Build Caching

Plugins can cache data as JSON objects and retrieve them on consecutive builds.

Caching is already used by Gatsby and plugins for example:

- any nodes created by source/transformer plugins are cached
- `gatsby-plugin-sharp` caches built thumbnails

Build outputs are stored in the `.cache` and `public` directories relative to your project root.

The cache API

The cache API is passed to Gatsby's Node APIs which is typically implemented by plugins.

```
exports.onPostBootstrap = async function ({ cache, store, graphql }) {}
```

The two functions you would want to use are:

`set`

Cache value

```
cache.set(key: string, value: any) => Promise<any>
```

`get`

Retrieve cached value

```
cache.get(key: string) => Promise<any>
```

The Node API helpers documentation offers more detailed information on the API.

Plugin Example

In your plugin's `gatsby-node.js` file, you can access the `cache` argument like so:

```
exports.onPostBuild = async function ({ cache, graphql }, { query }) {
  const cacheKey = "some-key-name"
```

```
  const twentyFourHoursInMilliseconds = 24 * 60 * 60 * 1000 // 86400000
  let obj = await cache.get(cacheKey)

  if (!obj) {
    obj = { created: Date.now() }
    const data = await graphql(query)
    obj.data = data
  } else if (Date.now() > obj.lastChecked + twentyFourHoursInMilliseconds) {
    /* Reload after a day */
    const data = await graphql(query)
    obj.data = data
  }

  obj.lastChecked = Date.now()

  await cache.set(cacheKey, obj)

  /* Do something with data ... */
}
```

Clearing cache

Since cache files are stored within the `.cache` directory, deleting it will clear all cache. You can also use `gatsby clean` to delete the `.cache` and `public` folders. The cache is also invalidated by Gatsby in a few cases, specifically:

- If `package.json` changes, for example a dependency is updated or added
- If `gatsby-config.js` changes, for example a plugin is added or modified
- If `gatsby-node.js` changes, for example if you invoke a new Node API, or change a `createPage` call

Conclusion

With the cache API you're able to persist data between builds, which is really helpful while developing a site with Gatsby (as you re-run `gatsby develop` really often). Performance-heavy operations (like image transformations) or downloading data can slow down the bootstrap of Gatsby significantly and adding this optimization to your plugin can be a huge improvement to your end users. You can also have a look at the following examples who implemented the cache API: gatsby-source-contentful, gatsby-source-shopify, gatsby-source-wordpress, gatsby-transformer-remark, gatsby-source-tmdb.