Requirements

- an internet connection
- Chrome: https://www.google.com/chrome/
- a GitHub account

Eclipse Theia

Building Cloud & Desktop IDEs

TypeFox

Outline

- **Theia Architecture**
- Application Shell
- React Integration
- JSON-Form Extension

SCOPE OF THEIA

- Support Desktop and Browser Apps
- Build engineering tools
- Allow fine-grained customization

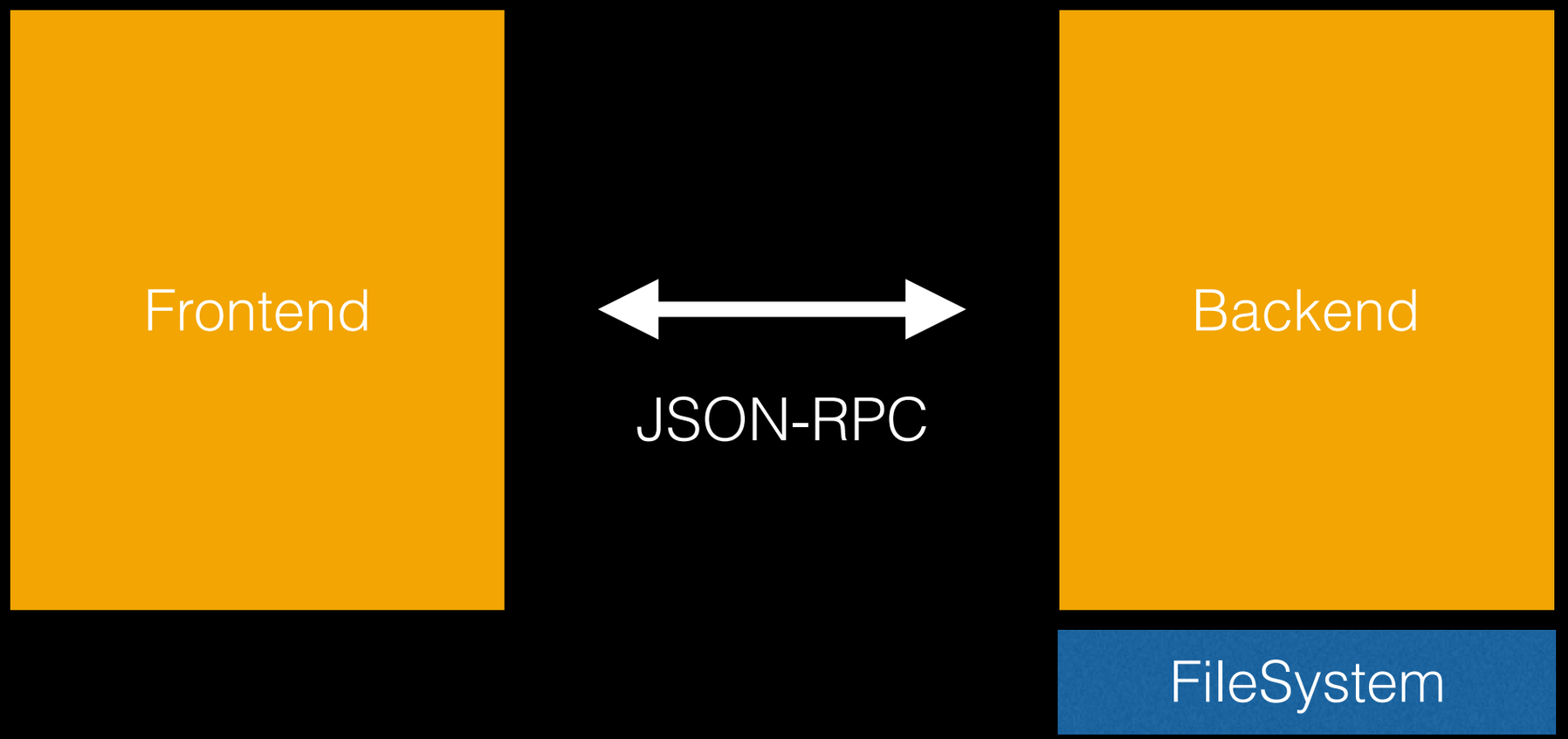
SUPPORT DESKTOP AND BROWSER APPS

ELECTRON

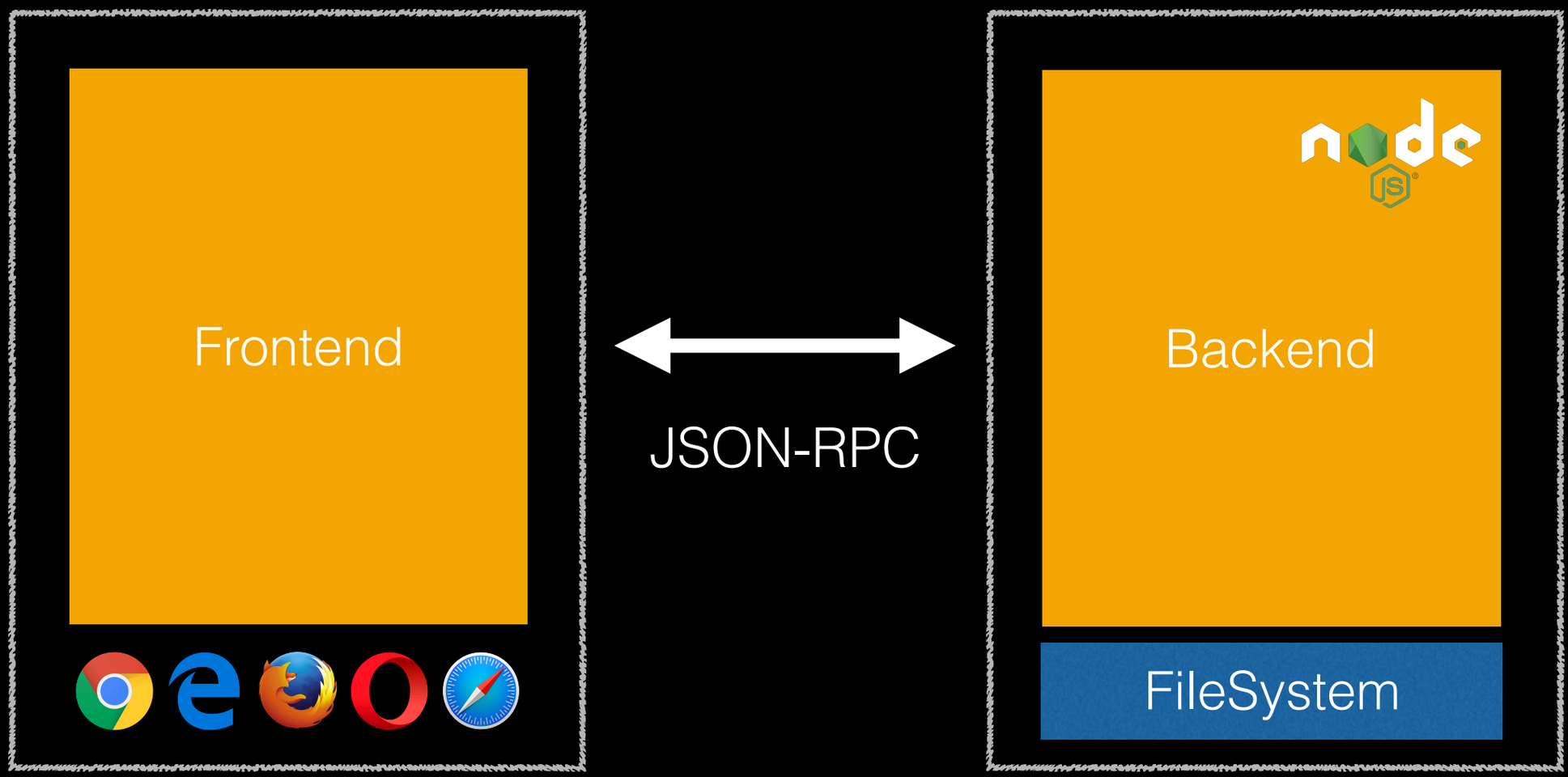
Docs Blog Community Apps Userland Releases Contact

