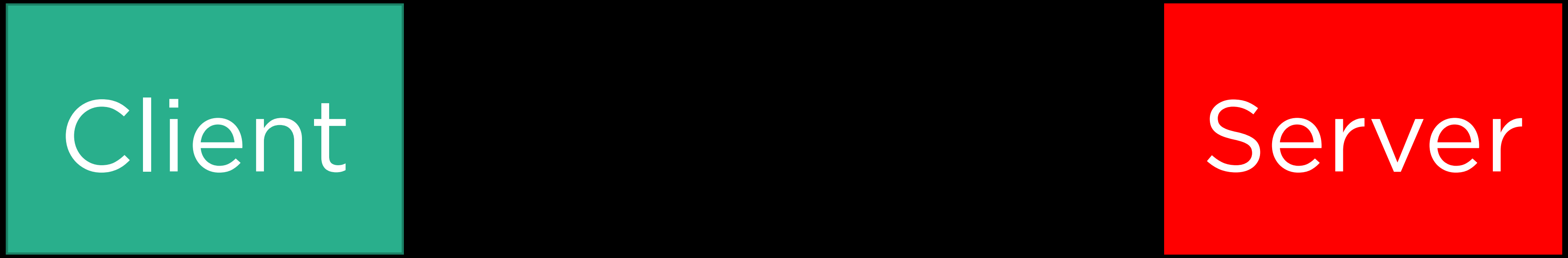
Software Engineering in the Arts and Humanities

Node.js

September 23, 2019

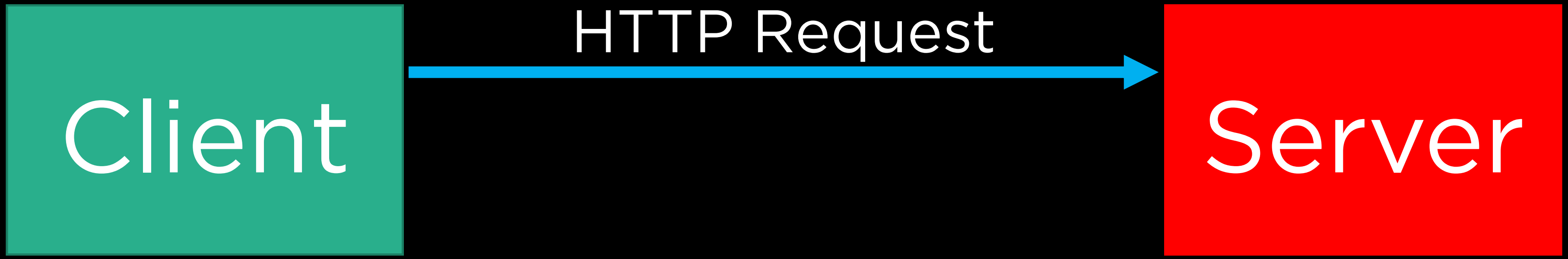
Agenda

- Node.js
- `npm`
- File manipulation
- CSVs
- And perhaps today, perhaps to start on Wednesday…
  - File Hashing
  - Express



