A Function Or Service That Is Called From A Web Application To Another Web Application Is Called A(n)

A Function Or Service That Is Called From A Web Application To Another Web Application Is Called A(n) in the realm of modern software development, especially within the context of web technologies, is known as an API (Application Programming Interface). APIs are fundamental to enabling different web applications to interact seamlessly, share data, and perform complex operations across platforms. This article will explore the concept of APIs in detail, discussing their types, how they work, their importance in web development, and best practices for using them effectively.

Understanding the Concept of API in Web Applications

What Is an API?

An Application Programming Interface (API) is a set of rules, protocols, and tools that allow different software applications to communicate with each other. When one web application needs to access or utilize services, data, or functionalities provided by another web application, it does so through an API.

APIs act as a bridge between different software systems, defining methods and data formats that enable interaction. They abstract the underlying complexities of the system and expose only the necessary parts for external interaction. This abstraction simplifies the integration process and ensures a standardized way for applications to communicate.

Why Are APIs Important in Web Development?

APIs are crucial because they enable:
- Interoperability: Different web applications, often built with diverse technologies, can work together.
- Extensibility: Developers can extend the functionality of existing applications without altering core code.
- Efficiency: APIs streamline communication, reducing development time and effort.
- Security: Properly designed APIs restrict access and ensure data security.
- Innovation: APIs facilitate new features and integrations, fostering innovation.

Types of Web APIs

Web APIs come in various forms, each suited for different purposes and use cases.

1. RESTful APIs

Representational State Transfer (REST) is an architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE) for communication. RESTful APIs are stateless, cacheable, and designed for scalability.

Features of RESTful APIs:
- Use of standard HTTP methods.
- Data typically formatted in JSON or XML.
- Stateless interactions.
- Easy to consume and widely supported.

2. SOAP APIs

Simple Object Access Protocol (SOAP) APIs are protocol-based and rely on XML messaging. They offer higher security features and are suitable for enterprise-level applications.

Features of SOAP APIs:
- Use of XML for message formatting.
- Built-in error handling.
- Supports WS-Security for secure communications.
- More rigid and complex compared to REST.

3. GraphQL APIs

GraphQL is a query language for APIs that allows clients to request only the data they need, reducing over-fetching and under-fetching.

Features of GraphQL APIs:
- Single endpoint for all requests.
- Precise data retrieval.
- Strongly typed schema.
- Suitable for complex and dynamic data requirements.

4. WebSocket APIs

WebSockets enable real-time, bidirectional communication between client and server, often used in chat applications, live feeds, and gaming.

Features of WebSocket APIs:
- Persistent connection.
- Low latency data transfer.
- Suitable for real-time applications.

How APIs Work: An Overview

APIs facilitate communication through a sequence of well-defined steps:

1. **Request Initiation:** A web application (client) sends a request to another application's API endpoint, specifying the desired operation and data.
2. **Processing:** The API server processes the request, executes the required logic, and interacts with databases or other services if necessary.
3. **Response:** The server sends back a response containing the requested data or status information, often formatted in JSON or XML.
4. **Consumption:** The client application processes the response and updates the user interface or performs other actions accordingly.

This interaction is typically stateless, meaning each request contains all the information needed for processing, which enhances scalability and simplicity.

Common Use Cases for Web APIs

APIs are versatile and are used across various scenarios in web development:

- **Third-party integrations:** Connecting to external services like payment gateways, social media platforms, or mapping services.
- **Mobile app communication:** Backend APIs serve mobile applications, providing data and services.
- **Microservices architecture:** Breaking down applications into smaller, interconnected services communicating via APIs.
- **Data sharing and analysis:** APIs facilitate data collection from multiple sources for analytics and reporting.
- **Real-time updates:** Using WebSocket APIs for live notifications, chat, or streaming data.

Designing Effective Web APIs

Designing a robust API requires attention to several best practices:

1. Consistency and Standardization

Use consistent naming conventions, HTTP methods, and data formats to make APIs predictable and easy to use.

2. Security Measures

Implement authentication (e.g., OAuth), authorization, input validation, and encryption to protect data and prevent unauthorized access.

3. Versioning

Maintain version control to ensure backward compatibility as APIs evolve.

4. Documentation

Provide comprehensive documentation, including endpoints, request/response examples, error codes, and usage guidelines.

5. Error Handling

Design clear and informative error messages to help developers troubleshoot issues efficiently.

6. Scalability and Performance

Optimize APIs for speed and reliability, considering caching strategies and load balancing.

Real-World Examples of Web APIs

Understanding practical applications helps grasp the importance of APIs:

1. Social Media APIs

Platforms like Twitter, Facebook, and Instagram offer APIs to retrieve posts, publish content, and analyze engagement.

2. Payment Gateway APIs

Services like Stripe or PayPal provide APIs for secure online transactions.

