qooxdoo Generator Cheat Sheet - v.1.0.0-1

This cheat sheet summarizes the options available for the generator and its Json config files. For full reference see: http://qooxdoo.org/documentation/1.0/tool/generator_config

Generator Options

```
shell> generator.py -h
Usage: generator.py [options] job,...

Arguments:
  job,...           a list of jobs (like 'source' or 'copy-files',
                    without the quotes) to run
  x                 use 'x' (or some undefined job name) to get a
                    list of all available jobs from the
                    configuration file

Options:
  -h, --help        show this help message and exit
  -c CFGFILE, --config=CFGFILE
                    path to configuration file containing job
                    definitions (default: config.json)
  -q, --quiet       quiet output mode (extra quiet)
  -v, --verbose     verbose output mode (extra verbose)
  -l FILENAME, --logfile=FILENAME
                    log file
  -s, --stacktrace enable stack traces on fatal exceptions
  -m KEY:VAL, --macro=KEY:VAL
                    define/overwrite a global 'let' macro KEY with
                    value VAL
```

Default Jobs

api	create api doc for the current library
build	create build version of current application
clean	remove local cache and generated .js files (source/build)
distclean	remove the cache and all generated artefacts of this library (source, build, ...)
fix	normalize whitespace in .js files of the current library (tabs, eol, ...)
info	print environment information (OS/Python/qooxdoo version,...)
inspector	create an inspector instance in the current library
lint	check the source code of the .js files of the current library
migration	migrate the .js files of the current library to the current qooxdoo version
pretty	pretty-formatting of the source code of the current library
source	create source version of current application
source-all	create source version of current application, with all classes
test	create a test runner app for unit tests of the current library
test-source	create a test runner app for unit tests (source version) of the current library
translation	create .po files for current library

Config File Layout

This is the general layout of a config file:

```
{
  “name” : “<descriptive text>”,
  “include” : [
    {  “path”   : “<os path>”,
       “as”     : “<library prefix>”,
       “import” : [<job>,...{“name”:<job>, “as”:<alias>},..],
       “block”  : [<job>,...]
    }
  ],
  “let”  : { <macro> : <value>, ...},
  “export” : [<job>,...],
  “jobs” : {
    <job> : {
      “add-script” : [...],
      “api” : {...},
      “asset-let” : {...},
      “cache” : {...},
      “clean-files” : {...},
      “combine-images” : {...},
      “compile-dist” : {...},
      “compile-source” : {...},
      “copy-files” : {...},
      “copy-resources” : {...},
      “dependencies” : {...},
      “desc” : “<descriptive text>”,
      “exclude” : [...],
      “export”  : [...],
      “extend”  : [...],
      “fix-files” : {...},
      “include” : [...],
      “let” : {...},
      “library” : [...],
      “lint-check” : {...},
      “log” : {...},
      “migrate-files” : {...},
      “packages” : {...},
      “pretty-print” : {...},
      “require” : {...},
      “run” : [...],
      “settings” : {...},
      “shell” : {...},
      “slice-images” : {...},
      “translate” : {...},
      “variants” : {...}
    }
  }
}
```

