⚡ API Quick Reference ⚡

🔷 HTTP Verbs

├ 🚀 **GET** : Retrieve data from the server

├ 🚀 **POST** : Send data to the server to create a resource

├ 🚀 **PUT** : Send data to the server to update a resource

├ 🚀 **PATCH** : Send data to the server to update a resource partially

└ 🚀 **DELETE** : Delete a resource from the server

├ 🚀 **TRACE** : Returns the full HTTP request received by the server for debugging and diagnostic purposes

├ 🚀 **OPTIONS** : Returns the HTTP methods supported by the server for the requested URL

├ 🚀 **CONNECT** : Converts the request connection to a transparent TCP/IP tunnel for secure communication

└ 🚀 **PURGE** : Invalidates a cached resource

├ 🚀 **LOCK** : Locks the resource for exclusive use by the client

├ 🚀 **UNLOCK** : Unlocks the resource previously locked by the client

├ 🚀 **MKCOL** : Creates a new collection resource

└ 🚀 **COPY** : Copies the resource identified by the Request-URI to the destination URI.

🔷 HTTP Status Codes

├ 🚦 **1xx** : Informational

├ 🚦 **2xx** : Success

├ 🚦 **3xx** : Redirection

├ 🚦 **4xx** : Client Errors

└ 🚦 **5xx** : Server Errors

⚡ API Quick Reference ⚡

🔷 Response Headers

├ 🌐 **Content-Type :** Specifies the MIME type of the data in the response body

├ 🌐 **Content-Length :** Specifies the length of the response body in bytes

├ 🌐 **Cache-Control :** Specifies the caching behavior of the response

├ 🌐 **Location :** Specifies the URI of a resource that can be used to retrieve the requested resource

├ 🌐 **Server :** Specifies the name and version of the server software that generated the response

├ 🌐 **Access-Control-Allow-Origin :** Specifies which origins are allowed to access the resource

├ 🌐 **Set-Cookie :** Specifies a cookie that should be stored by the client and sent back to the server with future requests

├ 🌐 **Expires :** Specifies the date and time after which the response is considered stale

└ 🌐 **Last-Modified :** Specifies the date and time the resource was last modified.

🔷 API Design

├ 💻 **REST :** Representational State Transfer a design pattern for building web services

├ 💻 **SOAP :** Simple Object Access Protocol, a messaging protocol for exchanging structured data

├ 💻 **GraphQL :** A query language and runtime for building APIs

└ 💻 **API Gateway :** A service that manages, protects, and scales APIs

⚡ API Quick Reference ⚡

🔷 API Architectures

├ 🏢 **SOA :** Service-Oriented Architecture, an architectural style for building distributed systems

├ 🏢 **Microservices :** An architectural style for building complex applications as a suite of small, independent services

├ 🏢 **Serverless :** A cloud computing execution model where the cloud provider manages the infrastructure and automatically allocates resources as needed

├ 🏢 **Event-Driven :** An architectural style where the flow of data between components is triggered by events

└ 🏢 **RESTful API :** An architectural style that uses HTTP requests to GET, POST, PUT, and DELETE data.

🔷 API Design Patterns

├ 🧩 **Adapter Pattern :** A pattern that converts the interface of a class into another interface that clients expect

├ 🧩 **Decorator Pattern :** A pattern that adds behavior to an individual object dynamically

├ 🧩 **Proxy Pattern :** A pattern that provides a surrogate or placeholder for another object to control access to it

├ 🧩 **Chain of Responsibility Pattern :** A pattern that delegates commands to a chain of processing objects

└ 🧩 **Observer Pattern :** A pattern that defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

⚡ API Quick Reference ⚡

🔷 API Security

├ 🔒 **OAuth :** An open standard for authorization used for protecting APIs

├ 🔒 **JWT :** JSON Web Tokens, a standard for securely transmitting information between parties as a JSON object

├ 🔒 **SSL/TLS :** Secure Sockets Layer/Transport Layer Security, a protocol for establishing a secure connection between a client and a server

├ 🔒 **API Key :** A secret token used to authenticate API requests

└ 🔒 **Rate Limiting :** A technique used to limit the number of requests that can be made to an API over a specific period of time

├ 🔒 **OpenID Connect :** An authentication layer built on top of OAuth that allows users to be authenticated across multiple domains

├ 🔒 **Cross-Origin Resource Sharing (CORS) :** A mechanism that allows many resources (e.g., fonts, JavaScript, etc.) on a web page to be requested from another domain outside the domain from which the resource originated

