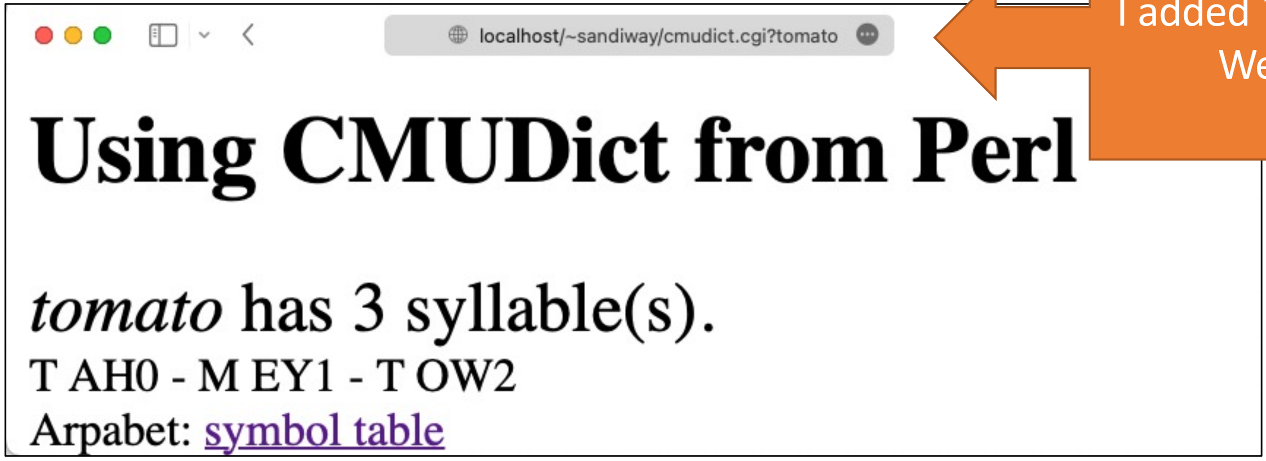
Lecture 20

408/508 *Computational Techniques for Linguists*

Last Time


localhost/~sandiway/cmudict.cgi?tomato

Using CMUDict from Perl

tomato has 3 syllable(s).
T AH0 - M EY1 - T OW2
Arpabet: symbol table

I added `?tomato` in manually on the URL line
We will define a <form> this time!

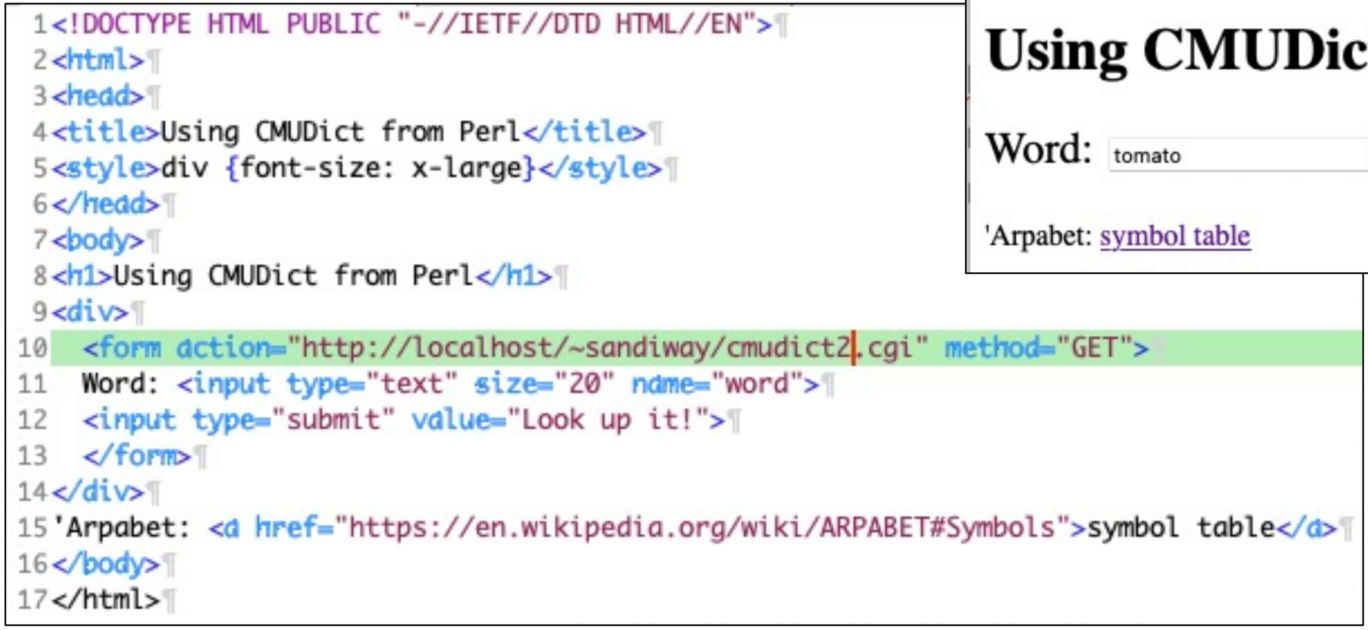
- **Note**: slides corrected for Lecture 19 wrt. macOS configuration files
- Apple Perl executable (brew.sh differences):
  - M1/M2 Mac: #!/opt/homebrew/bin/perl
  - Intel Mac: #!/opt/local/bin/perl

Today's Topic

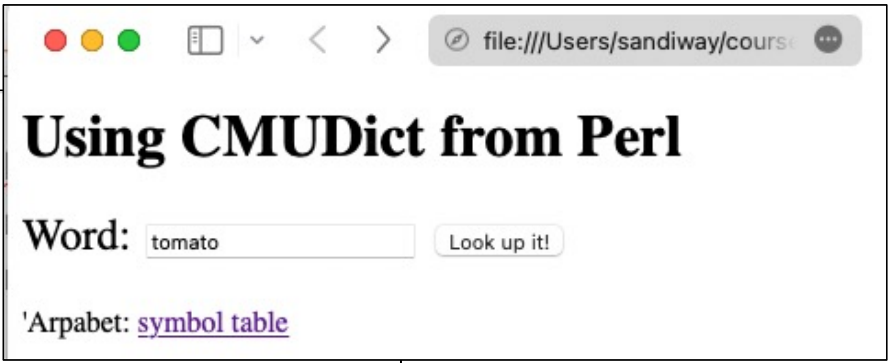
- Sending form information to a Webserver program.
- Two methods:
    1. GET method
    2. POST method

A worked example: `CMUDict`

- Let's define a `<form>` with a GET action
  File: `cmudictform.html`



```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>Using CMUDict from Perl</title>
<style>div {font-size: x-large}</style>
</head>
<body>
<h1>Using CMUDict from Perl</h1>
<div>
  <form action="http://localhost/~sandiway/cmudict2.cgi" method="GET">
  Word: <input type="text" size="20" name="word">
  <input type="submit" value="Look up it!">
  </form>
</div>
'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>
</body>
</html>
```



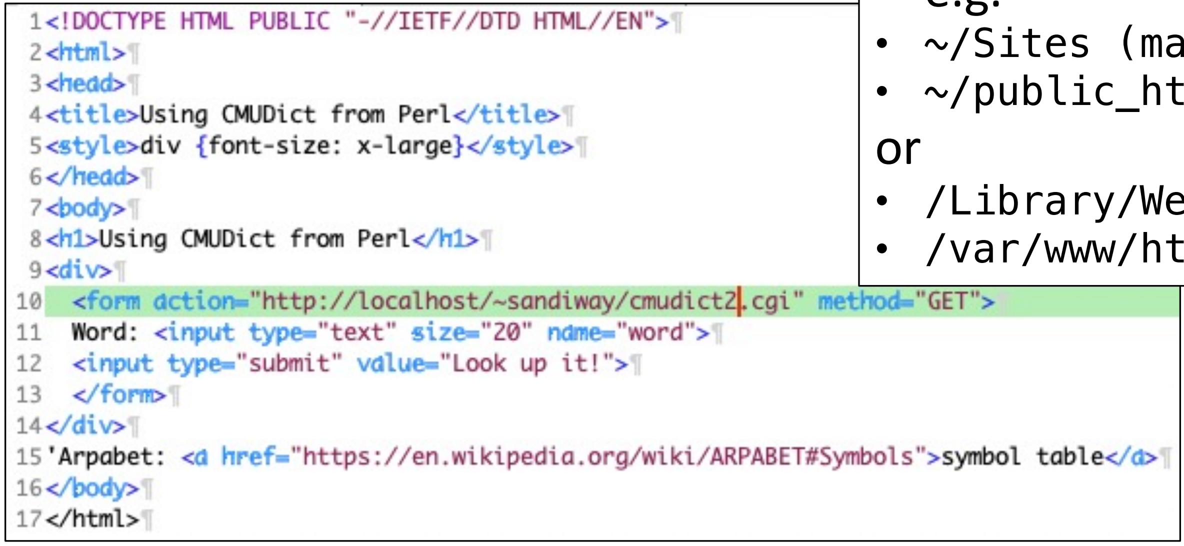
Using CMUDict from Perl

Word: tomato Look up it!

