Functional Programming

Traditional Programming

Reading List

Course Overview
Pros and Cons

Functional vs Imperative:

Pros:
- Lazy evaluation: lazy evaluation
- No nested functions
- Can use higher-order functions
- Easier to write functions
- No side effects

Cons:
- DOM model is harder to understand and use.
- Harder to write.
- Slower.

In fact, the library is full of functions like this:

- Don't hesitate to pass functions as parameters and return functions.
- If you really need them, make them arguments.
- No state variables.
- Debug and continue.
- Base case induction step.
- Think recursion, not loop.

Example of use:

double f = double factorial;

e = \theta \in \mathbb{R}
(f(\theta) = \theta!)

This function maps a function and outputs a function with \( g(x) = x^2 \).

- can create a new function on the fly
- can pass a function as a parameter
- Functions are first-class citizens. The data value are functional programming (cont.).
Expressions and Values

Expressions are things you want the computer to calculate.

Example:

```
x + y + has type Integer -> Integer -> Integer +
3 + 4 has type Integer
```

Interpretation:

```plaintext
3, 4, 5 has type Integer, Integer, Integer
```

Interpretation:

```plaintext
[3, 4, 5] has type [Integer, Integer, Integer]
```

Interpretation:

```plaintext
3 + 4 has type Integer
```

Interpretation:

```plaintext
3 + 4 + has type Integer - Integer - Integer
```

Types (cont.)

Each expression and value has a data type.

Interpretations:

```plaintext
Integer, Lazy evaluation.
Integer, Purely functional, No side effects.
```

Summary:

A lot of mature ideas, research, and experience in functional

The Haskell Language

Named after the logician Haskell Curry.
Local Bindings (cont.)

- Inside the scope of the expression, the binding is visible.
- Therefore, the value of the expression will be 9.
- Within the scope of the expression, x is temporarily bound to 3.
- The body of the expression is x³.
- This is called a let-expression.
- Let x = 3 in x³

Note: where clauses are only for definitions, not expressions.

\[ \begin{align*}
\text{let } & p = 1 \text{ in } \text{let } j = 1 \text{ then } (p + 1) \text{ end } \\
\text{let } & p = 1 \text{ in } (p + 1) \text{ end } \\
\text{fourth-power } & x = x^2 \text{ where } x = 2 \\
4 + x & = 9 \\
3 & = x \\
\text{fourth-power } & x = x^2 \text{ where } x = 2 \\
\text{Let } & p = 1 \text{ in } \text{let } j = 1 \text{ then } (p + 1) \text{ end } \\
\text{let } & p = 1 \text{ in } (p + 1) \text{ end } \\
\text{fourth-power } & x = x^2 \text{ where } x = 2 \\
\text{local bindings may also be used in a function expression:} \\
\lambda & x \cdot x \\
4 + x & = 9 \\
3 & = x \\
\text{multiple local bindings in a function expression:} \\
\lambda & x \cdot x
\end{align*} \]