Full Stack Application Development

BITS Pilani

innovate achieve lead

BITS Pilani

Pilani Campus

Request – Response APIs

Request-Response APIs

innovate achieve lead

Typically expose an interface through a HTTP-based web server

APIs define a set of endpoints

- ✓ Clients make HTTP requests for data to those endpoints
- ✓ Server returns responses

The response is typically sent back as JSON or XML

Three common paradigms used to expose request–response APIs:

- ✓ REST
- ✓ RPC
- ✓ and GraphQL

Representational State Transfer

The most popular choice for API development lately

- ✓ Used by providers like Google, Stripe, Twitter, and GitHub

REST is about resources

- ✓ A resource is an entity that can be identified, named, addressed, or handled on the web
- ✓ REST APIs expose data as resources
- ✓ Use standard HTTP methods to represent Create, Read, Update, and Delete (CRUD) transactions against these resources

General rules REST APIs follow:

- ✓ Resources are part of URLs, like /books
- ✓ For each resource, generally two URLs are implemented
  - ❖ one for the collection, like /books
  - ❖ one for a specific element, like /books/B123
- ✓ Nouns are used instead of verbs for resources
  - ❖ instead of /getBoooksInfo/B123, use /books/B123.
- ✓ HTTP methods like GET, POST, UPDATE, and DELETE inform the server about the action to be performed
- ✓ Standard HTTP response status codes are returned by the server indicating success or failure
  - ❖ codes in the 2XX range indicate success
  - ❖ 3XX codes indicate a resource has moved
  - ❖ codes in the 4XX range indicate a client-side error
  - ❖ codes in the 5XX range indicate server-side errors
- ✓ REST APIs might return JSON or XML responses
  - ❖ Due to its simplicity and ease of use with JavaScript, JSON has become the standard for modern APIs

CRUD operations, HTTP verbs, and REST conventions

innovate achieve lead

Operation	HTTP verb	URL: /books	URL: /books/123
Create	POST	Create new book	Not applicable
Read	GET	Get list of all books	Get one particular book with given ID
Update	PUT / PATCH	Batch update books	Update a particular book with given ID
Delete	DELETE	Delete all books	Delete a particular book with given ID

GITHUB API for getting user profile data

```
# GET /users/defunkt
$ curl https://api.github.com/users/defunkt

> {
>   "login": "defunkt",
>   "id": 2,
>   "url": "https://api.github.com/users/defunkt",
>   "html_url": "https://github.com/defunkt",
>   ...
> }
```

Remote Procedure Call

One of the simplest API paradigms, in which a client executes a block of code on another server

RPC is about actions

- ✓ Clients typically pass a method name and arguments to a server
- ✓ Receive back JSON or XML

Simple rules:

- ✓ The endpoints contain the name of the operation to be executed
- ✓ API calls are made with the HTTP verb that is most appropriate
- ✓ GET for read-only requests
- ✓ POST for others

Works great for APIs that

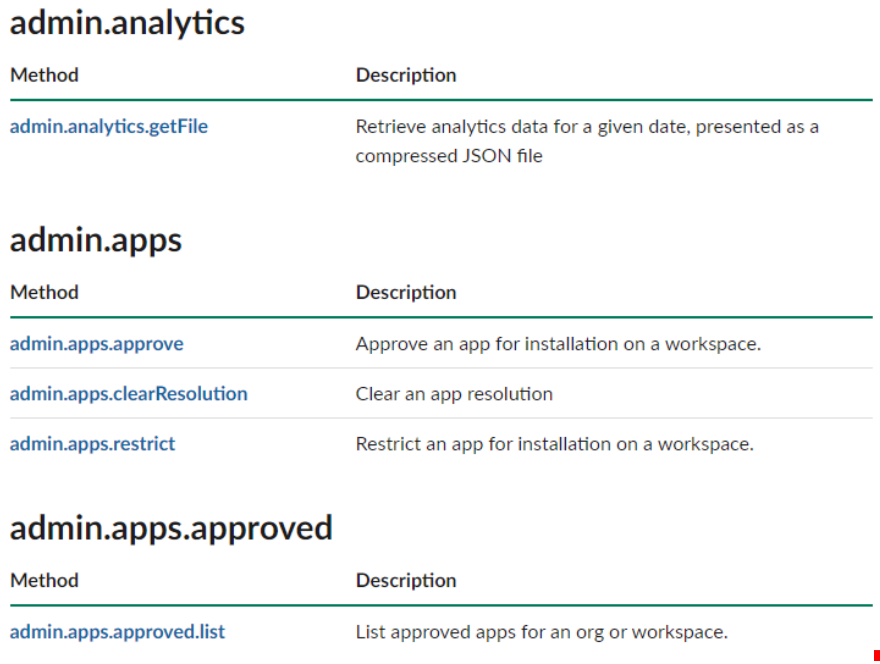
- ✓ expose a variety of actions which have complications that can be encapsulated with CRUD
- ✓ for which there are side effects unrelated to the “resource” at hand
- ✓ accommodate complicated resource models or actions upon multiple types of resources

Using the Slack Web API https://api.slack.com/web#basics

The Slack Web API is an interface for querying information from and enacting change in a Slack workspace

The Web API is a collection of HTTP RPC-style methods

All with URLs in the form https://slack.com/api/METHOD_FAMILY.method



admin.analytics

Method	Description
admin.analytics.getFile	Retrieve analytics data for a given date, presented as a compressed JSON file

admin.apps

Method	Description
admin.apps.approve	Approve an app for installation on a workspace.
admin.apps.clearResolution	Clear an app resolution
admin.apps.restrict	Restrict an app for installation on a workspace.

admin.apps.approved

Method	Description
admin.apps.approved.list	List approved apps for an org or workspace.