'Arpabet: symbol table

A worked example: `CMUDict`

File: `cmudictform.html`



```html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>Using CMUDict from Perl</title>
<style>div {font-size: x-large}</style>
</head>
<body>
<h1>Using CMUDict from Perl</h1>
<div>
  <form action="http://localhost/~sandiway/cmudict2.cgi" method="GET">
  Word: <input type="text" size="20" name="word">
  <input type="submit" value="Look up it!">
  </form>
</div>
'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>
</body>
</html>
```

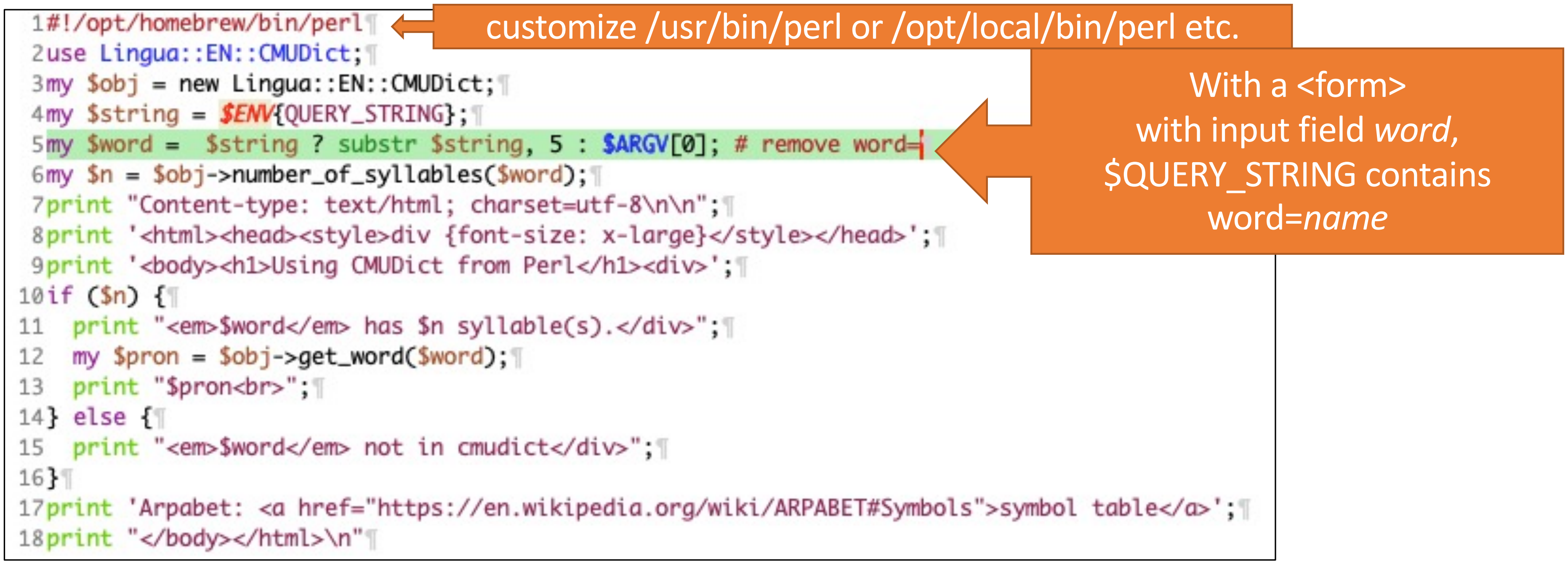
- the HTML form can be placed anywhere,
- e.g.
- `~/Sites (macOS)`
- `~/public_html (Ubuntu)`

or

- `/Library/WebServer/Documents/ (macOS)`
- `/var/www/html/ (Ubuntu)`

A worked example: `CMUDict`

- `File: cmudict2.perl`
- Permissions: `chmod a+x cmudict2.cgi`

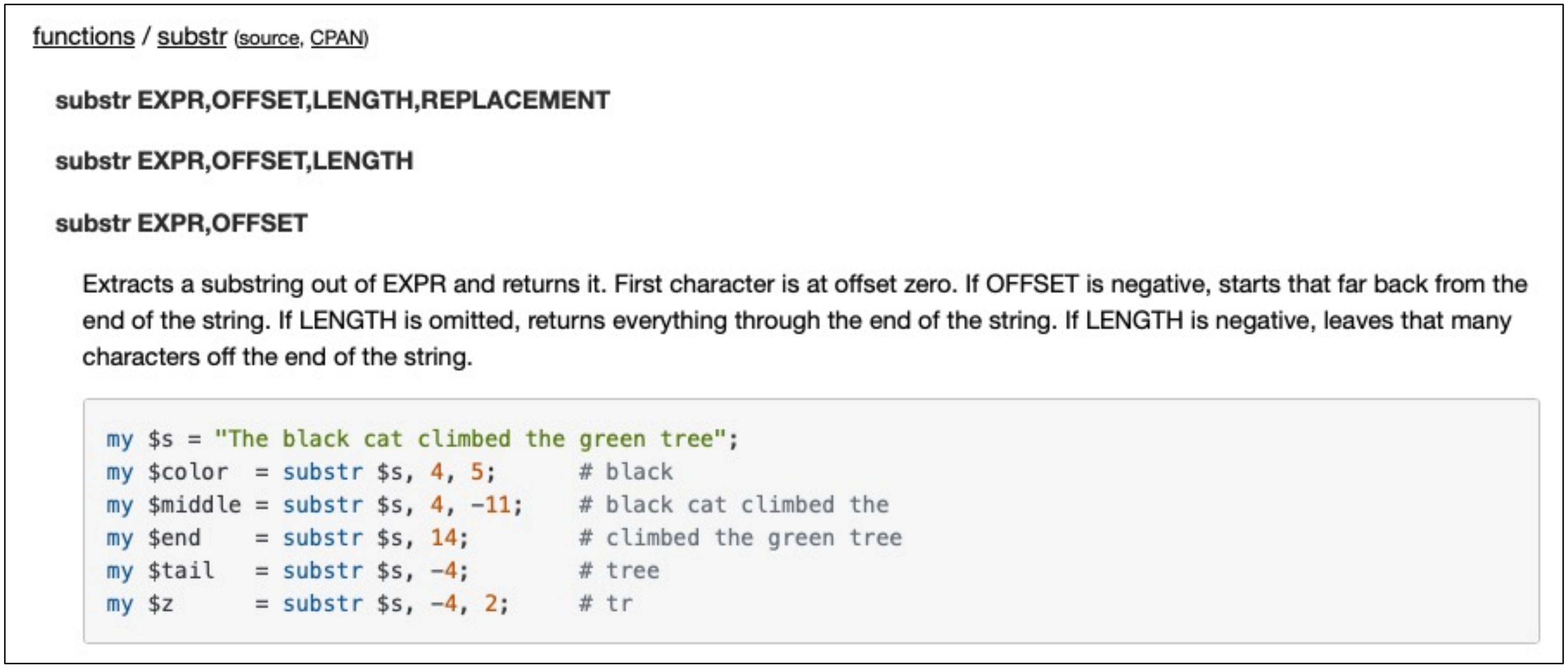


```perl
#!/opt/homebrew/bin/perl
use Lingua::EN::CMUDict;
my $obj = new Lingua::EN::CMUDict;
my $string = $ENV{QUERY_STRING};
my $word =  $string ? substr $string, 5 : $ARGV[0]; # remove word=
my $n = $obj->number_of_syllables($word);
print "Content-type: text/html; charset=utf-8\n\n";
print '<html><head><style>div {font-size: x-large}</style></head>';
print '<body><h1>Using CMUDict from Perl</h1><div>';
if ($n) {
  print "<em>$word</em> has $n syllable(s).</div>";
  my $pron = $obj->get_word($word);
  print "$pron<br>";
} else {
  print "<em>$word</em> not in cmudict</div>";
}
print 'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>';
print "</body></html>\n"
```

customize /usr/bin/perl or /opt/local/bin/perl etc.

With a <form> with input field *word*, $QUERY_STRING contains word=*name*

Perl substr function



functions / substr (source, CPAN)

substr EXPR,OFFSET,LENGTH,REPLACEMENT

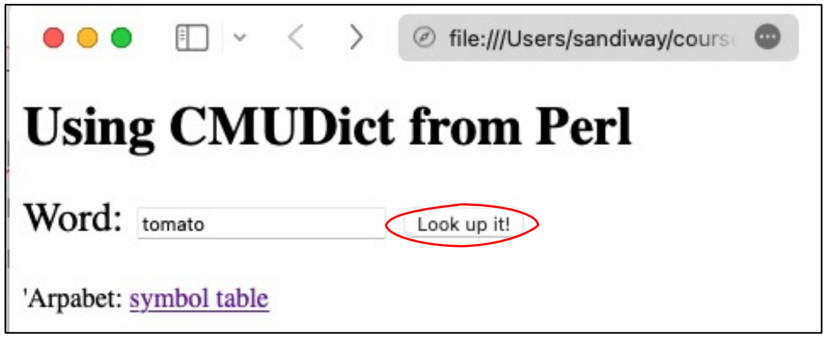
substr EXPR,OFFSET,LENGTH

substr EXPR,OFFSET

Extracts a substring out of EXPR and returns it. First character is at offset zero. If OFFSET is negative, starts that far back from the end of the string. If LENGTH is omitted, returns everything through the end of the string. If LENGTH is negative, leaves that many characters off the end of the string.

```
my $s = "The black cat climbed the green tree";
my $color  = substr $s, 4, 5;      # black
my $middle = substr $s, 4, -11;    # black cat climbed the
my $end    = substr $s, 14;        # climbed the green tree
my $tail   = substr $s, -4;        # tree
my $z      = substr $s, -4, 2;     # tr
```

A worked example: `CMUDict`



Using CMUDict from Perl

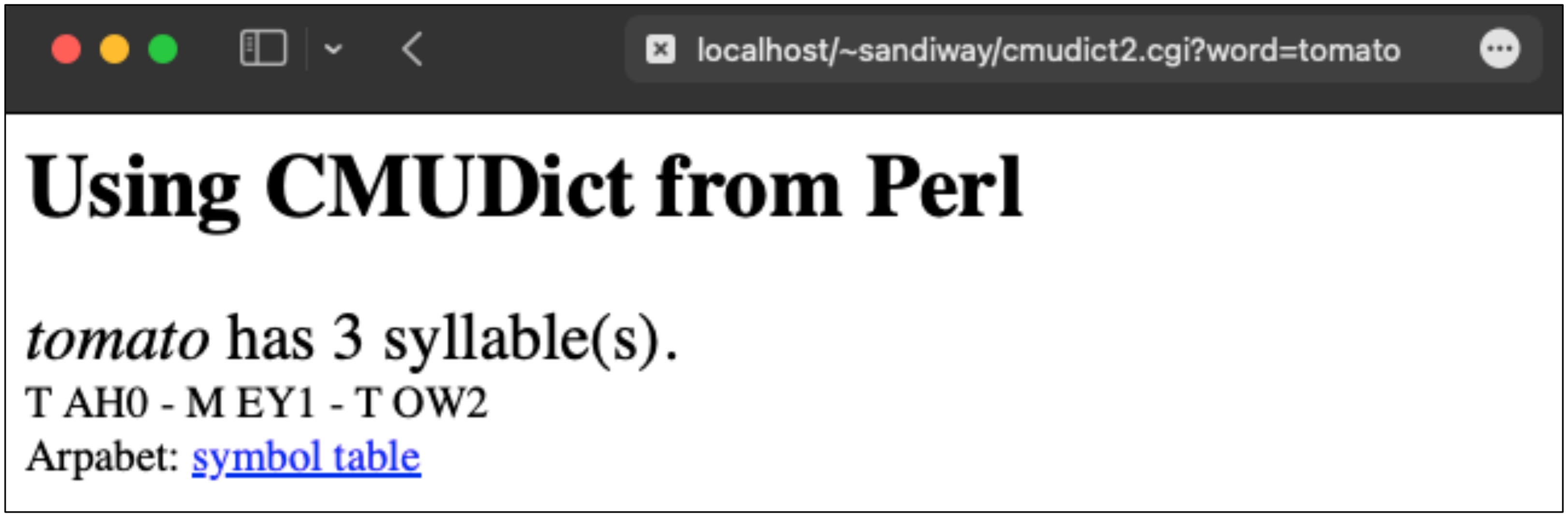
Word: tomato [Look up it!]

