THE ART OF

EXPLICIT AND CONSISTENT

USER INTERFACES

@farzad_yz

FARZAD YOUSEF ZADEH

@farzad_yz

SENIOR SOFTWARE ENGINEER

FORMER AEROSPACE ENGINEER 🚀

AND ASTROPHYSICIST 🌌

WE WILL TALK ABOUT

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

LOGIC

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

LOGIC

- MODELED

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

LOGIC

- MODELED
- VISUALIZED

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

LOGIC

- MODELED
- VISUALIZED
- DECOUPLED FROM IMPLEMENTATION

WE WILL TALK ABOUT

IMPLICIT VS EXPLICIT STATE

FINITE STATE vs INFINITE STATE

LOGIC

- MODELED
- VISUALIZED
- DECOUPLED FROM IMPLEMENTATION
- PORTED TO SEVERAL PLATFORMS

Scene #1

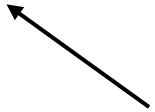
THE WHAT

State management tool

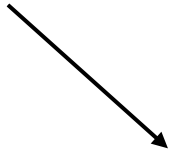
Tool specific approaches

Local state

Global state



```
renderer (state) = view
```



Component library / Framework

ONE PURPOSE

MULTIPLE APPROACHES

```
this.state = {
  title: "";
}

this.setState({title: "HolyJS"})
```

```
const [title, setTitle] =
useState("")

setTitle("HolyJS")
```

```
store.dispatch({
  type: "UPDATE_TITLE",
  payload: "Holyjs"
})
```

```
@action onClick() {
  this.props.title = "HolyJS"
}
```

```
new Vuex.Store({
  state: {
    title: ""
  },
  mutations: {
    updateTitle: (state, payload)
{
      state.title = payload.title
    }
  }
})
```

STATE MANAGEMENT TOOLS

ARE CONTAINERS

STATE MANAGEMENT TOOLS

ARE CONTAINERS

DEAL WITH THE ARCHITECTURE

(FLUX, PUSH BASED, PULL BASED)

STATE MANAGEMENT TOOLS

ARE CONTAINERS

DEAL WITH THE ARCHITECTURE (FLUX, PUSH BASED, PULL BASED)

TELL YOU HOW TO STORE STATES (SINGLE SOURCE, DISTRIBUTED)

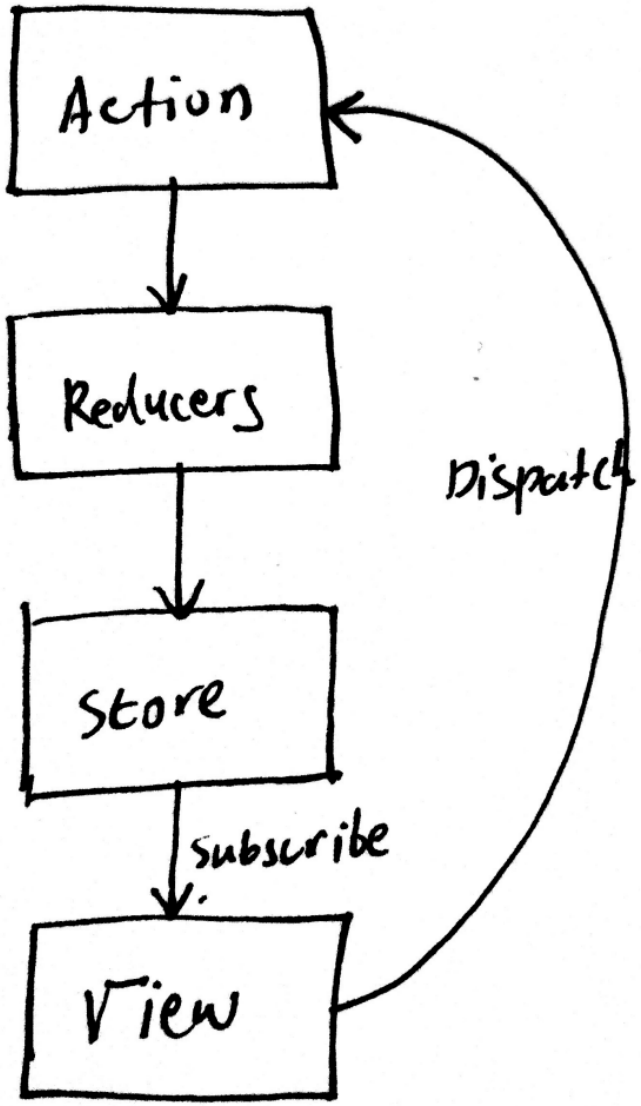
STATE MANAGEMENT TOOLS ARE CONTAINERS

DEAL WITH THE ARCHITECTURE (FLUX, PUSH BASED, PULL BASED)

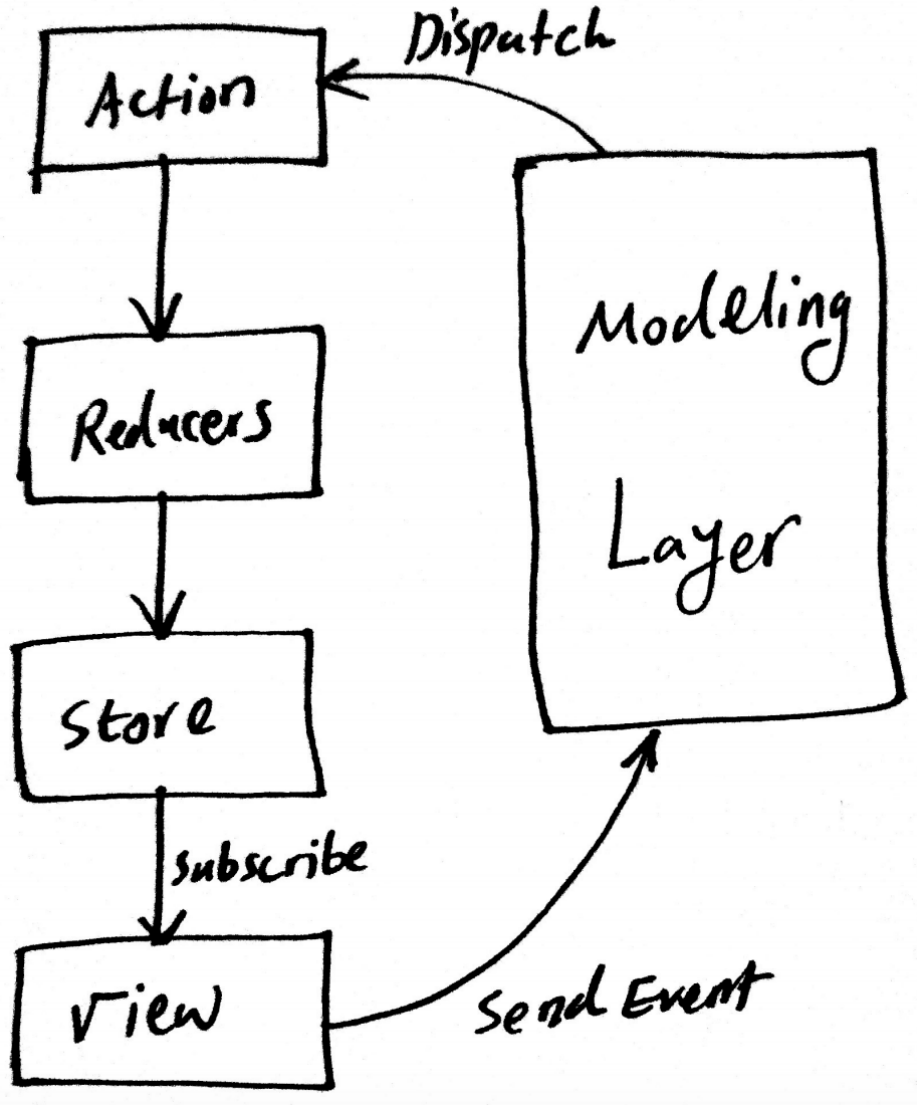
TELL YOU HOW TO STORE STATES (SINGLE SOURCE, DISTRIBUTED)

TELL YOU HOW TO UPDATE STATES (MUTABLE, IMMUTABLE, OBSERVABLE)

CURRENT STATE FLOW



AFTER ADDING MODELING LAYER



source: http://tiny.cc/eg2hfz

A FLASHBACK TO GUI HISTORY

Alan Kay, the Dynabook, 1972

GUI is built upon **events** and **messages**

GUI is **event-driven**

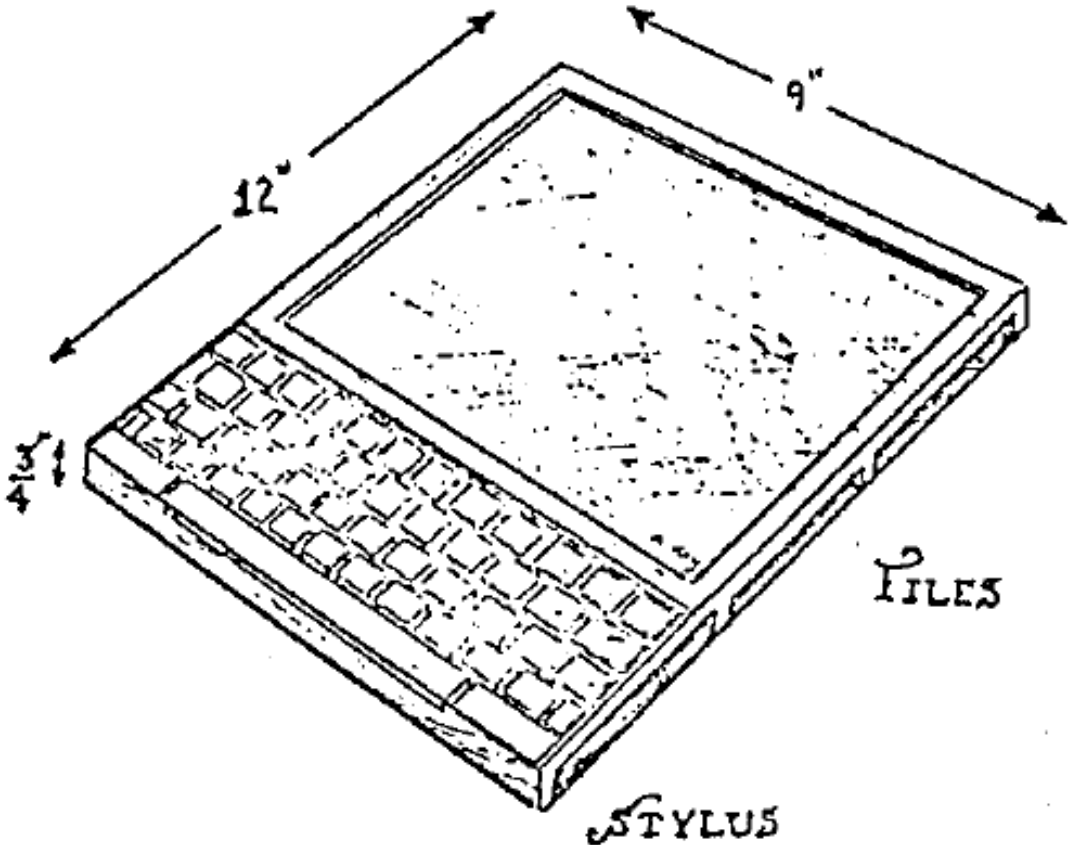
Listen to events and execute actions (side effects)

EVENT + ACTION PARADIGM

The DynaBook

"I wish to God these calculations were executed by steam!"
Charles Babbage (age 19)
ca. 1803

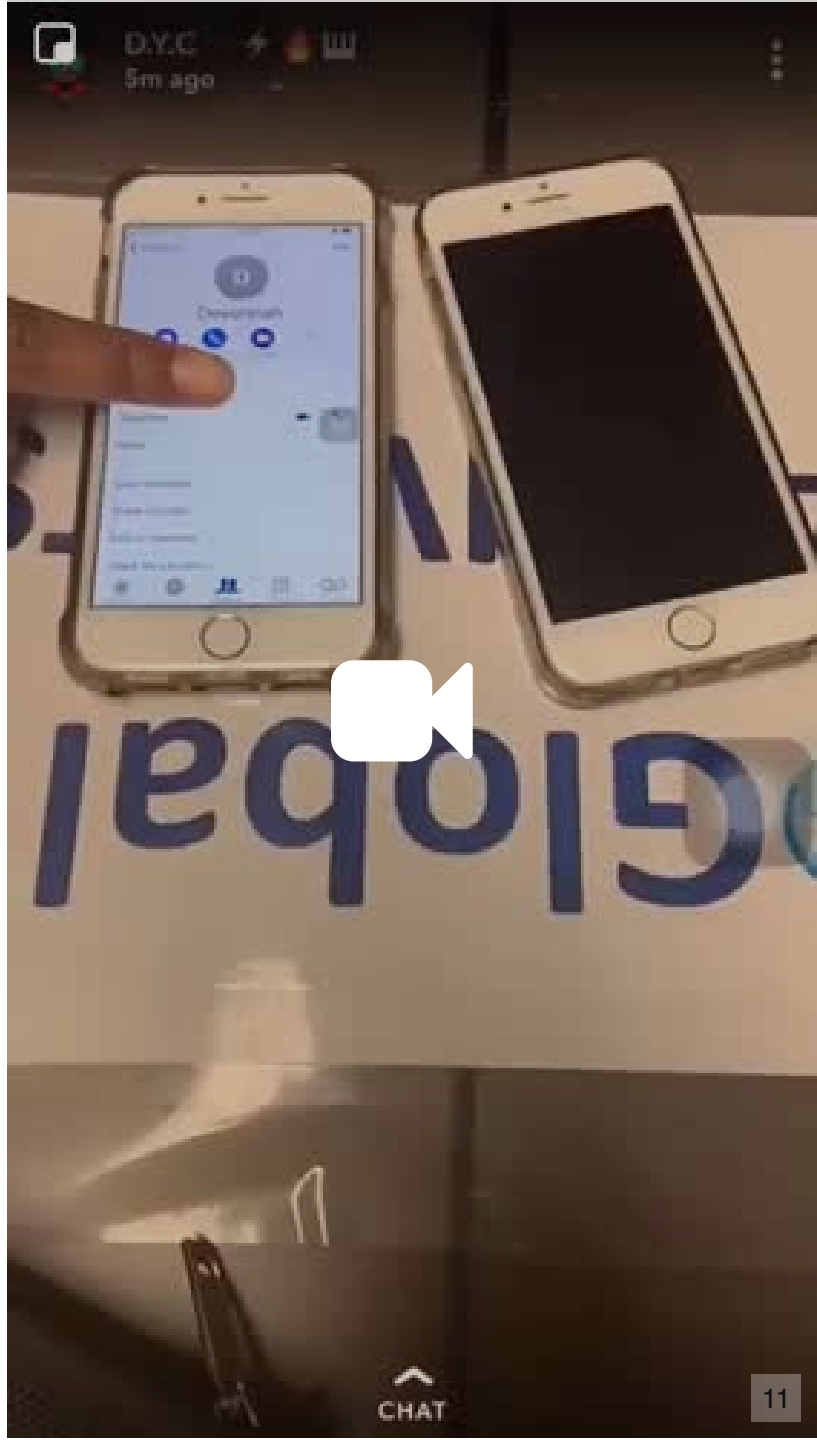
"The Analytical Engine weaves algebraic patterns, just as the Jacquard Loom weaves patterns in silk."
-Ada Augusta
Countess of Lovelace



Scene #2

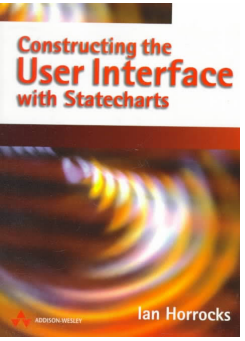
THE PROBLEM

FACETIME BUG



MS CALCULATOR BUG

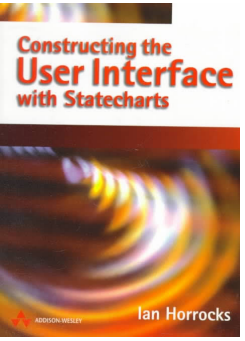
MS CALCULATOR BUG



Constructing the User Interface with Statecharts

Ian Horrocks 1999

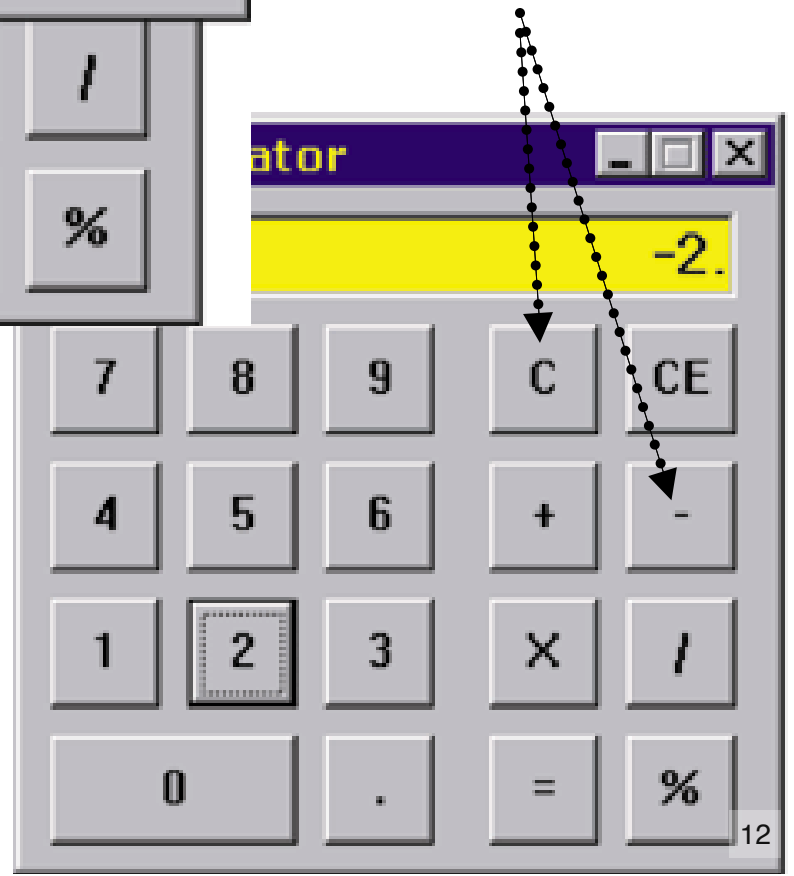
MS CALCULATOR BUG



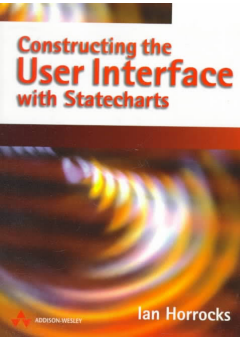
Constructing the User Interface with Statecharts

Ian Horrocks 1999

Press the (-) button



MS CALCULATOR BUG

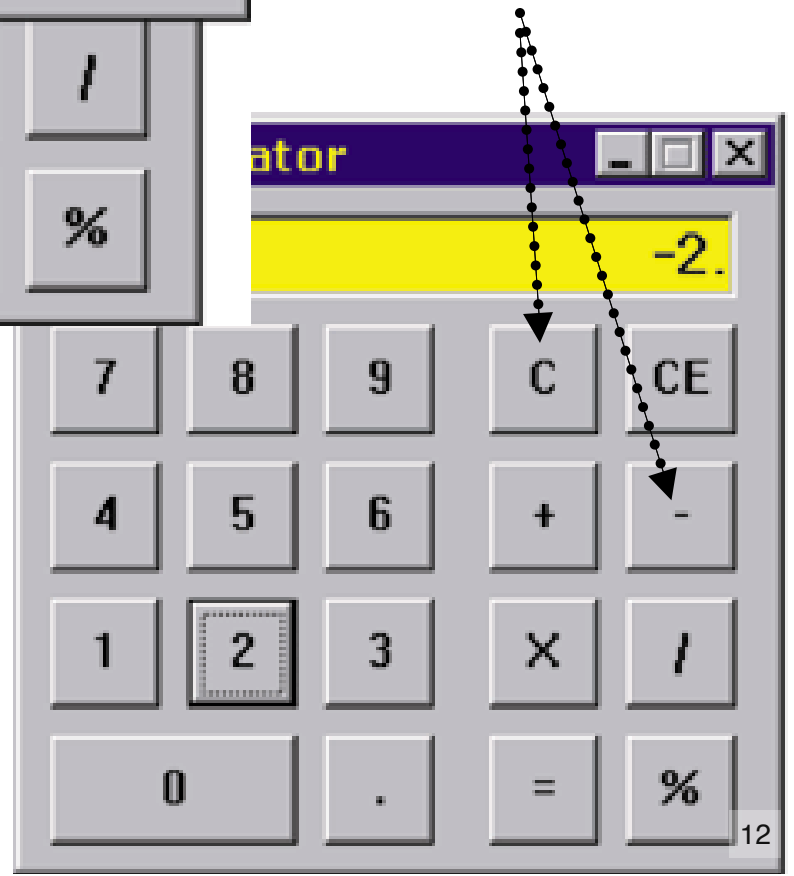


Constructing the User Interface with Statecharts

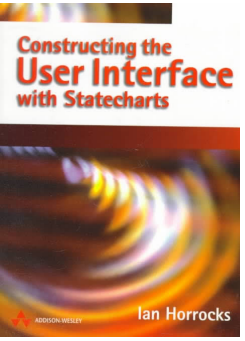
Ian Horrocks 1999

Event handler's logic based on previously happened events

Press the (-) button



MS CALCULATOR BUG



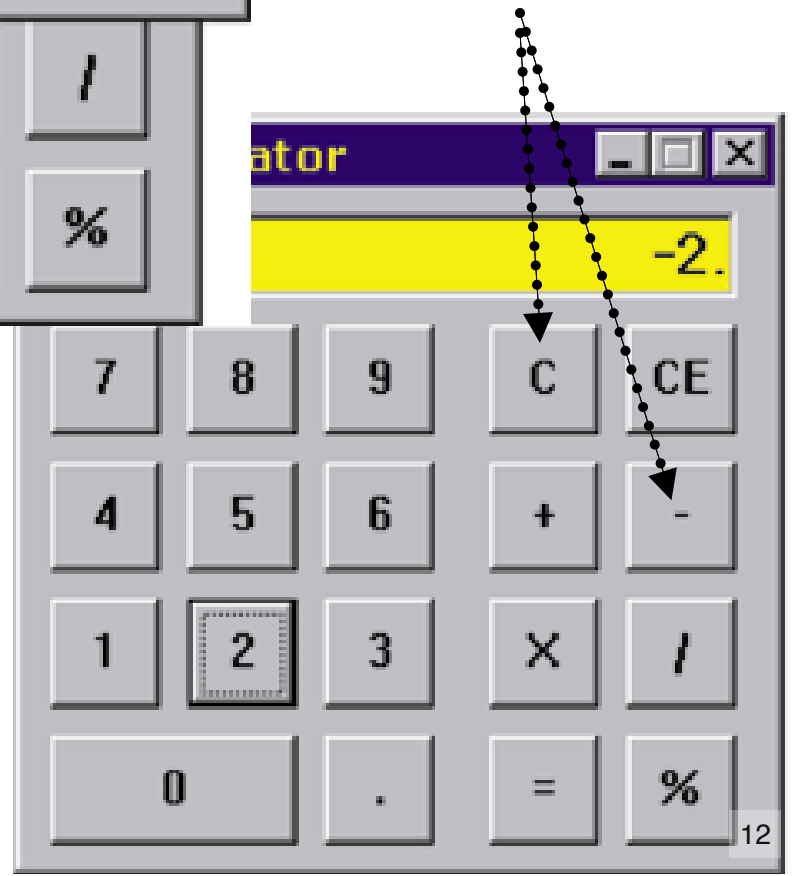
Constructing the User Interface with Statecharts

Ian Horrocks 1999

Event handler's logic based on previously happened events

Context aware event handlers

Press the (-) button



WHAT WAS WRONG WITH THOSE THEN?

IMPLICITLY HANDLING THE STATE

WHAT WAS WRONG WITH THOSE THEN?

IMPLICITLY HANDLING THE STATE

EVENT HANDLERS

```
button.addEventListener("click", ...)
```

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount)
{
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {

$(markup).insertBefore(_.$slides.eq(index));
  } else {

$(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

WHAT WAS WRONG WITH THOSE THEN?

IMPLICITLY HANDLING THE STATE

EVENT HANDLERS

```
button.addEventListener("click", ...)
```

MUTATIONS

```
this.setState({ ... })
```

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount)
{
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {

$(markup).insertBefore(_.$slides.eq(index));
  } else {

$(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

WHAT WAS WRONG WITH THOSE THEN?

IMPLICITLY HANDLING THE STATE

EVENT HANDLERS

```
button.addEventListener("click", ...)
```

MUTATIONS

```
this.setState({ ... })
```

ASYNCHRONY

```
new Promise(...)
```

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount)
{
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {

$(markup).insertBefore(_.$slides.eq(index));
  } else {

$(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

WHAT WAS WRONG WITH THOSE THEN?

IMPLICITLY HANDLING THE STATE

EVENT HANDLERS

```
button.addEventListener("click", ...)
```

MUTATIONS

```
this.setState({ ... })
```

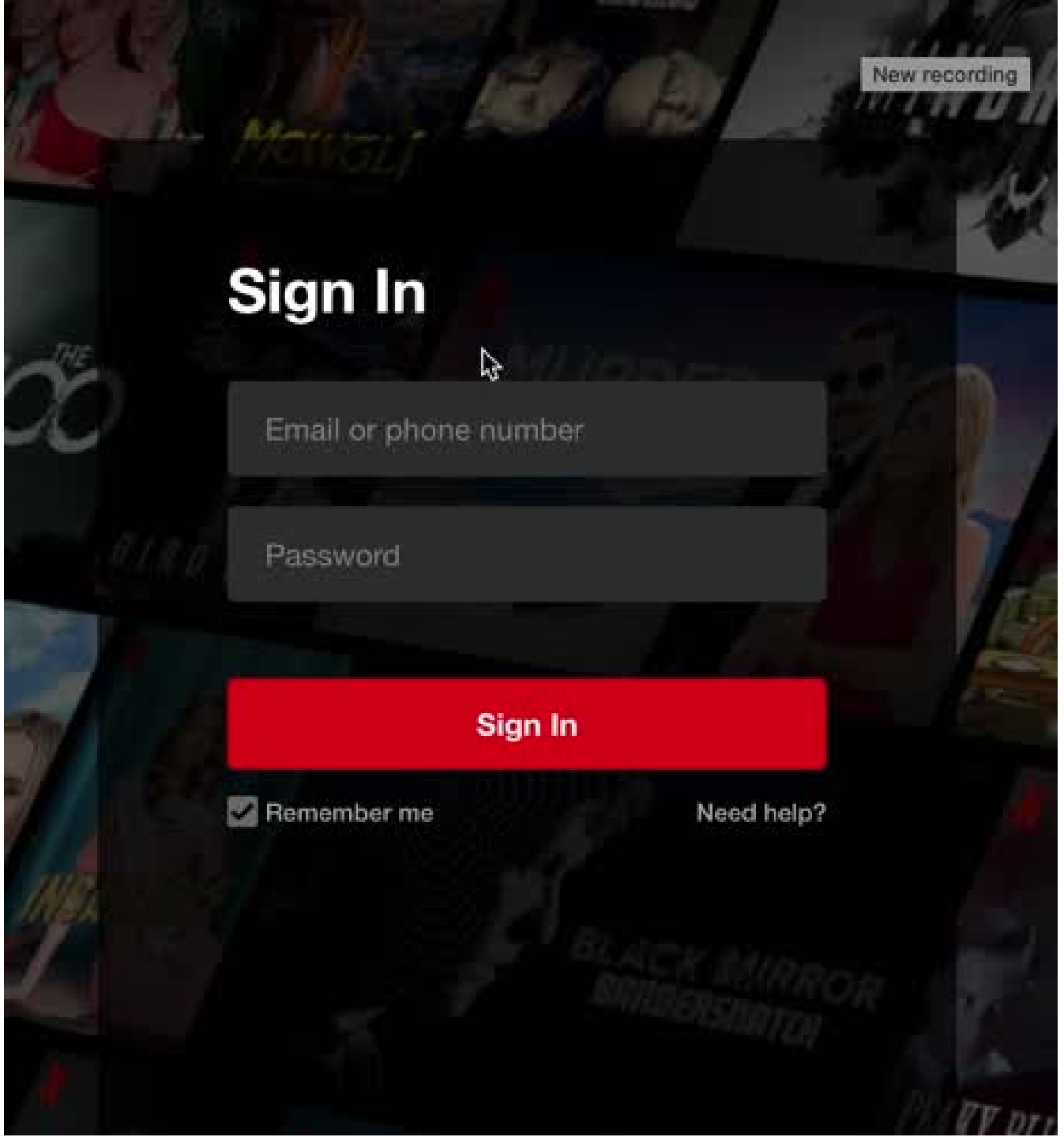
ASYNCHRONY

```
new Promise(...)
```

SIDE EFFECTS

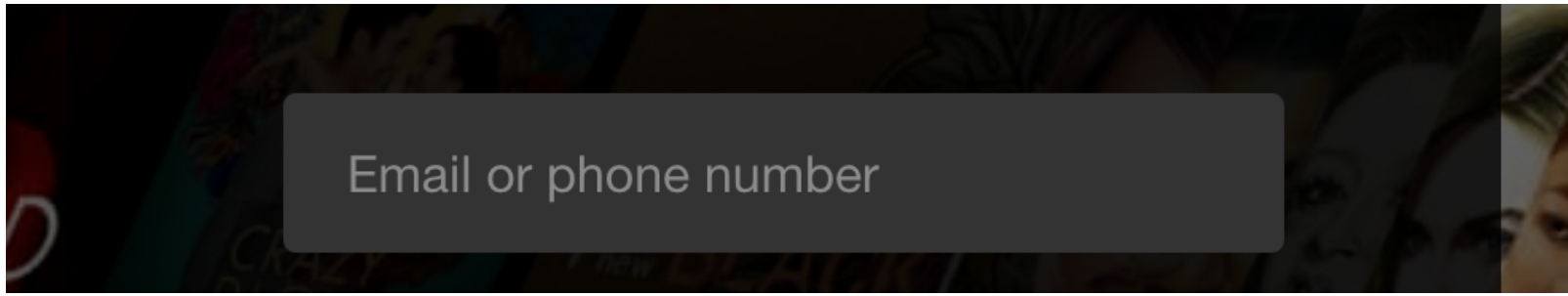
```
window.fetch(...)
```

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount)
{
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {

$(markup).insertBefore(_.$slides.eq(index));
  } else {

$(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```


New recording
Sign In
Email or phone number
Password
Sign In
Remember me
Need help?

