Mojolicious and Content-Security-Policy

Erik Johansen
DK Hostmaster A/S
Mojoconf 2014
2014-05-24

Content-Security-Policy

HTML response header

Add security constraints to a web page.

Content-Security-Policy

Enforced by your web browser.

Firefox	23.0	...
Chrome	25.0	...
Safari	7.0	...
Opera	15.0	...
Android browser	4.4	
Opera mobile	21.0	
Chrome for Android	33.0	

Content-Security-Policy

Deprecated headers:

X-Content-Security-Policy

X-Webkit-CSP

Check compatibility on
http://caniuse.com/contentsecuritypolicy

Content-Security-Policy

What can you restrict ?

Content-Security-Policy

Restricts acces by setting
the permitted origins
for each type of content.

Content-Security-Policy

What does it look like ?

Content-Security-Policy

```
Content-Security-Policy:

      report-uri    /some_report_uri;

      default-src    'none';

      img-src        'self' https:;
```

Content-Security-Policy

To get enforcement and reports:

```
Content-Security-Policy:
      report-uri   /content-security-policy-report;
      default-src  'none';
      img-src      'self' https:
```

To get enforcement (no reports):

```
Content-Security-Policy:
      default-src  'none';
      img-src      'self' https:
```

To get only reports (no enforcement):

```
Content-Security-Policy-Report-Only: …
```

Content-Security-Policy

```
Content-Security-Policy:

      report-uri    /some_report_uri;

      default-src    'none';

      img-src          'self' https:;
      script-src      'self' https:;
      style-src       'self' https:;
      font-src        'self' https:;

      <source type>  <origins>... ;
```

Content-Security-Policy

Source types

`'default-src'`	Default for unspecified source types
`'image-src'`	Images can be loaded from
`'script-src'`	Javascript can be loaded from
`'font-src'`	Fonts can be loaded from
`'style-src'`	Styles can be loaded from
`'frame-src'`	Embedded frames can be loaded from
`'media-src'`	Media (audio/video) can be delivered from
`'object-src'`	Flash and other plugins can be loaded from
`'connect-src'`	Connections can be made to (via XHR, WebSockets, and EventSource)
`'sandbox'`	Sandbox as for inside iframe (same origin policy)

Content-Security-Policy

Origins

`'none'`	None (no match)
`'self'`	Same host
`'unsafe-inline'`	Inline JS/CSS
`'unsafe-eval'`	eval()
`https:`	Only `https:`
`https://apis.google.com/`	Exact
`example.com`	Hostname
`*://*.example.com:*`	Wildcards

Content-Security-Policy

Sandbox options

`allow-forms`	Sandbox forms
`allow-same-origin`	Sandbox SO
`allow-scripts`	Sandbox scripts
`allow-top-navigation`	Sandbox nav.

Mojolicious
Generate lite app

```
$ mojo generate lite_app  \
content_security_policy.pl

$ morbo content_security_policy.pl
```

Lite app

content_security_policy.conf

```
{
  secrets => [ 'some_secret' ],

  'content-security-policy' =>
    join("; ",
      "report-uri /content-security-policy-report",

      "default-src   'none'",

      "script-src    'self'",
      "style-src     'self'",

      "font-src      'self'",
      "img-src       'self'",
    ),
}
```

Lite app
main part

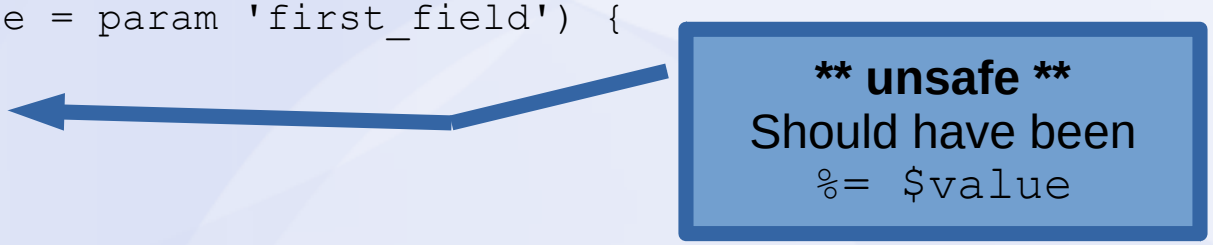
```
#!/usr/bin/env perl
use Mojolicious::Lite;

# Documentation browser under "/perldoc"
plugin 'PODRenderer';

my $cfg = plugin 'config';
app->secrets( $cfg->{secrets} );

if ( my $csp = $cfg->{'content-security-policy'} ) {
  # … (next slide) …
}

get '/' => sub {
  my $self = shift;
  $self->render('index');
};


app->start;
__DATA__
```

Lite app
main part

```
if ( my $csp = $cfg->{'content-security-policy'} ) {
    $csp =~ s/ {2,}/ /g;

    hook before_dispatch => sub {
        my $self = shift;

        $self->res->headers->header('Content-Security-Policy' => $csp);
    };

    if ( my $report_uri = $csp =~ /\breport-uri\s+([^ ;]+)/ && $1 ) {
        post $report_uri => sub {
            my $self = shift;
            my $report = $self->req->json;

            $self->app->log->warn("Policy report:",
                $self->dumper( $report )
            )  if $report;

            $self->render( text => "Thanks for the report!" );
        };
    }
}
```

