Lit Playground 학습 항목 정리 (HTML5 / CSS / JS 관점)

이 문서는 lit.dev/playground의 목차를 **HTML5, plain CSS, JavaScript 관점**에서 무엇을 배우는 단계인지 구조적으로 설명합니다.

---

1. Basics

Hello World

의미: Custom Element + Shadow DOM + Template 렌더링의 최소 단위

- HTML 관점: `<my-element></my-element>` 커스텀 태그
- CSS 관점: Shadow DOM 내부 스타일 격리
- JS 관점: `customElements.define()` + `render()`

👉목적: Web Component의 존재 이유와 Lit의 역할 이해

Full component

의미: 실제 UI 컴포넌트의 완전한 구조

- HTML: semantic tag, slot 포함 구조
- CSS: component-local layout
- JS: state, event, lifecycle

👉목적: 단일 파일 컴포넌트 모델 습득

---

2. Reactive properties

Basic properties

의미: state → DOM 자동 반영

- HTML: 텍스트/속성 바인딩
- JS: `@property()` 또는 static properties

👉React의 state와 동일 개념

Change detection

의미: 언제 re-render 되는가

- JS: setter, equality check, batching

👉불필요한 렌더 방지 메커니즘 이해

Custom attribute converter

의미: HTML attribute → JS 타입 변환

- HTML: `<my-el count="3">`
- JS: string → number/boolean/object

👉DSL, server-rendered HTML 연동 시 핵심

---

3. Template concepts

Expression types

의미: template 안에서 가능한 표현식 종류

- HTML: text, attribute, property, boolean
- JS: `${}` 위치에 따라 의미 달라짐

👉보안 + 성능 최적화 목적

Conditional templates

의미: if 문에 해당

- HTML: DOM 자체를 넣었다 뺐다
- JS: 삼항 연산, when

👉display:none 과 구조적 제거의 차이

Repeating templates

의미: for-loop rendering

- HTML: list item 생성
- JS: array.map

👉key 관리 필요성 (diff 알고리즘)

Slotting children

의미: Web Component 콘텐츠 투입

- HTML: `<slot>`
- CSS: ::slotted

👉React children 대응 개념

Element composition

의미: 컴포넌트 중첩

- HTML: tag nesting
- JS: component hierarchy

👉DSL → component tree 매핑 핵심

Template composition

의미: template 함수 재사용

- JS: render fragment factory

👉partial UI abstraction

---

4. Directives

의미: 템플릿 엔진 수준의 고급 제어 API

JS 코드가 DOM 생성 방식을 직접 제어

---

asyncAppend / asyncReplace

의미: streaming UI

- JS: async iterator
- HTML: 점진적 DOM 삽입

👉infinite scroll, log stream

---

cache

의미: DOM 재사용

👉expensive subtree rerender 방지

---

classMap

의미: class 토글

- HTML: class attribute
- JS: object → class string

👉Tailwind 대체 로직 핵심

---

guard

의미: 조건 동일 시 렌더 스킵

👉memoization 역할

---

ifDefined

의미: attribute 제거 제어

👉undefined vs empty string 차이 해결

---

live

의미: input value DOM 우선

👉controlled vs uncontrolled input 문제 해결

---

ref

의미: DOM element 직접 접근

- JS: querySelector 대체

👉canvas, focus, size 측정

---

repeat

의미: key 기반 list diff

👉성능 핵심

---

styleMap

의미: inline style 객체화

👉animation, dynamic layout

---

templateContent

의미: raw template 재사용

👉SSR hydration 대응

---

unsafeHTML / unsafeSVG

의미: raw HTML 삽입

👉XSS 위험 — CMS, Markdown 렌더링

---

until

의미: promise fallback UI

👉skeleton loading

---

5. Managing Data

Async Tasks

의미: fetch + cancel + retry 관리

- JS: AbortController

👉component 생명주기와 async 연동

---

6. Context

Context Basics

의미: 전역 상태 공유

- HTML: prop drilling 제거
- JS: dependency injection

👉React Context 동일 개념

Context Consume and Provide

의미: provider / consumer 관계

👉theme, user, locale

---

7. Signals

의미: fine-grained reactive primitive

- JS: observable variable
- DOM: 부분 업데이트

👉Solid.js, Vue reactivity와 유사

한 줄 요약

Lit Playground 목차는 단순 API 나열이 아니라:

1. Web Component 표준
2. HTML Template 엔진
3. DOM diff 전략
4. 비동기 UI 패턴
5. 상태 공유 구조

를 **프레임워크 없이 직접 다루는 법**을 단계적으로 가르치는 구조입니다.

DSL 설계와의 직접 연결 포인트

Lit 개념	DSL 매핑
Reactive properties	state 변수
Conditional template	when / if
repeat	foreach
slot	children 영역
classMap/styleMap	style rule mapping
context	global config / theme
directives	behavior primitives

즉, Lit은 **UI DSL의 중간 표현(IR)** 로 매우 적합합니다.

필요하시면 다음 단계로:

- "DSL → Lit template → HTML" 컴파일 파이프라인 구조
- Lit AST 변환 전략
- Nix 기반 DSL 빌드 체인 연결

까지 이어서 설명드릴 수 있습니다.