'Arpabet: symbol table

```
<form
action="http://localhost/~sandiway/cmudict2.cgi"
method="GET">
  Word: <input type="text" size="20" name="word">
  <input type="submit" value="Look up it!">
</form>
```

- Apache2 webserver receives:
  - `http://localhost/~sandiway/cmudict2.cgi?word=tomato`
- Webserver sets $QUERY_STRING:
  - `word=tomato`
- Webserver runs the program:
  - `~/Sites/cmudict2.cgi (macOS)`
  - `~/public_html/cmudict2.cgi (Ubuntu)`

A worked example: `CMUDict`



Using CMUDict from Perl

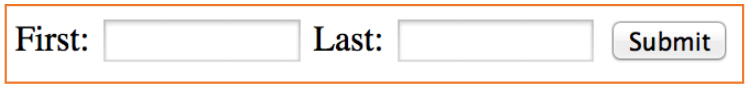
tomato has 3 syllable(s).
T AH0 - M EY1 - T OW2
Arpabet: symbol table

- This is the Perl-computed response
- by the Webserver to the client Browser

A worked example: `CMUDict`

- **Note**: the file
  - `cmudict2.cgi` could also be placed in `/Library/WebServer/CGI-Executables`
- **Then**
  - `<form action=http://localhost/cgi-bin/cmudict2.cgi method="GET">`
- Webserver runs the program:
  - `/Library/WebServer/CGI-Executables/cmudict2.cgi (macOS)`
  - `/var/www/html/cmudict2.cgi (Ubuntu)`

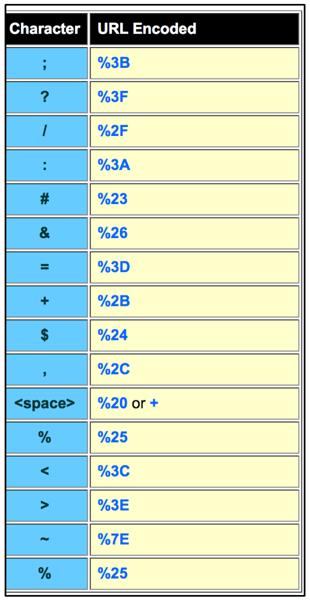
Sending information using **GET**



- HTML form:

```
<form action="http://localhost/cgi-bin/get.cgi" method="GET">
First: <input type="text" name="first" size=12>
Last:  <input type="text" name="last" size=12>
<input type="submit">
</form>
```

```
http://localhost/cgi-bin/get.cgi?first=Sandiway&last=Fong
```



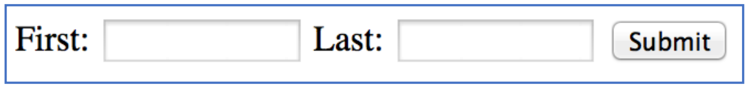
Character	URL Encoded
;	%3B
?	%3F
/	%2F
:	%3A
#	%23
&	%26
=	%3D
+	%2B
$	%24
,	%2C
<space>	%20 or +
%	%25
<	%3C
>	%3E
~	%7E
%	%25

- Information encoded using alphanumeric characters: why?
- URLs are restricted to alphanumeric characters only
- **bash** accesses the URL-encoded string via the environment variable **`QUERY_STRING`**

Today's Topic

- Sending form information to a Webserver program.
- Two methods:
    1. GET method
    2. **POST method**

Sending information using POST



- File: `form-post.html` could be in `/Library/WebServer/Documents`
- URL: `http:/localhost/form-post.html`
- HTML form:

```
<form action="http://localhost/cgi-bin/read.cgi" method="POST">
First: <input type="text" name="first" size=12>
Last:  <input type="text" name="last" size=12>
<input type="submit">
</form>
```

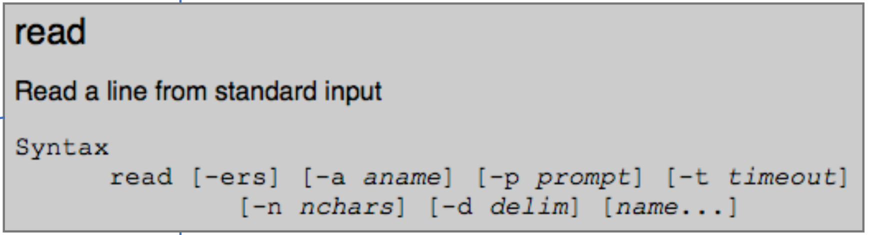
- **bash** accesses the URL-encoded string via command read
  - cf. GET using **`QUERY_STRING`**

Sending information using **POST**

- **bash** accesses the URL-encoded string on standard input via read

- read.cgi in /Library/WebServer/CGI-Executables:

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
read input
origIFS=$IFS
IFS='=&'
set -- $input
IFS=$origIFS
echo "Form data \$2:<$2> \$4:<$4>"
```



read

Read a line from standard input

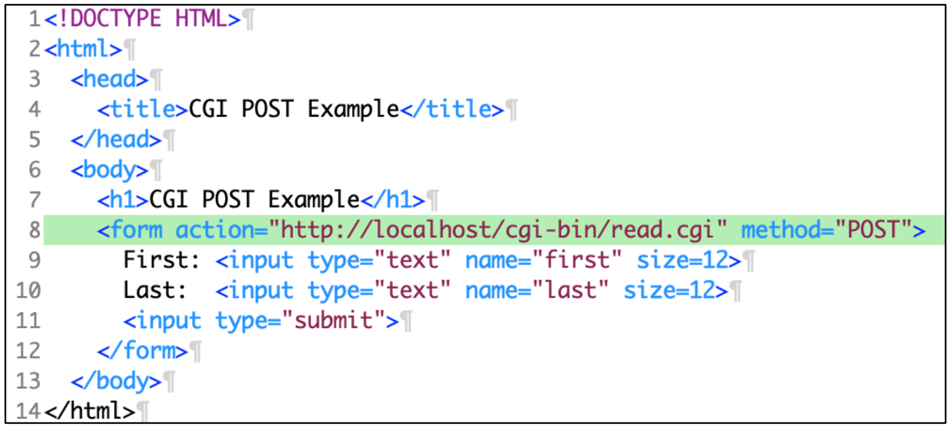
```
Syntax
      read [-ers] [-a aname] [-p prompt] [-t timeout]
             [-n nchars] [-d delim] [name...]
```

IFS = **internal field separator** (for arguments)
default: space newline tab
set it to = and & because
`first=String1&last=String2`

`set -- String`
-- option: positional parameters **`$1,$2`**,..etc. are set after splitting *String*

Sending information using **POST**

- Client-side code:



```html
<!DOCTYPE HTML>
<html>
  <head>
    <title>CGI POST Example</title>
  </head>
  <body>
    <h1>CGI POST Example</h1>
    <form action="http://localhost/cgi-bin/read.cgi" method="POST">
      First: <input type="text" name="first" size=12>
      Last:  <input type="text" name="last" size=12>
      <input type="submit">
    </form>
  </body>
</html>
```

`form-post.html`

place in

`macOS: /Library/Webserver/Documents/`

`Ubuntu: /var/www/html`

Server-side code:

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
read input
origIFS=$IFS
IFS='=&'
set -- $input
IFS=$origIFS
echo "Form data \$2:<$2> \$4:<$4>"
```

