developers

- Go to your profile
- Hire a developer
- Apply as a developer
- Log in

-
- Top 3%
- Why
- Clients
- Enterprise
- Community
- Blog
- About Us

- Go to your profile
- Hire a developer
- Apply as a developer
- Log in
-

- Questions?
- Contact Us
-
-
-

- Questions?
- Contact Us
-
-
-

Hire a developer
Find a world-class web developer for your team.Hire Toptal's web developers

10 Essential Web Interview Questions *

- 1.6Kshares
-
-
-

Submit an interview questionSubmit a question
Looking for experts? Check out Toptal's web developers.

What is CORS? How does it work?

View the answer →Hide answer

Cross-origin resource sharing (CORS) is 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. It's a mechanism supported in HTML5 that manages `XMLHttpRequest` access to a domain different.

CORS adds new HTTP headers that provide access to permitted origin domains. For HTTP methods other than GET (or POST with certain MIME types), the spe
mandates that browsers first use an HTTP OPTIONS request header to solicit a list of supported (and available) methods from the server. The actual request can t
submitted. Servers can also notify clients whether "credentials" (including Cookies and HTTP Authentication data) should be sent with requests.

Privacy - Terms

Explain the purpose of each of the HTTP request types when used with a RESTful web service.

View the answer →Hide answer

The purpose of each of the HTTP request types when used with a RESTful web service is as follows:

1. **GET:** Retrieves data from the server (should only retrieve data and should have no other effect).
2. **POST:** Sends data to the server for a new entity. It is often used when uploading a file or submitting a completed web form.
3. **PUT:** Similar to POST, but used to replace an existing entity.
4. **PATCH:** Similar to PUT, but used to update only certain fields within an existing entity.
5. **DELETE:** Removes data from the server.
6. **TRACE:** Provides a means to test what a machine along the network path receives when a request is made. As such, it simply returns what was sent.
7. **OPTIONS:** Allows a client to request information about the request methods supported by a service. The relevant response header is Allow and it simply lists the supported methods. (It can also be used to request information about the request methods supported for the server where the service resides by using a * wildcard in the URI.)
8. **HEAD:** Same as the GET method for a resource, but returns only the response headers (i.e., with no entity-body).
9. **CONNECT:** Primarily used to establish a network connection to a resource (usually via some proxy that can be requested to forward an HTTP request as TCP and maintain the connection). Once established, the response sends a 200 status code and a "Connection Established" message.

Explain the basic structure of a MIME multipart message when used to transfer different content type parts. Provide a simple example.

View the answer →Hide answer

A simple example of a MIME multipart message is as follows:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=frontier

This is a message with multiple parts in MIME format.
--frontier
Content-Type: text/plain

This is the body of the message.
--frontier
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw+Cg==
--frontier--
```

Each MIME message starts with a message header. This header contains information about the message content and boundary. In this case `Content-Type: multipart/mixed; boundary=frontier` means that message contains multiple parts where each part is of different content type and they are separated by `--frontier` as their boundary.

Each part consists of its own content header (zero or more `Content-` header fields) and a body. Multipart content can be nested. The `content-transfer-encoding` of a multipart type must always be `7bit`, `8bit`, or `binary` to avoid the complications that would be posed by multiple levels of decoding. The multipart block as a whole does not have a charset; non-ASCII characters in the part headers are handled by the `Encoded-Word` system, and the part bodies can have charsets specified if appropriate for their `content-type`.

Find top web developers today. Toptal can match you with the best engineers to finish your project.

Hire Toptal’s web developers

What is **Long polling**, how does it work, and why would you use it? Considering server and client resources, what is the main drawback of using long polling? Which HTML5 feature is the best alternative to long polling?

View the answer →Hide answer

The HTTP protocol is based on a request/response pattern, which means that the server cannot *push* any data to the client (i.e., the server can only provide data to the client in response to a client request). **Long polling** is a web application development pattern used to *emulate* pushing data from server to client. When the long polling pattern is used, the client submits a request to the server and *the connection then remains active until the server is ready to send data to the client*. The connection is closed only after data is sent back to the client or connection timeout occurs. The client then creates a new request when the connection is closed, thus restarting the loop.

There are two important drawbacks that need to be considered when using long polling:

1. Long polling requests are not different from any other HTTP request and web servers handle them the same way. This means that every long poll connection will reserve server resources, potentially maxing out the number of connections the server can handle. This can lead to HTTP connection timeouts.

2. Each web browser will limit the maximum number of connections web application can make. This means that your application load time and performance may be degraded.

In HTML5, a useful alternative to long polling is using a **WebSocket**. A WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. The WebSocket protocol makes possible more interaction between a browser and a web site, facilitating live content and eliminates the need for the long polling paradigm.

Another potential answer could be **Server-sent DOM Events**. Which is method of continuously sending data from a server to the browser, rather than repeatedly requesting it. However, this HTML5 feature is not supported by Microsoft Internet Explorer, thus making it less attractive solution.

Consider the following JavaScript code that is executed in a browser:

```
function startAjaxQueue(){
  for (var i = 0; i < 50; i++){
        executeAjaxCallAsync();
  }
};
```

Assuming that `executeAjaxCallAsync()` uses a standard `XmlHttpRequest` asynchronously to retrieve data from server, how many **concurrent** HTTP requests would you expect to be created by this loop?

View the answer →Hide answer

Number of concurrent HTTP requests and `XmlHttpRequest` is limited in all browsers. Specific limitations are different depending on browser type and version.

For example, according to Mozilla Developer Network Firefox 3 limits the number of XMLHttpRequest connections per server to 6 (previous versions limit this to 2 per server).

Having this mind, the number of concurrent HTTP requests created in this loop would never (by default) be larger than 6, and the browser would therefore execute this loop in chunks.

What is an ETag and how does it work?

View the answer →Hide answer

An ETag is an opaque identifier assigned by a web server to a specific version of a resource found at an URL. If the resource content at that URL ever changes, a new and different ETag is assigned.

In typical usage, when an URL is retrieved the web server will return the resource along with its corresponding ETag value, which is placed in an HTTP “ETag” field:

```
ETag: "unique_id_of_resource_version"
```

The client may then decide to cache the resource, along with its `ETag`. Later, if the client wants to retrieve the same URL again, it will send its previously saved copy of the ETag along with the request in a `"If-None-Match"` field.