Build cross platform desktop apps with JavaScript, HTML, and CSS

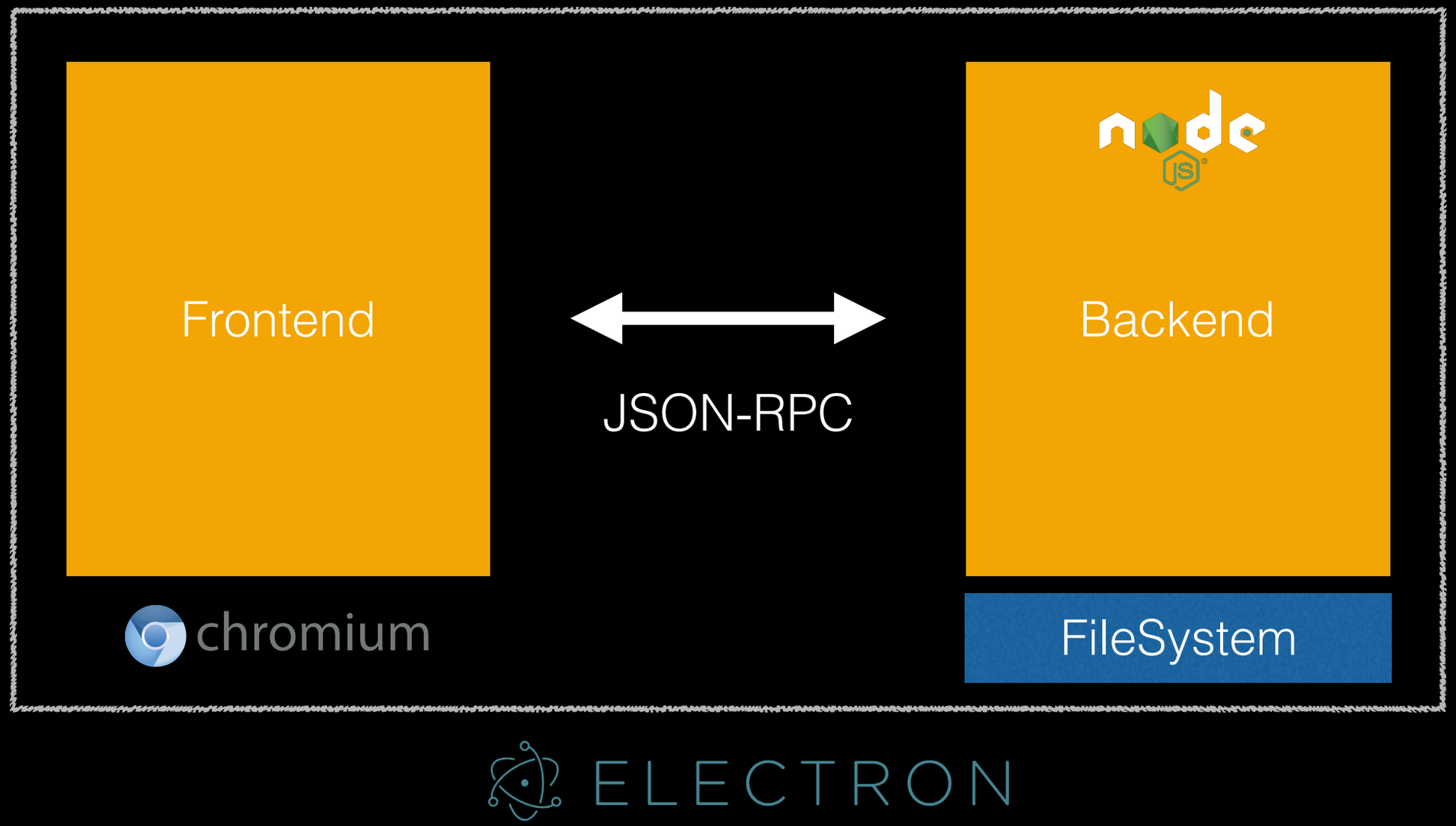
Architecture of THEIA



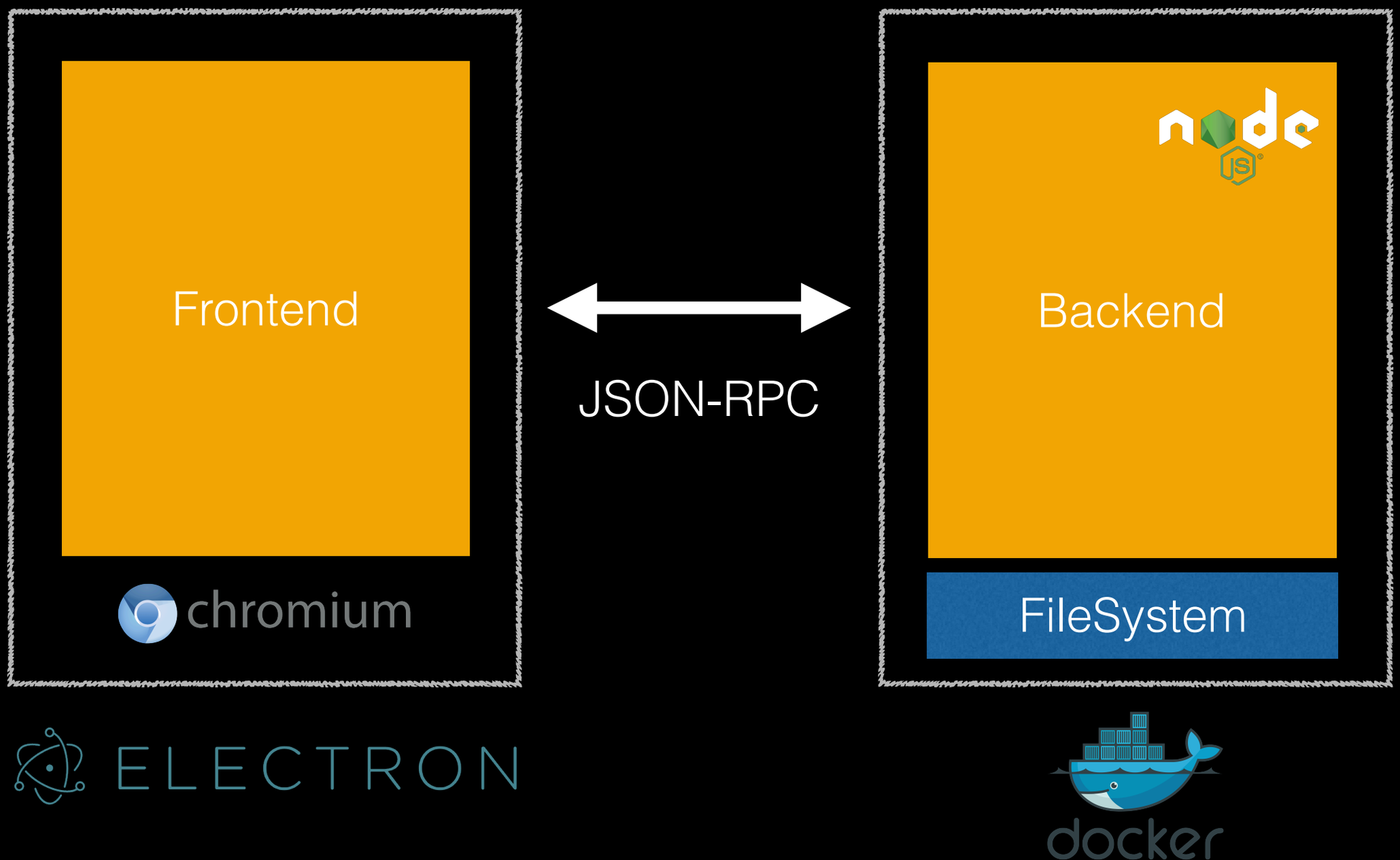
Architecture of THEIA

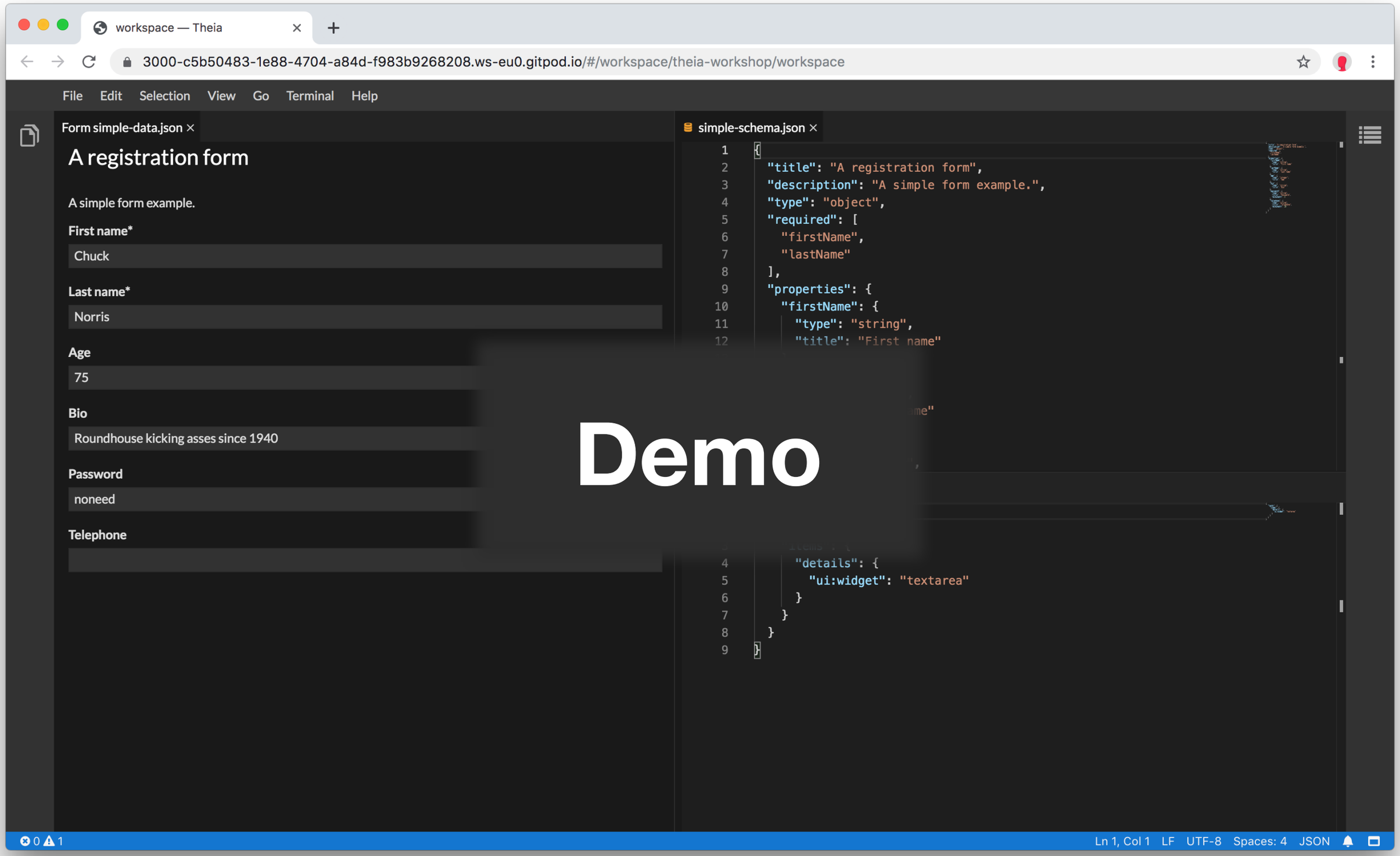


Architecture of THEIA



Architecture of THEIA




workspace — Theia
3000-c5b50483-1e88-4704-a84d-f983b9268208.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
"details": {
"ui:widget": "textarea"
}
}
}
}
Ln 1, Col 1 LF UTF-8 Spaces: 4 JSON

Demo

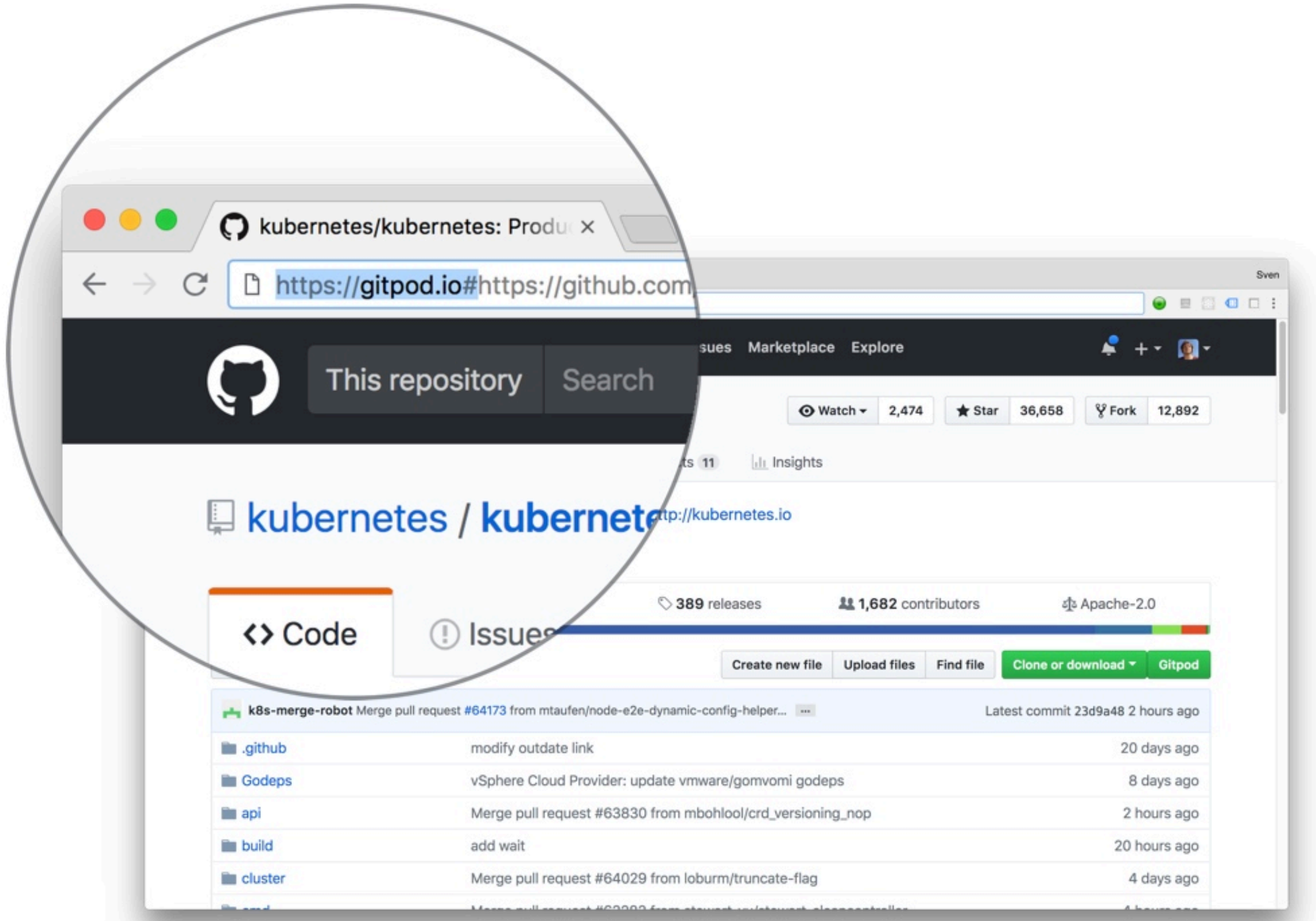
Gitpod

One-Click Online IDE

for GitHub and other code hosting platforms.

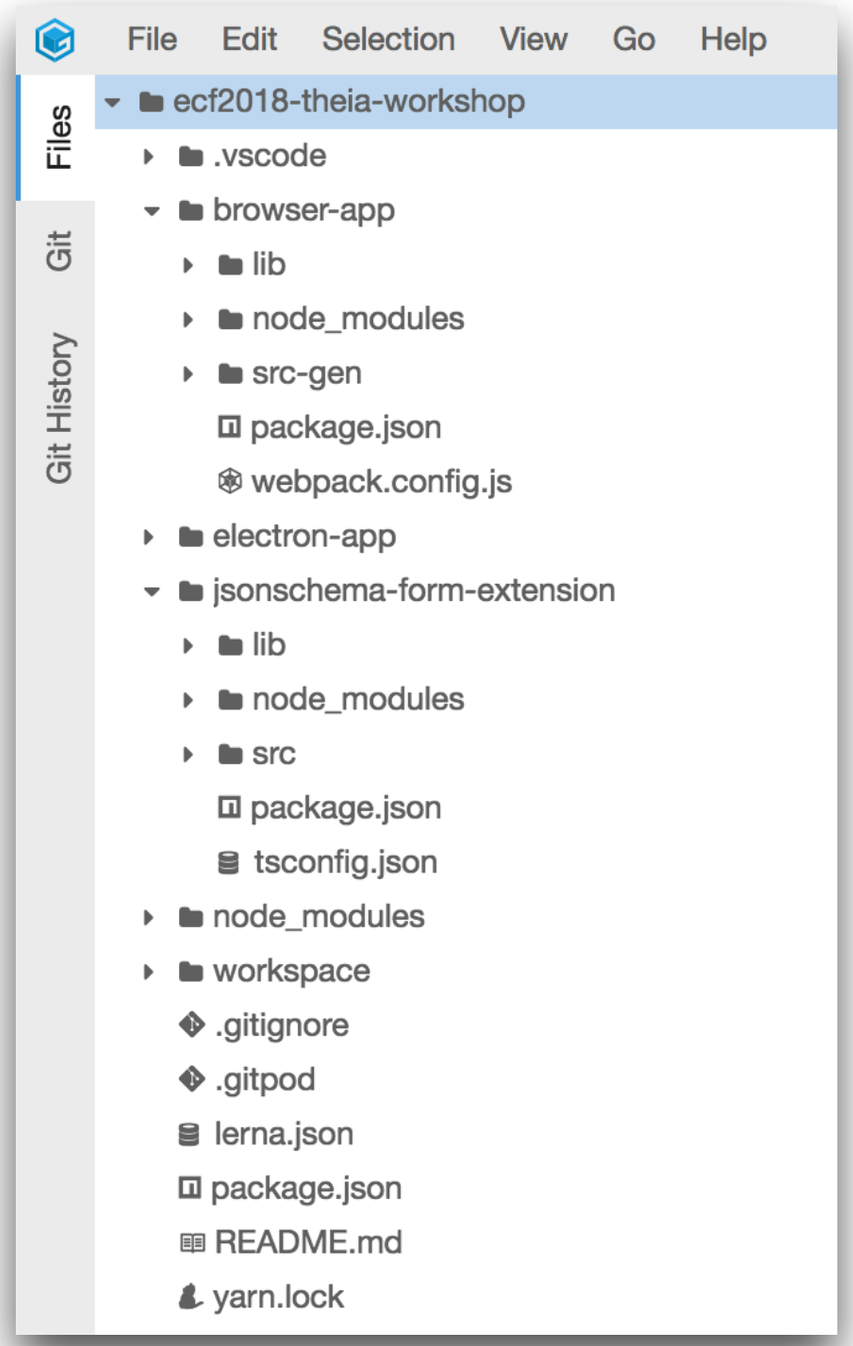
https://gitpod.io/

How does Gitpod work?