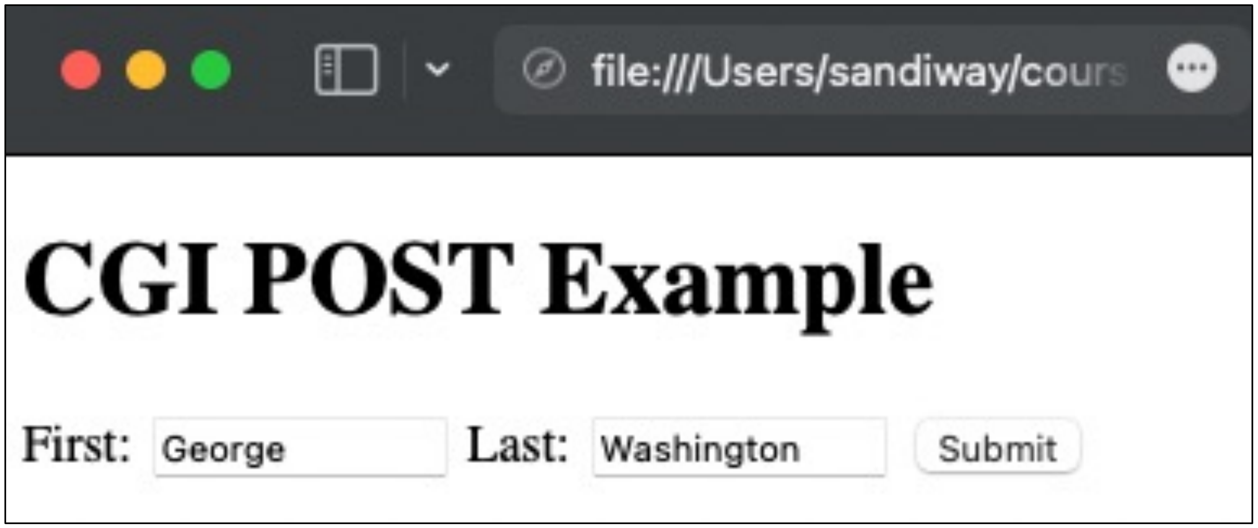
`read.cgi`

place in

`/Library/Webserver/CGI-Executables/`

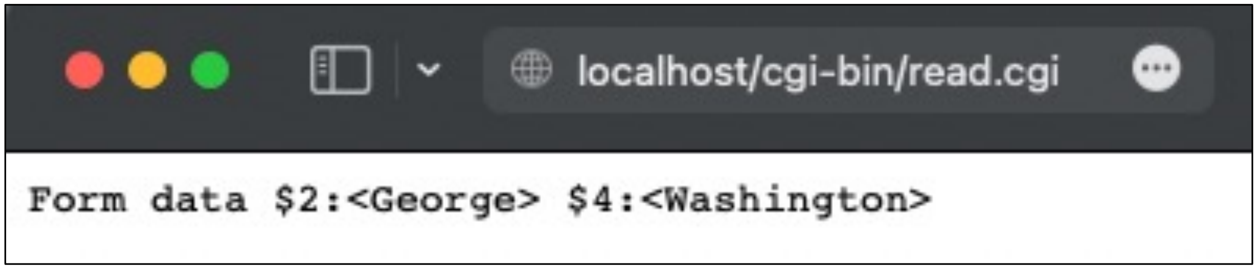
Ubuntu: `/usr/lib/cgi-bin`

Sending information using POST



CGI POST Example

First: George Last: Washington Submit

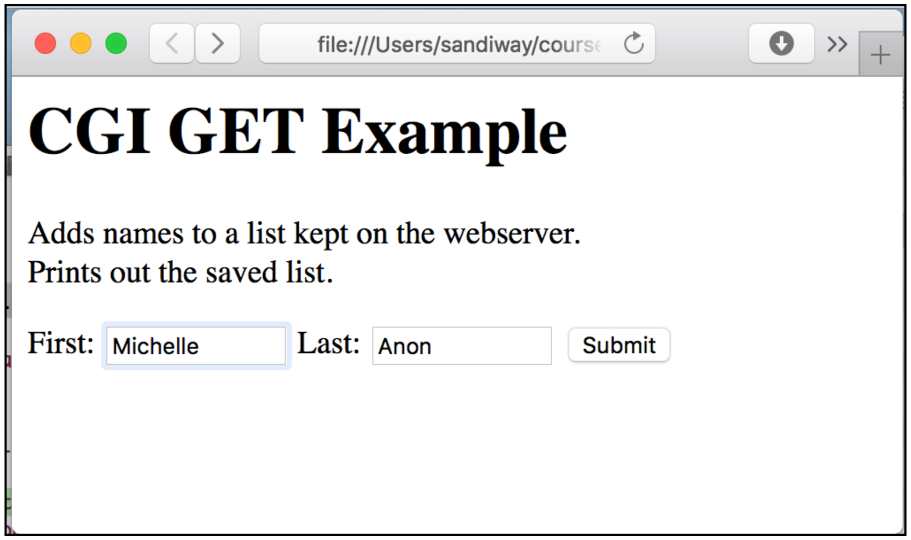


```
Form data $2:<George> $4:<Washington>
```

```
first=George&last=Washington
$1    $2       $3    $4
```

Example: adding names to a list on a server

- *Normally, you'd set up relational database software,e.g. mysql, on the webserver*


file:///Users/sandiway/course

CGI GET Example

Adds names to a list kept on the webserver.
Prints out the saved list.

First: Michelle Last: Anon Submit

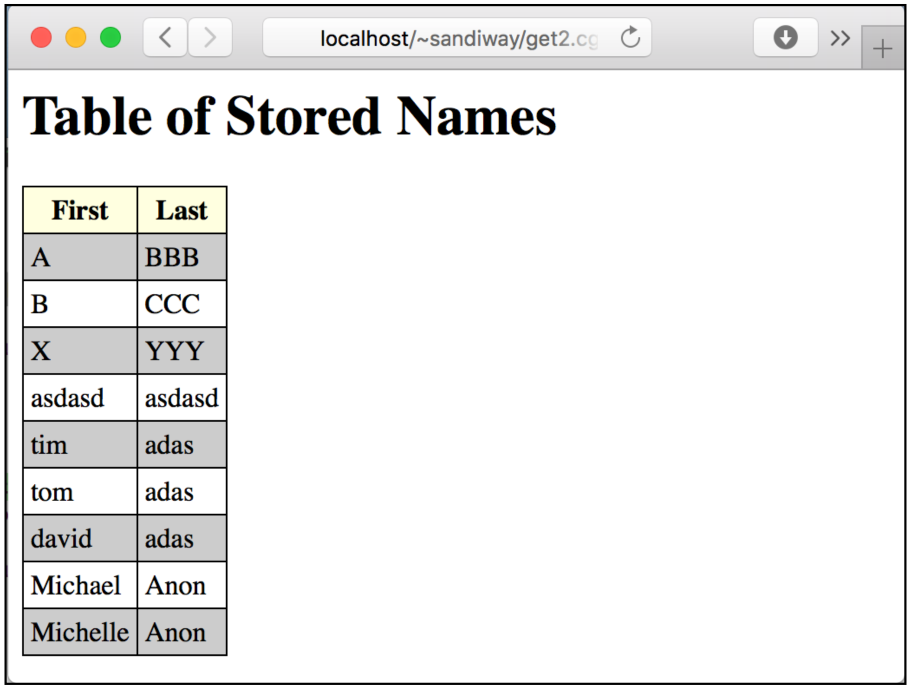

localhost/~sandiway/get2.cg

Table of Stored Names

First	Last
A	BBB
B	CCC
X	YYY
asdasd	asdasd
tim	adas
tom	adas
david	adas
Michael	Anon
Michelle	Anon

Example: adding names to a list on a server

Client side:
`addnames.html`

Server side:
`get2.cgi`
`names.txt`

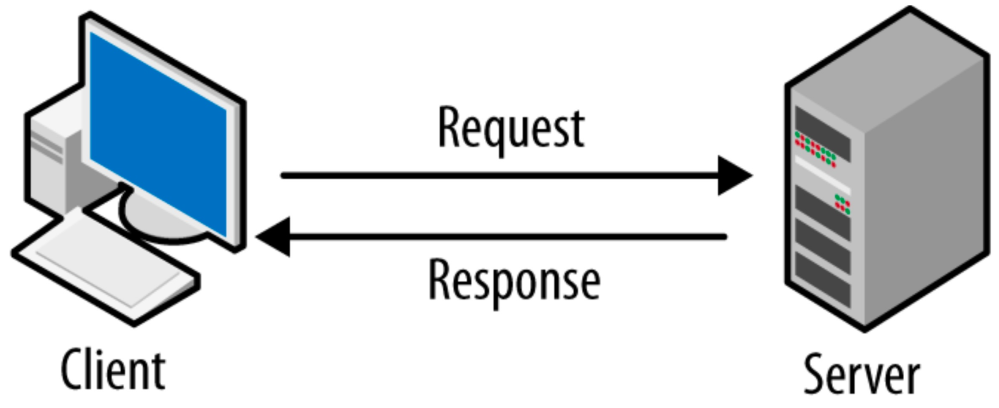


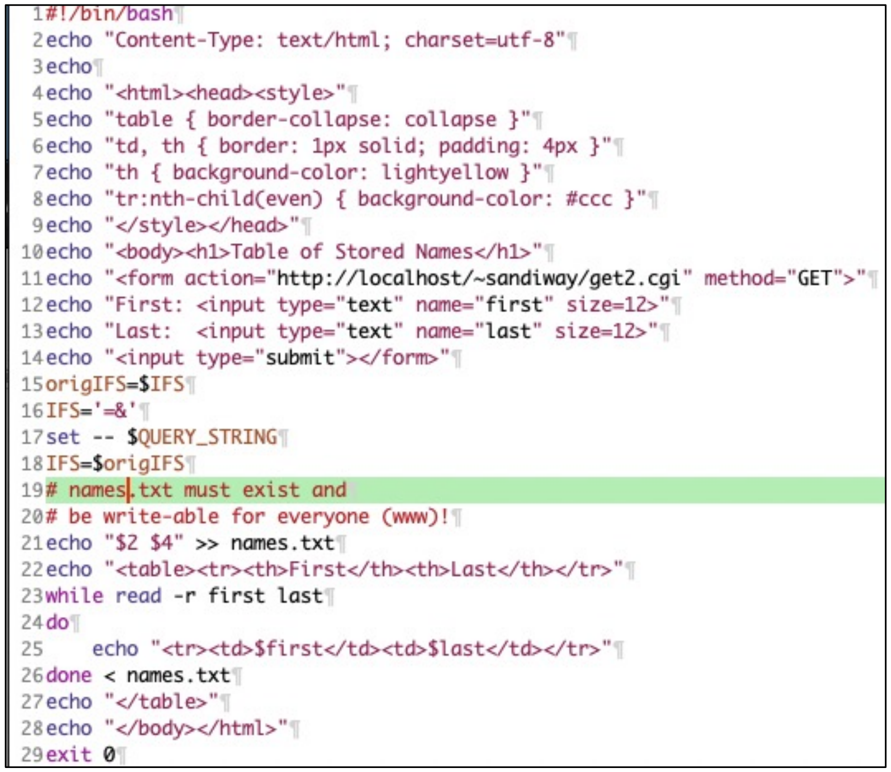
image from https://madooei.github.io/cs421_sp20_homepage/client-server-app/

A Note on Permissions

- Caution!
  - the Apache2.4 processes must be able to write to the directory in which `names.txt` is stored
  - this is not you, but typically the user is _www or www.
- On macOS:
  - `~/Sites` must have world write permission,
  - `chmod a+w ~/Sites`
- On Ubuntu:
  - `~/public_html` must have world write permission,
  - `chmod a+w ~/public_html`

