Content-Type header

- Syntax
- Directives
- Examples
  - Serving assets with correct content type
  - Content-Type in multipart forms
  - Content-Type in URL-encoded form submission
  - Content-Type in a REST API using JSON
- Specifications
- Browser compatibility
- See also

The HTTP **`Content-Type`** representation header is used to indicate the original media type of a resource before any content encoding is applied.

In responses, the `Content-Type` header informs the client about the media type of the returned data. In requests such as POST or PUT, the client uses the `Content-Type` header to specify the type of content being sent to the server. If a server implementation or configuration is strict about content type handling, a 415 client error response may be returned.

The `Content-Type` header differs from Content-Encoding in that `Content-Encoding` helps the recipient understand how to decode data to its original form.

> **NOTE:** This value may be ignored if browsers perform MIME sniffing (or content sniffing) on responses. To prevent browsers from using MIME sniffing, set the X-Content-Type-Options header value to `nosniff`. See MIME type verification for more details.

Header type	[Representation header](https://developer.mozilla.org/en-US/docs/Glossary/Representation%20header)

[Forbidden request header](https://developer.mozilla.org/en-US/docs/Glossary/Forbidden%20request%20header)	No
[CORS-safelisted response header](https://developer.mozilla.org/en-US/docs/Glossary/CORS-safelisted%20response%20header)	Yes
[CORS-safelisted request header](https://developer.mozilla.org/en-US/docs/Glossary/CORS-safelisted%20request%20header)	Yes*

* Values can't contain a CORS-unsafe request header byte: `"():<>?@[\]{},` Delete `0x7F`, and control characters `0x00` to `0x19` except for Tab `0x09`. It also needs to have a media type of its parsed value (ignoring parameters) of either `application/x-www-form-urlencoded`, `multipart/form-data`, or `text/plain`.

Syntax

```
Content-Type: <media-type>
```

For example:

```
Content-Type: text/html; charset=utf-8
Content-Type: multipart/form-data; boundary=ExampleBoundaryString
```

Directives

- `<media-type>`
  - : The media type of the resource or data. May contain the following parameters:
    - **`charset`**: Indicates the character encoding standard used. The value is case insensitive but lowercase is preferred.
    - **`boundary`**: For multipart entities, the `boundary` parameter is required. It is used to demarcate the boundaries of the multiple parts of the message. The value consists of 1 to 70 characters (not ending with white space) known to be robust in the context of different systems (e.g., email gateways). Often, the header boundary is prepended by two dashes in the request body, and the final boundary has two dashes appended at the end.

Examples

Serving assets with correct content type

In the following two example responses, JavaScript and CSS assets are served using `text/javascript` for JavaScript and `text/css` for CSS. The correct content type for these resources helps the browser handle them more securely and with better performance. See Properly configuring server MIME types for more information.

```
HTTP/1.1 200
content-encoding: br
content-type: text/javascript; charset=utf-8
vary: Accept-Encoding
date: Fri, 21 Jun 2024 14:02:25 GMT
content-length: 2978

const videoPlayer=document.getElementById...
```

```
HTTP/3 200
server: nginx
date: Wed, 24 Jul 2024 16:53:02 GMT
content-type: text/css
vary: Accept-Encoding
content-encoding: br

.super-container{clear:both;max-width:100%}...
```

`Content-Type` in multipart forms

In a POST request resulting from an HTML form submission, the `Content-Type` of the request is specified by the `enctype` attribute on the form element.

```
<form action="/foo" method="post" enctype="multipart/form-data">
  <input type="text" name="description" value="Description input value" />
  <input type="file" name="myFile" />
  <button type="submit">Submit</button>
```

```
</form>
```

The request looks something like the following example with some headers omitted for brevity. In the request, a boundary of `ExampleBoundaryString` is used for illustration, but in practice, a browser would create a string more like this `--------------------------100336341311965159528948576 5`.

```
POST /foo HTTP/1.1
Content-Length: 68137
Content-Type: multipart/form-data; boundary=ExampleBoundaryString

--ExampleBoundaryString
Content-Disposition: form-data; name="description"

Description input value
--ExampleBoundaryString
Content-Disposition: form-data; name="myFile"; filename="foo.txt"
Content-Type: text/plain

[content of the file foo.txt chosen by the user]
--ExampleBoundaryString--
```

`Content-Type` in URL-encoded form submission

When forms don't involve file uploads and are using simpler fields, URL-encoded forms may be more convenient where the form data is included in the request body:

```
<form action="/submit" method="post">
  <label for="comment">Comment:</label>
  <input type="text" id="comment" name="comment" value="Hello!" />
  <button type="submit">Submit</button>
</form>
```

```
POST /submit HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 16
```

```
comment=Hello%21
```

`Content-Type` in a REST API using JSON

Many REST APIs use `application/json` as a content type which is convenient for machine-to-machine communication or programmatic interaction. The following example shows a 201 Created response showing the result of a successful request:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "message": "New user created",
  "user": {
    "id": 123,
    "firstName": "Paul",
    "lastName": "Klee",
    "email": "p.klee@example.com"
  }
}
```

Specifications

Browser compatibility

See also

- Accept, Accept-Encoding, Accept-Language headers
- Vary
- Content-Encoding
- Content-Disposition
- 206 Partial Content
- X-Content-Type-Options

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:44 UTC