Intro Web Applications

Andrew Benson – ScottyLabs – CrashCourse F14

ScottyLabs

ScottyLabs is a student organization at Carnegie Mellon University. We organize educational events to help people learn how to make things, host events to give students the opportunity to work on projects outside of class, and develop applications and services for the campus community.

What are we making?
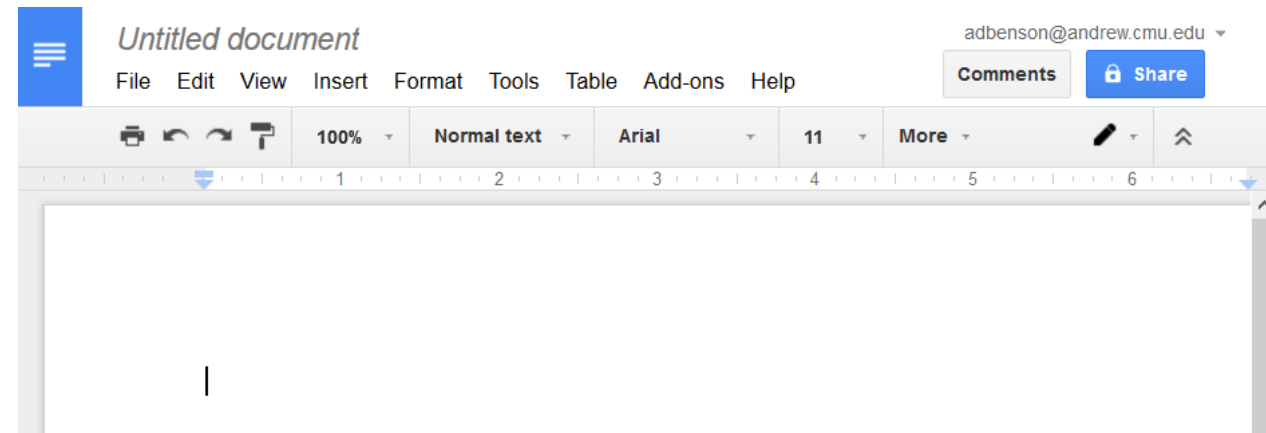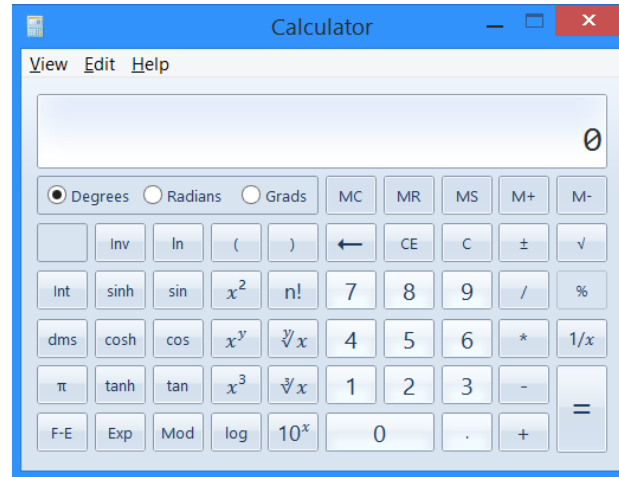
- crashdevblog.herokuapp.com