Client

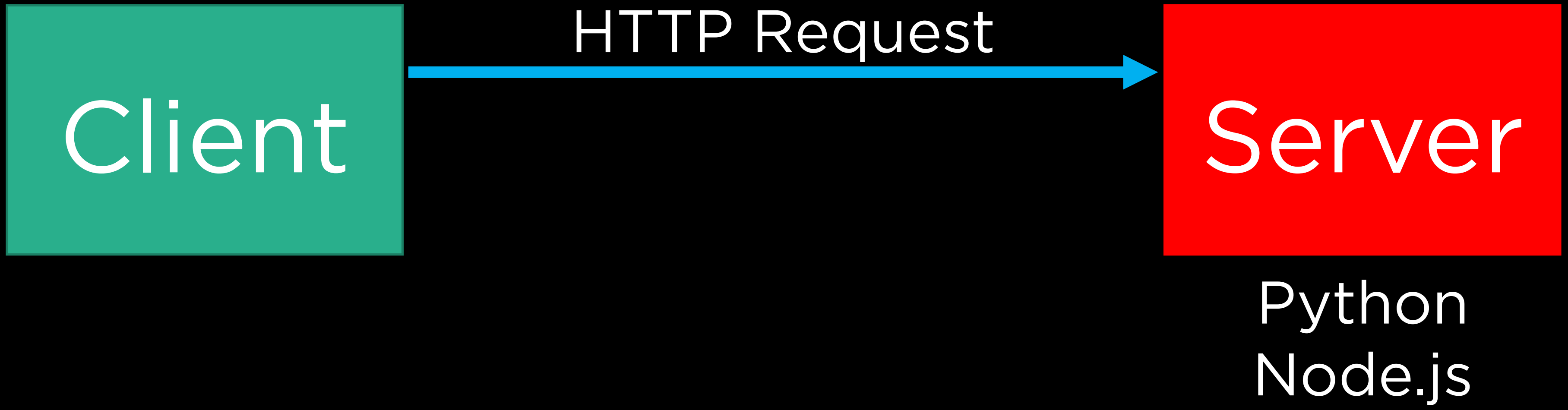
Server



Client

HTTP Request

Server

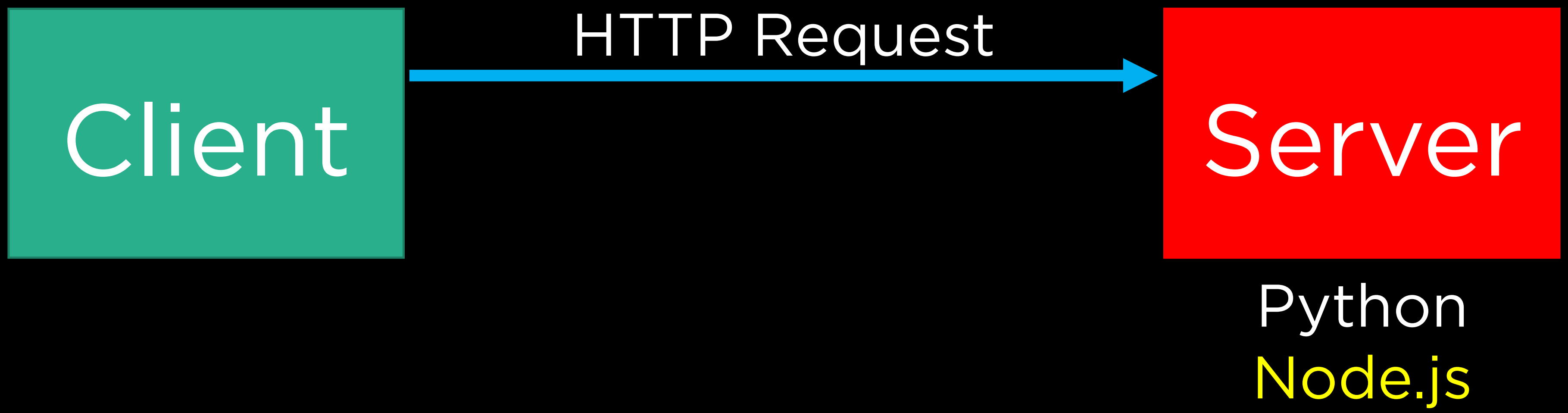


Client

HTTP Request

Server

Python
Node.js



Client

HTTP Request

Server

Python

Node.js

Node.js

Node.js

http://nodejs.org/

Node.js

- The basic idea behind Node.js ("Node") is to enable server-side scripting, meaning that JavaScript can be used outside of a browser context.

Node.js

- The basic idea behind Node.js ("Node") is to enable server-side scripting, meaning that JavaScript can be used outside of a browser context.

- In this way, it becomes more like other server-side languages, most popular being Python and PHP.

Node.js

- The basic idea behind Node.js ("Node") is to enable server-side scripting, meaning that JavaScript can be used outside of a browser context.

- In this way, it becomes more like other server-side languages, most popular being Python and PHP.

- At the outset, though, it means we can also run basic command line programs written in JavaScript.

npm

npm

- Much like other languages, JavaScript has a robust collection of pre-written code that you can incorporate.

npm

- Much like other languages, JavaScript has a robust collection of pre-written code that you can incorporate.

