Linked Lists Live

Out of the cups and onto the screen

Recap

- Arrays:
  - fast for random access
  - slow to change size
  - slow to insert an item
  - low memory overhead
- Lists:
  - fast to add items
  - fast to iterate through
  - slow for random access
  - small memory overhead

Sticks and Cups in Javascript

d.data == "Dr No"
d.tail.data == "Goat curry"
d.tail.tail.data == ??

```
var a = {
    data : "It has to end",
    tail : null
}
```

```
var b = {
    data: "Bling",
    tail: a
}
```

```
var c = {
    data : "Goat curry",
    tail : b
}
```

```
var d = {
    data : "Dr No",
    tail : c
}
```

Ugh... Work..

- We can access list from just "d"
- "a", "b" and "c" seem useless
- Seems silly to have lots of tails
- Good programmer == lazy programmer
- So what next?

List API

- Things you want to do with a list:
  - Create a new list:
    ```
    var fruit = new List()
    ```
  - Add a new element to the list:
    ```
    fruit = fruit.add("Apple")
    ```
  - Check for an empty list:
    ```
    if (fruit.isEmpty) { .... }
    ```
  - Access all elements in the list:
    ```
    while (fruit.hasNext()) { console.log(fruit.head); }
    ```
  - Count elements in list, copy list, join lists etc.

List API!

```
function List(data, tail) {
    this.data = data
    this.tail = tail

    // Add an element to the head of this list
    this.add = function(data) {
        return new List(data, this)
    }
}

var myList = new List(null,null)
myList = myList.add("a").add("b").add("c")

// mylist.data == "c"
// mylist.tail.data == "b"
```

More List API

- We need the following functions:
  - hasNext() returns true if there is another element in the list
  - next() returns the head of the tail
  - foreach(lambda): Apply function to each element in the list
  - length(): Calculate the number of elements in the list

"Programs must be written for people to read, and only incidentally for machines to execute"

H. Abelson and G. Sussman

"Weeks of programming can save you hours of planning"