A SINGLE INPUT WITH IMPLICIT STATE

```
{
  value: "",
  valid: false
}
```



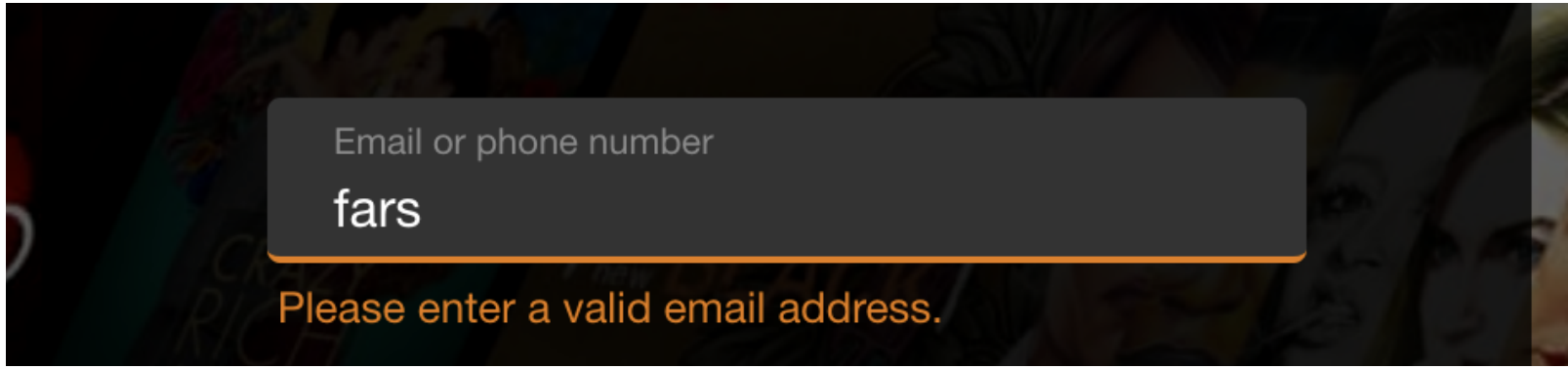
NO VALUE

INVALID

NO VALIDATION ERROR

A SINGLE INPUT WITH IMPLICIT STATE

```
{
  value: "fars",
  valid: false
}
```



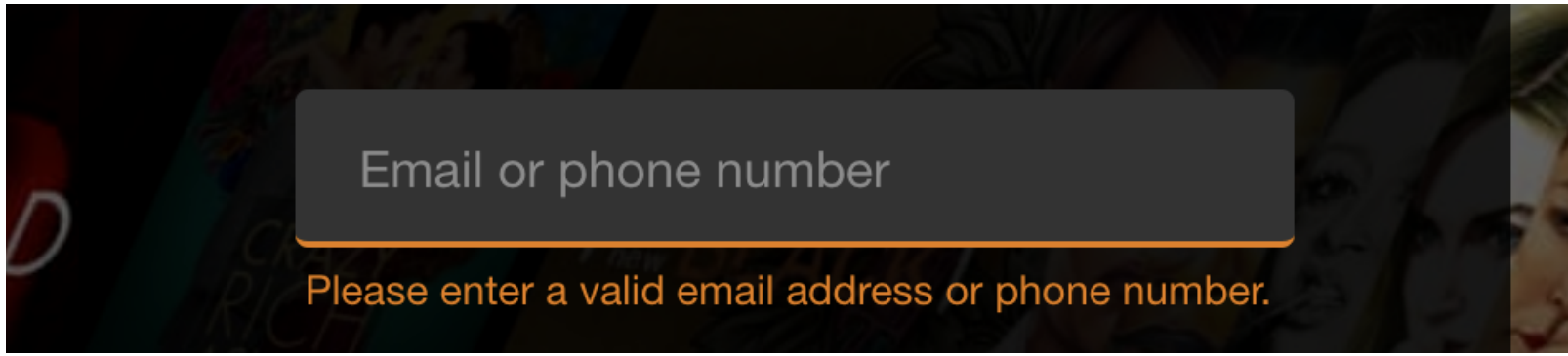
WITH VALUE

INVALID

VALIDATION ERROR SHOWN

A SINGLE INPUT WITH IMPLICIT STATE

```
{
  value: "",
  valid: false
}
```



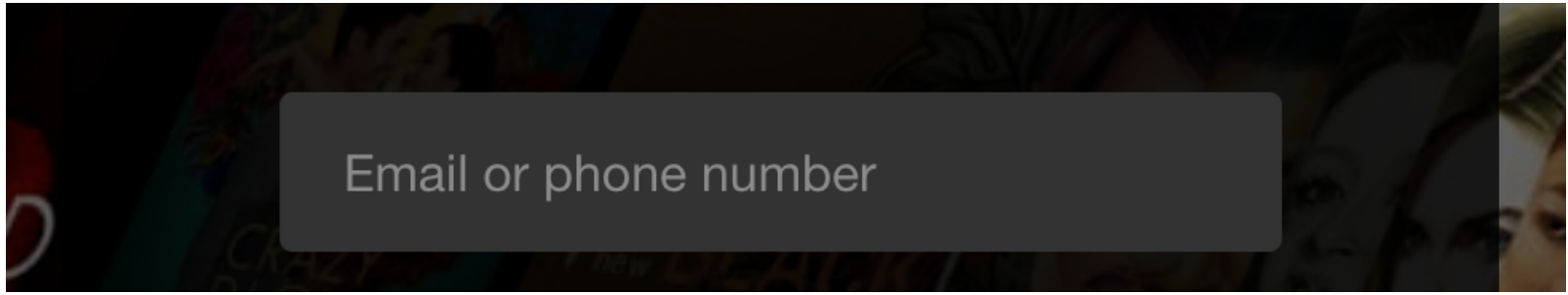
NO VALUE

INVALID

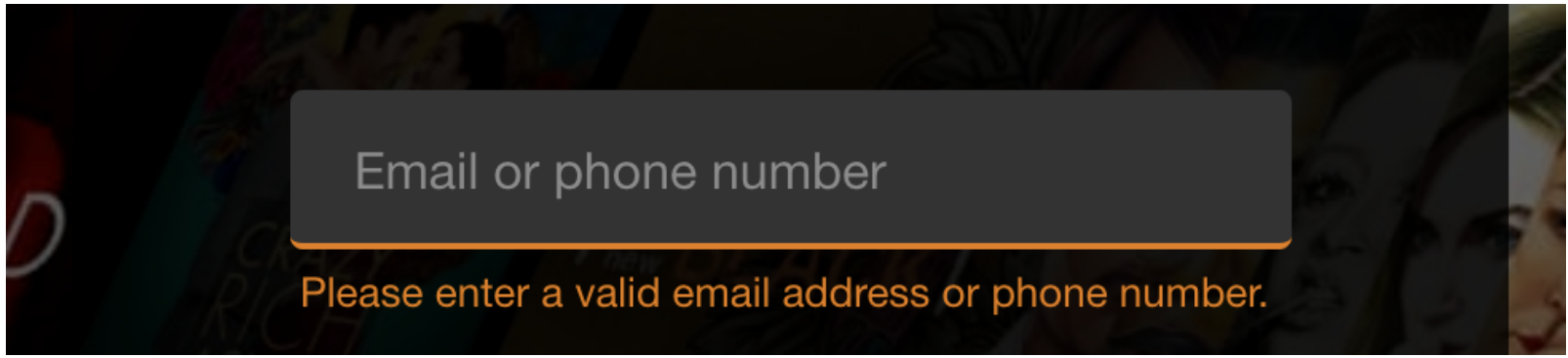
VALIDATION ERROR SHOWN

UH OH!

```
{
  value: "",
  valid: false
}
```

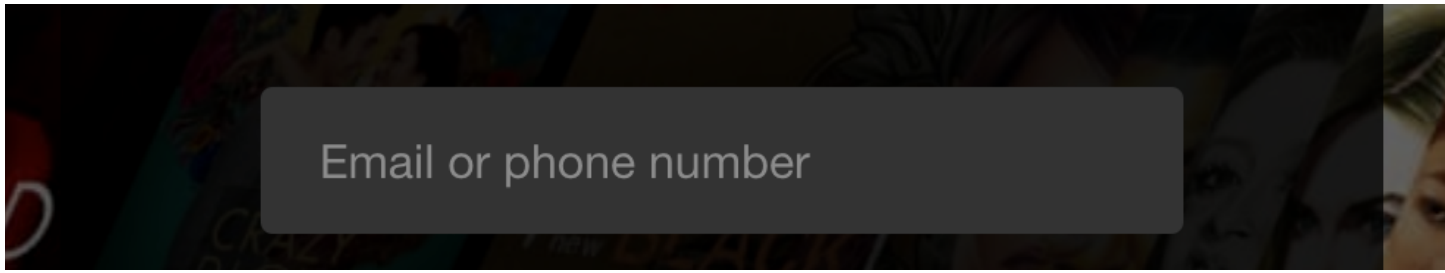



```
{
  value: "",
  valid: false
}
```



A SINGLE INPUT WITH IMPLICIT STATE

```
1 {
2     value: "",
3     valid: false,
4     isValidated: false
5 }
```



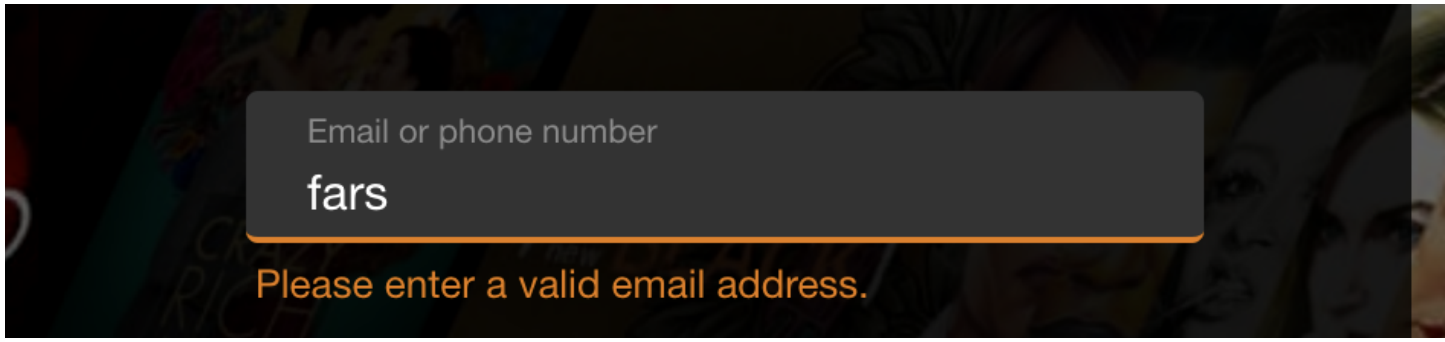
NO VALUE

INVALID

NO VALIDATION ERROR

A SINGLE INPUT WITH IMPLICIT STATE

```
1 {
2     value: "fars",
3     valid: false,
4     isValidated: true
5 }
```



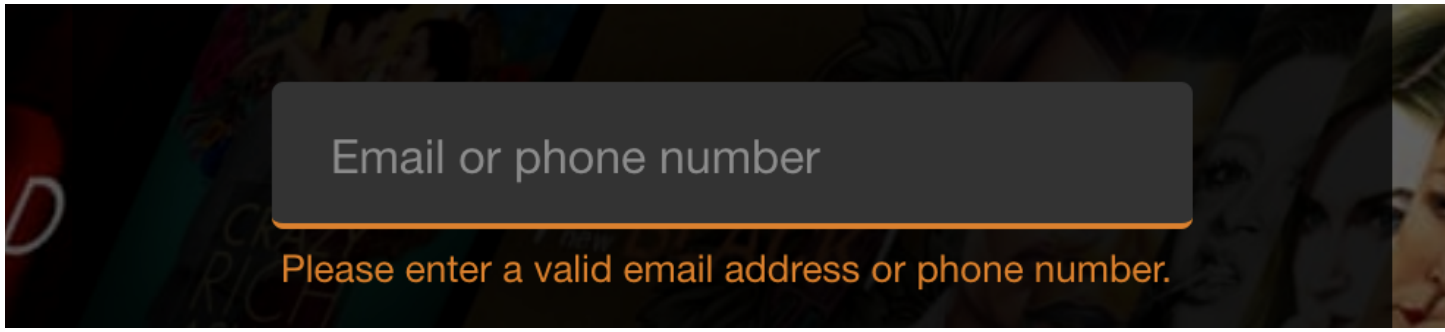
WITH VALUE

INVALID

VALIDATION ERROR SHOWN

A SINGLE INPUT WITH IMPLICIT STATE

```
{
    value: "",
    valid: false,
    isValidated: true
}
```



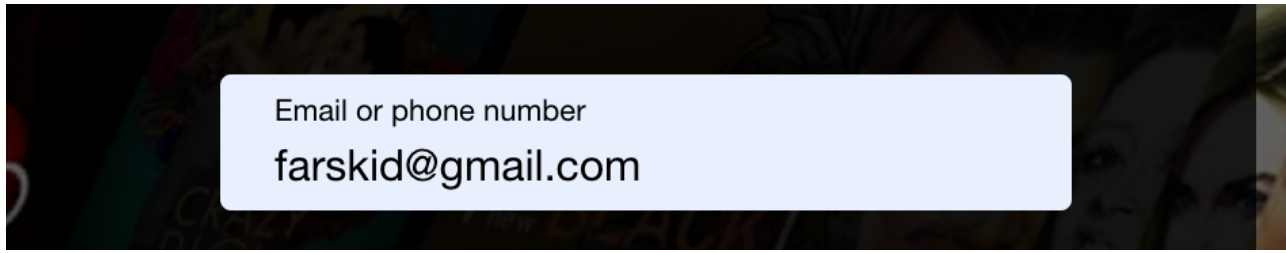
NO VALUE

INVALID

VALIDATION ERROR SHOWN

A SINGLE INPUT WITH IMPLICIT STATE

```
1 {
2   value:
  "farskid@gmail.com",
3   valid: true,
4   isValidated: true
5 }
```



WITH VALUE

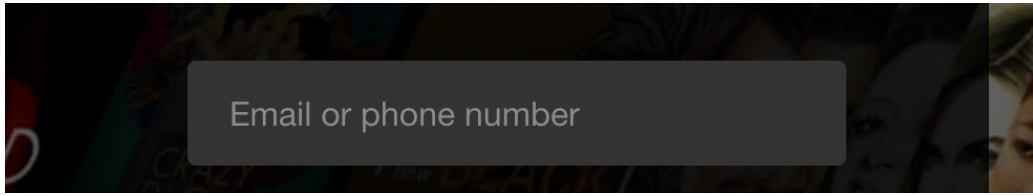
VALID

NO VALIDATION ERROR

value.length	= 0	> 0
valid	true	false
isValidated	true	false

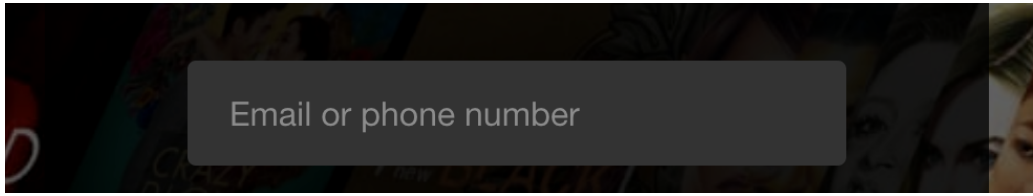
value.length	= 0	> 0
valid	true	false
isValidated	true	false

`value.length = 0, valid: false, isValidated: false`

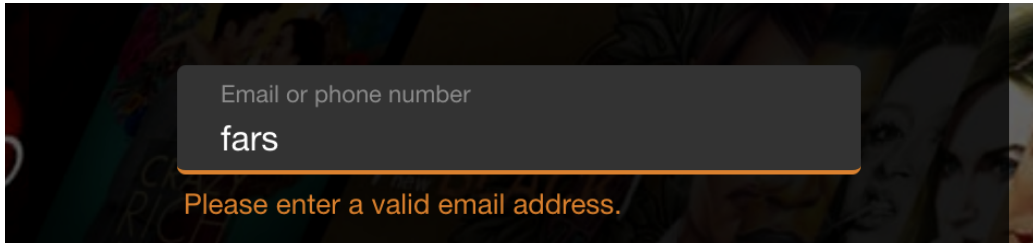


value.length	= 0	> 0
valid	true	false
isValidated	true	false

`value.length = 0, valid: false, isValidated: false`

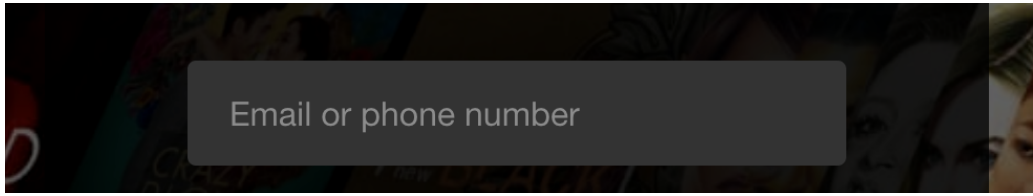


`value.length > 0, valid: false, isValidated: true`

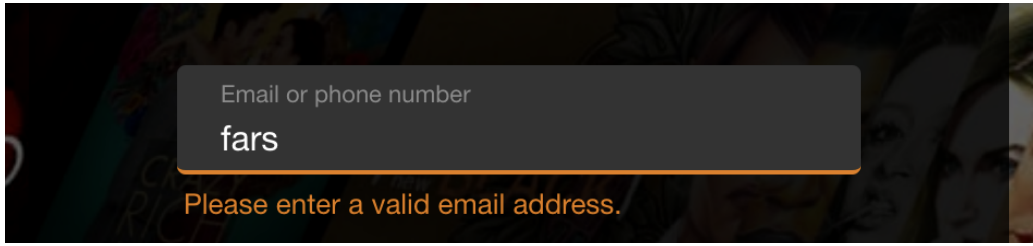


value.length	= 0	> 0
valid	true	false
isValidated	true	false

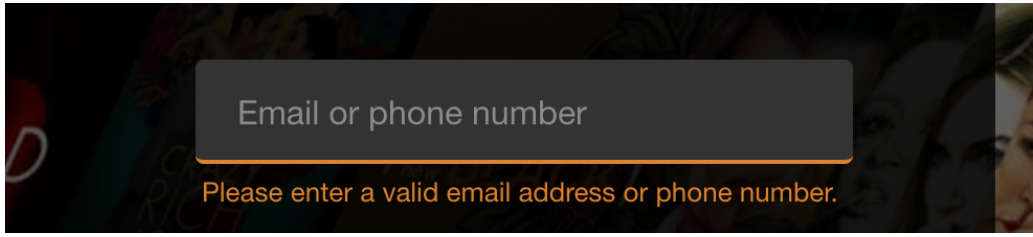
`value.length = 0, valid: false, isValidated: false`



`value.length > 0, valid: false, isValidated: true`

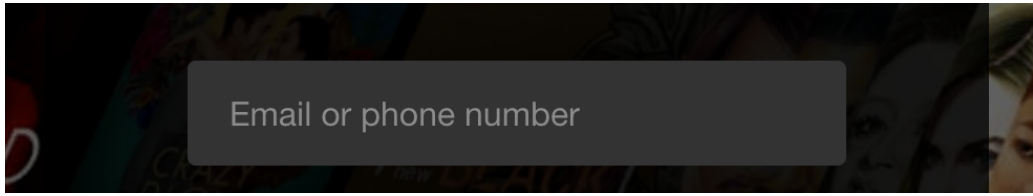


`value.length = 0, valid: false, isValidated: true`

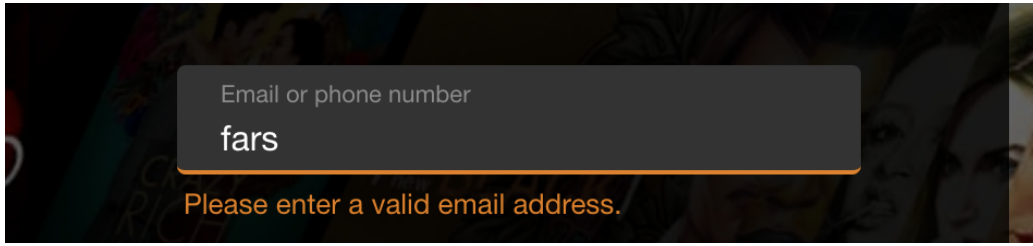


value.length	= 0	> 0
valid	true	false
isValidated	true	false

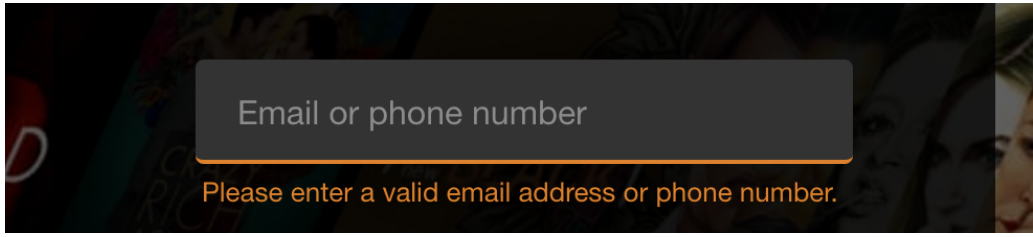
`value.length = 0, valid: false, isValidated: false`



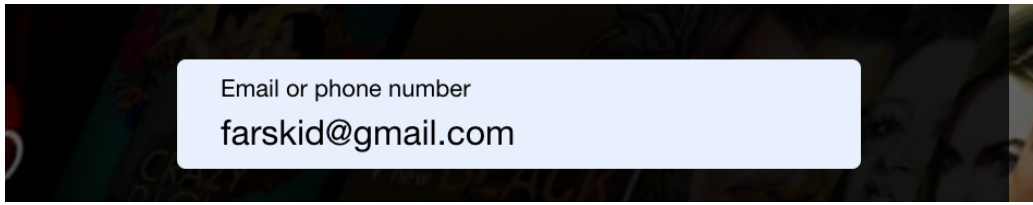
`value.length > 0, valid: false, isValidated: true`



`value.length = 0, valid: false, isValidated: true`



`value.length > 0, valid: true, isValidated: true`



value.length	= 0	> 0
valid	true	false
isValidated	true	false

value.length	= 0	> 0
valid	true	false
isValidated	true	false

2

```
v.length
```

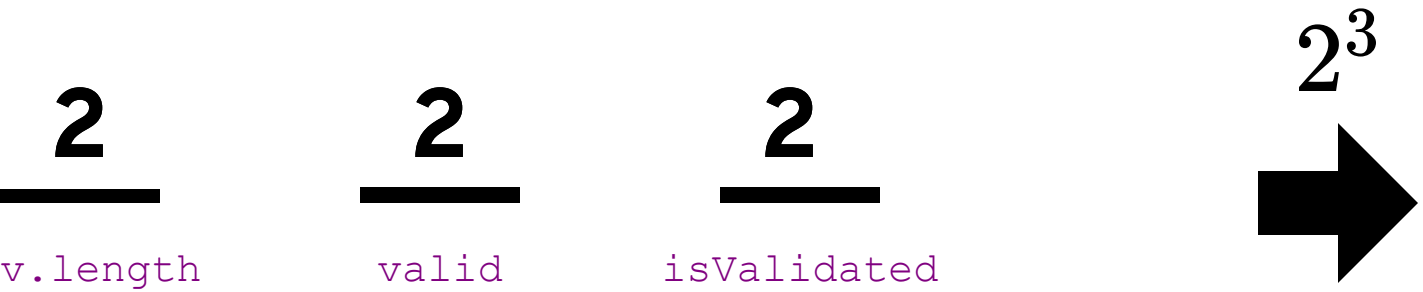
value.length	= 0	> 0
valid	true	false
isValidated	true	false

2 **2**

`v.length` `valid`

value.length	= 0	> 0
valid	true	false
isValidated	true	false

value.length	= 0	> 0
valid	true	false
isValidated	true	false



value.length	= 0	> 0
valid	true	false
isValidated	true	false

2 **2** **2**

`v.length` `valid` `isValidated`

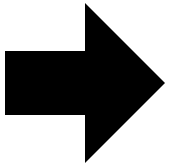
2^3

valid states: **4**

value.length	= 0	> 0
valid	true	false
isValidated	true	false

2 `v.length` **2** `valid` **2** `isValidated`

2^3



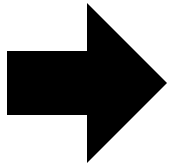
valid states: **4**

impossible states: **4**

value.length	= 0	> 0
valid	true	false
isValidated	true	false

2 **2** **2** 2^3

`v.length` `valid` `isValidated`



valid states: **4**

impossible states: **4**

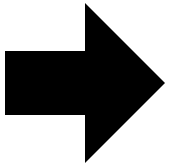
```
value.length = 0, valid: true, isValidated: false

value.length = 0, valid: true, isValidated: true

value.length > 0, valid: false, isValidated: false

value.length > 0, valid: true, isValidated: false
```

value.length	= 0	> 0
valid	true	false
isValidated	true	false

2 **2** **2**

v.length valid isValidated

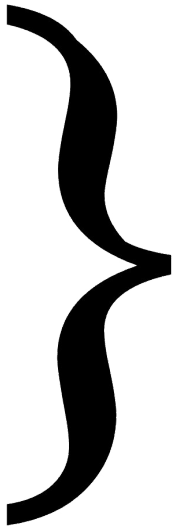
2^3



valid states: **4**

impossible states: **4**

```
value.length = 0, valid: true, isValidated: false
value.length = 0, valid: true, isValidated: true
value.length > 0, valid: false, isValidated: false
value.length > 0, valid: true, isValidated: false
```



IMPOSSIBLE STATES

WITH BAD MODELING,

YOUR CODE COMPLEXITY GROWS FASTER THAN THE DOMAIN COMPLEXITY

WITH BAD MODELING,

YOUR CODE COMPLEXITY GROWS FASTER THAN THE DOMAIN COMPLEXITY

WITH IMPOSSIBLE STATES,

YOU NEED TO TEST CASES THAT WON'T EVENT HAPPEN IN REAL LIFE

AN IMPOSSIBLE STATE IS WHERE YOU TELL USERS TO RESTART

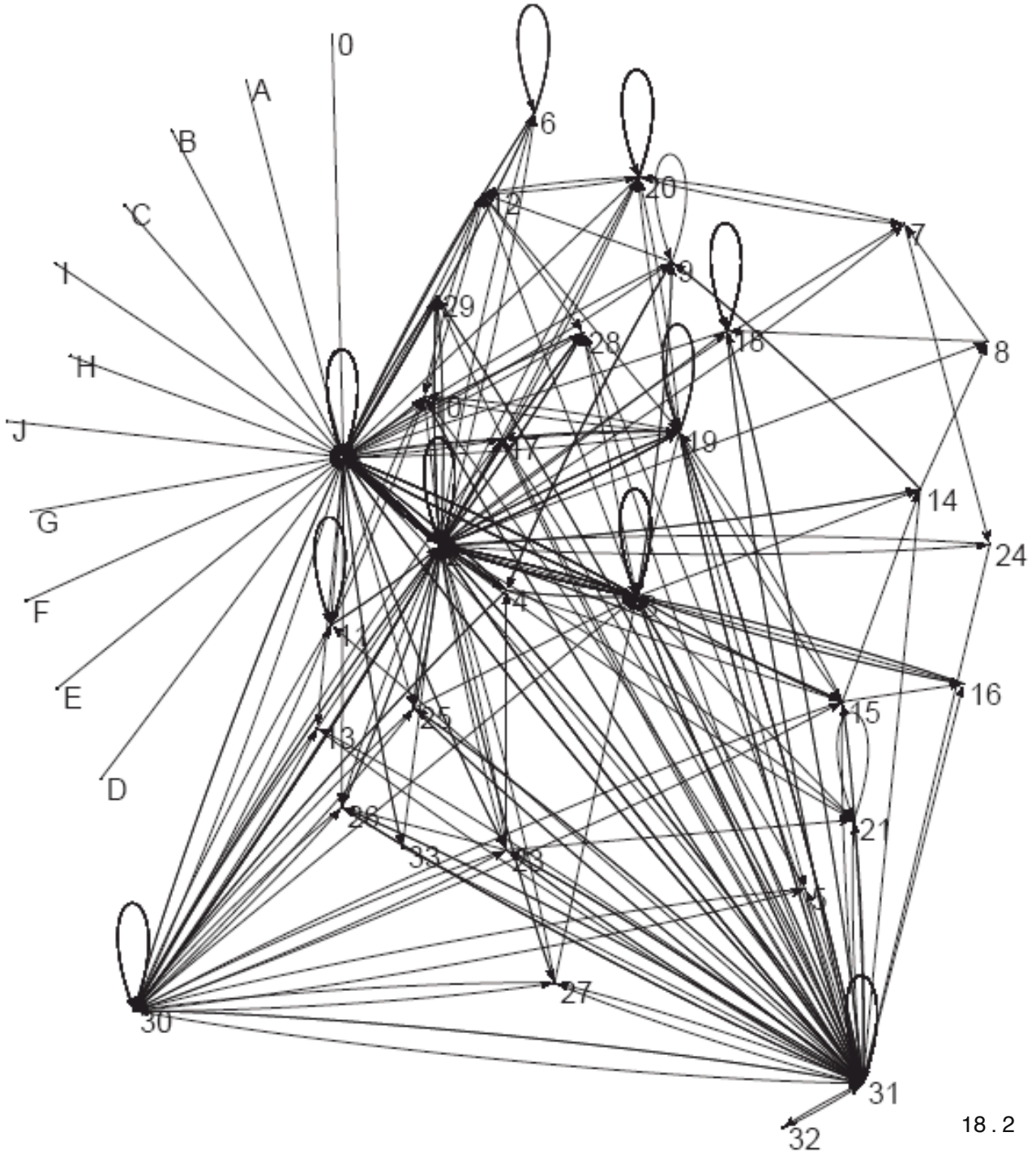
ADD GUARDS TO AVOID IMPOSSIBLE STATES

```
function handleChange(newValue) {
  if (!isValidated) {
    setValidated(true);
  }

  if (newValue.length === 0 || !validateEmail(newValue)) {
    setValid(false);
  } else {
    setValid(true);
  }

  setValue(newValue);
}
```

