Knex.js Cheat Sheet

A comprehensive cheat sheet for Knex.js, a SQL query builder for Node.js, covering database connections, schema building, querying, migrations, and seeding.

Getting Started & Connections

Installation

Install Knex.js via npm or yarn:

```
npm install knex --save
# or
yarn add knex
```

Also, install the database driver for your database (e.g., pg for PostgreSQL, mysql for MySQL, sqlite3 for SQLite):

```
npm install pg mysql sqlite3 mssql --save
# or
yarn add pg mysql sqlite3 mssql
```

Initializing Knex

After setting up the connection, you can use the `knex` instance to build and execute queries.

```
const knex = require('knex')({
  client: 'sqlite3',
  connection: {
    filename: './dev.sqlite3'
  },
  useNullAsDefault: true
})

module.exports = knex;
```

Database Connections

PostgreSQL

```
const knex = require('knex')({
  client: 'pg',
  connection: {
    host : 'localhost',
    port : 5432,
    user : 'your_db_user',
    password : 'your_db_password',
    database : 'your_db'
  }
});
```

MySQL

```
const knex = require('knex')({
  client: 'mysql',
  connection: {
    host : '127.0.0.1',
    user : 'your_db_user',
    password : 'your_db_password',
    database : 'your_db'
  }
});
```

SQLite3

```
const knex = require('knex')({
  client: 'sqlite3',
  connection: {
    filename: './mydb.sqlite'
  },
  useNullAsDefault: true
});
```

MSSQL

```
const knex = require('knex')({
  client: 'mssql',
  connection: {
    server : 'your_server',
    user : 'your_db_user',
    password : 'your_db_password',
    database : 'your_db'
  }
});
```

Connection String

```
const knex = require('knex')({
  client: 'pg',
  connection: process.env.DATABASE_URL
});
```

Schema Builder

Creating Tables

```
knex.schema.createTable('users', function
(table) {
  table.increments('id');
  table.string('name');
  table.string('email').unique();
  table.timestamps();
}).then(() => console.log("table created"))
.catch((err) => { console.log(err); throw err
})
```

Column Types

`table.increments(columnName)`	Auto-incrementing primary key.
`table.string(columnName, length)`	String column with optional length.
`table.integer(columnName)`	Integer column.
`table.boolean(columnName)`	Boolean column.
`table.date(columnName)`	Date column.
`table.datetime(columnName)`	Datetime column.
`table.timestamp(columnName)`	Timestamp column.
`table.json(columnName)`	JSON column.
`table.jsonb(columnName)`	JSONB column.

Constraints

`table.unique(columns)`	Adds a unique constraint to the specified column(s).
`table.primary(columns)`	Sets the specified column(s) as the primary key.
`table.foreign(column).references(refColumn).inTable(refTable)`	Adds a foreign key constraint.

Query Builder

Basic Queries

Selecting data:

```
knex.select('*').from('users').then(rows => {console.log(rows)})
```

Inserting data:

```
knex('users').insert({ name: 'John', email: 'john@example.com' }).then(() => console.log("inserted"))
```

Updating data:

```
knex('users').where('id', 1).update({ name: 'Jonathan' }).then(() => console.log("updated"))
```

Deleting data:

```
knex('users').where('id', 1).del().then(() => console.log("deleted"))
```

Where Clauses

`where(column, value)`	Basic where clause.
`where(column, operator, value)`	Where clause with operator (e.g., '=', '>', '<').
`where(object)`	Where clause with an object for multiple conditions.
`whereIn(column, array)`	Where column's value is in the array.
`whereNull(column)`	Where column's value is null.
`whereBetween(column, [value1, value2])`	Where column's value is between value1 and value2.

Joins

```
knex('users')
  .join('contacts', 'users.id', 'contacts.user_id')
  .select('users.name', 'contacts.phone')
```

Other join types:

`leftJoin`, `rightJoin`, `outerJoin`, `innerJoin`

Migrations & Seedings

Migrations Setup

Initialize Knex migrations:

```
knex init
```

This command creates a `knexfile.js` (or `.ts`) in your project root, where you can configure database connections for different environments.

Migration Commands

`knex migrate:make migration_name`	Creates a new migration file.
`knex migrate:latest`	Runs all pending migrations.
`knex migrate:rollback`	Rolls back the latest migration batch.
`knex migrate:status`	Shows the current migration status.

Seedings

Create seed file:

```
knex seed:make seed_name
```

Run seed file:

```
knex seed:run
```

Example Seed File:

```
exports.seed = function(knex, Promise) {
  // Deletes ALL existing entries
  return knex('users').del()
    .then(function () {
      // Inserts seed entries
      return knex('users').insert([
        {id: 1, name: 'John', email:
'john@example.com'},
        {id: 2, name: 'Jane', email:
'jane@example.com'},
        {id: 3, name: 'Mike', email:
'mike@example.com'}
      ]);
    });
};
```