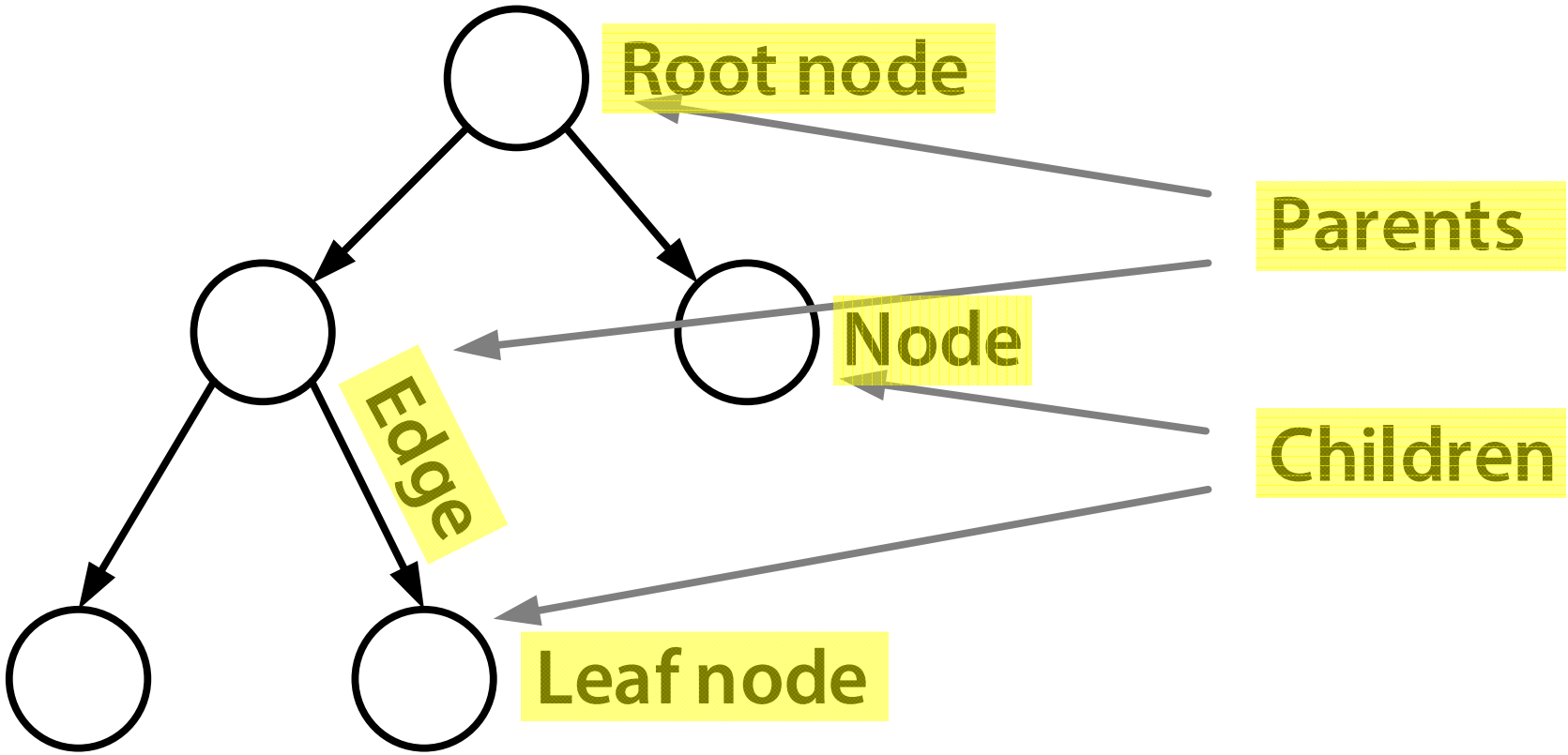
Trees

Kit Eason
www.kiteason.com
@kitlovesfsharp

pluralsight
hardcore dev and IT training

Tree Terminology



A Cake is a Tree!

