

FIT2102

# Programming Paradigms

## Tutorial 3

Pretty Printing



# TypeScript

- Type annotations on top of JS.

```
$ npm install -g typescript  
$ npm run build
```

The TypeScript logo, consisting of the letters 'TS' in a bold, white, sans-serif font, centered within a solid blue square.

# WHO SAID TYPES?

First, deactivate implicit any which means everything has to be typed!

Basic Example:

```
let x : number;
```

More Complex Example:

```
function map<T,V>(f: (x:T)=>V, l: ConsList<T>): ConsList<V>
```

# Cons Lists (Church encoding)

Can we create lists with only lambda (anonymous) functions?

```
const cons = (head, rest) => f => f(head, rest)
```



A diagram consisting of two yellow boxes. The top box contains the text 'f => f(head, rest)' and is part of the 'cons' function definition. The bottom box contains the text 'list => list((head, rest) => head)' and is part of the 'head' variable definition. Two yellow lines originate from the 'f' box and point to the 'head' and 'rest' arguments in the 'aList' definition.

```
const aList = cons('Lists', cons("don't", cons("get", cons('any',  
    cons('simpler', cons('than', cons('this',  
        null)))))))
```

```
const head = list => list((head, rest) => head)
```

```
const rest = list => list((head, rest) => rest)
```

```
head(rest(rest(aList)))  ⇨ "get"
```

# How do we get stuff out?

```
const aList = cons('Lists', cons("don't", cons("get",  
    cons('any', cons('simpler',  
    cons('than', cons('this'))))))))
```

```
function listToString(l:ConsList<string>): string {  
    if (!l) {  
        return ''  
    } else {  
        return head(l) + ' ' + listToString(rest(l))  
    }  
}
```

```
console.log(listToString(aList));  
> Lists don't get any simpler than this!
```

# CONSLIST AND LIST

1. Define a cons list, functional representation of a list.
2. Define a List object containing a ConsList. As an object, its methods always return an object.

Create functions on both:

- map
- filter
- reduce
- etc.