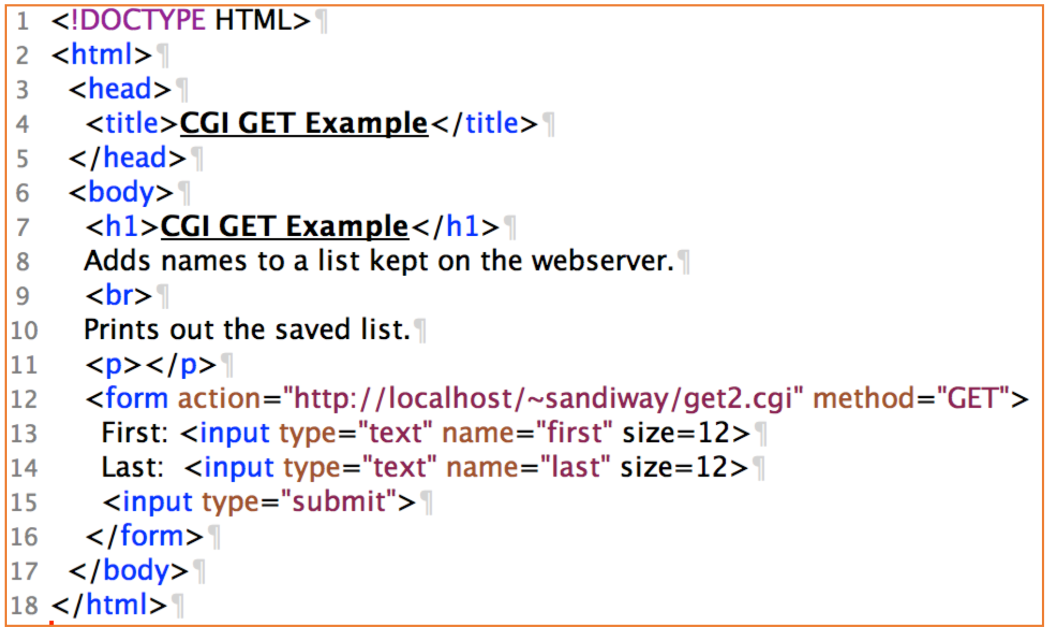
Example: adding names to a list on a server

Server-side: `get2.cgi`



```
#!/bin/bash
echo "Content-Type: text/html; charset=utf-8"
echo
echo "<html><head><style>"
echo "table { border-collapse: collapse }"
echo "td, th { border: 1px solid; padding: 4px }"
echo "th { background-color: lightyellow }"
echo "tr:nth-child(even) { background-color: #ccc }"
echo "</style></head>"
echo "<body><h1>Table of Stored Names</h1>"
echo "<form action="http://localhost/~sandiway/get2.cgi" method="GET">"
echo "First: <input type="text" name="first" size=12>"
echo "Last:  <input type="text" name="last" size=12>"
echo "<input type="submit"></form>"
origIFS=$IFS
IFS='=&'
set -- $QUERY_STRING
IFS=$origIFS
# names.txt must exist and
# be write-able for everyone (www)!
echo "$2 $4" >> names.txt
echo "<table><tr><th>First</th><th>Last</th></tr>"
while read -r first last
do
    echo "<tr><td>$first</td><td>$last</td></tr>"
done < names.txt
echo "</table>"
echo "</body></html>"
exit 0
```

Client-side: `addnames.html`



```
<!DOCTYPE HTML>
<html>
  <head>
    <title>CGI GET Example</title>
  </head>
  <body>
    <h1>CGI GET Example</h1>
    Adds names to a list kept on the webserver.
    <br>
    Prints out the saved list.
    <p></p>
    <form action="http://localhost/~sandiway/get2.cgi" method="GET">
      First: <input type="text" name="first" size=12>
      Last:  <input type="text" name="last" size=12>
      <input type="submit">
    </form>
  </body>
</html>
```

Create database file: in ~/Sites

```
touch names.txt
chmod a+w names.txt
```

(*create an empty file*)

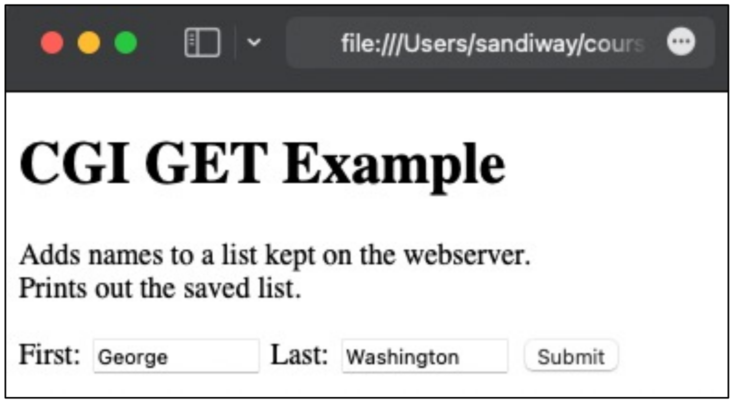
Example: adding names to a list on a server

- Terminal (in ~/Sites):

```
Sites$ touch names.txt
Sites$ chmod a+w names.txt
Sites$ ls –l names.txt
-rw-rw-rw-  1 sandiway  staff  0
Nov  8 19:46 names.txt
```

- After Submit is pressed:

```
Sites$ ls –l names.txt
-rw-rw-rw-  1 sandiway  staff  18
Nov  8 19:49 names.txt
(base) Sites$ cat names.txt
George Washington
```



CGI GET Example

Adds names to a list kept on the webserver.
Prints out the saved list.

First: George Last: Washington

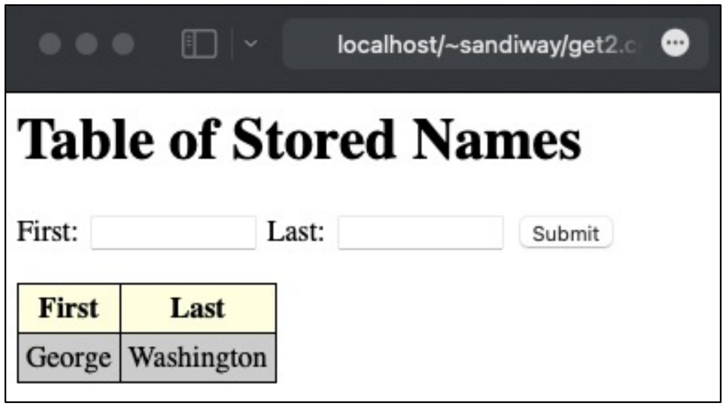


Table of Stored Names

First: Last:

First	Last
George	Washington

Example: adding names to a list on a server

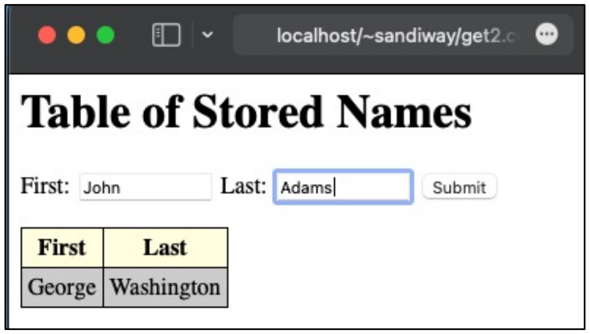


Table of Stored Names

First:	John	Last:	Adams

First	Last
George	Washington

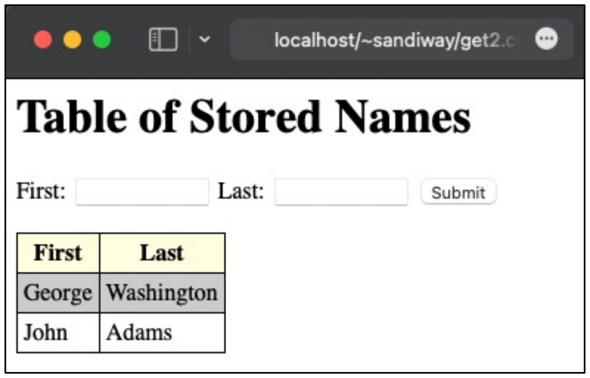


Table of Stored Names

First:		Last:	

First	Last
George	Washington
John	Adams

```
Sites$ ls -l names.txt
-rw-rw-rw-  1 sandiway  staff  29 Nov  8 19:53 names.txt
Sites$ cat names.txt
George Washington
John Adams
```

