Parsing JWT Headers Across Programming Paradigms

Aidan Pace

[2025-04-28 Mon]

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques

What We'll Cover

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025

What We'll Cover

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review
- Base64url encoding challenges

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review
- Base64url encoding challenges
- Header parsing patterns across languages

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review
- Base64url encoding challenges
- Header parsing patterns across languages
- Functional vs object-oriented approaches

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review
- Base64url encoding challenges
- Header parsing patterns across languages
- Functional vs object-oriented approaches
- Language-specific idioms and pitfalls

Introduction

Parsing JWT Headers Across Programming Paradigms

- A cross-language exploration of JWT header parsing techniques
- SPLASH/StrangeLoop/PyConf/RacketCon/EuroLISP 2025
- Aidan Pace (@aygp-dr)

What We'll Cover

- JWT structure: quick review
- Base64url encoding challenges
- Header parsing patterns across languages
- Functional vs object-oriented approaches
- Language-specific idioms and pitfalls
- Performance considerations

JWT Basics

JWT Structure Refresher

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIn0.dozjgNryP4J

Three dot-separated base64url-encoded segments:

1. Header (algorithm & token type)
2. Payload (claims)
3. Signature

The Base64url Challenge
Standard Base64 vs Base64url encoding:

- URL-safe variant replaces + with - and / with _
- Padding (=) often omitted

Language Implementations

JavaScript (Browser)

```
const authHeader = "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOi..
const token = authHeader.split(' ')[1];

// Decode the header part
const headerPart = token.split('.')[0];
const decodedHeader = JSON.parse(atob(headerPart));
console.log(decodedHeader);
```

Note: `atob()` handles base64 but not base64url specifically

Node.js

```
// Using built-in modules
const authHeader = "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOi..
```

Common Patterns & Variations

1. Token extraction: Split by space or regex

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries
4. Error handling: Idiomatic differences

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries
4. Error handling: Idiomatic differences

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries
4. Error handling: Idiomatic differences

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries
4. Error handling: Idiomatic differences

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

Analysis

Common Patterns & Variations

1. Token extraction: Split by space or regex
2. Base64url handling:
   - Character replacement (- → +, _ → /)
   - Padding calculation
   - URL-safe decoder availability (JVM advantage)
3. JSON parsing: Native vs libraries
4. Error handling: Idiomatic differences

Cross-Language Performance Analysis

Language	Parsing Time (s)	Memory Usage (KB)
Rust	5.2	1.8
JavaScript	24.7	12.3

JWT in Production

- API Gateway token validation

JWT Flow

```
digraph {
  rankdir=LR;
  node [shape=box, style=rounded];
  Client -> "Auth Service" [label="1. Login"];
  "Auth Service" -> Client [label="2. JWT"];
  Client -> "API Gateway" [label="3. Request + JWT"];
```

Real-World Applications

JWT in Production

- API Gateway token validation
- Microservice authorization

JWT Flow

```
digraph {
  rankdir=LR;
  node [shape=box, style=rounded];
  Client -> "Auth Service" [label="1. Login"];
  "Auth Service" -> Client [label="2. JWT"];
  Client -> "API Gateway" [label="3. Request + JWT"];
```

Real-World Applications

JWT in Production

- API Gateway token validation
- Microservice authorization
- Single Sign-On implementations

JWT Flow

```
digraph {
  rankdir=LR;
  node [shape=box, style=rounded];
  Client -> "Auth Service" [label="1. Login"];
  "Auth Service" -> Client [label="2. JWT"];
  Client -> "API Gateway" [label="3. Request + JWT"];
```

Real-World Applications

JWT in Production

- API Gateway token validation
- Microservice authorization
- Single Sign-On implementations
- Mobile app authentication

JWT Flow

```
digraph {
  rankdir=LR;
  node [shape=box, style=rounded];
  Client -> "Auth Service" [label="1. Login"];
  "Auth Service" -> Client [label="2. JWT"];
  Client -> "API Gateway" [label="3. Request + JWT"];
```

Conclusion

Takeaways

1. Base64url encoding requires special attention

Questions?
Thank you!

Slides & examples available at: github.com/aidan-pace/jwt-parsing-examples

Conclusion

Takeaways

1. Base64url encoding requires special attention
2. Each language has idiomatic parsing advantages

Questions?
Thank you!

Slides & examples available at: github.com/aidan-pace/jwt-parsing-examples

Conclusion

Takeaways

1. Base64url encoding requires special attention
2. Each language has idiomatic parsing advantages
3. Functional approaches shine for transformation pipelines

Questions?
Thank you!

Slides & examples available at: github.com/aidan-pace/jwt-parsing-examples

Conclusion

Takeaways

1. Base64url encoding requires special attention
2. Each language has idiomatic parsing advantages
3. Functional approaches shine for transformation pipelines
4. Libraries save time but understanding internals matters

Questions?
Thank you!

Slides & examples available at: github.com/aidan-pace/jwt-parsing-examples

Conclusion

Takeaways

1. Base64url encoding requires special attention
2. Each language has idiomatic parsing advantages
3. Functional approaches shine for transformation pipelines
4. Libraries save time but understanding internals matters
5. Consider performance for high-volume applications

Questions?
Thank you!

Slides & examples available at: github.com/aidan-pace/jwt-parsing-examples