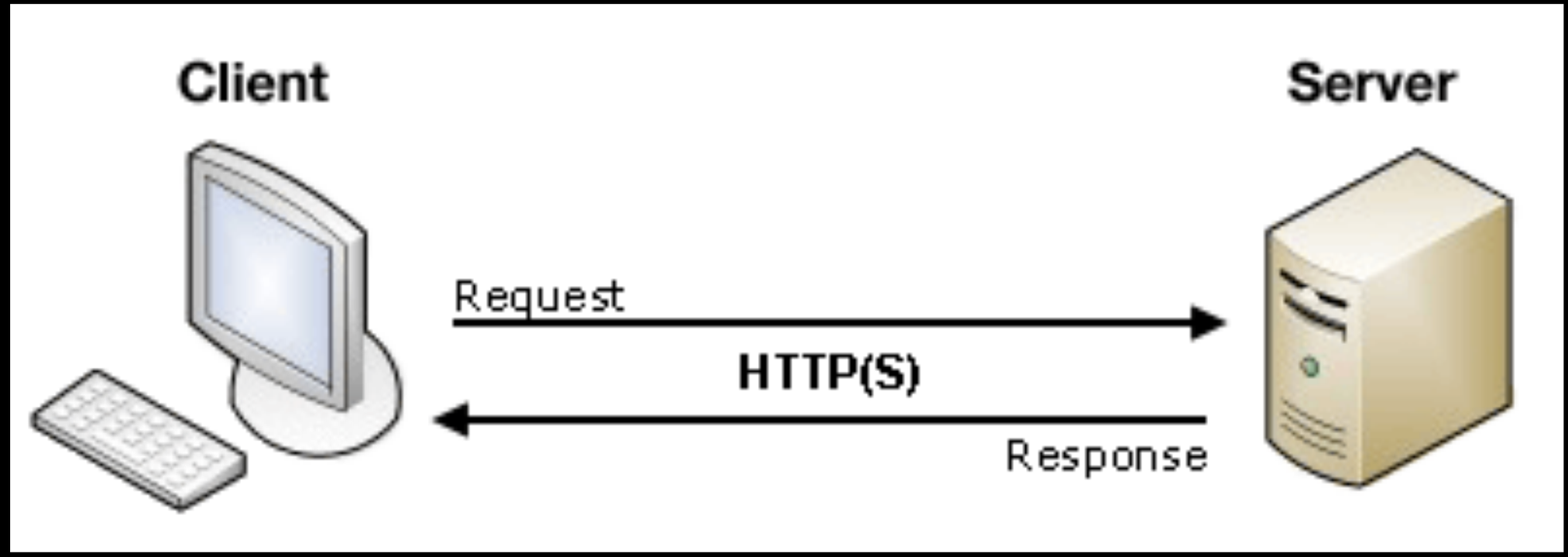
html forms

The web is based on a very basic client/server architecture that can be summarized as follows: a client (usually a Web browser) sends a request to a server (most of the time a web server like Apache, Nginx, IIS, Tomcat, etc.), using the HTTP protocol. The server answers the request using the same protocol.



```
<form>
 Your Name: <input type="text" name="theirText"></input>
     <input type="submit" name="theSubmitButton" value="Submit" />
    <input type="hidden" name="inVisible" value="..." />
    <br>
    <br>
    <input type="button" name="theButton" value="click!" id="thisAwesomeButton" />
</form>
```

On the client side, an HTML form is nothing more than a convenient user-friendly way to configure an HTTP request to send data to a server. This enables the user to provide information to be delivered in the HTTP request.

input // html forms

HTML Forms are one of the main points of interaction between a user and a web site or application. They allow users to send data to the web site. Most of the time that data is sent to the web server, but the web page can also intercept it to use it on its own.

An HTML Form is made of one or more widgets. Those widgets can be text fields (single line or multiline), select boxes, buttons, checkboxes, or radio buttons. Most of the time those widgets are paired with a label that describes their purpose — properly implemented labels are able to clearly instruct both sighted and blind users on what to enter into a form input.

```
<form method="GET" >
 Your Name: <input type="text" name="theirText"></input>
     <input type="submit" name="theSubmitButton" value="Submit" />
    <input type="hidden" name="inVisible" value="..." />
    <br>
    <br>
    <input type="button" name="theButton" value="click!" id="thisAwesomeButton" />
</form>
```

The main difference between a HTML form and a regular HTML document is that most of the time, the data collected by the form is sent to a web server.

my details

LOG-IN / REGISTER

You don't have to be registered to shop with Debenhams, but if you do you'll find that it makes sho
saved securely in My Details for you to amend at any time.

If you are already registered simply enter your *Log-in ID email address and password in the boxe
note that your password is case sensitive.