🔷 API Testing

├ 🧪 **Postman :** A popular tool for testing and debugging APIs

├ 🧪 **SoapUI :** A tool for testing SOAP and REST web services

├ 🧪 **Swagger :** A tool for designing, building, and testing APIs

├ 🧪 **JMeter :** A tool for testing the performance of APIs

└ 🧪 **TestRail :** A test management tool for planning, executing, and tracking API tests

├ 🧪 **Dredd :** A command-line tool for testing API documentation against its backend implementation

├ 🧪 **REST Assured :** A Java-based library for testing RESTful APIs

⚡ API Quick Reference ⚡

├ 🧪 **Karate DSL** : A testing framework for API testing using Gherkin syntax

├ 🧪 **HttpMaster** : A tool for testing and debugging APIs

└ 🧪 **Assertible** : A tool for testing and monitoring APIs with automated tests.

🔷 API Development

├ 🧑‍💻 **Node.js** : A JavaScript runtime for building server-side applications

├ 🧑‍💻 **Express** : A popular framework for building web applications and APIs with Node.js

├ 🧑‍💻 **Django** : A Python web framework for building web applications and APIs

├ 🧑‍💻 **Flask** : A lightweight Python web framework for building web applications and APIs

└ 🧑‍💻 **Spring** : A Java framework for building enterprise-level web applications and APIs

├ 🛠️ **Swagger Editor** : A tool for designing and documenting APIs using the OpenAPI specification

├ 🛠️ **Postman** : A tool for testing and debugging APIs

├ 🛠️ **Insomnia** : A tool for designing, testing, and debugging APIs

├ 🛠️ **Paw** : A tool for designing and testing APIs on Mac OS

└ 🛠️ **API Blueprint** : A high-level API description language for building RESTful APIs.

⚡ API Quick Reference ⚡

🔷 API Implementation Platforms

├ 🌐 **Firebase :** A mobile and web application development platform developed by Google

├ 🌐 **Backendless :** A mobile and web application development platform that allows developers to build and deploy applications without backend coding

├ 🌐 **Parse Server :** An open-source version of the Parse backend that can be deployed to any infrastructure

├ 🌐 **Amazon API Gateway :** A fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs

└ 🌐 **Microsoft Azure API Management :** A fully managed service that enables users to publish, secure, transform, maintain, and monitor APIs.

🔷 API Performance

├ 🚀 **Caching :** A technique for improving API performance by storing responses in a cache

├ 🚀 **Throttling :** A technique for limiting the rate of requests to an API to prevent overload

├ 🚀 **Load Balancing :** A technique for distributing traffic evenly across multiple servers to improve API performance

├ 🚀 **Content Delivery Network (CDN) :** A distributed system of servers that delivers content to users based on their geographic location to improve API performance

└ 🚀 **Edge Computing :** A computing paradigm that brings computation and data storage closer to the location where it is needed to reduce latency and improve API performance.

⚡ API Quick Reference ⚡

🔷 API Monitoring

├ 🕵️ **Pingdom :** A tool for monitoring the uptime and performance of APIs

├ 🕵️ **New Relic :** A tool for monitoring the performance of APIs and other web applications

├ 🕵️ **Datadog :** A monitoring and analytics platform for cloud-scale applications and APIs

├ 🕵️ **Sumo Logic :** A cloud-based log management and analytics platform for APIs and other applications

└ 🕵️ **Loggly :** A cloud-based log management platform for monitoring APIs and other applications

🔷 API Standards

├ 📏 **JSON API :** A specification for building APIs that use JSON as the data format

├ 📏 **HAL :** Hypertext Application Language, a standard for building hypermedia-driven APIs

├ 📏 **JSON-LD :** A format for representing linked data on the web

├ 📏 **OData :** Open Data Protocol, a standard for building and consuming RESTful APIs

└ 📏 **AsyncAPI :** A specification for building event-driven APIs.

⚡ API Quick Reference ⚡

🔷 API Standards Organizations

├ 🌎 **W3C :** The World Wide Web Consortium, an international community that develops web standards

├ 🌎 **IETF :** The Internet Engineering Task Force, an open standards organization that develops and promotes Internet standards

├ 🌎 **OASIS :** Organization for the Advancement of Structured Information Standards, a nonprofit consortium that drives the development, convergence, and adoption of open standards for the global information society

├ 🌎 **RESTful API Modeling Language (RAML) :** A YAML-based language for describing RESTful APIs developed by MuleSoft

└ 🌎 **JSON API :** A specification for building APIs that use JSON as the data format.

🔷 API Infrastructure

├ 📦 **Kubernetes :** An open-source platform for managing containerized workloads and services

├ 📦 **OpenShift :** A container application platform that builds on top of Kubernetes

├ 📦 **Docker Swarm :** A native clustering and orchestration solution for Docker

├ 📦 **Consul :** A service mesh solution that provides service discovery, configuration, and segmentation capabilities

└ 📦 **Istio :** A service mesh solution that provides traffic management, security, and observability capabilities.

