Connecting to PostgreSQL

Environment variables

node-postgres uses the same environment variables as libpq to connect to a PostgreSQL server. Both individual clients & pools will use these environment variables. Here's a tiny program connecting node.js to the PostgreSQL server:

```
const { Pool, Client } = require('pg')

// pools will use environment variables
// for connection information
const pool = new Pool()

pool.query('SELECT NOW()', (err, res) => {
  console.log(err, res)
  pool.end()
})

// you can also use async/await
const res = await pool.query('SELECT NOW()')
await pool.end()

// clients will also use environment variables
// for connection information
const client = new Client()
await client.connect()

const res = await client.query('SELECT NOW()')
await client.end()
```

To run the above program and specify which database to connect to we can invoke it like so:

```
$ PGUSER=dbuser \
  PGHOST=database.server.com \
  PGPASSWORD=secretpassword \
  PGDATABASE=mydb \
  PGPORT=3211 \
  node script.js
```

This allows us to write our programs without having to specify connection information in the program and lets us reuse th
to connect to different databases without having to modify the code.

The default values for the environment variables used are:

```
PGHOST='localhost'
PGUSER=process.env.USER
PGDATABASE=process.env.USER
PGPASSWORD=null
PGPORT=5432
```

Programmatic

node-postgres also supports configuring a pool or client programmatically with connection information. Here's our same script from above modified to use programmatic (hard-coded in this case) values. This can be useful if your application already has a way to manage config values or you don't want to use environment variables.

```
const { Pool, Client } = require('pg')

const pool = new Pool({
  user: 'dbuser',
  host: 'database.server.com',
  database: 'mydb',
  password: 'secretpassword',
  port: 3211,
})

pool.query('SELECT NOW()', (err, res) => {
  console.log(err, res)
  pool.end()
})

const client = new Client({
  user: 'dbuser',
  host: 'database.server.com',
  database: 'mydb',
  password: 'secretpassword',
  port: 3211,
})
client.connect()

client.query('SELECT NOW()', (err, res) => {
  console.log(err, res)
  client.end()
})
```

Connection URI

You can initialize both a pool and a client with a connection string URI as well. This is common in environments like Herok where the database connection string is supplied to your application dyno through an environment variable. Connection string parsing brought to you by pg-connection-string.

```
const { Pool, Client } = require('pg')
const connectionString = 'postgresql://dbuser:secretpassword@database.server.com:3211/mydb'

const pool = new Pool({
  connectionString: connectionString,
})

pool.query('SELECT NOW()', (err, res) => {
  console.log(err, res)
  pool.end()
})

const client = new Client({
  connectionString: connectionString,
})
client.connect()

client.query('SELECT NOW()', (err, res) => {
  console.log(err, res)
  client.end()
})
```