3. Mapping and Geolocation APIs

Google Maps API allows web applications to embed maps, geocode addresses, and provide directions.

4. Cloud Service APIs

AWS, Azure, and Google Cloud offer APIs for managing cloud resources and services.

The Future of Web APIs

As technology advances, APIs continue to evolve, emphasizing:
- Increased security with OAuth 2.0 and OpenID Connect.
- More flexible data querying through GraphQL.
- Enhanced real-time communication via WebSockets and HTTP/2.
- Greater adoption of API gateways and management platforms for better control and analytics.
- Emphasis on API standardization and specifications like OpenAPI (Swagger).

Conclusion

A function or service that is called from a web application to another web application is called an API, a vital component in the interconnected world of modern web development. APIs enable seamless data exchange, integration of services, and expansion of functionalities across diverse platforms and devices. Understanding how they work, their types, and best practices for design and implementation is essential for developers aiming to build scalable, secure, and efficient web applications. Whether facilitating third-party integrations, powering mobile apps, or supporting real-time features, APIs are the backbone of dynamic, feature-rich web experiences.

Meta Description: Discover what a function or service called from a web application to another is called—APIs. Learn about types, how they work, and their importance in modern web development.

Frequently Asked Questions

What is the term for a function or service called from one web application to another?

It is called an API (Application Programming Interface).

Which term describes a web service that enables communication between two web applications?

An API (Application Programming Interface).

What is the common name for a service that allows web applications to interact and exchange data?

A web API or simply API.

In web development, what do you call a function that one web app uses to access data or services from another?

A remote procedure call (RPC) or API endpoint.

Which term is used for a standardized way for web applications to communicate with each other?

Web API.

What is the term for a service that provides an interface for communication between different web applications?

An API (Application Programming Interface).

When one web application calls a function in another web application, what is this called?

Making an API call.

What do you call the mechanism that allows web applications to invoke functions or services in external applications?

An API (Application Programming Interface).

Additional Resources

A Function Or Service That Is Called From A Web Application To Another Web Application Is Called A(n)

In the rapidly evolving landscape of web development, seamless integration between different applications has become paramount. Whether it's sharing data, leveraging third-party services, or orchestrating complex workflows, web applications often need to communicate with one another. This inter-application communication is facilitated through various mechanisms, each with its own terminology, protocols, and use cases. At the core of this interaction lies a fundamental concept: a function or service that is invoked from one web application to another. This article explores this concept in depth, shedding light on its definitions, mechanisms, significance, and best practices.

Understanding the Core Concept: What Is a Web Application Call?

At its most basic level, when a web application needs to access functionality or data hosted by another application, it makes a "call" or "request" to that application. This process involves initiating a communication over the internet, typically through HTTP (Hypertext Transfer Protocol) or HTTPS, to invoke a specific service or function hosted elsewhere.

The Nature of Inter-Application Calls

Inter-application calls are fundamental to distributed computing and web service architectures. They enable applications to:

- Share Data: Accessing shared resources or information stored elsewhere.
- Use External Services: Relying on third-party APIs like payment gateways, social media integrations, or mapping services.
- Modularize Functionality: Breaking complex systems into smaller, manageable services (microservices).
- Automate Workflows: Triggering processes across different systems to streamline operations.

Depending on the architecture and technology stack, these calls can take various forms, including REST API requests, SOAP messages, GraphQL queries, or even remote procedure calls (RPC).

Defining the Term: What Is the Formal Name?

When a web application calls a function or service hosted by another web application, it is generally referred to as an API call or API request. More specifically, the term used to describe such a call depends on the context and the architecture employed.

API (Application Programming Interface)

An API is a set of rules and protocols that define how software components should interact. It specifies the methods, data formats, and conventions used for communication.

Common Terminology

- API Call / API Request: The act of invoking an API endpoint exposed by another application.
- Web Service: A software system designed to support interoperable machine-to-machine interaction over a network.
- Remote Procedure Call (RPC): A protocol that one program can use to request a service from a

program located on another machine within a network.
- HTTP Request: A message sent by a client to initiate an action on a server, often used to invoke functions or retrieve data.

The Precise Term

Given the broad scope, the most comprehensive and accurate term for the concept is:

"Web Service Call" or more specifically, "API Call".

In formal documentation, this is often described as invoking a remote API or remote service.

How Do These Calls Work? The Underlying Mechanisms

Understanding the technical workings behind these calls provides clarity on their importance and implementation.

1. The Request-Response Model

Most web application calls follow a request-response pattern:

- Request: The calling application sends an HTTP request to the target application's API endpoint.
- Processing: The server hosting the API processes the request, executes the corresponding function or service.
- Response: The server sends back a response, often containing data, status codes, or confirmation messages.