GUARDS INCREASE

CYCLOMATIC COMPLEXITY

NUMBER OF INDEPENDENT PATHS A PROGRAM CAN TAKE



GUARDS INCREASE CYCLOMATIC COMPLEXITY

NUMBER OF INDEPENDENT PATHS A PROGRAM CAN TAKE

if, else,
while, for,
switch, case

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount) {
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {
    $(markup).insertBefore(_.$slides.eq(index));
  } else {
    $(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

GUARDS INCREASE CYCLOMATIC COMPLEXITY

NUMBER OF INDEPENDENT PATHS A PROGRAM CAN TAKE

`if, else,`
`while, for,`
`switch, case`

MORE GUARDS:

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount) {
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {
    $(markup).insertBefore(_.$slides.eq(index));
  } else {
    $(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

GUARDS INCREASE CYCLOMATIC COMPLEXITY

NUMBER OF INDEPENDENT PATHS A PROGRAM CAN TAKE

`if, else,`
`while, for,`
`switch, case`

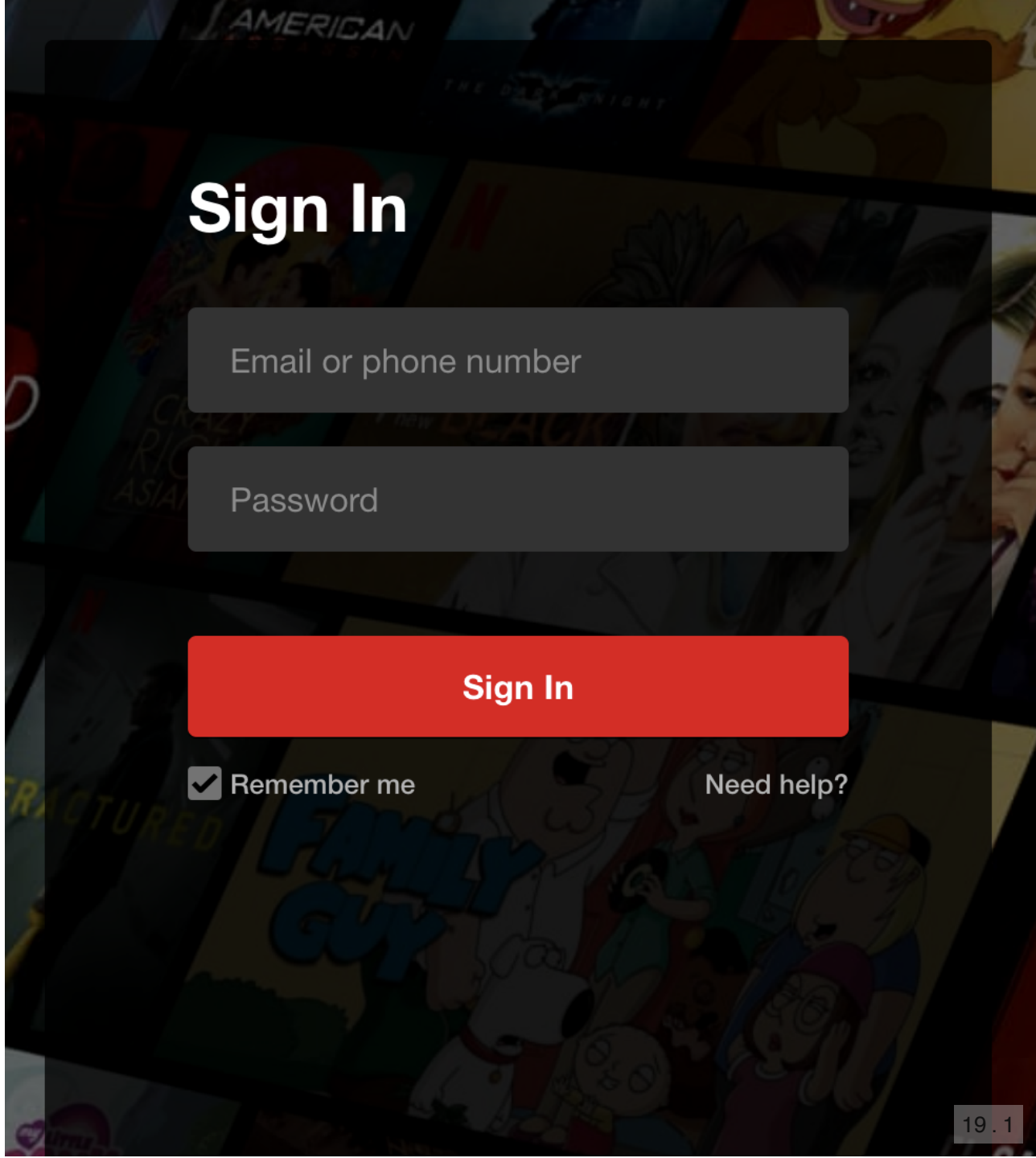
MORE GUARDS:

- HIGHER CYCLOMATIC COMPLEXITY
- LESS PREDICTABLE LOGIC
- Harder to track logic

```
if (typeof index === "boolean") {
  addBefore = index;
  index = null;
} else if (index < 0 || index >= _.slideCount) {
  return false;
}

_.unload();

if (typeof index === "number") {
  if (index === 0 && _.$slides.length === 0) {
    $(markup).appendTo(_.$slideTrack);
  } else if (addBefore) {
    $(markup).insertBefore(_.$slides.eq(index));
  } else {
    $(markup).insertAfter(_.$slides.eq(index));
  }
} else {
  if (addBefore === true) {
    $(markup).prependTo(_.$slideTrack);
  } else {
    $(markup).appendTo(_.$slideTrack);
  }
}
```

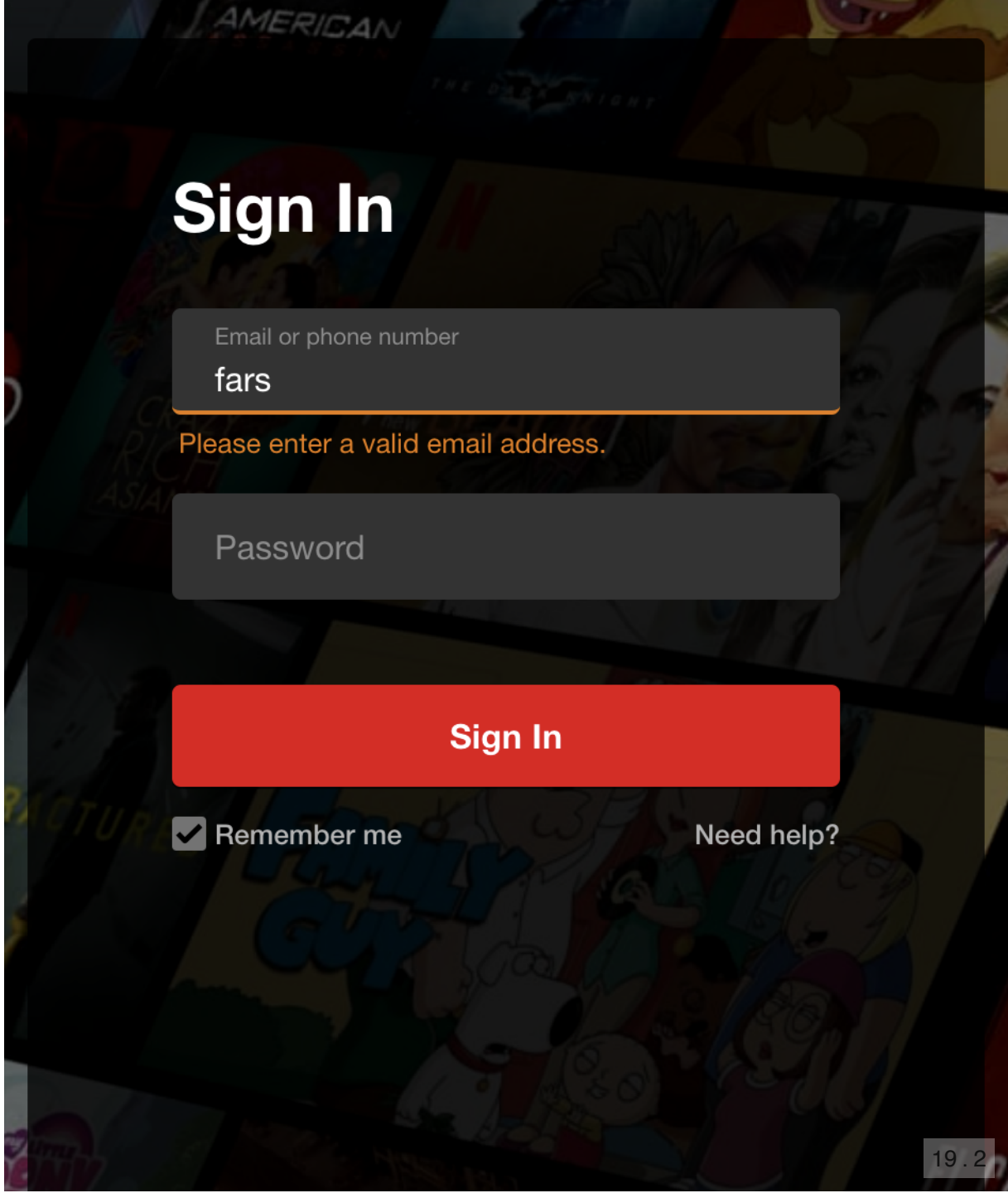
EMBED INPUT IN FORM

```
{
  inputs: {
    email: {
      value: "",
      valid: false,
      isValidated: false
    }
  },
  submitting: false,
  submitError: undefined
}
```



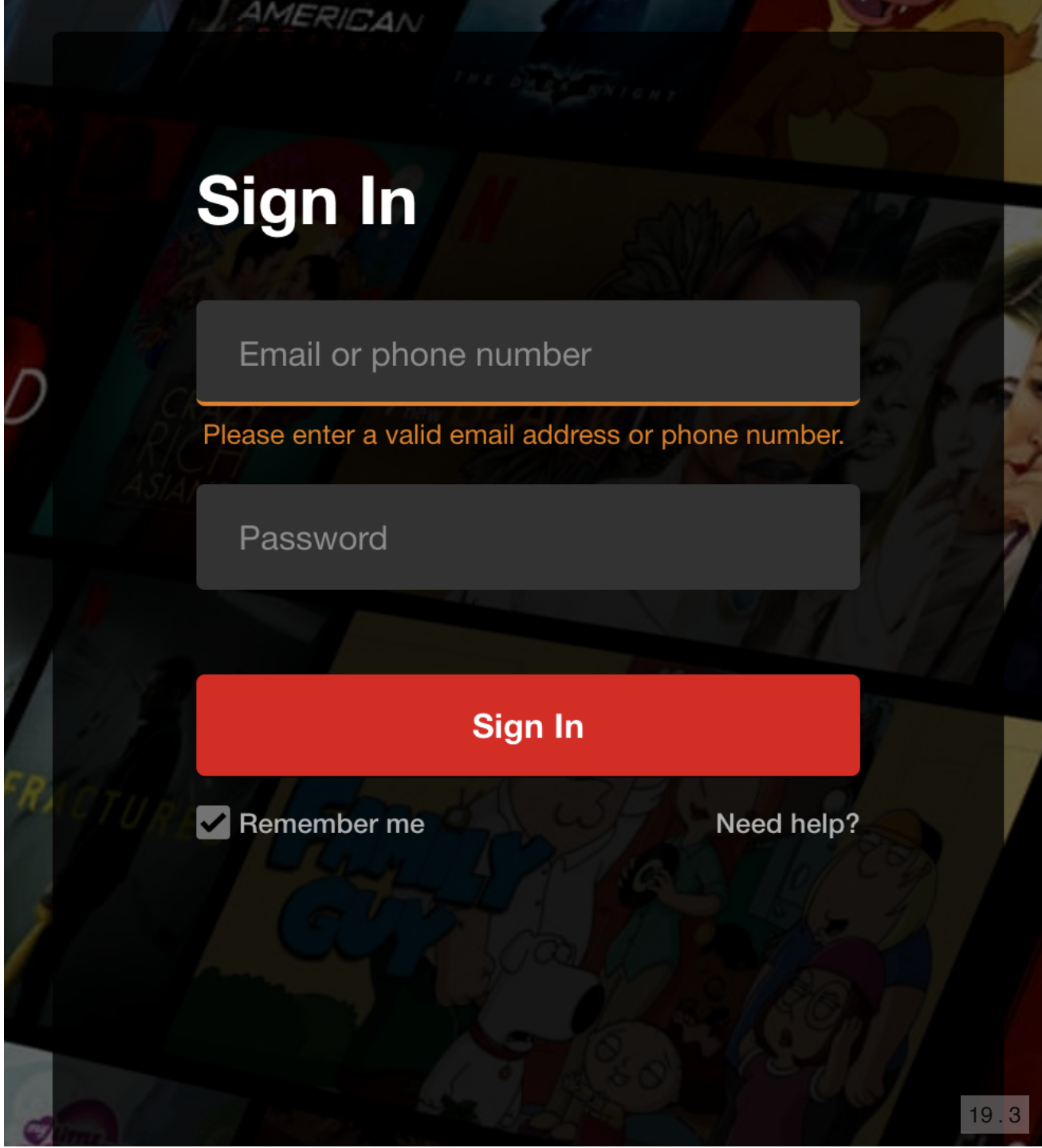
EMBED INPUT IN FORM

```
{
  inputs: {
    email: {
      value: "fars",
      valid: false,
      isValidated: true
    }
  },
  submitting: false,
  submitError: undefined
}
```



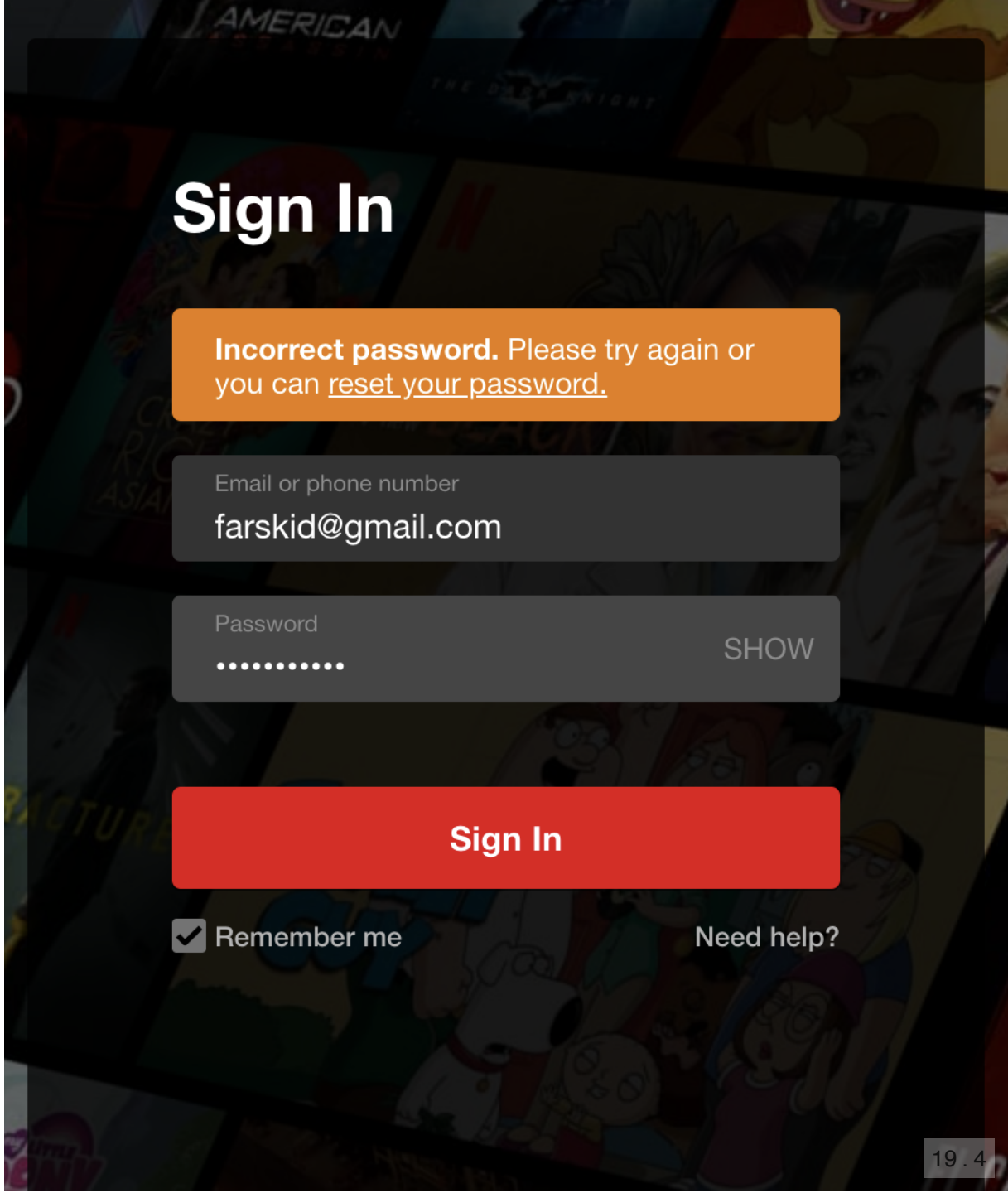
EMBED INPUT IN FORM

```
{
  inputs: {
    email: {
      value: "",
      valid: false,
      isValidated: true
    }
  },
  submitting: false,
  submitError: undefined
}
```

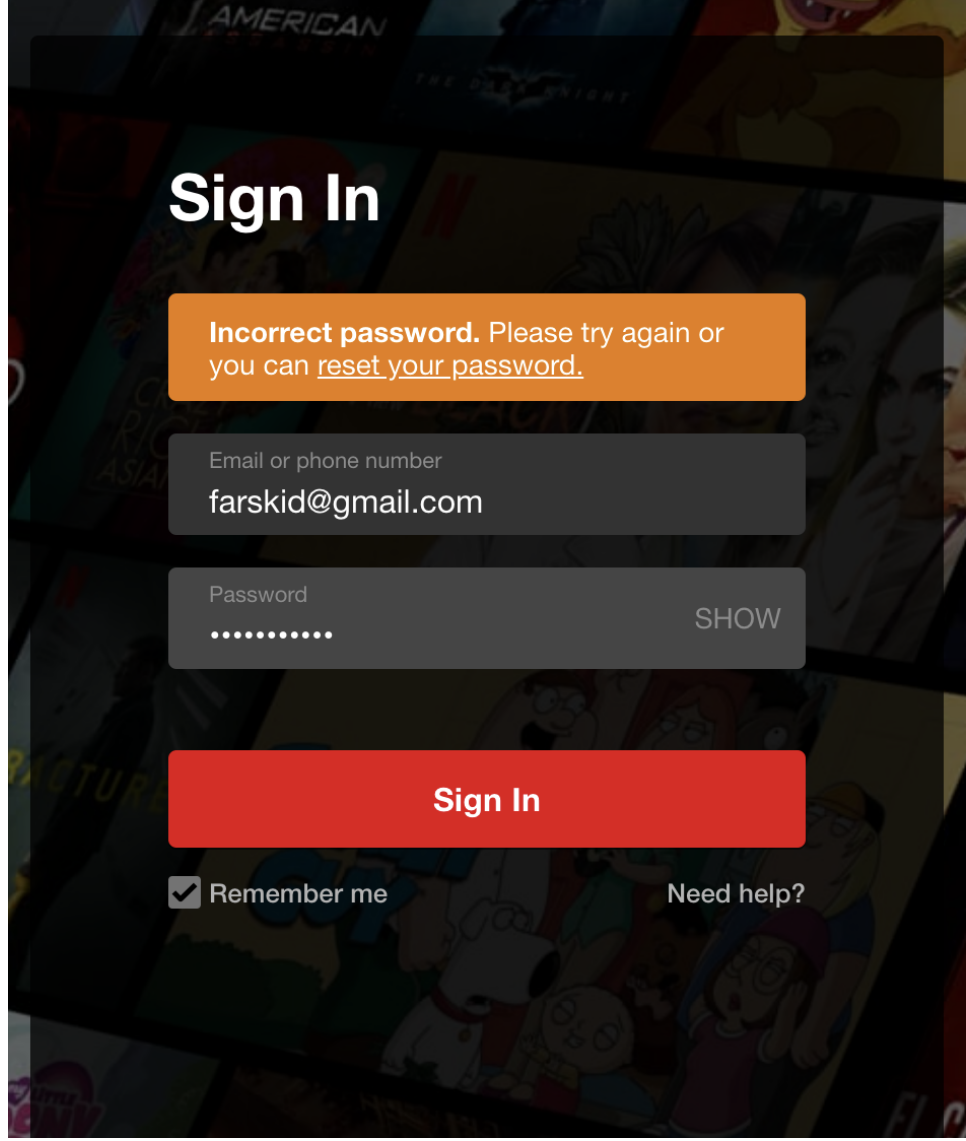


```
{
  ...
  submitting: true,
  submitError: undefined
}
```

```
{
  inputs: {
    email: {
      value: "farskid@gmail.com",
      valid: true,
      isValidated: true,
    }
  },
  submitting: false,
  submitError: "Incorrect
password."
}
```



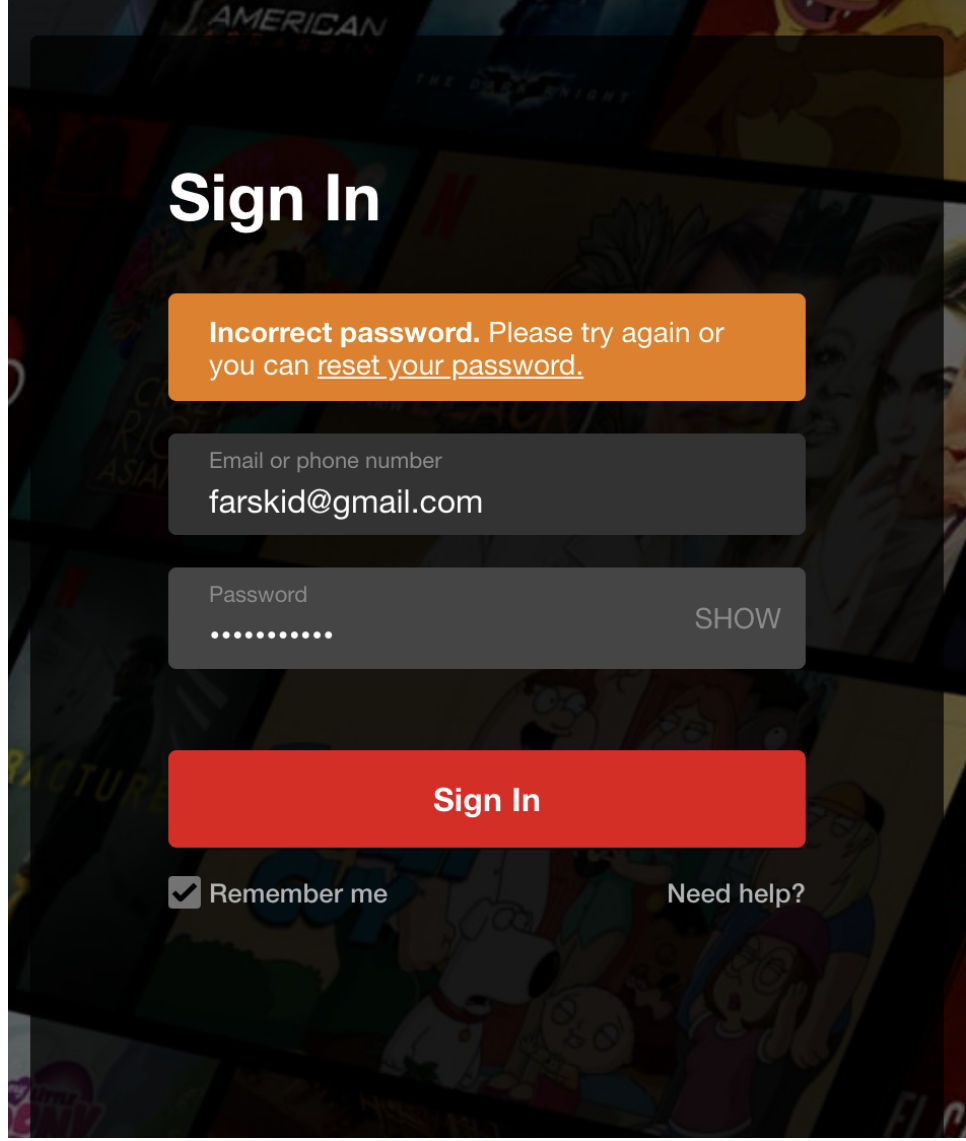
```
{
  submitting: boolean;
  submitError: string | undefined;
}
```



```
{
    submitting: boolean;
    submitError: string | undefined;
}
```

```
submitting: false, error: undefined
```

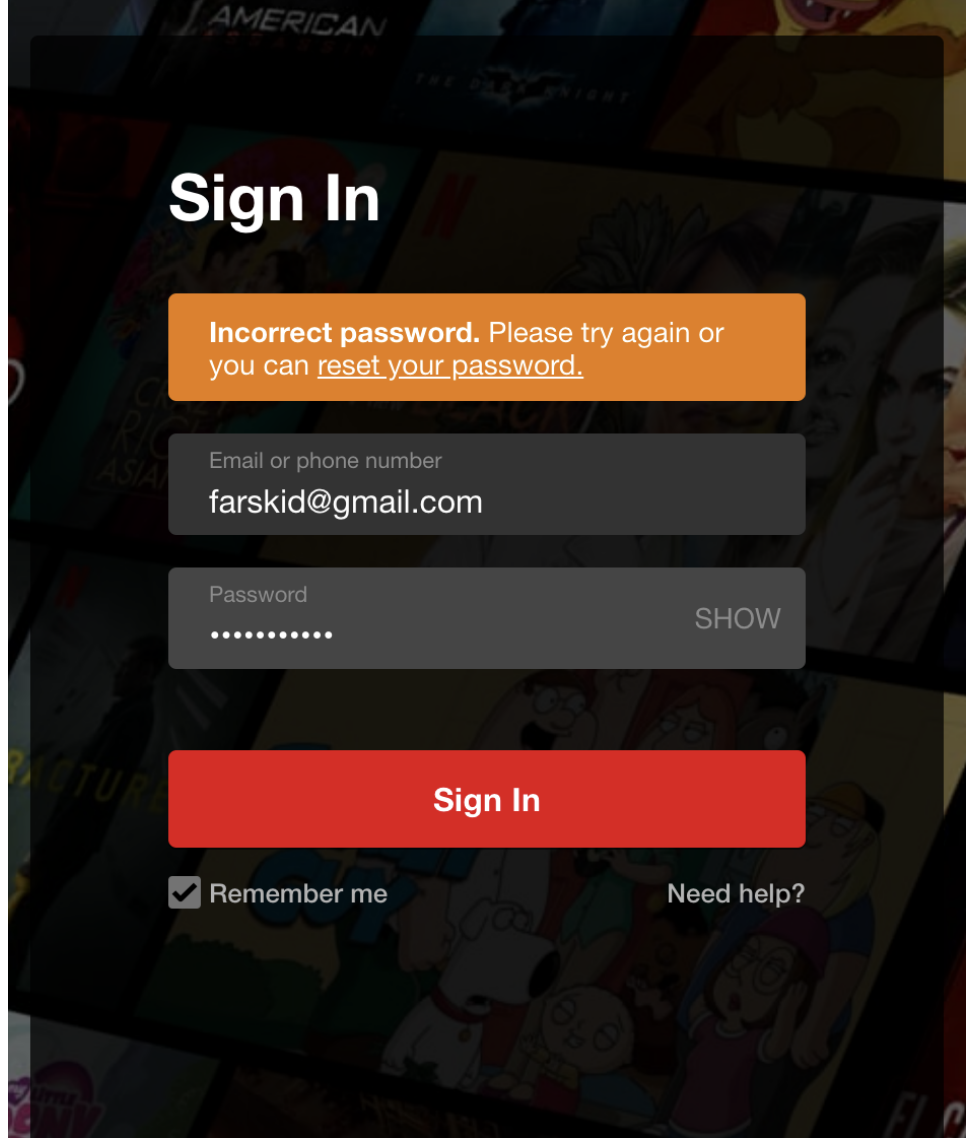
Editing



```
1 {
2     submitting: boolean;
3     submitError: string | undefined;
4 }
```