- Node refers to these collections of code as "packages," which you can think of as equivalent to Python "modules."

`npm`

- Much like other languages, JavaScript has a robust collection of pre-written code that you can incorporate.
- Node refers to these collections of code as "packages," which you can think of as equivalent to Python "modules."
- These packages can be included in your code by way of the `require()` method.

`npm`

- It is not simply enough to `require()` a package in order to use it; you may also need to install it into your project. For this, we ordinarily use a tool called `npm`. (Node Package Manager.)

`npm`

- It is not simply enough to `require()` a package in order to use it; you may also need to install it into your project. For this, we ordinarily use a tool called `npm`. (Node Package Manager.)

- This is true in most other languages as well; though you may not have had occasion to do it much in Python, this is the role that pip serves when importing modules.

`npm`

- It is not simply enough to `require()` a package in order to use it; you may also need to install it into your project. For this, we ordinarily use a tool called `npm`. (Node Package Manager.)

- This is true in most other languages as well; though you may not have had occasion to do it much in Python, this is the role that pip serves when importing modules.

- `npm` should come bundled with most installations of Node.

File Manipulation

fs

File Manipulation

- Thus far, the only "persistent data" we've worked with is using JavaScript `localStorage`.

File Manipulation

- Thus far, the only "persistent data" we've worked with is using JavaScript `localStorage`.

- As we turn our eye towards actual web applications that users will visit though, we need to rely on storage that is more robust.

File Manipulation

- Thus far, the only "persistent data" we've worked with is using JavaScript `localStorage`.
- As we turn our eye towards actual web applications that users will visit though, we need to rely on storage that is more robust.
- Next week we'll turn our attention to databases, but for now, we'll use Node to manipulate files.

File Manipulation

- The easiest way to get going working with, reading, and creating files in Node is via the `fs` package.

File Manipulation

- The easiest way to get going working with, reading, and creating files in Node is via the `fs` package.

```
let fs = require('fs');
```

File Manipulation

- The easiest way to get going working with, reading, and creating files in Node is via the `fs` package.

```
let fs = require('fs');
```

- Now, any file system manipulation methods can be invoked:

File Manipulation

- The easiest way to get going working with, reading, and creating files in Node is via the `fs` package.

```
let fs = require('fs');
```

- Now, any file system manipulation methods can be invoked:

```
fs.method(args...);
```

lodash

http://lodash.com/

lodash

- Similar in spirit to jQuery in terms of being a utility library, often times you may benefit from using the JS lodash library to help streamline certain function calls.

lodash

- Similar in spirit to jQuery in terms of being a utility library, often times you may benefit from using the JS lodash library to help streamline certain function calls.