Remote Procedure Call (3)

RPC-style APIs are not exclusive to HTTP

Other high performance protocols that are available for RPC-style API

- ✓ Apache Thrift
- ✓ gRPC

Apache Thrift

JSON options available for gRPC

- ✓ Both Thrift and gRPC requests are serialized
- ✓ Thrift and gRPC also have built-in mechanisms for editing the data structures

GraphQL

innovate achieve lead

GraphQL is a query language for APIs that has gained significant momentum recently

- ✓ Developed internally by Facebook, publicly released in 2015
- ✓ Adopted by API providers like GitHub, Yelp, and Pinterest

GraphQL allows clients to define the structure of the data required

Server returns exactly that structure

```
{
  hero {
    name
    # Queries can have comments!
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
```

GraphQL APIs need only a single URL endpoint

- ✓ Do not need different HTTP verbs to describe the operation
- ✓ Instead, indicate in the JSON body whether you're performing a query or a mutation

Comparison of request–response API paradigms

	REST	RPC	GraphQL
What?	Exposes data as resources Uses standard HTTP methods to represent CRUD operations	Exposes action-based API methods—clients pass method name and arguments	Exposes action-based API methods—clients pass method name and arguments
Providers	Stripe, GitHub, Twitter, Google	Slack, Flickr	Slack, Flickr
Example usage	GET /books/<id>	GET /books.get?id=<id>	query ($id: String!) { book() { name author } }
HTTP Verbs	GET, POST, PUT, PATCH, DELETE	GET, POST	GET, POST

Comparison (Contd)

	REST	RPC	GraphQL
Pros	• Standard method name, arguments format, and status codes • Utilizes HTTP features • Easy to maintain	• Easy to understand • Lightweight payloads • High performance	• Saves multiple round trips • Avoids versioning • Smaller payload size • Strongly typed • Built-in introspection
Cons	• Big payloads • Multiple HTTP round trips	• Discovery is difficult • Limited standardization • Can lead to function explosion	• Requires additional query parsing • Backend performance optimization is difficult • Too complicated for a simple API
When to use?	For APIs doing CRUD like operations	For APIs exposing several actions	When you need querying flexibility; great for providing querying flexibility and maintaining consistency

GraphQL

What is GraphQL

- An open-source query language for APIs and a runtime for fulfilling those queries with existing data
- Enables clients to request exactly the data they need from an API
- Eliminates over-fetching and under-fetching of data

GraphQL is a query language that allows clients to request exactly the data they need from an API. This is in contrast to traditional REST APIs, which often require clients to make multiple requests to fetch all of the data they need. GraphQL eliminates the need for over-fetching and under-fetching of data, which can improve the performance of your applications.

REST

- REST is an architectural style for designing networked applications.
- It typically uses standard HTTP methods like GET, POST, PUT, DELETE, etc., to perform CRUD (Create, Read, Update, Delete) operations on resources.
- RESTful APIs expose a set of endpoints, each representing a resource (e.g., /users, /posts).
- Responses are usually in JSON or XML format.
- Clients have to make multiple requests to fetch related data or nested resources.
- The structure of the response is predefined by the server, and clients receive all the data defined in the response schema.

Example

Let's consider a simple RESTful API for a blog platform:

- To fetch all posts: **GET /posts**
- To fetch a specific post: **GET /posts/{postId}**
- To create a new post: **POST /posts**
- To update a post: **PUT /posts/{postId}**
- To delete a post: **DELETE /posts/{postId}**

GraphQL

Query to fetch all posts with their titles and authors:

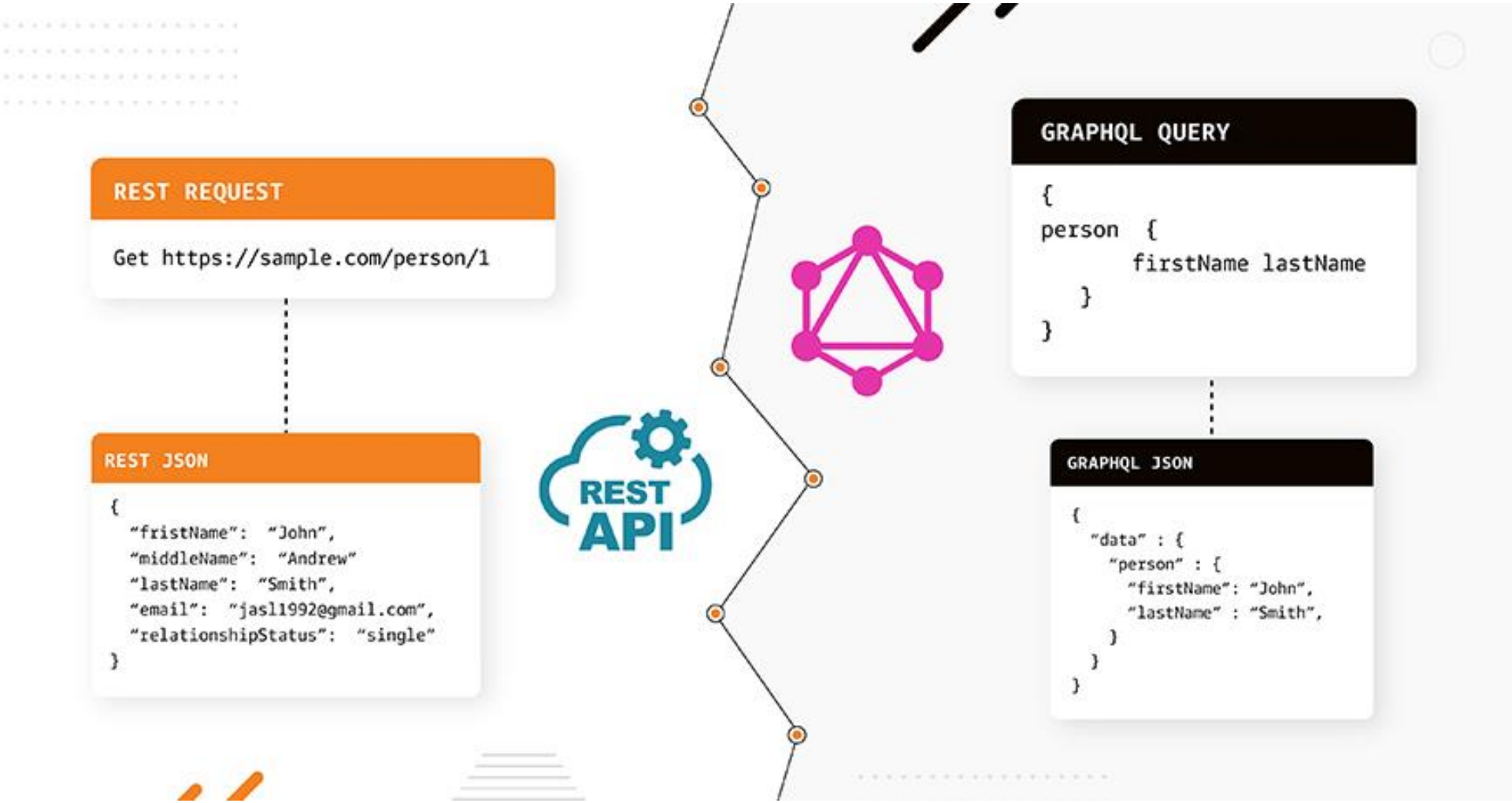
```
query {
  posts {
    id
    title
    author {
      name
    }
  }
}
```

