DS-210: PROGRAMMING FOR DATA SCIENCE

LECTURE 28

1. EXTERNAL CRATE EXAMPLE: `csv` (READING CSV)

2. BASIC COLLECTIONS: STACK AND QUEUE

3. GRAPH EXPLORATION: BREADTH-FIRST SEARCH (BFS)

1. EXTERNAL CRATE EXAMPLE: `csv` (READING CSV)

2. BASIC COLLECTIONS: STACK AND QUEUE

3. GRAPH EXPLORATION: BREADTH-FIRST SEARCH (BFS)

CRATE `csv` (AND `serde`): READING A CSV FILE

See:

- https://crates.io/crates/csv
- https://crates.io/crates/serde

- Create a new project
- Add to `Cargo.toml`:

```
csv = "1.1.6"
serde = "1.0.136"
```

Crate `csv` (and `serde`): reading a CSV file

See:

- https://crates.io/crates/csv
- https://crates.io/crates/serde

- Create a new project
- Add to `Cargo.toml`:

```
csv = "1.1.6"
serde = "1.0.136"
```

- Copy the second example from the `csv` docs
- Update the field names

CRATE `csv` (AND `serde`): READING A CSV FILE

See:

- https://crates.io/crates/csv
- https://crates.io/crates/serde

- Create a new project
- Add to `Cargo.toml`:

```
csv = "1.1.6"
serde = "1.0.136"
```

- Copy the second example from the `csv` docs
- Update the field names

Doesn't work!!!!!

- Search for solution online!

CRATE `csv` (AND `serde`): READING A CSV FILE

Solution: modify `Cargo.toml` for `serde`

```
serde = { version = "1.0.136", features = ["derive"] }
```

CRATE `csv` (AND `serde`): READING A CSV FILE

Solution: modify `Cargo.toml` for `serde`

```
serde = { version = "1.0.136", features = ["derive"] }
```

Our case: add this before `Record` to supress warnings

```
#[allow(dead_code,non_snake_case)]
```

CRATE `csv` (AND `serde`): READING A CSV FILE

Solution: modify `Cargo.toml` for `serde`

```
serde = { version = "1.0.136", features = ["derive"] }
```

Our case: add this before `Record` to supress warnings

```
#[allow(dead_code,non_snake_case)]
```

Bottom line:

- parameters other than the version number possible in `Cargo.toml`

RELYING ON EXTERNAL PROJECTS

Things to consider about external libraries:

- trustworthy?
- stable?
- long–term survival?
- do you really need it?

RELYING ON EXTERNAL PROJECTS

Things to consider about external libraries:

- trustworthy?
- stable?
- long–term survival?
- do you really need it?

Many things best left to professionals:

Never implement your own crypto!

RELYING ON EXTERNAL PROJECTS

Things to consider about external libraries:

- trustworthy?
- stable?
- long–term survival?
- do you really need it?

Many things best left to professionals:

Never implement your own crypto!

Implementing your own things can be a great educational experience!

EXTREME EXAMPLE

theregister.com/2016/03/23/npm_left_pad_chaos/

The Register

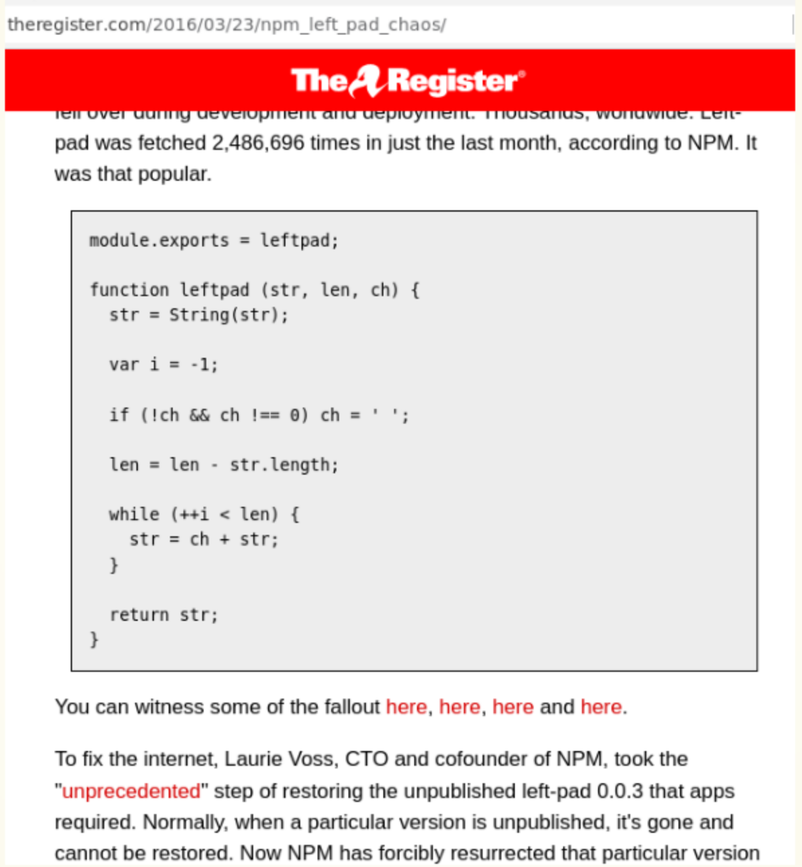
fell over during development and deployment. Thousands, worldwide. Left-pad was fetched 2,486,696 times in just the last month, according to NPM. It was that popular.

```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

You can witness some of the fallout here, here, here and here.

To fix the internet, Laurie Voss, CTO and cofounder of NPM, took the "unprecedented" step of restoring the unpublished left-pad 0.0.3 that apps required. Normally, when a particular version is unpublished, it's gone and cannot be restored. Now NPM has forcibly resurrected that particular version

EXTREME EXAMPLE


theregister.com/2016/03/23/npm_left_pad_chaos/

The Register

fell over during development and deployment. Thousands, worldwide. Left-pad was fetched 2,486,696 times in just the last month, according to NPM. It was that popular.

