Programlama Dilleri Proje Ödevi

Gerçek Zamanlı Sözdizimi

Vurgulayıcı ve Ayrıştırıcı

Melike Rana YOZGATLI

23360859019

1. Giriş

Bu proje, temel programlama dili yapılarını anlayıp işleyebilen, aynı zamanda yazılan kodu gerçek zamanlı olarak vurgulayan basit bir JavaScript tabanlı ifade ayrıştırıcı (parser) ve sözdizimi vurgulayıcı (syntax highlighter) geliştirmeyi amaçlamaktadır.

Amaç, kullanıcıların kod yazarken anlık sözdizimi kontrolü yapabilmesi ve hataların kolayca fark edilmesini sağlamaktır. Projede temel ifadeler, fonksiyon çağrıları, aritmetik işlemler gibi dil yapıları ayrıştırılır ve görsel olarak vurgulanır.

2. Programlama Dilleri ve Araçlar

Bu projede, ifade ayrıştırma ve sözdizimi vurgulaması işlemleri için JavaScript programlama dili tercih edilmiştir. JavaScript, hem web tabanlı uygulamalarda yaygın olarak kullanılmakta hem de dinamik ve etkileşimli kullanıcı arayüzleri geliştirmeye olanak sağlamaktadır. Böylece, geliştirdiğimiz sözdizimi vurgulayıcıyı doğrudan bir web tarayıcısı üzerinde çalıştırmak mümkün olmaktadır.

Kullanılan Temel Teknolojiler

- **JavaScript**: Programın temel mantığının ve ayrıştırma işlemlerinin yazıldığı dil.
- **HTML5**: Kullanıcı arayüzü oluşturmak için kullanılmıştır.
- **CSS**: Kod vurgulamasının ve kullanıcı arayüzünün görsel tasarımı için kullanılmıştır.

Araçlar ve Kütüphaneler

- **Tarayıcı**: Geliştirme ve test sürecinde kullanılan ana platform.
- **Kod Editörü (VSCode):** Geliştirme ortamı olarak kullanılmıştır.
- Proje herhangi bir harici parser kütüphanesi veya sözdizimi vurgulama kütüphanesi kullanmadan, tamamen sıfırdan yazılmıştır.

3. Dil ve Grammar Seçimi

3.1. Kullanılan Dil

Proje kapsamında, JavaScript diliyle yazılmış bir yapı oluşturulmuştur. Bu yapı, temel aritmetik işlemler, fonksiyon çağrıları ve değişken tanımlamaları gibi basit ifadelerin ayrıştırılmasını hedeflemektedir.

3.2. Dilin Temel Özellikleri

- Temel veri tipleri
- Operatörler
- Fonksiyon Çağrıları
- Parantezli ifadeler
- If-Else
- For
- Switch - Case

4. Lexical Analiz

Sözcüksel analiz, kaynak kodun atomik birimler olan token'lara (belirteçlere) dönüştürülmesi sürecidir. Bu projede, sozcukAnalizci fonksiyonu bu görevi üstlenir.

Token Tanımlama: Farklı token tipleri ve bunlara karşılık gelen CSS sınıfları belirlenmiştir. Bu tipler arasında anahtar kelimeler (genel, case ve değişken tanımlama için), tanımlayıcılar, sayılar, operatörler, boşluklar, dizeler, yorumlar ve bilinmeyen karakterler bulunur.

Düzenli İfadeler (Regex): Her bir token tipi için özel olarak tanımlanmış düzenli ifadeler (belirtecOzellikleri dizisi) kullanılır. Kaynak kod üzerinde adım adım ilerlenir ve her bir noktada en uzun eşleşen düzenli ifade aranır.

Algoritma: sozcukselAnalizci fonksiyonu, kaynak kod üzerinde karakter karakter ilerler. Her konumda, tanımlanmış düzenli ifadelerle eşleşme denenir. En uzun eşleşme bulan kısım bir token olarak kabul edilir, değeri ve tipiyle birlikte bir listeye eklenir. İmleç eşleşen tokenın sonuna ilerletilir. Eğer hiçbir düzenli ifade eşleşmezse, o karakter bilinmeyen token tipi olarak işaretlenir.

Anahtar Kelime Kontrolü: Başlangıçta bir tanımlayıcı olarak tanınan tokenlar, daha sonra tanımlı anahtar kelime listeleri (anahtarKelimeGenel, anahtarKelimeDurum, anahtarKelimeDegisken) ile karşılaştırılarak doğru anahtar kelime tipine dönüştürülür.
Bu aşama, ham metni, ayrıştırma için anlamlı birimlere dönüştürerek sözdizimi analizine zemin hazırlar.