GraphQL

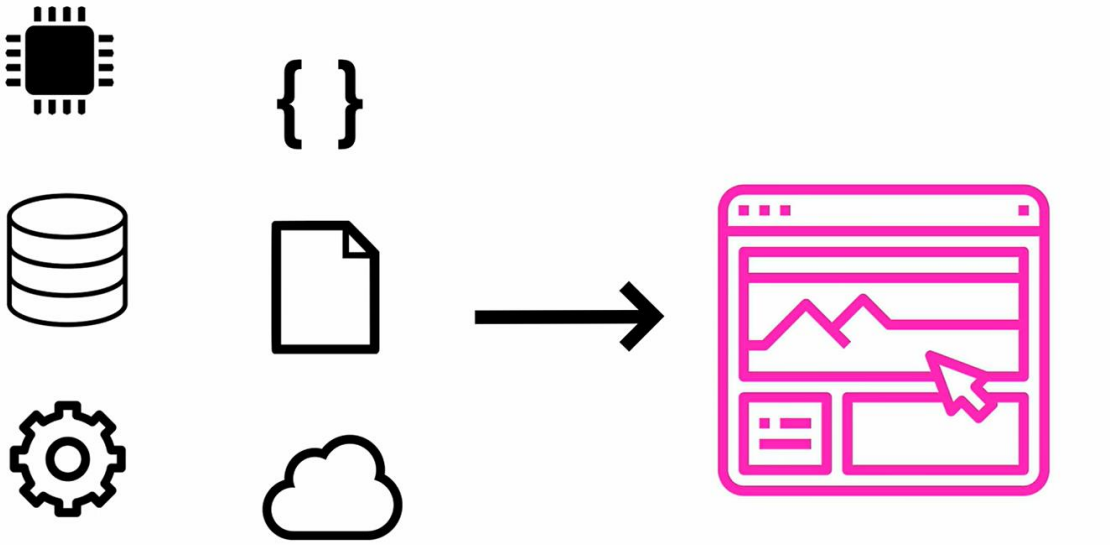
Query to fetch a specific post with its comments:

```
query {
  post(id: "postId") {
    title
    content
    comments {
      id
      text
      user {
        name
      }
    }
  }
}
```

REST vs GraphQL



GraphQL



GraphQL is a query language for your API. It was created at Facebook and open sourced in 2015. And something to really remember about GraphQL is that it's a spec. It's a document that describes everything that is part of the language, including a query language and a schema definition language for designing your API's types. GraphQL gives us a way of taking all of our data sources, our rest APIs, our databases, our cloud services, and put it into some sort of user interface

DSL

GraphQL also gives us a domain specific language, in other words, a way of designing our API's types in a readable language that everybody can understand. We can pull everybody together to work on the schema, and then everybody can go off and work on their own parts of the application based on that document as a single source of truth.

How does GraphQL Works?

GraphQL APIs are defined using a schema, which specifies the types of data that can be queried.

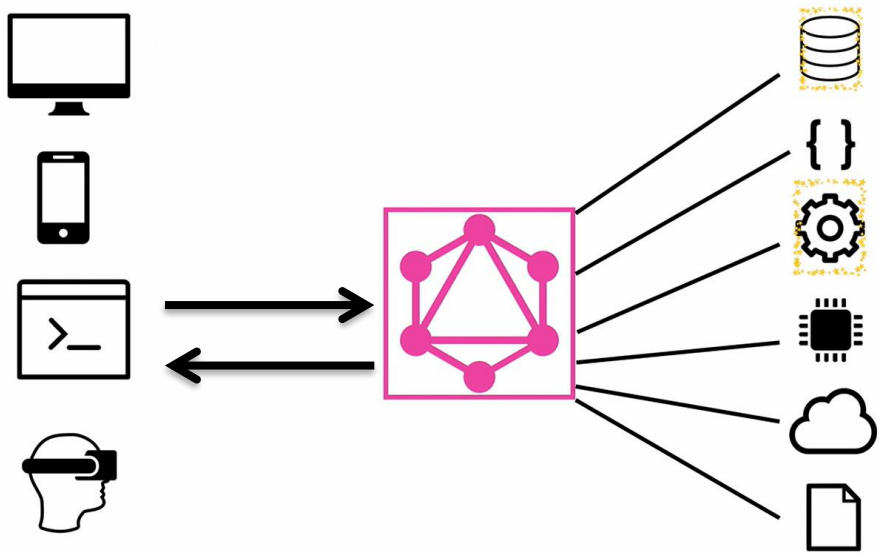
Clients can request data using queries, which are structured like the data they want to receive.

Resolvers on the server side fetch the requested data and return it to the client in the requested format.

How GraphQL works?

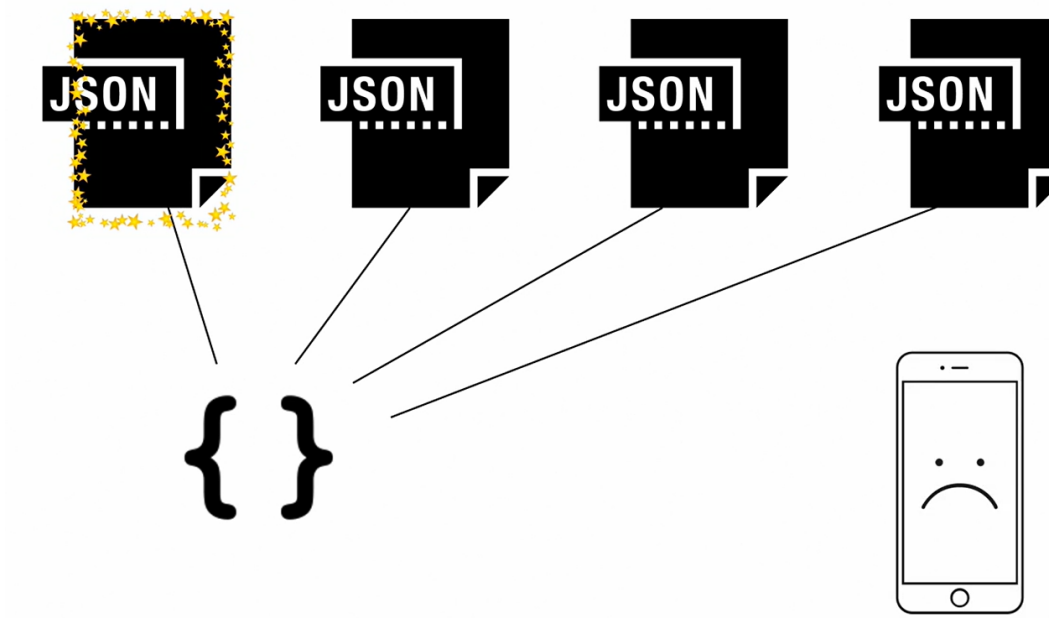
1. Clients send queries to the GraphQL server.
2. Queries specify the data that the client needs. The query syntax is similar to JSON and allows you to specify the fields you want to be returned for each object.
3. The server executes the query and returns the requested data. The response is also in JSON format and includes only the data that was requested in the query.