Theia **applications** and **extensions** are regular npm packages.

Monorepo



- root package
- extension package
- browser app package
- electron app package

Theia Extension Generator

```
[MacKosyakov-2:my-extension kosyakov$ yo theia-extension
[? The extension's name my
 My
    create package.json
    create lerna.json
    create .gitignore
    create README.md
    create .vscode/launch.json
    create my-extension/package.json
    create my-extension/tsconfig.json
    create my-extension/src/browser/my-frontend-module.ts
    create my-extension/src/browser/my-contribution.ts
    create browser-app/package.json
    create electron-app/package.json
yarn install v1.5.1
info No lockfile found.
[1/4] 🔍  Resolving packages...
```

https://github.com/theia-ide/generator-theia-extension

Root Package

https://lernajs.io/

```
{
  "private": true,
  "scripts": {
    "prepare": "lerna run prepare",
    "rebuild:browser": "theia rebuild:browser",
    "rebuild:electron": "theia rebuild:electron"
  },
  "devDependencies": {
    "lerna": "2.4.0"
  },
  "workspaces": [
    "jsonschema-form-extension",
    "browser-app",
    "electron-app"
  ]
}
```

Application Package

```
{
  "private": true,
  "name": "browser-app",
  "version": "0.0.0",
  "dependencies": {
    "@theia/core": "next",
    "@theia/filesystem": "next",
    "@theia/workspace": "next",
    "@theia/preferences": "next",
    "@theia/navigator": "next",
    "@theia/process": "next",
    "@theia/terminal": "next",
    "@theia/editor": "next",
    "@theia/languages": "next",
    "@theia/markers": "next",
    "@theia/monaco": "next",
    "@theia/typescript": "next",
    "@theia/messages": "next",
    "jsonschema-form-extension": "0.0.0"
  },
  "devDependencies": {
    "@theia/cli": "next"
  },
  "scripts": {
    "prepare": "theia build --mode development",
    "start": "theia start",
    "watch": "theia build --watch --mode development"
  },
  "theia": {
    "target": "browser"
  }
}
```

Extension Package

```
{
  "name": "jsonschema-form-extension",
  "keywords": [
    "theia-extension"
  ],
  "version": "0.0.0",
  "files": [
    "lib",
    "src"
  ],
  "dependencies": {
    "@theia/core": "next"
  },
  "devDependencies": {
    "rimraf": "latest",
    "typescript": "latest"
  },
  "scripts": {
    "prepare": "yarn run clean && yarn run build",
    "clean": "rimraf lib",
    "build": "tsc",
    "watch": "tsc -w"
  },
  "theiaExtensions": [
    {
      "frontend": "lib/browser/jsonschema-form-frontend-module"
    }
  ]
}
```

Extension Modules

- **frontend** - loaded in browser and electron frontends
- **frontendElectron** - loaded in electron frontend
  - overrides **frontend** module if present
- **backend** - loaded in browser and electron backends
- **backendElectron** - loaded in electron backend
  - overrides **backend** module if present

Extension Module

```
export default new ContainerModule(bind => {
    bind(CommandContribution)
        .to(JsonschemaFormCommandContribution)
        .inSingletonScope();

    bind(MenuContribution)
        .to(JsonschemaFormMenuContribution)
        .inSingletonScope();
});
```

An extension module should have a default export of **ContainerModule | Promise<ContainerModule>** type.

InversifyJS

A powerful and lightweight inversion of control container
for JavaScript & Node.js apps powered by TypeScript.

http://inversify.io/

Dependency Injection

```
export default new ContainerModule(bind => {
    bind(CommandContribution)
        .to(JsonschemaFormCommandContribution)
        .inSingletonScope();
});
```

```
@injectable()
export class JsonschemaFormCommandContribution implements CommandContribution {

    @inject(MessageService) private readonly messageService: MessageService;

    registerCommands(registry: CommandRegistry): void {
        registry.registerCommand(JsonschemaFormCommand, {
            execute: () => this.messageService.info('Hello World!')
        });
    }
}
```

Exercise 0

Build Theia Application

https://github.com/TypeFox/theia-workshop/tree/exercise-0

Outline

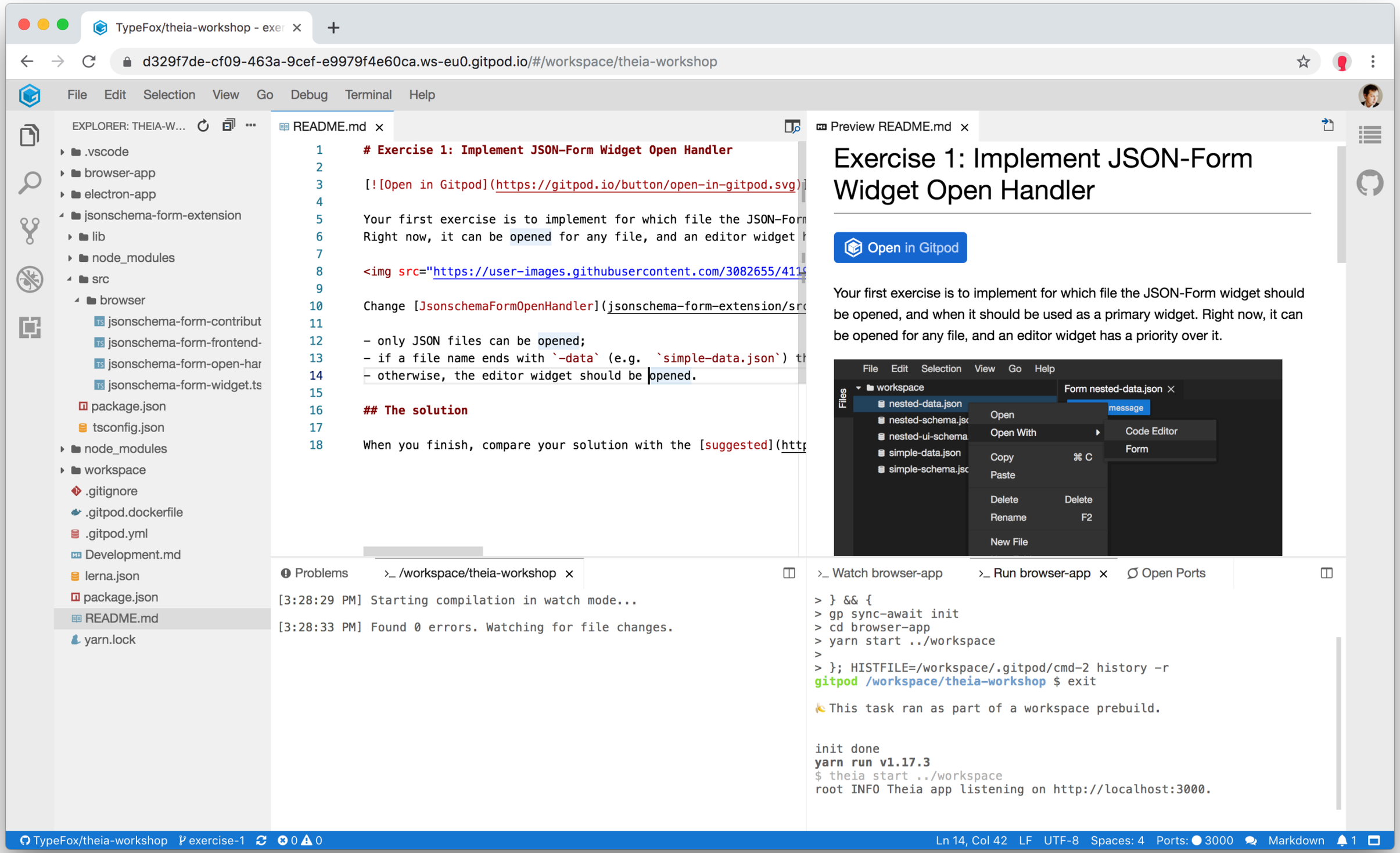
- Theia Architecture
- **Application Shell**
- React Integration
- JSON-Form Extension

PhosphorJS

a collection of libraries
designed to make it easy
to build high-performance,
pluggable, desktop-style
web applications

https://phosphorjs.github.io/

Application Shell



What is a widget?

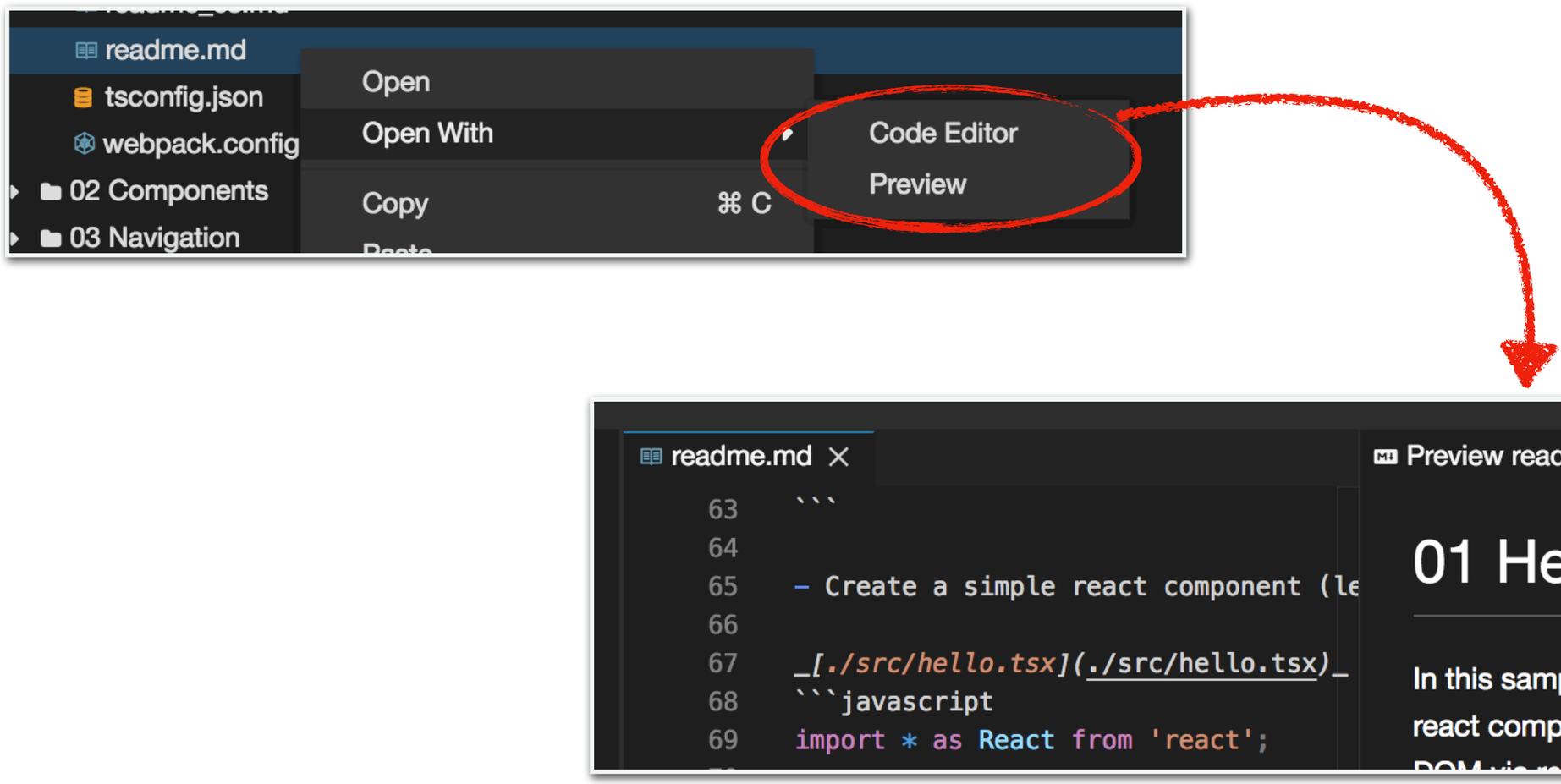
What is a widget?