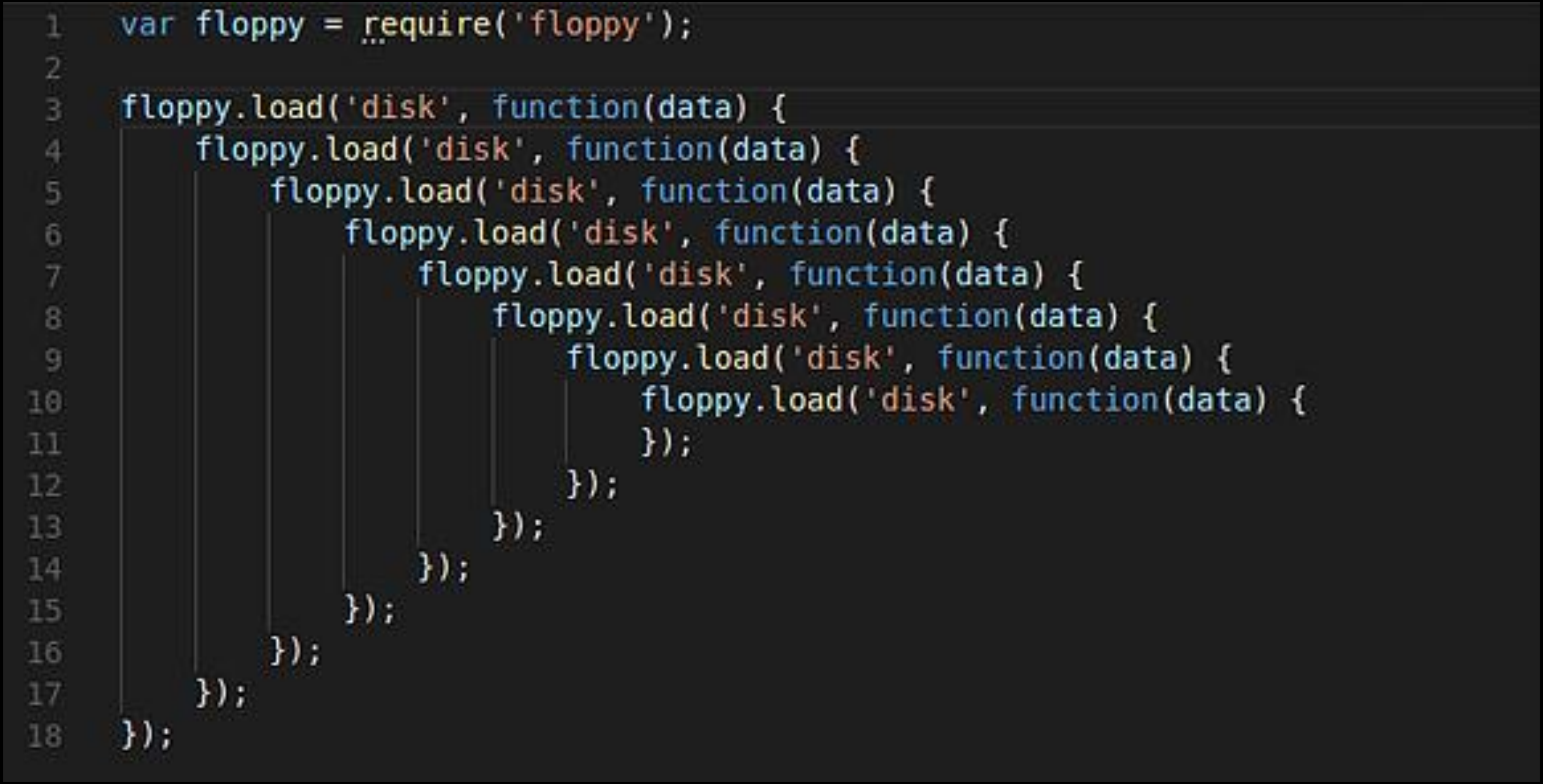
- Conventionally, when using lodash, it is assigned to the const variable with the name _ (underscore), similar to jQuery being associated with $.

lodash

- Similar in spirit to jQuery in terms of being a utility library, often times you may benefit from using the JS lodash library to help streamline certain function calls.