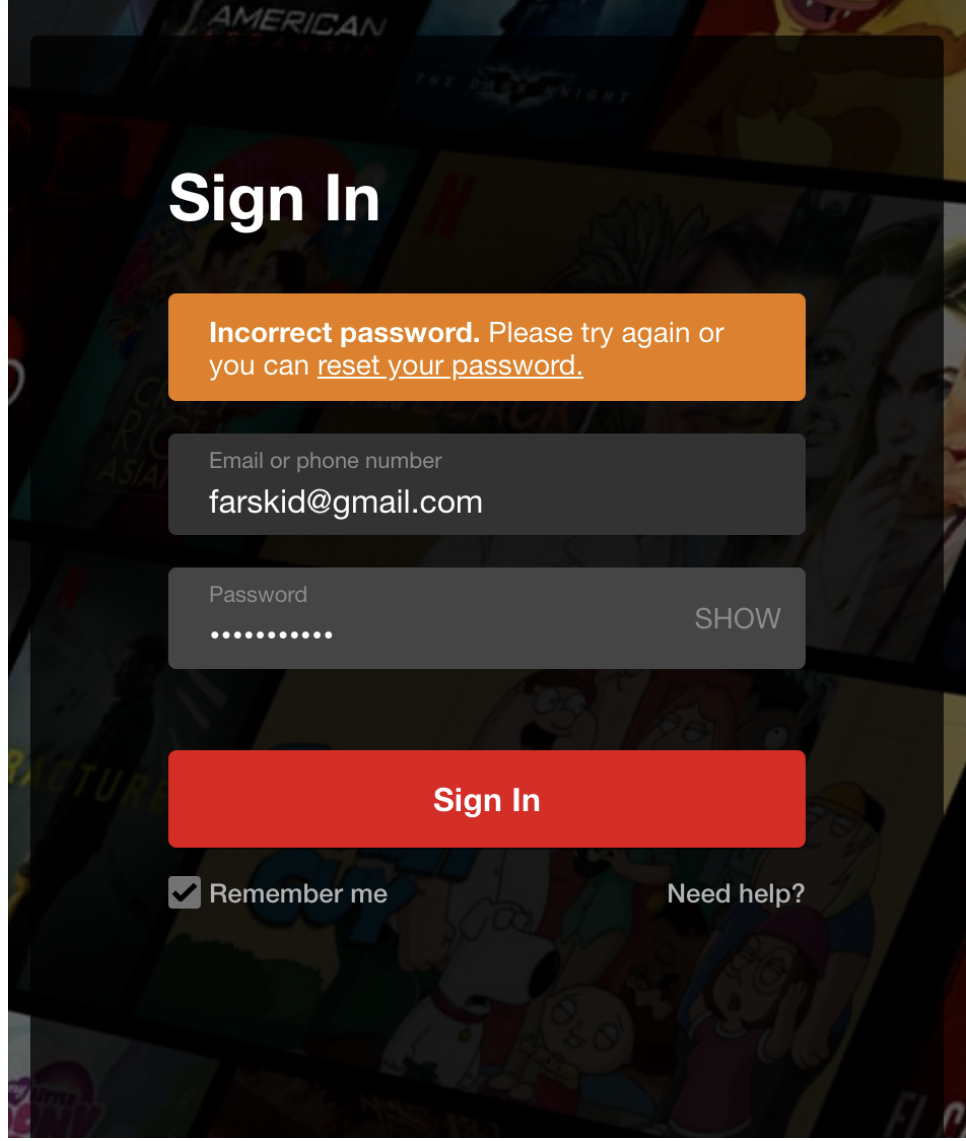
`submitting: false, error: undefined` Editing

`submitting: true, error: undefined` Submitting



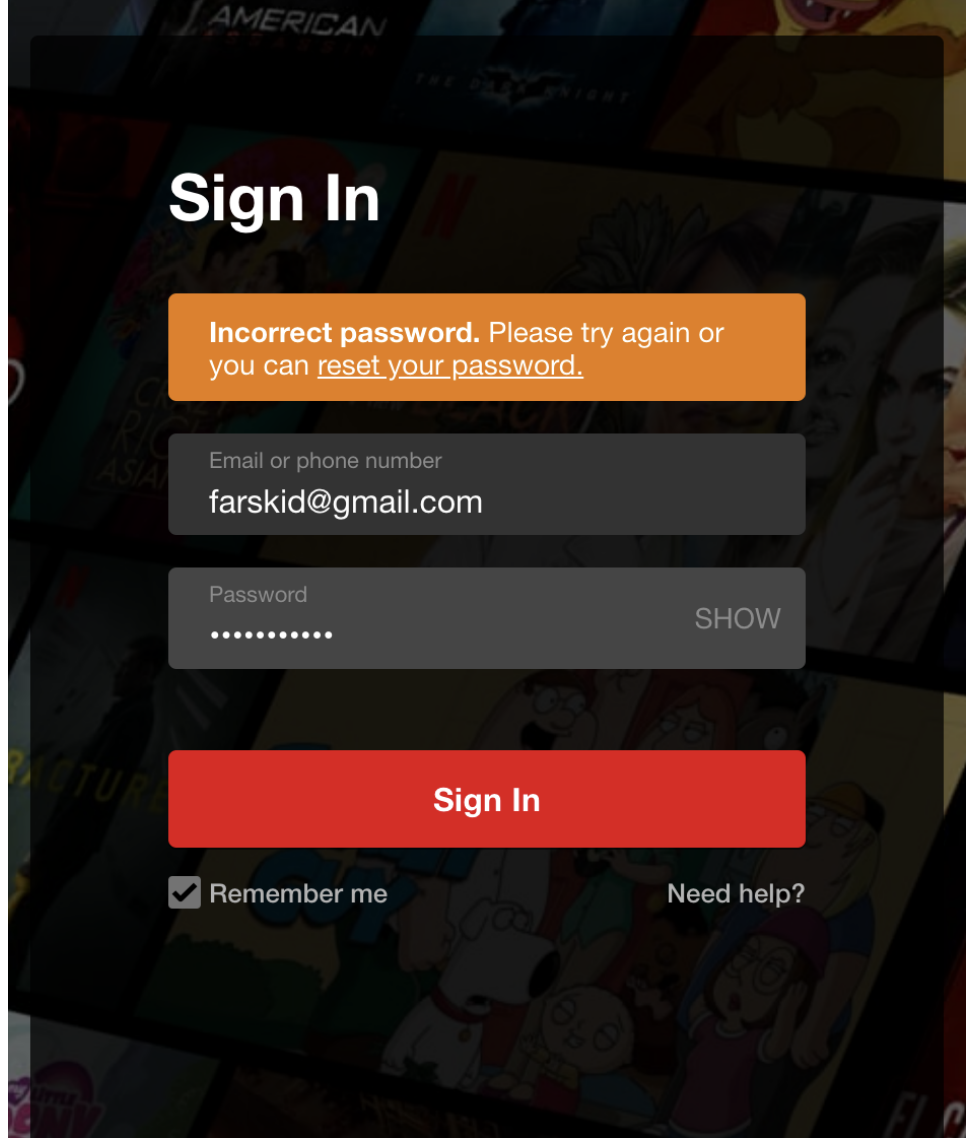
```
1 {
2    submitting: boolean;
3    submitError: string | undefined;
4 }
```

`submitting: false, error: undefined`	Editing
`submitting: true, error: undefined`	Submitting
`submitting: false, error: "Error"`	Failed



```
{
  submitting: boolean;
  submitError: string | undefined;
}
```

`submitting: false, error: undefined`	Editing
`submitting: true, error: undefined`	Submitting
`submitting: false, error: "Error"`	Failed
`submitting: false, error: undefined`	Succeeded



IMPOSSIBLE STATES AGAIN!

```
1 {
2    submitting: boolean;
3    submitError: string | undefined;
4 }
```

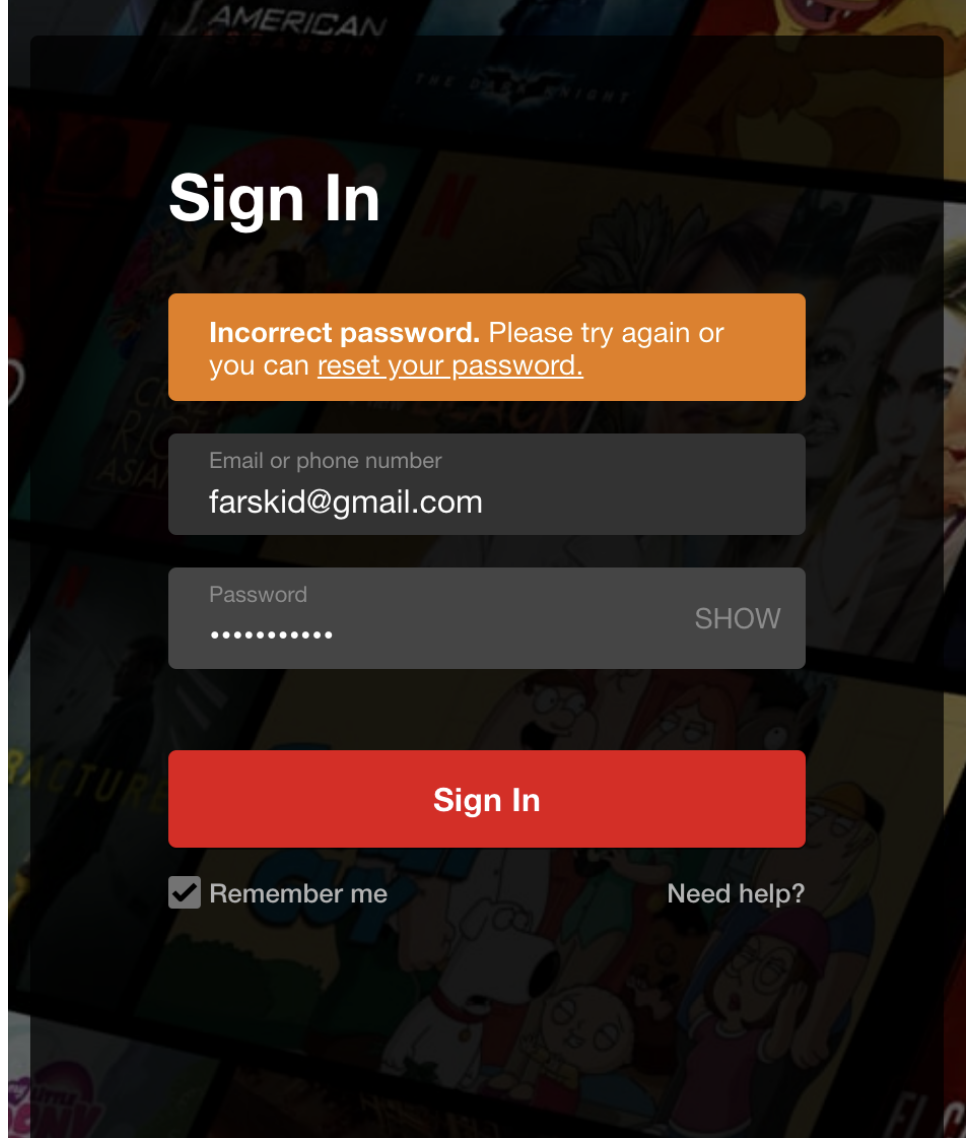
```
submitting: false, error: undefined
submitting: true, error: undefined
submitting: false, error: "Error"
submitting: false, error: undefined
```

Editing

Submitting

Failed

Succeeded



Same state object, different views

IMPOSSIBLE STATES AGAIN!

```
1 {
2     submitting: boolean;
3     submitError: string | undefined;
4     isSuccess: boolean;
5 }
```

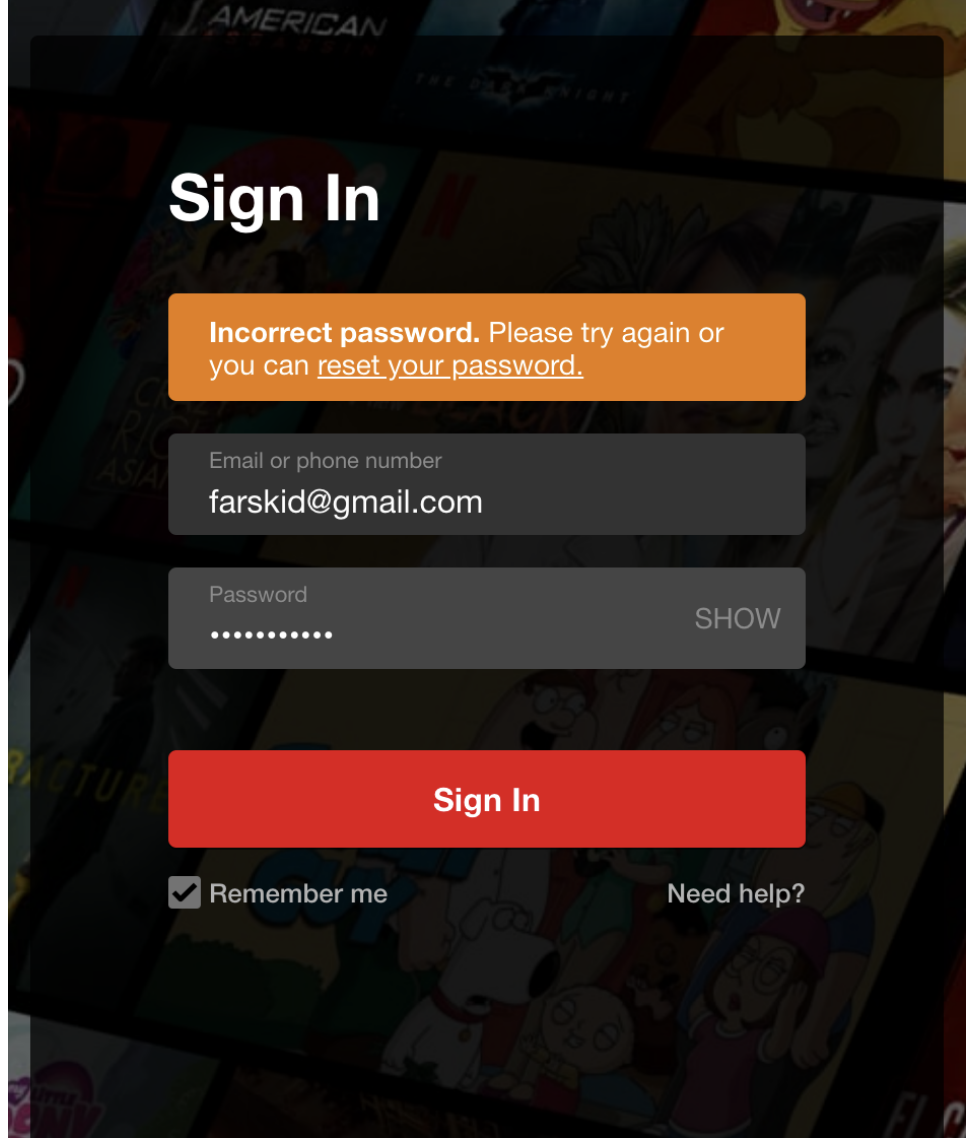
```
submitting: false, error: undefined
submitting: true, error: undefined
submitting: false, error: "Error"
submitting: false, error: undefined
```

Editing

Submitting

Failed

Succeeded



Same state object, different views

ADDING GUARDS

Avoid transitioning to the impossible states

source: https://twitter.com/DavidKPiano/status/1189509124196913158

ADDING GUARDS

Avoid transitioning to the impossible states

Avoid **mutually exclusive** behaviors
from happening simultaneously

source: https://twitter.com/DavidKPiano/status/1189509124196913158

ADDING GUARDS

Avoid transitioning to the impossible states

Avoid **mutually exclusive** behaviors from happening simultaneously

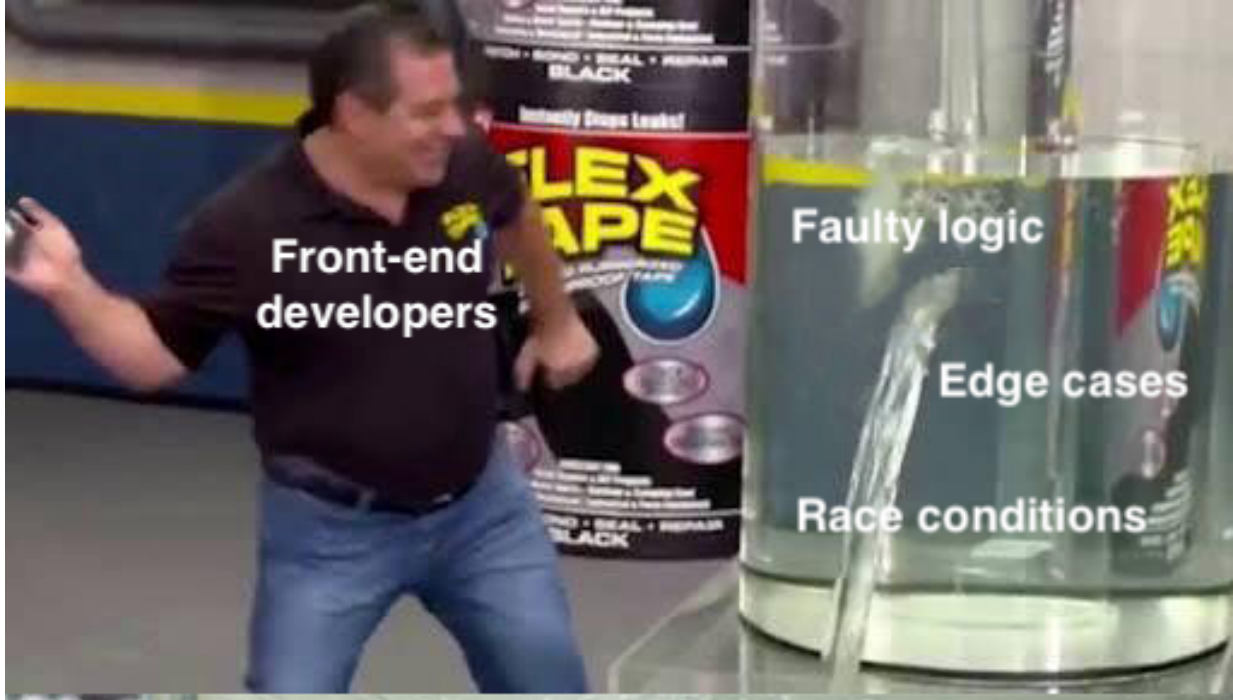
source: https://twitter.com/DavidKPiano/status/1189509124196913158

ADDING GUARDS

Avoid transitioning to the impossible states

Avoid **mutually exclusive** behaviors from happening simultaneously

```
function handleSubmit(e) {
  e.preventDefault();

  const canSubmit = Object.values(state.inputs)
    .map(v => v.valid)
    .every(v => v === true);

  if (!canSubmit) {
    return;
  }
}
```



source: https://twitter.com/DavidKPiano/status/1189509124196913158

Scene #3

THE SOLUTION

THINKING ABOUT STATES EXPLICITLY

THINKING ABOUT STATES EXPLICITLY

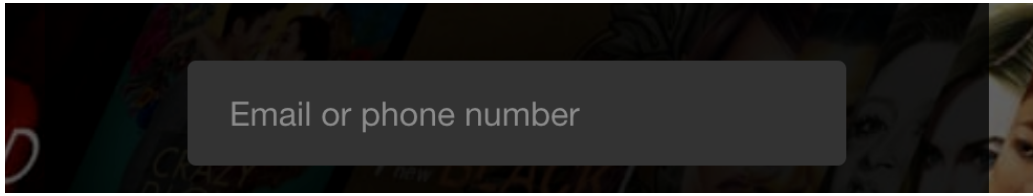
DISCOVERING FINITE STATES

THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

THINK IN STATES EXPLICITLY

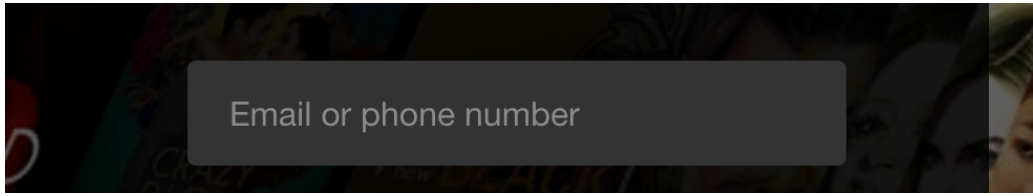
IN THE INPUT EXAMPLE



THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

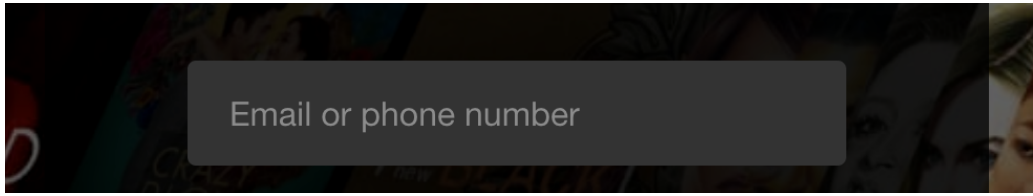
EMPTY `{value: ""}`

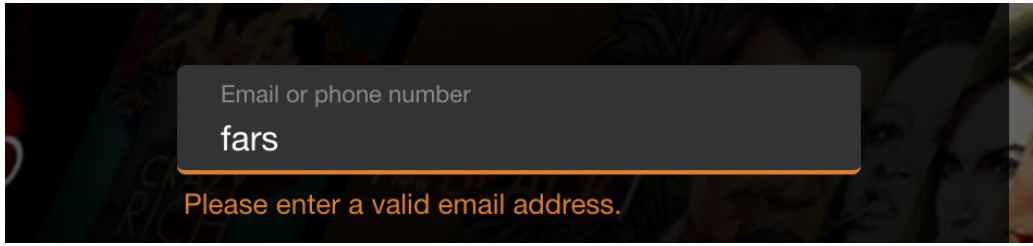


THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

EMPTY `{value: ""}`

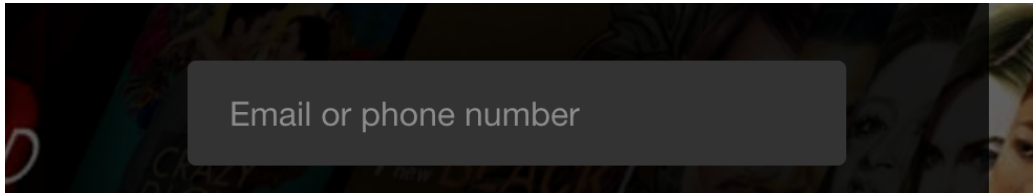




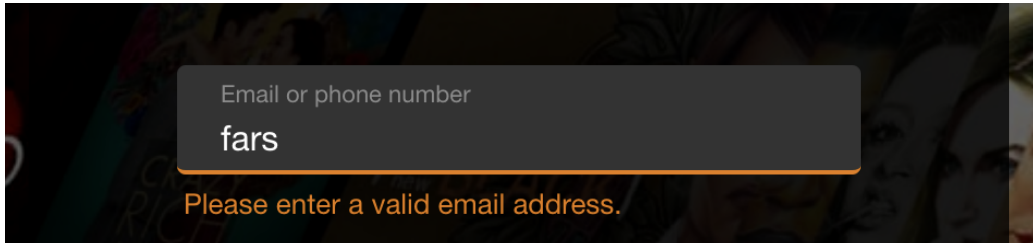
THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

EMPTY `{value: ""}`



INVALID `{value: "fars"}`

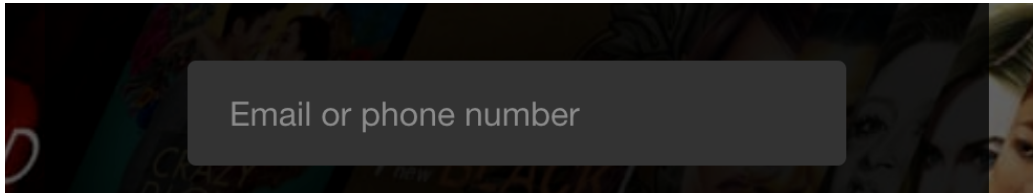


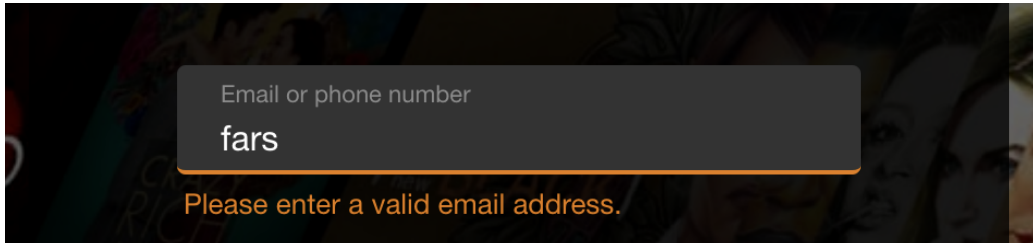
THINK IN STATES EXPLICITLY

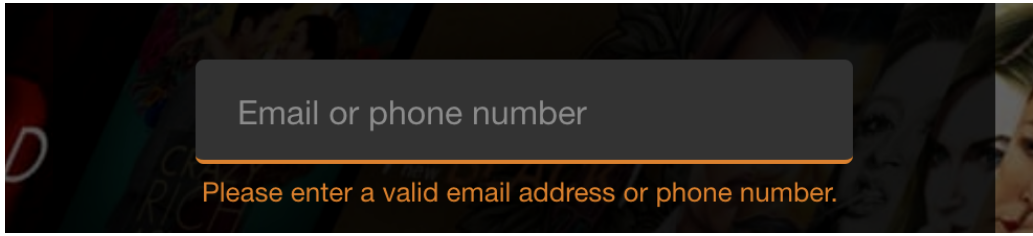
IN THE INPUT EXAMPLE

EMPTY `{value: ""}`

INVALID `{value: "fars"}`



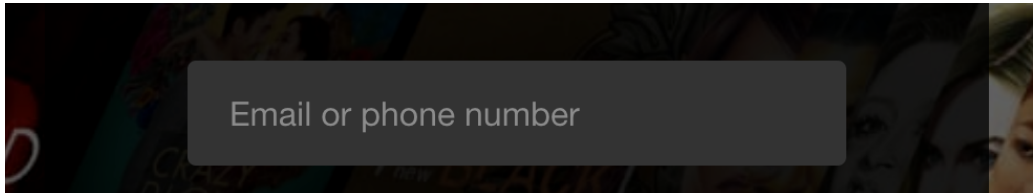




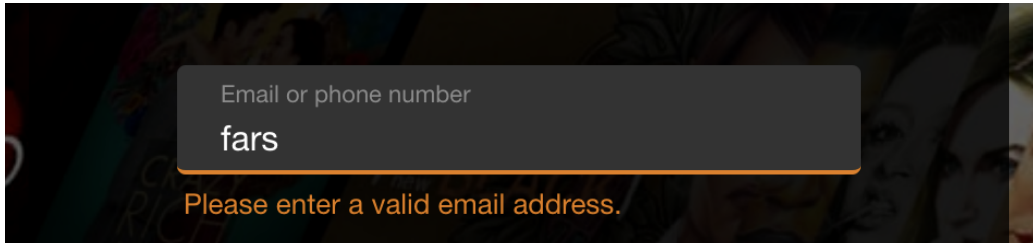
THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

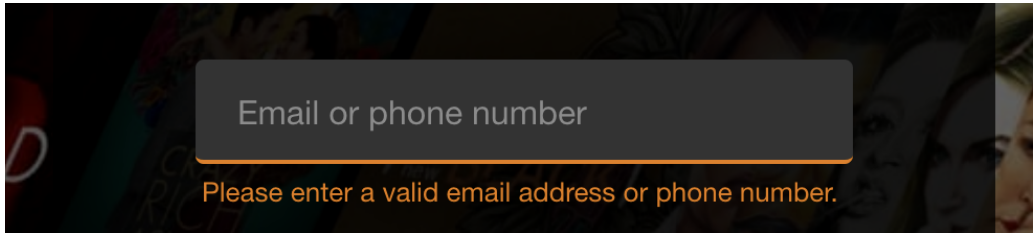
EMPTY `{value: ""}`



INVALID `{value: "fars"}`



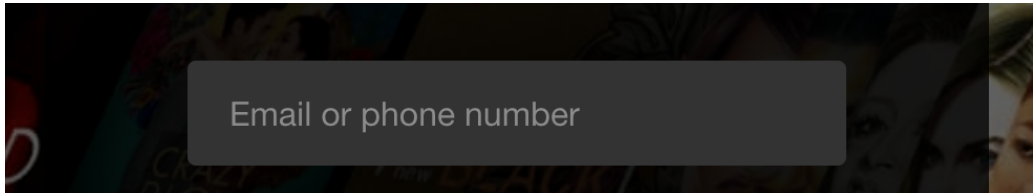
INVALID `{value: ""}`



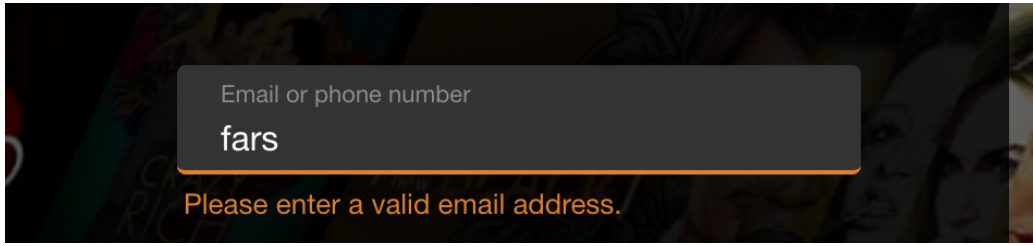
THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

