JavaScript Validation con’t

JavaScript Reusability

- Functions are reusable blocks of code
- Any new function that you want to create should be in a separate JavaScript file
  - Ease of bug tracking
  - Reusing same functions across multiple pages
  - Code Readability
  - Manageable to push updates
  - Cached JavaScript files can speed up page loads
  - Creation of custom libraries

Recap: How to keep JavaScript in a separate

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="form_html_validation.css">
</head>
<body>

<form name="registration_form" onsubmit="validation_function()">
First name: <input type="text" name="fname" required pattern="[A-Za-z]+"><br>
Last name: <input type="text" name="lname" required><br>
Email: <input type="text" name="email" required><br>
Phone: <input type="text" name="phone" size=12 required pattern
<button type="submit" value="Submit">Submit</button>
<button type="reset" value="Reset">Reset</button>
<p id="ErrMsg"></p>
</form>

<script src="form_validation.js"></script>

</body>
</html>
</html>
```

```javascript
validation_function()
{
    var firstname = document.forms["registration_form"]["fname"];
    var lastname = document.forms["registration_form"]["lname"];
    var email = document.forms["registration_form"]["email"];
    var phone = document.forms["registration_form"]["phone"];
    if(!firstname.checkValidity())
    {
        firstname.focus();
        return false;
    }
    if(!lastname.checkValidity())
    {
        lastname.focus();
        return false;
    }
    if(!email.checkValidity())
    {
        email.focus();
        return false;
    }
    if(!phone.checkValidity())
    {
        phone.focus();
        return false;
    }
    return true;
```

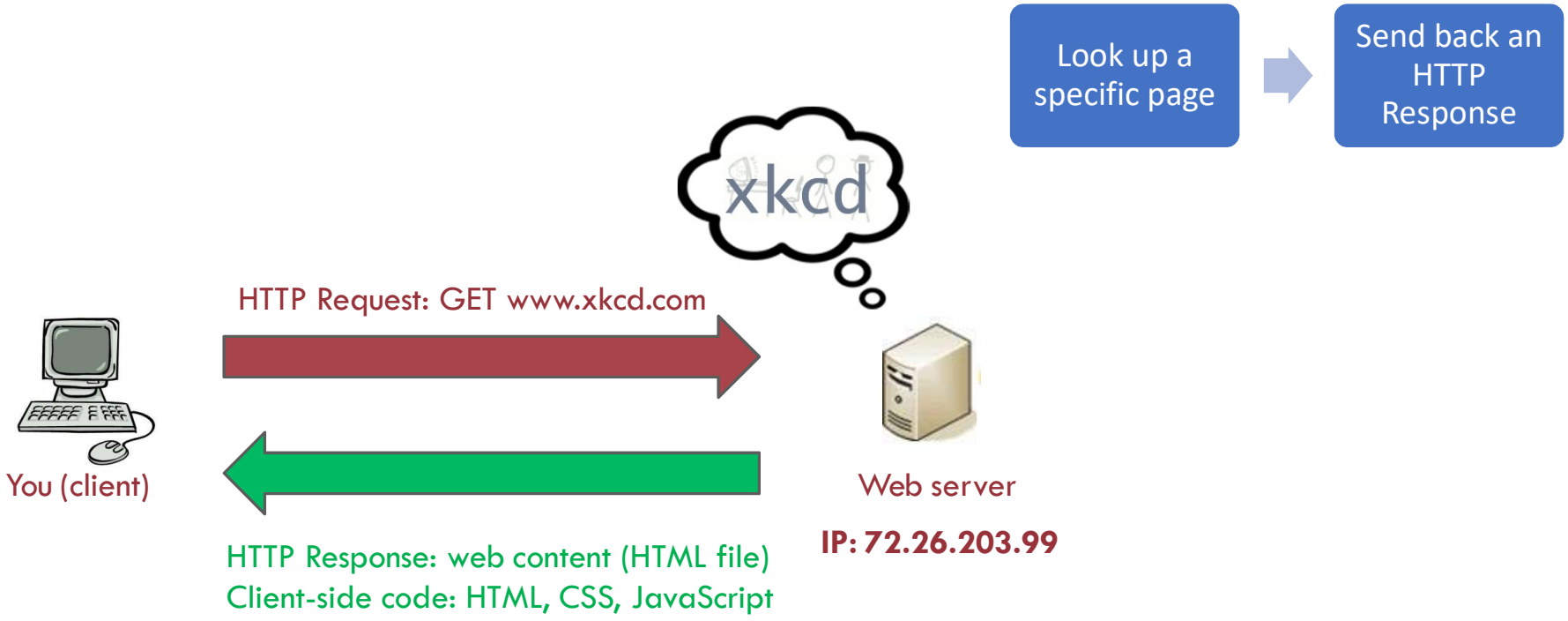
```
|{4}"><br>
```

Things to think about

- Use of parameters to pass information
  - Example
