Introduction: Structured data in XML, JSON and CSV

- Structured formats are ubiquitous. Open-government initiatives release data as CSV, web services communicate using JSON or XML.
- Schema vs. examples. Real-world data often do not carry explicit schema. Examples are more common. Documentation for services usually includes examples of “typical” server responses.
- Type-safe data access is hard. Data extraction expects known format, but statically-typed languages do not understand it.

Motivation: Printing names and ages from JSON file

```fsharp
[ {
    "name" : null,
    "age" : 23,
} ,
{
    "name" : "Alexander",
    "age" : 1.5,
} ,
{
    "name" : "Tomas"
} ]
```

```fsharp
match data with
| Array items ->
    for item in items do
    match item with
    | Object o -> print (Map.find o "name")
    | _ -> failwith "Incorrect format"
    | _ -> failwith "Incorrect format"

Records of matching names are unified, introducing optional fields. Collections are unified by lifting the type of their elements. Types of different kinds are combined into a flattened sum type.

```

Solution: Using the F# Data JSON type provider

```fsharp
let items = People.Parse(data)
for item in items do
    printf "%s" item.Name
    Option.iter (printf "%d") item.Age
```

Summary: What makes F# Data interesting?

- **Schema inference.** The People.Parse method returns array of entities with Name of type string and Age of type int option. The member names and types are inferred from sample JSON.
- **Ease of use and tooling.** Full type information is available. Used by F# tools to provide auto-complete, type hints and docs (available in Xamarin Studio, Visual Studio, Emacs and more)
- **Safety properties.** Same as in the original implementation. Guaranteed to work if input value is a subtype of sample(s). Otherwise throws a runtime exception that can be handled.

Approach: Structural type inference algorithm

Primitive types are inferred from values. The following hierarchy is used to find the most specific common subtype. Note that 0 and 1 are treated as Boolean values and null+τ is an option type.

```

Thanks to: Don Syme, Gustavo Guerra & other contributors.