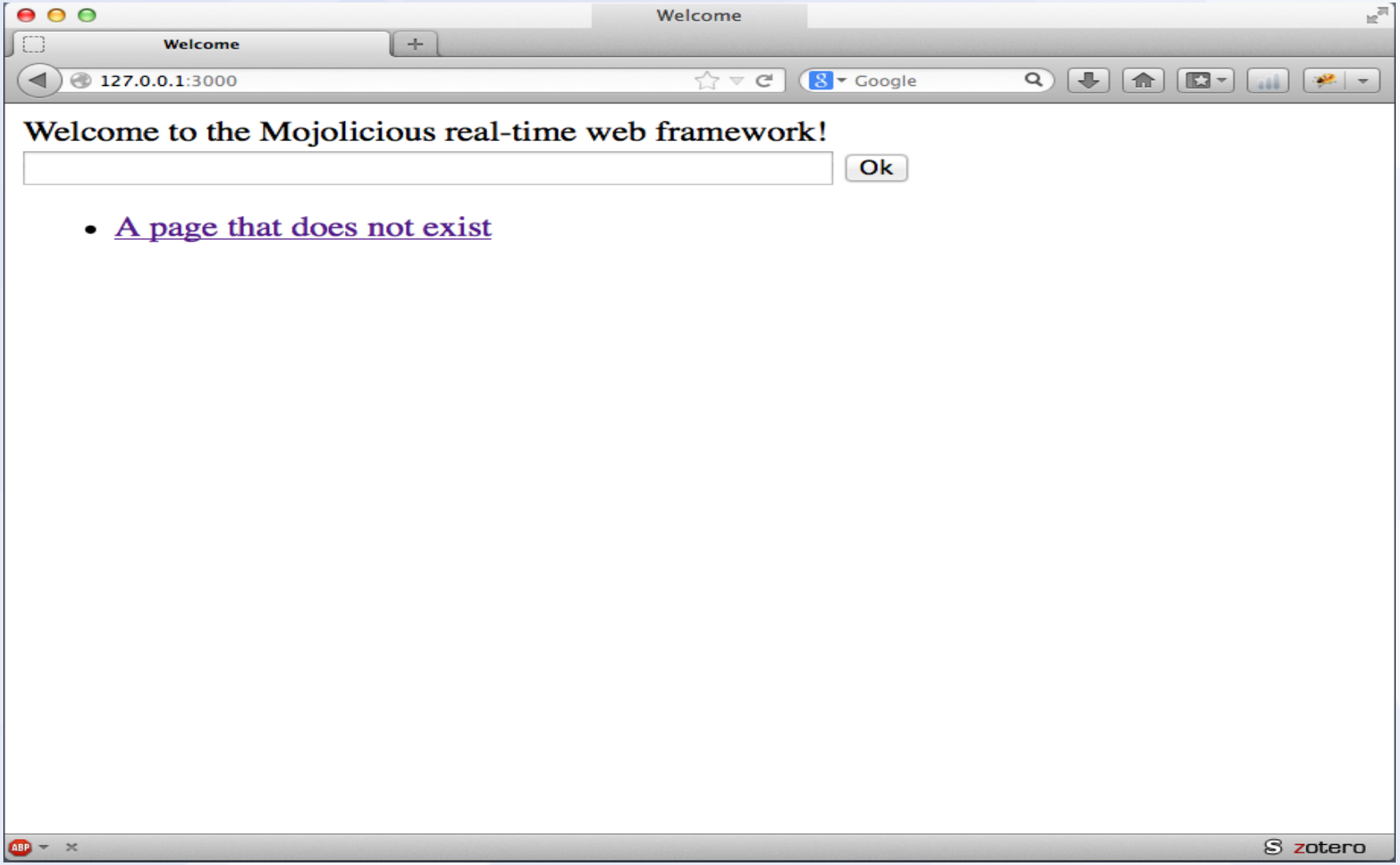
Lite app
templates

```
__DATA__

@@ index.html.ep
% layout 'default';
% title 'Welcome';
Welcome to the Mojolicious real-time web framework!

%= form_for "" => begin
  %= text_field 'first_field', size => 50
  %= submit_button
%= end

% if (my $value = param 'first_field') {
  You typed:
   %== $value
% }

<ul>
 <li> <a href="/nosuch">A page that does not exist</a> </li>
</ul>

@@ layouts/default.html.ep
<!DOCTYPE html>
<html>
  <head><title><%= title %></title></head>
  <body><%= content %></body>
</html>
```