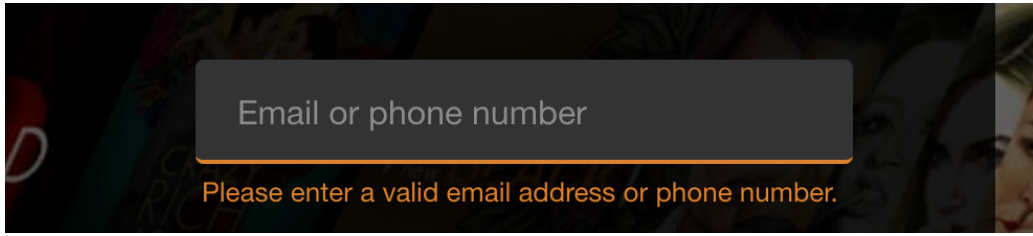
EMPTY `{value: ""}`

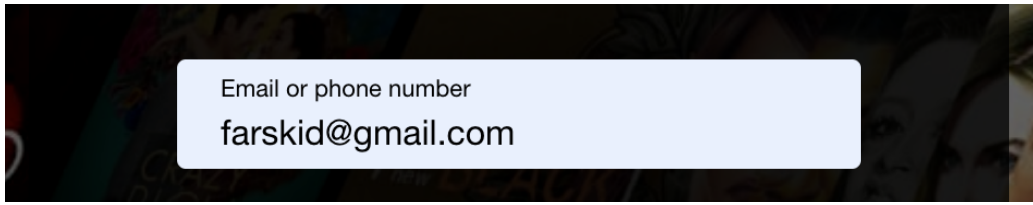


INVALID `{value: "fars"}`



INVALID `{value: ""}`

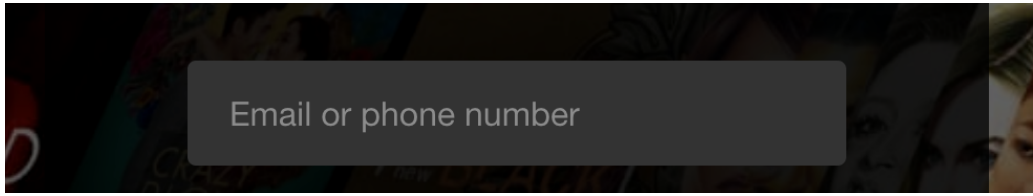




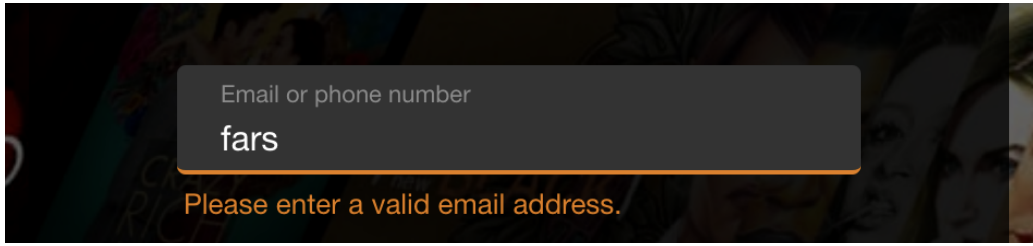
THINK IN STATES EXPLICITLY

IN THE INPUT EXAMPLE

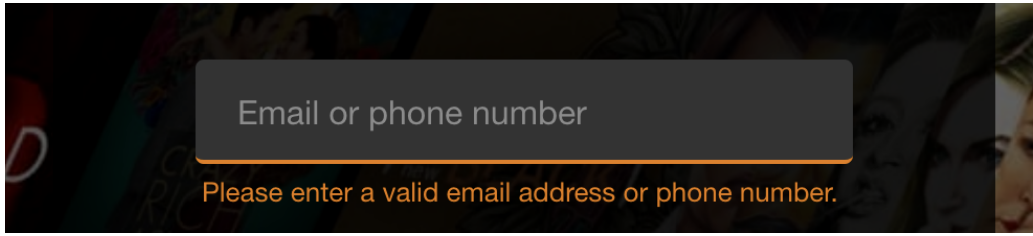
EMPTY `{value: ""}`



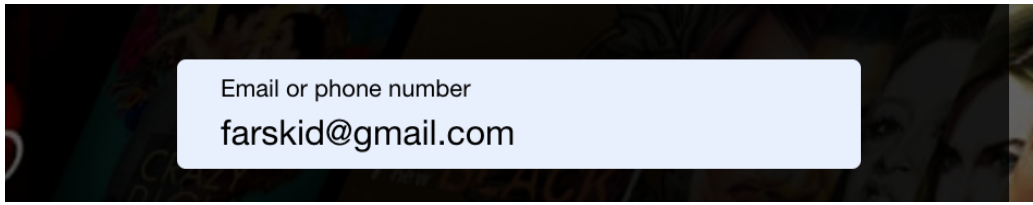
INVALID `{value: "fars"}`



INVALID `{value: ""}`



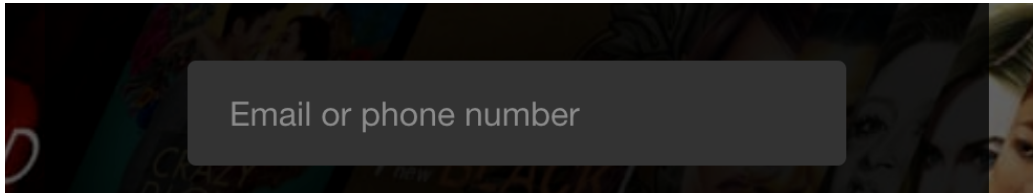
VALID `{value: "farskid@gmail.com"}`

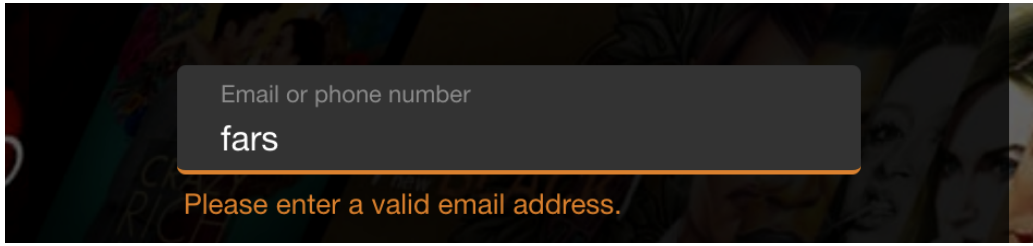


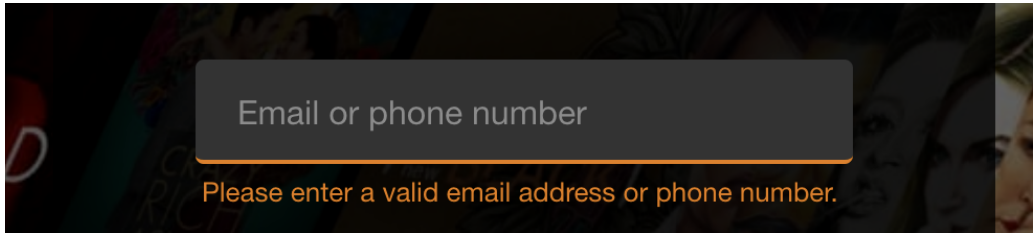
THINK IN STATES EXPLICITLY

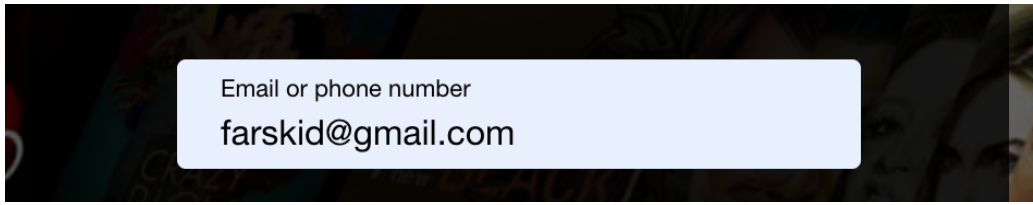
IN THE INPUT EXAMPLE

FINITE STATE	INFINITE STATE
EMPTY	`{value: ""}`
INVALID	`{value: "fars"}`
INVALID	`{value: ""}`
VALID	`{value: "farskid@gmail.com"}`







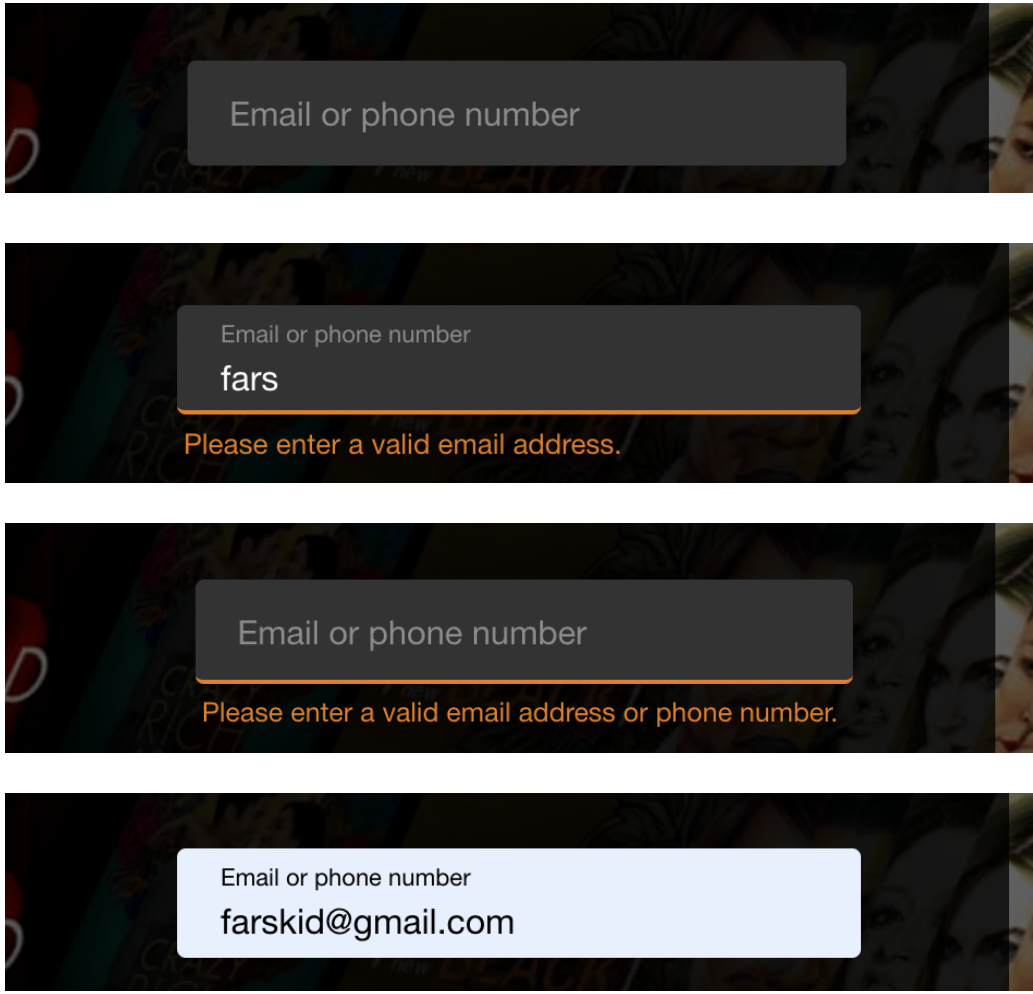


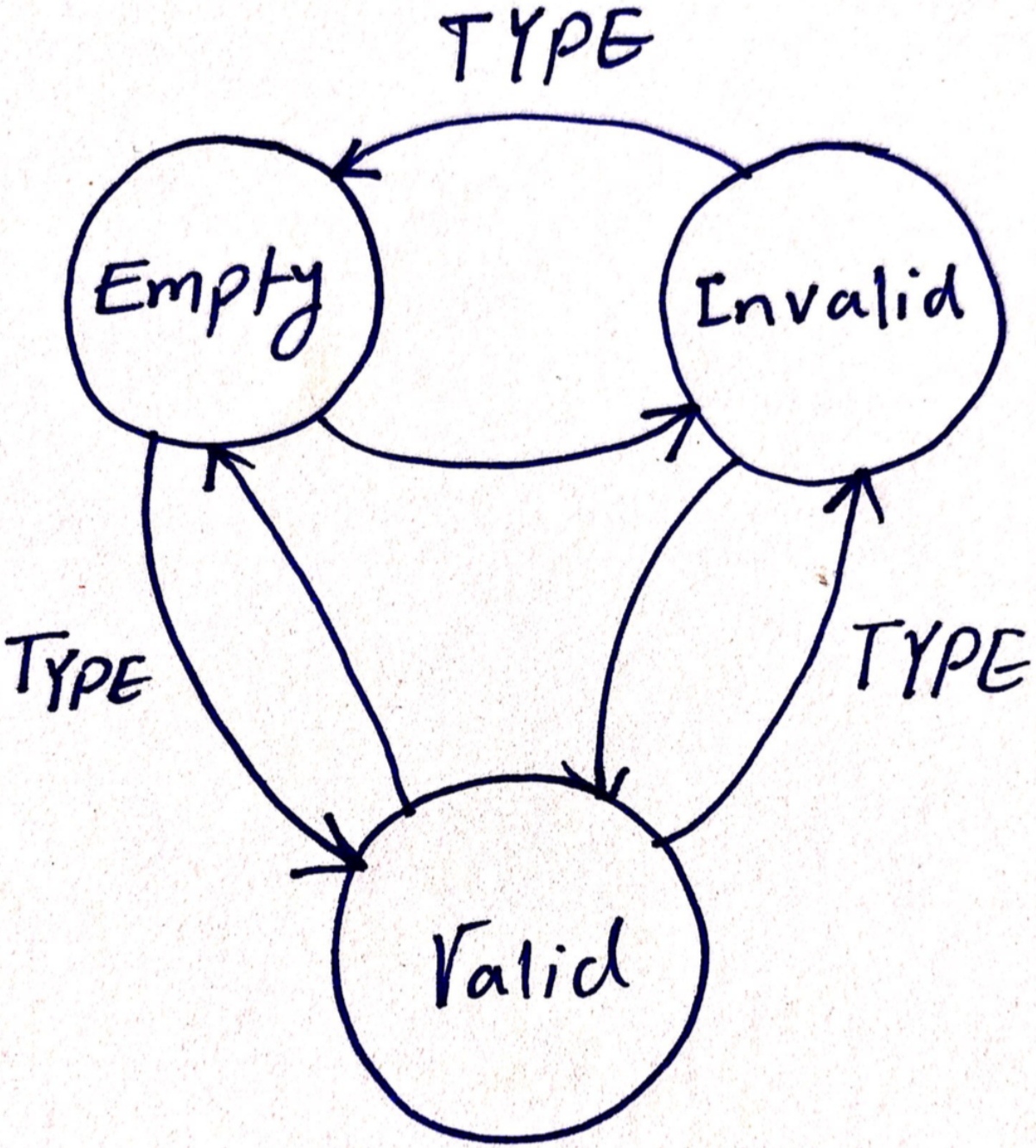
THINK IN STATES EXPLICITLY

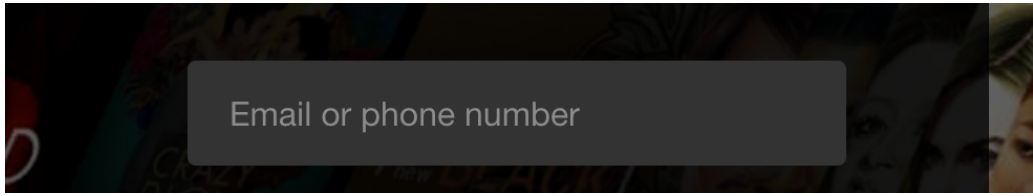
IN THE INPUT EXAMPLE

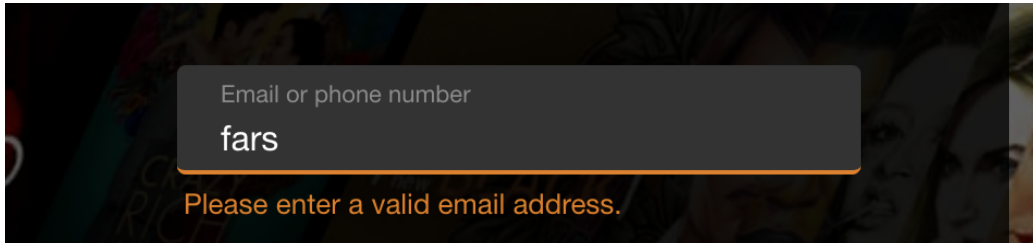
FINITE STATE	INFINITE STATE
EMPTY	`{value: ""}`
INVALID	`{value: "fars"}`
INVALID	`{value: ""}`
VALID	`{value: "farskid@gmail.com"}`

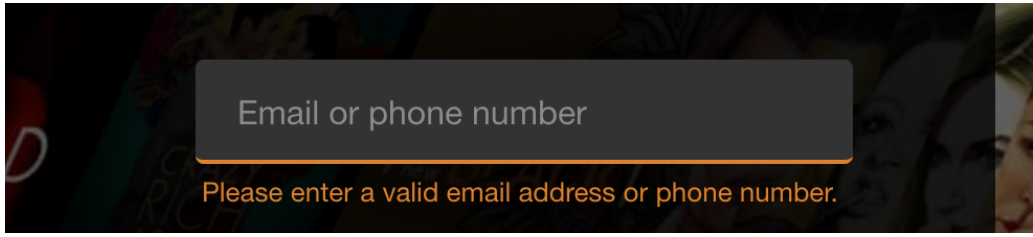
Type

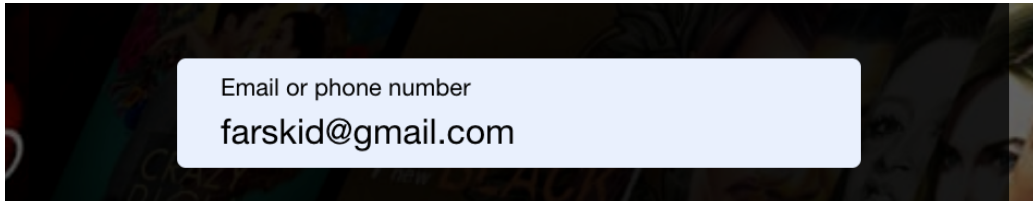



TYPE
Empty
Invalid
TYPE
TYPE
Valid

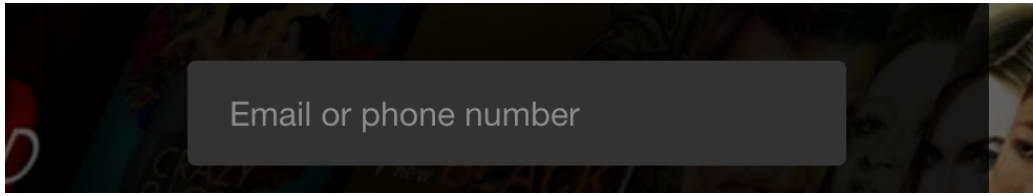

Email or phone number

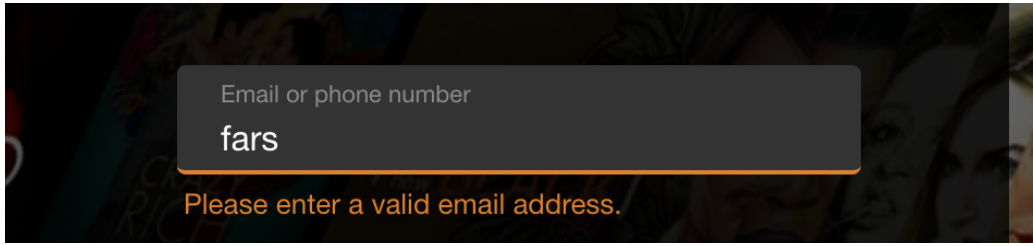

Email or phone number
fars
Please enter a valid email address.

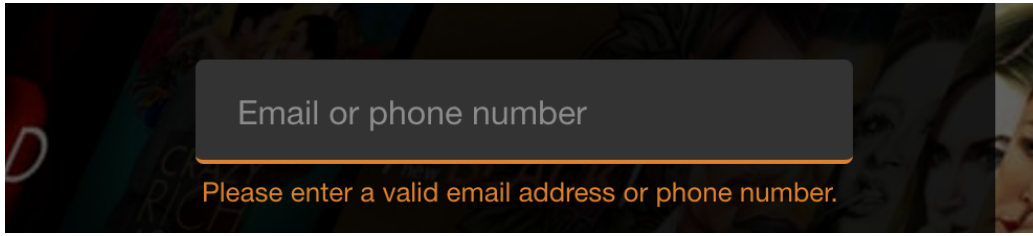

Email or phone number
Please enter a valid email address or phone number.

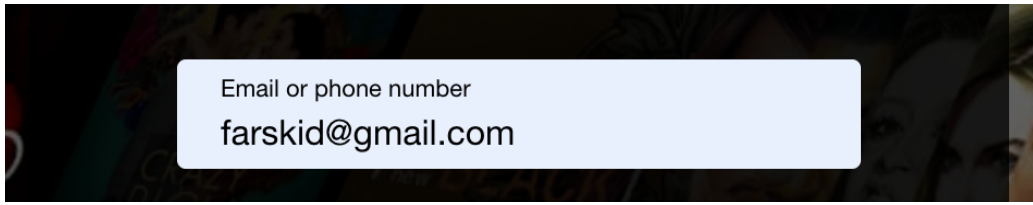

Email or phone number
farskid@gmail.com

```
InputState = "Empty" | "Invalid" | "Valid"
```



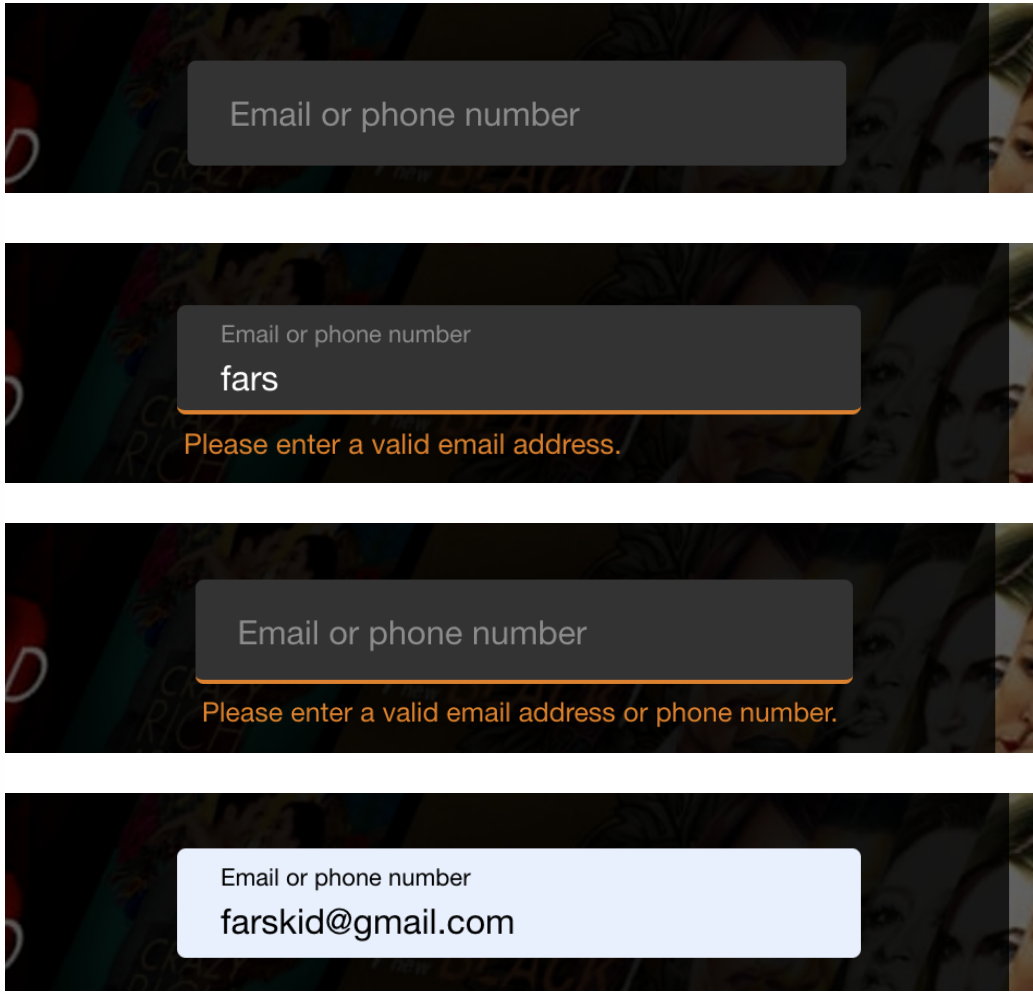






```
InputState = "Empty" | "Invalid" | "Valid"
```

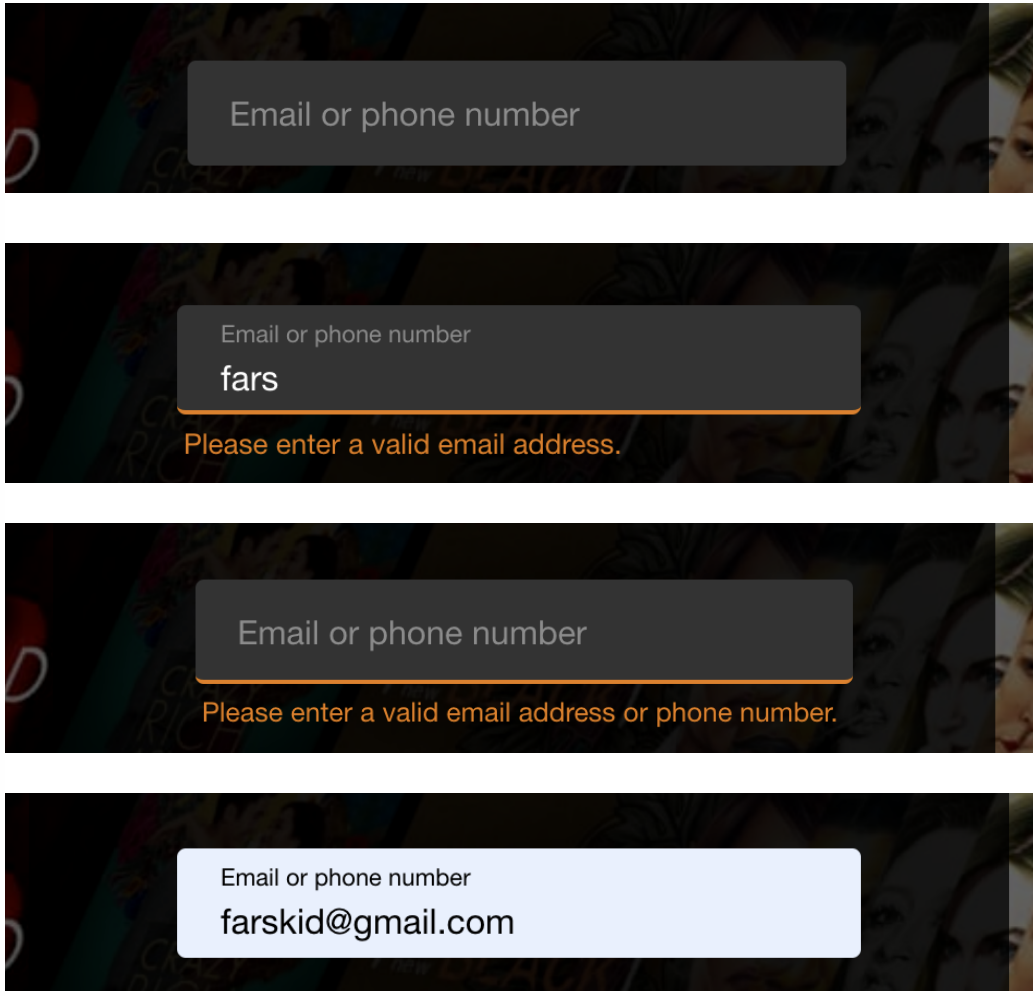
```
function transition(state, event) {
  switch(state) {
    case "Empty":
    case "Invalid":
    case "Valid":
      switch(event.type) {
        case "TYPE":
          return validateEmail(event.payload)
            ? "Valid" : "Invalid"
        default:
          return state;
      }
    default:
      throw Error("Impossible state")
  }
}
```



```
InputState = "Empty" | "Invalid" | "Valid"
```

```
onChange(e) {
  setInputContext(e.target.value)
  setInputState(transition(inputState, "TYPE"))
}
```

