

시작하기 및 CLI (Ember CLI)

Ember.js는 생산성 높은 관례 기반(Convention over Configuration) 엔터프라이즈 SPA 프레임워크입니다.

- 프로젝트 생성: `npx ember-cli new my-app`
- 개발 서버 실행: `ember serve` (포트 4200)
- 컴포넌트 생성: `ember generate component user-card`
- 라우트 생성: `ember generate route posts`
- 서비스 생성: `ember generate service auth`
- 테스트 실행: `ember test/ember test --server`
- 프로덕션 빌드: `ember build -prod`

글리머 컴포넌트 (Glimmer Components)

`@tracked` 데코레이터를 통한 초경량 반응형 시스템을 제공합니다.

JavaScript 컴포넌트 클래스 (app/components/counter.js)

```
import Component from '@glimmer/component';
import { tracked } from '@glimmer/tracking';
import { action } from '@ember/object';

export default class CounterComponent extends Component {
  @tracked count = 0;

  get doubleCount() {
    return this.count * 2;
  }

  @action
  increment() {
    this.count++;
  }
}
```

Handlebars 템플릿 (app/components/counter.hbs)

```
<div class="counter">
  <p>현재 값: {{this.count}} (2배: {{this.doubleCount}})</p>
  <button type="button" {{on "click"
```

```
this.increment}}>+1 증가</button>
  <p>전달받은 제목: {{@title}}</p>
</div>
```

템플릿 문법 (Handlebars Syntax)

- 프로퍼티 출력: `{{this.propertyName}}`
- 인자(Args) 접근: `{{@argumentName}}`
- 이벤트 리스너: `{{on "click" this.handleClick}}`
- 조건문: `{{#if this.isLoggedIn}} ... {{/if}}`
- 반복문:


```
<ul>
  {{#each this.items as |item index|}}
    <li>{{index}}: {{item.name}}</li>
  {{/each}}
</ul>
```
- 라우트 링크: `<LinkTo @route="posts.show" @model={{post.id}}>보기</LinkTo>`

라우팅 및 모델 (Routing & Model Hook)

URL 변경에 따른 데이터 로드와 뷰 전환을 담당합니다.

```
// app/routes/posts.js
import Route from '@ember/routing/route';
import { service } from '@ember/service';

export default class PostsRoute extends Route {
  @service store;

  async model(params) {
    // Ember Data 스토어에서 데이터 조회
    return this.store.findAll('post');
  }
}
```

엠버 데이터 (Ember Data & Models)

모델 정의 (app/models/post.js)

```
import Model, { attr, belongsTo,
  hasMany } from '@ember-data/model';
```

```
export default class PostModel extends Model {
  @attr('string') title;
  @attr('string') content;
  @attr('date') createdAt;

  @belongsTo('user', { async: true,
    inverse: 'posts' }) author;
  @hasMany('comment', { async: true,
    inverse: 'post' }) comments;
}
```

Store CRUD 연산

- 전체 조회: `const posts = await this.store.findAll('post');`
- 단일 조회: `const post = await this.store.findRecord('post', id);`
- 쿼리 필터: `const results = await this.store.query('post', { tag: 'tech' });`
- 레코드 생성:


```
const newPost =
  this.store.createRecord('post',
    { title: '새 글' });
await newPost.save();
```
- 삭제: `await post.destroyRecord();`

서비스 및 의존성 주입 (Services)

```
// app/services/auth.js
import Service from '@ember/service';
import { tracked } from '@glimmer/tracking';

export default class AuthService extends Service {
  @tracked currentUser = null;
  login(user) { this.currentUser = user; }
}

// 컴포넌트나 라우트에서 주입
import { service } from '@ember/service';
// ...
@service auth;
```