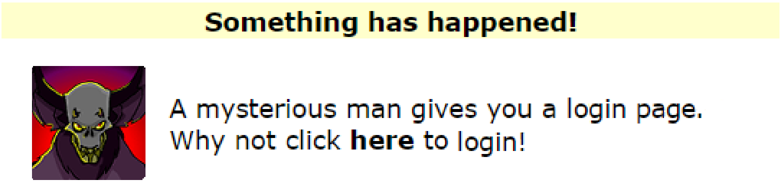
Browser Literacy Lessons From Neopets by Anne Hero



Something has happened!

A mysterious man gives you a login page.
Why not click **here** to login!

I will always look back fondly to the Flash Games Generation– those of us who were online from 2000-2007, who were violent towards stick figures, and who played dress up doll games, and who fed virtual pets– when children’s online entertainment belonged to browsers or the occasional offline CD-ROM game. Before the rise of the app **(**around 2008) [1].

The past decade has seen a universal shift from website-based media to application-based media on personal smart devices (like tablets and cell phones) [2]. Apps are designed to sustain a user’s attention while being very sleek- you can’t access an app’s source code[1] without **decompiling** it in **3rd party software** [3] or using **deobfuscation tools** [4]. In short, app source code isn’t designed to be looked at by anybody; why would they give their code away to nosy developers? To the average user, an app is a **black box**– they only know the **inputs and outputs**: whatever happens on the inside is locked away, and users are expected to trust whatever is on the inside. Another big difference between apps and websites is that apps can easily access **device capabilities**– why does the mobile game Peggle Blast need to make and manage phone calls, have access to contacts, photos, media, and files on your device [5] ? *“Who cares, I’m 14 years old and I want to play Peggle Blast. Allow.“*

[1] Both Android and Apple manufacturers design cell phones that prevent users from accessing **low-level** resources, installing certain applications, or messing with specific settings. The only way around these restrictions is to escalate your privileges via Jailbreaking (Apple) or Rooting (Android). Closed source iOS applications are compiled in such a way that makes it near impossible to see the full source code; on Android, it’s a bit easier to access and decompile applications, but it’s still more complicated than accessing the source code of websites.

One consequence of the mass migration from browser to app is that the average youth user is going to be more exploitable. There was a study by Zhao et. al on data collection of young children (around preschool age) that ended in 2020 [6] that concludes:

- Federal privacy rules (as outlined by **COPPA**) are largely not being enforced by apps
- Preschool age children have a large volume of identifying information being transmitted to **3rd party domains**, such as email address and location
- Older children, children with their own devices, and children from low-education households are at higher risk of privacy violations.

Another consequence of relying on apps is that kids are less browser-literate. [7]

> “ Mobile has killed technical competence. We now all carry around computers that pretend to be mobile phones or tablets. Most people don't even think of their phone as a computer. It's a device to get quick and easy access to Google…[it] allows us to take photos and post them to Facebook…[it] allows us to play games… It's a device that locks away the file system (or hides it from us). It's a device that only allows installation of sanitized apps through a regulated app store. It's a device whose hardware can't be upgraded or replaced and will be obsolete in a year or two.” – computing teacher Marc Scott, 2013