⚡ API Quick Reference ⚡

🔷 API Governance

├ 📝 **API Management :** The process of creating, publishing, and monitoring APIs in a secure and scalable way

├ 📝 **API Monetization :** The process of generating revenue from APIs by charging developers for usage

├ 📝 **API Versioning :** The process of managing changes to APIs over time

├ 📝 **API Analytics :** The process of collecting and analyzing data on API usage and performance

└ 📝 **API Gateway :** A service that manages, protects, and scales APIs.

🔷 API Documentation

├ 📖 **OpenAPI :** A specification for building APIs in YAML or JSON format

├ 📖 **API Blueprint :** A high-level API description language for building RESTful APIs

├ 📖 **RAML :** A YAML-based language for describing RESTful APIs

├ 📖 **Swagger UI :** A tool for visualizing and interacting with APIs that have been described using the OpenAPI specification

└ 📖 **Slate :** A tool for generating beautiful, responsive API documentation.

🔷 API Deployment

├ 🚀 **Heroku :** A cloud platform for deploying, managing, and scaling web applications and APIs

├ 🚀 **AWS Elastic Beanstalk :** A service for deploying and scaling web applications and APIs on AWS

⚡ API Quick Reference ⚡

├ 🚀 **Azure App Service :** A service for deploying and scaling web applications and APIs on Azure

├ 🚀 **Google App Engine :** A service for deploying and scaling web applications and APIs on GCP

└ 🚀 **Docker :** A containerization platform used for packaging and deploying applications

├ 🚀 **AWS Lambda :** A serverless compute service for running code in response to events

├ 🚀 **Azure Functions :** A serverless compute service for running code in response to events

├ 🚀 **Google Cloud Functions :** A serverless compute service for running code in response to events

├ 🚀 **Netlify :** A cloud platform for deploying and managing static websites and APIs

└ 🚀 **Vercel :** A cloud platform for deploying and managing static websites and APIs

🔷 API Security

├ 🔒 **OAuth :** An open standard for authorization used by many social media platforms and APIs

├ 🔒 **OpenID Connect :** An authentication layer built on top of OAuth that allows users to be authenticated across multiple domains

├ 🔒 **JSON Web Tokens (JWT) :** A method for representing claims securely between two parties

⚡ API Quick Reference ⚡

├ 🔒 **Cross-Origin Resource Sharing (CORS) :** A mechanism that allows many resources (e.g., fonts, JavaScript, etc.) on a web page to be requested from another domain outside the domain from which the resource originated

└ 🔒 **API Keys :** A secret token that identifies an API client to the server and allows the client to access resources.

💠 API Best Practices

├ 📝 **Versioning :** A technique for managing changes to APIs over time

├ 📝 **Pagination :** A technique for breaking up large API responses into smaller, more manageable chunks

├ 📝 **Caching :** A technique for improving API performance by storing responses in a cache

├ 📝 **Error Handling :** A technique for returning meaningful error messages to API clients

└ 📝 **HATEOAS :** Hypermedia as the Engine of Application State, a constraint of RESTful APIs that requires the API to provide links to related resources

💠 API Tutorials

├ 📚 Getting Started with RESTful APIs by Tania Rascia

├ 📚 API Design Best Practices by Martin Fowler

├ 📚 Testing RESTful Web Services Made Easy Using the REST Assured Framework by Dinesh Rajput

├ 📚 API Gateway Concepts and Options by AWS

├ 📚 Building Secure APIs by Auth0

└ 📚 RESTful API Designing guidelines – The best practices by Mahesh Haldar

⚡ API Quick Reference ⚡

🔷 API Guides

├ 📖 REST API Tutorial by Guru99

├ 📖 A Beginner's Guide to HTTP and REST by Linode

├ 📖 REST API Design: Resource Modeling by Oracle

├ 📖 API Security Best Practices by Google Cloud

└ 📖 API Governance Handbook by WSO2.

🔷 API Tools

├ 🛠 **API Studio :** A web-based IDE for designing and testing APIs

├ 🛠 **Stoplight :** A collaborative platform for designing, documenting, and testing APIs

├ 🛠 **Apigee :** A full lifecycle API management platform that allows developers to design, secure, deploy, and analyze APIs

├ 🛠 **Azure API Management :** A fully managed service that enables users to publish, secure, transform, maintain, and monitor APIs

└ 🛠 **Postman Learning Center :** A hub for learning how to use Postman to design, develop, and test APIs.