An HTML element with lifecycle hooks

Life of a Widget

- Initialization - constructor or @postConstruct inversifyJS callback
- Attaching - on[After|Before][Attach|Detach]
- Displaying - on[After|Before][Show|Hide]
- Rendering - onUpdateRequest
- Activation - onActivateRequest
- Resizing - onResizeRequest
- Layouting - onChild[Added|Removed]
- Closing - onCloseRequest
- Destruction - dispose method

Opening URIs



Widget Open Handler

```
@injectable()
export class JsonschemaFormOpenHandler
    extends WidgetOpenHandler<JsonschemaFormWidget> {

    readonly id = JsonschemaFormWidget.id;
    readonly label = "Form";

    @inject(EditorManager)
    protected readonly editorManager: EditorManager;

    /**
     * Test whether this handler can open the given URI for given options.
     * Return a positive number if this handler can open; otherwise it cannot.
     *
     * A returned value indicating a priority of this handler.
     */
    canHandle(uri: URI): number {
        return this.editorManager.canHandle(uri) / 2;
    }

}
```

URI

```
/*
 *  foo://example.com:8042/over/there?name=ferret#nose
 *  \_/   \______________/\_________/ \_________/ \__/
 *   |           |            |            |        |
 * scheme     authority     path         query  fragment
 *   |   _____________________|__
 *  / \ /                        \
 *  urn:example:animal:ferret:nose
 */
export class URI {
    constructor(uri?: string | Uri);
    readonly parent: URI;
    readonly scheme: string;
    readonly authority: string;
    readonly path: Path;
    readonly query: string;
    readonly fragment: string;
    resolve(path: string | Path): URI;
    toString(skipEncoding?: boolean): string;
    withPath(path: string | Path): URI;
    withoutPath(): URI;
}
```

Path

```
/**
 *  ┌─────────────────────┬────────────┐
 *  │         dir         │    base    │
 *  ├──────┬              ├──────┬─────┤
 *  │ root │              │ name │ ext │
 * "  /    │ home/user/dir / file  .txt "
 *  └──────┴──────────────┴──────┴─────┘
 */
export class Path {
    readonly isAbsolute: boolean;
    readonly isRoot: boolean;
    readonly root: Path | undefined;
    readonly base: string;
    readonly name: string;
    readonly ext: string;
    constructor(raw: string);
    readonly dir: Path;
    join(...paths: string[]): Path;
    toString(): string;
}
```

Exercise 1

Implement Widget Open Handler

https://github.com/TypeFox/theia-workshop/tree/exercise-1

Outline

- Theia Architecture
- Application Shell
- **React Integration**
- JSON-Form Extension

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

https://reactjs.org/

Rendering with JSX

```
const element = <div>
    <h1>Hello World!</h1>
</div>;

ReactDOM.render(
    element,
    document.getElementById('root')
);
```

Rendering with JSX

```
const name = 'World';
const element = <div>
    <h1>Hello {name}!</h1>
</div>;

ReactDOM.render(
    element,
    document.getElementById('root')
);
```

Component & Props

```
export interface Greeting {
    name: string
}

export class GreetingView extends React.Component<Greeting> {
    render(): JSX.Element {
        return <h1>Hello {this.props.name}!</h1>;
    }
}

ReactDOM.render(
    <Greeting name='World'/>,
    document.getElementById('root')
);
```

State

```
export class JsonschemaFormView extends React.Component<{}, Greeting> {

    constructor(props: {}) {
        super(props);
        this.state = {
            name: 'World'
        }
    }

    render(): JSX.Element {
        return <React.Fragment>
            <GreetingView name={this.state.name} />
            Greet <input value={this.state.name} />
        </React.Fragment>;
    }

}
```

Event Handling

```
export class JsonschemaFormView extends React.Component<{}, Greeting> {

    constructor(props: {}) {
        super(props);
        this.state = {
            name: 'World'
        }
    }

    render(): JSX.Element {
        return <React.Fragment>
            <GreetingView name={this.state.name} />
            Greet <input value={this.state.name} onChange={this.updateName} />
        </React.Fragment>;
    }

    protected updateName = (e: React.ChangeEvent<HTMLInputElement>) =>
        this.setState({
            name: e.currentTarget.value
        });

}
```

Component Lifecycle

```
export class Clock extends React.Component {
    constructor(props) {
        super(props);
        this.state = { date: new Date() };
    }

    componentDidMount() {
        this.timerID = setInterval(() => this.tick(), 1000);
    }
    componentWillUnmount() {
        clearInterval(this.timerID);
    }

    tick = () => this.setState({
        date: new Date()
    });
    render() {
        return <div>
            <h1>Hello, world!</h1>
            <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
        </div>;
    }
}
```

Widget Integration

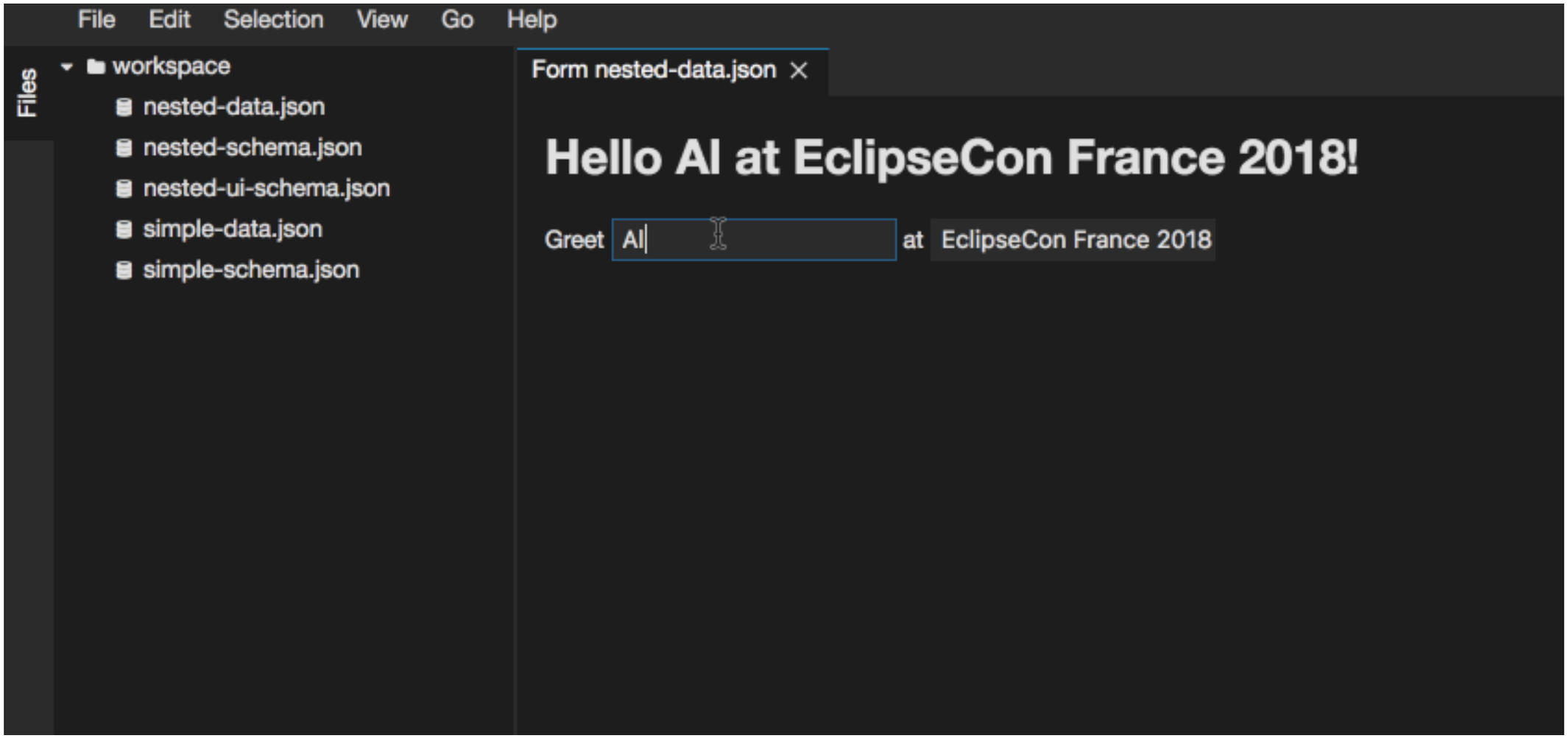
```
@injectable()
export class JsonschemaFormWidget extends BaseWidget {

    @postConstruct()
    protected async init(): Promise<void> {
        this.toDispose.push(Disposable.create(() =>
            ReactDOM.unmountComponentAtNode(this.node)
        ));
        ReactDOM.render(<JsonschemaFormView />, this.node);
    }

}
```

Exercise 2

Improve React Component



https://github.com/TypeFox/theia-workshop/tree/exercise-2

Outline

- Theia Architecture
- Application Shell
- React Integration
- **JSON-Form Extension**

Disposable

```ts
export interface Disposable {
   dispose(): void;
}
```

```ts
const disposable = commands.registerCommand(command);
disposable.dispose();
```

Disposable

```ts
export interface Disposable {
   dispose(): void;
}
```

```ts
const disposable = commands.registerCommand(command);
disposable.dispose();
```

Unregister a command

Disposable Collection

```
toDispose: DisposableCollection;

componentWillMount(): void {
    this.toDispose.push(this.schemaStorage);
}

componentWillUnmount(): void {
    this.toDispose.dispose();
}
```

Disposable Collection

```
toDispose: DisposableCollection;

componentWillMount(): void {
    this.toDispose.push(this.schemaStorage);
}

componentWillUnmount(): void {
    this.toDispose.dispose();
}
```

Collect all widget services and listeners!

Disposable Collection

```
toDispose: DisposableCollection;

componentWillMount(): void {
    this.toDispose.push(this.schemaStorage);
}

componentWillUnmount(): void {
    this.toDispose.dispose();
}
```

Exceptions are handled

Event Handling

```
protected readonly onDidChangeEmitter = new Emitter<T>();
readonly onDidChange: Event<T> = this.onDidChangeEmitter.event;
```

```
protected reconcile(): void {
    const schema = this.parse();
    this.onDidChangeEmitter.fire(schema);
}
```

```
this.schemaStorage.onDidChange(schema =>
    this.setState({ schema })
)
```

Event Handling

```
protected readonly onDidChangeEmitter = new Emitter<T>();
readonly onDidChange: Event<T> = this.onDidChangeEmitter.event;
```

Emitting

```
protected reconcile(): void {
    const schema = this.parse();
    this.onDidChangeEmitter.fire(schema);
}
```

```
this.schemaStorage.onDidChange(schema =>
    this.setState({ schema })
)
```