2. Common Protocols and Data Formats

- HTTP/HTTPS: The standard protocol for web communications.
- REST (Representational State Transfer): An architectural style using standard HTTP methods (GET, POST, PUT, DELETE).
- SOAP (Simple Object Access Protocol): A protocol for exchanging structured information, often in XML.
- GraphQL: A query language allowing clients to specify exactly what data they need.

Data exchanged between applications is typically formatted in:

- JSON (JavaScript Object Notation): Widely used due to its lightweight nature.
- XML (eXtensible Markup Language): Common in SOAP services.

3. Authentication and Security

Secure inter-application calls implement mechanisms such as:

- API Keys: Unique identifiers for access control.
- OAuth: An open standard for token-based authorization.
- TLS/SSL: Encryption protocols to secure data in transit.

Types of Web Application Calls

Not all calls are created equal. Different types serve different purposes and have varying complexity.

1. Synchronous Calls

- The calling application waits for the response before proceeding.
- Suitable for real-time operations.
- Example: Fetching user data from an authentication service.

2. Asynchronous Calls

- The calling application proceeds immediately after initiating the request.
- Responses are handled via callbacks, promises, or message queues.
- Example: Sending data to a logging service without waiting for confirmation.

3. One-way Calls

- The calling application sends a request but does not expect a response.
- Used for fire-and-forget operations.
- Example: Triggering a background process.

Practical Examples and Use Cases

Understanding theoretical concepts is enriched by real-world examples where web application calls are indispensable.

Example 1: Payment Processing

An e-commerce web application needs to process payments securely. It calls an external payment gateway API to authorize and capture payments. This involves:

- Sending transaction details via an API request.
- Receiving a response indicating success or failure.
- Updating order status accordingly.

Example 2: Social Media Integration

A news website allows users to share articles on social media platforms like Facebook or Twitter. When a user clicks "Share," the site makes an API call to the respective social media platform's API to publish the content.

Example 3: Data Synchronization

A logistics company’s internal application calls a third-party tracking service API to fetch real-time shipment data, enabling their customer portal to display current statuses.

Example 4: Microservices Architecture

In a microservices setup, different services (user management, billing, notifications) communicate via API calls to perform complex workflows, enhancing modularity and scalability.

Best Practices for Implementing Inter-Application Calls

Ensuring that these calls are efficient, secure, and reliable is crucial for maintaining system integrity.

1. Use Standardized Protocols and Formats

- Prefer RESTful APIs with JSON, as they are widely supported and easy to use.
- Use SOAP only when necessary, especially in enterprise environments requiring strict standards.

2. Implement Robust Authentication and Authorization

- Use OAuth 2.0 for token-based access.
- Validate API keys and tokens rigorously.
- Enforce HTTPS to encrypt data in transit.

3. Handle Errors Gracefully

- Implement retries for transient failures.
- Use appropriate HTTP status codes to indicate issues.
- Log errors for troubleshooting.

4. Optimize for Performance

- Use caching where appropriate.
- Minimize payload sizes.
- Limit the number of API calls to prevent rate limiting.

5. Document APIs Clearly

- Provide comprehensive API documentation.
- Use standards like OpenAPI/Swagger for clarity.
- Include versioning to manage updates.

Future Trends and Evolving Technologies

The landscape of inter-application communication continues to evolve, influenced by emerging technologies.

1. GraphQL and Beyond

- GraphQL allows clients to request exactly the data they need, reducing over-fetching.

- Increasing adoption in modern web applications.

2. API Gateways and Management Platforms

- Centralized platforms for managing, monitoring, and securing APIs.
- Facilitate scalability and governance.

3. Event-Driven Architectures

- Shift from request-response to event-driven models using message queues and pub/sub systems.
- Enhance decoupling and scalability.

4. Serverless and Function-as-a-Service (FaaS)

- Call serverless functions exposed via HTTP endpoints.
- Simplify deployment and scaling.

Conclusion

A function or service that is called from a web application to another web application is more than just a technical detail; it's a fundamental building block of modern distributed systems. Whether referred to as an API call, web service invocation, or remote procedure call, these interactions enable web applications to be more modular, scalable, and feature-rich. As web technologies advance, understanding the mechanisms, best practices, and emerging trends in inter-application communication will remain essential for developers and organizations aiming to build resilient and innovative digital solutions. Through secure, efficient, and well-documented API integrations, businesses can unlock new levels of functionality and user engagement in the interconnected web ecosystem.

A Function Or Service That Is Called From A Web Application To Another Web Application Is Called A N

A Function Or Service That Is Called From A Web Application To Another Web Application Is Called A N

Related Articles

- An Analysis Of Domestic Returns For The U.S. Bond Markets Ranks Fourth Out Of Six Countries. When The
- Arranging Tasks, People, And Other Resources To Accomplish The Work Is Known As The ____ Function Of
- An Economist Wants To Determine Whether Average Price/earnings (P/E) Ratios Differ For Firms In Three

Back to Home