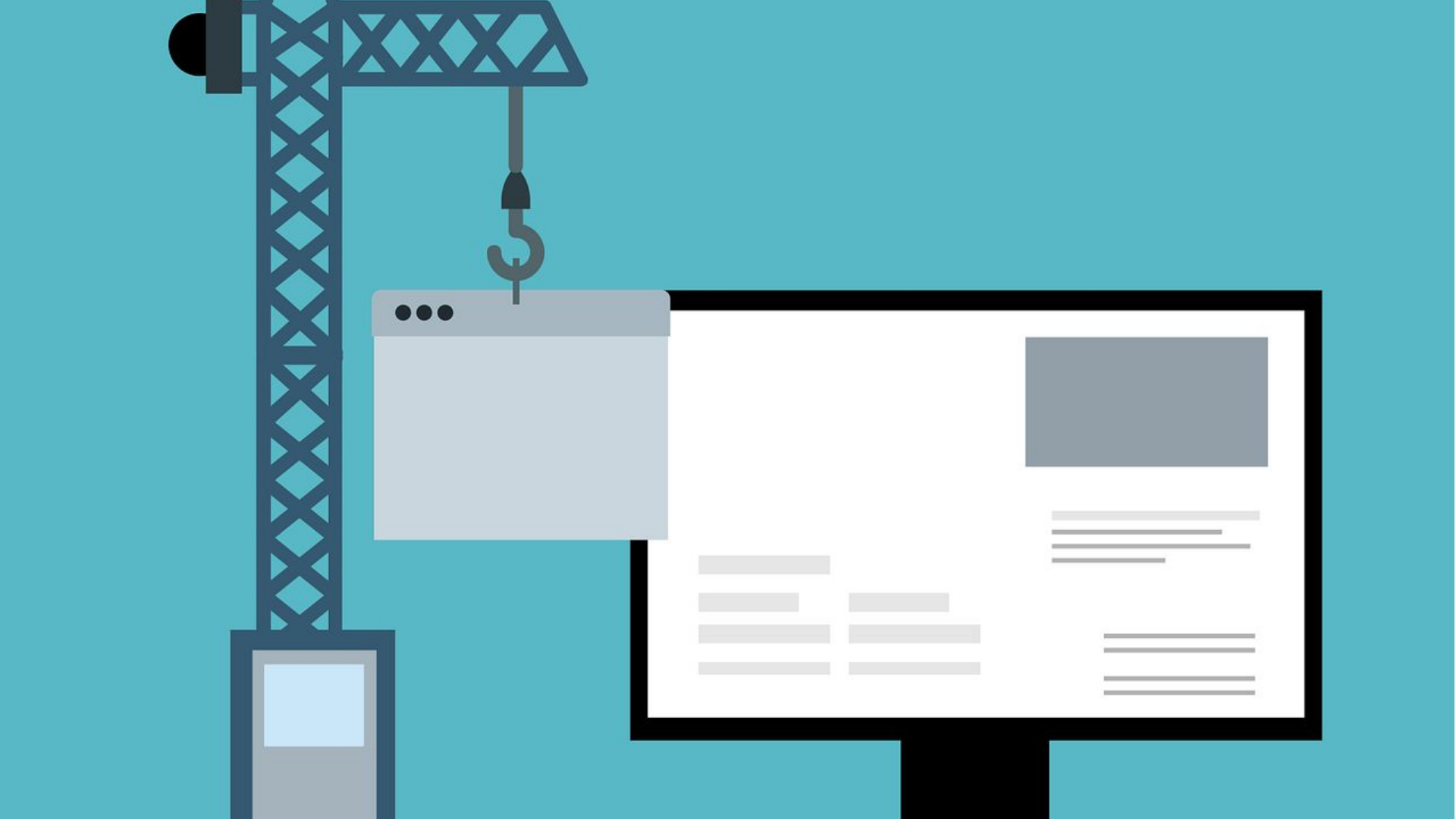

API

- Application programming interface

API

- used to provide an interface for web sites and client applications to have data access

REST

- Representational State Transfer

de-facto way to construct a web API

- follow the RESTful approach

REST APIs

- resource-based interfaces
    - Represent data resources (eg. JSON, XML) by URIs (paths)
    - Access via HTTP

- Actions such as **CRUD** *(Create, Read, Update, and Delete)* are made against these resources with **HTTP methods** *(POST, GET, PUT, PATCH, DELETE)*

JS on the Server

Client
(Browser)

request

Server

https://dogs.com

Client
(Browser)

Response
(HTML Page)

Server
https://dogs.com

- RESTful constraints

Client-server architecture

- Separate of concerns
    - RESTful API shouldn't care about UI

Stateless

- Each request made to a REST

  API is stateless.

Stateless

- no client-context (eg. session) is stored on the server

Stateless

- server handling your request maintains

  no context between requests

- each request is interpreted equally

Stateless

- Any context required to process the request

  must be provided with the request itself

  - (eg. an authorization token).

Cacheability

- Responses must define themselves as cacheable or non-cacheable

Cacheability

- based on context returned with the request's response by the server, you can cache the item.

Cacheability

- server may return explicit instructions for how long a resource may be cached

Cacheability

- eg) instructs not to cache
- eg) provides a recommended length of time to cache before refreshing

Layered system

- intermediate servers may be used

  without the client knowing about it

Uniform interface

- Good REST APIs maintain a uniform interface.

Uniform interface

- same shape of request which is made against a particular resource can reasonably be expected to act the same way against another resource

Uniform interface

- Example:
- The request modifying an attribute on a User resource is uniform to that of modifying a similar attribute on a Course resource

Uniform interface

- resources are identified in requests
- Transferred data is decoupled from DB schema
- Self-descriptive messages
- Links to further resources

Code on demand (optional)

- Executable code should be transferable

Express.js

- great for constructing a REST API
- provides an easy interface to segregate resources by type and action.

Routes

- used in Express for defining application behaviour to run when a request is received.