- A cake has ingredients…
- …which might have other ingredients

```
type FoodStuff(name : string, FoodStuffs : FoodStuff list) =
    member this.Name = name
    member this.FoodStuffs = FoodStuffs
```

Options for Tree Representation

- **A recursive type**
  - …like the cake
- **Discriminated unions**

Discriminated Unions

```
type Shape =
    | Rectangle of width : float * length : float
    | Circle of radius : float
    // This is the formal name of a 'running track' shape!
    | Stadium of straight : float * radius : float
```

- Represent a number of ‘shapes’ under the same type
- Each ‘shape’ has an identifier…
- …and can have a tuple of properties

F# 3.1 Versus F# 3.0

- F# 3.1 onward – properties can have names (optional)

```
type Shape =
    | Rectangle of width : float * length : float
    | Circle of radius : float
    // This is the formal name of a 'running track' shape!
    | Stadium of straight : float * radius : float
```

- F# 3.0 – properties just a tuple

```
type Shape30 =
    | Rectangle of float * float
    | Circle of float
    | Stadium of float * float
```

Binary Tree With Discriminated Union

- Binary tree of integers

```
type Tree =
    | Leaf of int
    | Node of Tree * Tree
```

- Generic binary tree

```
type Tree<'T> =
   | Leaf of 'T
   | Node of Tree<'T> * Tree<'T>
```

A Car is a Tree!

- Binary trees – sometimes useful
- More on Red-Black trees etc. here:
  http://tinyurl.com/FSRedBlack

Description

```
type Description =
   { Name : string;
     PartNumber : string;
     Cost : decimal option }
```

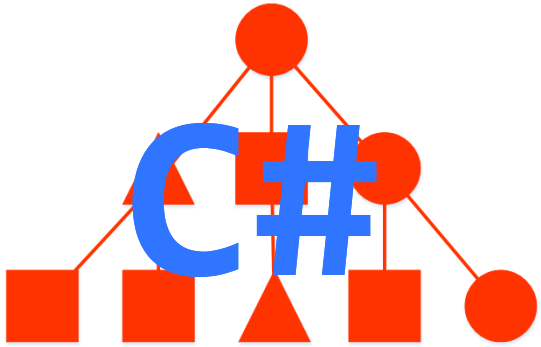
```
let pad =
   { Name = "Brake Pad"
     PartNumber = "B12345"
     Cost = Some 15.90M }
```

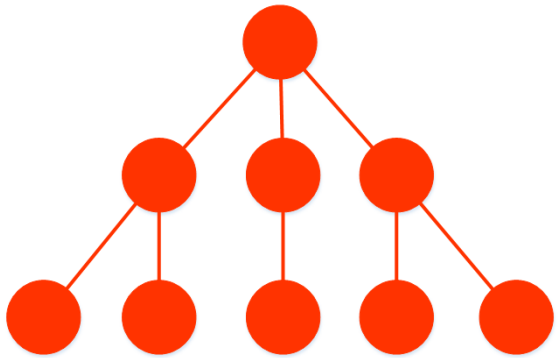
Kinds of Tree Item

- Single part
- Repeated part
- Compound part

Choosing a Tree Representation

- **Nodes each one of a set of kinds?**
  - Tree of Discriminated Unions
- **Thinking of OO inheritance to make a tree?**
  - Tree of Discriminated Unions
- **Exposing tree internals to C#, VB.NET; serializing**
  - Consider an OO approach
- **Nodes all the same kind**
  - An OO recursive type is fine
  - A DU is also fine!
    - Single-case
    - Node and Leaf cases





Summary

- **Represent a tree as a recursive OO type**
- **…or as a Discriminated Union**
- **One or more DU cases will recurse to the same DU type**
- **Traverse a DU tree with a recursive function**
  - F# ‘match’ on DU cases
  - Where DU case recurses, the function recurses
- **Choose OO or DU carefully**
  - Simplest code
  - Consider other languages, serialization