The Gatsby REPL ("read-eval-print loop") is available via the Gatsby CLI by running the command `gatsby repl`. This gives you access to an interactive REPL shell within the context of your Gatsby environment. It can be used to retrieve general data and programmatically interact with it. If you have an idea for a new command, feel free to submit a PR for it!

This doc will give a brief description of each REPL command, expected output, and an example of what you can do with the command to digest the data. Examples are using the Gatsby Starter Blog as a demo environment, since it is currently the highest rated starter, and provides standard output for most of these commands.

To get started, in your terminal, after running the initial site setup steps here, run the command `gatsby repl` to enter the interactive shell. If you are writing out a function, you can write that over multiple lines, as long as you do not use a semicolon or close a parenthesis or brace prematurely. This is helpful for running GraphQL queries and callback functions.

REPL Commands

`babelrc`

Returns an object with the global `babelrc` settings.

Usage: `babelrc`

Example:

```
// Command:
gatsby > babelrc
// Returns:
{ stages:
   { develop: { plugins: [], presets: [], options: [Object] },
     'develop-html': { plugins: [], presets: [], options: [Object] },
     'build-html': { plugins: [], presets: [], options: [Object] },
     'build-javascript': { plugins: [], presets: [], options: [Object] } } }
```

`components`

Returns a Map object with all of the components in your Gatsby environment (see Mozilla Map docs for more information on Map objects and how to use them). Properties that get returned: `name`, `componentPath`, `query`, `pages`, and `isInBootstrap:`.

Usage: `components`

Example:

```
// Command:
gatsby > for( var [key, value] of components ) { console.log(key + ' = ' +
value.pages); }
// Returns: a list of components and the pages they are used on...
.../my-blog-starter/src/templates/blog-post.js = /hi-folks/,/my-second-post/,/hello-
world/
.../my-blog-starter/src/pages/404.js = /404/,/404.html
.../my-blog-starter/src/pages/index.js = /
.../my-blog-starter/.cache/dev-404-page.js = /dev-404-page/
```

`getNode()`

Get a single node by its ID, typically a string.

Usage: `getNode(<id>)`

Example:

```
// Command:
gatsby > getNode('SitePage /404.html')
// Returns:
{ internalComponentName: 'Component404Html',
  path: '/404.html',
  ...
  id: 'SitePage /404.html',
  ...
  internal:
   { type: 'SitePage',
     contentDigest: '3688d3ee613222ebfe3f44bbdaeb8ca0',
     description: 'f795702c-a3b8-5a88-88ee-5d06019d44fa',
     owner: 'internal-data-bridge' } }
```

`getNodes()`

Returns an array of objects (the nodes).

Usage: `getNodes()`

Examples:

```
// Command:
gatsby > getNodes().map(node=>node.internal.type)
// Returns:
[ 'SitePage',
  'SitePlugin',
  'SitePlugin',
  'SitePlugin',
  'SitePlugin',...] // An array of each node's internal type.

// Command:
gatsby > getNodes().filter(node=> {if('MarkdownRemark' == node.internal.type) return
node}).map(node=> node.frontmatter.title)
// Returns:
[ 'Hello World', 'My Second Post!', 'New Beginnings' ] // First returns an array of
just nodes with a MarkdownRemark type, then creates an array of titles (blog posts).
```

`nodes`

`nodes` is like `getNodes()`, but returns an **indexed** array of objects (the nodes). You can also pass in the index of the node you want, and that will return an array with a single node object.

Usage: `nodes` returns the array, or `nodes[<id>]` returns an array with a node object.

Examples:

```
// Command:
gatsby > nodes.length
// Returns:
48 // The length of the nodes array.

// Command
gatsby > nodes[47]
// Returns:
[ 47,
  { internalComponentName: 'Component404Html',
    path: '/404.html',
    ...
    componentPath: '.../my-blog-starter/src/pages/404.js',
    id: 'SitePage /404.html',
    ...
    internal:
     { type: 'SitePage',
       contentDigest: '1047b0e301924ae75175482834ef7b1a',
       description: 'f795702c-a3b8-5a88-88ee-5d06019d44fa',
       owner: 'internal-data-bridge' } } ]
```

`pages`

Returns an indexed array of arrays. Each array contains a key which is the slug, and a value which is the page node object.

Usages: `pages` or `pages[<index>]`

Example:

```
// Command:
gatsby > pages[0]
// Returns:
[ '/hi-folks/',
  { internalComponentName: 'ComponentHiFolks',
    path: '/hi-folks/',
    ...
    componentPath:
     '.../my-blog-starter/src/templates/blog-post.js' } ]
```

`schema`

Returns the GraphQL schema of your Gatsby environment as an object.

Usages: `schema` or `schema[<property>]`

Example:

```
// Command:
gatsby > schema._implementations
// Result:
[Object: null prototype] {
  Node:
   [ File,
     MarkdownRemark,
     ImageSharp,
     SitePage,
     SitePlugin,
     Site,
     Directory ] } // Returns the property value of implementations.
```

`siteConfig`

Returns the settings you would find in the `gatsby-config.js` file of your site as an object.

Usages: `siteConfig` or `siteConfig[<property>]`

Example:

```
// Command:
gatsby > siteConfig.siteMetadata
// Returns:
{ title: 'Gatsby Starter Blog',
  author: 'Kyle Mathews',
  description: 'A starter blog demonstrating what Gatsby can do.',
  siteUrl: 'https://gatsby-starter-blog-demo.netlify.app/',
  social: { twitter: 'kylemathews' } } // returns just the siteMetadata value of the
config.
```

`staticQueries`

Returns a Map object with all of the static queries in your Gatsby environment. Properties that get returned: `name`, `componentPath`, `id`, `query`, and `hash`.

Usage: `staticQueries`

Example:

```
// Command:
gatsby > for( var [key, value] of staticQueries ) { console.log(key + ' = ' +
value.componentPath); }
// Returns:
sq--src-components-seo-js = .../my-blog-starter/src/components/seo.js
sq--src-components-bio-js = .../my-blog-starter/src/components/bio.js
```