Single Query, Single Response



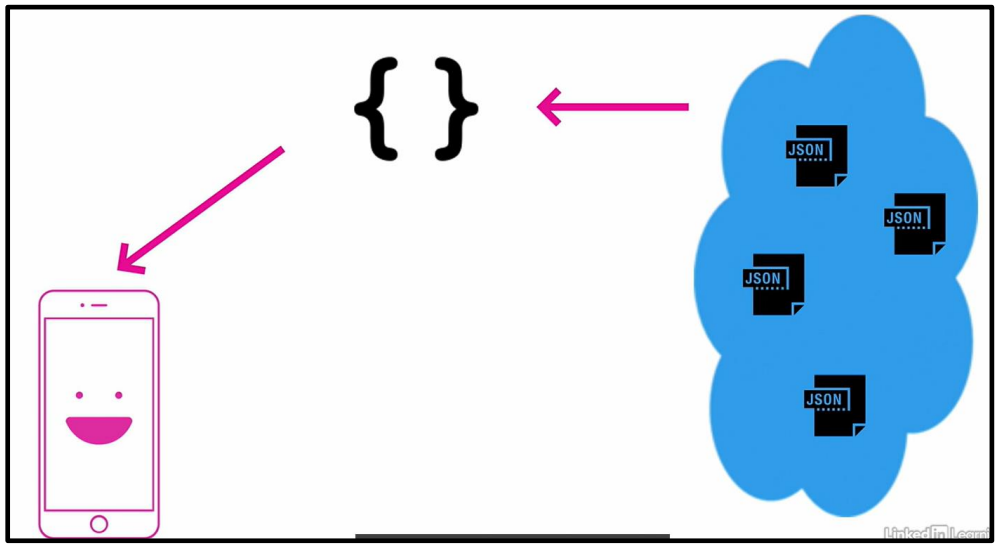
So over here on the left side of the screen, we have all of our clients, phones, CLIs, desktop applications, even VR, and we can send a query to a GraphQL server that sits in front of all of the various data sources. This will then collect information from all of those sources and return the data in a single response, so that's the idea with a GraphQL query. **We have a single query and a single response**

Client-Side Shaping



Also, once we get back these huge blobs of JSON data, they're in no shape to display. We have to parse it, sort it, filter it, do all the stuff we need to do on the client, turn the data into the shape that we want, and that's a lot of effort, particularly if we're using a phone.

Client – Only Rendering



Another really amazing thing about GraphQL is that we can move a lot of the sorting, parsing, and filtering back to the server, transform the data into the shape that we want, and then we can use the phone simply for rendering. We can use the client for rendering, which is going to be a lot faster, and it makes those clients a lot happier.

Advantages?

Flexible and intuitive data fetching

Better performance and reduced network overhead

Strongly typed schema provides documentation and validation

Allows for easy evolution of APIs without breaking existing clients

Tools and Libraries

- Apollo: A popular GraphQL client and server library
- GraphiQL: A web-based IDE for exploring and testing GraphQL APIs
- Prisma: An ORM and query builder for GraphQL backends
- Hasura: A platform for building GraphQL APIs on top of existing databases.

innovate achieve lead

BITS Pilani

Pilani Campus

API Development Platform

API Development Platform

Offers a toolbox for API design and development

- ✓ Contains API building blocks, which are proven, reusable and configurable
- ✓ APIs are constructed by composing these building blocks
- ✓ There is no need to reinvent the wheel for the development of each new API

Offers an integrated engineering environment with tools for the development and design of APIs

The toolbox consists of:

- ✓ Library of API building blocks
- ✓ Language for implementing APIs
- ✓ IDE for API development with editor, debugger and deployment tools
- ✓ Language for designing APIs
- ✓ Design tool for creating API interface designs
- ✓ Tool for generating documentation and code skeletons based on the design.

Targeted at the API developers, who work for the API provider

- ✓ Supports the developers and enables them to develop APIs quickly and with high quality

Library of API Building Blocks

When developing APIs, certain functionality is needed over and over again

Extremely helpful to have this functionality available in a shared library of building blocks

- ✓ Tested and proven in practice, so bugs are extremely seldom
- ✓ Configurable and can be adapted for many purposes
- ✓ Composable, i.e. an API can be built from a collection of properly configured blocks

The building blocks has the following features:

- ✓ Processing of HTTP requests and HTTP responses
  - ❖ including header parameters, query parameters, URI processing, HTTP status code, HTTP methods
- ✓ Security
  - ❖ threat protection, IP-based access limitation, location-based access limitation, time-based access limitation
- ✓ Frontend authentication and authorization with OAuth, Basic Authorization, API key
- ✓ Frontend protocols
  - ❖ REST, SOAP, XML-RPC, JSON-RPC, WebSockets,
- ✓ Protocol mediation - SOAP to REST, REST to SOAP
- ✓ Data format transformation - XML to JSON, JSON to XML
- ✓ Aggregation and orchestration of multiple APIs and/or multiple backend services
- ✓ Throttling to protect backends and platforms - rate limitation and throughput limitation
- ✓ Load balancing for incoming requests to the API platform and outgoing requests to the backends
- ✓ Cache for incoming requests to the API platform and outgoing requests to the backends
- ✓ Hooks for logging, analytics
- ✓ Monetization capabilities and enforcement

Language for Designing APIs

The design of the API needs to be described

API development platforms provide two types of languages:

- ✓ a low-level API implementation language
- ✓ a high-level API description language

API description language

- ✓ Used to express the "what" of APIs

API development Platform provides

- ✓ design tools for creating API interface designs
- ✓ tools for generating documentation from the API description
- ✓ tools for generating implementations

An API description language is a domain-specific language for expressing API design

- ✓ Provides the well-defined semantics and the tooling ecosystem

Open API

https://swagger.io/specification/

Product API Specification

REST API	API Client/Consumer
GET /product	Receives `[ {"id":1, "name":"Shirt"}, {"id":2, "name":"Shorts"} ]`
GET /product/1	Receives `{"id":1, "name":"Shirt"}`
POST /product	Receives `{ "id":3, "name":"Lemon Water" }`

