AN INTRODUCTION TO

RUST

“A CURLY-BRACE, BLOCK-STRUCTURED EXPRESSION LANGUAGE”

RUST SYNTAX

```rust
fn main() {
    println!("Hello, Vienna.rb!");
}
```

```
fn is_three(num: int) -> bool {
   num % 3 == 0
}
fn is_five(num: int) -> bool {
   num % 5 == 0
}
fn is_fifteen(num: int) -> bool {
    num % 15 == 0
}

fn main() {
        for num in range(1, 101) {
            println(
              if is_fifteen(num) { ~"FizzBuzz" }
              else if is_three(num) { ~"Fizz" }
              else if is_five(num) { ~"Buzz" }
              else { num.to_str() }
            );
        }
    }
```

```
match my_number {
  0      => println("zero"),
  1 | 2 => println("one or two"),
  3..10 => println("three to ten"),
  _      => println("something else")
}
```

MEMORY SAFETY

```c
int *dangling(void)
{
    int i = 1234;
    return &i;
}

int add_one(void)
{
    int *num = dangling();
    return *num + 1;
}
```

```
fn dangling() -> &int {
    let i = 1234;
    return &i;
}

fn add_one() -> int {
    let num = dangling();
    return *num + 1;
}
```

```
dangling.rs:3:9: 3:11 error: borrowed value does not live long
enough
dangling.rs:3  return &i;
                      ^~
dangling.rs:1:23: 4:2 note: reference must be valid for the
anonymous lifetime #1 defined on the block at 1:22...
dangling.rs:1 fn dangling() -> &int {
dangling.rs:2  let i = 1234;
dangling.rs:3  return &i;
dangling.rs:4 }
dangling.rs:1:23: 4:2 note: ...but borrowed value is only valid
for the block at 1:22
dangling.rs:1 fn dangling() -> &int {
dangling.rs:2  let i = 1234;
dangling.rs:3  return &i;
dangling.rs:4 }
error: aborting due to previous error
```

OWNERSHIP OF MEMORY

```
fn dangling() -> &int {
    let i = 1234;
    return &i;
}

fn add_one() -> int {
    let num = dangling();
    return *num + 1;
}
```

```
fn dangling() -> ~int {
    let i = ~1234;
    return i;
}

fn add_one() -> int {
    let num = dangling();
    return *num + 1;
}
```

```rust
// Rust
let i = ~1234;
```

```cpp
// C++
int *i = new int;
*i = 1234;
```

CONCURRENCY

```
fn main() {
    let nums = [1, 2];
    let noms = ["Tim", "Eston", "Aaron", "Ben"];

    let mut odds = nums.iter().map(|&x| x * 2 - 1);

    for num in odds {
        do spawn {
            println!("{:s} says hello from a lightweight
thread!", noms[num]);
        }
    }
}
```

```
fn main() {
    let numbers = [1,2,3];

    let (port, chan)  = Chan::new();
    chan.send(numbers);

    do spawn {
        let numbers = port.recv();
        println!("{:d}", numbers[0]);
    }
}
```

```rust
extern mod extra;
use extra::arc::Arc;

fn main() {
    let numbers = [1,2,3];

    let numbers_arc = Arc::new(numbers);

    for num in range(0, 3) {
        let (port, chan)  = Chan::new();
        chan.send(numbers_arc.clone());

        do spawn {
            let local_arc = port.recv();
            let task_numbers = local_arc.get();
            println!("{:d}", task_numbers[num]);
        }
    }
}
```

```rust
extern mod extra;
use extra::arc::RWArc;

fn main() {
    let numbers = [1,2,3];

    let numbers_arc = RWArc::new(numbers);

    for num in range(0, 3) {
        let (port, chan)  = Chan::new();
        chan.send(numbers_arc.clone());

        do spawn {
            let local_arc = port.recv();

            local_arc.write(|nums| {
                nums[num] += 1
            });

            local_arc.read(|nums| {
                println!("{:d}", nums[num]);
            })
        }
    }
}
```

NOT DONE YET

0.9 STABLE

0.10-PRE

SYNTAX NOT YET STABLE

HTTPS://GITHUB.COM/MOZILLA/SERVO

HTTP://WWW.RUSTFORRUBYISTS.COM