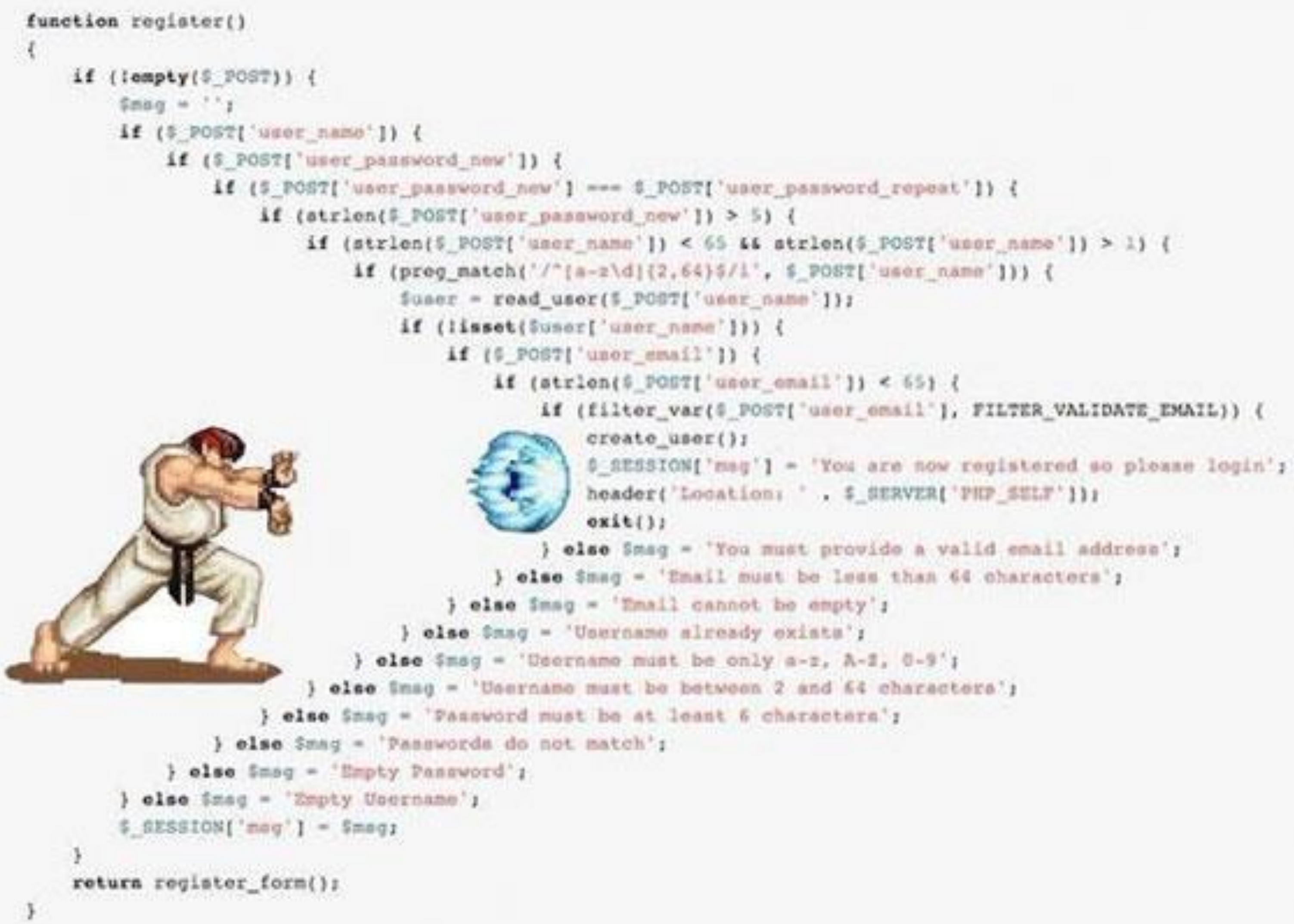
- Conventionally, when using lodash, it is assigned to the const variable with the name _ (underscore), similar to jQuery being associated with $.

- Let's see a few examples of this in action.



```javascript
var floppy = require('floppy');

floppy.load('disk', function(data) {
    floppy.load('disk', function(data) {
        floppy.load('disk', function(data) {
            floppy.load('disk', function(data) {
                floppy.load('disk', function(data) {
                    floppy.load('disk', function(data) {
                        floppy.load('disk', function(data) {
                            floppy.load('disk', function(data) {
                            });
                        });
                    });
                });
            });
        });
    });
});
```



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

async

```
async.series(tasks, callback);
async.parallel(tasks, callback);
async.each(tasks, callback);
async.eachSeries(tasks, callback);
async.series(tasks, callback);
async.parallel(tasks, callback);
async.each(tasks, callback);
async.eachSeries(tasks, callback);
async.series(tasks, callback);
async.parallel(tasks, callback);
async.each(tasks, callback);
async.eachSeries(tasks, callback);
```

CSV

File Hashing

File Hashing

- What is hashing?

File Hashing

- What is hashing?

- File hashing is a convenient (although *technically* not 100% foolproof) way to verify whether data (e.g., a file) has been seen before.