How you and your co-worker describe the API

Your Approach

```
{
  "resource":"product",
  "endpoints":[{
    "url":"/product",
    "method":"get",
    "output": “array",
  },{
    "url":"/product/{id}",
    "method": "get",
    "input": "int",
    "output": "object"
  }]
```

Coworker Approach

```
{
  "method":"get",
  "urls":[{
    "path":"/product",
    "output": "list",
  },{
    "path":"/product/{id}",
    "input": "integer",
    "output": "product"
  }]
```

Swagger VS OpenAPI

innovate achieve lead

Swagger

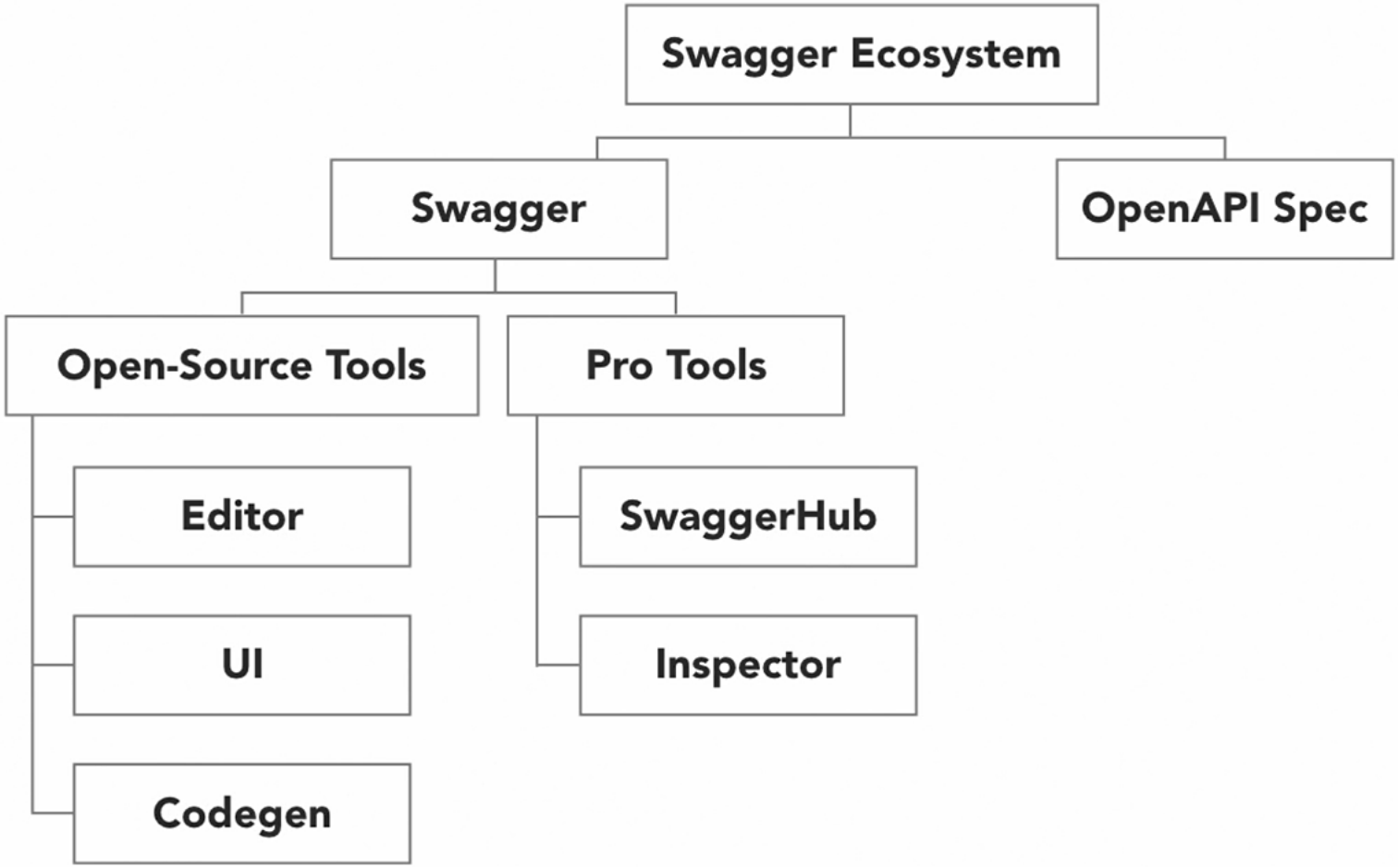
- A toolset and structured approach for creating API designs, documentation, and code throughout the API lifecycle

OAS

- Standard format for metadata used to define RESTful services

- Microsoft
- Google
- IBM
- PayPal
- SmartBear

Swagger

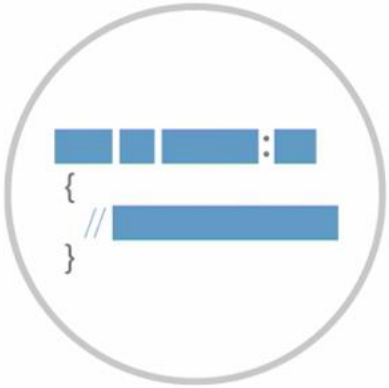

Swagger Ecosystem
Swagger
OpenAPI Spec
Open-Source Tools
Pro Tools
Editor
UI
Codegen
SwaggerHub
Inspector

Eco System

innovate achieve lead

Swagger Editor

Create OpenAPI definitions.

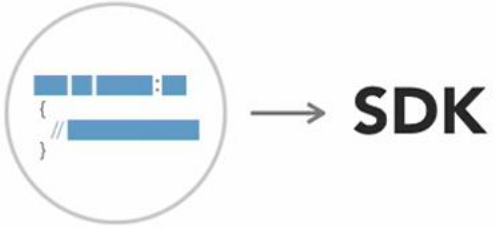


Swagger UI

Generate docs from API definitions.