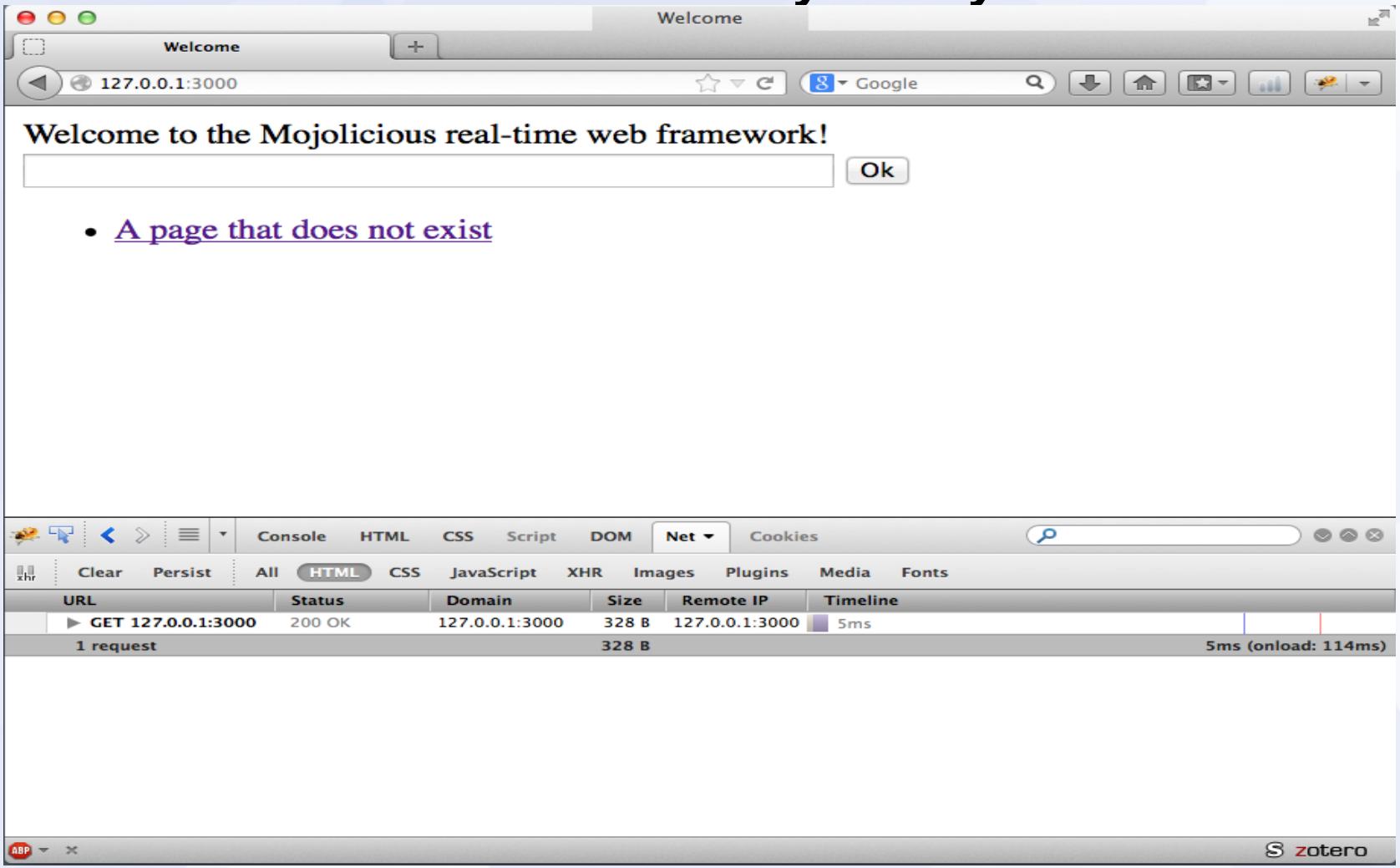
Lite app

test



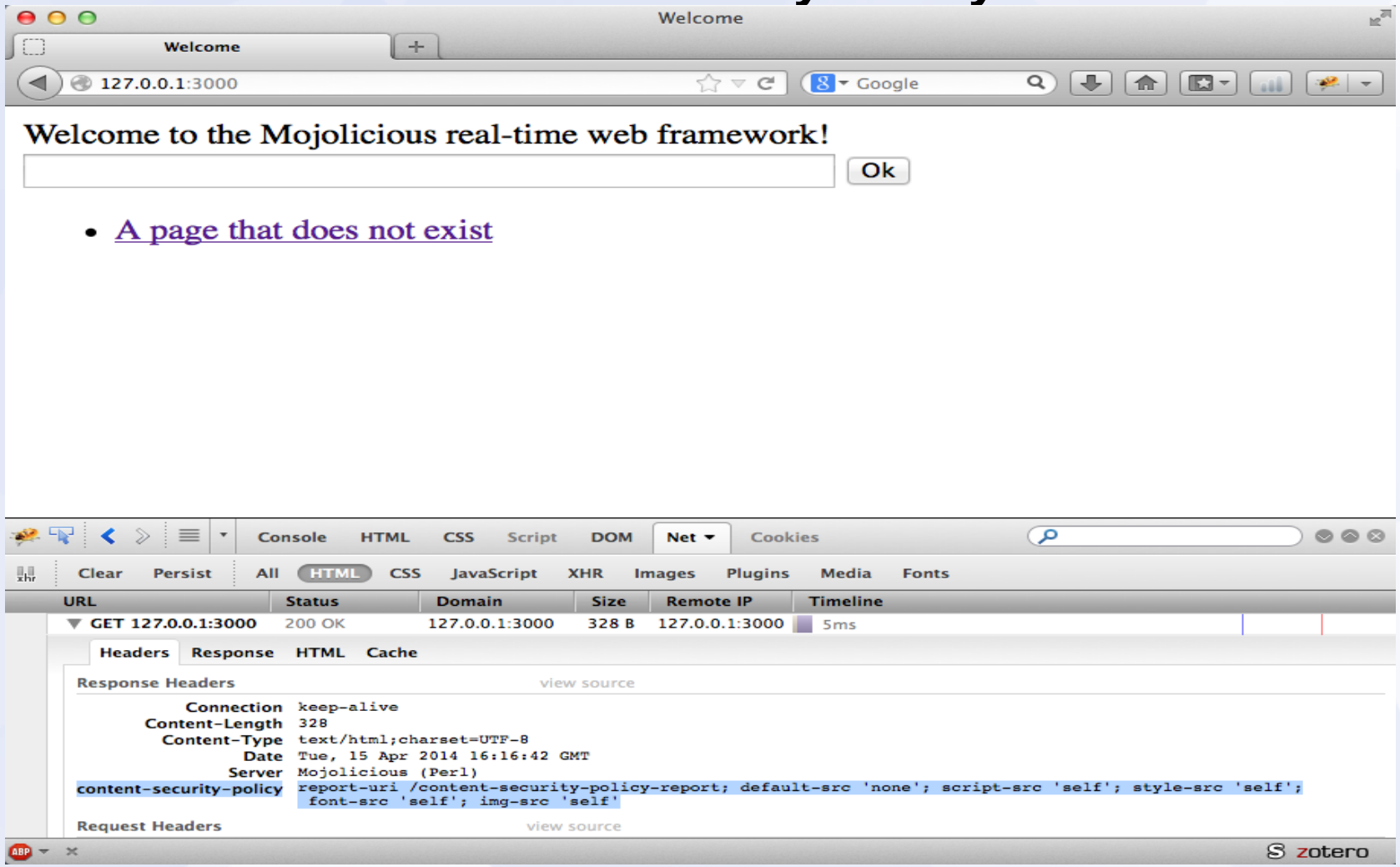
Lite app

Added Content-Security-Policy header