Event Handling

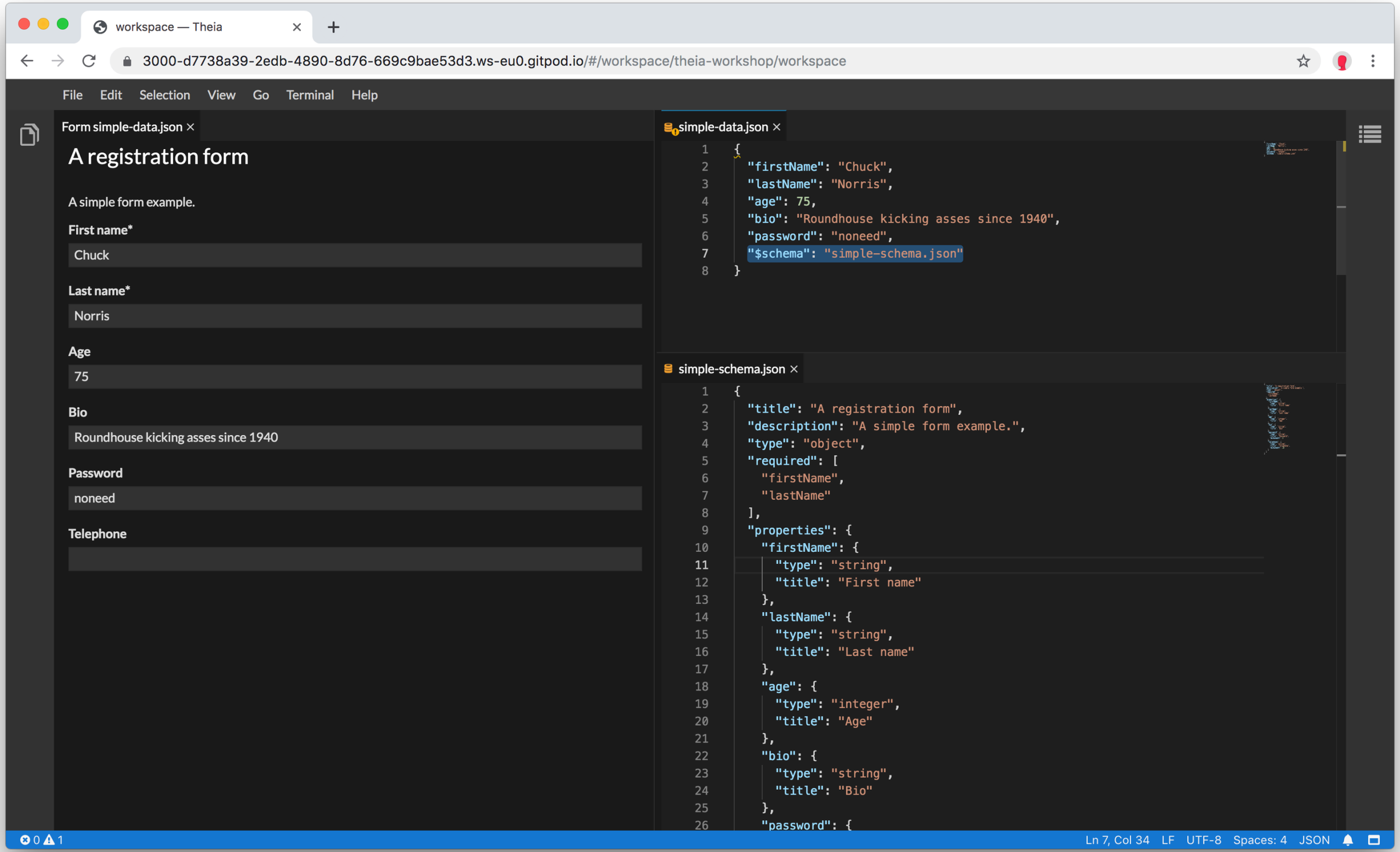
```
protected readonly onDidChangeEmitter = new Emitter<T>();
readonly onDidChange: Event<T> = this.onDidChangeEmitter.event;
```

```
protected reconcile(): void {
    const schema = this.parse();
    this.onDidChangeEmitter.fire(schema);
}
```

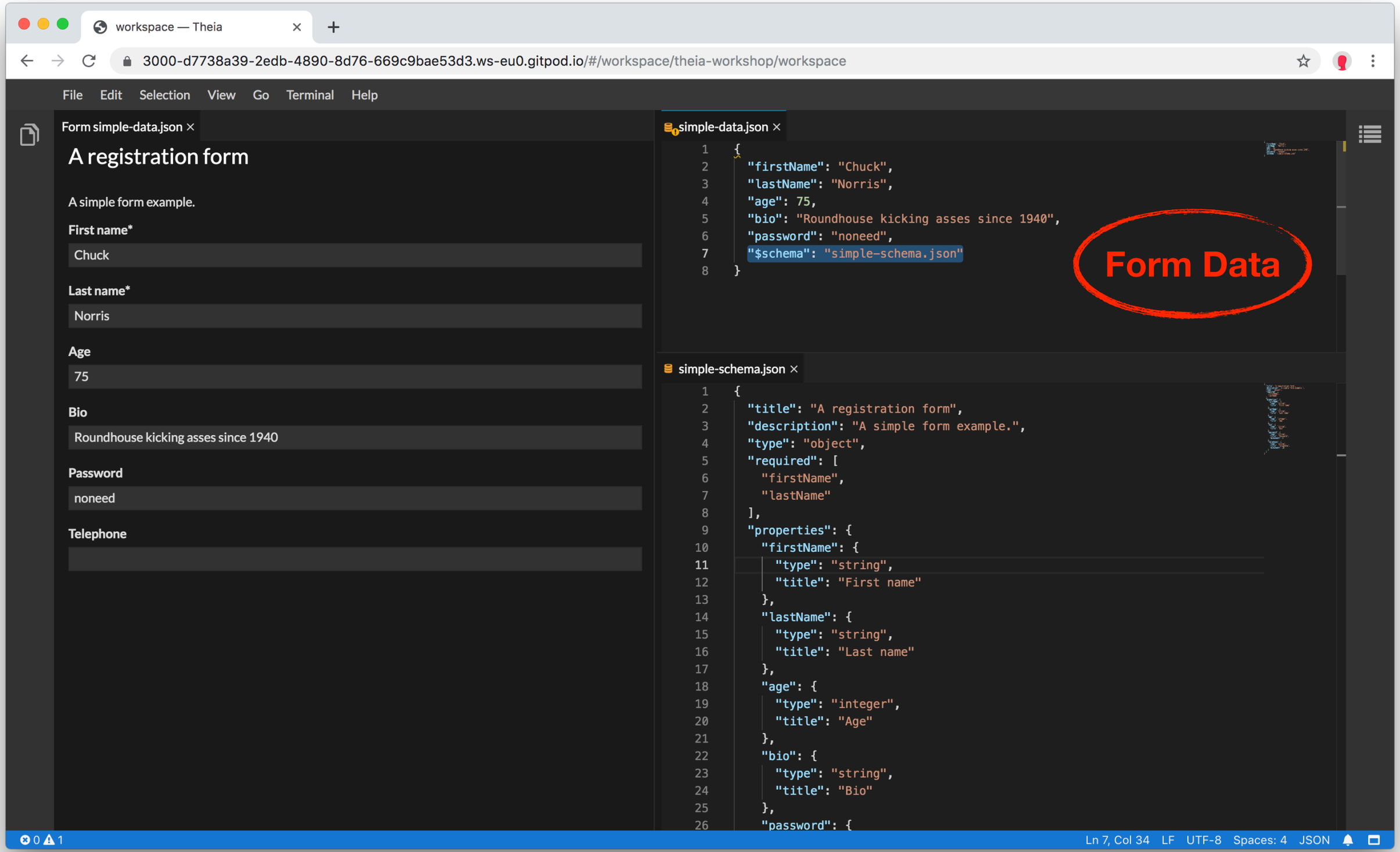
```
this.schemaStorage.onDidChange(schema =>
    this.setState({ schema })
)
```

Listening

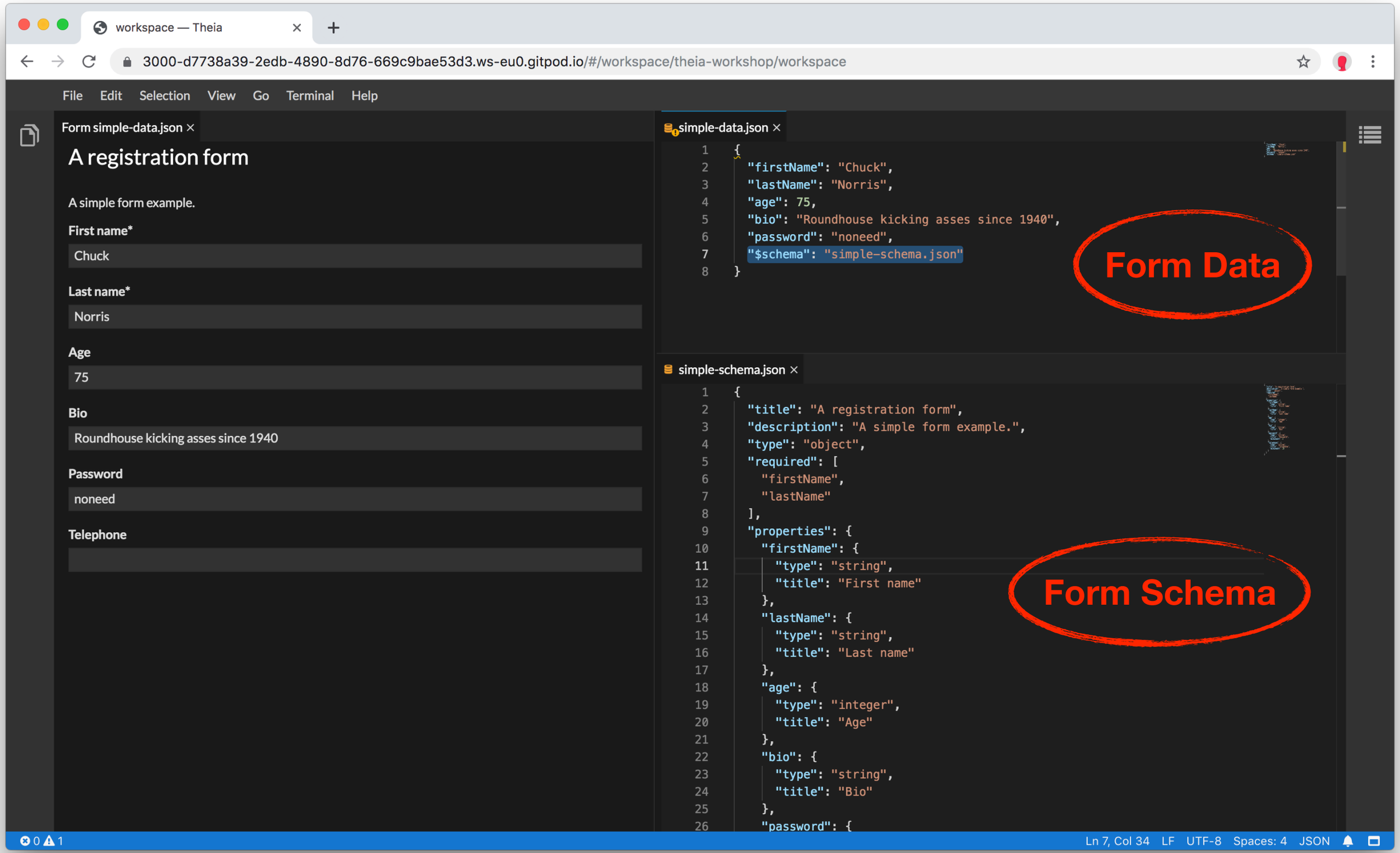
Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

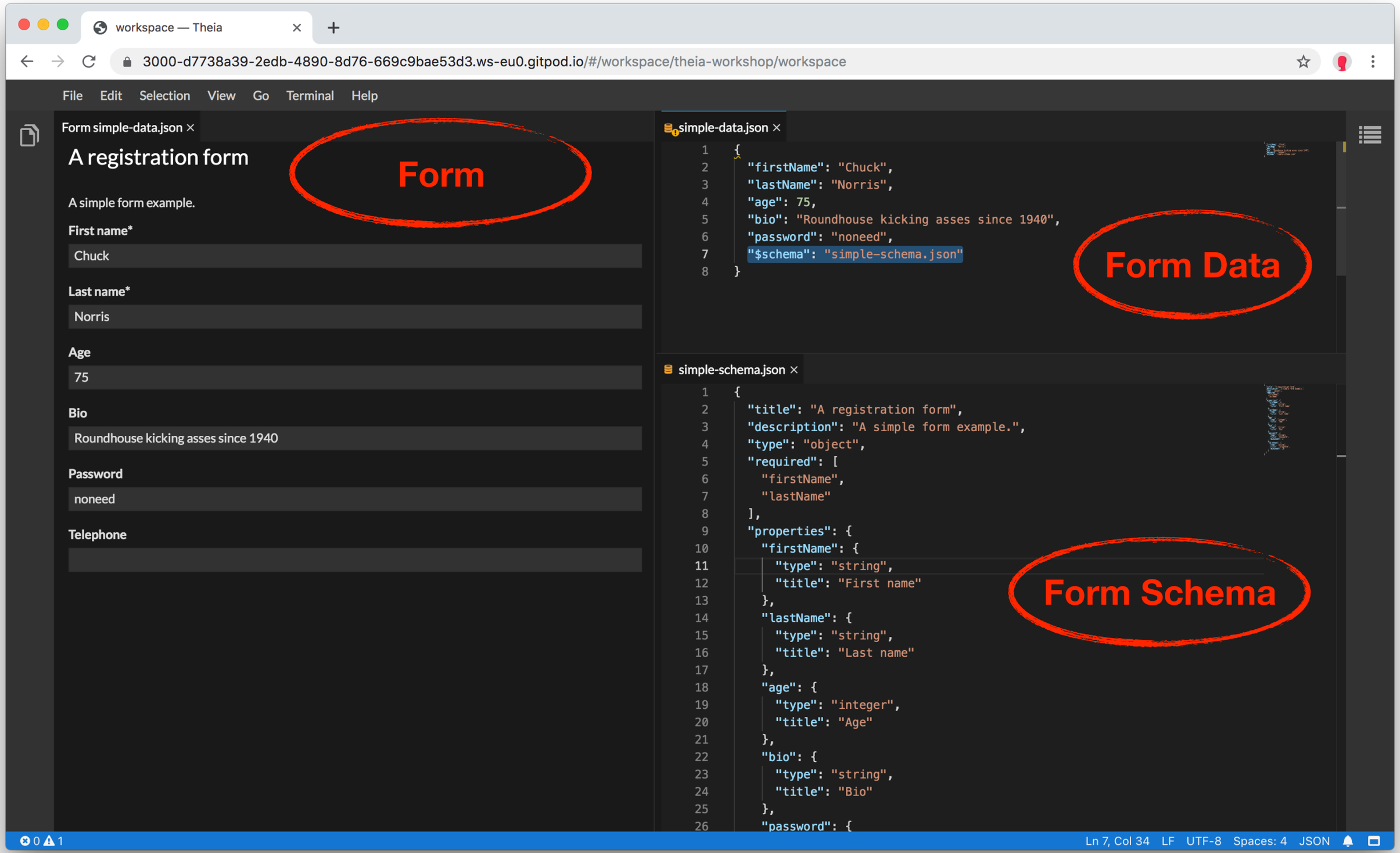
Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
Form Schema
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