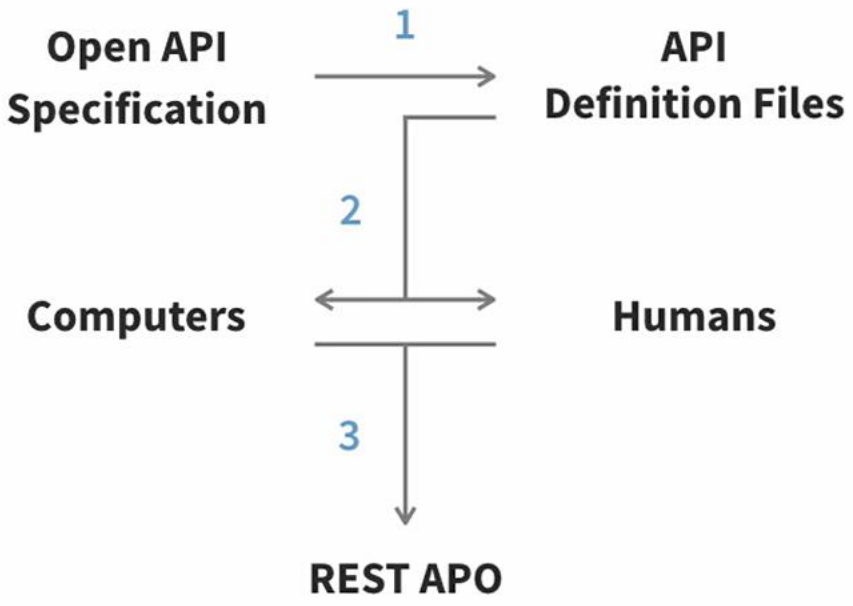
Swagger Codegen

Build client SDKs from API definitions.

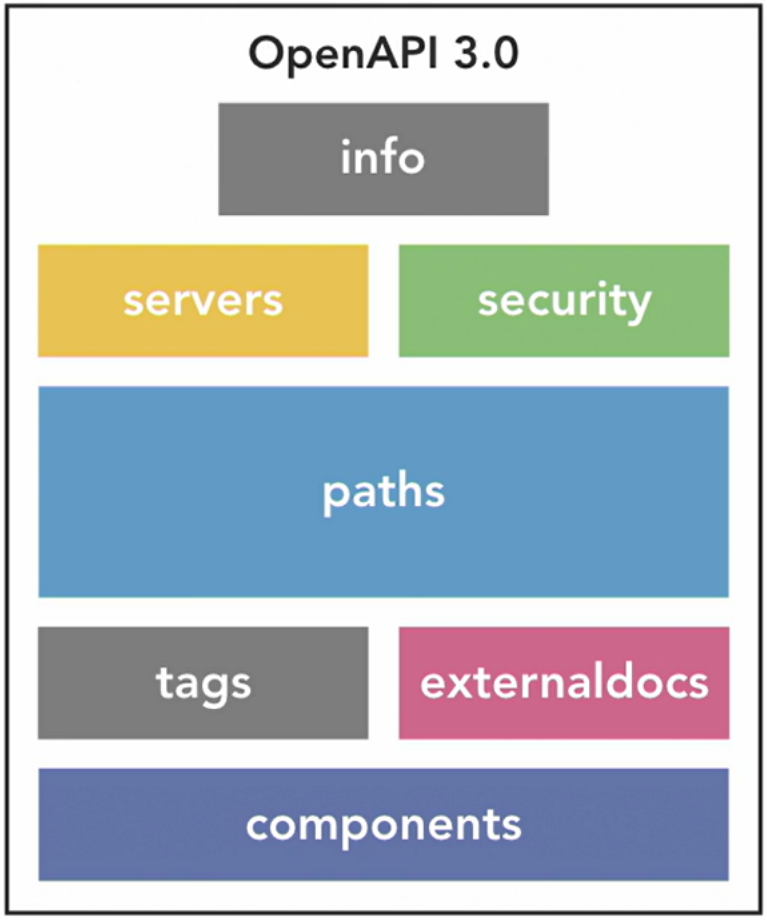


OpenAPI

The OpenAPI Specification tells us how to build API definition files that can be understood by humans or computers to support interaction with REST-based services.