```
If-None-Match: "unique_id_of_resource_version"
```

On this subsequent request, the server may now compare the client’s `ETag` with the `ETag` for the current version of the resource. If the `ETag` values match, meaning that the resource has not changed, then the server may send back a very short response with a `HTTP 304 Not Modified` status. The 304 status tells the client that its cached version is still good and that it should use that.

However, if the `ETag` values do not match, meaning the resource has likely changed, then a full response including the resource’s content is returned, just as if `ETag` were not being used. In this case the client may decide to replace its previously cached version with the newly returned resource and the new `ETag`.

Explain the difference between stateless and stateful protocols. Which type of protocol is HTTP? Explain your answer.

View the answer →Hide answer

A **stateless** communications protocol treats each request as an independent transaction. It therefore does not require the server to retain any session, identity, or status information spanning multiple requests from the same source. Similarly, the requestor can not rely on any such information being retained by the responder.

In contrast, a **stateful** communications protocol is one in which the responder maintains “state” information (session data, identity, status, etc.) across multiple requests from the same source.

HTTP is a stateless protocol. HTTP does not require server to retain information or status about each user for the duration of multiple requests.

Some web servers implement states using different methods (using cookies, custom headers, hidden form fields etc.). However, in the very core of every web application everything relies on HTTP which is still a stateless protocol that is based on simple request/response paradigm.

Describe the key advantages of HTTP/2 as compared with HTTP 1.1.

View the answer →Hide answer

HTTP/2 provides decreased latency to improve page load speed by supporting:

- Data compression of HTTP headers
- Server push technologies
- Loading of page elements in parallel over a single TCP connection
- Prioritization of requests

An important operational benefit of HTTP/2 is that it avoids the head-of-line blocking problem in HTTP 1.

What is a “MIME type”, what does it consist of, and what is it used for? Provide an example.

View the answer →Hide answer

MIME is an acronym for **M**ulti-purpose **I**nternet **M**ail **E**xtensions. It is used as a standard way of classifying file types over the Internet.

Web servers and browsers have a defined list of MIME types, which facilitates transfer of files of a known type, irrespective of operating system or browser.

A MIME type actually has two parts: a *type* and a *subtype* that are separated by a slash (/). For example, the MIME type for Microsoft Word files is `application/msword` (i.e., type is `application` and the subtype is `msword`).

What’s the difference between GET and POST?

View the answer →Hide answer

Both are methods used in HTTP requests. Generally it is said that GET is to download data and PUT is to upload data. But we can do both downloading as well as uploading either by GET/POST.

GET:

1. If we are sending parameters in a GET request to the server, then those parameters will be visible in the URL, because in GET, parameters are append to the URL. So there’s a lack of security while uploading to the server.
2. We can only send a limited amount of data in a GET request, because the URL has its max limit and we can not append a long data string to the URL.

POST:

1. If we are using POST then we are sending parameters in the body section of a request. If we send data after using encryption in the body of an http request, it’s quite a bit more secure.
2. We can send a lot more data using POST.

Note: GET is faster in the case of just getting data using a static API call in cases where we don’t have to pass any parameters.

* There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every “A” candidate worth hiring will be able to answer them all, nor does answering them all guarantee an “A” candidate. At the end of the day, hiring remains an art, a science — and a lot of work.
Submit an interview question
Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name
Email
Enter your question here
Enter your answer here
All fields are required
☐ I agree with the Terms and Conditions of Toptal, LLC's Privacy Policy
Submit a Question

Thanks for submitting your question.
Our editorial staff will review it shortly. Please note that submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.
Looking for Web experts? Check out Toptal’s web developers.
View full profile »
Darin Erat Sleiter
Brazil
Darin is a data scientist and engineer with a PhD in physics from Stanford. He's passionate about data and machine learning and has worked on data science projects across numerous industries and applications. Darin's co-founded an AI company and led a team of data scientists to build a product which uses machine learning and optimization techniques to reduce energy consumption in data centers. He's eagerly waiting for quantum computers.
Web DeveloperPythonC#SQL+9 more
Hire Darin
View full profile »
Josh Smith
United States
Josh is a freelance full stack developer, from graphic design on down to systems administration. He has founded two companies and led product development and engineering work at two others. He loves data-driven design, continuous deployment, and customer development. He fully believes in applying the scientific method to everything he does.
Web DeveloperHTML5CSS3jQuery+9 more
Hire Josh
View full profile »
Richard Rozsa
Netherlands
Richard Rozsa offers a vision of data as a self formatting entity. For more than 30 years, he's delivered top quality technical architecture, programming, testing and solutions for complex problems--on-time and within budget. He's extremely flexible and able to integrate as a standalone freelancer or within teams.
Web Developer.NETASP.NET+10 more
Hire Richard
Toptal connects the top 3% of freelance talent all over the world.

Join the Toptal community.

Hire a developer
or
Apply as a developer

Highest In-Demand Talent

- iOS Developers
- Front-End Developers
- UX Designers
- UI Designers
- Financial Modeling Consultants
- Interim CFOs
- Digital Project Managers

About

- Top 3%
- Clients
- Freelance Developers
- Freelance Designers
- Freelance Finance Experts
- Freelance Project Managers
- Freelance Product Managers
- About Us

Contact

- Contact Us
- Press Center
- Careers
- FAQ

Social

Privacy - Terms

- Facebook
- Twitter
- Instagram
- LinkedIn

Hire the top 3% of freelance talent™

- Privacy Policy
- Website Terms

Privacy - Terms