Frontend Framework

Je bouwt een frontend framework naar analogie met React.

We nemen React als voorbeeld om een soort leidraad te hebben (bvb. jsx syntax). Je beperkt je implementatie tot een soort van 'Hello World'-ondersteuning zonder dynamische functionaliteit zoals events.
De focus van deze opdracht ligt op twee domeinen:

- Het parsen van een syntax waarin twee stijlen door elkaar voorkomen (bvb. JSX: JavaScript code + HTML-achtige code)
- Het volledige proces van broncode (bvb. JSX) naar code die door een moderne browser ondersteund wordt en een correct resultaat geeft (document.createElement).

Opdracht

- In je README.md neem je een voorbeeld op van broncode waarin alle syntactische features voorkomen die je ondersteunt. **Minimaal**:
    - Declaratie van variabelen en functies
    - Literals (number, boolean, string)
    - Functie-aanroepen, met doorgeven van argumenten
    - Geneste JSX expressies (<h1>Hello</h1>)
- In je README.md toon je de compilatie-output die uiteindelijk zou moeten worden geproduceerd door het succesvol compileren van het voorbeeld (zie vorig puntje).
- In je README.md geef je de EBNF-beschrijving van je grammatica met ondersteuning voor alle language-features uit je voorbeeld (vorig puntje).
- Je bouwt een compiler met:
    - Een duidelijke opsplitsing tussen 'parser' en 'lexer'.
        - Je mag (optioneel) gebruik maken van een parser/lexer generator.
    - Indien je een compiler plugin zou willen bouwen voor een bestaande compiler (bvb. Babel), neem je eerst contact op met je docent en maak je hier afspraken rond.
    - Opbouw van een AST en/of CST.
    - Generatie van JavaScript code die door een moderne browser uitgevoerd kan worden.
- Je implementeert een webpack loader die jouw compiler aanroept en zo de omzetting doet van .jsx naar .js.
- Je voorziet een demo-applicatie waarmee je de volledige werking kan laten zien:
    - Broncode (bvb. .jsx)
    - webpack configuratie voor jouw plugin.
    - HTML-pagina waar at runtime (bvb. bij het inladen van de pagina) HTML op verschijnt die afkomstig is uit je origineel bronbestand.
- Je voorziet volgende unit tests voor de compiler:

    - Unit tests die aantonen dat er een correcte AST of CST voortkomt uit de taalconstructies die je ondersteunt (declaratie, toekenning, aanroep, jsx-expressie, ...)
    - Enkele tests die aantonen dat syntax-fouten op een gepaste manier gerapporteerd worden.
    - End-to-end test die aantoont dat er correcte JavaScript code voortkomt uit originele broncode.
- In je README.md noteer je - in correct Markdown-formaat - het volgende:
    - Jouw naam en de naam van je framework.
    - Met welke commando's:
        - … je de verschillende projecten kan "builden". (Geef aan in welke volgorde dit moet gebeuren.)
        - … je de demo-applicatie kan uitvoeren.
        - … je de tests kan uitvoeren.

Je levert minimaal twee projecten op. Je demo-applicatie moet volledig gescheiden zijn van je webpack plugin.

Niet Vereist

Voor de duidelijkheid:

- Je hoeft geen dynamisch gedrag te voorzien (bvb. onClick="...", Hello {name} , ...)
- Je hoeft geen ondersteuning te voorzien voor React.Component en de lifecycle.

Je mag in totaal tot vier projecten opleveren, maar het moeten er minimaal slechts twee zijn.

- Je mag je compiler scheiden van je webpack plugin of je mag je compiler volledig opnemen in je webpack plugin (voor de eenvoud).
- Je mag een project (npm package) aanmaken dat functionaliteit aanbiedt voor de creatie van je HTML elementen (analoog aan 'react'/'react-dom'). Deze code roep je dan aan vanuit de gegenereerde JavaScript code. ***Of*** je mag JavaScript code genereren waarin alle HTML elementen *on-the-spot* aangemaakt worden.
- Je mag ervoor kiezen om je verschillende projecten samen in één git repository te plaatsen, in apparte directories.