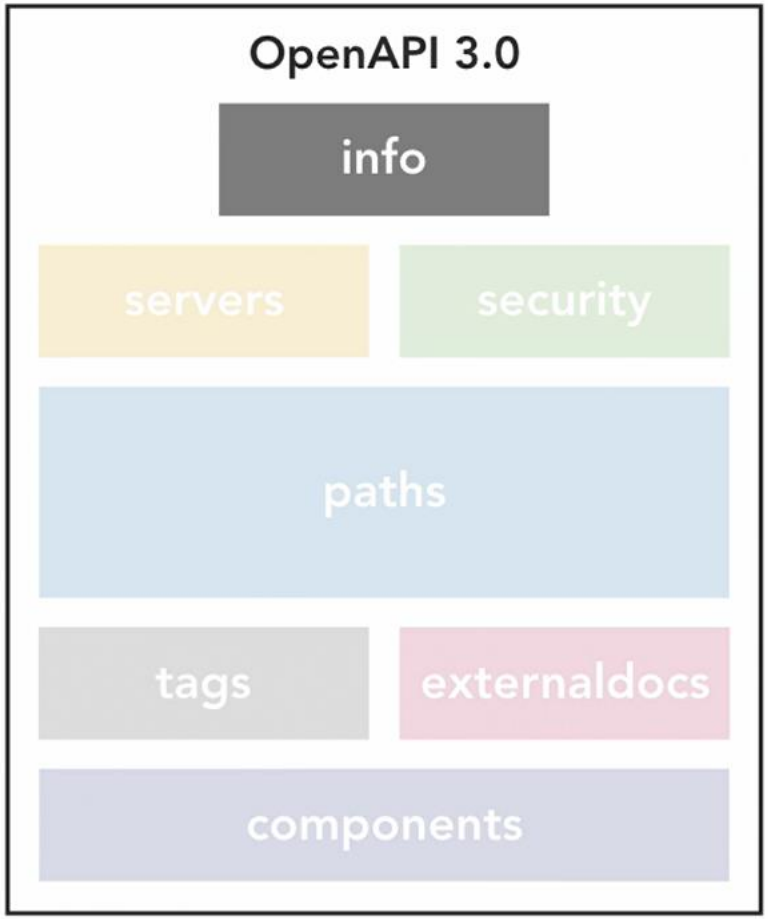
OpenAPI


OpenAPI 3.0
info
servers
security
paths
tags
externaldocs
components

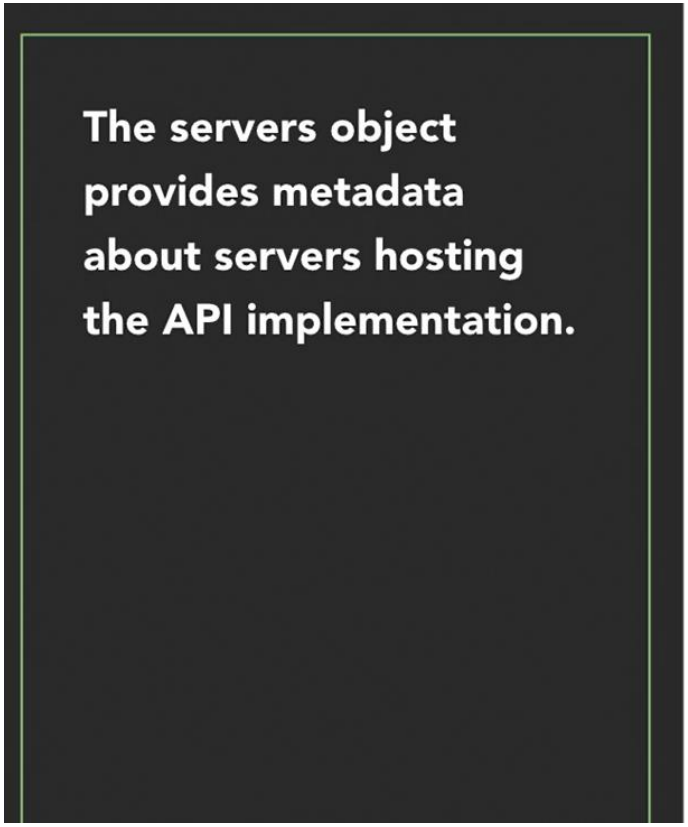
info

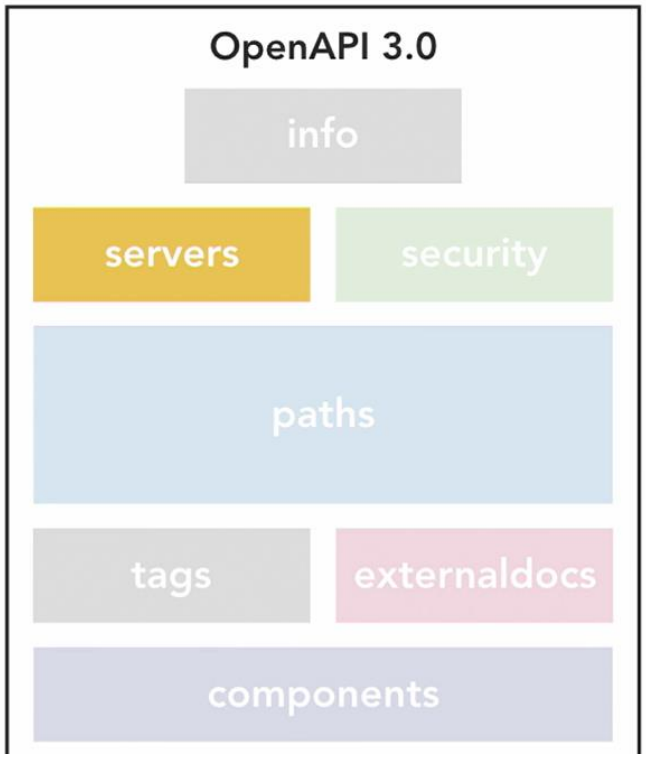
The info object provides metadata about the API such as:

- Description
- Terms of service
- Contact information
- License
- Version



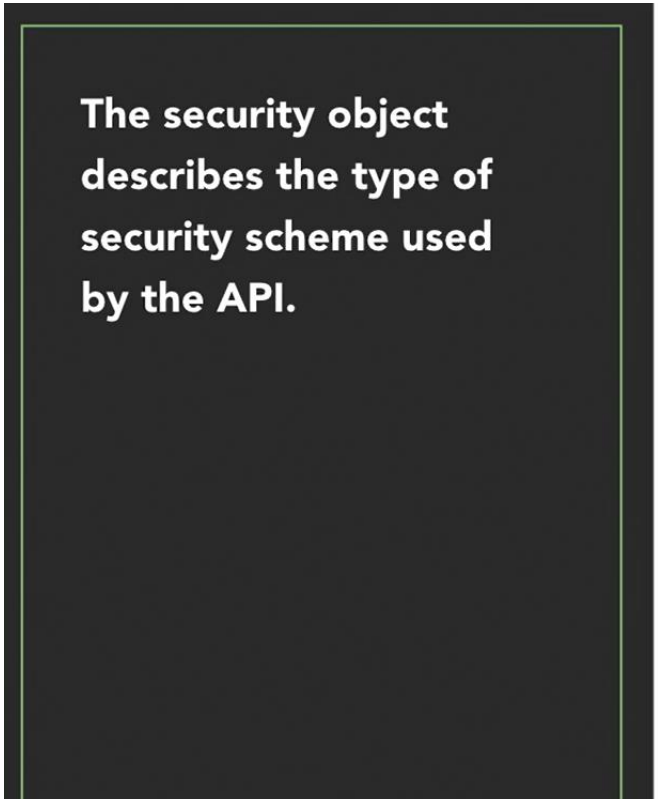
servers





The server's object can identify the URL and a description of the servers hosting the API. This can be very useful for tagging development servers. Often we'll provide URLs to the multiple environments that are hosting our API implementations such as URLs to the dev, UAT, and production instances of our API.

security



The security object allows you to describe how the API is secured by specifying information regarding API keys, OWA flows, or security cookies. When consumers of the API definition read this section, they will understand what is required from an authentication and authorization perspective for access to API resources.

paths

The paths object describes the API endpoints and the different operations that can be performed on those endpoints.

Within the paths object, we can specify multiple paths or end points that describe how to access resources on the API. The paths object probably requires the most information of all sections of the spec because we describe the incoming and outgoing data requirements in detail.

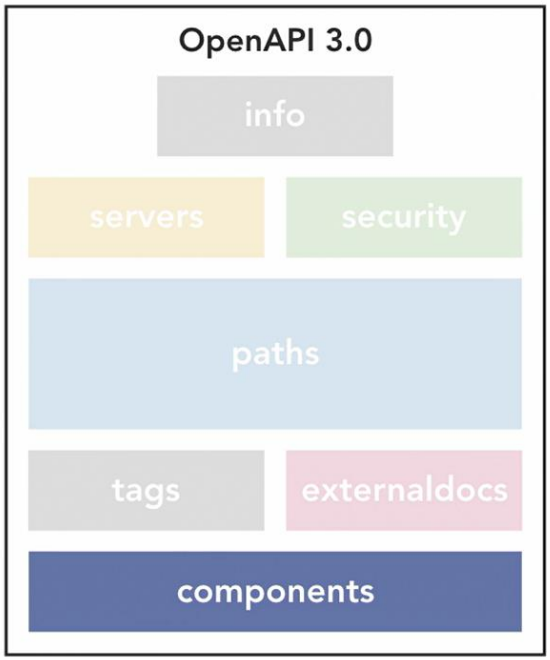
Tags , externaldocs

The tags and externalDocs objects supply additional metadata and links to external API documentation.