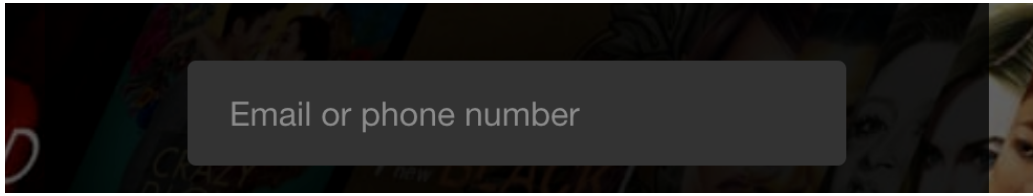
```
function transition(state, event) {
  switch(state) {
    case "Empty":
    case "Invalid":
    case "Valid":
      switch(event.type) {
        case "TYPE":
          return validateEmail(event.payload)
            ? "Valid" : "Invalid"
        default:
          return state;
      }
    default:
      throw Error("Impossible state")
  }
}
```

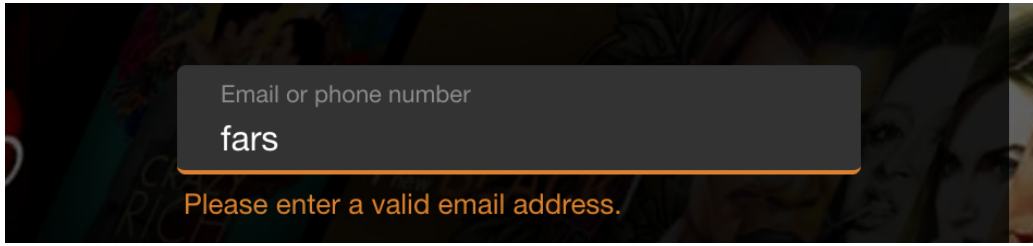


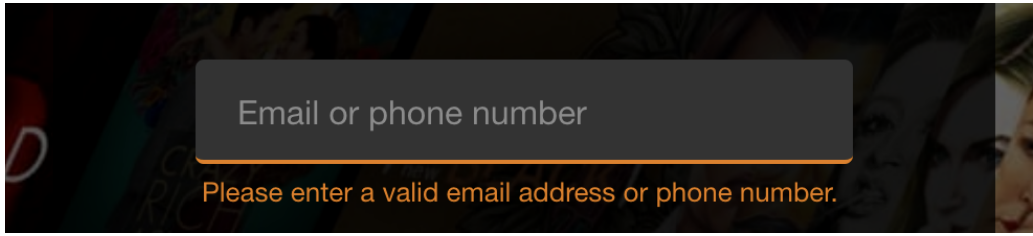
```
InputState = "Empty" | "Invalid" | "Valid"
```

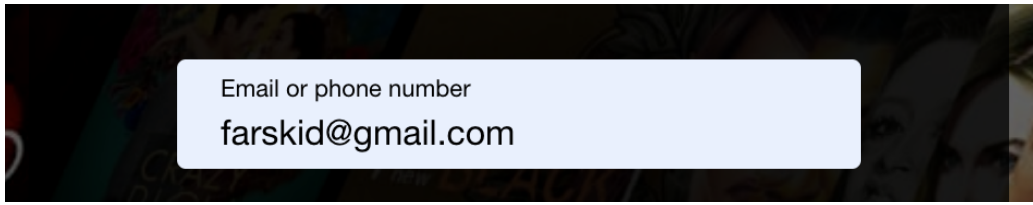
```
onChange(e) {
  setInputContext(e.target.value)
  setInputState(transition(inputState, "TYPE"))
}
```

```
function transition(state, event) {
  switch(state) {
    case "Empty":
    case "Invalid":
    case "Valid":
      switch(event.type) {
        case "TYPE":
          return validateEmail(event.payload)
            ? "Valid" : "Invalid"
        default:
          return state;
      }
    default:
      throw Error("Impossible state")
  }
}
```

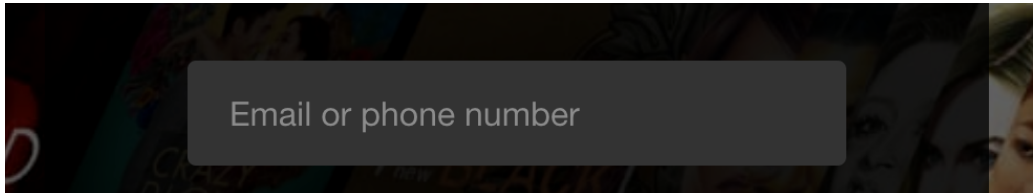


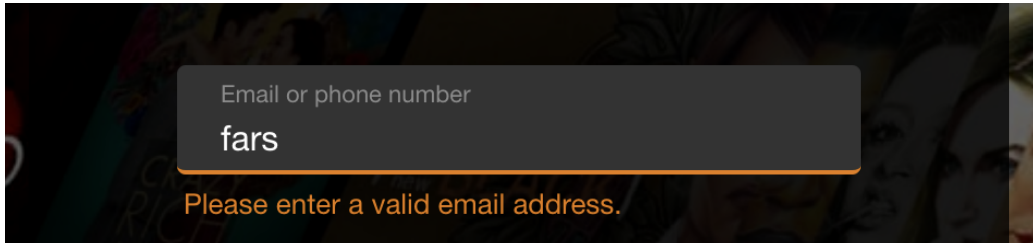


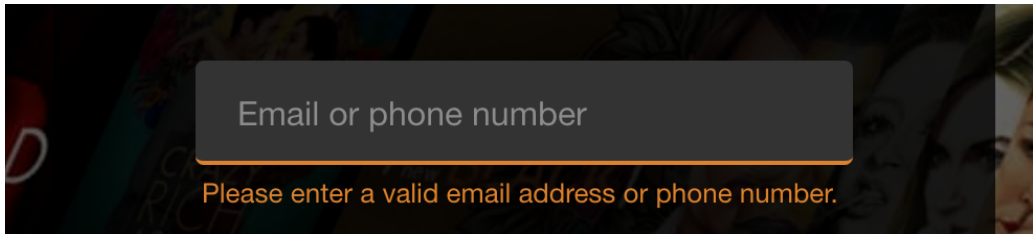


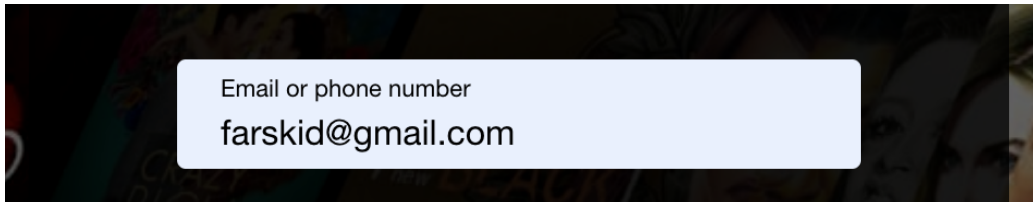


CONDITIONAL RENDERING BASED ON FINITE STATE



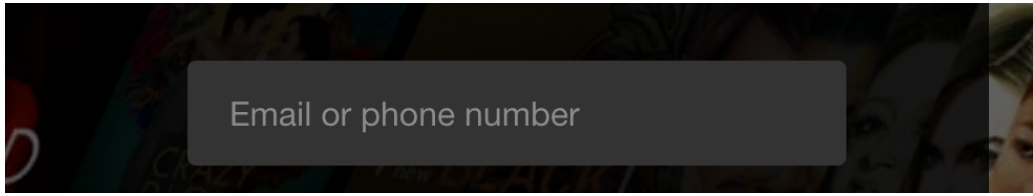


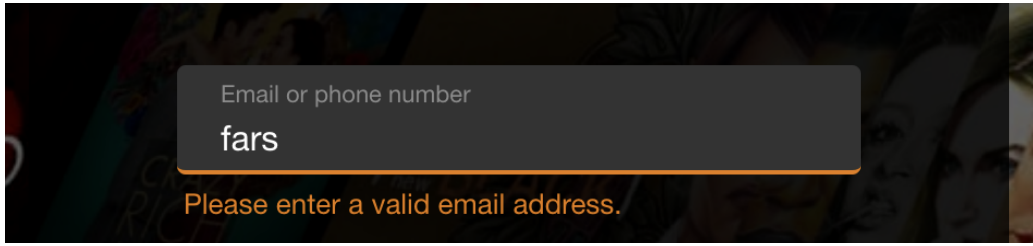


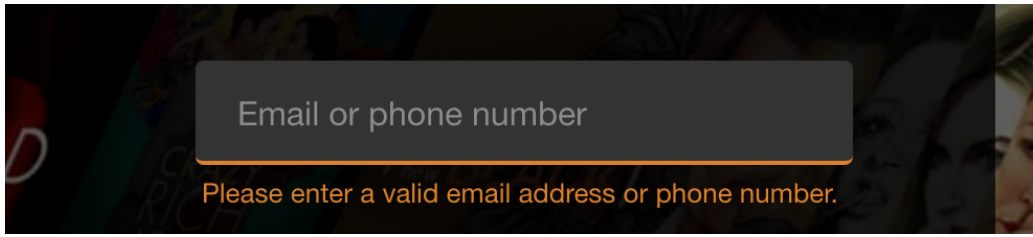


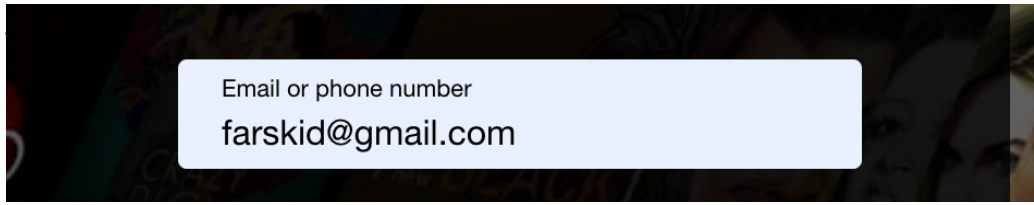
CONDITIONAL RENDERING BASED ON FINITE STATE

```
<input
  type="text"
  onChange={e => {
    setInputContext(e.target.value);
    setInputState(
      transition(inputState, "TYPE")
    );
  }}
/>;

{
  inputState === "Invalid" &&
  <p>Enter a valid email address</p>
}
```



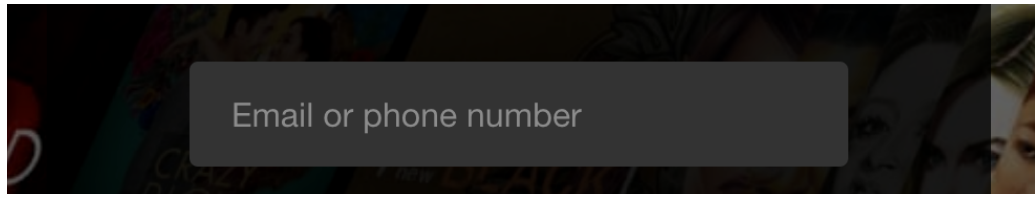


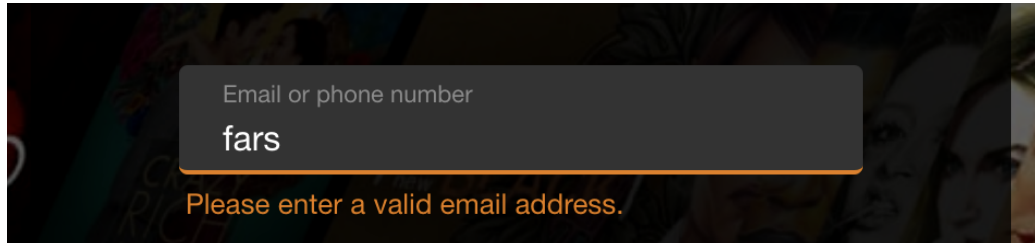




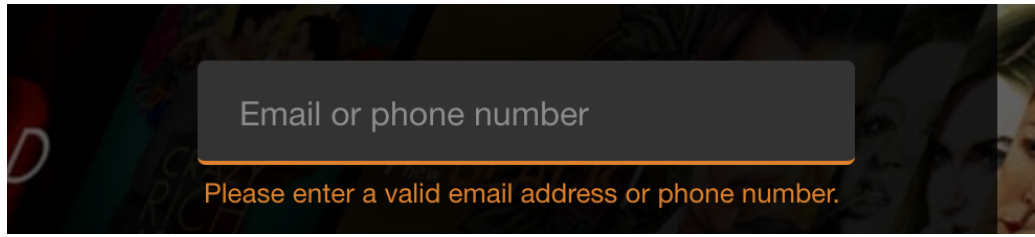
CONDITIONAL RENDERING BASED ON FINITE STATE

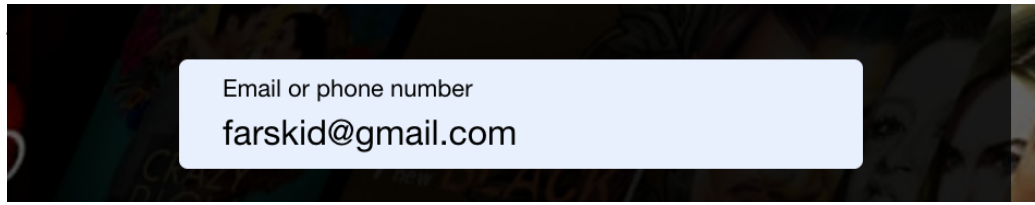
```
{
  !valid && isValidated &&
  <p>Enter a valid email address</p>;
}
```






```
{
  inputState === "Invalid" &&
  <p>Enter a valid email address</p>
}
```





THINK IN STATES EXPLICITLY IN THE FORM EXAMPLE

FINITE STATE

```
type State =
  | {
      formState:
        | "Valid" | "Submitting"
        | "SubmitSucceeded" | "SubmitFailed"
      InputStates: {
        email: "Valid"
        password: "Valid"
      };
    }
  | {
      formState: "Invalid";
      inputStates: {
        email: "Empty" | "Invalid" | "Valid"
        password: "Empty" | "Invalid" | "Valid"
      };
    };
```

THINK IN STATES EXPLICITLY IN THE FORM EXAMPLE

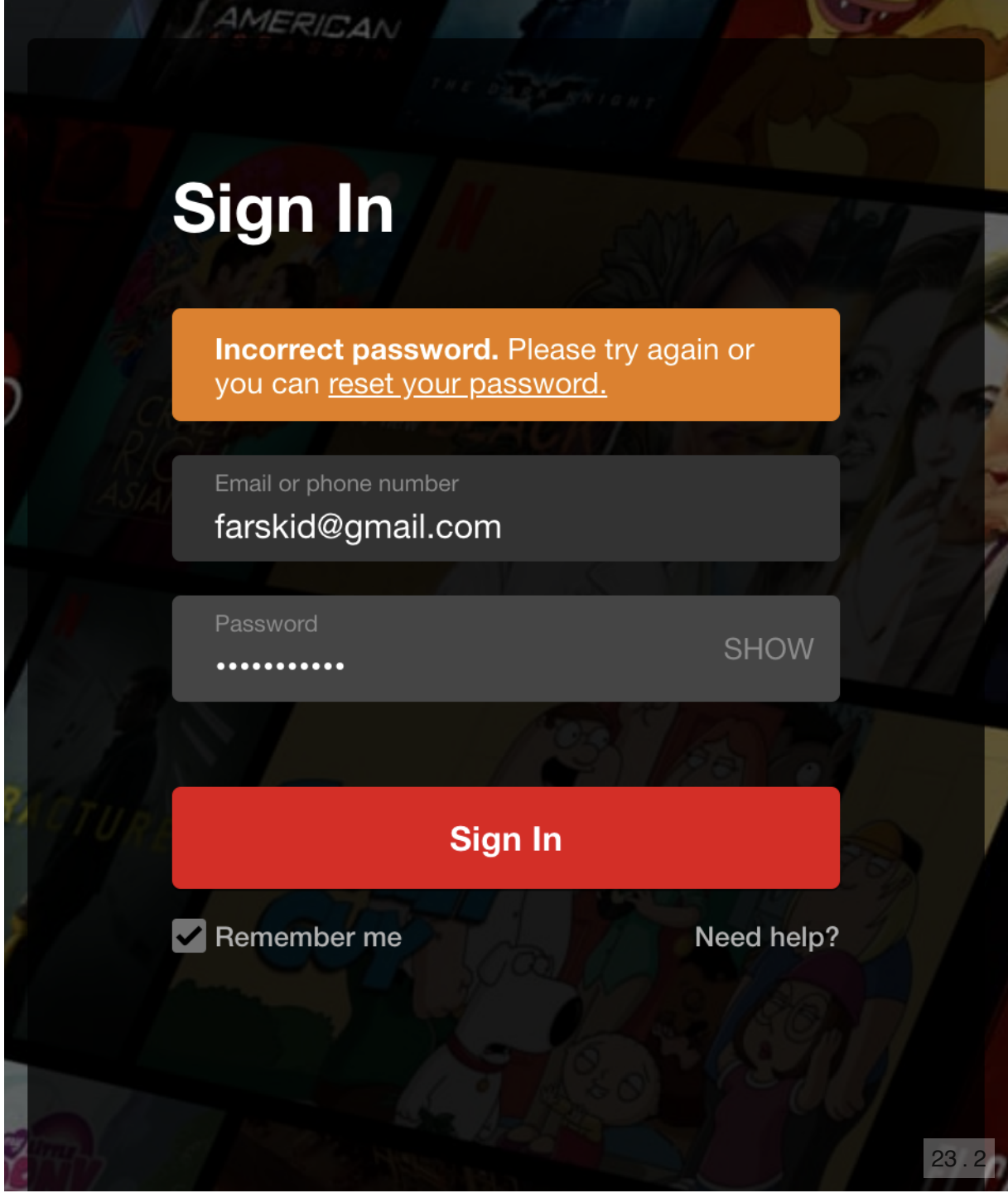
FINITE STATE

```
type State =
  | {
      formState:
        | "Valid" | "Submitting"
        | "SubmitSucceeded" | "SubmitFailed"
      InputStates: {
        email: "Valid"
        password: "Valid"
      };
    }
  | {
      formState: "Invalid";
      inputStates: {
        email: "Empty" | "Invalid" | "Valid"
        password: "Empty" | "Invalid" | "Valid"
      };
    };
```

INFINITE STATE

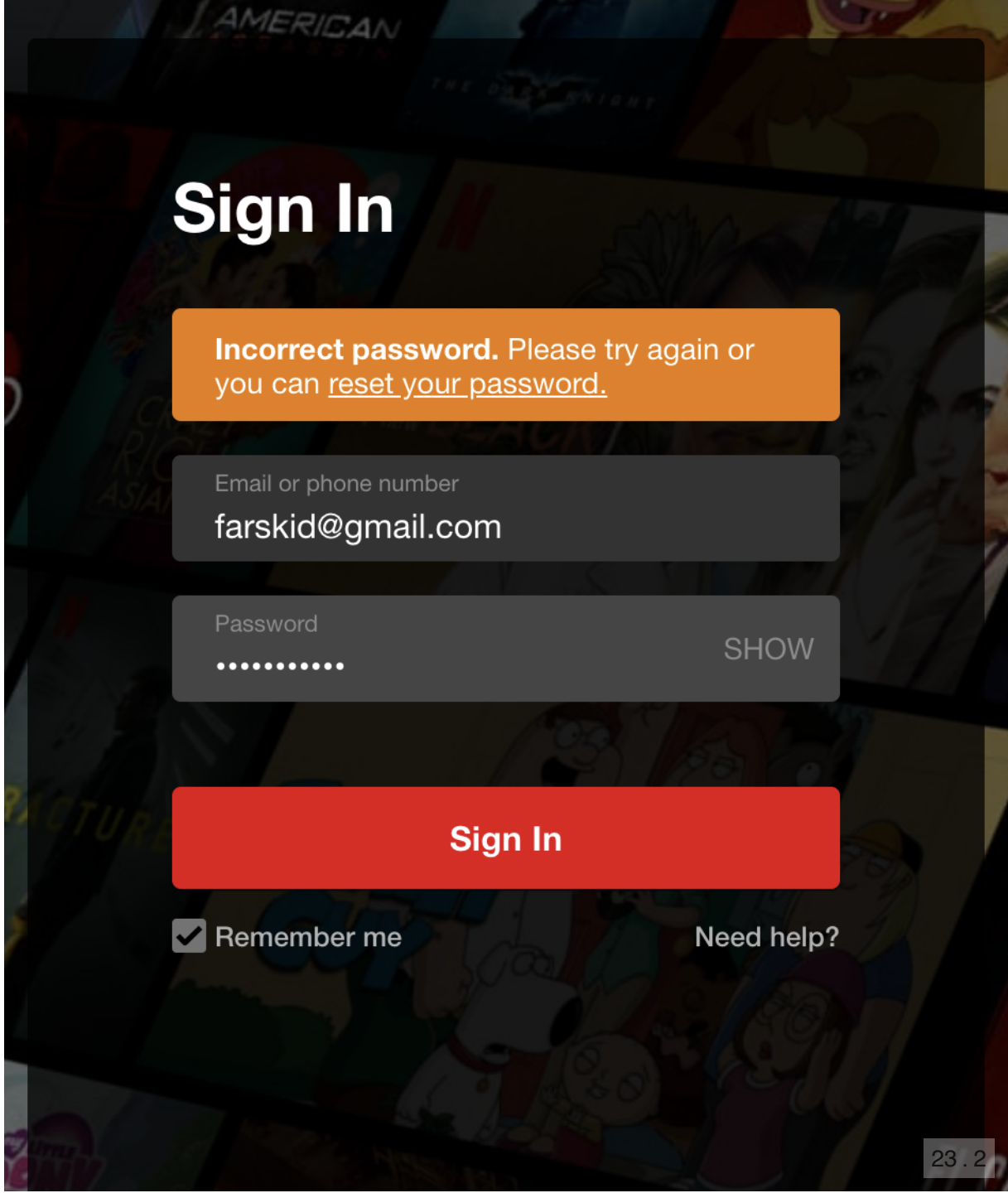
```
type FormContext = {
  email: string
  password: string
  submitError:
    | Error
    | undefined
}
```

CONDITIONAL RENDERING BASED ON FINITE STATE



CONDITIONAL RENDERING BASED ON FINITE STATE

```
<button
  type="submit"
  disabled={
    state.formState === "Submitting"
}>
  Sign In
</button>

{
  formState.state === "SubmitFailed"
&&
  <p>{ formContext.submitError }</p>;
}
```



ELEMENTS HAVE FINITE STATES

TOOLTIP/MODAL/DROPDOWN

```
type State = "Opened" | "Closed"
```

ELEMENTS HAVE FINITE STATES

TOOLTIP/MODAL/DROPDOWN

```
type State = "Opened" | "Closed"
```

BUTTON

```
type State =
  | "Normal"
  | "Hovered"
  | "Active.Idle"
  | "Active.Focused"
  | "Disabled"
```

ELEMENTS HAVE FINITE STATES

TOOLTIP/MODAL/DROPDOWN

```
type State = "Opened" | "Closed"
```

BUTTON

```
type State =
  | "Normal"
  | "Hovered"
  | "Active.Idle"
  | "Active.Focused"
  | "Disabled"
```

RANGE INPUT / PROGRESS

```
type State =
  | "Min"
  | "Mid"
  | "Max"

const context = {
  min: number,
  max: number,
  value: number
}
```

TIME BASED SIDE EFFECTS HAVE FINITE STATES

```
type Promise =
| Pending
| Settled.Fulfilled
| Settled.Rejected

const Context = {
  resolvedValue?: any;
  rejectedError?: any
}
```

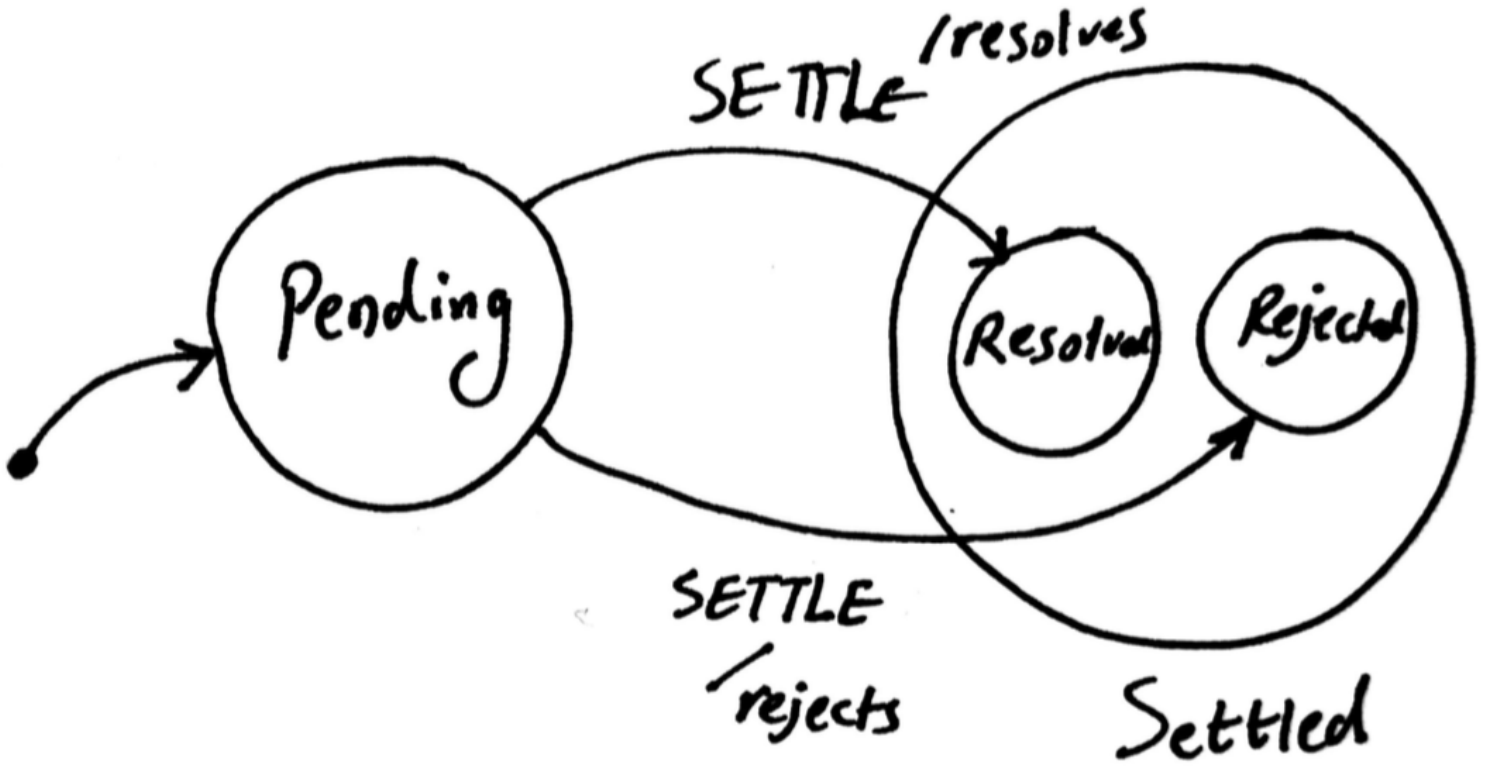


Table 59 — Internal Slots of Promise Instances

Internal Slot	Description
[[PromiseState]]	A String value that governs how a promise will react to incoming calls to its **then** method. The possible values are: **"pending"**, **"fulfilled"**, and **"rejected"**.

source: https://www.ecma-international.org/ecma-262/6.0/#sec-properties-of-promise-instances