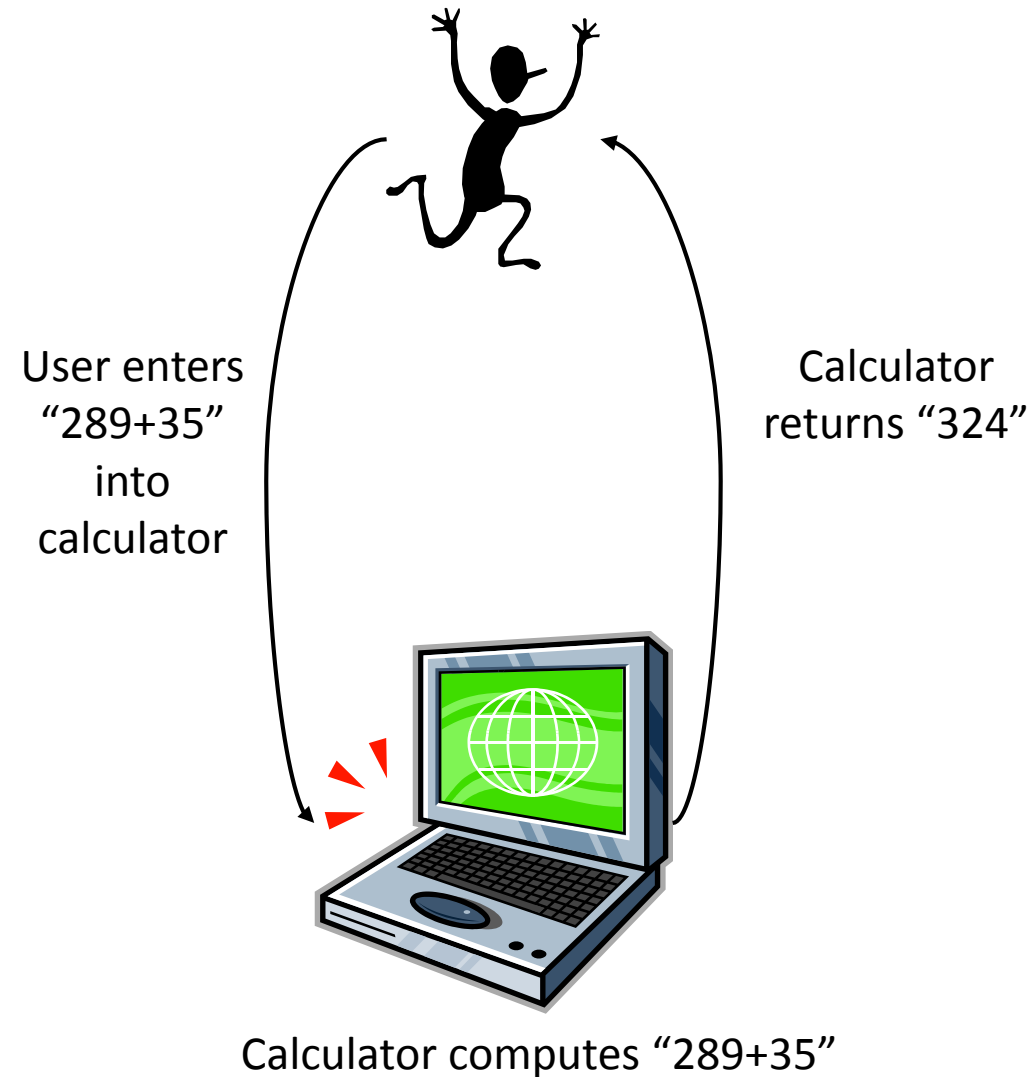# "Packaged Apps" vs Web Apps


```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

How a "packaged" application works

User enters "289+35" into calculator

Calculator returns "324"

Calculator computes "289+35"

How a web application works

Firefox renders the HTML as a webpage, where the user sees "324"

WolframAlpha Server

WolframAlpha responds through HTTP with some HTML that has the answer, "324"

