

Angular モダン Web 開発 セミナー教材

2018 年 1 月 11 日 1.4.16 版

スタッフネット株式会社

1	テキスト(パワーポイント 2 画面印刷)
2	実習環境インストール手順
3	複利計算アプリ実習手順
4	顧客管理システム解説 (フロントエンド、バックエンド)

テキスト目次

1.概要	
1-1.Angular とモダン Web [9]
HTML5 による機能拡張	
HTML5 で未解決の課題	
モダン Web による解決	
モダン Web のシステム概念図	
サンプルアプリ	
複利計算アプリを体験	
モダン Web の分散処理	
SPA(Single Page Application)	
分散処理の課題	
分散処理の課題を解決	
モダン Web の将来	
モダン Web のまとめ	
1-2.Angular の特徴 [23]
Why Angular?	
モダン Web 開発のハードル	
モダン Web ブラウザが標準へ	
TypeScript	
Angular (プラス要素)	
Angular (マイナス要素)	
幅広い機能	
開発作業の自動化	
Angular 特徴のまとめ	
Angular でモダン Web 開発がカンタンに	
同様のソフトウェアとの比較	
2.基礎知識	
2-1.npm によるパッケージ管理 [39]
従来のライブラリ利用方法	
npm によるパッケージ管理	
npm 関連の用語	
npm インストールコマンド	
package.json による依存関係解決	

package.json の内容	
2-2.Angular プロジェクトの作成 [46]
Angular 開発の概要	
前提ソフトのインストール	
Angular プロジェクトの生成	
Visual Studio Code ターミナル・ウィンドウ	
新規プロジェクトの生成を体験	
Angular プロジェクトのフォルダ構造	
Angular プロジェクトのビルドと実行	
ビルドと実行を体験	
Angular のビルド処理	
ビルドの実行	
ビルド結果の展開	
2-3.開発ツール [59]
IDE とドキュメント生成ツール	
Angular CLI Proxy 機能	
テストツール	
単体テストの実行	
単体テストサンプルシナリオ	
初期値設定	
テスト用コンポーネントの生成	
テストの実施と評価 (計算機能)	
テストの実施と評価 (保存機能)	
単体テスト実行結果	
E2E テストの実行	
E2E テストシナリオ	
ページごとのプロパティとメソッド定義	
初期値	
前処理と後処理	
テストの実施と評価(計算機能)	
テストの実施と評価(保存機能)	
E2E テストの実行結果	
Chrome デベロッパーツール	
2-4.開発環境のバージョン固定 [86]
Angular のバージョンアップ	
Angular のバージョン固定	

2-5.Angular の構成要素 [86]
Angular アプリの構成要素	
コンポーネントの構成	
コンポーネントの記述	
コンポーネントの処理フロー	
データバインド	
子コンポーネント	
コンポーネントのライフサイクル	
その他の構成要素	
サービスの登録	
サービスの利用	
ルーターの機能	
ディレクティブとパイプの例	
2-6.テストプログラムで動作確認 [101]
テストプログラムの全体像	
フォルダ構造	
テストプログラムの実行	
外部から見た動作	
2 ページ目コード	
HTML 出力確認	
2 ページ目 HTML 出力	
1 ページ目 HTML テンプレート	
親子コンポーネント	
親子コンポーネントの連携	
CSS のカプセル化(page01)	
カプセル化のしくみ(page01)	
データバインドの副作用	
データバインドの副作用検知機能	
起動シーケンス	
モジュール定義の内容(app.module.ts)	
2-7.Angular を支える技術 [121]
関連技術	
分散処理の基盤	
Application Cache	
Web Storage	
Non SQL データベース	

- ブラウザ内ストレージの使い分け
- 非同期通信
- セッションを使用しないユーザー認証
- ユーザー認証の比較
- トークン(JWT)を用いた認証
- JWT によるシングルサインオン
- ng2-Bootstrap
- material2
- Covalent
- モバイル対応
- グリッドレイアウト
- タブ切り替え
- Angular 対応モバイル UI ライブラリ
- データビジュアライゼーション
- Angular の技術情報

2-8.データの同期 [147]

- データ同期処理のタイプ
- 更新頻度が低い処理
- 更新頻度が高い処理
- 即時更新処理
- 更新頻度が高く、即時更新処理
- 複数サーバー接続

2-9.変化と考慮点 [154]

- システム構成
- 対応方針
- 特長を活かす (データ量無制限の表示)
- 特長を活かす(オフラインを意識させない)
- 特長を活かす (高速表示のテクニック)
- 特長を活かす (高速表示のテクニック)
- 参考通信リトライの考慮点
- 通信先の制約
- 弱点を補強(データ消失対応)
- 弱点を補強(データ機密保護)
- 弱点を補強(自動消去と暗号化)
- 弱点を補強(プログラムの保護)
- 弱点を補強(集中管理)

データ管理の参考事例	
導入効果	
導入ステップ	
2-10.TypeScript を知る [172]
Angular 用の開発言語	
TypeScript とは	
TypeScript の型指定	
既存 JavaScript 資産に型情報付加	
TypeScript コンパイル設定ファイル	
TypeScript 記述例	
3.システム開発	
3-1.大規模システム対応の実装 [181]
大規模対応の検討項目	
モジュール分割	
コンポーネント分割	
フォルダ構造	
コンポーネント指向	
分割した構成要素の同期	
動作の同期	
データの同期の必要性	
直接同期方式	
階層構造で同期	
処理の集中化	
Redux による同期	
集中化にともなうクラス肥大化の回避	
集中管理のループ処理	
3-2. 実装例(顧客管理システム) [196]
業務シナリオ	
業務詳細	
主な機能	
セキュリティ強化	
顧客管理システムの実行	
画面フロー	
プロジェクトの読み込み	
Angular CLI Proxy 接続	
サーバー構成	

システム全体構成	
3-3.顧客管理システム画面 [207]
ログイン	
顧客一覧	
顧客情報	
報告入力	
報告履歴	
モバイル対応	
処理フロー	
サンプルアプリの処理フロー	
3-4.顧客管理システムの DB [216]
SQL データベース構造の確認	
サンプルアプリのデータベース構造	
顧客情報(t_customer)の構造	
サンプルアプリのデータベース構造	
3-5.顧客管理システムの処理フロー [222]
アクティビティ図	
アプリ全体の処理フロー	
ダウンロード画面の処理フロー	
顧客リスト画面の処理フロー	
自動ログアウト処理フロー	
3-6.顧客管理システム機能の実装 [232]
国際化対応	
レスポンシブデザイン実装	
モジュールの遅延ローディング	
エラーログの保存	