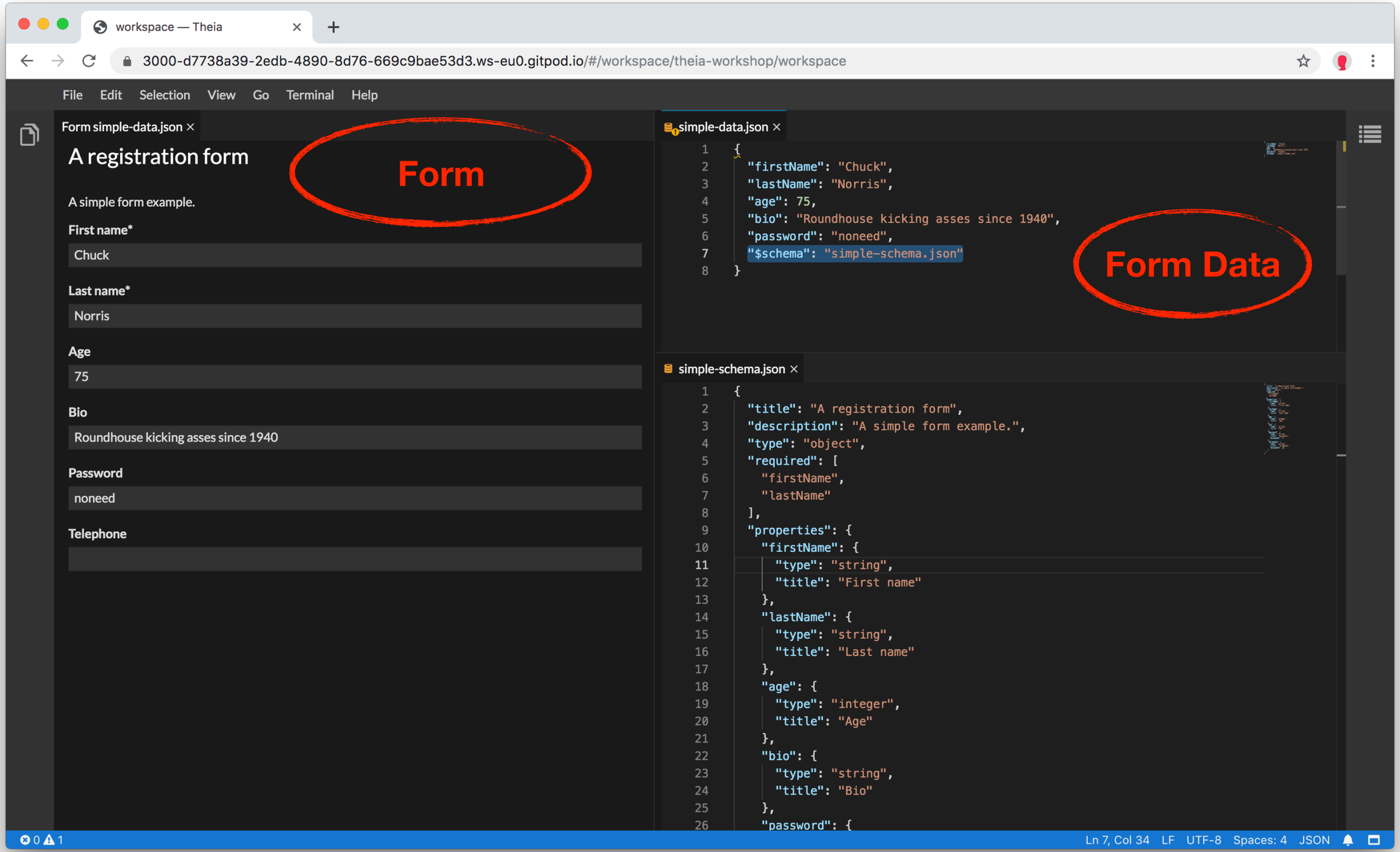
Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
Form
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
Form Schema
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

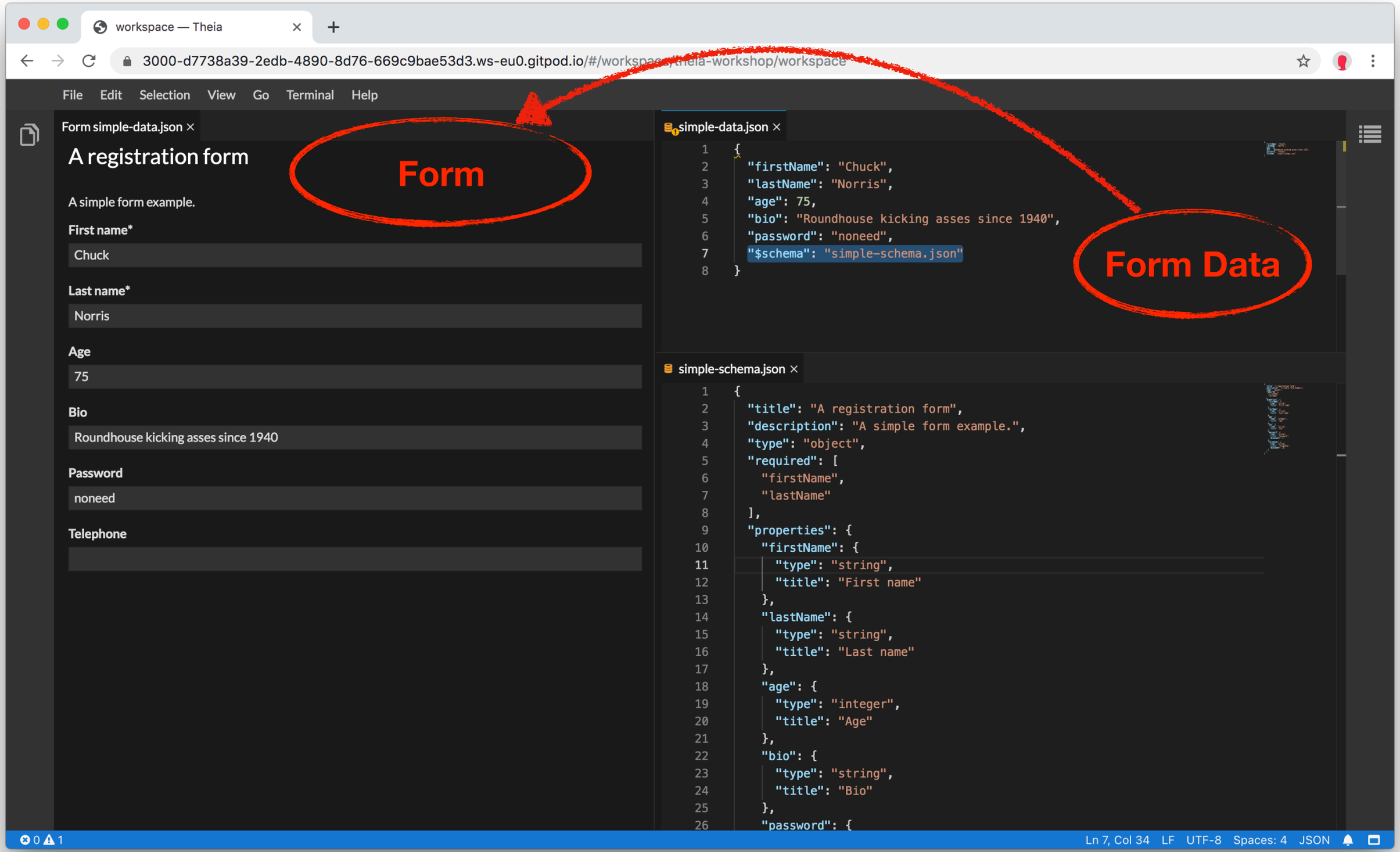
Data Flow

- Files -> UI State
  - **Form Data File -> Form Data**
  - Form Schema File -> Form Schema
- UI State -> UI
- UI -> Form Data File

Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
Form
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
Form
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

Updating Form Data

```
componentWillMount(): void {
    this.reconcileFormData();
    this.toDispose.push(
        this.props.model.onDidChangeContent(() =>
            this.reconcileFormData()
        )
    );
}
```

Updating Form Data

```
componentWillMount(): void {
    this.reconcileFormData();
    this.toDispose.push(
        this.props.model.onDidChangeContent(() =>
            this.reconcileFormData()
        )
    );
}
```

Update the form data from initial file content.

Updating Form Data

```
componentWillMount(): void {
    this.reconcileFormData();
    this.toDispose.push(
        this.props.model.onDidChangeContent(() =>
            this.reconcileFormData()
        )
    );
}
```

Whenever the content of form data file is changed…

Updating Form Data

```
componentWillMount(): void {
    this.reconcileFormData();
    this.toDispose.push(
        this.props.model.onDidChangeContent(() =>
            this.reconcileFormData()
        )
    );
}
```

…update the form data.

Updating Form Data

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText())) || {};
    this.setState({ formData });
}
```

Updating Form Data

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText())) || {};
    this.setState({ formData });
}
```

Update the state with new form data.

Data Flow

- Files -> UI State
  - Form Data File -> Form Data
  - Form Schema File -> Form Schema
- **UI State -> UI**
- UI -> Form Data File

Rendering Form Data

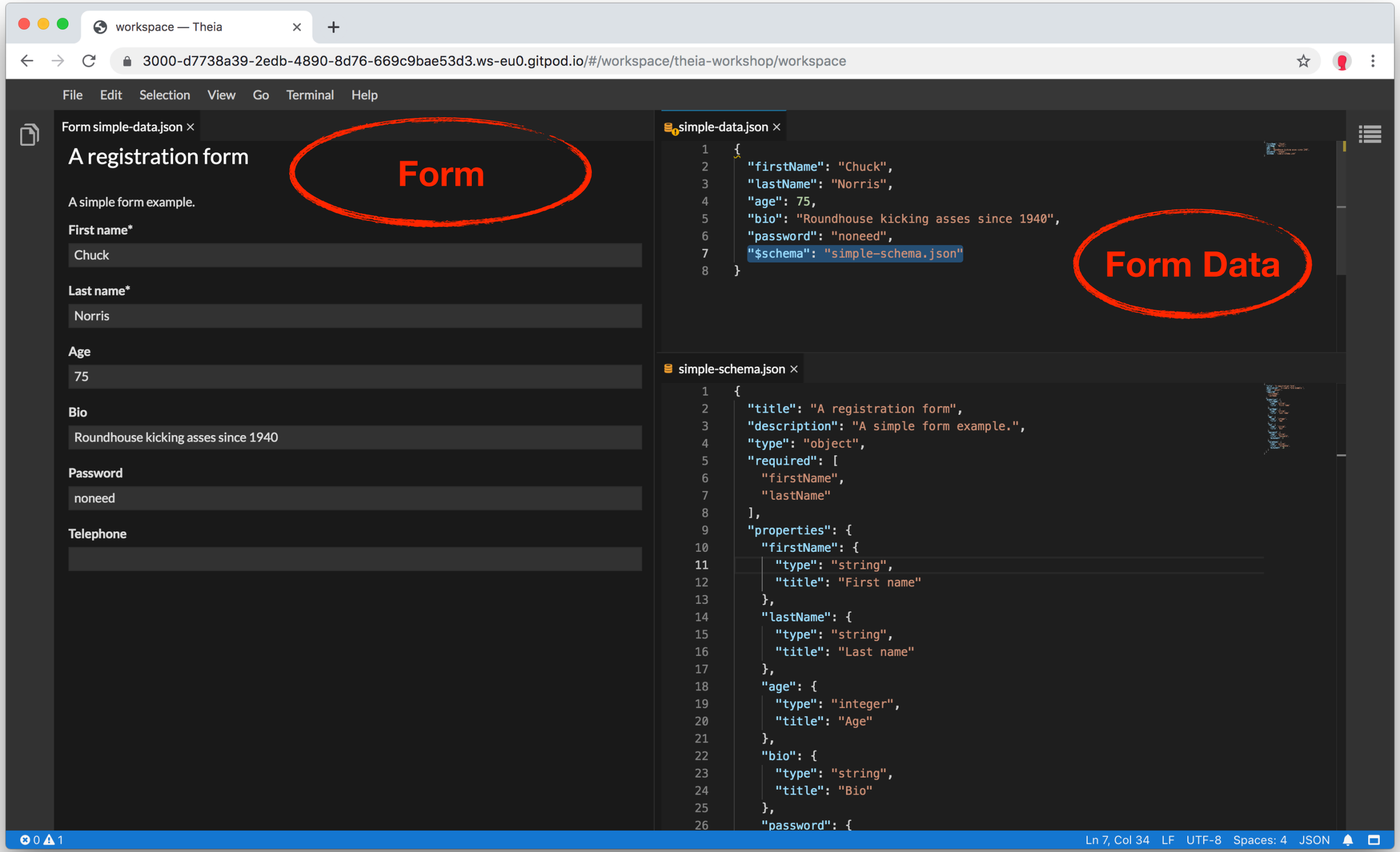
```
render(): JSX.Element | null {
    const { schema, formData } = this.state;
    return <Form
        schema={schema}
        formData={formData}>
        <div />
    </Form>;
}
```

Render new form data from the state.