    - One set of JS Form Validation Functions can be used across all forms on a site as long as you are able to pass the form name as a parameter to the function
- Clearly and strictly defining input and output of each function
  - Try to ensure that there is minimal code repeatability
    - Copy paste errors
    - Fixing same problem across many functions
    - Testing and Debugging

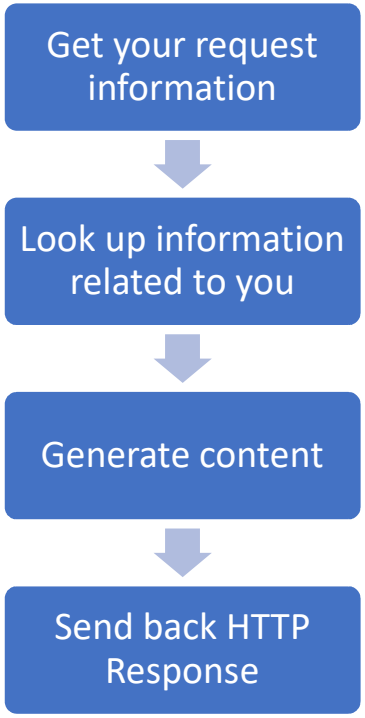
Server Side Programming

Request to a Static Page



Request to a Dynamic Page

- The server must respond dynamically if it needs to provide different client-side code depending on the situation
  - Date and time
  - Specifics of the user's request
  - Database contents – forms and authentication



HTTP Response Code Categories

- 1XX
  - informational
- 2XX
  - success
- 3XX
  - redirect
- 4XX
  - client error
- 5XX
  - server error

HTTP Response Common Codes

- 200 OK
  - request succeeded, resource is in message body
- 404 Not Found
  - resource does not exist
- 500 Server Error
  - general server error

Server Side Technologies

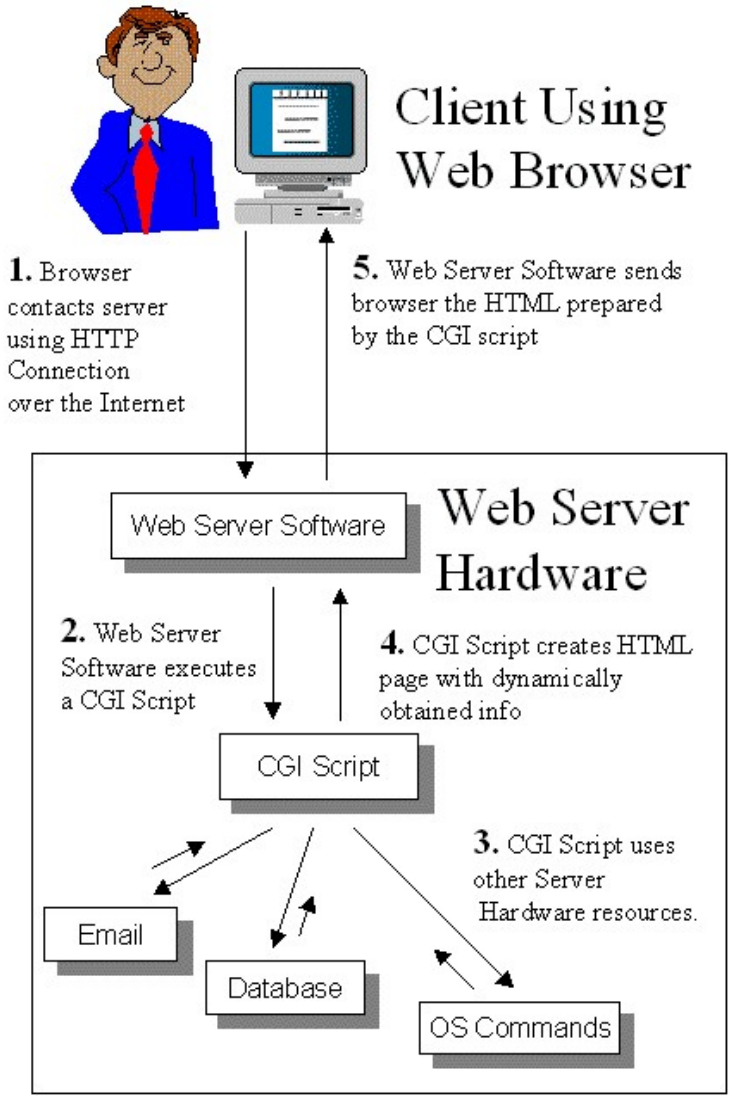
- PHP
- Perl
- .Net
- Ruby
- Java

Common Server Side Frameworks

- Symfony for PHP
- Django for Python
- Express for Node.js
- Ruby on Rails
- Grails for Java
- Catalyst/Mojolicious for Perl

Selection of Server Side Framework Factors

- Difficulty to learn
- Efficiency/Productivity
- Performance
- Caching
- Scalability (e.g., performance, load distribution)
- Web security (e.g., form sanitization)


Client Using
Web Browser
1. Browser contacts server using HTTP Connection over the Internet
5. Web Server Software sends browser the HTML prepared by the CGI script
Web Server Software
Web Server
Hardware
2. Web Server Software executes a CGI Script
4. CGI Script creates HTML page with dynamically obtained info
CGI Script
3. CGI Script uses other Server Hardware resources.
Email
Database
OS Commands

Example of a Form with a Perl Server Side Script Link

form.html

