pyodide-pack

Release 0.2.1.dev29+ga9043f4

Pyodide maintainers

Apr 02, 2024

CONTENTS:

Python package bundler and minifier for the web

Pyodide-pack aims to reduce the size and load time of Python applications running in the browser using different strategies:

- Minification of the source code via AST rewrite
- Transformation of the source code into a different format such as Python bytecode (.pyc files)
- Dead code elimination, by removing unused Python modules (detected at runtime)

Each of these approaches have different tradeoffs, and can be used separately or in combination.

(ast-rewrite=)

CHAPTER

ONE

ABSTRACT SYNTAX TREE (AST) REWRITE

In this section we apply Abstract Syntax Tree (AST) rewrites to the package source code. These include,

- removal of comments
- removal of function and class docstrings
- removal of module docstrings

To apply rewrites on one or multiple wheels, run,

```
pyodide minify <path_to_dir_with_py_files>
```

CHAPTER

TWO

MODULE REMOVAL BY RUNTIME DETECTION

1. Create file with the code of your Python application running in the web. As example we will take, `examples/pandas/app.py`

   app.py

   ```
   import pandas as pd  # noqa

   pd.DataFrame(range(10))
   ```

   This application can run with Pyodide, and will need to download around 10.5 MB of packages, including numpy and pandas in addition to ~7MB for CPython with stdlib.

2. Create the package bundle,

   ```
   pyodide pack examples/pandas/app.py
   ```

   which would produce the following output

```
Running pyodide-pack on examples/pandas/app.py

Note: unless otherwise specified all sizes are given for gzip compressed files to
be representative of CDN compression.

Loaded requirements from: examples/pandas/requirements.txt
Running the input code in Node.js to detect used modules..

[...]
Done input code execution in 3.8 s

Using stdlib (547 files) with a total size of 2.25 MB.
Detected 5 dependencies with a total size of 8.92 MB  (uncompressed: 35.46 MB)
In total 487 files and 0 dynamic libraries were accessed.
Total initial size (stdlib + dependencies): 11.17 MB


                                    Packing..

 No  Package                        All files  .so libs    Size (MB)  Reduction

|  0 | stdlib                        | 547 → 151 |          | 2.25 → 0.75 |    ␣
↪66.7 % |
|  1 | numpy-1.25.2-cp311-cp311-emsc... | 430 → 111 |   19 → 0 | 3.06 → 2.36 |    ␣
```

(continues on next page)

(continued from previous page)

```
↪23.0 % │
│  2 │ pandas-1.5.3-cp311-cp311-emsc... │ 462 → 292 │   42 → 0 │ 5.17 → 4.64 │  ␣
↪10.3 % │
│  3 │ python_dateutil-2.8.2-py2.py3... │   25 → 15 │    0 → 0 │ 0.24 → 0.22 │  ␣
↪ 9.4 % │
│  4 │ pytz-2023.3-py2.py3-none-any.... │   614 → 5 │    0 → 0 │ 0.43 → 0.02 │  ␣
↪96.1 % │
│  5 │ six-1.16.0-py2.py3-none-any.w... │     6 → 1 │    0 → 0 │ 0.01 → 0.01 │  ␣
↪18.5 % │
└──────────────────────────────────────────────────────────────────────────────────
Wrote pyodide-package-bundle.zip with 7.37 MB (17.4% reduction)

Spawning webserver at http://127.0.0.1:52009 (see logs in /tmp/tmpx0ktv9fw/http-
↪server.log)
Running the input code in Node.js to validate bundle..

        Validating and benchmarking the output bundle..

 Step                  Load time (s)  Fraction of load time

│ loadPyodide           │         1.34 │                36.1 % │
│ fetch_unpack_archive  │         0.27 │                 7.4 % │
│ load_dynamic_libs     │         0.00 │                 0.1 % │
│ import_run_app        │         2.10 │                56.5 % │
│ TOTAL                 │         3.72 │                 100 % │
└──────────────────────────────────────────────────────────────

Total output size (stdlib + packages): 8.12 MB (27.3% reduction)

Bundle validation successful.
```

3. Load your Python web application with,

```
let pyodide = await loadPyodide({fullStdLib: false});

await pyodide.runPythonAsync(`
  from pyodide.http import pyfetch

  response = await pyfetch("<your-server>/pyodide-package-bundle.zip")
  await response.unpack_archive(extract_dir='/')
`)

await pyodide.pyimport('pyodide_pack_loader').setup();
```

2.1 Implementation

This bundler runs your applications in a Node.js and intercepts,

- `FS.open` calls in read mode, which includes accessed files in the Emscripten's MEMFS file system opened from Python, C or Javascript.
- calls to load a dynamic library

Package wheels are then repacked into a single bundle with the accessed files and dynamic libraries.

CHAPTER

THREE

PYODIDE-PACK CLI

This page documents the pyodide-pack Command Line Interface (CLI) interface,

3.1 pyodide pack

Create a minimal bundle for a Pyodide application with the required dependencies

Experimental: this is based on runtime dependency analysis and may not work for all applications.

```
pyodide pack [OPTIONS] EXAMPLE_PATH
```

Options

-v

Increase verbosity (currently ignored)

Default
`False`

--config <config_path>

Path to the pyproject.toml with the tool.pyodide_pack section

--include <include_paths>

One or multiple glob patterns separated by “,” of extra files to include

--write-debug-map, --no-write-debug-map

Write a debug map (to ‘./debug-map.json’) with allthe detected imports for the generated bundle

Default
`False`

--install-completion

Install completion for the current shell.

--show-completion

Show completion for the current shell, to copy it or customize the installation.

Arguments

EXAMPLE_PATH

Required argument

3.2 pyodide minify

Minify a folder of Python files.

Note: this API will change before the next release

```
pyodide minify [OPTIONS] INPUT_DIR
```

Options

--strip-docstrings, --no-strip-docstrings

Strip docstrings

Default

`False`

--strip-module-docstrings, --no-strip-module-docstrings

Strip module lebel docstrings

Default

`False`

--install-completion

Install completion for the current shell.

--show-completion

Show completion for the current shell, to copy it or customize the installation.

Arguments

INPUT_DIR

Required argument

CHAPTER

FOUR

CONFIGURATION

`pyodide pack` can be configured via a `pyproject.toml` file in the root of your project, or in a any of the parent directories.

Below is an example of configuration with default values. In most cases, the defaults should be fine, and you can only include fields you want to change.

```
[tool.pyodide_pack]
requires = []
include_paths =  []

[tool.pyodide_pack.py]
strip_module_docstrings = false
strip_docstrings = false
py_compile = false

[tool.pyodide_pack.so]
drop_unused_so = true
```

4.1 Configuration options

4.1.1 `requires`

List of dependencies to load. This list is passed to micropip, so it can be any valid micropip specifier.

4.1.2 `include_paths`

List of paths to include in the bundle. This is useful for including files that were otherwise excluded by `pyodide pack`

4.1.3 py.strip_module_docstrings

Whether to strip module docstrings. Default: `false`

4.1.4 py.strip_docstrings

Whether to strip docstrings. Default: `false`

4.1.5 py.py_compile

Whether to compile python files. Default: `false`

4.1.6 so.drop_unused_so

Whether to drop unused `.so` files. Default: `true`

INDEX

Symbols

E

I

P