THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Mark Chernov
Team 3
CSCI 6221

Reason Programming Language

Introduction

- ReasonML is a syntax extension for the OCaml language created by Facebook (by the same team that built React web framework). OCaml is a battle-tested functional systems programming language that's been around since the late 1990s.
- Reason can be compiled to assembly and run as native code or to JavaScript to run in the web browser.
- Reason can interact with existing JavaScript code via a FFI (foreign function interface).
- Similar to what they're doing with React, Facebook tests all of the new additions to the language *internally* before they actually change the language.
- Notable users include: Facebook's Messenger app.

Names, Binding, and Scopes

- A "let binding" binds values to names. In other languages they might be called a "variable declaration".

```
let greeting = "hello!";
let score = 10;
let newScore = 10 + score;
```

- Reason let bindings are "immutable", they cannot change after they are created.

```
let x = 10;
/* Error: Invalid code! */
x = x + 13;
```

- Every .re file is a module with scoped variables. Bindings can be manually scoped using {}

```
let message = {
  let part1 = "hello";
  let part2 = "world";
  part1 ++ " " ++ part2
};
/* `part1` and `part2` not accessible here! */
```

Data Types

Primitives

- string `let s = "Hello " ++ "World!";`
- int `let x = 23 + 1 - 7 * 2 / 5;`
- float `let x = 23.0;`
- bool `let z = x && y || false;`
- char `let c = "Hello".[1];`

Basic Data Structures

- Records are structures used for storing data in named fields. They are similar to objects or structs in other languages.

```
type person = {
  name: string,
  age: int,
};
```

```
let alice = {
  name: "Alice",
  age: 42,
};
```

- List, Array, Tuple

```
let listA = [1, 2, 3];
let arrayA = [| 1, 2, 3 |];
let pair = (1, "hello");
```

Expressions and Assignment Statements

- Functions are a core part of Reason language. They perform logic and return values based on the arguments provided. Functions are first class citizen and could be assign to variables.

```
let add = (x, y) => {
  x + y;
};
```

- In Reason everything is an expression and returns value.

Feature	Example
If-Else expressions	`if (condition) { a; } else { b; }`
Ternary expressions	`condition ? a : b;`

- Use Pipe operator to pass values down to function calls.

```
let data = [1, 3, 5, 2, 0, 4 ];
data
  |> List.filter(i => i != 0)
  |> List.map(i => i * 2)
  |> List.map(i => string_of_int(i));
```

Functional Programming

- Reason is a functional programming language
- Reason doesn't have support for **object-oriented programming**.
- Can simulate with help of Advanced Types, but is preferable to keep logic and data separate.

```
type tesla = {.
  drive: int => int
};

let obj: tesla = {
  val hasEnvy = ref(false);
  pub drive = (speed) => {
    this#enableEnvy(true);
    speed
  };
  pri enableEnvy = (envy) => hasEnvy := envy
};
```

Concurrency

- Reason compiles down to JavaScript and uses JavaScript VM concurrency model based on an **event loop**, which is responsible for executing the code, collecting and processing events, and executing sub-tasks from the queue running in a single tread.
- This model is quite different from multithreaded models in other languages like C and Java.

Exception Handling and Event Handling

- Functional programming tries to minimize side effects, so throwing exceptions is generally avoided.
- Instead, if an operation can fail, it should return a representation of its outcome including an indication of success or failure. But it is still possible to raise exception to stop program execution or catch it with “try” keyword.
- Reason contains an Option type to use instead of throwing exceptions.

```
type option('value) =
  | None
  | Some('value);
```

```
let happyBirthday = (user) => {
  switch (user) {
  | Some(person) => "Happy birthday " ++ person.name
  | None => "Please login first"
  };
};
```

Summary and Questions?

- **A rock solid type system.** Reason types have 100% coverage (every line of code), so once it compiles, the types are guaranteed to be accurate).
- **A focus on simplicity & pragmatism and performance**. Reason is pure, immutable and functional with very fast compiler.
- **Great ecosystem & tooling**. Use favourite JavaScript NPM packages, and your existing React JavaScript framework.