```html
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form id="form-id" action="/cgi-bin/form1.pl" method="post">
  First name:<br>
  <input type="text" name="firstname">
  <br>
  Last name:<br>
  <input type="text" name="lastname">
  <br><br>
  <button type="submit">Submit</button>
</form>

</body>

</html>
```

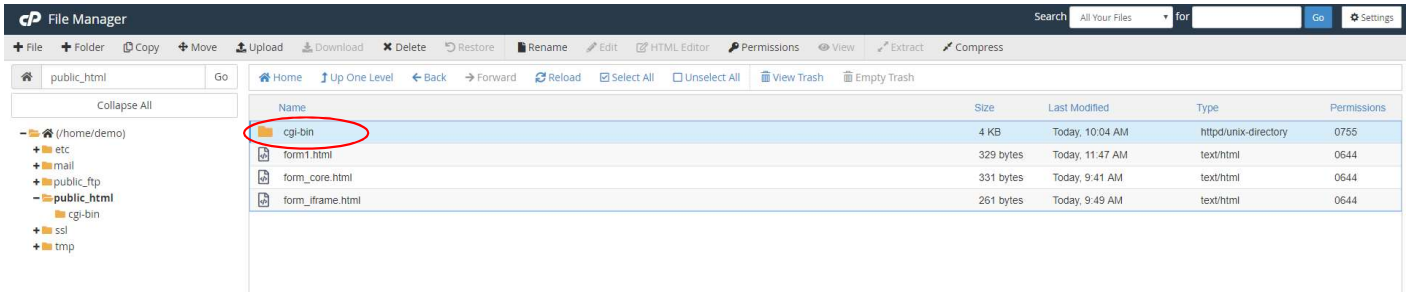
form1.pl

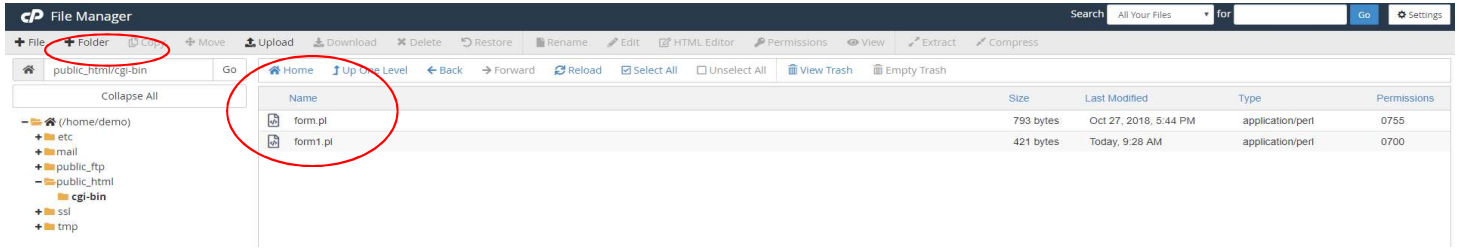
```perl
#!/usr/bin/perl -w
use strict;
use warnings;
use DBI;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);
print "Content-type: text/html\n\n";