*Your Log-in ID email is the email address that you registered with.

Log-in (if you're already registered)

Email address:

*

Password:

*

Forgotten your password?

LOG-IN

Register (for new customers)

Email address:

*

Confirm your email address:

*

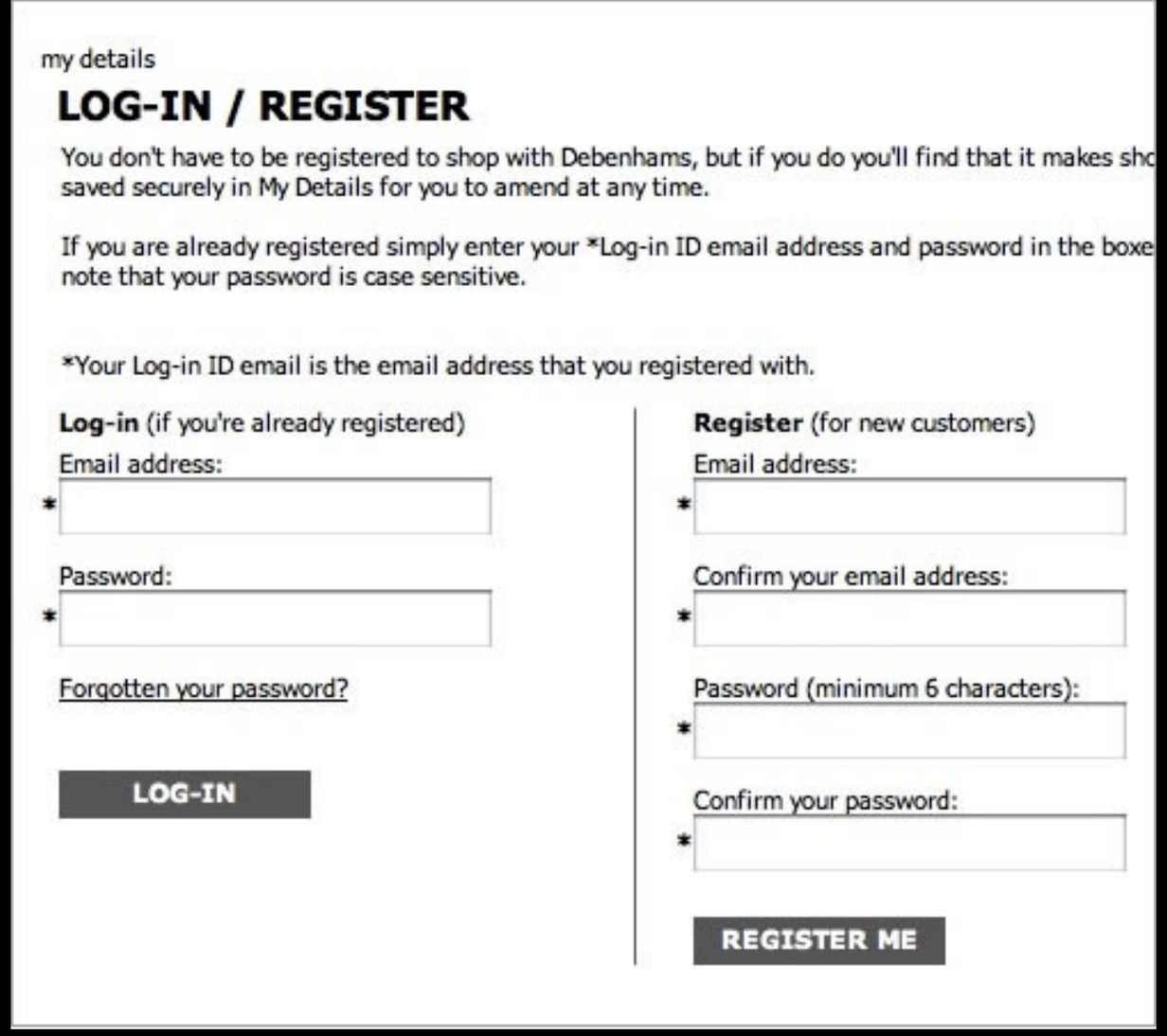
Password (minimum 6 characters):

*

Confirm your password:

*

REGISTER ME



Before starting to code, it's always better to step back and take the time to think about your form. Designing a quick mockup will help you to define the right set of data you want to ask your user. From a user experience (UX) point of view, it's important to remember that the bigger your form, the more you risk losing users. Keep it simple and stay focused: ask only for that data you absolutely need.

action attribute

This attribute defines where the data gets sent. Its value must be a valid URL. If this attribute isn't provided, the data will be sent to the URL of the page containing the form.