Example: adding names to a list on a server

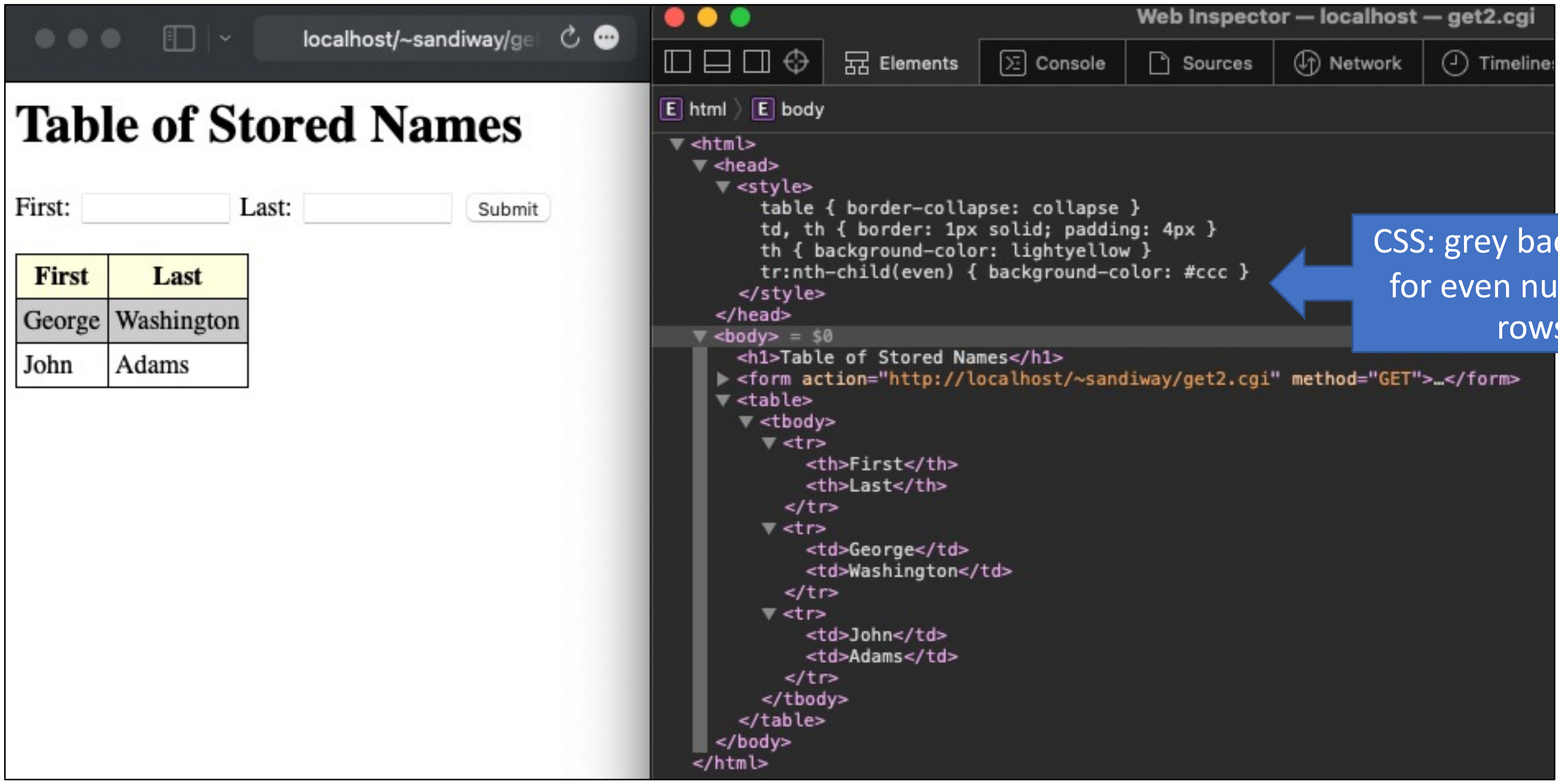


Table of Stored Names

First: Last: Submit

First	Last
George	Washington
John	Adams

```
<html>
  <head>
    <style>
      table { border-collapse: collapse }
      td, th { border: 1px solid; padding: 4px }
      th { background-color: lightyellow }
      tr:nth-child(even) { background-color: #ccc }
    </style>
  </head>
  <body> = $0
    <h1>Table of Stored Names</h1>
    <form action="http://localhost/~sandiway/get2.cgi" method="GET">…</form>
    <table>
      <tbody>
        <tr>
          <th>First</th>
          <th>Last</th>
        </tr>
        <tr>
          <td>George</td>
          <td>Washington</td>
        </tr>
        <tr>
          <td>John</td>
          <td>Adams</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

CSS: grey background for even numbered rows

Example: adding/deleting names on a server

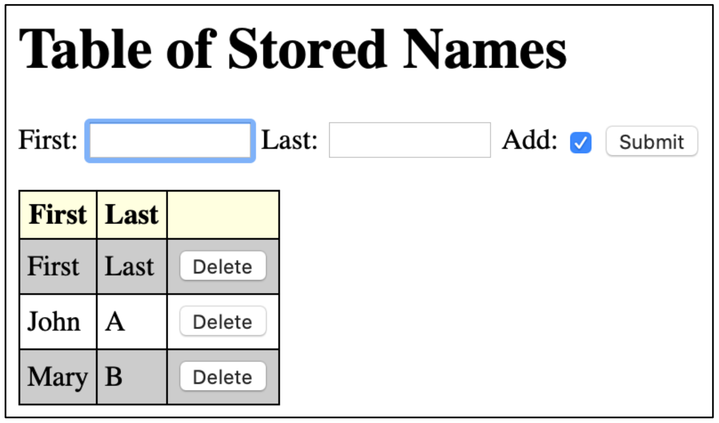


Table of Stored Names

First: [ ] Last: [ ] Add: ✓ Submit

First	Last	
First	Last	Delete
John	A	Delete
Mary	B	Delete

Client side:
`addnames3.html`

Server side:
`get3.cgi`
`names.txt`

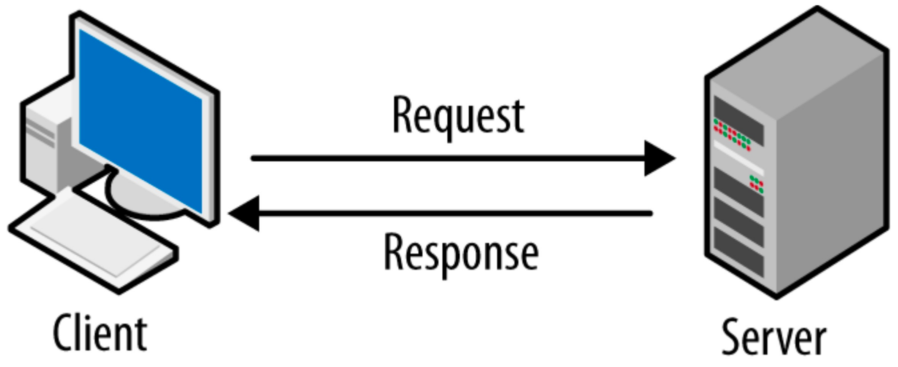
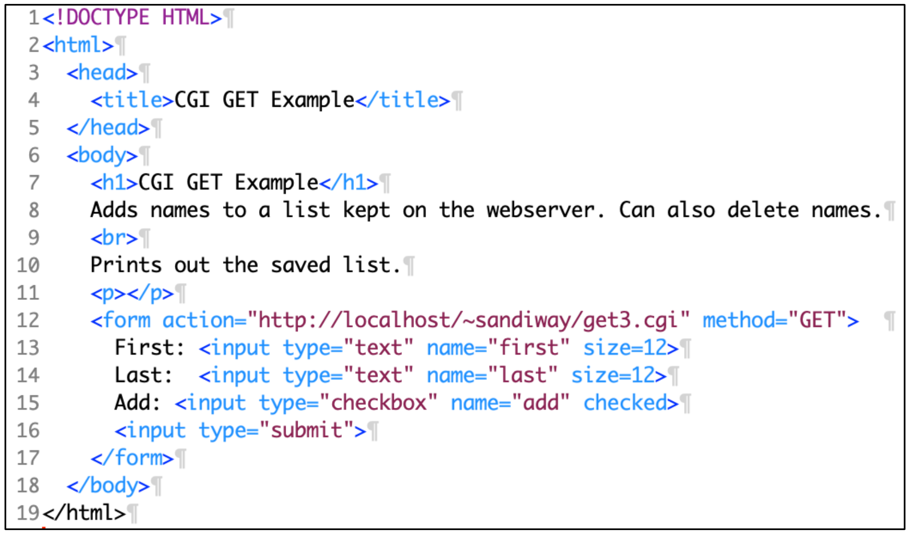
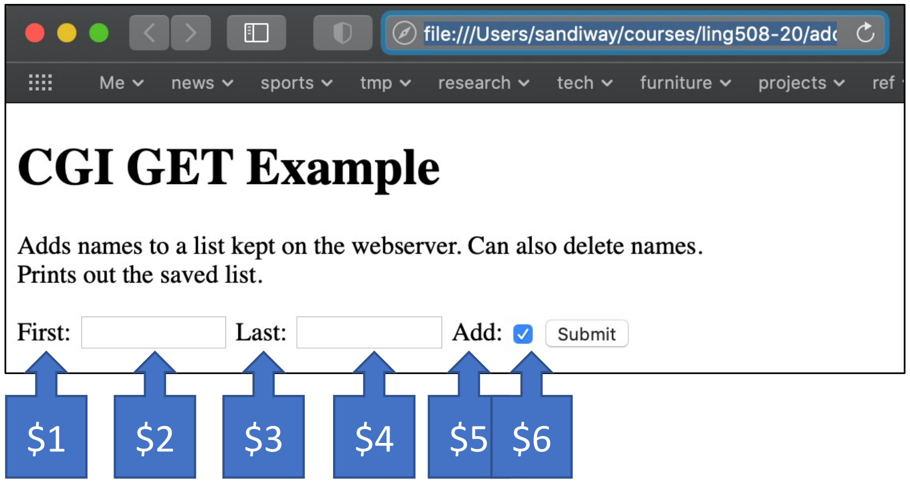


image from https://madooei.github.io/cs421_sp20_homepage/client-server-app/

addnames3.html



```
<!DOCTYPE HTML>
<html>
  <head>
    <title>CGI GET Example</title>
  </head>
  <body>
    <h1>CGI GET Example</h1>
    Adds names to a list kept on the webserver. Can also delete names.
    <br>
    Prints out the saved list.
    <p></p>
    <form action="http://localhost/~sandiway/get3.cgi" method="GET">
      First: <input type="text" name="first" size=12>
      Last:  <input type="text" name="last" size=12>
      Add: <input type="checkbox" name="add" checked>
      <input type="submit">
    </form>
  </body>
</html>
```



CGI GET Example

Adds names to a list kept on the webserver. Can also delete names.
Prints out the saved list.

First: Last: Add: ✓

Note: actually $1, $3 and $5 are names "first", "last" and "add"

get3.cgi

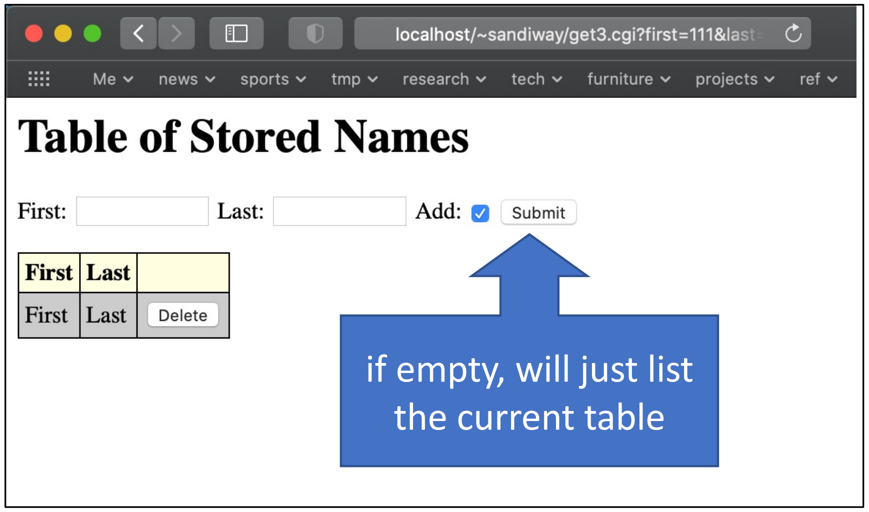
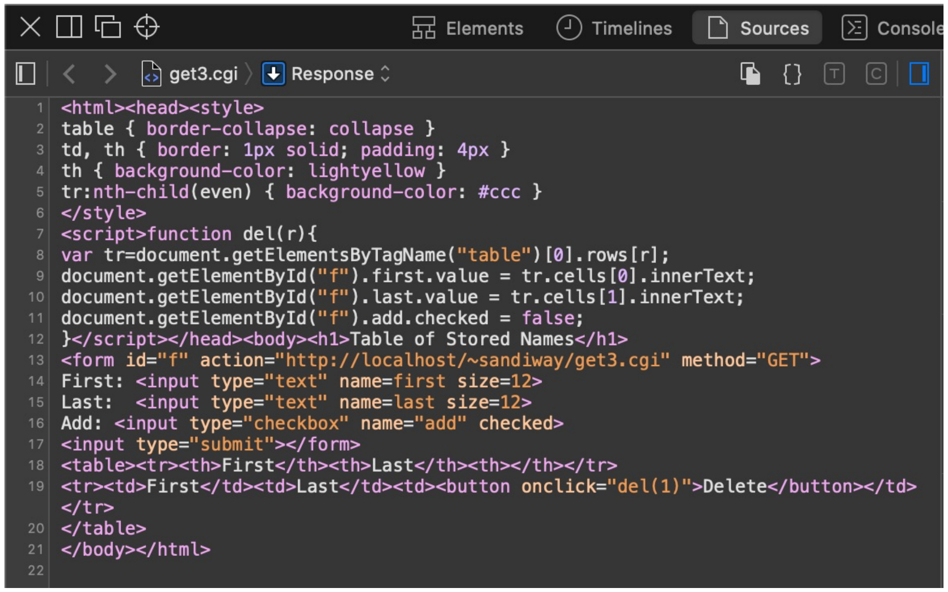

localhost/~sandiway/get3.cgi?first=111&last=
Me news sports tmp research tech furniture projects ref

Table of Stored Names

First: Last: Add: ✓ Submit

First	Last	
First	Last	Delete

if empty, will just list the current table