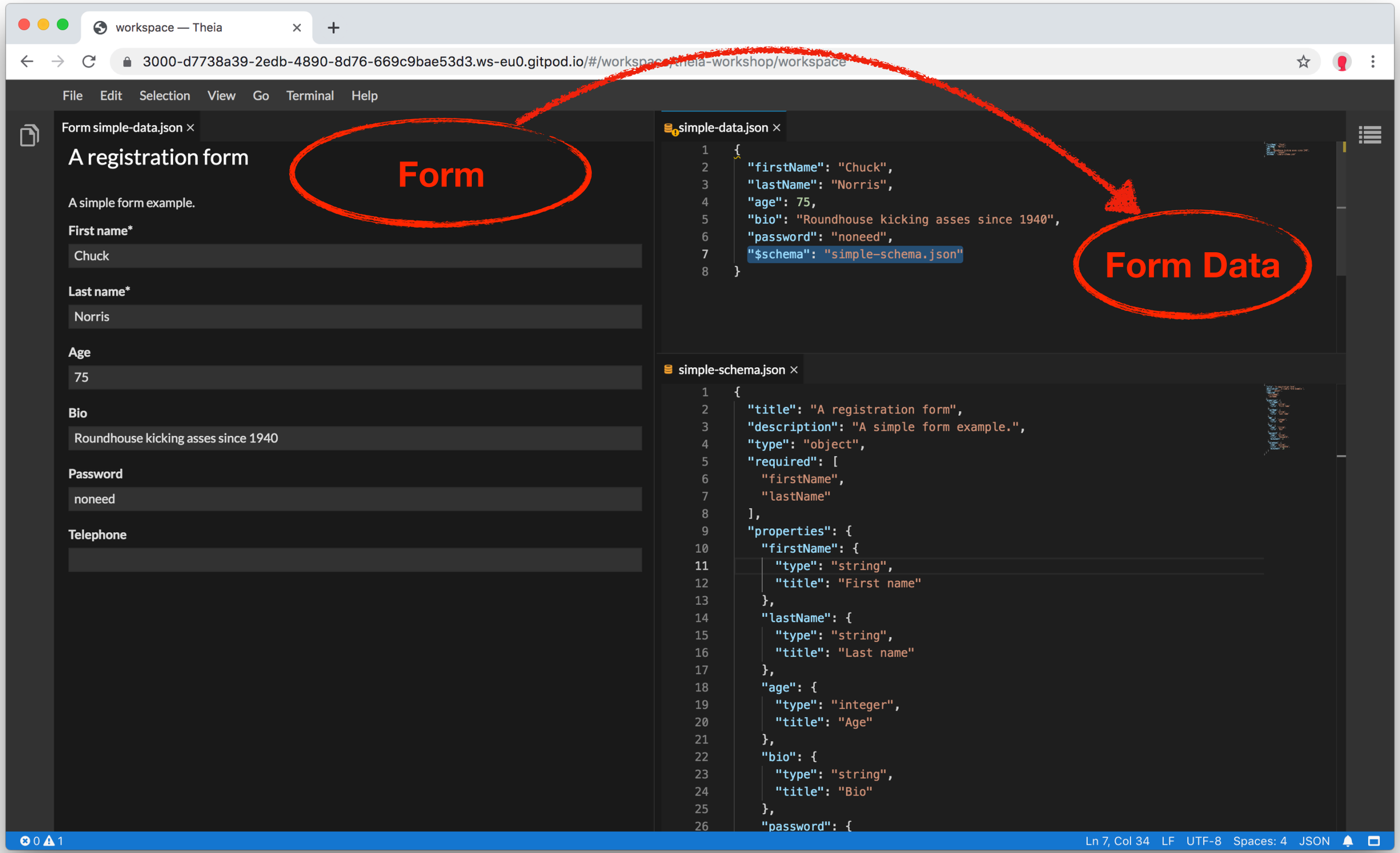
Data Flow

- Files -> UI State
  - Form Data File -> Form Data
  - Form Schema File -> Form Schema
- UI State -> UI
- **UI -> Form Data File**

Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
Form
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
Form
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

Updating Form Data File

```
render(): JSX.Element | null {
    const { schema, formData } = this.state;
    return <Form
        schema={schema}
        formData={formData}
        onChange={this.submit}>
        <div />
    </Form>;
}
```

Listen to the form data changes…

Updating Form Data File

```
render(): JSX.Element | null {
    const { schema, formData } = this.state;
    return <Form
        schema={schema}
        formData={formData}
        onChange={this.submit}>
        <div />
    </Form>;
}
```

…and submit them.

Updating Form Data File

```
protected submit = (e: IChangeEvent<any>) => {
    const model = this.props.model.textEditorModel;
    const content = model.getValue();
    const edits = jsoncparser.modify(content, [], e.formData);
    model.applyEdits(edits.map(e => {
        const start = model.getPositionAt(e.offset);
        const end = model.getPositionAt(e.offset + e.length);
        return {
            range: monaco.Range.fromPositions(start, end),
            text: e.content
        }
    }));
}
```

Compute minimal edits for original content...

Updating Form Data File

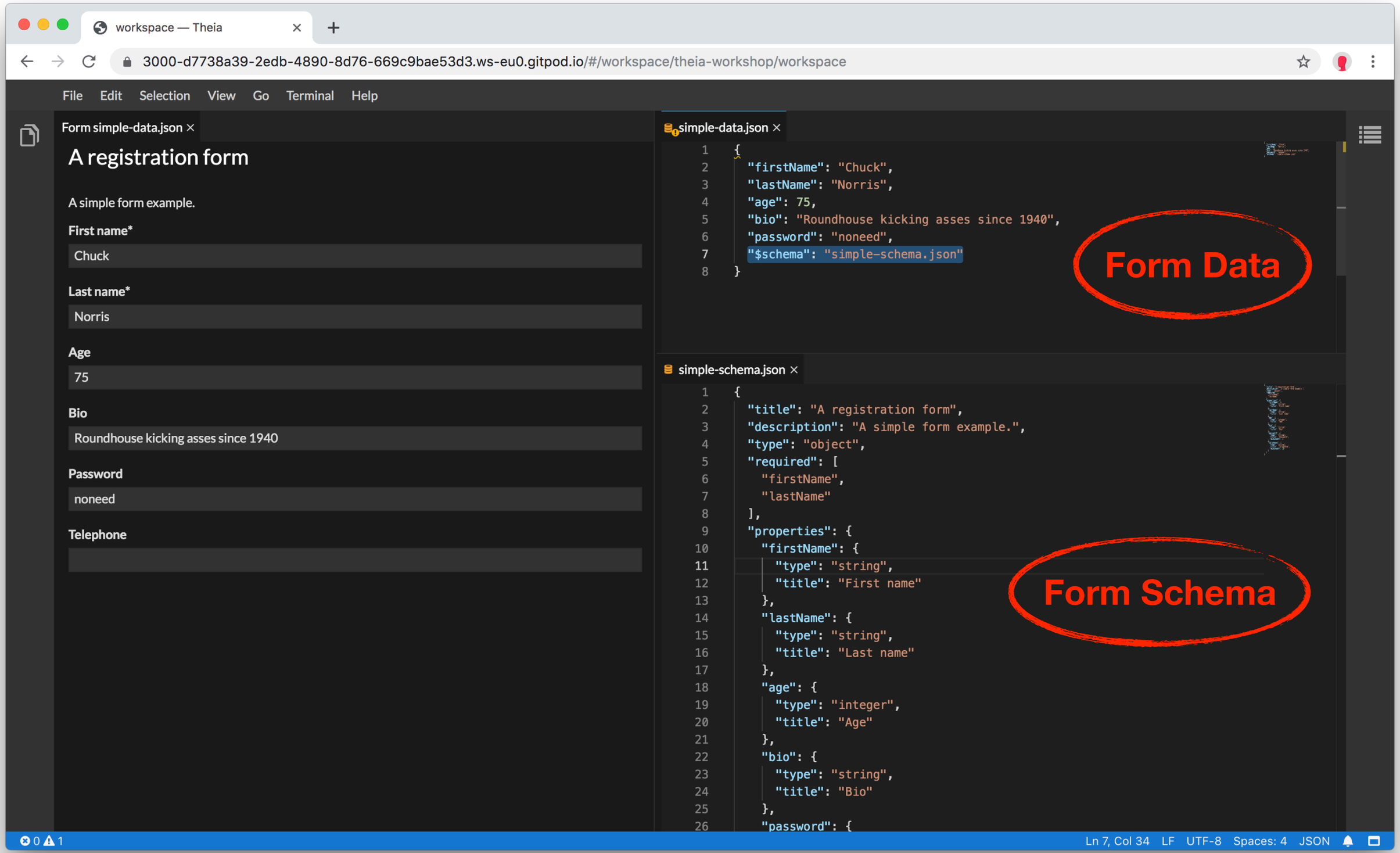
```
protected submit = (e: IChangeEvent<any>) => {
    const model = this.props.model.textEditorModel;
    const content = model.getValue();
    const edits = jsoncparser.modify(content, [], e.formData);
    model.applyEdits(edits.map(e => {
        const start = model.getPositionAt(e.offset);
        const end = model.getPositionAt(e.offset + e.length);
        return {
            range: monaco.Range.fromPositions(start, end),
            text: e.content
        }
    }));
}
```

…apply edits to the file.

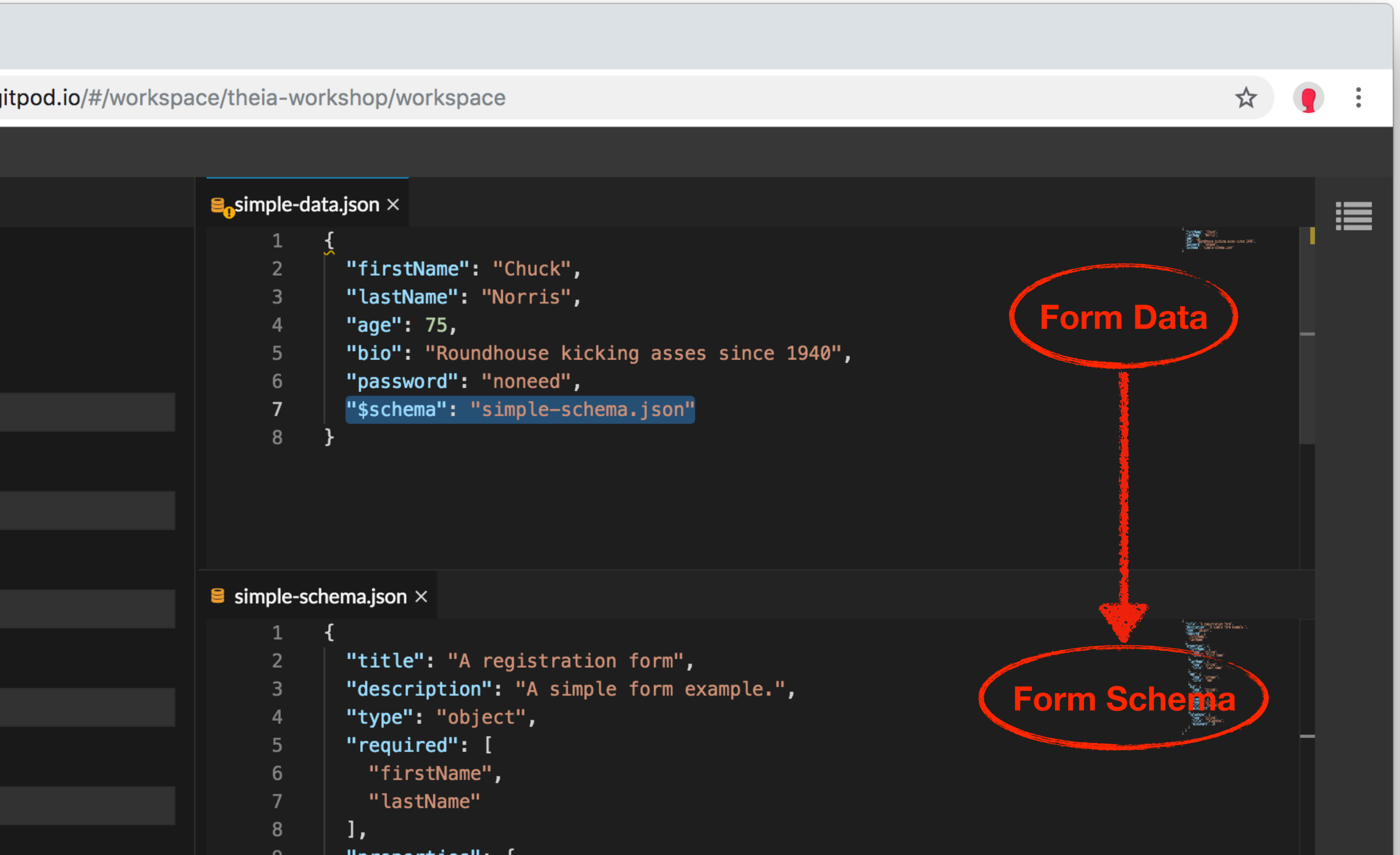
Data Flow

- Files -> UI State
  - Form Data File -> Form Data
  - **Form Schema File -> Form Schema**
- UI State -> UI
- UI -> Form Data File