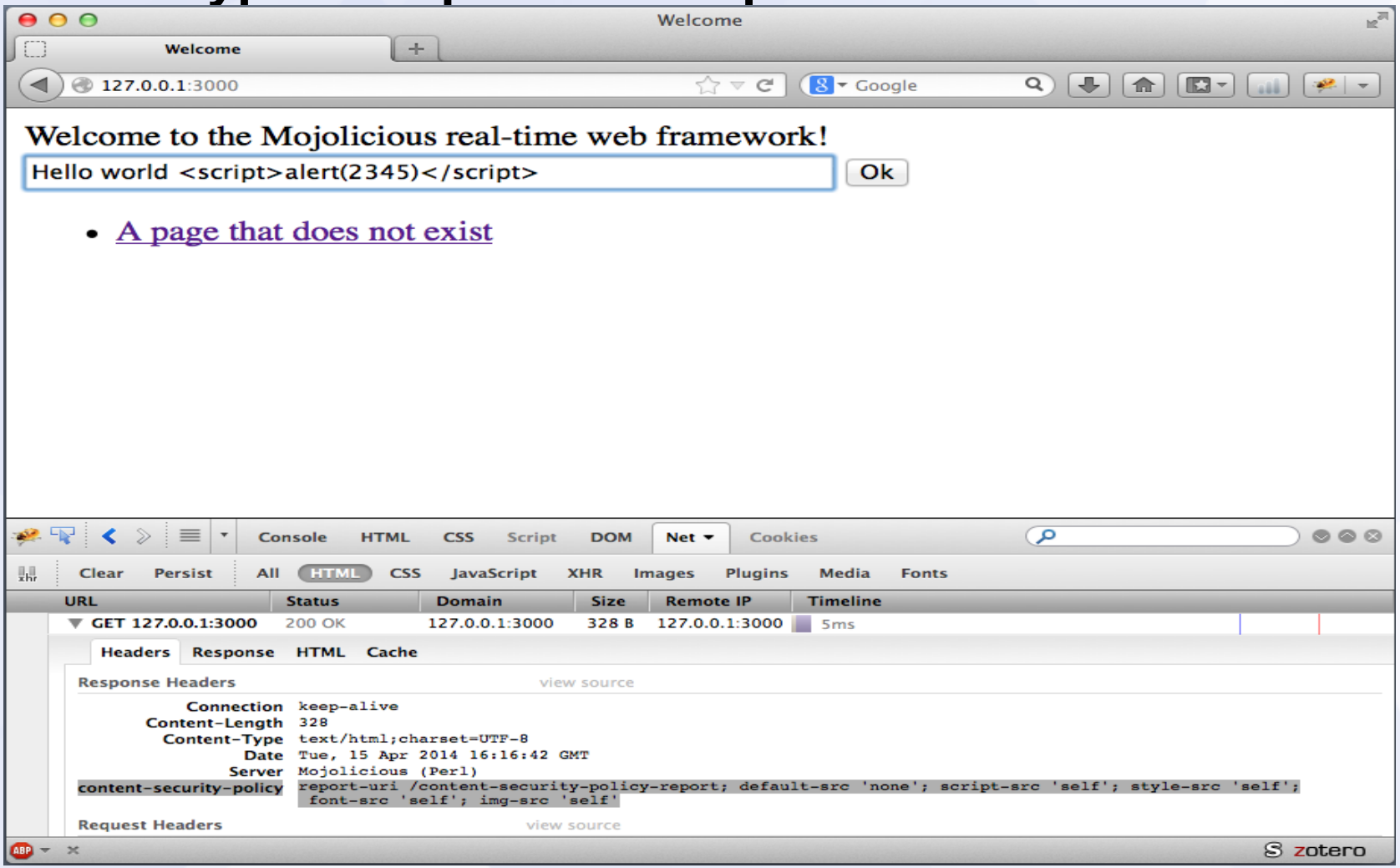
Lite app

Added Content-Security-Policy header



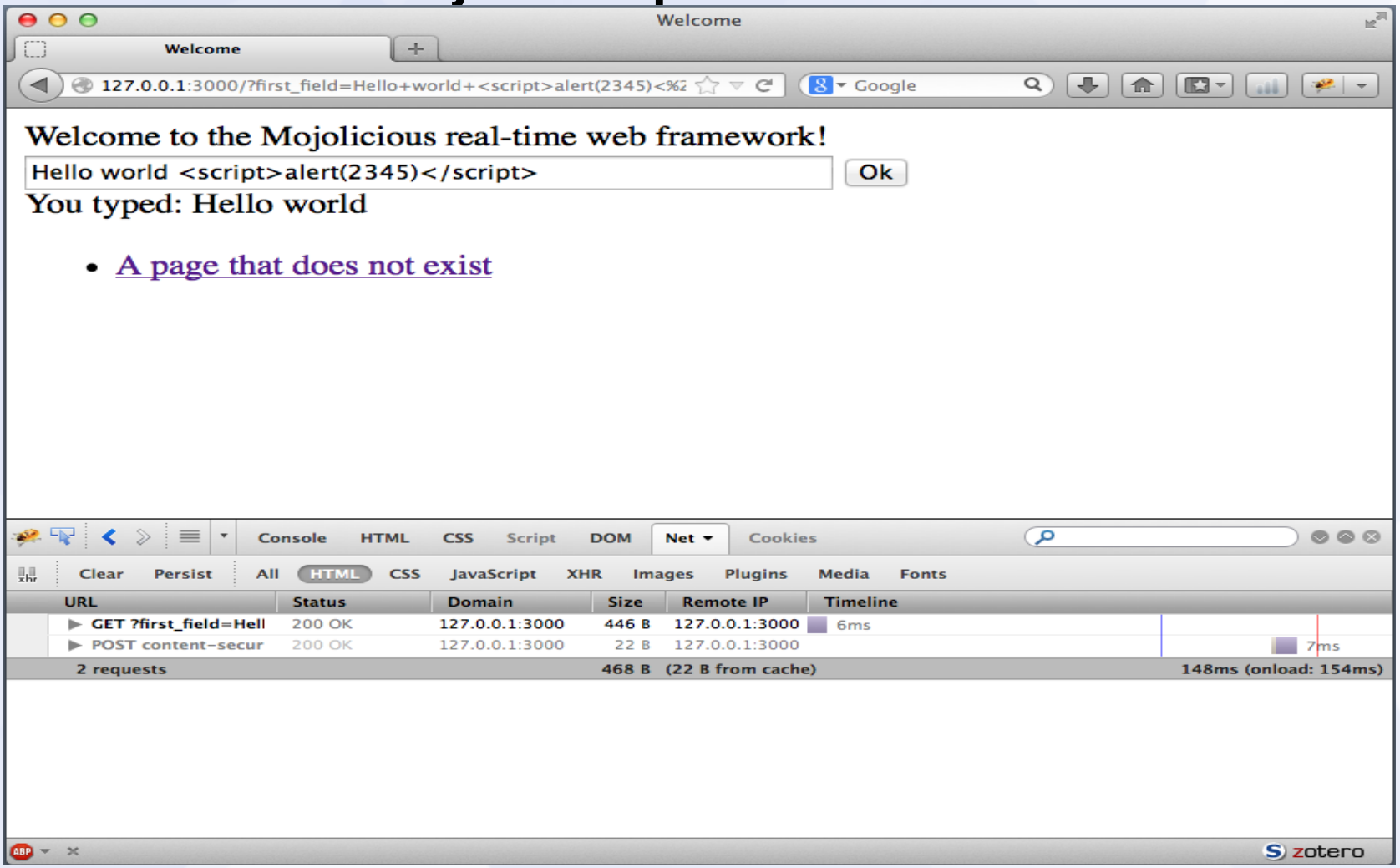
Lite app

Type <script>... into input box. Press OK.