Config Keys

```
“add-script” :
{
  "uri" : "<script-uri>"
}

“api” :
{
  "path" : "<path>"
}

"asset-let" :
{
  "<macro_name>" : [ "foo", "bar", "baz" ]
}

"cache" :
{
  "compile"   : "<path>",
  "downloads" : "<path>"
}

"clean-files" :
{
  "<doc_string>" :
  [
    "<path>",
    "<path>"
  ]
}

"combine-images" :
{
  "images" :
  {
    "<output_image>" :
    {
      “prefix”: [ “<string>”, “<altstring>” ],
      "layout": ("horizontal"|"vertical"),
      "input" :
      [
        {
          “prefix” : [ “<string>”, “<altstring>” ],
          “files”  : [ "<path>", "<path>" ]
        }
      ]
    }
  }
}

"compile-dist" :
{
  "paths" :
  {
    "file"     : "<path>",
    "gzip"     : (true|false),
    “loader-template” : “<path>”
  },
  "uris" :
  {
    "script"   : "script",
    "resource" : "resource"
  },
  "code" :
  {
    "format"   : (true|false),
    "locales"  : ["de", "en"],
    "optimize" : ["variables", "basecalls", "privates",
                  "strings"],
    “decode-uris-plug” : “<path>”
  }
}

"compile-source" :
{
  "file" : "<filepath>",
  "root" : "<path_to_html_dir>",
  "locales" : ["de", "en"],
  "gzip" : (true|false),
  “decode-uris-plug” : “<path>”,
  “loader-template”  : “<path>”
}

"copy-files" :
{
  "files"   : [ "<path>", "<path>" ],
  "source"  : "<path>",
  "target"  : "<path>"
}

"copy-resources" :
{
  "target" : "<path>"
}

"dependencies" :
{
  "follow-static-initializers"  : (true|false),
  “sort-topological”            : (true|false)
}
```

```
"desc" : "Some text."

"exclude" : ["qx.util.*"]

"export" : ["job1", "job2", "job3"]

"extend" : [ "job1", "job2", "job3" ]

"fix-files" : {}

"include" : ["qx.util.*"]  // job-level

"include" :   // top-level
[
  {
    "path"   : "<path>",
    "as"     : "<name>",
    "import" : ["extjob1", "extjob2", "extjob3"],
    "block"  : ["extjob4", "extjob5"]
  }
]

"jobs" :
{
  "<job_name>" : { <job_definition> }
}

"let" :
{
  "<macro_name>"  : "<string>",
  "<macro_name1>" : [ ... ],
  "<macro_name2>" : { ... }
}

"library" :
[
  {
    "manifest"   : "<path>",
    "uri"        : "<from_html_to_manifest_dir>",
    "namespace"  : "<string>"
  }
]

"lint-check" :
{
  "allowed-globals" : [ "qx", "${APPLICATION}" ]
}

"log" :
{
  "classes-unused" : [ "custom.*", "qx.util.*" ],
  "privates"       : ("on"|"off"),
  "dependencies"   :
  {
    "type"         : ("using"|"used-by"),
    "using"        :
    {
      "phase"      : ("runtime"|"loadtime")
    },
    "format"       : ("txt"|"dot"|"json"|"flare"),
    "dot"          :
    {
      "root"           : "custom.Application",
      "file"           : "<filename>",
      "radius"         : 5,
      "span-tree-only" : (true|false),
      "compiled-class-size" : (true|false),
      "optimize"   : [<optimize-keys>]
    },
    "json"         :
    {
      "file"       : "<filename>",
      "pretty"     : (true|false)
    },
    "flare"        :
    {
      "file"       : "<filename>",
      "pretty"     : (true|false)
    }
  }
}

"migrate-files" :
{
   "from-version" : "0.7",
   "migrate-html" : false
}

"name" : "Some text."

"packages" :
{
  "parts"  :
  {
    "<part_name>" :
    {
      "include"   : [ "app.class1", “app.class2.*" ],
      "expected-load-order"      : 1
      "no-merge-private-package" : (true|false)
    }
  },
  "sizes"  :
  {
    "min-package"           : 1,
    "min-package-unshared"  : 1
  },
  "init"             : "<part_name>",
  "loader-with-boot" : (true|false),
  "i18n-with-boot"   : (true|false)
}

"pretty-print" :
{
  "general" :
  {
    "indent-string"        : "  "
  },
  "comments" :
  {
    "trailing" :
    {
      "keep-column"        : false,
      "comment-cols"       : [50, 70, 90],
      "padding"            : "  "
    }
  },
  "blocks" :
  {
    "align-with-curlies"   : false,
    "open-curly" :
    {
      "newline-before"     : "m",
      "indent-before"      : false
    }
  }
}

"require" :
{
  "<class_name>" : [ "qx.util", "qx.fx" ]
}

"run" : [ "<job1>", "<job2>", "<job3>" ]

"settings" :
{
  "qx.application" : "myapp"
}

"shell" :
{
  "command" : "echo foo bar baz"
}

"slice-images" :
{
  "images" :
  {
    "<input_image>" :
    {
        "prefix"       : "<string>",
        "border-width" : 5
    }
  }
}

"translate" :
{
  "namespaces" : [ "qx.util" ],
  "locales"    : [ "en", "de" ],
  "poentry-with-occurrences : (true|false)
}

"variants" :
{
  "qx.debug" : [ "on" , "off" ]
}
```