** `http` package has been deprecated. Please use the `fetch` package instead. **

`HTTP` provides an HTTP request API on the client and server. To use these functions, add the HTTP package to your project by running in your terminal:

```
meteor add http
```

{% apibox "HTTP.call" %}

This function initiates an HTTP request to a remote server.

On the server, this function can be run either synchronously or asynchronously. If the callback is omitted, it runs synchronously and the results are returned once the request completes successfully. If the request was not successful, an error is thrown. This is useful when making server-to-server HTTP API calls from within Meteor methods, as the method can succeed or fail based on the results of the synchronous HTTP call. In this case, consider using `this.unblock()` to allow other methods on the same connection to run in the mean time.

On the client, this function must be used asynchronously by passing a callback. Note that some browsers first send an `OPTIONS` request before sending your request (in order to determine CORS headers).

Both HTTP and HTTPS protocols are supported. The `url` argument must be an absolute URL including protocol and host name on the server, but may be relative to the current host on the client. The `query` option replaces the query string of `url` . Parameters specified in `params` that are put in the URL are appended to any query string. For example, with a `url` of `'/path?query'` and `params` of `{ foo: 'bar' }` , the final URL will be `'/path?query&foo=bar'` .

The `params` are put in the URL or the request body, depending on the type of request. In the case of request with no bodies, like GET and HEAD, the parameters will always go in the URL. For a POST or other type of request, the parameters will be encoded into the body with a standard `x-www-form-urlencoded` content type, unless the `content` or `data` option is used to specify a body, in which case the parameters will be appended to the URL instead.

When run in asynchronous mode, the callback receives two arguments, `error` and `result` . The `error` argument will contain an Error if the request fails in any way, including a network error, time-out, or an HTTP status code in the 400 or 500 range. In case of a 4xx/5xx HTTP status code, the `response` property on `error` matches the contents of the result object. When run in synchronous mode, either `result` is returned from the function, or `error` is thrown.

Contents of the result object:

statusCode Number
: Numeric HTTP result status code, or `null` on error.

content String
: The body of the HTTP response as a string.

data Object or `null`
: If the response headers indicate JSON content, this contains the body of the document parsed as a JSON object.

headers Object
: A dictionary of HTTP headers from the response.

Example server method:

```
Meteor.methods({
  checkTwitter(userId) {
    check(userId, String);
    this.unblock();

    try {
      const result = HTTP.call('GET', 'http://api.twitter.com/xyz', {
        params: { user: userId }
      });

      return true;
    } catch (e) {
      // Got a network error, timeout, or HTTP error in the 400 or 500 range.
      return false;
    }
  }
});
```

Example asynchronous HTTP call:

```
HTTP.call('POST', 'http://api.twitter.com/xyz', {
  data: { some: 'json', stuff: 1 }
}, (error, result) => {
  if (!error) {
    Session.set('twizzled', true);
  }
});
```

{% apibox "HTTP.get" %} {% apibox "HTTP.post" %} {% apibox "HTTP.put" %} {% apibox "HTTP.del" %}