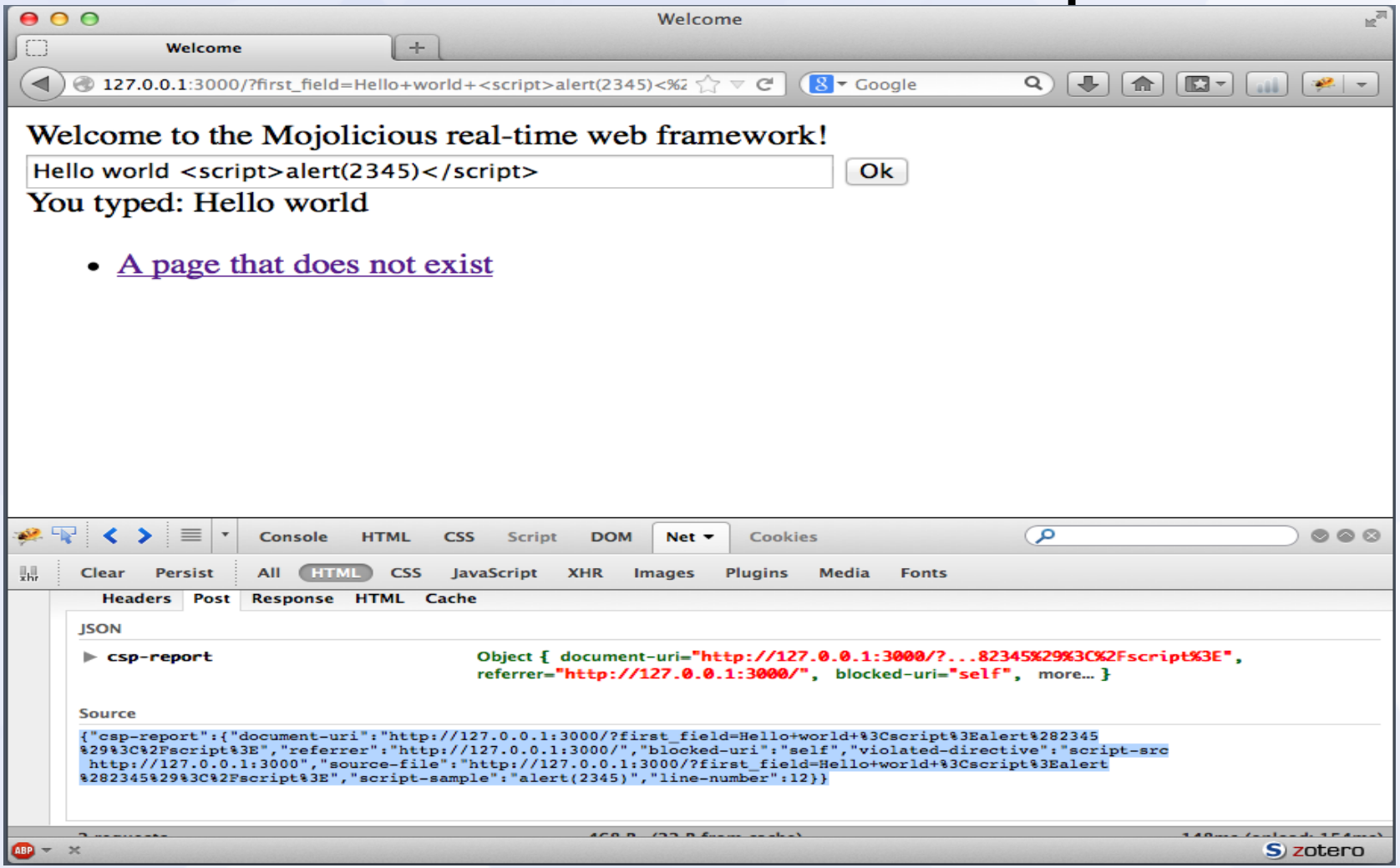
Lite app

The javascript does not run.



Lite app

Instead browser sends a JSON report.



Lite app

The JSON report is written to the log.

```
[Wed Apr 16 17:01:55 2014] [debug] 200 OK (0.005466s, 182.949/s).
[Wed Apr 16 17:01:55 2014] [debug] POST "/content-security-policy-
report".
[Wed Apr 16 17:01:55 2014] [debug] Routing to a callback.
[Wed Apr 16 17:01:55 2014] [warn] Policy report:
{
  "csp-report" => {
    "blocked-uri" => "self",
    "document-uri" => "http://127.0.0.1:3000/?first_field=Hello+world+
%3Cscript%3Ealert%282345%29%3C%2Fscript%3E",
    "line-number" => 12,
    "referrer" => "http://127.0.0.1:3000/",
    "script-sample" => "alert(2345)",
    "source-file" => "http://127.0.0.1:3000/?first_field=Hello+world+
%3Cscript%3Ealert%282345%29%3C%2Fscript%3E",
    "violated-directive" => "script-src http://127.0.0.1:3000"
  }
}

[Wed Apr 16 17:01:55 2014] [debug] 200 OK (0.001660s, 602.410/s).
```