5. Parsing

Ayrıştırma, sözcüksel analizden gelen token dizisinin, dilin gramer kurallarına uygun olup olmadığını kontrol ederek bir **Abstract Syntax Tree** oluşturma sürecidir. Bu projede **Ayristirici sınıfı** bu görevi üstlenir.

Top-Down Ayrıştırma: Ayrıştırıcı, Recursive Descent (Özyinelemeli İniş) prensibine dayalı yukarıdan aşağıya bir yaklaşımla çalışır. Her dilbilgisi kuralı (örneğin, Program, Ifade, FonksiyonTanimi, IfadeIfadesi vb.) için ayrı bir metot (programiAyristir, ifadeyiAyristir, fonksiyonTaniminiAyristir gibi) bulunur.

Kullanılan Grammer Kuralları:

<Program> ::= (<Ifade>)* <DosyaSonu>

<Ifade> ::=
  <FonksiyonTanimi>
  | <EgerIfadesi>
  | <DonusIfadesi>
  | <ForIfadesi>
  | <SwitchIfadesi>
  | <BreakIfadesi>
  | <ContinueIfadesi>
  | <DegiskenBildirimi>
  | <Atama>
  | <BosIfade>
  | <IfadeIfadesi>

<BlokIfadesi> ::= '{' (<Ifade>)* '}'

<FonksiyonTanimi> ::= 'function' <Tanimlayici> '(' [<Tanimlayici> (',' <Tanimlayici>)*] ')' <BlokIfadesi>

<DonusIfadesi> ::= 'return' [<GenelIfade>] ';'

<EgerIfadesi> ::= 'if' '(' <GenelIfade> ')' (<BlokIfadesi> | <Ifade>) ['else' (<BlokIfadesi> | <Ifade>)]

<DegiskenBildirimi> ::= ('var' | 'let' | 'const') <Tanimlayici> ['=' <GenelIfade>] ';'

<Atama> ::= <Tanimlayici> '=' <GenelIfade> ';'

<IfadeIfadesi> ::= <GenelIfade> ';'

<ForIfadesi> ::= 'for' '(' (<DegiskenBildirimi> | <IfadeIfadesi> | ';') <GenelIfade> ';' <GenelIfade> ')' <BlokIfadesi>

<BreakIfadesi> ::= 'break' ';'

<ContinueIfadesi> ::= 'continue' ';'

<SwitchIfadesi> ::= 'switch' '(' <GenelIfade> ')' '{' (<SwitchDurumu>)* '}'

<SwitchDurumu> ::=
  'case' <GenelIfade> ':' (<Ifade>)*
  | 'default' ':' (<Ifade>)*

<BosIfade> ::= ';'

<GenelIfade> ::= <KarsilastirmaIfadesi>

<KarsilastirmaIfadesi> ::= <ToplamaIfadesi> (('>' | '<' | '==' | '!=' | '<=' | '>=' | '===' | '!==')
<ToplamaIfadesi>)*

<ToplamaIfadesi> ::= <CarpmaIfadesi> (('+' | '-') <CarpmaIfadesi>)*

<CarpmaIfadesi> ::= <TemelIfade> (('*' | '/') <TemelIfade>)*

<TemelIfade> ::=
  <Sayi>
  | <Dize>
  | 'true' | 'false' | 'null' | 'undefined'
  | <CagriIfadesi>
  | <Tanimlayici>
  | '(' <GenelIfade> ')'

<CagriIfadesi> ::= <Tanimlayici> '(' [<GenelIfade> (',' <GenelIfade>)*] ')'

ilerlet Metodu: Bu kritik metot, beklenen token tipini (ve isteğe bağlı olarak değerini) kontrol eder ve token listesinde ilerler. Eğer beklenen token bulunamazsa, bir **SyntaxError** fırlatır ve hata listesine kaydeder. Bu, hata kurtarma mekanizması olmadan parse işlemini durdurur, ancak hatanın konumunu ve mesajını bildirir.

Dilbilgisi Kuralları Uygulaması:

Program Yapısı: programiAyristir metodu, dosya sonuna kadar ifadeleri ayrıştırır.

İfade Çeşitleri: ifadeyiAyristir metodu, bir sonraki tokenın tipine ve değerine bakarak hangi alt ayrıştırma metodunun çağrılması gerektiğine karar verir (değişken bildirimi, atama, if, for, switch, fonksiyon tanımı, return, break, continue veya genel ifade).

Karmaşık Yapılar:

Fonksiyon Tanımları: fonksiyonTaniminiAyristir metodu function anahtar kelimesi, fonksiyon adı, parametreler ve blok gövdesini ayrıştırır.

If/Else İfadeleri: ifIfadesiniAyristir metodu if anahtar kelimesi, koşul, ana gövde ve isteğe bağlı else bloğunu işler.