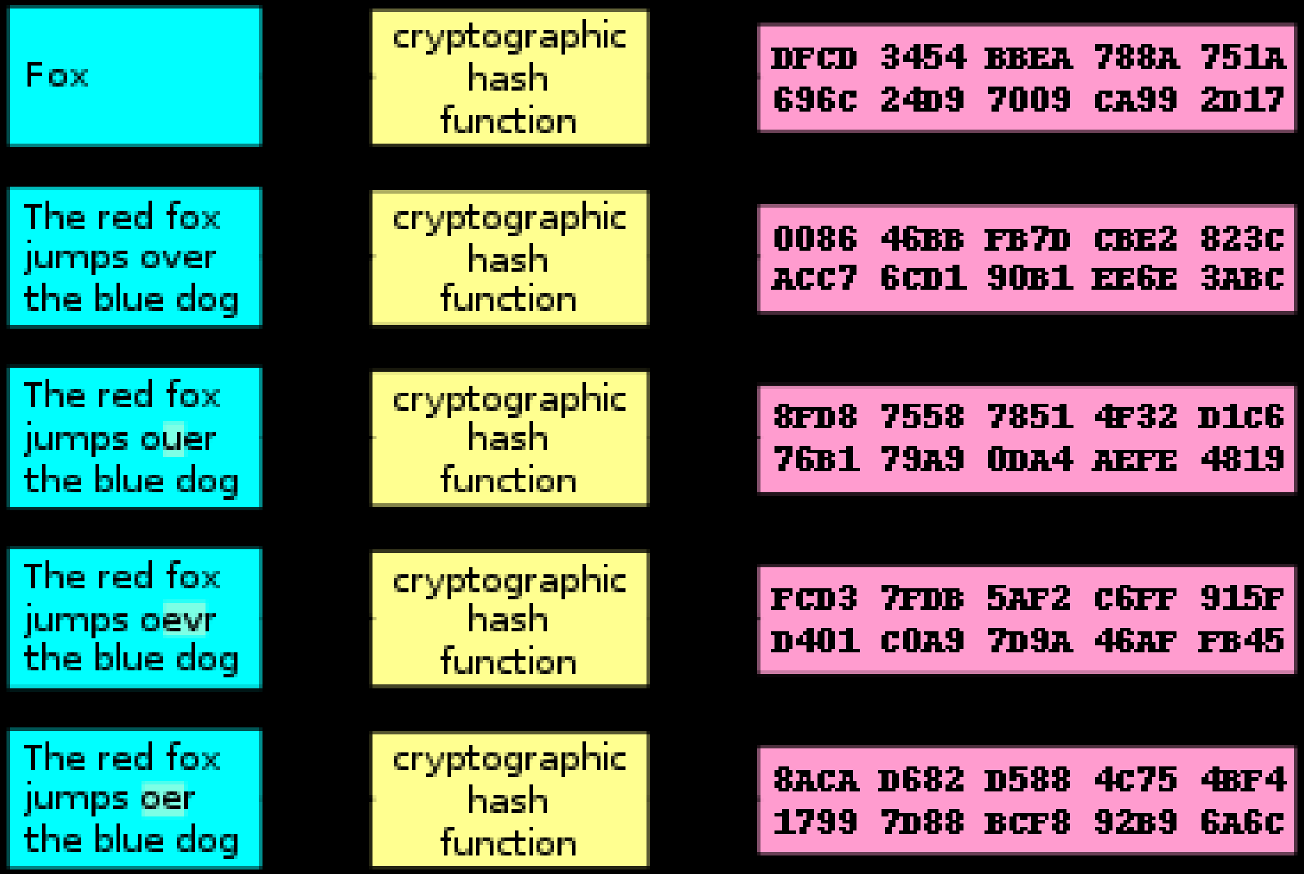
File Hashing

- What is hashing?

- File hashing is a convenient (although *technically* not 100% foolproof) way to verify whether data (e.g., a file) has been seen before.

- Since hashes are deterministic, you will always get the same value if you run the data through the hash function multiple times.

