<html>
  <head>
    <script src="js/jquery.js"></script>
    <script src="js/webpd-latest.js"></script>
    <script>
        var patch
        $.get('patches/myPatch.pd', function(patchStr) {
          patch = Pd.loadPatch(patchStr)
          Pd.start()
        })
    </script>
  </head>

  <body></body>
</html>
```

If you are testing locally, the ajax request might be blocked by your browser because of same-origin policy. For a workaround, see the troubleshooting section.

Step-by-step guide

The following instructions provide a detailed guide to start working with WebPd. If you have experience with web development, you might want to read the quick start instead.

1. First create a folder `myProject` in which you will create your first WebPd project. In this folder, create the following structure :

   ```
   myProject/
       patches/
       js/
   ```

2. Download the latest version of WebPd, save it as `webpd-latest.js` onto your computer, in the folder `myProject/js`.

3. Download the latest version of jquery, save it as `jquery.js` onto your computer, in the folder `myProject/js`.

4. In the folder `myProject`, create a file called `index.html` and copy/paste the following code in it

   ```
   <!doctype HTML>
   <html>
     <head>
       <script src="js/jquery.js"></script>
       <script src="js/webpd-latest.js"></script>
       <script>
           var patch
           $.get('patches/myPatch.pd', function(patchStr) {
             patch = Pd.loadPatch(patchStr)
             Pd.start()
           })
       </script>
     </head>

     <body></body>
   </html>
   ```

   Save that file. Be sure to use a text editor that is programmer-friendly. Microsoft Word and other text processors will add a lot of extra informations, making your code impossible to understand by a web browser. I recommend using something like *notepad*, *gedit*, ...

5. Create a patch using Pure Data. Make sure that you use only features and objects supported by WebPd. Save that patch as `myPatch.pd` in the folder `myProject/patches`.

6. Because we are working locally on our computer, we now need to run a web server to be able to

open the web page `index.html` properly. For this, we will use the web server that comes with Python.

Chances are, you already have Python installed on your computer. To check this, open a terminal (or command prompt), and run `python --version`, this should print the version of Python installed. If instead you get something like `command not found`, then you need to install Python.

In the terminal use the command `cd` to navigate to the folder `myProject`. When you've arrived there, run the command `python -m SimpleHTTPServer` if you have **Python 2** or `python -m http.server` if you have **Python 3**.

7. You can finally open your web page and listen to your patch, by opening a web browser and navigating to http://localhost:8000/index.html.

Examples

There is a few examples in the examples/ folder. You can also open them online :

- abstractions
- delays
- gui-controls
- phasor
- subpatches
- tabread-line
- tabread-phasor

Troubleshooting

I can't run any WebPd demo on my computer

For security reasons, browsers control access to your file system from web pages. Because of this, getting Pd files with Ajax might fail on your local machine. A workaround is to start a local server and access all the example web pages through it.

Python comes bundled with such a web server. Open a terminal, navigate to the folder containing the web page you want to open, then run `python -m SimpleHTTPServer` if you are using **Python 2** or `python -m http.server` if you are using **Python 3**. Then open your web browser to http://localhost:8000 and things should start working.

A patch that used to work fine with WebPd has stopped working after I modified it

WebPd has a few limitations. For example, some of the Pd objects are not available. Open your browser's developer console (`ctrl+shift+i` on firefox and chrome for linux or windows), and you should get a clear error message telling you what is wrong. If the error is unclear, or if there is no error, it might be a bug with WebPd. In that case, it would be great if you could submit a bug report.

A patch that works fine on the desktop doesn't seem to work on mobile

WebPd uses Web Audio API, and as it happens, running Web Audio API on mobile is not always easy. First, make sure that you use a browser **that does support Web Audio API**. For example the default Android browser does not, and so on Android you have to use Chrome or Firefox.

On iPhone and iPad, things are even trickier. For security reasons, audio is blocked by iOS, unless you start it in direct answer to a user action (click, touch, ...). So to get sound with WebPd, you will need to do exactly that and for example call `Pd.start` in a button's `onclick` handler : `onclick="Pd.start()"`.

My patch doesn't work on some browser which should support Web Audio API

Web Audio API is a work in progress, so there is discrepencies between different browser implementations. WebPd builds on the latest version of Web Audio API. To make sure that your web page also works with browsers implementing older versions of Web Audio API, include that javascript in your web page, before you include WebPd.

List of implemented objects and other limitations

Not all of Pure Data's objects are available in WebPd. Please check-out the list of available objects.

Abstractions are implemented, but at the moment they require a bit of extra JavaScript in order to work. You can check-out the abstractions example, to see how this works.

While WebPd uses only Web Audio API and should therefore be quite efficient, you might find that some patches perform poorly on mobile devices, or if there is too many objects running at the same time. This is because Web Audio API is not optimized to work in the same way as Pure Data. For example, modulating parameters with an audio signal (frequencies, delay times, ...), though it is very frequent in Pd, can cause audio glitches in the browser if you use it too much or in a big patch.

Submitting a bug report

If you find a bug, you can submit a bug report on the project's issue tracker.

Please try to include as much information as possible. Also try to include code, and **the patch** that you cannot get to work.

API

Pd

Pd.loadPatch(patchStr)

Loads a Pd patch, and returns a `Patch` object. `patchStr` is the whole contents of a Pd file (and not only a file name).

Pd.receive(name, callback)

Receives messages from named senders within a patch (e.g. `[send someName]`). Example :

```
Pd.receive('someName', function(args) {
    console.log('received a message from "someName" : ', args)
})
```

Pd.send(name, args)

Sends messages from JavaScript to a named receiver within a patch (e.g. `[receive someName]`). Example :

```
Pd.send('someName', ['hello!'])
```

Instructions for building webpd.js

To build WebPd yourself, you will need node.js and gulp.js.

When these are installed, run `npm install` in WebPd root folder.

Finally, run `npm run build` to build a new version of WebPd in `dist/webpd-latest.js`.

Instructions for running the tests

WebPd comes with two test suites.

Automated tests

The tests in `test/src` run on **node.js** using mocha. To run them, simply execute the command `npm test`.

Browser tests

The tests in `test/browser` run in a web browser.

To build them, first scaffold by running `node node_modules/waatest/bin/scaffold.js ./waatest`. This will create a folder `waatest` containing a test web page.

Build the tests by running `npm run test.browser.build`.

Then start a local web server (see troubleshooting), and open `waatest/index.html` in your web browser.

Contributing

Any kind of contribution would be very welcome. Check out the issue tracker or contact me directly.