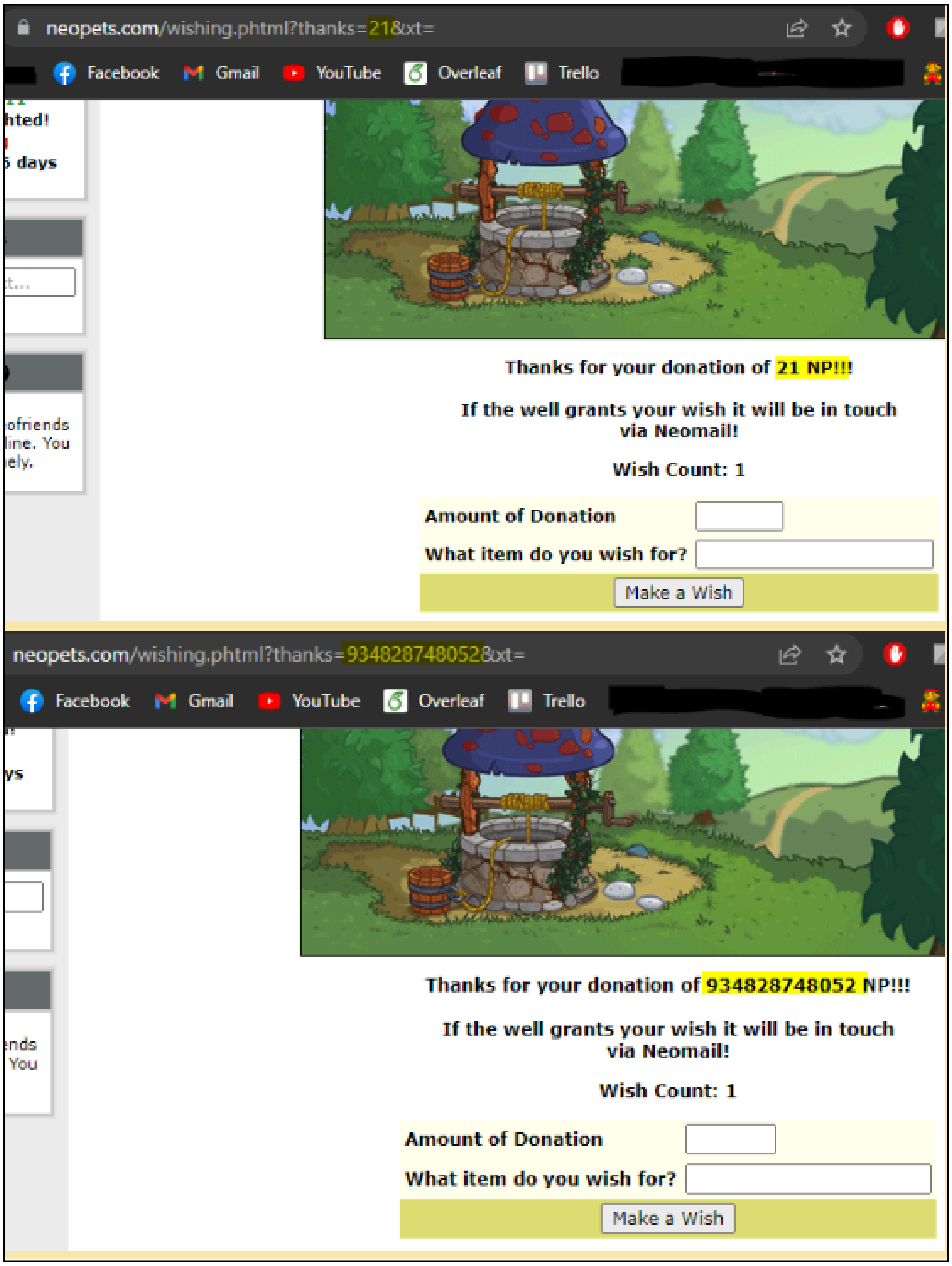
In contrast to apps, dealing with in-browser websites allows young users to fumble around in HTML and learn about cookies and cross-site requests and scripts and hash lists and redirection to other websites and the dangers of sharing your account with strangers and investigating suspicious looking login pages– sounds impressive, right? Explanations for each item listed are available in kid-friendly format on the Safety page of Jellyneo.net, arguably the most reputable and beloved Neopets fan site. [8]

Playing in browsers also means that kids are likely to tinker with the webpage- something you largely cannot do in the air tight confines of an app. In the browser, you may get the rare opportunity to see how **strings** and **values** are treated (or not treated), or what secrets lie in the **source code**. Users of all ages have been known to poke holes in **Neopets** (it is.. notoriously vulnerable).

Strings and Values

The Wishing Well is a place on Neopets where you can pay for the chance at winning a prize. If you donate 21 Neopoints, the following webpage would show the donated amount in the **URL’s parameter**: “thanks=21”. As a kid, I used to change the parameter to some absurd number and refresh the page just because I thought it was cool.

Changing the string in the URL doesn't actually redo the entire action, but it is a convincing illusion– furthermore, it taught me the importance of paying attention to information in URLs. Years later, I was able to bypass two-factor authentication by changing something along the lines of "authuser=0" to "authuser=1" in one of the Google Workspace tools (under which falls Gmail, Drive, Meet, Docs and more).



neopets.com/wishing.phtml?thanks=21&xt=

[redacted]

Thanks for your donation of 21 NP!!!

If the well grants your wish it will be in touch via Neomail!

Wish Count: 1

Amount of Donation

What item do you wish for?

neopets.com/wishing.phtml?thanks=934828748052&xt=

[redacted]

Thanks for your donation of 934828748052 NP!!!

If the well grants your wish it will be in touch via Neomail!

