PRESERVING STATE WITH QUERY STRINGS

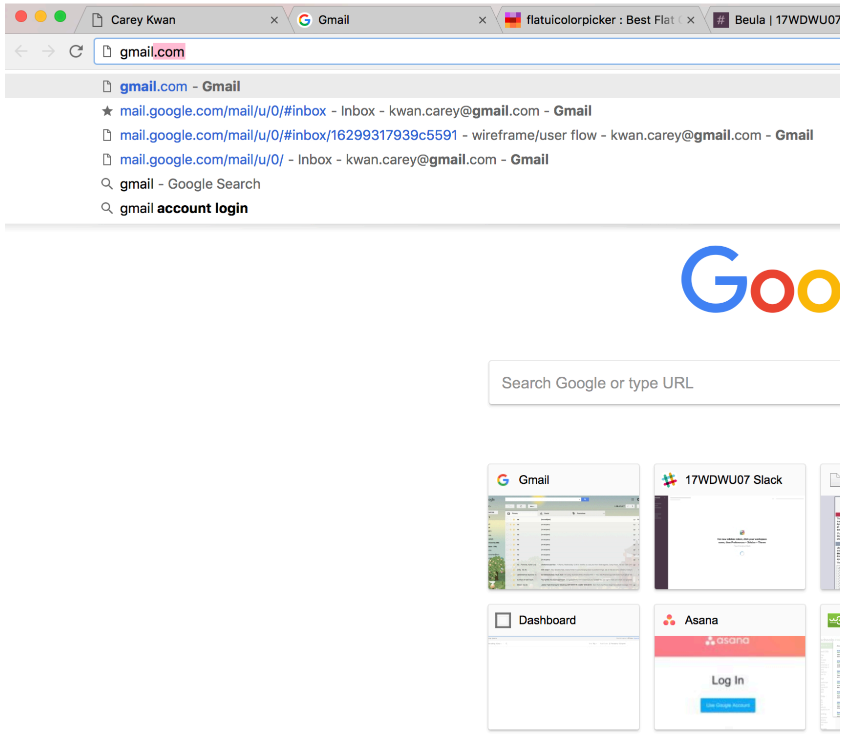
What is State?

State – in the context of web development – means that the same URL and query string may return a different web page, depending on the state of the server.

For example when I log into my Gmail account from a computer I have used before, I only have to do it once; I don't have to log in over and over again every time I change pages. Between requests, Gmail 'remembers' me, without my changing the URL.

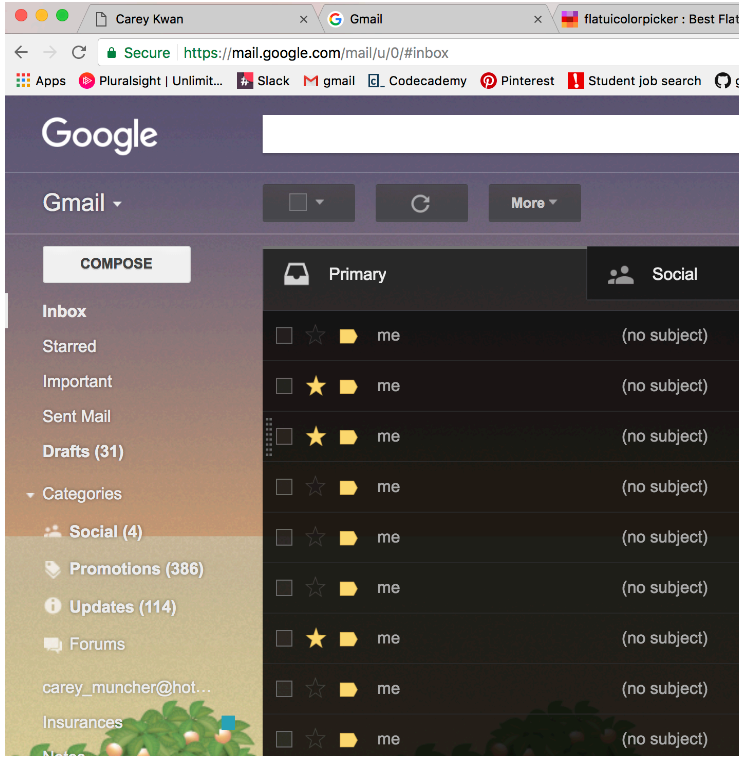
Likewise, when I browse to the page, sometimes I've received a new email and the page is different. Likewise, when I delete or move an email from my inbox, the next time I return, it hasn't moved back to my inbox. Gmail 'remembers' the state of my inbox from one request to the next.