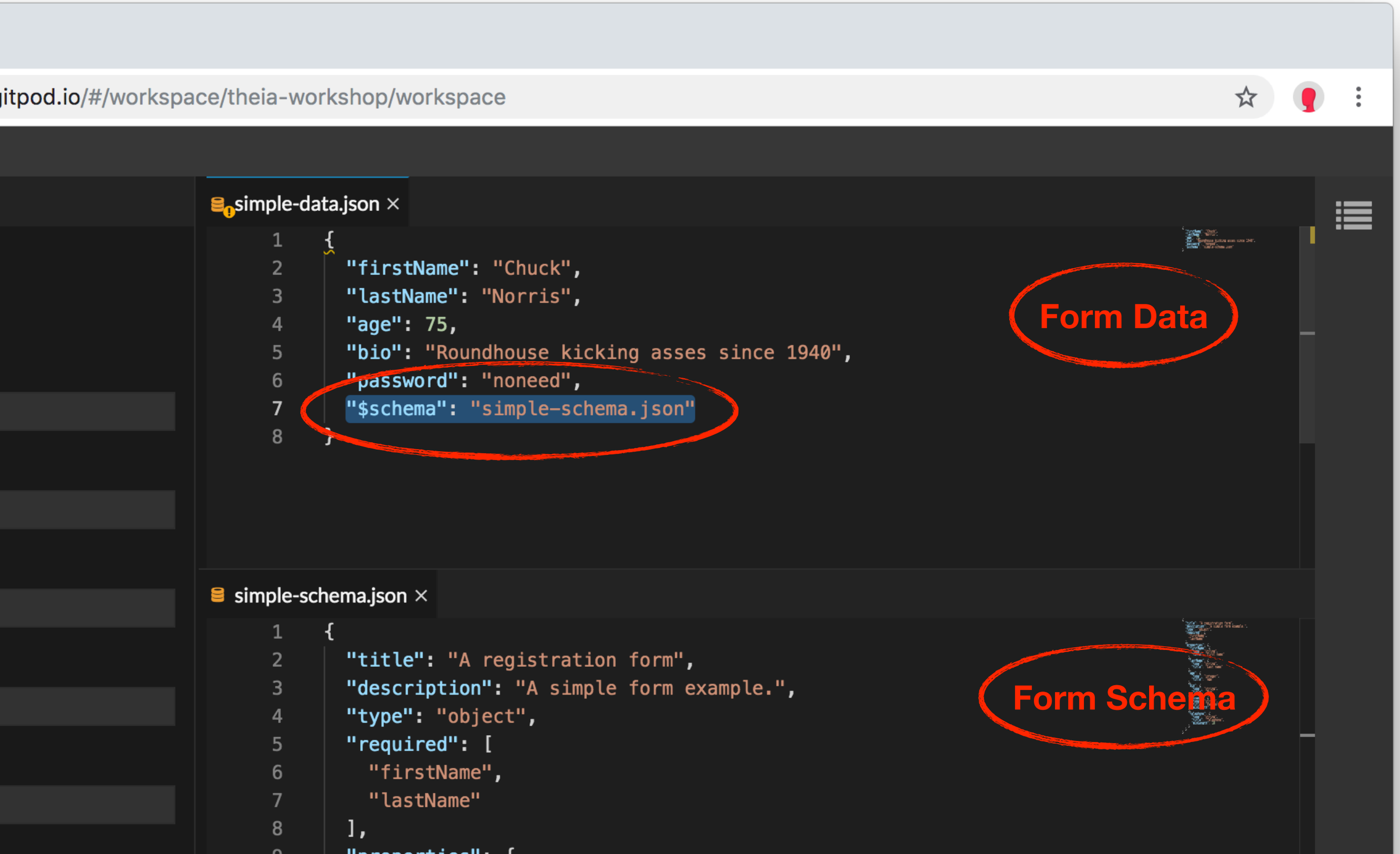
Data Flow


workspace — Theia
3000-d7738a39-2edb-4890-8d76-669c9bae53d3.ws-eu0.gitpod.io/#/workspace/theia-workshop/workspace
File Edit Selection View Go Terminal Help
Form simple-data.json
A registration form
A simple form example.
First name*
Chuck
Last name*
Norris
Age
75
Bio
Roundhouse kicking asses since 1940
Password
noneed
Telephone
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
"properties": {
"firstName": {
"type": "string",
"title": "First name"
},
"lastName": {
"type": "string",
"title": "Last name"
},
"age": {
"type": "integer",
"title": "Age"
},
"bio": {
"type": "string",
"title": "Bio"
},
"password": {
Form Schema
0 1
Ln 7, Col 34 LF UTF-8 Spaces: 4 JSON

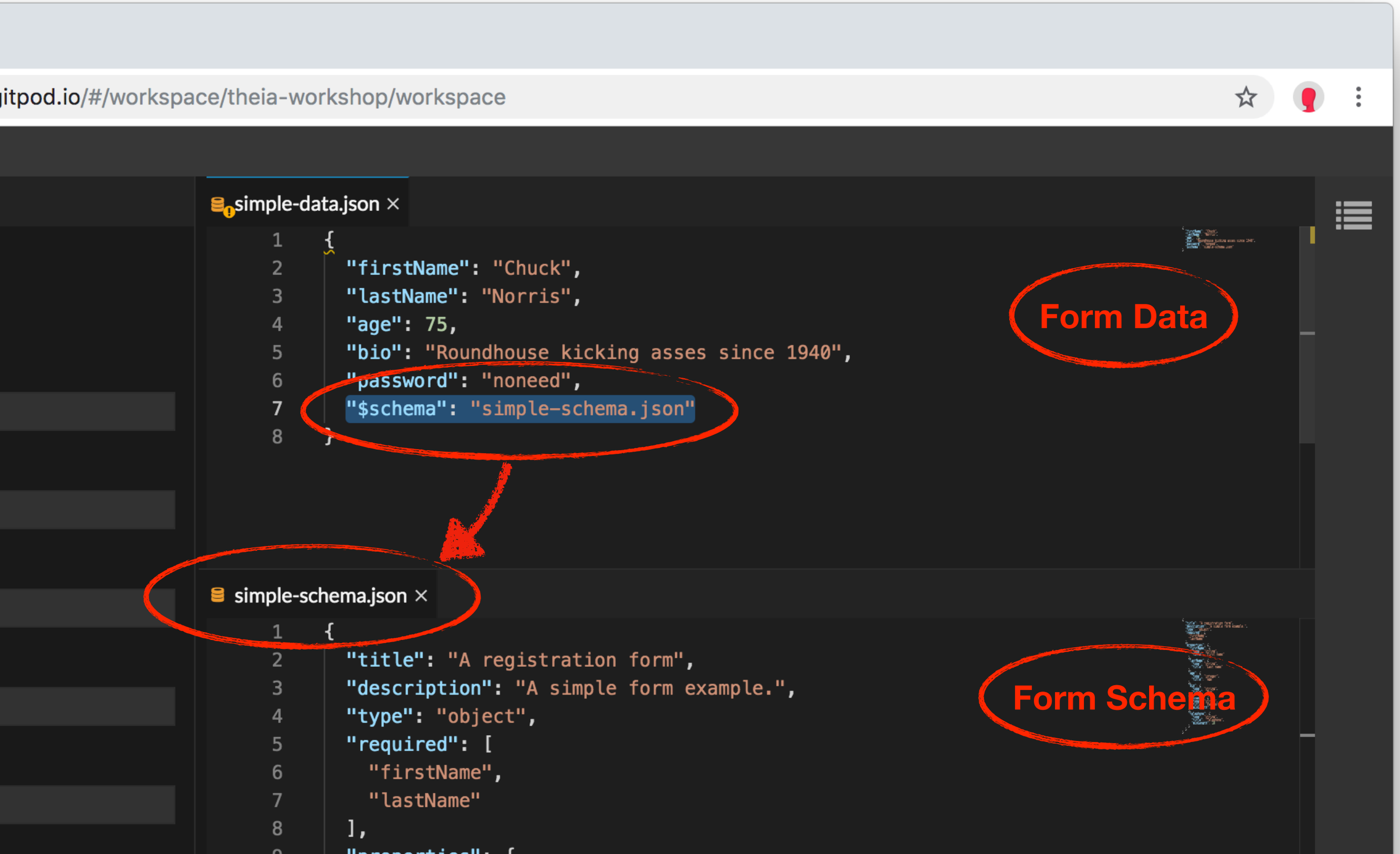
Data Flow


itpod.io/#/workspace/theia-workshop/workspace
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
Form Schema

Data Flow


itpod.io/#/workspace/theia-workshop/workspace
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
Form Schema

Data Flow


itpod.io/#/workspace/theia-workshop/workspace
simple-data.json
{
"firstName": "Chuck",
"lastName": "Norris",
"age": 75,
"bio": "Roundhouse kicking asses since 1940",
"password": "noneed",
"$schema": "simple-schema.json"
}
Form Data
simple-schema.json
{
"title": "A registration form",
"description": "A simple form example.",
"type": "object",
"required": [
"firstName",
"lastName"
],
Form Schema

Form Schema Tracking

```
new ReferencedModelStorage(
    model, modelService, '$schema', { default: {} }
);
```

Form Schema Storage

```
new ReferencedModelStorage(
    model, modelService, '$schema', { default: {} }
);
```

Form Data File

Form Schema Storage

```
new ReferencedModelStorage(
    model, modelService, '$schema', { default: {} }
);
```

A property name of a reference to Form Schema File

Form Data -> Form Schema

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText()) || {};
    this.schemaStorage.update(formData);
}
```

```
componentWillMount(): void {
   this.toDispose.push(this.schemaStorage.onDidChange(schema =>
       this.setState({ schema })
    ));
}
```

```
    render(): JSX.Element | null {
        const { schema, formData } = this.state;
        return <Form schema={schema} formData={formData} />;
    }
```

Form Data -> Form Schema

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText()) || {};
    this.schemaStorage.update(formData);
}
```

```
componentWillMount(): void {
   this.toDispose.push(this.schemaStorage.onDidChange(schema =>
       this.setState({ schema })
    ));
}
```

```
    render(): JSX.Element | null {
        const { schema, formData } = this.state;
        return <Form schema={schema} formData={formData} />;
    }
```

Form Schema -> UI State

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText()) || {};
    this.schemaStorage.update(formData);
}
```

```
componentWillMount(): void {
    this.toDispose.push(this.schemaStorage.onDidChange(schema =>
        this.setState({ schema })
    ));
}
```

```
    render(): JSX.Element | null {
        const { schema, formData } = this.state;
        return <Form schema={schema} formData={formData} />;
    }
```

UI State -> UI

```
protected async reconcileFormData(): Promise<void> {
    const formData = parse(this.props.model.getText()) || {};
    this.schemaStorage.update(formData);
}
```

```
componentWillMount(): void {
   this.toDispose.push(this.schemaStorage.onDidChange(schema =>
        this.setState({ schema })
    ));
}
```

```
    render(): JSX.Element | null {
        const { schema, formData } = this.state;
        return <Form schema={schema} formData={formData} />;
    }
```

Exercise 3

Implement UI Schema Support
for JSON-Form Widget

https://github.com/TypeFox/theia-workshop/tree/exercise-3

Get involved!

- Join community: https://spectrum.chat/theia
- Contribute: https://github.com/eclipse-theia/theia
- Learn more: https://www.typefox.io/trainings
- Self-Host with Gitpod Enterprise
  - https://www.gitpod.io/pricing/#enterprise