Scene #4

SCALING

MAKING MODELING PRACTICAL

MAKING MODELING PRACTICAL

DEFINE STATES EXPLICITLY

MAKING MODELING PRACTICAL

DEFINE STATES EXPLICITLY

SEPARATE FINITE AND INFINITE STATES

MAKING MODELING PRACTICAL

DEFINE STATES EXPLICITLY

SEPARATE FINITE AND INFINITE STATES

ABSTRACT WITH FOCUS ON LOGIC

MAKING MODELING PRACTICAL

DEFINE STATES EXPLICITLY

SEPARATE FINITE AND INFINITE STATES

ABSTRACT WITH FOCUS ON LOGIC

REDUCE GUARDS AND CYCLOMATIC COMPLEXITY

MAKING MODELING PRACTICAL

DEFINE STATES EXPLICITLY

SEPARATE FINITE AND INFINITE STATES

ABSTRACT WITH FOCUS ON LOGIC

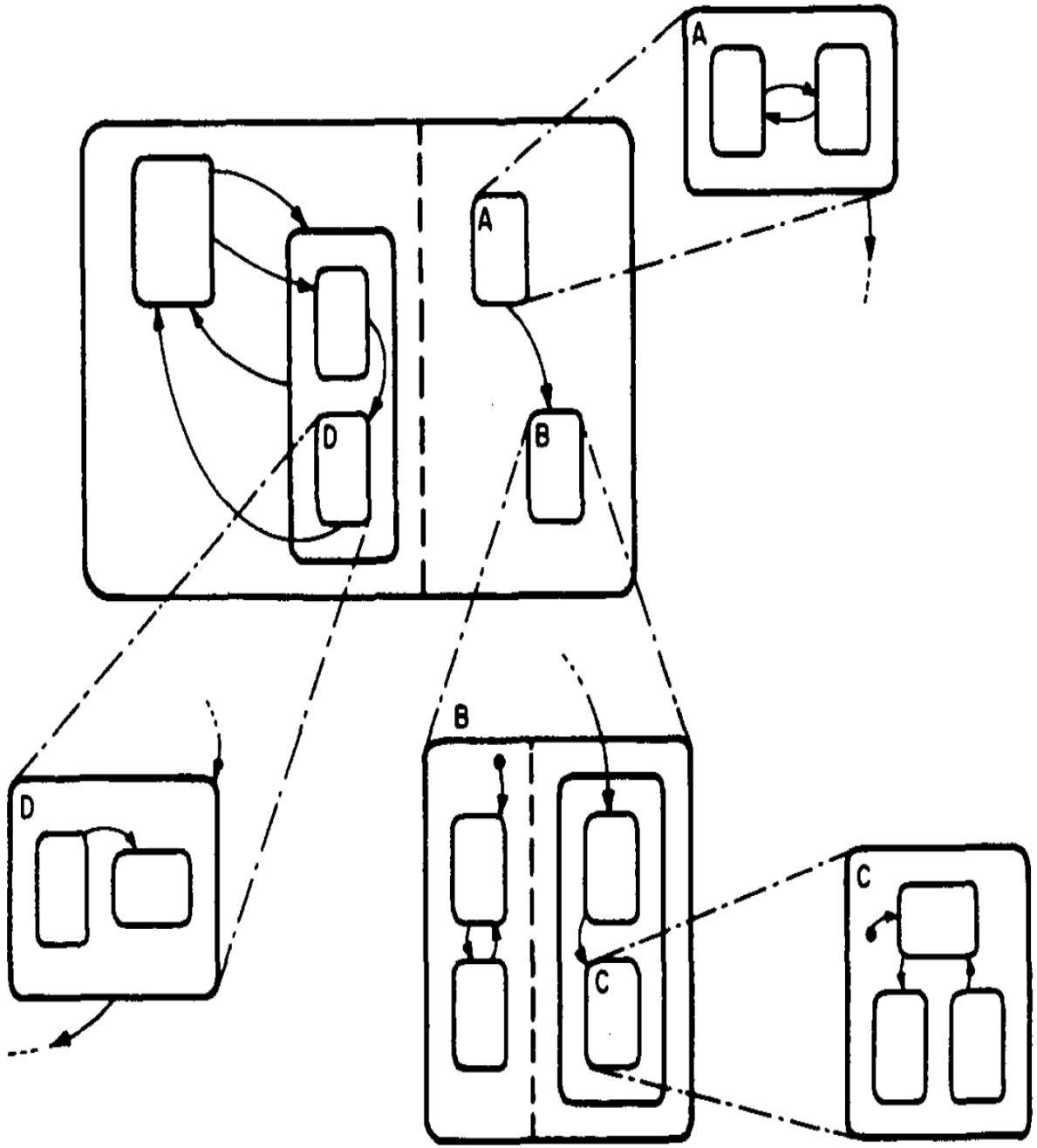
REDUCE GUARDS AND CYCLOMATIC COMPLEXITY

CAPABLE OF MODELING COMPLEX GUI

STATECHARTS

David HAREL (1987):

A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

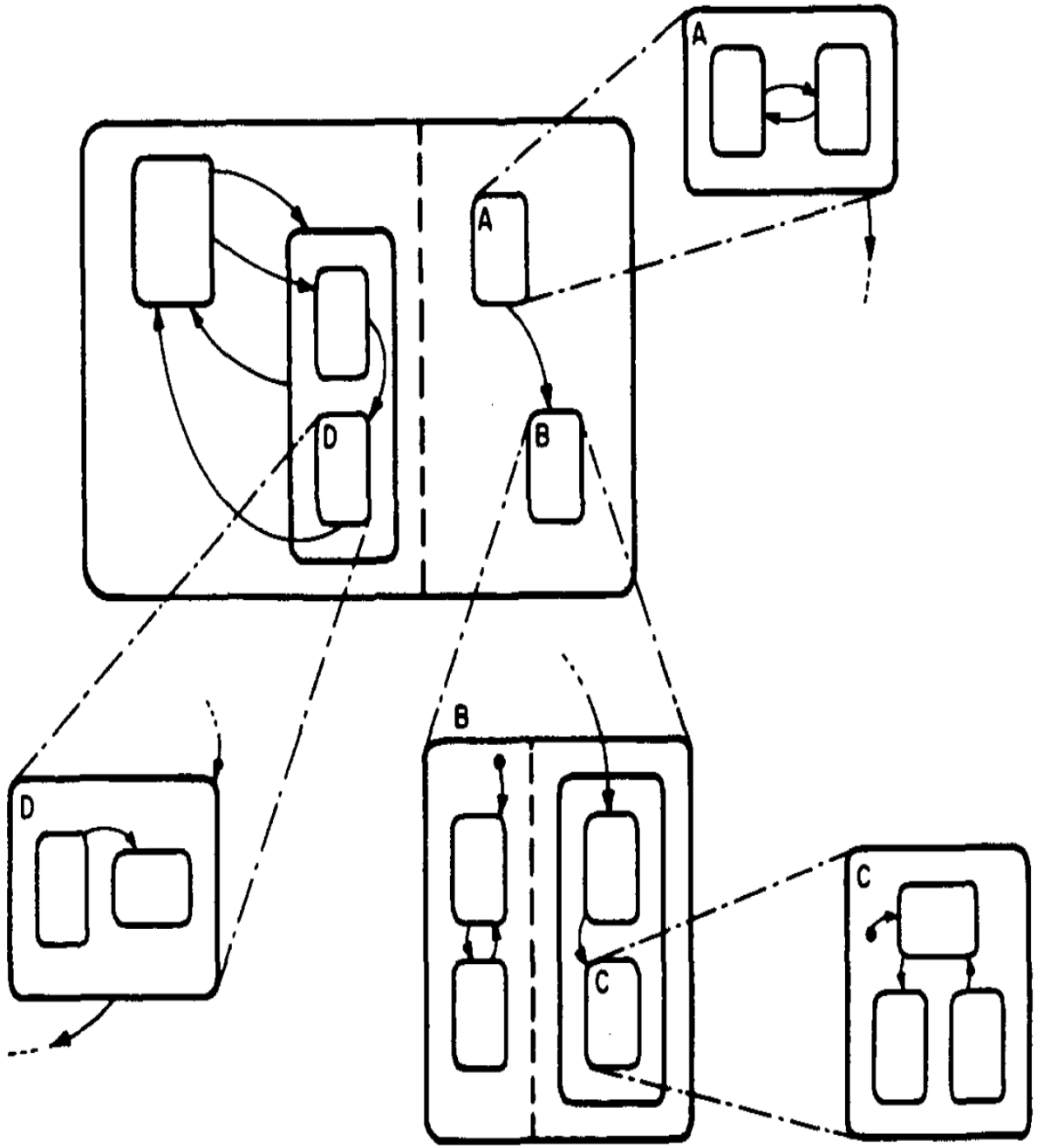


STATECHARTS

David HAREL (1987):

A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

EXTENDS FSM MODEL



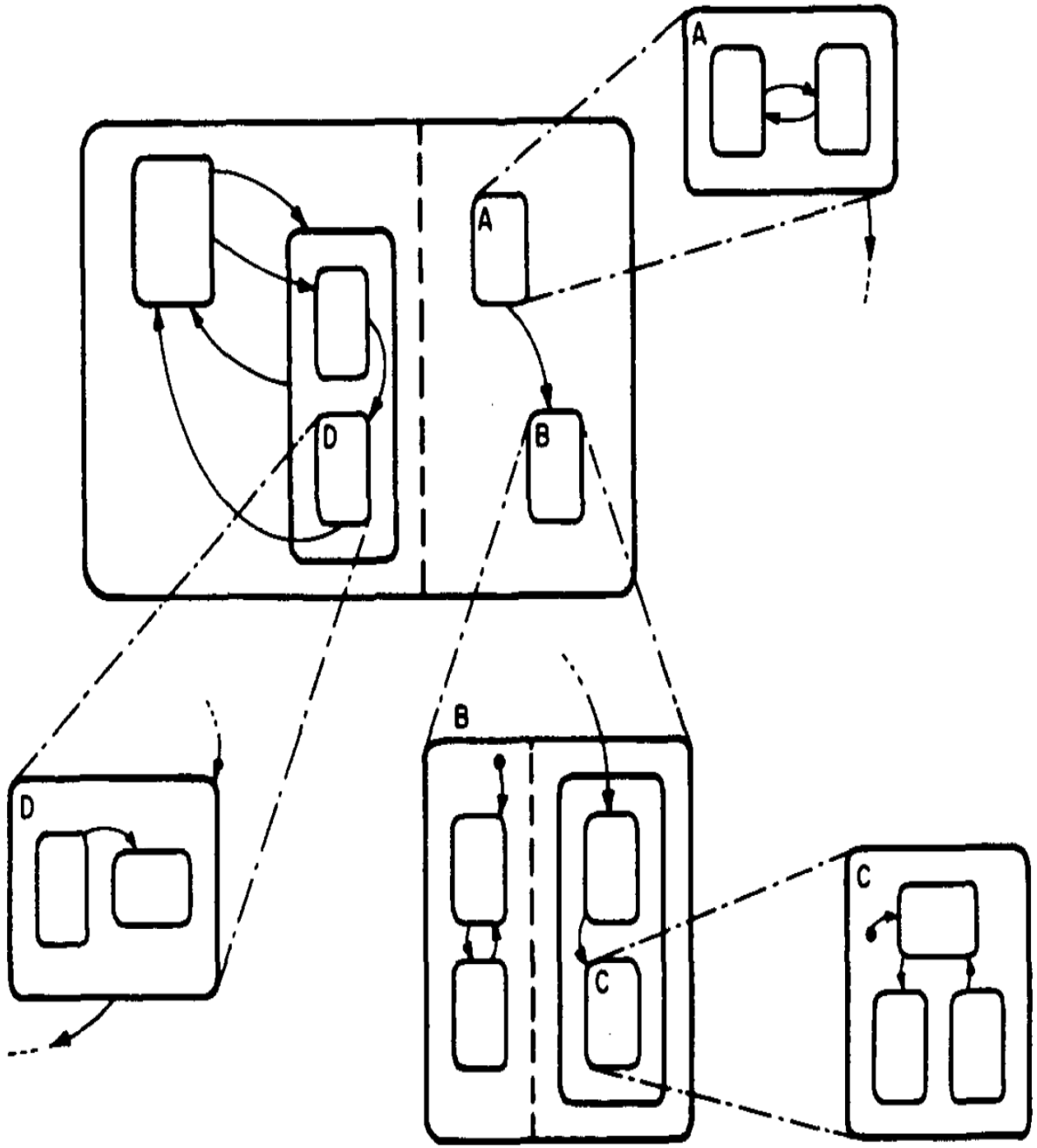
STATECHARTS

David HAREL (1987):

A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

EXTENDS FSM MODEL

STATES WITH RELATIONS



STATECHARTS

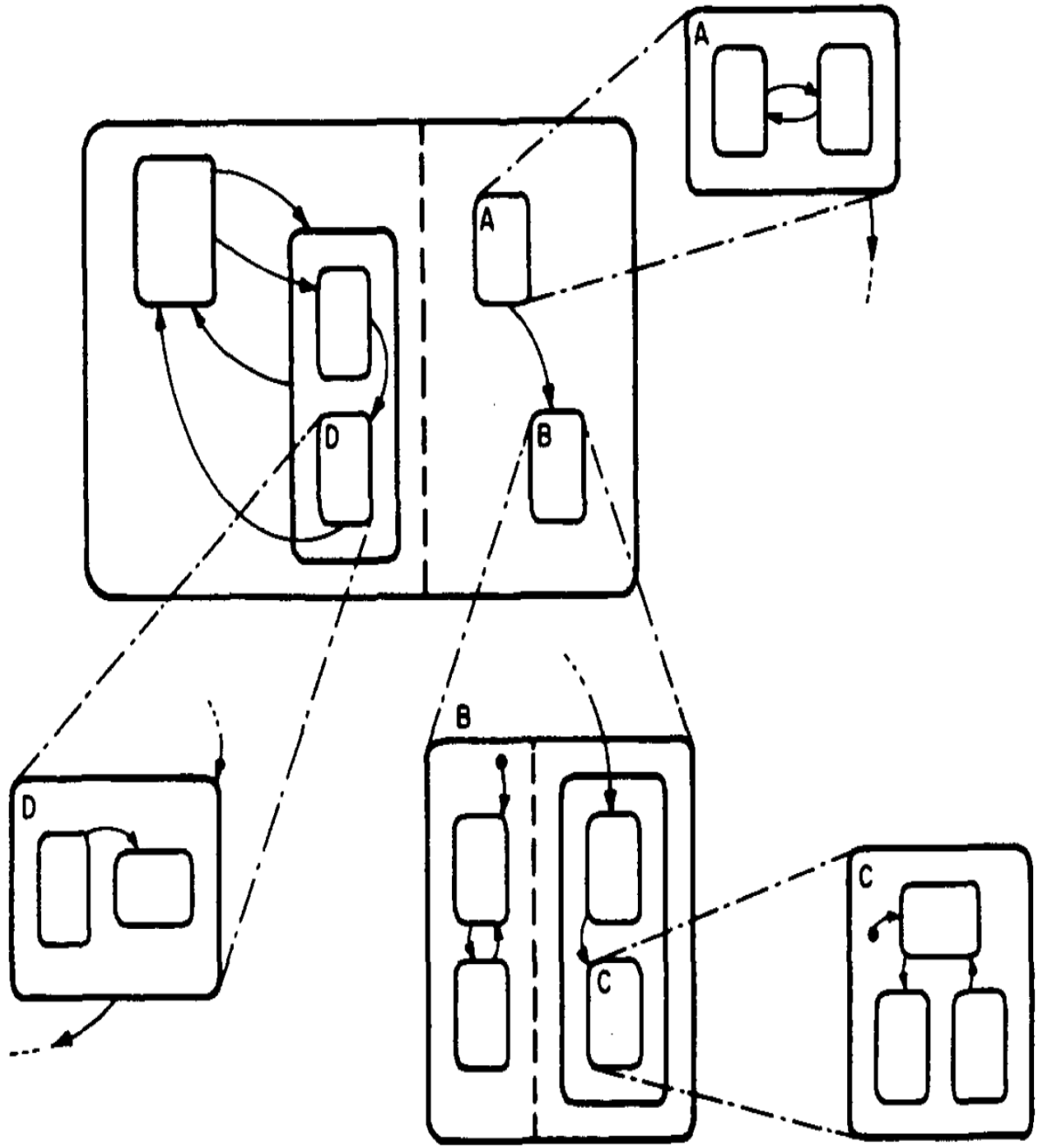
David HAREL (1987):

A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

EXTENDS FSM MODEL

STATES WITH RELATIONS

SEVERAL PLACES TO RUN SIDE EFFECTS



STATECHARTS

David HAREL (1987):

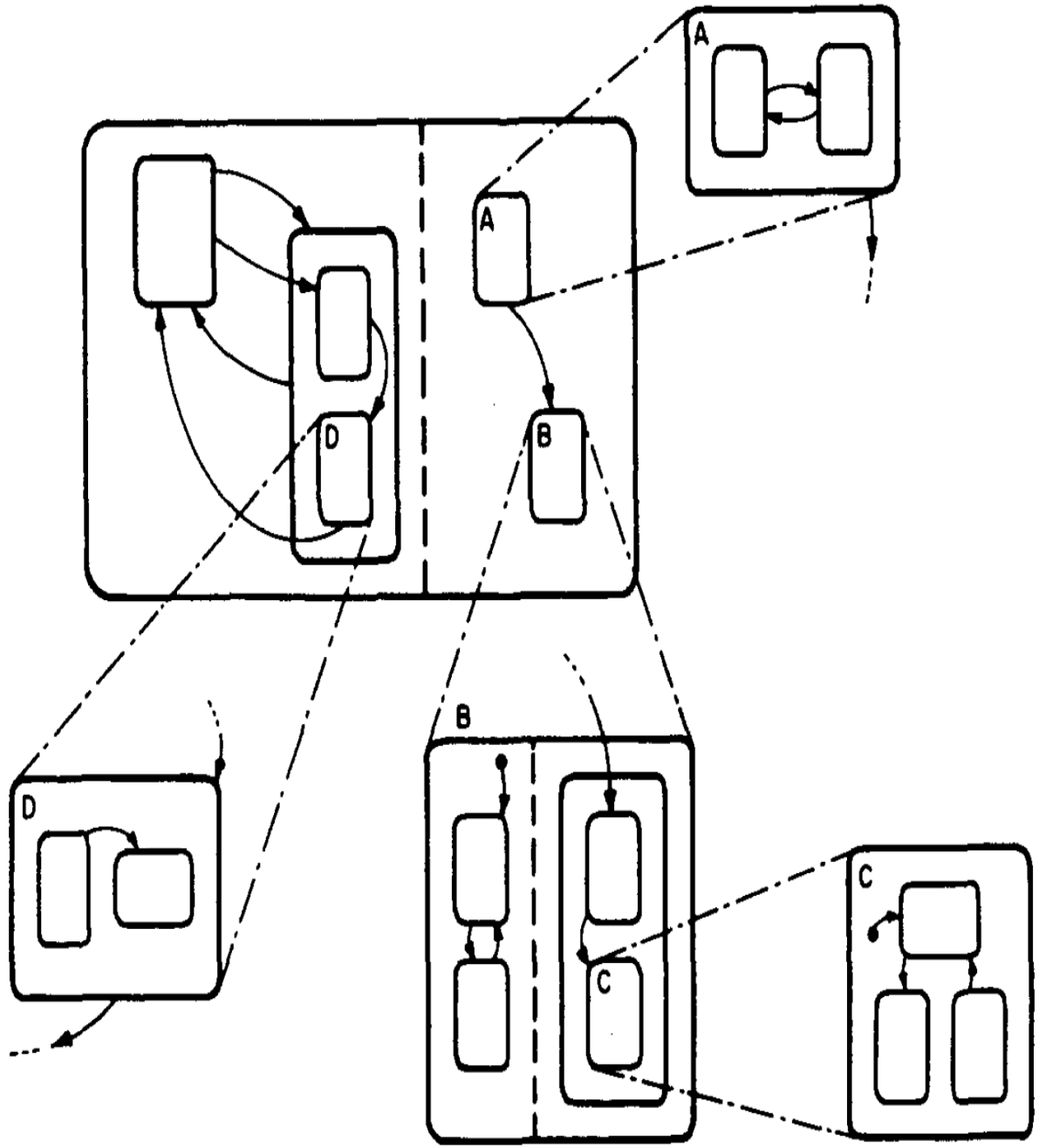
A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

EXTENDS FSM MODEL

STATES WITH RELATIONS

SEVERAL PLACES TO RUN SIDE EFFECTS

SEVERAL TYPES OF SIDE EFFECTS



STATECHARTS

David HAREL (1987):