Preserving state is essentially a technical way of saying "the web site remembers things."



gmail.com

gmail.com - Gmail
mail.google.com/mail/u/0/#inbox - Inbox - kwan.carey@gmail.com - Gmail
mail.google.com/mail/u/0/#inbox/16299317939c5591 - wireframe/user flow - kwan.carey@gmail.com - Gmail
mail.google.com/mail/u/0/ - Inbox - kwan.carey@gmail.com - Gmail
gmail - Google Search
gmail account login



https://mail.google.com/mail/u/0/#inbox

me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)
me	(no subject)

What is query string?

A query string is the portion of a URL where data is passed to a web application and/or back-end database. The reason we need query strings is that the HTTP protocol is stateless by design. For a website to be anything more than a brochure, you need to maintain state (store data). There are a number of ways to do this: On most web servers, you can use something like session state server-side. On the client, you can store via cookies. Or in the URL, you can store data via a query string.

On the world wide web, all URLs can be broken down into the protocol, the location of the file (or program) and the query string. The protocol you see in a browser is almost always HTTP; the location is the typical form of the hostname and filename (for example, www.techopedia.com/somefile.html), and the query string is whatever follows the question mark sign ("?").

For example, in the URL below, the bolded area is the query string that was generated when the term "database" was searched on the Techopedia website.

https://mail.google.com**/mail/u/0/#inbox/162d6488396ff975**

You've seen query strings before, very often, probably. For example, when you perform a Google search, you will get a URL similar to this on:

```
http://www.google.ca/search?hl=en&q=richard%20le%20guen&btnG=Google%20Search&meta=
```

Likewise, you may notice that when you visit someone's Facebook profile, the URL in your browser looks like this:

```
http://www.facebook.com/profile.php?id=48305422
```

... if you change the numeric value for 'id', you end up at someone else's profile:

```
http://www.facebook.com/profile.php?id=XXXXXXXX
```

How relevant is it to Web development? & What is the use of it?

Query string state is only useful for the most basic tasks -- e.g., providing a path to redirect the user to after they complete a given task (e.g., logging in). Otherwise, query string state is horribly insecure, difficult to implement, and in order to do it justice, it needs to be tied to some server-side state machine by containing a key to tie the client to the server's maintained state for that client.

HOW TO IMPLEMENT IT?

www.example.com?btnColor=blue

```
//pseudo code
if(isset($_GET['btnColor']) // for query string
 echo  '<button style="background:'.$_GET['btnColor'].'">'
```

```
#!/usr/bin/perl --

# We read GET parameters out of the query string
%params = ();
$queryString = $ENV{"QUERY_STRING"};
@keyValuePairs = split(/\&/, $queryString);
foreach my $keyValuePair (@keyValuePairs) {
        ($key, $value) = split(/\=/, $keyValuePair);
        $key =~ s/%([a-f0-9]{2})/chr(hex($1))/eig; # URL decode the key
        $value =~ s/%([a-f0-9]{2})/chr(hex($1))/eig; # URL decode the value
        $params{$key} = $value;
}

if(!$params{"SESSIONID"} || !-e "/tmp/r_leguen.cgi_blackjack.".$params{"SESSIONID"}) {
                                # if no session ID is provided,
                                # we initialize a session
        # ...
}
else {  # if a session ID is provided, we initialize session data.
        # read from a file where we stored the session info.
        open("SESSION", "</tmp/r_leguen.cgi_blackjack.".$sessionID);
        # read data from the session file…
        close(SESSION);
}

# code for the logic
# and rules of Blackjack

# save session information in a file
open("SESSION", ">/tmp/r_leguen.cgi_blackjack.".$sessionID);
# write data to the session file…
close(SESSION);
```