```
<html><head><style>
table { border-collapse: collapse }
td, th { border: 1px solid; padding: 4px }
th { background-color: lightyellow }
tr:nth-child(even) { background-color: #ccc }
</style>
<script>function del(r){
var tr=document.getElementsByTagName("table")[0].rows[r];
document.getElementById("f").first.value = tr.cells[0].innerText;
document.getElementById("f").last.value = tr.cells[1].innerText;
document.getElementById("f").add.checked = false;
}</script></head><body><h1>Table of Stored Names</h1>
<form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">
First: <input type="text" name=first size=12>
Last:  <input type="text" name=last size=12>
Add: <input type="checkbox" name="add" checked>
<input type="submit"></form>
<table><tr><th>First</th><th>Last</th><th></th></tr>
<tr><td>First</td><td>Last</td><td><button onclick="del(1)">Delete</button></td>
</tr>
</table>
</body></html>
```

Note: action is `http://localhost/~sandiway/get3.cgi`

get3.cgi

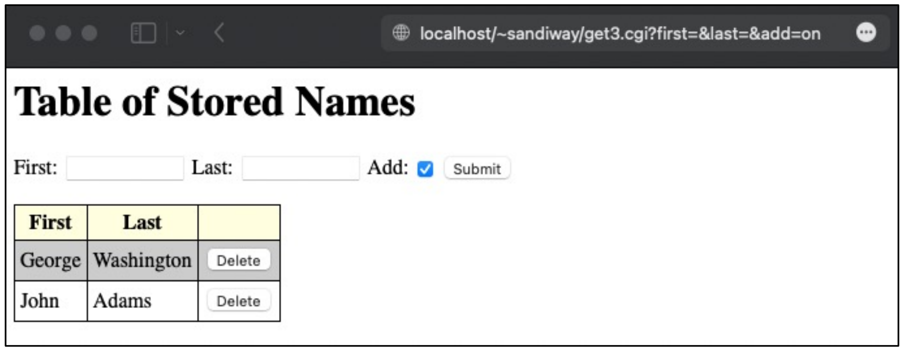


Table of Stored Names

First: ___ Last: ___ Add: [✓] Submit

First	Last	
George	Washington	Delete
John	Adams	Delete

- Current state of names.txt

get3.cgi

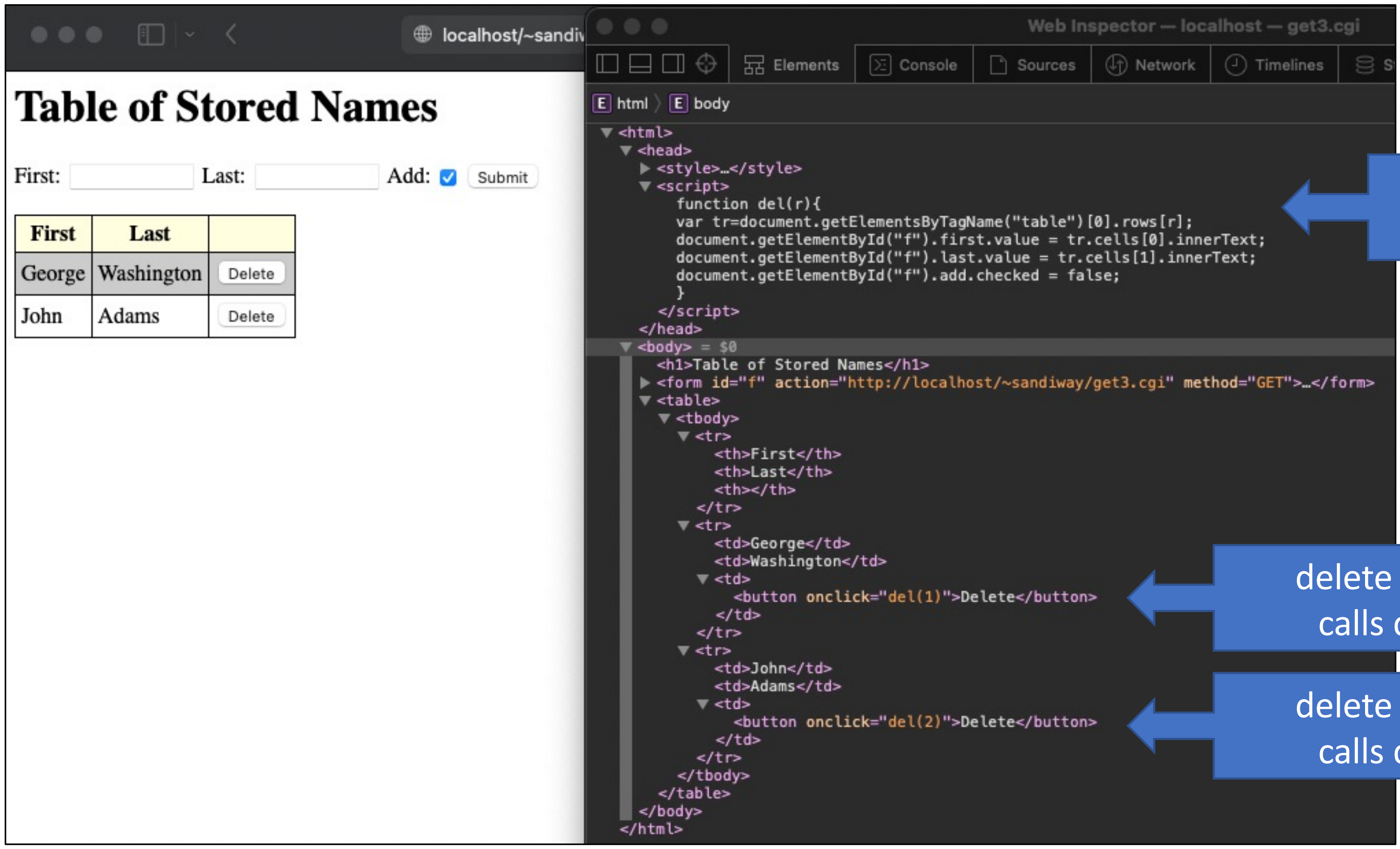


Table of Stored Names

First	Last	
George	Washington	Delete
John	Adams	Delete

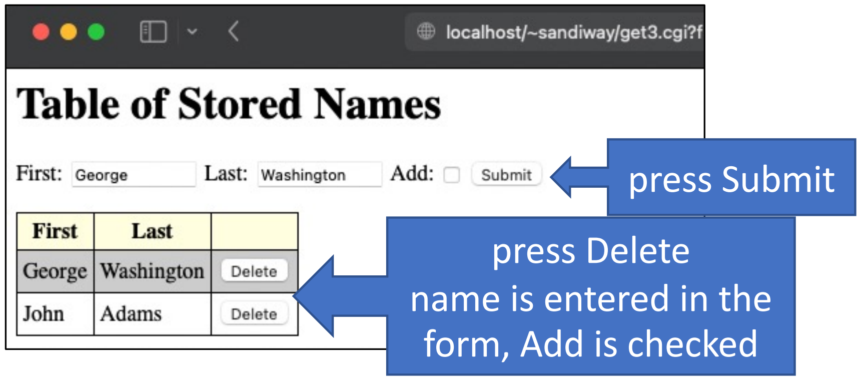
```
<html>
  <head>
    <style>…</style>
    <script>
      function del(r){
      var tr=document.getElementsByTagName("table")[0].rows[r];
      document.getElementById("f").first.value = tr.cells[0].innerText;
      document.getElementById("f").last.value = tr.cells[1].innerText;
      document.getElementById("f").add.checked = false;
      }
    </script>
  </head>
  <body> = $0
    <h1>Table of Stored Names</h1>
    <form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">…</form>
    <table>
      <tbody>
        <tr>
          <th>First</th>
          <th>Last</th>
          <th></th>
        </tr>
        <tr>
          <td>George</td>
          <td>Washington</td>
          <td>
            <button onclick="del(1)">Delete</button>
          </td>
        </tr>
        <tr>
          <td>John</td>
          <td>Adams</td>
          <td>
            <button onclick="del(2)">Delete</button>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

del(row) function

delete button calls del(1)

delete button calls del(2)

