**Lyra / Astra 통합 기술 명세서 v2.0**

**Version: 2.0 (2025-10-16)**

**Author: Uzu Lee**

**Ⅰ. 서문 (Preface)**

**본 문서는 Lyra / Astra 언어 체계의 공식 기술 명세서로서, \*\*'살아있는 문서(Living Document)'\*\*의 구조, 구문, 동작 원리 및 구현 요구사항을 정의합니다.**

* **Lyra는 문서를 "의미 기반 구조체"로 다루는 차세대 DSL(Domain-Specific Language)입니다.**
* **Astra는 Lyra의 타입, 관계, 패널, 테마 등을 정의하는 메타 언어로 작동합니다.**

**Lyra/Astra는 단순한 마크업 언어가 아니라, 인간이 작성한 텍스트를 기계와 AI가 의미적으로 해석하고 재구성할 수 있는 구조적 문서 언어를 목표로 합니다. v2.0은 기존의 완성도 높은 이론에 개발자 친화적인 생태계와 실용적인 문법 확장을 더하여, 단순한 문서 중심 DSL에서 완전한 개발자 친화형 메타 언어로의 진화를 목표로 합니다.**

**Ⅱ. 언어 구조 (Language Architecture)**

**Lyra/Astra 언어는 다음의 3계층 구조를 기반으로 합니다.**

|  |  |  |
| --- | --- | --- |
| **계층** | **이름** | **설명** |
| **L1** | **Narrative Layer** | **인간이 읽고 쓰는 자연어 서술 계층. 문단, 주석 등의 비구조적 콘텐츠를 포함합니다.** |
| **L2** | **Structural Layer** | **문서의 데이터 구조와 관계를 정의하는 구문 계층. @panel, @graph 등이 포함됩니다.** |
| **L3** | **Semantic Layer** | **문서의 의미, 감정, 의도, 주제 등을 정의하는 메타 계층. AI와 기계 해석을 위한 정보입니다.** |

**각 계층은 파서에 의해 통합되어 단일 \*\*AST(Abstract Syntax Tree)\*\*로 변환됩니다. 모든 문서는 @document 블록으로 시작하며, 그 안에 메타데이터와 세부 블록이 포함됩니다.**

**@document "orion/exhibition-2031"**

**title: "오리온 전시회 2031"**

**date: 2031-04-21**

**locale: ko-KR**

**@end**

**Ⅲ. 문법 체계 및 구문 정의 (Syntax and Grammar)**

**Lyra의 문법은 가독성과 명시성을 중심으로 설계되었습니다. 모든 구조는 @로 시작하여 @end로 종료되는 블록 형태를 가집니다.**

**1. 기본 구조 및 단축 문법**

* **기본 블록: @ 식별자로 시작하며, 속성이나 하위 블록을 포함하고 @end로 종료됩니다.**
* **단축 문법 (v2.0): 간단한 블록은 @를 사용하여 한 줄로 종료할 수 있습니다.**

**# 기본 문법**

**@panel introduction**

**title: "Lyra의 기본 구조"**

**description: "모든 블록은 '@'로 시작하고 '@end'로 끝난다."**

**@end**

**# 단축 문법 (v2.0)**

**@panel quick.note @ title="단축 구문 테스트" text="간단 구문 적용" @**

**2. 속성(Properties)과 주석(Comments)**

* **속성: Key: Value 형태로 정의됩니다. (v2.0부터 property=value 형태도 선택적으로 지원)**
* **주석: #로 한 줄 주석, ## ... ##로 여러 줄 주석을 사용합니다.**