Mojolicious
Side effects

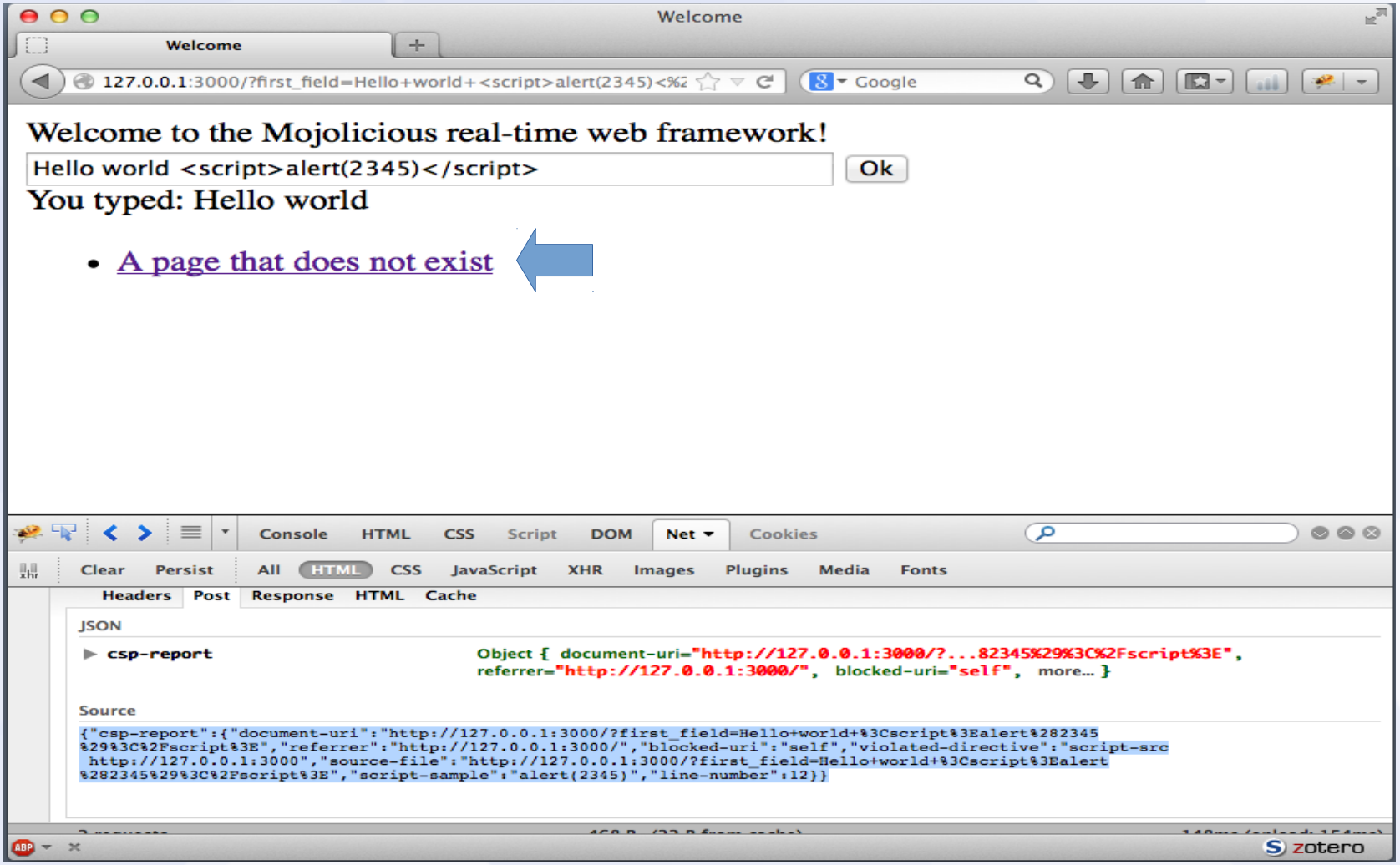
Any side effects ?

Mojolicious
Side effects

Mojolicious ”not_found” page
uses inline styles and scripts
and will be refused by a strict policy.

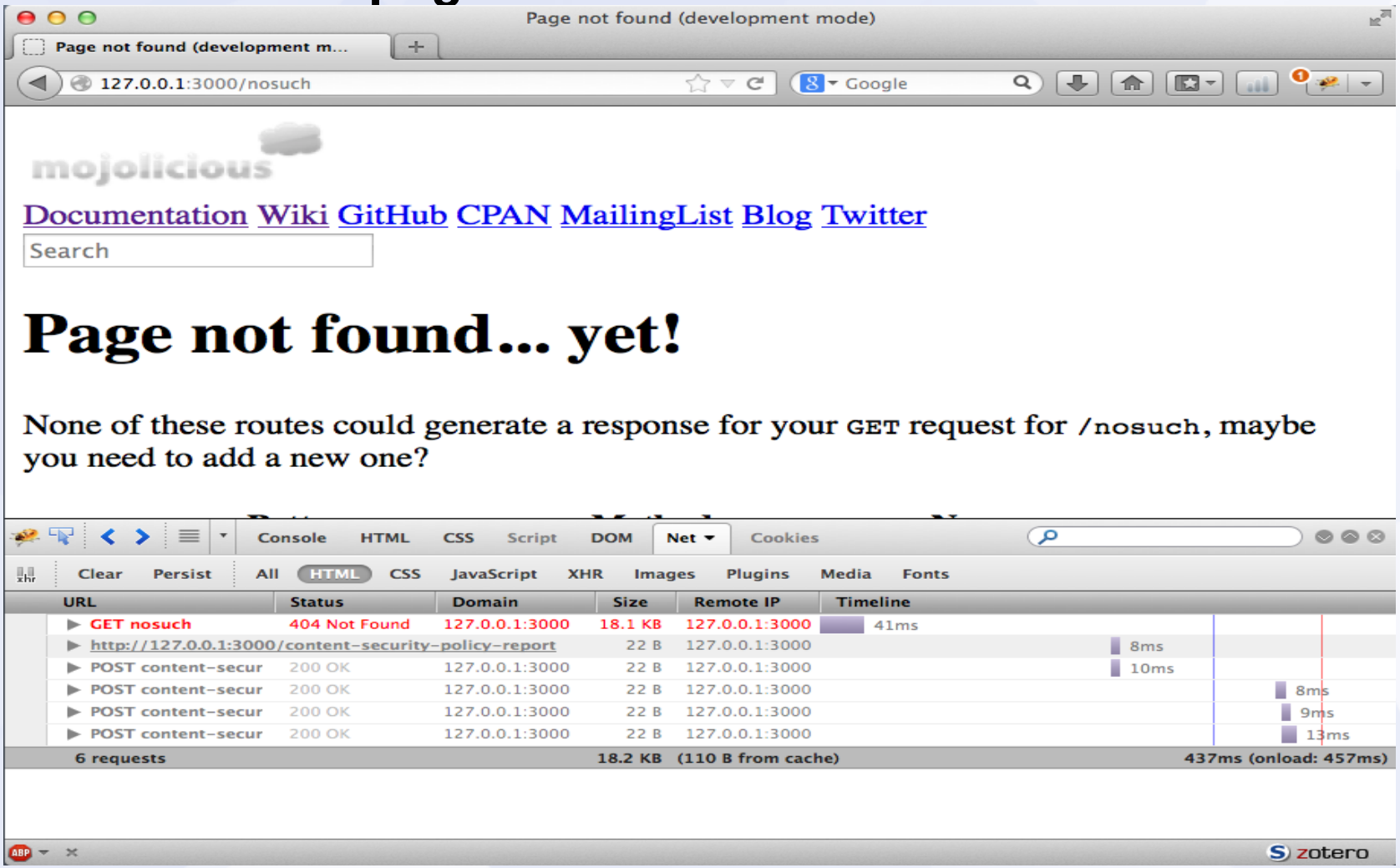
Lite app

Use /nosuch link.



Mojolicious

Error page when unsafe-inline denied



Mojolicious

Side effects

If you want it to look nice again.

Mojolicious

Side effects

One way is to allow
unsafe-inline scripts
and
unsafe-inline css
in development mode.