get3.cgi

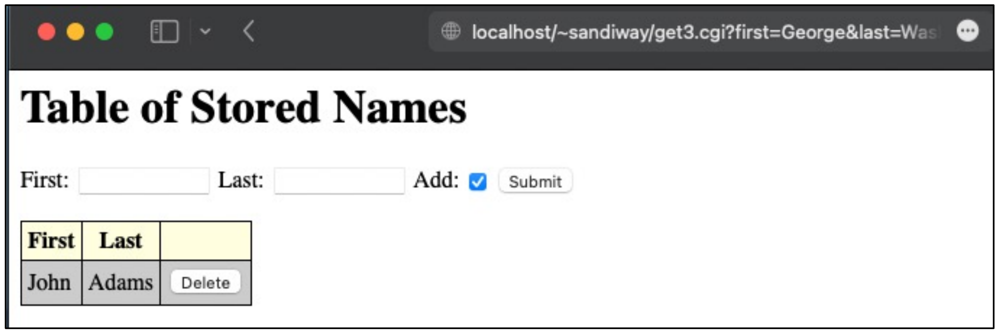


localhost/~sandiway/get3.cgi?f

Table of Stored Names

First: George Last: Washington Add: ☐ Submit

First	Last	
George	Washington	Delete
John	Adams	Delete



localhost/~sandiway/get3.cgi?first=George&last=Was

Table of Stored Names

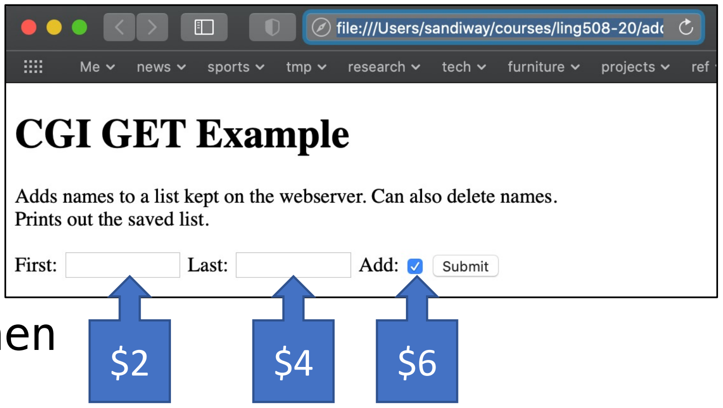
First: Last: Add: ☑ Submit

First	Last	
John	Adams	Delete

http://localhost/~sandiway/get3.cgi?first=George&last=Washington

get3.cgi

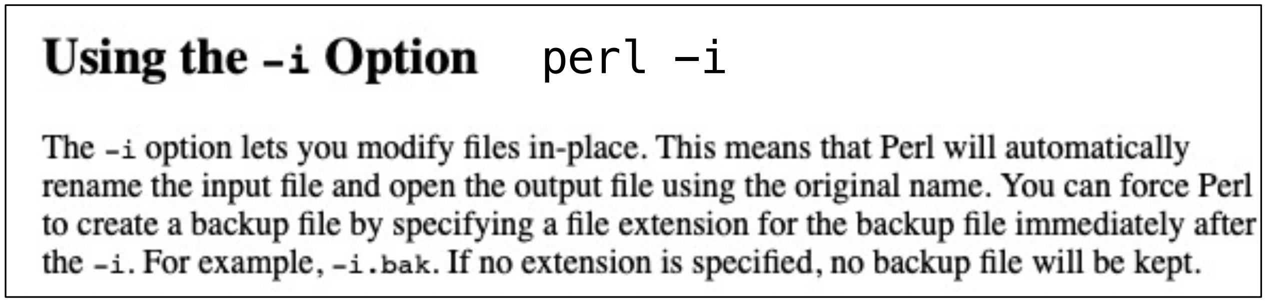
```
if [ "$6" = 'on' ]; then
    if [ "$2" != '' ] && [ "$4" != '' ]; then
      echo "$2 $4" >> names.txt
    fi
else
    perl -i -ne "print unless m/^$2 $4$/" names.txt
fi
```


file:///Users/sandiway/courses/ling508-20/ad

CGI GET Example

Adds names to a list kept on the webserver. Can also delete names.
Prints out the saved list.

First: Last: Add: ✓ Submit


Using the -i Option `perl -i`

The `-i` option lets you modify files in-place. This means that Perl will automatically rename the input file and open the output file using the original name. You can force Perl to create a backup file by specifying a file extension for the backup file immediately after the `-i`. For example, `-i.bak`. If no extension is specified, no backup file will be kept.

get3.cgi

```
echo "<table><tr><th>First</th><th>Last</th><th></th></tr>"
i=1
while read -r first last
do
    echo "<tr><td>$first</td><td>$last</td><td><button onclick=\"del($i)\">Delete</button></td></tr>"
    ((i++))
done < names.txt
echo "</table>"
echo "</body></html>"
```

-r do not allow backslashes to escape any characters

< names.txt means read from file names.txt

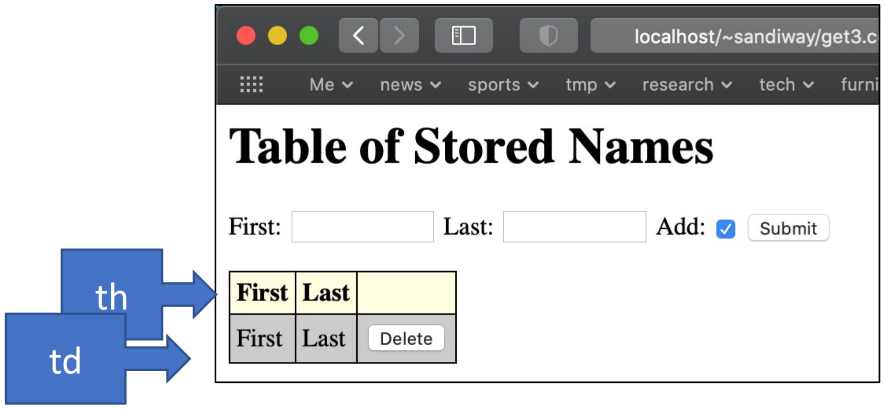
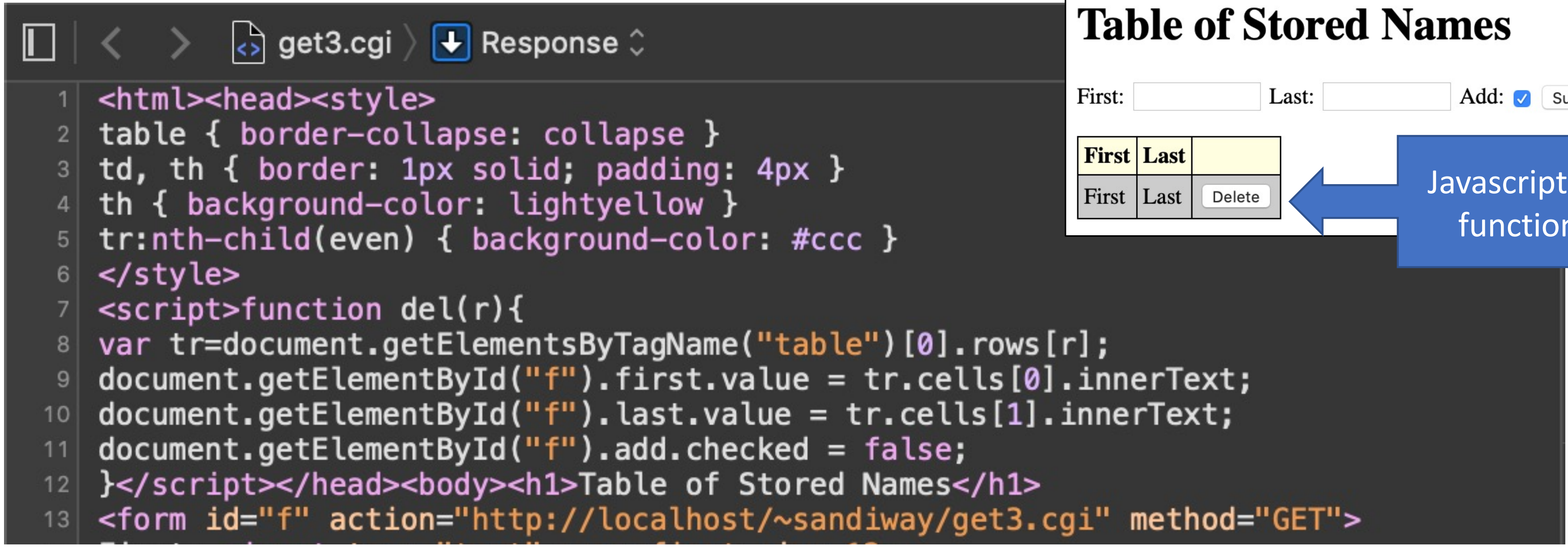


Table of Stored Names

First: Last: Add: ✓

First	Last	
First	Last	Delete

get3.cgi



```
<html><head><style>
table { border-collapse: collapse }
td, th { border: 1px solid; padding: 4px }
th { background-color: lightyellow }
tr:nth-child(even) { background-color: #ccc }
</style>
<script>function del(r){
var tr=document.getElementsByTagName("table")[0].rows[r];
document.getElementById("f").first.value = tr.cells[0].innerText;
document.getElementById("f").last.value = tr.cells[1].innerText;
document.getElementById("f").add.checked = false;
}</script></head><body><h1>Table of Stored Names</h1>
<form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">
```

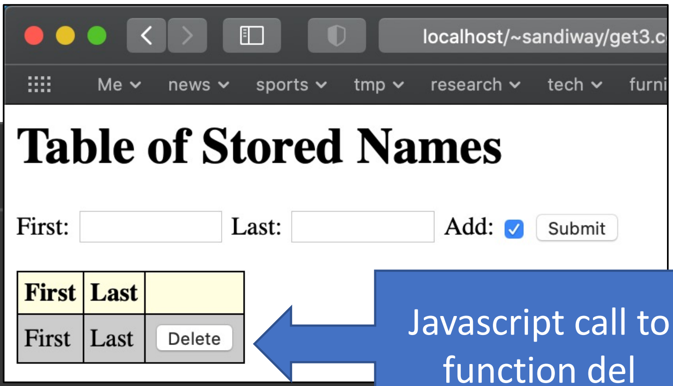
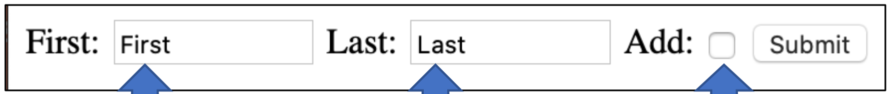


Table of Stored Names

First: Last: Add: ✓ Submit

First	Last	
First	Last	Delete

Javascript call to function del



First: First Last: Last Add: ☐ Submit

first.value

last.value

checked = false