**@config general**

**author: "Orion Systems" # 한 줄 주석**

**version: 1.2.4**

**##**

**이 블록은**

**여러 줄 주석입니다.**

**##**

**@end**

**3. 중첩 구조와 스코프 (Nested Blocks & Scope)**

**블록은 다른 블록을 포함할 수 있으며, 이는 계층적 데이터 표현을 가능하게 합니다. 내부 블록은 상위 블록의 속성에 접근할 수 있으나, 역참조는 허용되지 않습니다.**

**@panel system.config**

**name: Core Settings**

**@subpanel database**

**type: SQL**

**host: db.local**

**# 'name' 속성에 접근 가능**

**@end**

**@end**

**4. 참조(References) 단축 문법**

**다른 문서나 노드를 참조할 때 ref: 대신 @를 사용할 수 있습니다.**

* **원래 문법: ref:document.id**
* **단축 문법 (v2.0): @document.id**

**5. 확장 문법 블록 (v2.0)**

**개발자 중심 콘텐츠 작성을 위해 테이블, 수학 수식, 코드 블록 등을 지원합니다.**

**# 테이블 예시**

**@table dataset.summary**

**headers: ["Name", "Type", "Count"]**

**rows:**

**- ["User", "entity", 1200]**

**- ["Post", "text", 5400]**

**@end**

**# 수학 수식 예시 (LaTeX 호환)**

**@math formula.energy @ latex="E = mc^2" @**

**# 코드 블록 예시**

**@code javascript.example**

**lang: javascript**

**content: """**

**function greet() {**

**console.log("Hello, Lyra!");**

**}**

**"""**

**@end**

**6. 오류 처리 규칙**

**파서는 오류 발생 시 라인 번호와 구문 컨텍스트를 함께 출력해야 하며, 다음 규칙을 준수합니다.**

* **모든 블록은 @end로 종료되어야 합니다.**
* **속성 키는 중복될 수 없습니다.**
* **예약어는 식별자로 사용할 수 없습니다.**
* **참조(@ 또는 ref:) 대상이 존재해야 합니다.**

**Ⅳ. 데이터 타입 및 타입 시스템**

**Lyra는 데이터의 일관성을 위해 정형화된 타입 시스템을 사용합니다.**

**1. 기본 타입 (Primitive Types)**

|  |  |  |
| --- | --- | --- |
| **타입명** | **설명** | **예시** |
| **text** | **문자열** | **"예시 문자열"** |
| **number** | **정수 또는 실수** | **42, 3.14** |
| **boolean** | **참/거짓 값** | **true, false** |
| **date** | **ISO8601 날짜** | **2031-06-11** |
| **datetime** | **날짜 및 시간** | **2031-06-11T14:30:00Z** |
| **duration** | **시간 간격 (ISO8601)** | **P2Y6M5DT12H30M5S** |
| **ref** | **다른 문서/노드 참조** | **@module.id** |
| **null** | **값이 없음** | **null** |

**2. 컬렉션 타입 (Collection Types)**

**배열(Array) 및 맵(Map) 형태의 복합 자료형을 지원합니다.**

**@panel attendees**

**list: ["Aria", "Rion", "Seren"]**

**metadata: { role: "speaker", verified: true }**

**@end**

**3. 확장 타입 (Extended Types)**

**Astra 모듈에서 @type 블록을 통해 새로운 타입을 정의하고 제약 조건을 추가할 수 있습니다. (자세한 내용은 Ⅴ장 참고)**

**Ⅴ. Astra 메타언어 및 정의 시스템**

**Astra(.astra)는 Lyra 문서에서 사용하는 타입, 패널, 테마 등을 정의하는 메타 언어입니다. Lyra가 '내용'이라면 Astra는 '형식'과 '의미'를 담당합니다.**

**1. Astra 파일 구조**

**@module astra.common.ui**

**# 커스텀 타입 정의**

**@type ui.font.size**

**base: number**

**constraint: value >= 8 && value <= 72**

**default: 14**

**@end**

**# 커스텀 패널 정의**

**@panel card.info**

**title: text**

**subtitle: text**

**slot: image.display**

**@end**

**# 커스텀 테마 정의**

**@theme ui.dark**

**background: "#111111"**

**text: "#eeeeee"**

**accent: "#4488ff"**

**@end**

**@end**

**2. 핵심 기능**

* **타입 정의 (@type): 새로운 데이터 타입을 생성하고 constraint로 유효성 검증 규칙을 추가합니다.**
* **패널 정의 (@panel): 재사용 가능한 UI 또는 논리적 구성요소를 정의합니다.**
* **테마 정의 (@theme): 문서의 색상, 폰트 등 시각적 스타일을 정의합니다.**
* **상속 (@extend): 기존 Astra 정의를 확장하여 새로운 정의를 만듭니다.**
* **모듈 관리 (@import/@export): 다른 Astra 파일의 정의를 가져오거나 내보냅니다.**