In this example, the data is sent to an absolute URL — `http://foo.com`:

```
<form action="http://foo.com">
```

Here, we use a relative URL — the data is sent to a different URL on the server:

```
<form action="/somewhere_else">
```

method attribute

This attribute defines how data is sent. The HTTP protocol provides several ways to perform a request; HTML form data can be transmitted via a number of different ones, the most common of which are the **GET** method and the **POST** method.

To understand the difference between those two methods, let's step back and examine how HTTP works: Each time you want to reach a resource on the Web, the browser sends a request to a URL.

An HTTP request consists of two parts:

a header that contains a set of global metadata about the browser's capabilities,

a body that can contain information necessary for the server to process the specific request.

the GET method

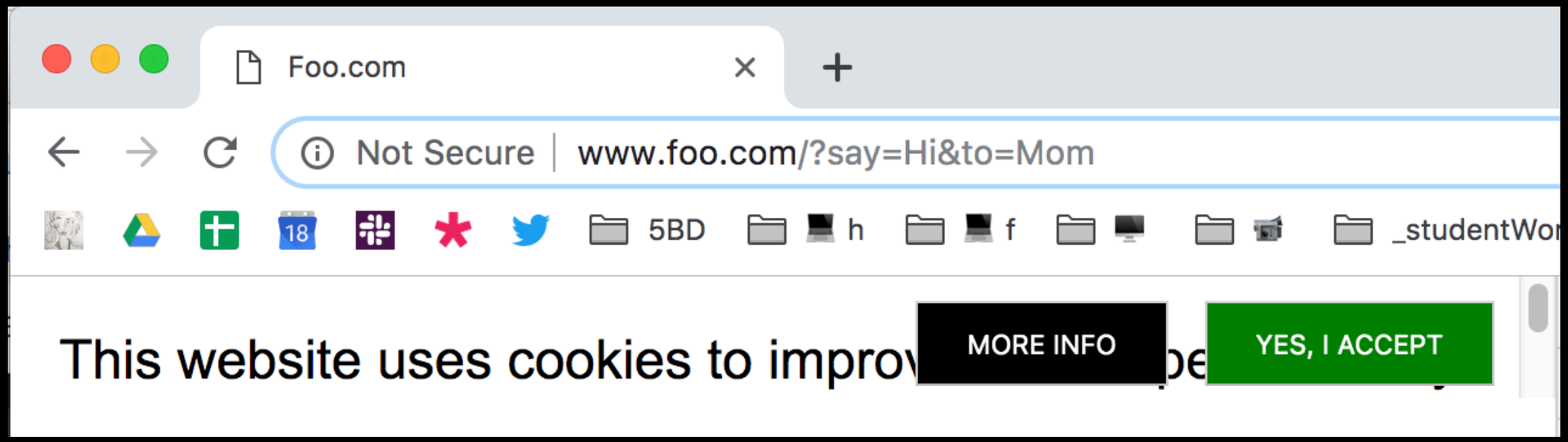
The GET method is the method used by the browser to ask the server to send back a given resource: "Hey server, I want to get this resource." In this case, the browser sends an empty body. Because the body is empty, if a form is sent using this method the data sent to the server is appended to the URL.

```
<form action="http://foo.com" method="get">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

the GET method

```
<form action="http://foo.com" method="get">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

Since the GET method has been used, you'll see the URL www.foo.com/?say=Hi&to=Mom appear in the browser address bar when you submit the form.



The data is appended to the URL as a series of name/value pairs. After the URL web address has ended, we include a question mark (?) followed by the name/value pairs, each one separated by an ampersand (&). In this case we are passing two pieces of data to the server:

say, which has a value of **Hi**
to, which has a value of **Mom**

the HTTP request looks like this:

```
GET /?say=Hi&to=Mom HTTP/2.0
Host: foo.com
```

the POST method

The `POST` method is a little different. It's the method the browser uses to talk to the server when asking for a response that takes into account the data provided in the body of the HTTP request: "Hey server, take a look at this data and send me back an appropriate result." If a form is sent using this method, the data is appended to the body of the HTTP request.

```
<form action="http://foo.com" method="post">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

the POST method

When the form is submitted using the POST method, you get no data appended to the URL, and the HTTP request looks like so, with the data included in the request body instead:

```
POST / HTTP/2.0
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

The Content-Length header indicates the size of the body, and the Content-Typeheader indicates the type of resource sent to the server.