```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

You can witness some of the fallout here, here, here and here.

To fix the internet, Laurie Voss, CTO and cofounder of NPM, took the "unprecedented" step of restoring the unpublished left-pad 0.0.3 that apps required. Normally, when a particular version is unpublished, it's gone and cannot be restored. Now NPM has forcibly resurrected that particular version

Rust and `cargo`: can't delete libraries that were published

1. EXTERNAL CRATE EXAMPLE: `csv` (READING CSV)

2. BASIC COLLECTIONS: STACK AND QUEUE

3. GRAPH EXPLORATION: BREADTH-FIRST SEARCH (BFS)

BASIC DATA STRUCTURES: STACK AND QUEUE

Stack (same name as in "stack vs. heap"):

- FILO: first in last out / LIFO: last in first out
- put items on the top
- get items from the top
- can use `Vec` for this: methods `push` and `pop`

BASIC DATA STRUCTURES: STACK AND QUEUE

Stack (same name as in "stack vs. heap"):

- FILO: first in last out / LIFO: last in first out
- put items on the top
- get items from the top
- can use `Vec` for this: methods `push` and `pop`

Queue:

- FIFO: first in last out
- add items at the end
- get items from the front

RUST: `std::collections::VecDeque<T>`

- generalization of queue and stack
- accessing front: methods `push_front(x)` and `pop_front()`
- accessing back: methods `push_back(x)` and `pop_back()`
- `pop_front` and `pop_back` return `Option<T>`

RUST: `std::collections::VecDeque<T>`

- generalization of queue and stack
- accessing front: methods `push_front(x)` and `pop_front()`
- accessing back: methods `push_back(x)` and `pop_back()`
- `pop_front` and `pop_back` return `Option<T>`

In [2]:
```
use std::collections::VecDeque;

// using as a stack: push_back & pop_back
let mut stack = VecDeque::new();

stack.push_back(1);
stack.push_back(2);
stack.push_back(3);

println!("{:?}",stack.pop_back());
println!("{:?}",stack.pop_back());

stack.push_back(4);
stack.push_back(5);

println!("{:?}",stack.pop_back());
```

```
Some(3)
Some(2)
Some(5)
```

RUST: `std::collections::VecDeque<T>`

- generalization of queue and stack
- accessing front: methods `push_front(x)` and `pop_front()`
- accessing back: methods `push_back(x)` and `pop_back()`
- `pop_front` and `pop_back` return `Option<T>`

In [2]:
```
use std::collections::VecDeque;

// using as a stack: push_back & pop_back
let mut stack = VecDeque::new();

stack.push_back(1);
stack.push_back(2);
stack.push_back(3);

println!("{:?}",stack.pop_back());
println!("{:?}",stack.pop_back());

stack.push_back(4);
stack.push_back(5);

println!("{:?}",stack.pop_back());
```

```
Some(3)
Some(2)
Some(5)
```

In [3]:
```
// using as a queue: push_back & pop_front
let mut queue = VecDeque::new();

queue.push_back(1);
queue.push_back(2);
queue.push_back(3);

println!("{:?}",queue.pop_front());
println!("{:?}",queue.pop_front());

queue.push_back(4);
queue.push_back(5);

println!("{:?}",queue.pop_front());
```

```
Some(1)
Some(2)
Some(3)
```

IMPLEMENTATION OF `VecDeque`

How would you do it?

IMPLEMENTATION OF `VecDeque`

How would you do it?

- use an array allocated on the heap
- keep index of the front and end
- wrap around

IMPLEMENTATION OF `VecDeque`

How would you do it?

- use an array allocated on the heap
- keep index of the front and end
- wrap around

Out of space?

- double the size
- good complexity due to amortization

1. EXTERNAL CRATE EXAMPLE: `csv` (READING CSV)

2. BASIC COLLECTIONS: STACK AND QUEUE

3. GRAPH EXPLORATION: BREADTH-FIRST SEARCH (BFS)

GRAPH EXPLORATION

Sample popular methods:

- breadth–first search (BFS)
  - next lecture
  - uses a queue
  - great for computing distances!

GRAPH EXPLORATION

Sample popular methods:

- breadth–first search (BFS)
  - next lecture
  - uses a queue
  - great for computing distances!
- depth–first search (DFS)
  - next lecture
  - uses a stack

GRAPH EXPLORATION

Sample popular methods:

- breadth–first search (BFS)
  - next lecture
  - uses a queue
  - great for computing distances!
- depth–first search (DFS)
  - next lecture
  - uses a stack
- random walks
  - example: PageRank (see HW 10)