**Ⅵ. 핵심 시스템 (Core Systems)**

**1. 시간 시스템 (Chrono 2.0)**

**문서의 생성, 수정, 표시 시점을 정밀하게 제어합니다. 작성자(author)와 열람자(viewer)의 시간대를 모두 인식하여 시간을 자동으로 변환하고 표시합니다.**

* **시간 객체 (@time): value, zone, policy, format 속성을 가집니다.**
* **시간 정책 (Time Policy):**
  + **author: 작성자 지역 시간 고정**
  + **viewer: 열람자 기준으로 자동 변환**
  + **fixed: 절대 UTC 기준**

**@time schedule.opening**

**value: 2031-04-21T09:00:00Z**

**zone: Asia/Seoul**

**policy: viewer**

**@end**

**2. 그래프 시스템 (Graph System)**

**문서 간의 관계를 노드(Node)와 간선(Edge)으로 시각화합니다. @ 구문은 자동으로 노드 간 연결을 생성합니다.**

**@graph knowledge.web**

**node: doc.alpha @ title="서론" @**

**node: doc.beta @ title="본문" @**

**link: doc.alpha -> doc.beta**

**label: "references"**

**type: semantic**

**weight: 0.8**

**@end**

**@end**

**3. 의미 계층 (Semantic Layer) 및 AI 협업**

**문서의 의도(Intent), 감정(Emotion), 주제(Topic), 개체(Entity) 등을 정의하여 AI가 문맥을 이해하게 합니다. @ai 블록을 통해 요약, 번역, 분석 등의 작업을 문서 내에서 직접 지시할 수 있습니다.**

* **의미 정의 (@semantic): 문서의 목적, 분위기, 주제 등을 명시합니다.**
* **AI 블록 (@ai): context, instruction, mode, output을 지정하여 AI 작업을 정의합니다.**

**@entity system.alpha**

**type: software**

**name: Lyra Parser**

**@end**

**@semantic report.analysis**

**intent: evaluation**

**emotion: neutral**

**topic: software.performance**

**entity: @system.alpha**

**@end**

**@ai auto.summary**

**context: @report.analysis**

**instruction: "이 보고서를 300자 이내로 요약하라."**

**mode: summarize**

**output: @report.summary**

**@end**

**4. 버전 및 시간축 관리 시스템**

**Git과 유사하게 문서의 수정 이력, 버전 간 차이, 변경 로그를 구조적으로 추적하고 관리합니다.**

* **버전 블록 (@version): 버전 정보, 작성자, 타임스탬프, 변경 요약을 포함합니다.**
* **변경 로그 (@changelog): 버전별 added, removed, modified 내용을 기록합니다.**
* **타임라인 (@timeline): 문서의 모든 주요 이벤트를 시간순으로 시각화합니다.**

**@version v1.3.2**

**author: "Orion Systems"**

**timestamp: 2031-05-21T09:30:00Z**

**based\_on: @v1.3.1**

**summary: "타임라인 기능 추가 및 Astra 호환성 개선."**

**@end**

**@changelog v1.3.2**

**added: ["chrono.timeline", "graph.autolink"]**

**modified: ["ai.feedback"]**

**@end**

**Ⅶ. 렌더링 및 표현 계층 (Rendering and Presentation Layer)**

**문서의 시각적 출력, 상호작용, 반응형 레이아웃을 관리합니다.**

* **렌더링 파이프라인: Parsing → Normalization → Styling → Layout → Presentation 단계를 거칩니다.**
* **테마 시스템: Astra로 정의된 @theme를 기반으로 렌더링되며, 다크/라이트 모드를 자동 지원합니다.**
* **인터랙션 (@interact): 클릭, 호버 등의 이벤트와 애니메이션(fade-in, slide-up 등)을 선언적으로 정의합니다.**
* **출력 모드: web(HTML), static(PDF), terminal, markdown 등 다양한 포맷을 지원합니다.**
* **접근성: WCAG 2.2 표준을 준수하여 텍스트 대비, 키보드 포커스 등을 보장합니다.**