##read form data
my $firstname = param('firstname');
my $lastname = param('lastname');

##display
print "<h1> Thank you for submitting the form</h1>";
print "<p> Your first name is ", $firstname,"</p>";
print "<p> Your last name is ", $lastname,"</p>";
```

Take a closer look at the Perl script

```
#!/usr/bin/perl -w
use strict;
## disables certain Perl expressions that could behave
##unexpectedly or are difficult to debug, turning them into errors.
use warnings;
## enables the use of additional/optional warnings
use DBI;
##The DBI is a database access module for the Perl programming language.
##It defines a set of methods, variables, and conventions that provide a
##consistent database interface, independent of the actual database being used.
use CGI qw(:standard);
##CGI is a Common Gateway Interface module that allows to process and prepare
##HTTP requests and responses. qw(:standard) indicates that the standard set of
##functions is being imported
use CGI::Carp qw(fatalsToBrowser);
##CGI::Carp is a module for giving better error messages

print "Content-type: text/html\n\n";
##gets sent back as part of the HTTP header to specify the type of content


##read form data
my $firstname = param('firstname');
my $lastname = param('lastname');
##param function/routine gives the value of the form field
##whose name is passed as a parameter

## a new variable is declared as my $variablename;
## later that variable may by called by just using $variablename (without the "my")


##display
print "<h1> Thank you for submitting the form</h1>";
print "<p> Your first name is ", $firstname,"</p>";
print "<p> Your last name is ", $lastname,"</p>";

##sends back newly generated content. Text goes into quotes.
##variables ahve to be seperated by commas (similar to + in javascript).
```

Placement of the script



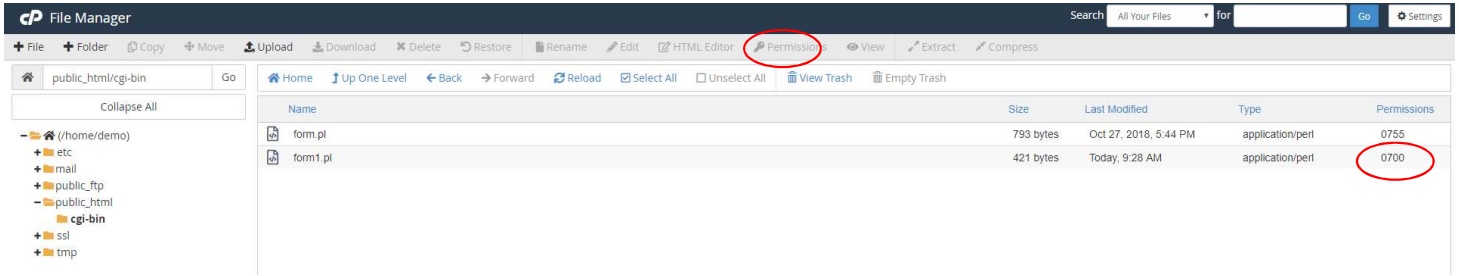


Permissions on files

- 3 actions can be performed on any file
  - Read
  - Write
  - Execute
- There are three types of users
  - Owner
  - Group
  - World

Value	Meaning
777	No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666	All users may read and write the file.
644	The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Permission of the script



Value	Meaning
777	No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	**The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.**
666	All users may read and write the file.
644	The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Information Flow

HTML Form gets user input → User submits HTML Form to a script → Script parses the information → Script generates content and sends it back → User receives information that the script generated

HTML Forms

First name:
John
Last name:
Does

Submit

Submit