File Hashing



Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A
696C 24D9 7009 CA99 2D17

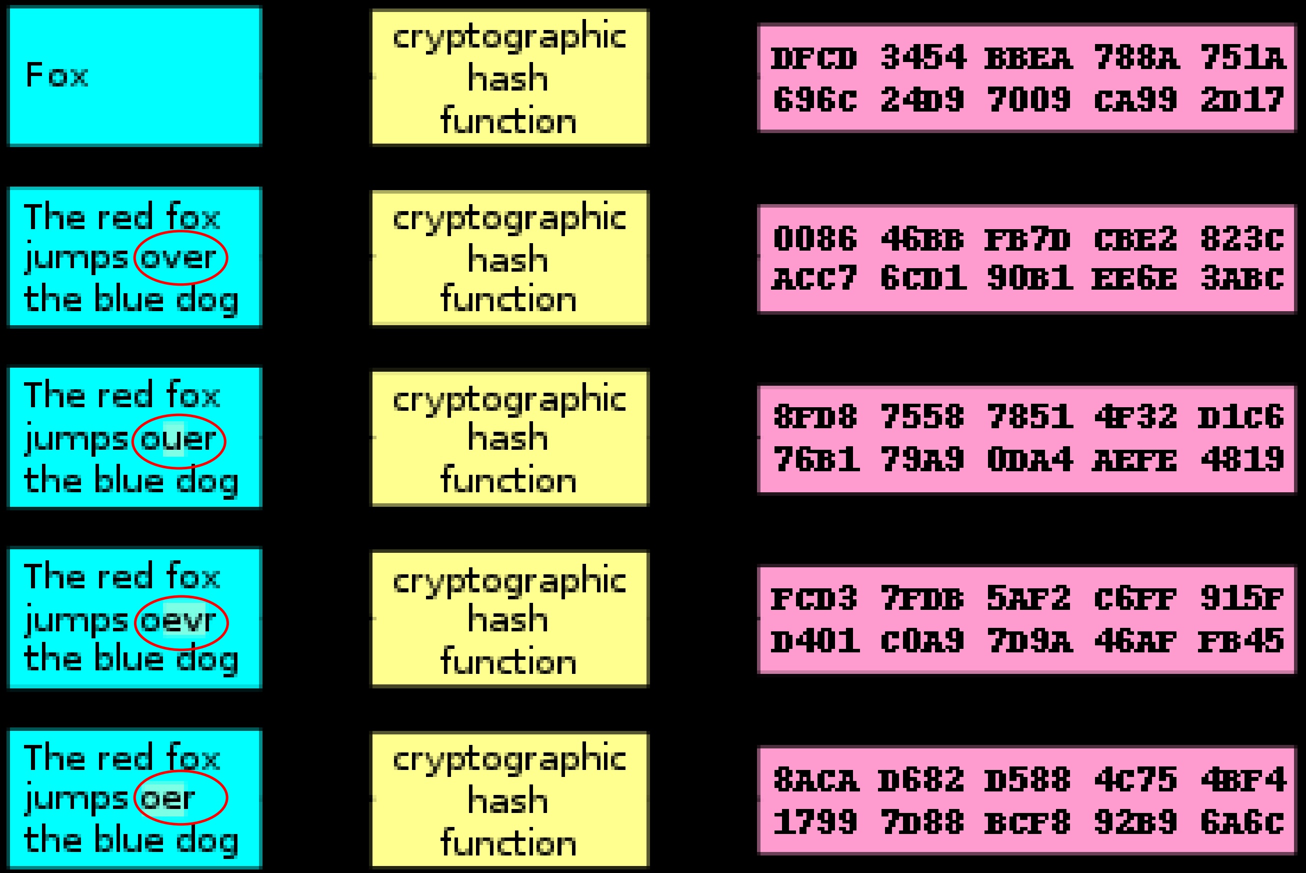
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C
ACC7 6CD1 90B1 EE6E 3ABC

The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6
76B1 79A9 0DA4 AEFE 4819

The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F
D401 C0A9 7D9A 46AF FB45

The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4
1799 7D88 BCF8 92B9 6A6C

File Hashing



Fox

cryptographic hash function

DFCD 3454 BBEA 788A 751A
696C 24D9 7009 CA99 2D17

The red fox jumps over the blue dog

cryptographic hash function

0086 46BB FB7D CBE2 823C
ACC7 6CD1 90B1 EE6E 3ABC

The red fox jumps ouer the blue dog

cryptographic hash function

8FD8 7558 7851 4F32 D1C6
76B1 79A9 0DA4 AEFE 4819

The red fox jumps oevr the blue dog

cryptographic hash function

FCD3 7FDB 5AF2 C6FF 915F
D401 C0A9 7D9A 46AF FB45

The red fox jumps oer the blue dog

cryptographic hash function

8ACA D682 D588 4C75 4BF4
1799 7D88 BCF8 92B9 6A6C

Web Server

`http`

Express

```
npm install express
```