**@interact panel.expand**

**trigger: onClick**

**action: toggle(@panel.detail)**

**animation: fade-in 0.3s ease**

**@end**

**Ⅷ. 개발자 생태계 및 도구 (Developer Ecosystem & Tooling, v2.0)**

**1. IDE 통합 및 LSP 지원**

**LSP(Language Server Protocol) 기반 확장을 통해 VSCode, JetBrains 등 주요 IDE에서 아래 기능을 공식 지원합니다.**

* **구문 하이라이팅 (Syntax Highlighting)**
* **자동 완성 (Autocomplete)**
* **실시간 오류 검출 및 힌트**
* **참조 추적 (@ 구문 기반)**

**2. SDK 및 CLI 도구**

|  |  |  |
| --- | --- | --- |
| **도구명** | **언어** | **주요 기능** |
| **lyra-sdk** | **Python / TypeScript** | **파서 및 AST 접근 API** |
| **astra-tools** | **Node.js** | **Astra 정의 검증, 모듈 병합** |
| **lyra-cli** | **CLI** | **문법 검사, 렌더링, 빌드, 버전 관리** |

**# CLI 사용 예시**

**$lyra build main.lyra --out dist/$ lyra render main.lyra --format html**

**$ lyra commit -m "update semantic layer"**

**3. 개발자 지원 도구**

|  |  |  |
| --- | --- | --- |
| **도구** | **목적** | **설명** |
| **VSCode Plugin** | **편집기 통합** | **하이라이팅, 구조 트리, 블록 접기 지원** |
| **RenderKit** | **출력 엔진** | **HTML, PDF, Markdown, Terminal 렌더러** |
| **Astra Studio** | **시각 관리 UI** | **모듈 관리, 그래프 시각화** |

**Ⅸ. 성능 및 보안 (Performance & Security, v2.0)**

**1. 성능 최적화**

**대규모 문서를 효율적으로 처리하기 위해 다음 기능을 도입합니다.**

* **스트리밍 파서 (Streaming Parser): 메모리 사용량을 40% 감소시킵니다.**
* **병렬 구문 분석 (Parallel Parsing): 멀티스레드를 활용하여 파싱 속도를 높입니다.**
* **지연 렌더링 (Lazy Rendering): 화면에 보이는 부분만 렌더링하여 초기 로딩 속도를 개선합니다.**

**2. 보안 정책**

* **리소스 로드: 외부 리소스(이미지, 아이콘 등)는 HTTPS 프로토콜만 허용합니다.**
* **AI 실행: @ai 블록 실행 시 모든 활동을 @log에 필수로 기록합니다.**
* **렌더링: 웹 렌더링 시 CSP(Content Security Policy)를 활성화하여 XSS 공격을 방지합니다.**
* **SDK: 모든 파일 접근은 샌드박스 환경에서 수행됩니다.**

**부록 (Annex)**

**A.1 표준 라이브러리 (Standard Library)**

|  |  |  |
| --- | --- | --- |
| **구분** | **이름** | **설명** |
| **타입** | **color.hex, id.ref, text.rich** | **색상 코드, 문서 참조 ID, 서식 포함 텍스트 등** |
| **패널** | **panel.location, panel.calendar** | **위치 정보, 캘린더 패널** |
| **관계** | **references, depends\_on** | **인용, 기능적 의존 관계** |
| **테마** | **ui.light, ui.dark, ui.paper** | **밝은/어두운/인쇄용 테마** |

**A.2 확장 문법 샘플 (Extended Syntax Examples)**

**@document travel.journal**

**title: "유럽 여행 일지"**

**author: Luna**

**@use ui.dark from "astra/common/ui.astra"**

**@apply ui.dark**

**@panel location**

**name: Colosseum**

**address: Rome, Italy**

**link: "[https://goo.gl/maps/](https://goo.gl/maps/)..."**

**@end**

**@semantic document.journal**

**intent: narrate**

**emotion: reflective**

**@end**

**@end**

**A.3 구현자 참고 사양 (Developer Appendix)**

* **파서 구조: Lexer -> Parser -> Validator -> Renderer**
* **파일 시스템: main.lyra, src/\*.astra, assets/\* 구조 권장**
* **표준 에러 코드:**
  + **E001: 블록 종료 태그 (@end) 누락**
  + **E002: 중복된 식별자 선언**
  + **E003: 잘못된 타입 또는 제약 조건 위반**
  + **E004: 참조 대상 (ref: / @)이 존재하지 않음**
  + **E005: 외부 모듈 로딩 실패**

**문서 종료**