Wish Count: 1

Amount of Donation

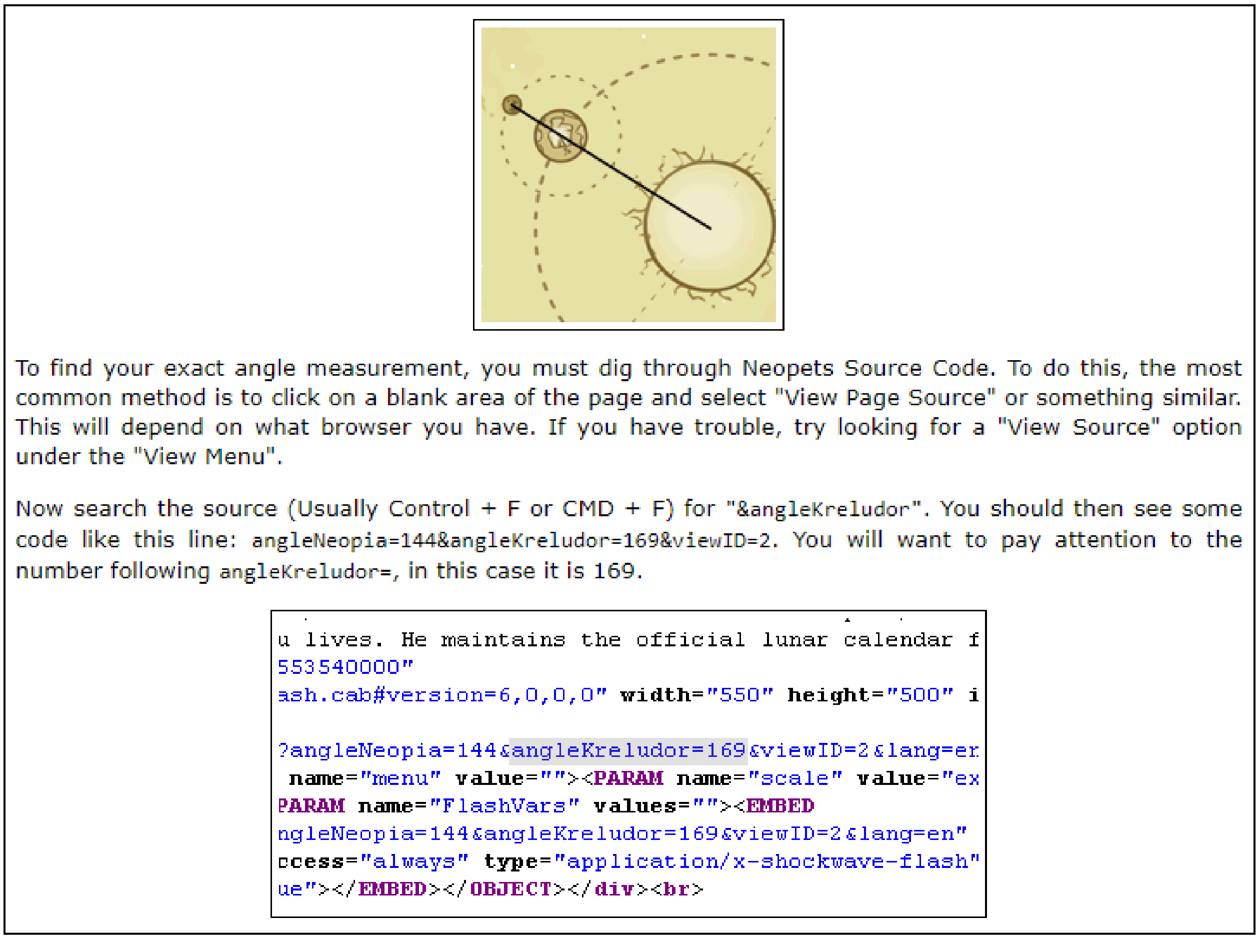
What item do you wish for?

I did not actually donate 934828748052 Neopoints!

Source Code

The Shenkuu Lunar Temple puzzle can be solved via the page's source code; if you find the value associated with a **variable** `&angleKreludor` in the source code, there is a value that corresponds mathematically to the solution. Similarly, you can look for information about a page in its source code, such as `article:published_time` and `article:modified_time`.

Knowing when a webpage was published and modified is useful whether you are citing it in a paper or just trying to contextualize its content; many things can change over time, such as a country's laws or technological advances. (You can try it out for yourself here! Although the original article does not have a date on its page, you can still see when it was published in its source code).