Lite app

content_security_policy.conf

```
{
  secrets => [ 'some_secret' ],

  'content-security-policy' =>
    join("; ",
      "report-uri /content-security-policy-report",

      "default-src   'none'",


      "script-src    'self'",
      "style-src     'self'",


      "font-src      'self'",
      "img-src       'self'",
    ),
}
```

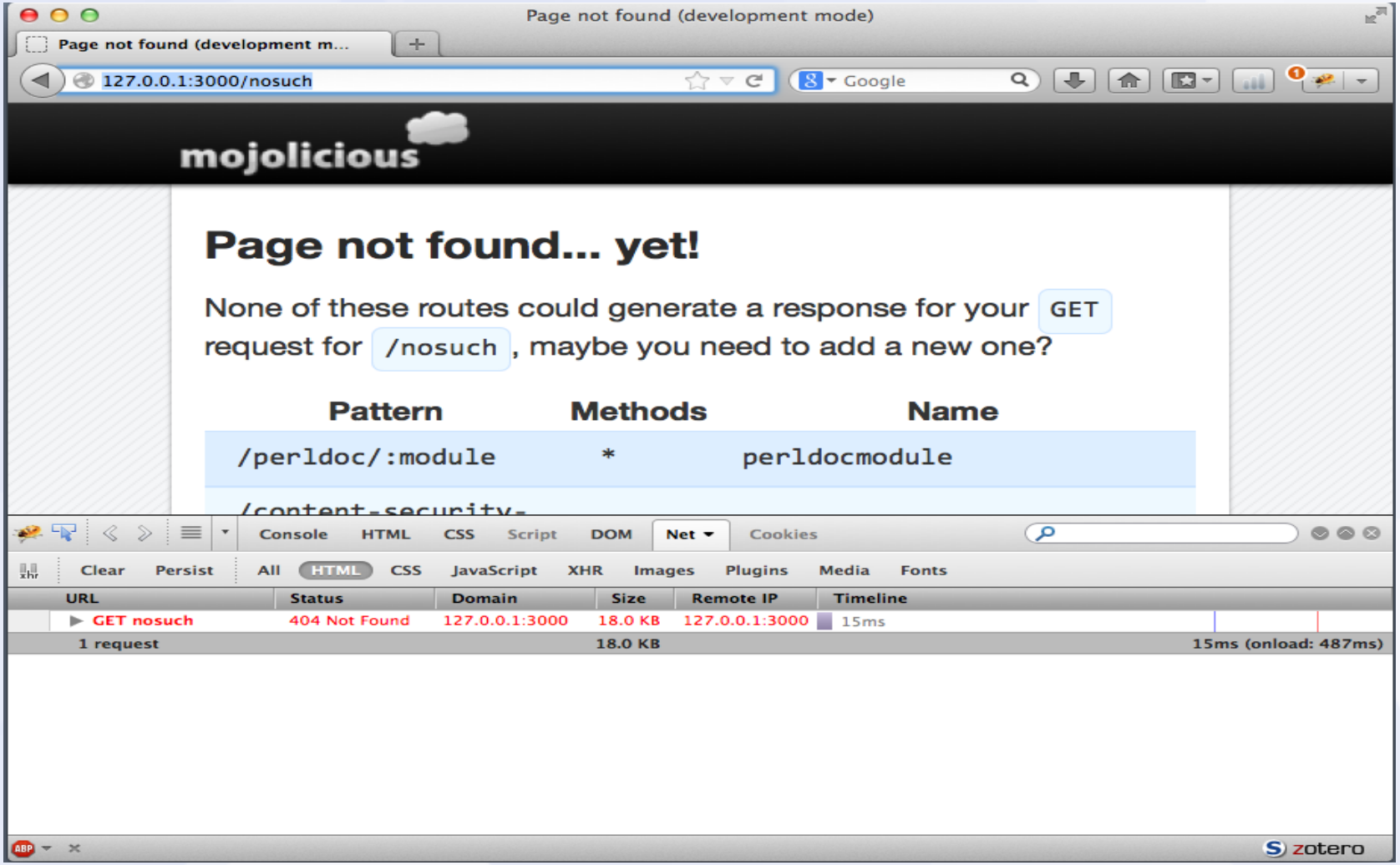
Lite app

content_security_policy.development.conf

```
{
  secrets => [ 'some_development_secret' ],

  'content-security-policy' =>
    join("; ",
      "report-uri /content-security-policy-report",

      "default-src   'none'",

      # for mojo not_found page
      "script-src    'self' 'unsafe-inline'",
      "style-src     'self' 'unsafe-inline' https://google-code-
prettify.googlecode.com",

      "font-src      'self'",
      "img-src       'self'",
    ),
}
```