```
##read form data
my $firstname = param('firstname');
my $lastname = param('lastname');
```

```
print "Content-type: text/html\n\n";
##display
print "<h1> Thank you for submitting the form</h1>";
print "<p> Your first name is ", $firstname,"</p>";
print "<p> Your last name is ", $lastname,"</p>";
```

Not secure | demo.heyuhnem.com/cgi-bin/form1.pl

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

Thank you for submitting the form

Your first name is John

Your last name is Does

DIY: Customize script to accept form we made last class

- Take form_extended.html
  - Make sure action will point to the correct script you make
  - Make sure method is post
- Take form1.pl and edit it in notepad++
- Add two additional variables (email and phone) and parse their values using param(). Make sure names of input fields in the form match those in perl file.
- Add two more print statements where HTML code is being generated to output the variables being read.
- Upload the html document to public_html
- Upload the perl script to public_html/cgi-bin and set the correct permissions
- Test!

Expected Output

First name: John
Last name: Doe
Email: doe@gmai.com
Phone: 1111111111
Submit Reset

Not secure | demo.heyuhnem.com/cgi-bin/form2.pl

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

Thank you for submitting the form

Your first name is John

Your last name is Doe

Your email is doe@gmai.com

Your phone is 1111111111

What happened?

- Form was filled out
- Form was submitted to the server
- A server-side script parsed the form field values
- A server side script responded with its own page with the parameter values displayed

HTML Iframes

- Iframe is used to display an HTML page within an HTML page
  - Embed third party media (YouTube)
  - Embed your own media
  - Embed code examples
  - Embed third party applets (payment forms)

Iframes and forms

- You can put your form into an iframe
- The basic syntax
  - <iframe src="*URL*"></iframe>

index.html

```html
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<iframe src="form_core.html"> </iframe>


<p>If you click the "Submit" button, the form-data will be
sent to a page".</p>

</body>

</html>
```

form_core.html

```html
<!DOCTYPE html>
<html>
<head></head>
<body>
<form id="form-id" action="/cgi-bin/form1.pl" method="post">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <button type="submit">Submit</button>
</form>
</body>
</html>
```

What does the page look like?

HTML Forms

First name:
Mickey
Last name:
Mouse

Submit

If you click the "Submit" button, the form-data will be sent to a page".

Iframe “prettification”

- Remove border
  - frameborder="0"
- Set sizing correctly
  - width="400" height="200"
  - By default iframe needs to have a specific size. It can be set once statically, or using CSS or JS it can be responsive. Example link.

DIY: Make an iframe

- Take form_extended.html and rename it form_iframe.html
  - Put the actual form into another html file, form_core.html
  - Using iframe link the form to the form_iframe.html
- Upload the html documents to public_html
- Make sure your server side script is still set correctly
- Test!

Output

HTML Forms

First name: John
Last name: Doe
Email: johndoe@gmail.com
Phone: 1112223333
Submit Reset

If you click the "Submit" button, the form-data will be sent to a page".

HTML Forms

Thank you for submitting the form

Your first name is John

Your last name is Doe

Your email is johndoe@gmail.com

Your phone is 1112223333

If you click the "Submit" button, the form-data will be sent to a page".