To find your exact angle measurement, you must dig through Neopets Source Code. To do this, the most common method is to click on a blank area of the page and select "View Page Source" or something similar. This will depend on what browser you have. If you have trouble, try looking for a "View Source" option under the "View Menu".

Now search the source (Usually Control + F or CMD + F) for "&angleKreludor". You should then see some code like this line: `angleNeopia=144&angleKreludor=169&viewID=2`. You will want to pay attention to the number following `angleKreludor=`, in this case it is 169.

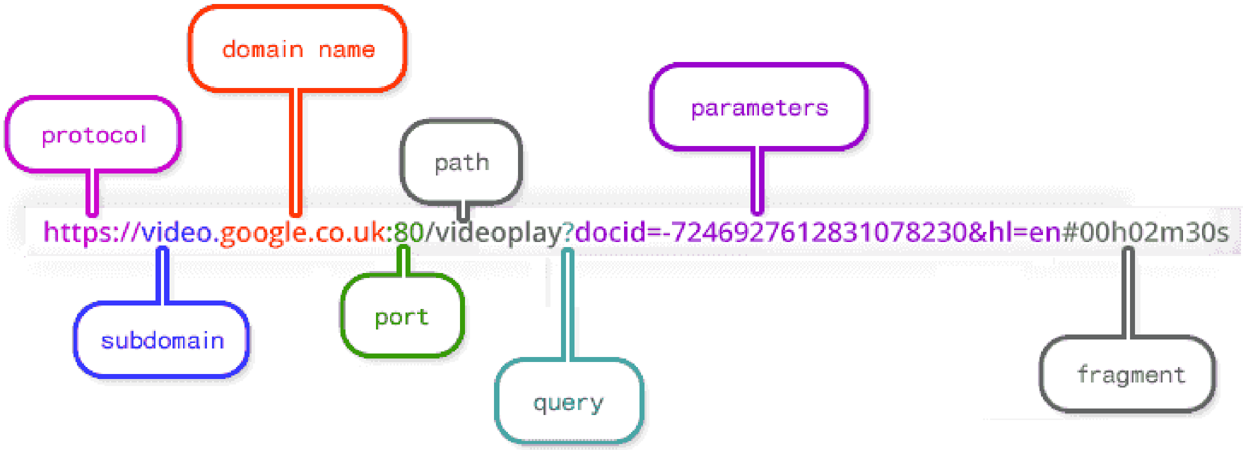
```
u lives. He maintains the official lunar calendar f
553540000"
ash.cab#version=6,0,0,0" width="550" height="500" i

?angleNeopia=144&angleKreludor=169&viewID=2&lang=er
 name="menu" value=""><PARAM name="scale" value="ex
PARAM name="FlashVars" values=""><EMBED
ngleNeopia=144&angleKreludor=169&viewID=2&lang=en"
ccess="always" type="application/x-shockwave-flash"
ue"></EMBED></OBJECT></div><br>
```

Here is an explanation of how to solve a Neopets puzzle using the page's source code.

Glossary

-**Source code**: (right click > View Page Source, or CTRL+U on windows) code behind a web page or application.
-**2FA** (*Two-factor authentication*): A method to strengthen security to your account by requiring 2 methods of verifying your identity. Many websites will text you a one time code.
-**HTML** (*Hypertext Markup Language*): the language for formatting web pages in browsers.
-**HTTP URL**: (*HypertextUniform Resource Locator*): the address of a webpage. [9]


domain name
protocol
path
parameters
https://video.google.co.uk:80/videoplay?docid=-7246927612831078230&hl=en#00h02m30s
subdomain
port
query
fragment

-**Low level:** close to the hardware.
-**Decompiling**: retrieving the source code.
-**Obfuscation**: intentionally making your code incomprehensible
-**Deobfuscation**: making code readable again
-**Black box**: we don't know what is happening or how it works on the inside, we just know its inputs and outputs.
-**Inputs and Outputs**: I give the app my location and a picture of my face (input), and it gives me contact with strangers (output). This is a general, non-technical definition.
-**COPPA** (*Children's Online Privacy Protection Act*): federal regulation that prohibits the collection and use of personal information of minors (under 13) through the internet.
-**3rd party domain**: basically there are companies we haven't heard of that know everything about us because they buy our data from websites and apps we use and sell us products back on the sites they got our data from..
-**Device capabilities**: your device’s camera, GPS, microphone, etc.
-**Strings**: text
-**Values**: a number that means either true (1) or false (0).
-**Neopets:** website aimed at children (although there are many adults on it) for taking care of a virtual pet, participating in a fictional stock market, and playing games