User enters "289+35" into Firefox displaying WolframAlpha

Firefox contacts the WolframAlpha server through HTTP, asking what "289+35" is
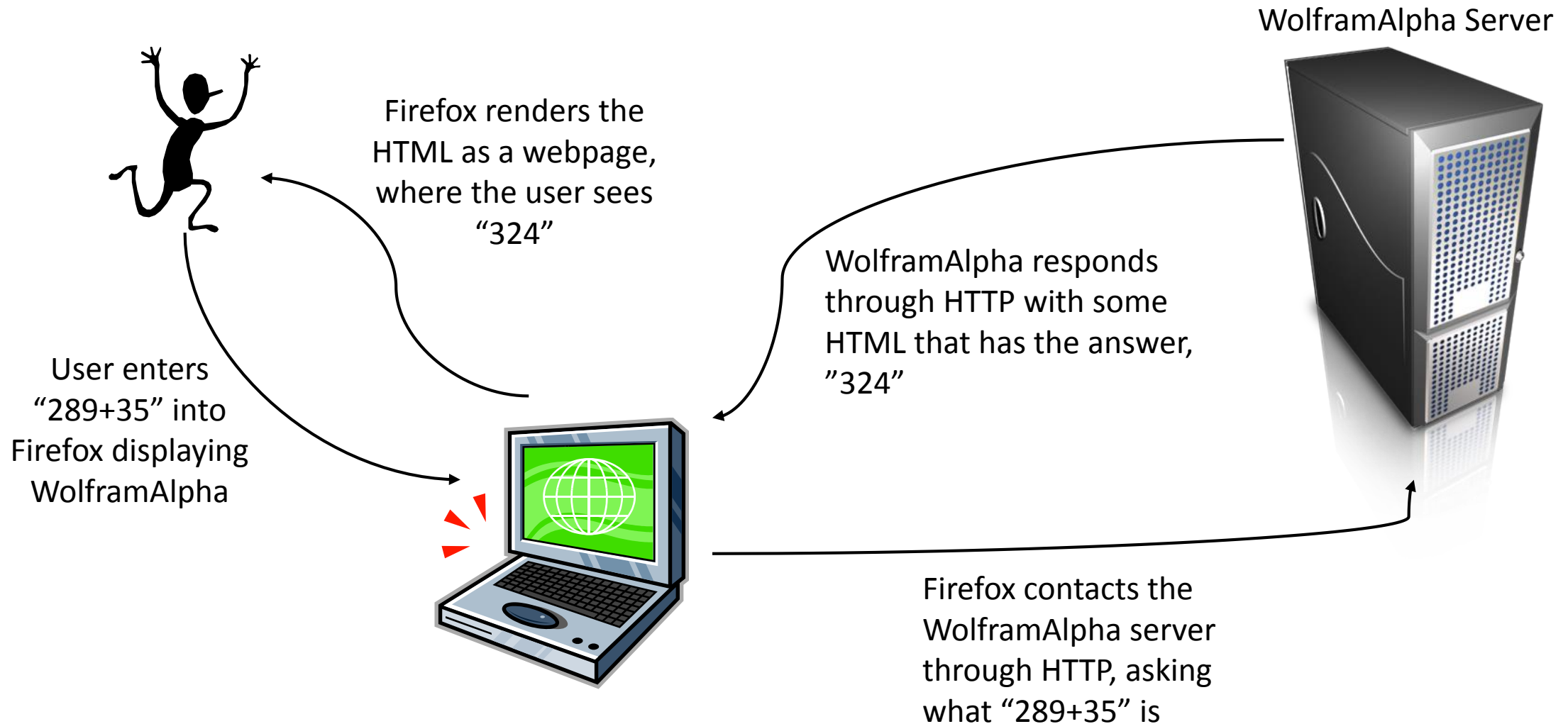
Making a server that says "Hello World"

SSH into Andrew Linux, and enter the following in a suitable directory:
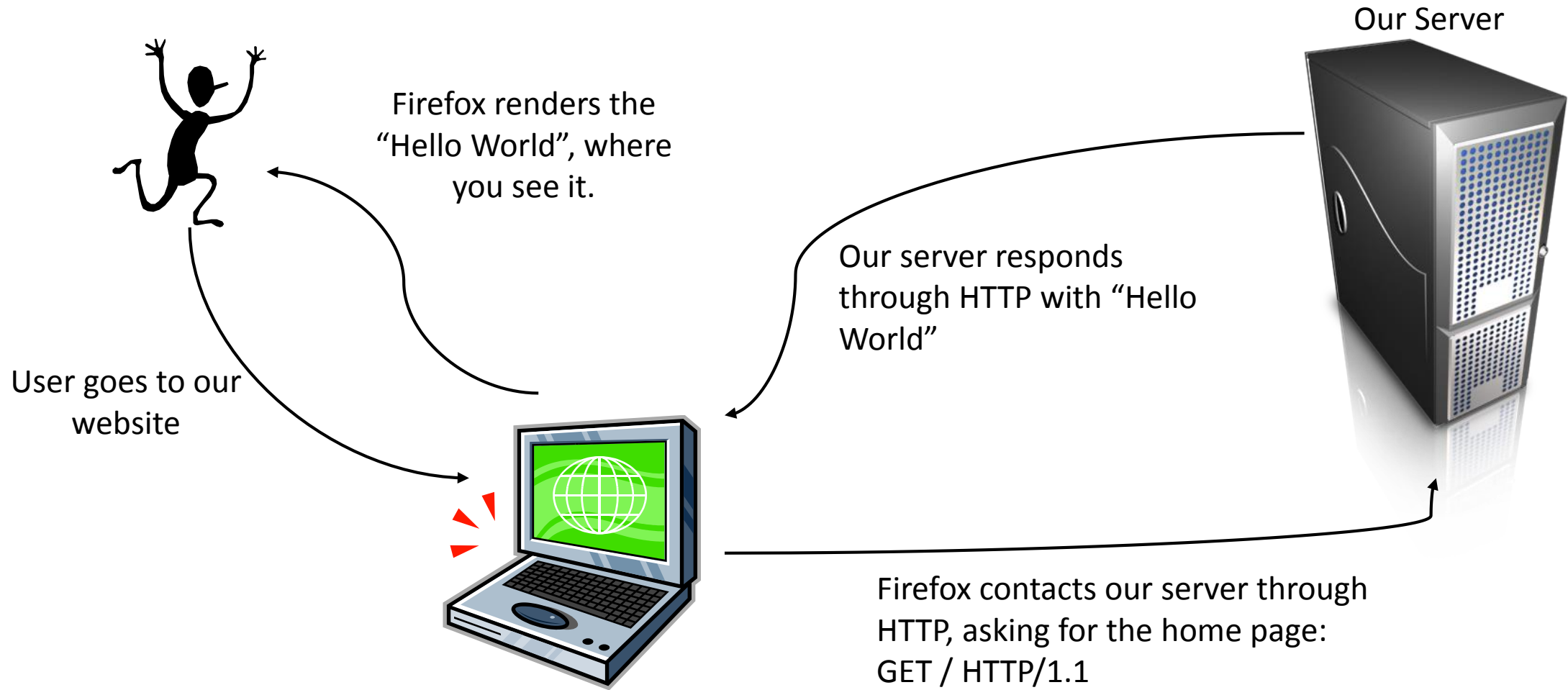
- git clone https://github.com/anbenson/crashdevblog-resources.git
- cd crashdevblog-resources
- curl -O https://pypi.python.org/packages/source/v/virtualenv/virtualenv-1.11.6.tar.gz
- tar -xzvf virtualenv-1.11.6.tar.gz
- python virtualenv-1.11.6/virtualenv.py venv
- source venv/bin/activate
- pip install flask
- hostname
- python main.py

Now open a web browser, and enter the result of typing "hostname" followed by ":8080". So if hostname returned "unix3.andrew.cmu.edu", I would enter "unix3.andrew.cmu.edu:8080". Some of you may have conflicting ports – I will come around to help you.

How our primitive server works

Firefox renders the "Hello World", where you see it.

