Nodes are created by calling the createNode action. Nodes can be any object.

A node is stored in Redux under the `nodes` namespace, whose state is a map of the node ID to the actual node object.

Sourcing Nodes

The creation of nodes occurs primarily in the sourceNodes bootstrap phase. Nodes created during this phase are top level nodes. I.e, they have no parent. This is represented by source plugins setting the node's `parent` field to `null`. Nodes created via transform plugins (who implement onCreateNode) will have source nodes as their parents, or other transformed nodes. For a rough overview of what happens when source nodes run, see the `traceId` illustration.

Parent/Child/Refs

There are a few different scenarios for creating parent/child relationships.

Node relationship storage model

All nodes in Gatsby are stored in a flat structure in the Redux `nodes` namespace. A node's `children` field is an array of node IDS, whose nodes are also at the top level of the Redux namespace. Here's an example of the `nodes` namespace.

```
{
  `id1`: { type: `File`, children: [`id2`, `id3`], ...other_fields },
  `id2`: { type: `markdownRemark`, ...other_fields },
  `id3`: { type: `postsJson`, ...other_fields }
}
```

An important note here is that we do not store a distinct collection of each type of child. Rather we store a single collection that they're all packed into. This has some implications on child field inference in the Schema Generation phase.

Explicitly recording a parent/child relationship

This occurs when a transformer plugin implements onCreateNode in order to create some child of the originally created node. In this case, the transformer plugin will call createParentChildLink, with the original node, and the newly created node. All this does is push the child's node ID onto the parent's `children` collection and resave the parent to Redux.

This does **not** automatically create a `parent` field on the child node. If a plugin author wishes to allow child nodes to navigate to their parents in GraphQL queries, they must explicitly set `childNode.parent: 'parent.id'` when creating the child node.

Foreign Key reference (`___NODE`)

We've established that child nodes are stored at the top level in Redux, and are referenced via ids in their parent's `children` collection. The same mechanism drives foreign key relationships. Foreign key fields have a `___NODE` suffix on the field name. At query time, Gatsby will take the field's value as an ID, and search Redux for a matching node. This is explained in more detail in schema gqlTypes.

Plain objects at creation time

Let's say you create the following node by passing it to `createNode`

```
{
  foo: 'bar',
  baz: {
    car: 10
  }
}
```

The value for `baz` is itself an object. That value's parent is the top level object. In this case, Gatsby saves the top level node as is to Redux. It doesn't attempt to extract `baz` into its own node. It does however track the subobject's root NodeID using Node Tracking

During schema compilation, Gatsby will infer the sub object's type while creating the gqlType.

Fresh/stale nodes

Every time a build is re-run, there is a chance that a node that exists in the Redux store no longer exists in the original data source. E.g. a file might be deleted from disk between runs. We need a way to indicate that fact to Gatsby.

To track this, there is a Redux `nodesTouched` namespace that tracks whether a particular node ID has been touched. This occurs whenever a node is created (handled by CREATE_NODE), or an explicit call to touchNode.

When a `source-nodes` plugin runs again, it generally recreates nodes (which automatically touches them too). But in some cases, such as transformer-screenshot, a node might not change, but we still want to keep it around for the build. In these cases, we must explicitly call `touchNode`.

Any nodes that aren't touched by the end of the `source-nodes` phase, are deleted. This is performed via a diff between the `nodesTouched` and `nodes` Redux namespaces, in source-nodes.ts

Changing a node's fields

From a site developer's point of view, nodes are immutable. In the sense that if you change a node object, those changes will not be seen by other parts of Gatsby. To make a change to a node, it must be persisted to Redux via an action.

So, how do you add a field to an existing node? E.g. perhaps in onCreateNode, you want to add a transformer specific field? You can call createNodeField and this will add your field to the node's `node.fields` object and then persists it to Redux. This can then be referenced by other parts of your plugin at later stages of the build.

Node Tracking

When a node is created, `createNode` will track all its fields against its nodeId. See Node Tracking Docs for more.