For Döngüleri: forIfadesiniAyristir metodu başlangıç, koşul ve artırma ifadelerini ve döngü gövdesini ayrıştırır.

Switch İfadeleri: switchIfadesiniAyristir metodu, case ve default durumlarını ve bu durumların altındaki ifadeleri işler.

Değişken Bildirimleri ve Atamalar: degiskenBildiriminiAyristir ve atamayiAyristir metotları var/let/const ile tanımlamaları ve = ile atamaları ayrıştırır.

İkili İfadeler: carpmaIfadesiniAyristir, toplamaIfadesiniAyristir, karsilastirmaIfadesiniAyristir gibi metotlar, operatör önceliklerine göre aritmetik ve karşılaştırma ifadelerini ayrıştırır.

Fonksiyon Çağrıları: cagrilfadesiniAyristir metodu, tanımlayıcıdan sonra gelen parantezlerle fonksiyon çağrılarını tanır ve argümanlarını ayrıştırır.

Temel İfadeler: temelIfadeyiAyristir metodu sayılar, dizeler, boolean değerler, tanımlayıcılar ve parantez içindeki ifadeler gibi en temel yapı taşlarını ayrıştırır.

Hata Yönetimi: Ayrıştırma sırasında gramer kurallarına uymayan bir durumla karşılaşıldığında, hata metodu çağrılarak hata mesajı, ilgili token ve konumu hatalar listesine eklenir.
Bu kapsamlı ayrıştırma mantığı, kullanıcının yazdığı kodun yapısal geçerliliğini denetler ve hataları bildirir.

6. Syntax Highlighting

Sözdizimi vurgulama, kodun okunabilirliğini artırmak için farklı kod elemanlarını (anahtar kelimeler, sayılar, yorumlar vb.) farklı renk ve stillerle görsel olarak ayırt etme işlemidir. Bu projede, sözcüksel analiz sonucunda elde edilen tokenlar kullanılarak gerçek zamanlı vurgulama sağlanır.

Vurgula Fonksiyonu: sozcukselAnalizci tarafından üretilen token listesini alır.

HTML Yapılandırması: Her bir token için, tipiyle eşleşen bir CSS sınıfına sahip etiketi oluşturulur. Örneğin, keyword-general sınıfına sahip bir anahtar kelime mavi ve kalın, string sınıfına sahip bir dize yeşil görünür.

Boşluk ve Yeni Satır Yönetimi: Boşluk karakterleri ()   ile, tab karakterleri (\t) dört   ile ve yeni satırlar (\n)
 etiketiyle değiştirilerek HTML içinde düzgün görüntülenmeleri sağlanır.

Özel Karakterlerden Kaçma: htmlKarakterlerdenKac yardımcı fonksiyonu, HTML özel karakterlerini uygun HTML varlıklarına dönüştürerek tarayıcının bunları metin olarak yorumlamasını sağlar, kod olarak değil.

Gerçek Zamanlı Güncelleme: duzenleyici alanındaki input olayına bağlı olan vurgulamaVeAyrismayiGuncelle fonksiyonu, kullanıcının her tuş vuruşunda veya metin değiştiğinde tetiklenir. Bu fonksiyon, metni tokenlara ayırır, vurgular ve düzenleyici alanının innerHTML'ini günceller.

İmleç Konumu Yönetimi: imlecKonumunuAyarla fonksiyonu, innerHTML güncellemesi sırasında imlecin kaybolmasını engellemek için mevcut imleç konumunu kaydeder ve güncelleme sonrası doğru yere yeniden konumlandırır. Bu, kullanıcı deneyimi açısından kritik bir detaydır.

7. GUI Tasarımı

Projenin kullanıcı arayüzü, basitlik ve işlevsellik ön planda tutularak tasarlandı. Kod yazma alanı olarak contenteditable bir HTML div kullanıldı ve CSS ile modern bir görünüm, token tiplerine göre renkli vurgulama sağlandı. Kullanıcılar kod yazarken, sözdizimi hataları anında kırmızı kenarlık ve açıklayıcı mesajla bildirilirken, geçerli kod durumunda da olumlu geri bildirim verildi. Bu gerçek zamanlı görsel geri bildirim ve özenli Enter tuşu yönetimi, kullanıcılara akıcı ve yardımcı bir kodlama deneyimi sunuyor.

8. İleri Çalışmalar

Daha kapsamlı dil desteği getirilebilir.

9. YouTube Videosu ve Sayfa Ulaşımı

Youtube videosuna ve sayfaya githubdaki Read.me dosyasından ulaşılabilir.

10. Referanslar

- W3Schools
- Regex for JavaScript
- MDN Web Docs
- Chatgpt