Our server responds through HTTP with "Hello World"

User goes to our website

Our Server

Firefox contacts our server through HTTP, asking for the home page:
GET / HTTP/1.1
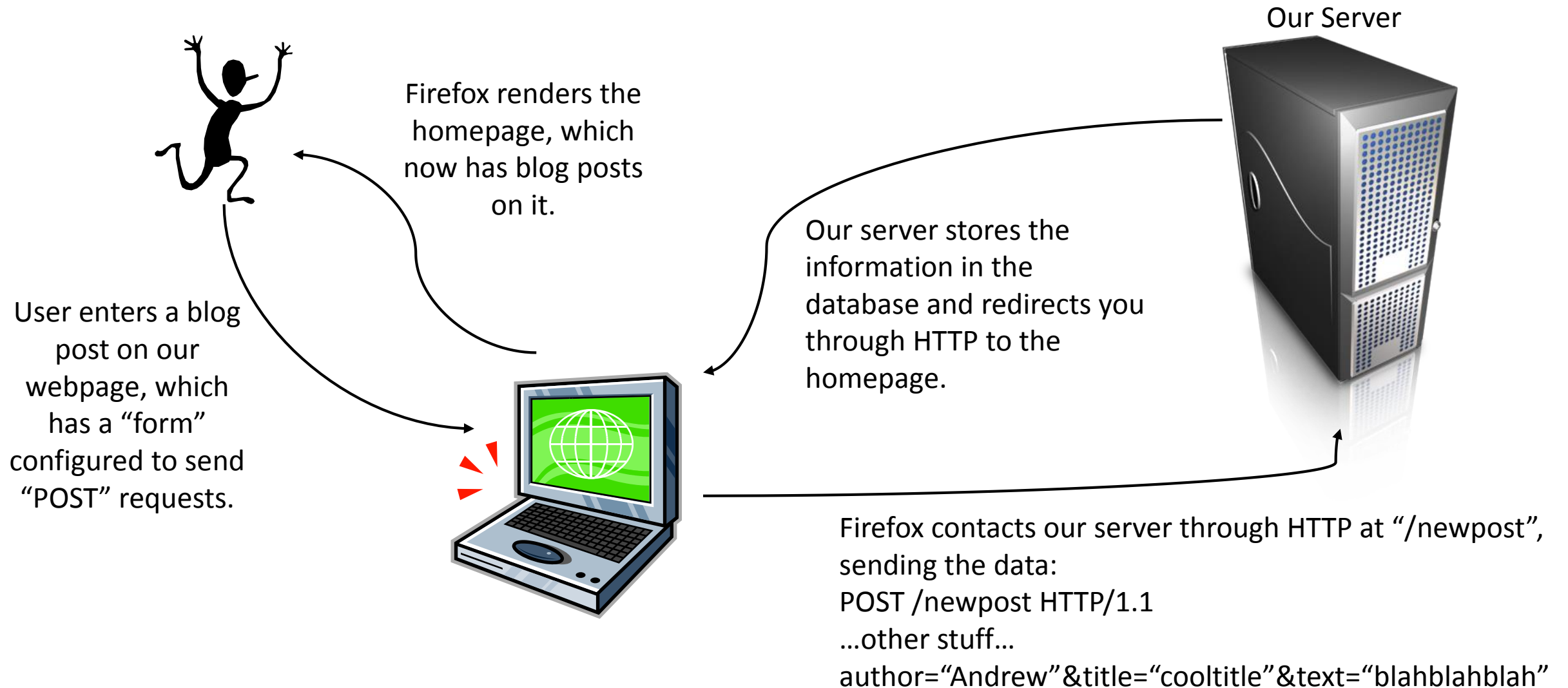
Rendering webpages instead of plain text

Exercise: Create handlers for "/about" and "/create".

Sending information to the server

Our Server

Firefox renders the homepage, which now has blog posts on it.

Our server stores the information in the database and redirects you through HTTP to the homepage.

User enters a blog post on our webpage, which has a "form" configured to send "POST" requests.

Firefox contacts our server through HTTP at "/newpost", sending the data:
POST /newpost HTTP/1.1
…other stuff…
author="Andrew"&title="cooltitle"&text="blahblahblah"

Putting blog posts in the database

AJAX – Asynchronous JavaScript And XML

Firefox runs some JavaScript, which updates the webpage with the blog posts without loading a new page.

User clicks the refresh button on our website.

Our server finds our data, turns it into JSON, and sends it to us through HTTP WITHOUT SENDING A WEBPAGE.

Firefox runs the JavaScript the button is connected to, which contacts our server through HTTP at "/refreshEntries", asking for data in the form of JSON (a way of representing data)

Coding up the AJAX request on both ends