SVELTE 빠른 참조

컴포넌트, 반응성, 스토어, 트랜지션, SvelteKit

컴포넌트

기본 컴포넌트

```
<script>
  let name = "world";
</script>
<h1>Hello {name}!</h1>
<style>
  h1 { color: purple; }
</style>
```

컴포넌트 구조

`<script>`	컴포넌트 로직 (JS/TS)
마크업	`{표현식}`이 있는 HTML 템플릿
`<style>`	범위가 지정된 CSS (컴포넌트에 자동 범위)
`<script context="module">`	인스턴스가 아닌 모듈당 한 번 실행

반응성

반응형 할당

```
<script>
  let count = 0;
  function increment() { count += 1; } // 재렌더링 트리거
</script>
<button on:click={increment}>{count}</button>
```

반응형 선언

```
<script>
  let width = 10;
  let height = 5;
  $: area = width * height;       // 의존성 변경 시 재계산
  $: console.log("area is", area); // 반응형 문장
</script>
```

$:는 반응형 선언과 문장을 표시

반응형 규칙

할당 트리거	`count += 1`이 업데이트 트리거; `obj.x = 1`도 동일
배열 변이	`arr = [...arr, item]` 사용 (트리거를 위해 재할당)
`$:` 선언	참조된 변수 변경 시 자동 재계산
`$:` 문장	반응형으로 사이드 이펙트 실행

Props

Props 선언 및 전달

```
<!-- Child.svelte -->
<script>
  export let name;
  export let greeting = "Hello";  // 기본값
</script>
<p>{greeting}, {name}!</p>

<!-- Parent.svelte -->
<Child name="Alice" />
```

스프레드 Props

```
<script>
  const props = { name: "Alice", greeting: "Hi" };
</script>
<Child {...props} />
```

이벤트

DOM 이벤트

```
<button on:click={handleClick}>Click</button>
<input on:input={(e) => value = e.target.value} />
<form on:submit|preventDefault={handleSubmit}>
```

이벤트 수정자

`preventDefault`	`e.preventDefault()` 호출
`stopPropagation`	이벤트 버블링 중지
`once`	핸들러가 한 번만 실행
`self`	`event.target`이 요소인 경우에만
`capture`	캡처 단계 사용

컴포넌트 이벤트

```
<!-- Child.svelte -->
<script>
  import { createEventDispatcher } from "svelte";
  const dispatch = createEventDispatcher();
</script>
<button on:click={() => dispatch("greet", { text: "hi" })}>

<!-- Parent.svelte -->
<Child on:greet={(e) => alert(e.detail.text)} />
```

바인딩

양방향 바인딩

```
<input bind:value={name} />
<input type="checkbox" bind:checked={agreed} />
<select bind:value={selected}>
  <option value="a">A</option>
</select>
```

요소 및 컴포넌트 바인딩

```
<div bind:this={element}></div>
<canvas bind:clientWidth={w} bind:clientHeight={h}></canvas>
<Child bind:value={childValue} />
```

바인딩 유형

`bind:value`	Input/select/textarea 값
`bind:checked`	체크박스 상태
`bind:group`	라디오/체크박스 그룹
`bind:this`	DOM 요소 참조
`bind:clientWidth/Height`	읽기 전용 요소 크기

스토어

쓰기 가능한 스토어

```
// store.js
import { writable } from "svelte/store";
export const count = writable(0);

// 컴포넌트 — $로 자동 구독
<script>
  import { count } from "./store.js";
</script>
<button on:click={() => $count += 1}>{$count}</button>
```

스토어 메서드

```
count.set(10);              // 값 설정
count.update(n => n + 1);   // 현재값에서 업데이트
const unsub = count.subscribe(v => console.log(v));
```

스토어 유형

`writable(val)`	읽기-쓰기 스토어
`readable(val, fn)`	읽기 전용, 시작 함수로 설정
`derived(stores, fn)`	다른 스토어에서 계산
`$store`	컴포넌트의 자동 구독 문법

트랜지션

내장 트랜지션

```
<script>
  import { fade, slide, fly } from "svelte/transition";
  let visible = true;
</script>
{#if visible}
  <div transition:fade>페이드 인/아웃</div>
  <div in:fly={{ y: 200 }} out:fade>플라이 인, 페이드 아웃</div>
{/if}
```

트랜지션 옵션

`fade`	불투명도 0에서 1
`fly`	x/y 오프셋 + 불투명도 애니메이션
`slide`	슬라이드 인/아웃 (높이)
`scale`	스케일 및 페이드
`draw`	SVG 경로 스트로크 애니메이션
`duration`	트랜지션 시간 (ms)
`delay`	시작 전 지연

슬롯

기본 및 이름 있는 슬롯

```
<!-- Card.svelte -->
<div class="card">
  <slot name="header">기본 헤더</slot>
  <slot>기본 내용</slot>
</div>

<!-- 사용 -->
<Card>
  <h2 slot="header">제목</h2>
  <p>본문 내용</p>
</Card>
```

슬롯 Props

```
<!-- List.svelte -->
{#each items as item}
  <slot {item} index={item.id} />
{/each}

<!-- 사용 -->
<List {items} let:item let:index>
  <p>{index}: {item.name}</p>
</List>
```

컨텍스트

컨텍스트 설정 및 가져오기

```
<!-- Parent.svelte -->
<script>
  import { setContext } from "svelte";
  setContext("theme", { color: "dark" });
</script>

<!-- Descendant.svelte -->
<script>
  import { getContext } from "svelte";
  const theme = getContext("theme"); // { color: "dark" }
</script>
```

컨텍스트 vs 스토어

컨텍스트	컴포넌트 트리 범위, 기본적으로 반응형 아님
스토어	전역, 반응형, 어디서나 임포트 가능
컨텍스트 + 스토어	범위가 지정된 반응성을 위해 컨텍스트를 통해 스토어 전달

SvelteKit 기본

파일 기반 라우팅

```
src/routes/
  +page.svelte          <!-- / -->
  about/+page.svelte    <!-- /about -->
  blog/[slug]/+page.svelte  <!-- /blog/:slug -->
```

로드 함수

```
// +page.js (클라이언트 및 서버에서 실행)
export async function load({ params, fetch }) {
  const res = await fetch(`/api/posts/${params.slug}`);
  return { post: await res.json() };
}
```

주요 파일

`+page.svelte`	페이지 컴포넌트
`+page.js / +page.ts`	클라이언트/유니버설 로드 함수
`+page.server.js`	서버 전용 로드 / 폼 액션
`+layout.svelte`	공유 레이아웃 래퍼
`+error.svelte`	에러 페이지
`+server.js`	API 엔드포인트 (GET, POST, ...)