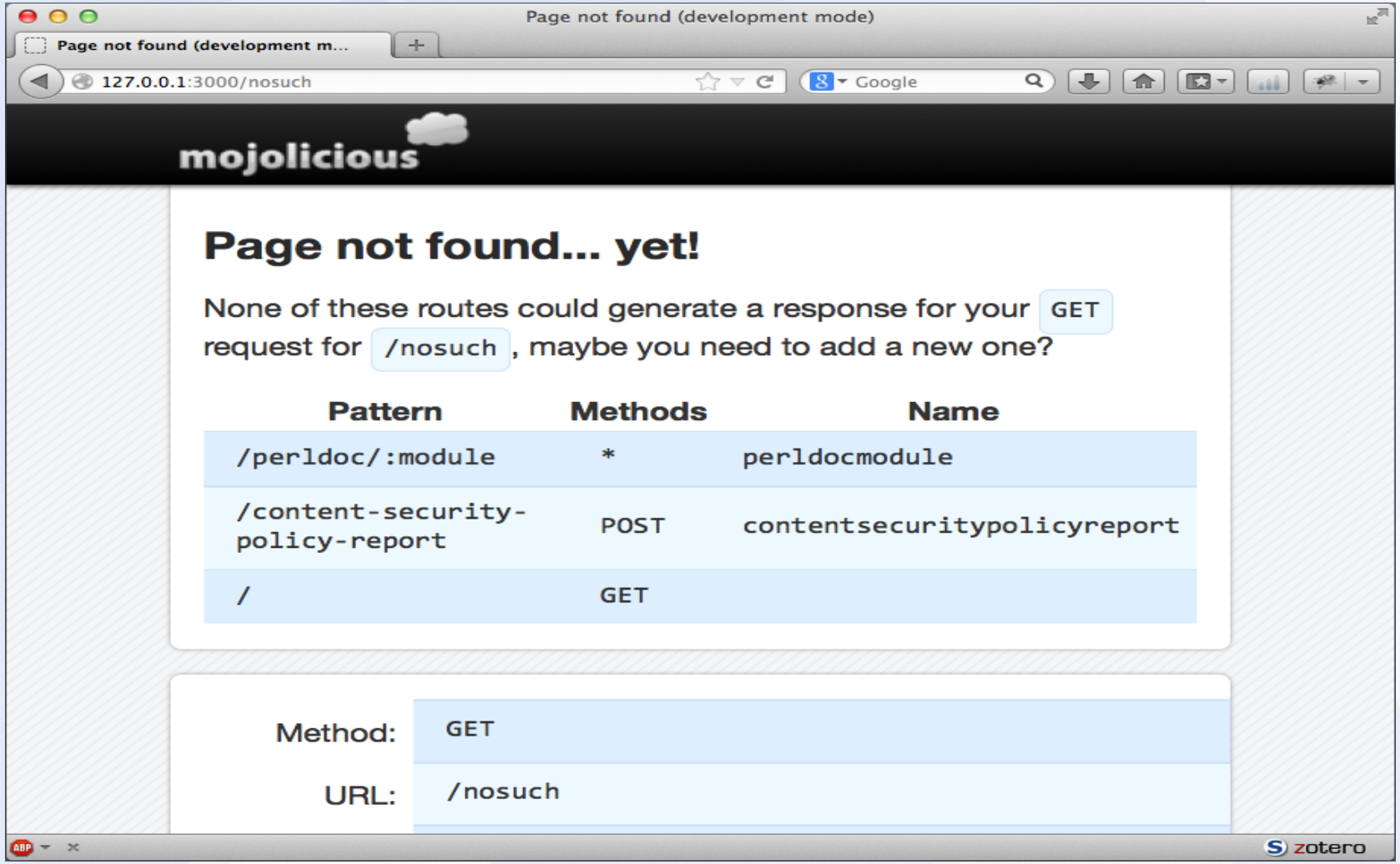
Mojolicious

Now it looks ok



Mojolicious

Now it looks ok

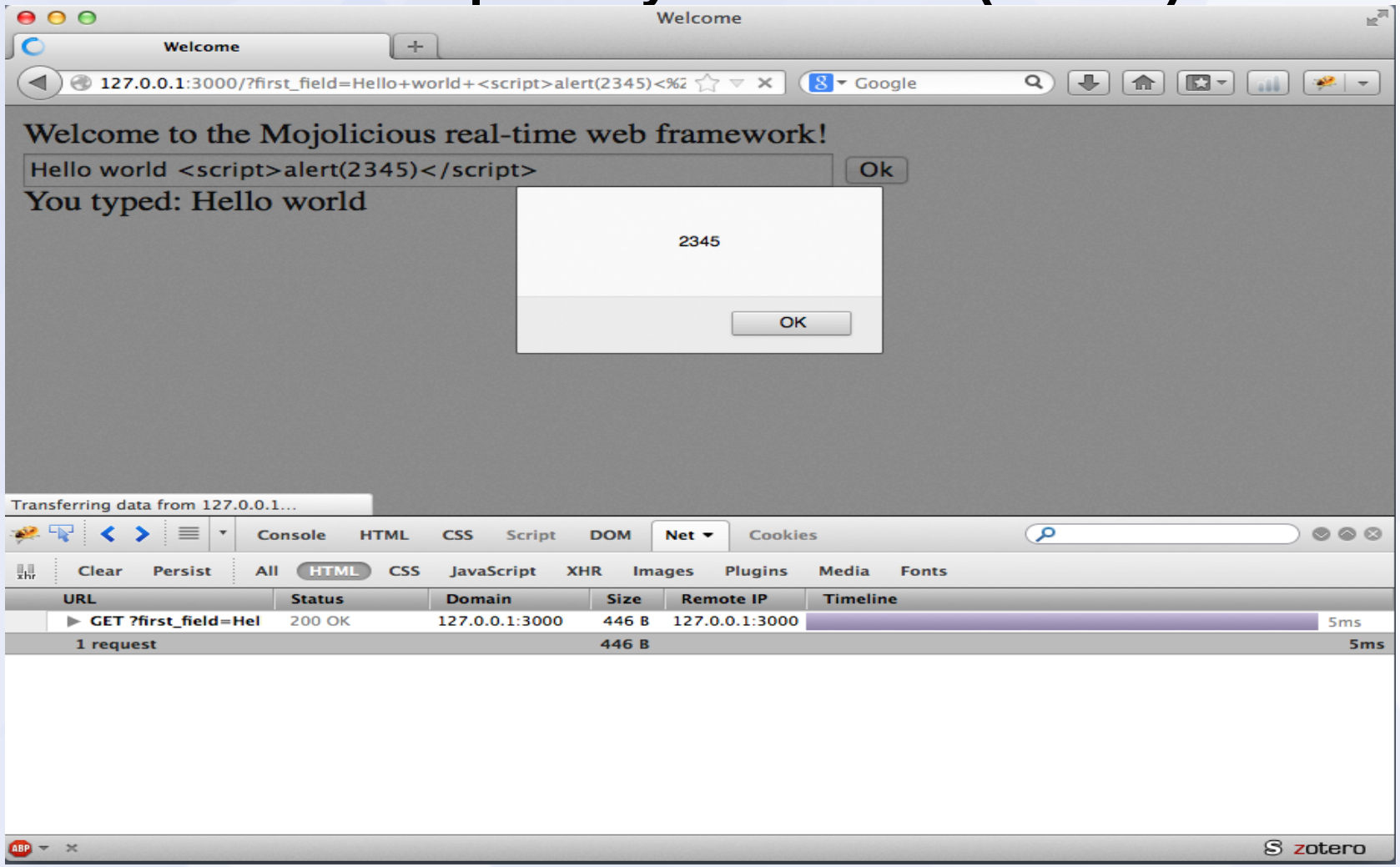


Mojolicious

But then...

Lite app

Other scripts may run as well (unsafe)



End of presentation...

References

Download this presentation:

http://www.uniejo.dk/presentations/2014-05-24-csp.pdf

Download test script:

http://www.uniejo.dk/presentation/2014-05-24-csp.tgz

Read more on
”HTML5 Rocks”:
http://www.html5rocks.com/en/tutorials/security/content-security-policy/

”Content Security Policy Best Practices”:
https://www.isecpartners.com/media/106598/csp_best_practices.pdf

End of presentation...