Using the tags object, we can define tags that can be used to group API operations. The external doc section allows us to provide links to external resources about the API that are hosted elsewhere.

components

The components object defines a set of reusable objects that can be referenced throughout the API.



Finally, the components object allows you to define common schema objects, parameters, request bodies, and responses that may be used throughout the API. This section creates the ability to reuse components as opposed to repeating the same definitions over and over. Without this reusability, API definitions would be much harder to manage. So that is a glimpse of the open API specification at the top level. We'll dive into the details for most of these sections so you can build your API definition files to suit your needs.

innovate achieve lead

BITS Pilani

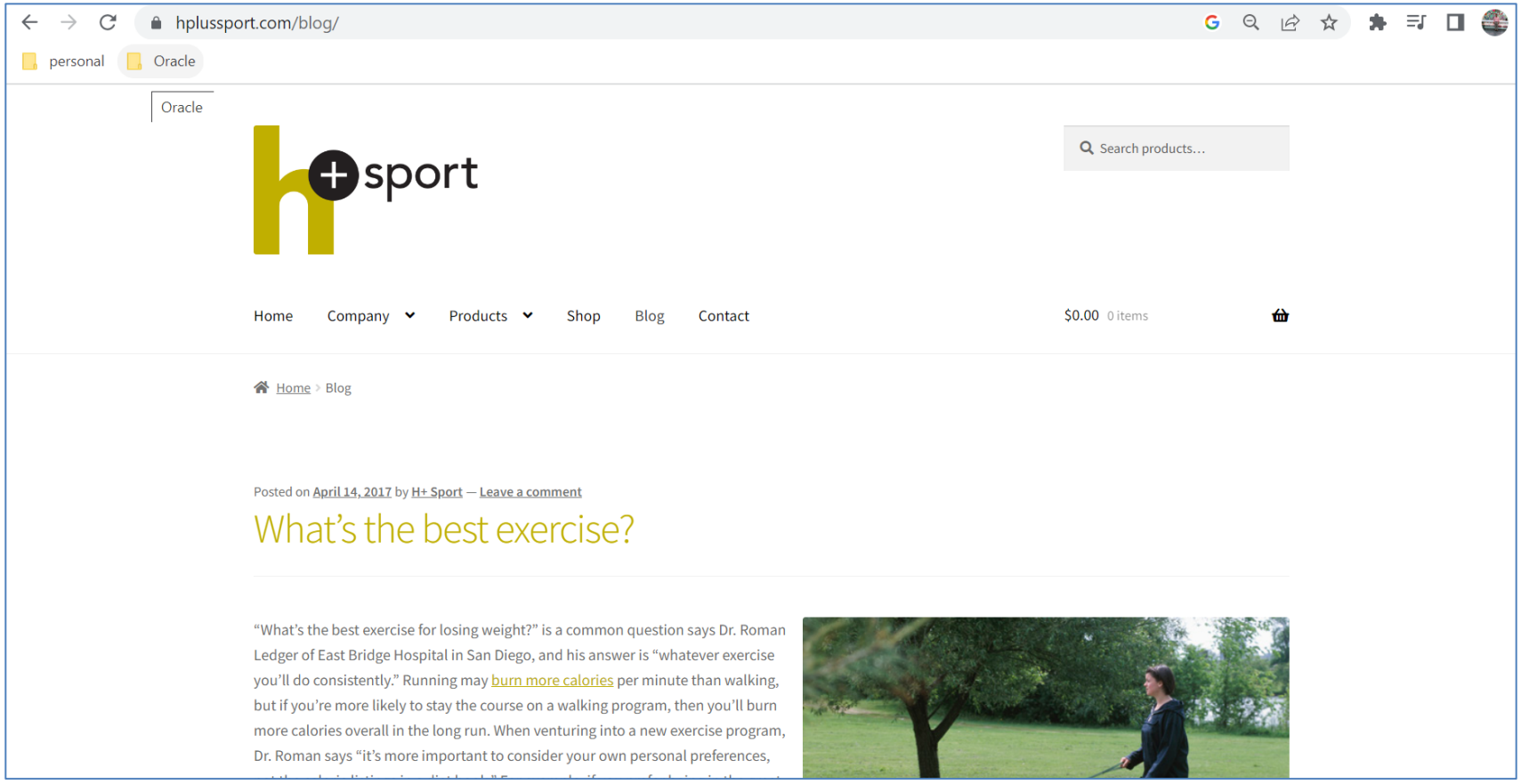
Pilani Campus

Open API – Lab-3

Lab-3

innovate achieve lead

https://hplussport.com/blog/



Lab-3

- Create an OpenAPI 3.0 definition for the blog API
- Build an endpoint to retrieve an array of articles

build a definition file for the API that powers the H Plus Sport Company's blog. To complete the challenge, you must create a new API definition file that exposes a single path used to access an array of articles. The array returned by the end point is used to populate the list of articles found on the front page of the blog. Before you dive in, let me give you a hint. Start out by inspecting the H Plus Sport blog to determine the properties required for an article schema object. At a quick glance, I would recommend adding properties for the created date, tagline, content, category, and tags.

Lab-3

innovate achieve lead

Article Resource

Posted on April 14, 2017 by H+ Sport — 1 Comment

What's the best exercise?

"What's the best exercise for losing weight?" is a common question says Dr. Roman Ledger of East Bridge Hospital in San Diego, and his answer is "whatever exercise you'll do consistently." Running may burn more calories per minute than walking, but if you're more likely to stay the course on a walking program, then you'll burn more calories overall in the long run. When venturing into a new exercise program, Dr. Roman says "it's more important to consider your own personal preferences, not the calorie listings in a diet book." For example, if you prefer being in the great outdoors to exercising indoors, you may find that hiking becomes a constant in your life, whereas the treadmill might be tossed aside after a brief interlude. Start with your likes and dislikes, and find ways to make exercise a part of your daily life, such as walking the dog. With this approach, you're likely to make a long commitment to an exercise program simply because it naturally enhances your life.

Category: Exercise

Tags: exercise, treadmill, walking, weight loss

Properties

- Created date
- Tagline
- Content
- Category
- Tags