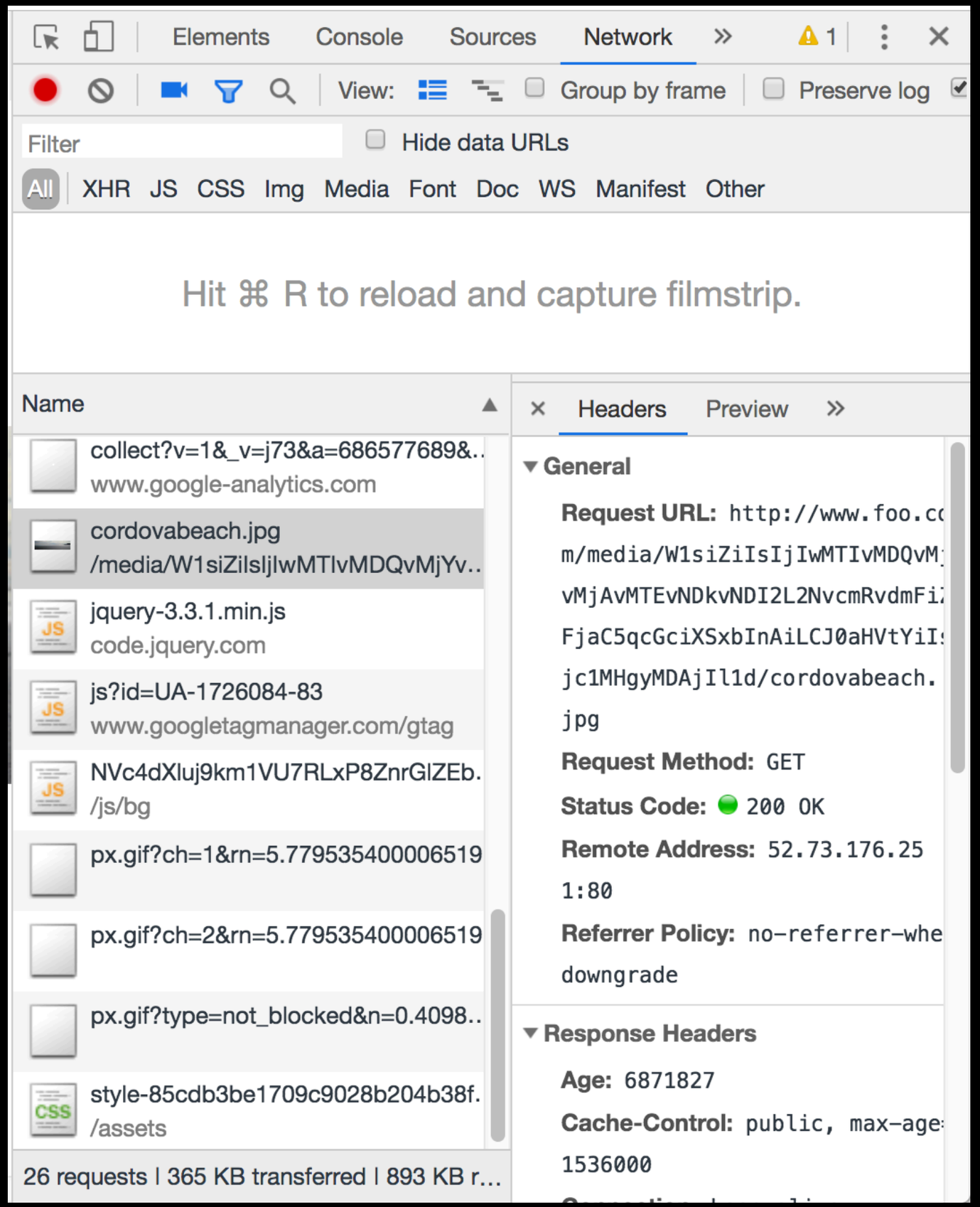
viewing HTTP requests

HTTP requests are never displayed to the user (if you want to see them, you need to use tools. As an example, your form data will be shown as follows in the Chrome Network tab. After submitting the form:

1. Press F12
2. Select "Network"
3. Select "All"
4. Select "foo.com" in the "Name" tab
5. Select "Headers"

viewing HTTP requests

You can then get the form data, as shown in the image.



viewing HTTP requests

The only thing displayed to the user is the URL called. As we mentioned above, with a **GET** request the user will see the data in their URL bar, but with a **POST** request they won't. This can be very important for two reasons:

1. If you need to send a password (or any other sensitive piece of data), never use the **GET** method or you risk displaying it in the URL bar, which would be very insecure.

2. If you need to send a large amount of data, the **POST** method is preferred because some browsers limit the sizes of URLs. In addition, many servers limit the length of URLs they accept.

mdn example in js w/ express (!!)

```html
<form action="/sayMom" method="get">
  <div>
    <label for="say">What greeting do you want to say?</
    label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

```
theApp.get('/sayMom', function(theRequest, theResponse){
    theResponse.send('Hi Mom!');
});
```

mdn example in js w/ express (!!)