A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

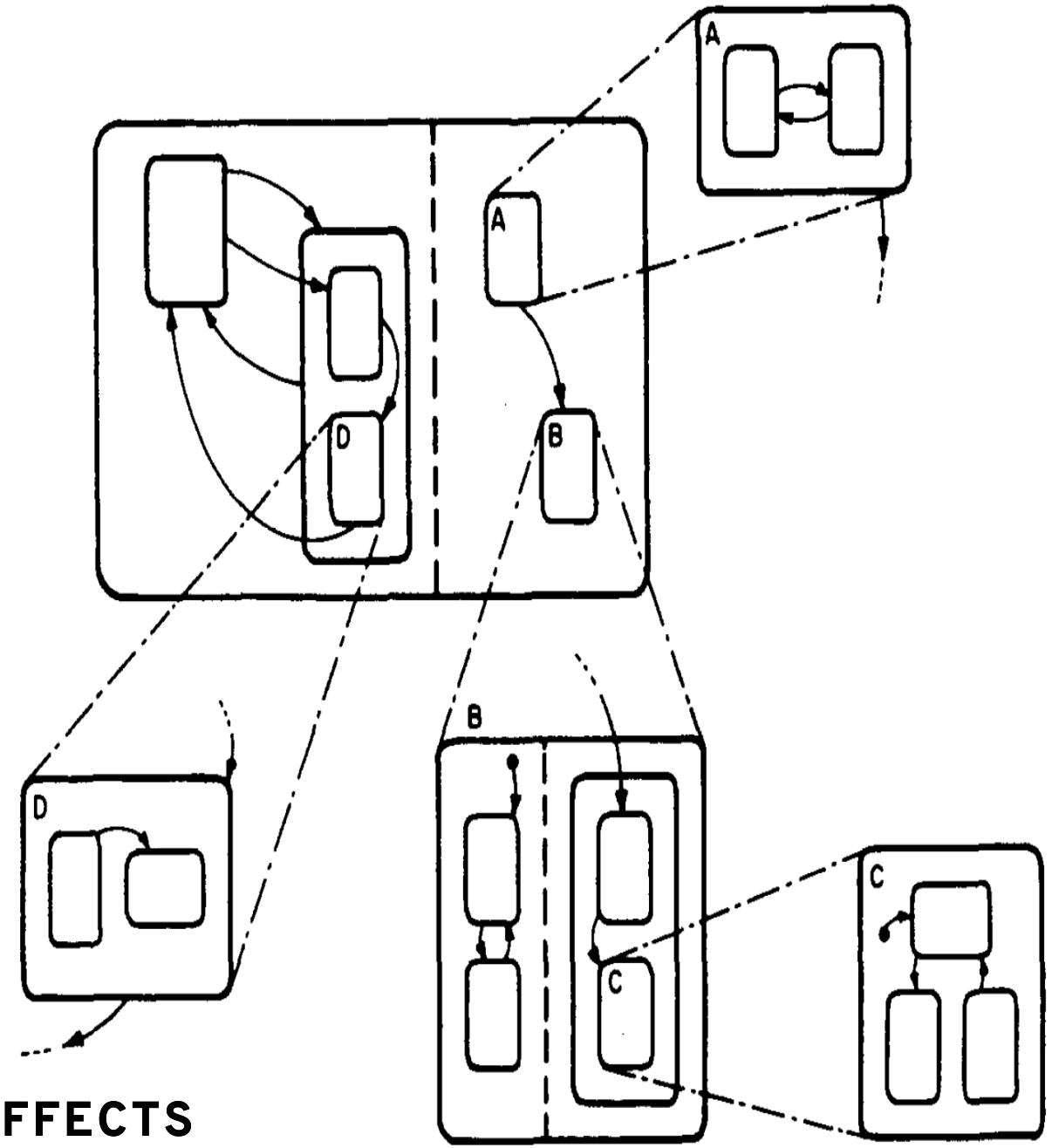
EXTENDS FSM MODEL

STATES WITH RELATIONS

SEVERAL PLACES TO RUN SIDE EFFECTS

SEVERAL TYPES OF SIDE EFFECTS

STATE + EVENT => NEXT STATE + SIDE EFFECTS



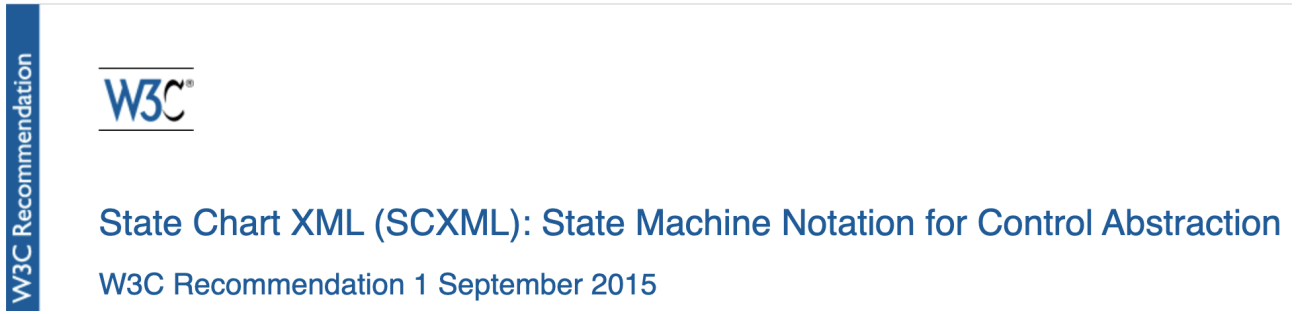
STATECHART IMPLEMENTATION LIBRARY

XSTATE https://xstate.js.org/

STATECHART IMPLEMENTATION LIBRARY

XSTATE https://xstate.js.org/

BASED ON SCXML

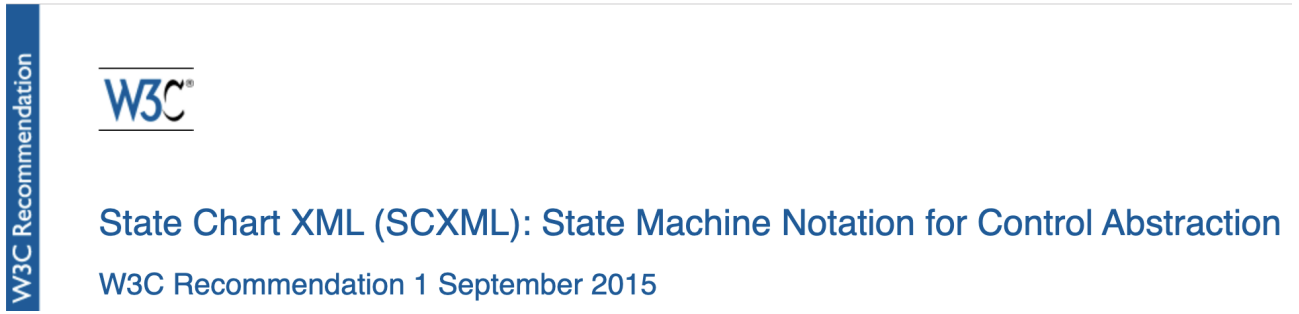


STATECHART IMPLEMENTATION LIBRARY

XSTATE https://xstate.js.org/

BASED ON SCXML

JSON DEFINITION



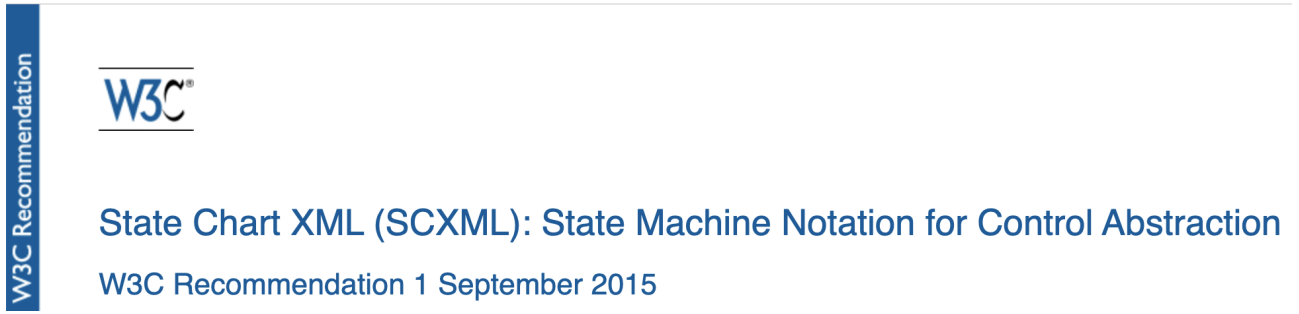
STATECHART IMPLEMENTATION LIBRARY

XSTATE https://xstate.js.org/

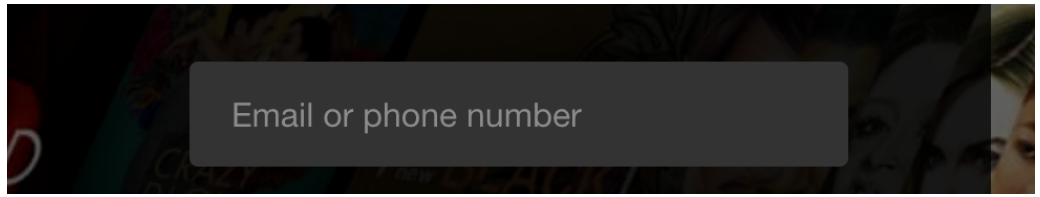
BASED ON SCXML

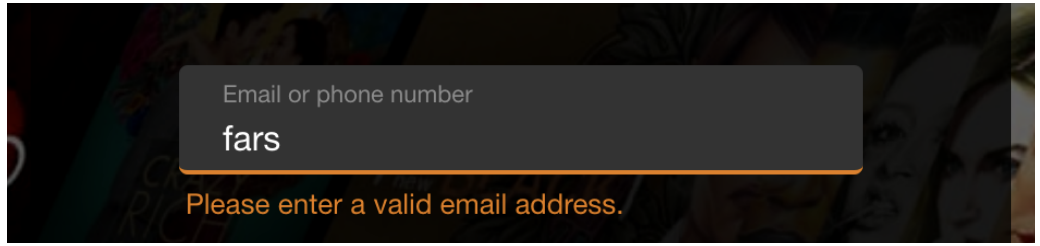
JSON DEFINITION

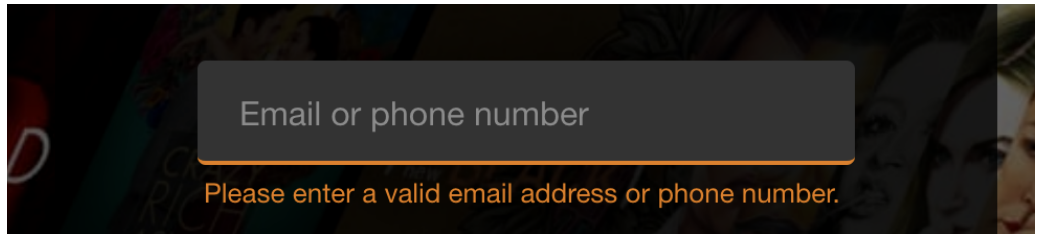
BUILT-IN VISUALIZOR

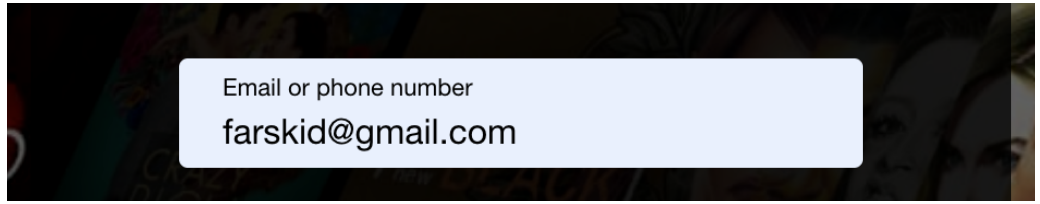


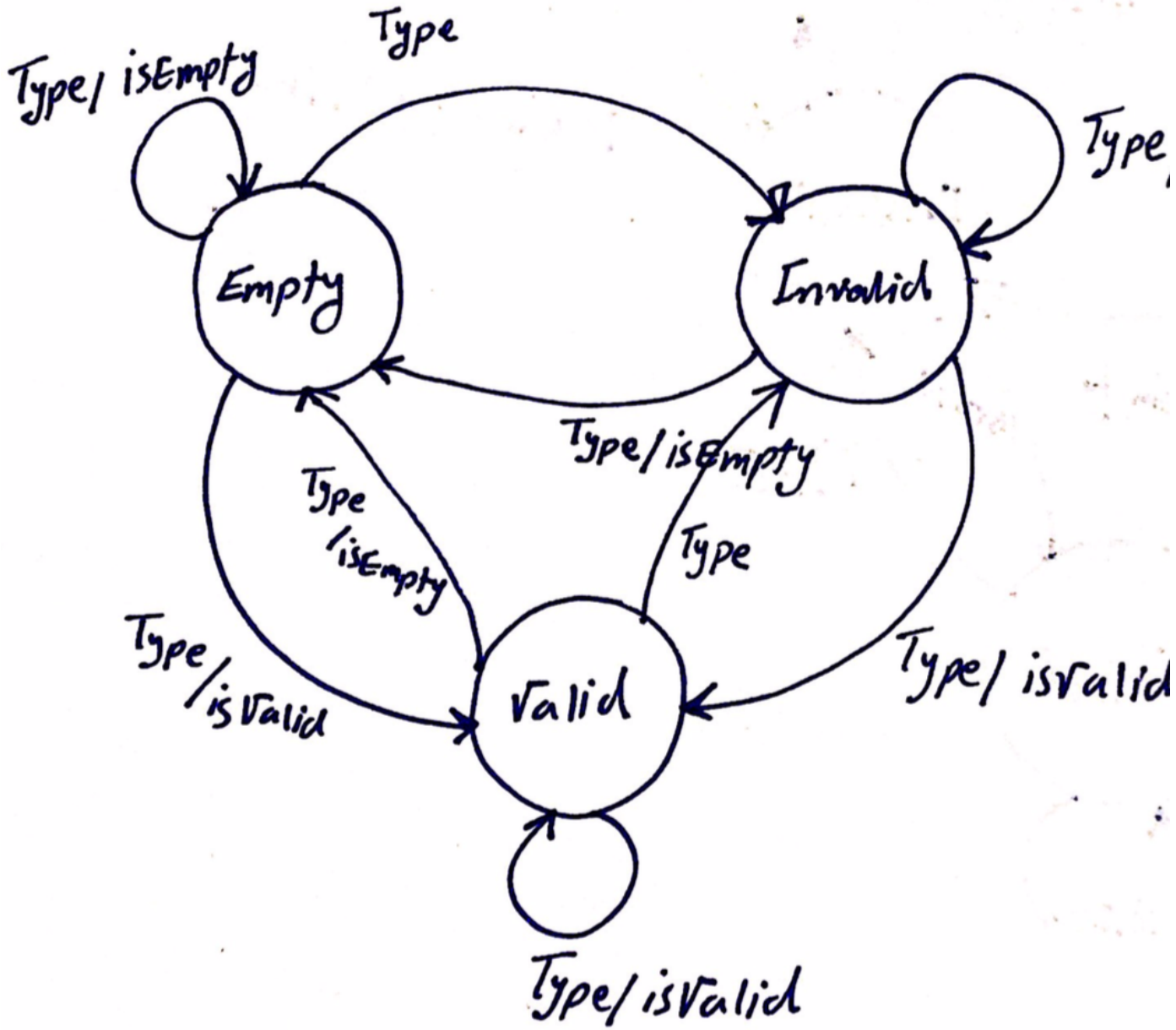
REWRITE THE INPUT IN STATECHARTS





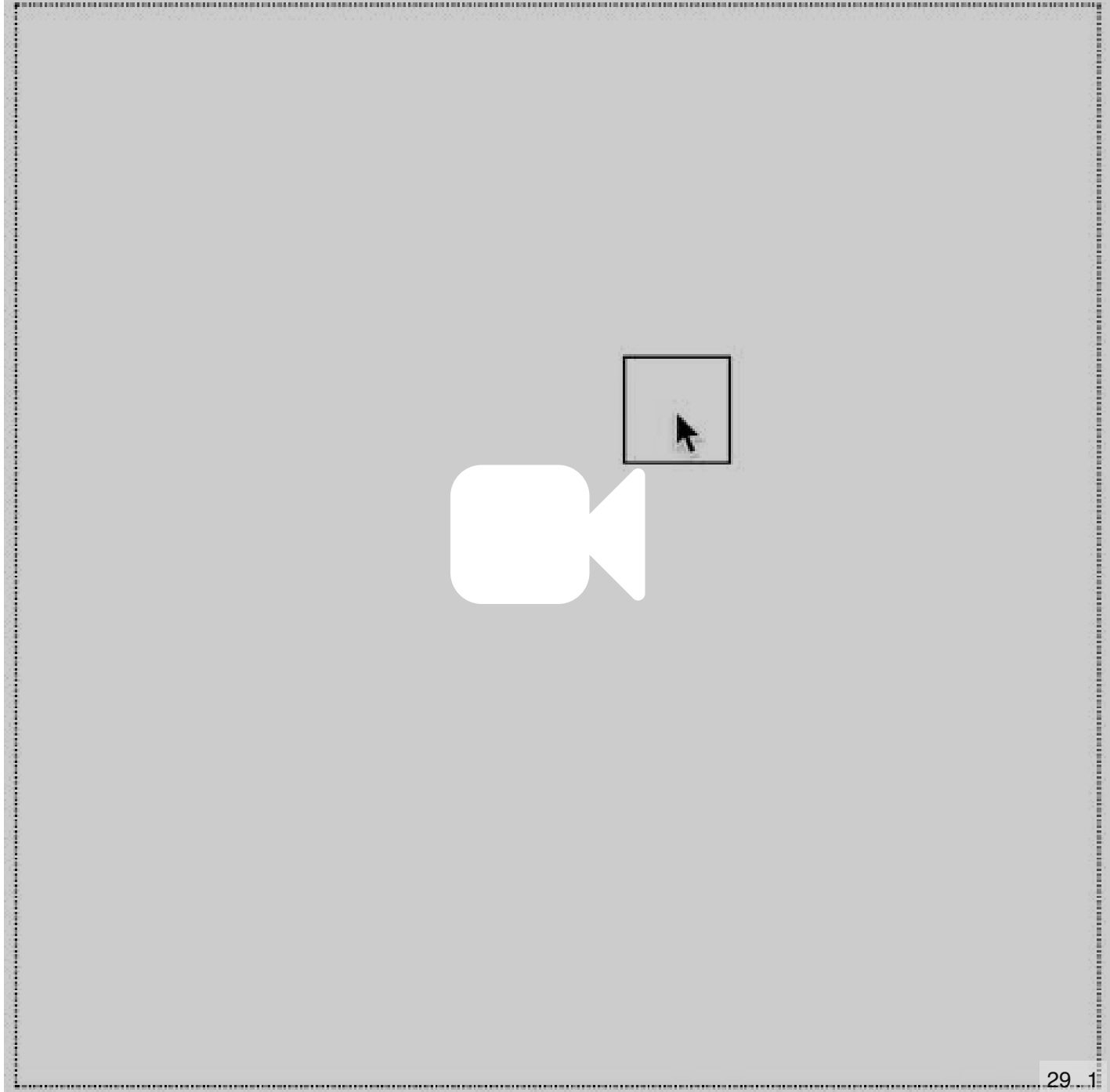




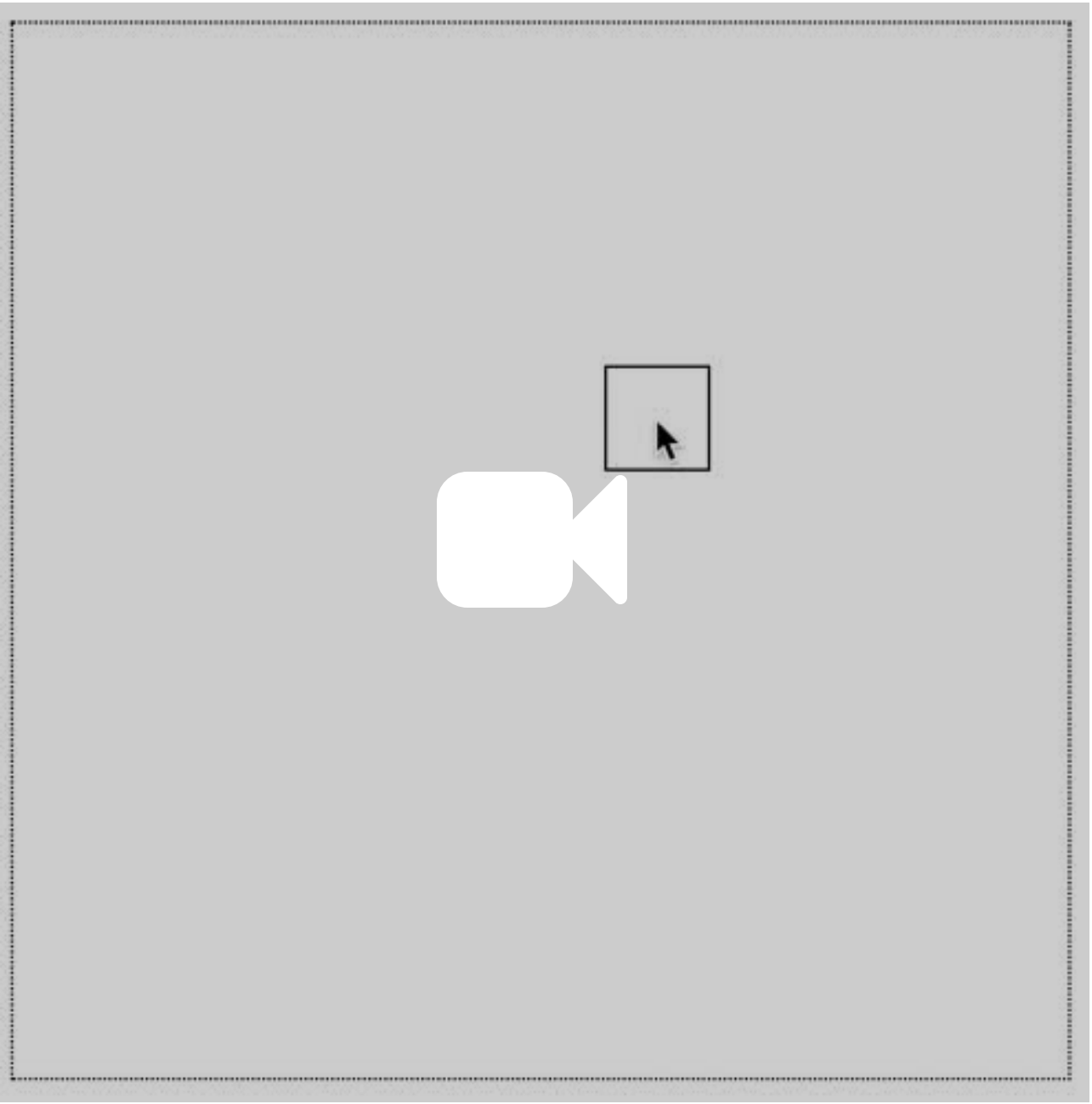


INTERACTIONS IN STATECHARTS

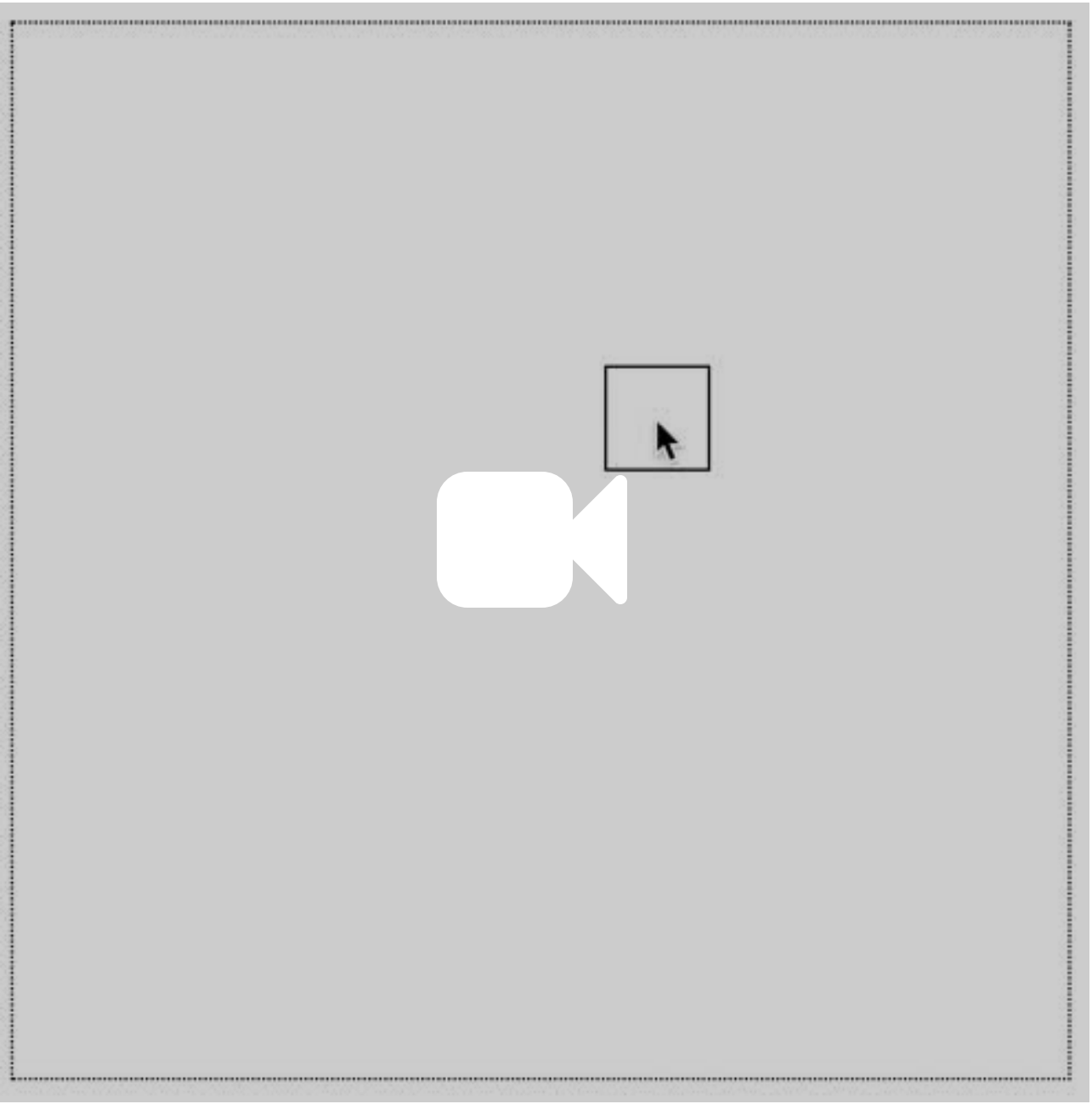
MODELING A DRAGGING BOX



INTERACTIONS WITH STATECHARTS

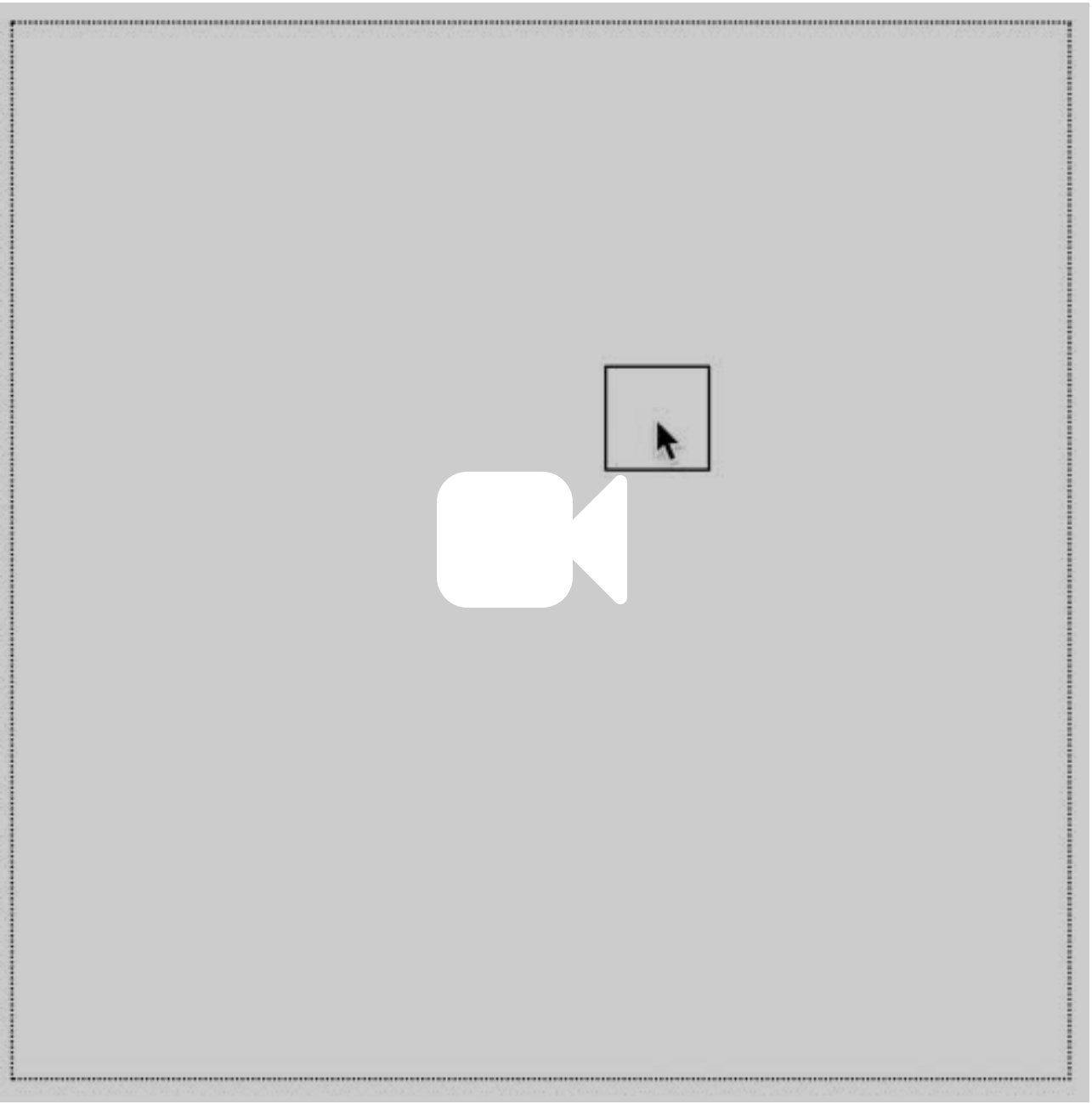


INTERACTIONS WITH STATECHARTS

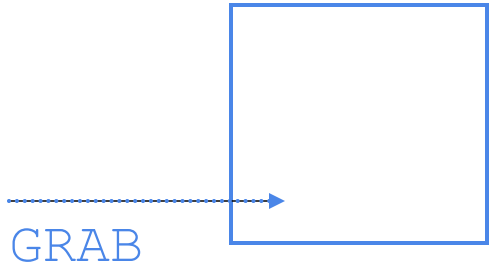


Released

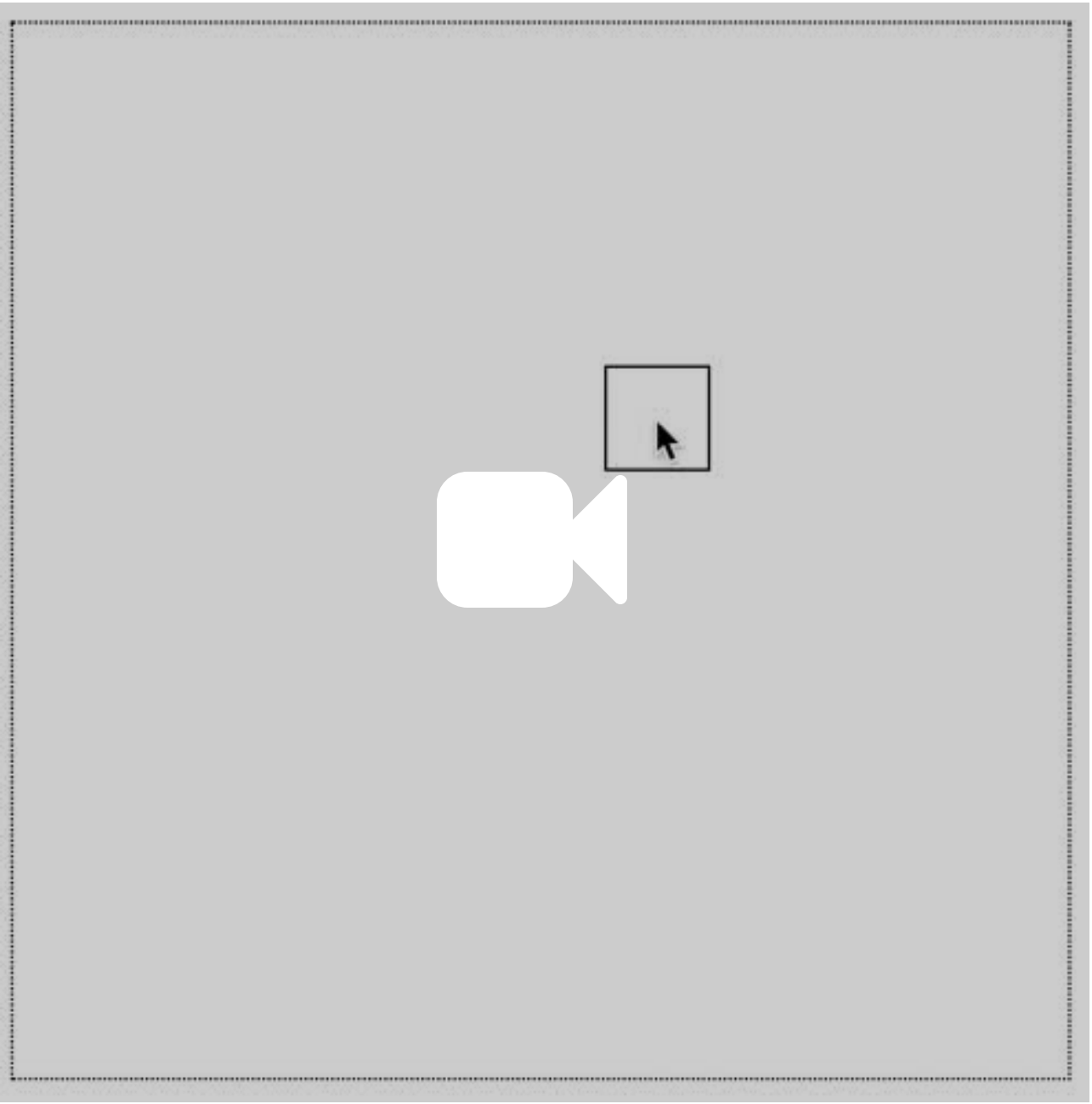
INTERACTIONS WITH STATECHARTS

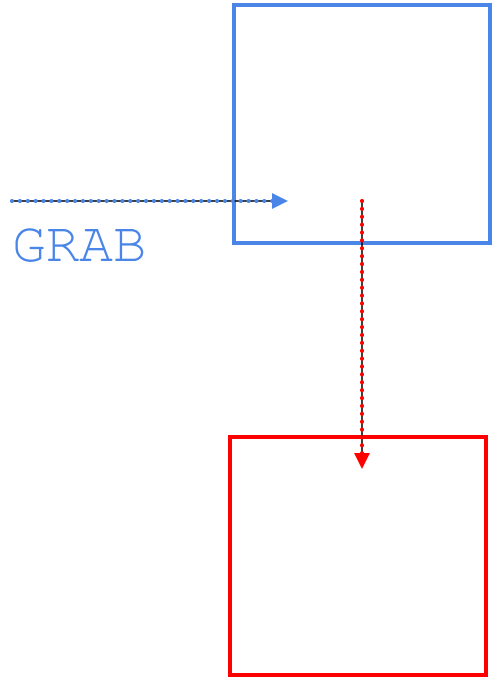


Released

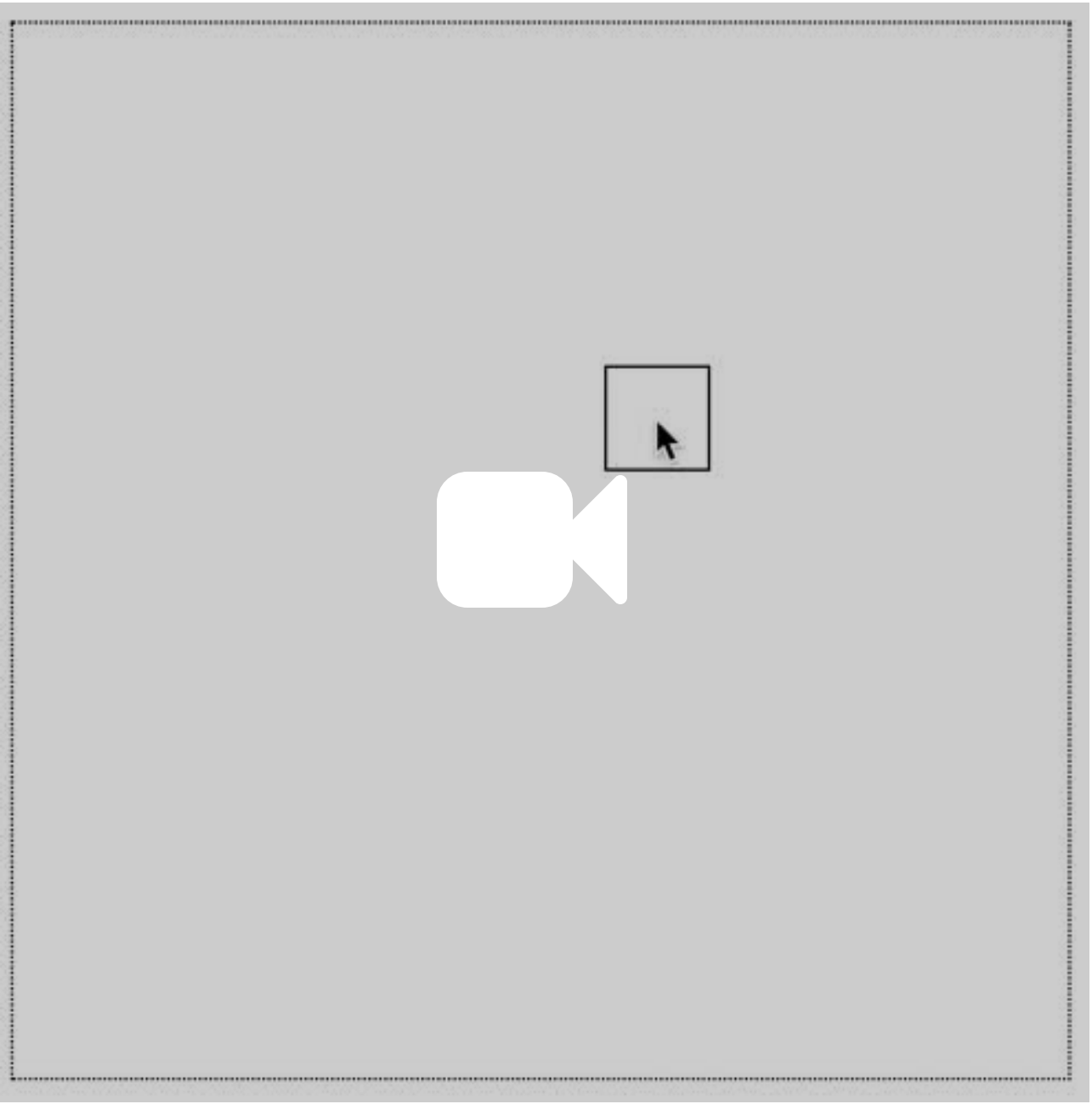


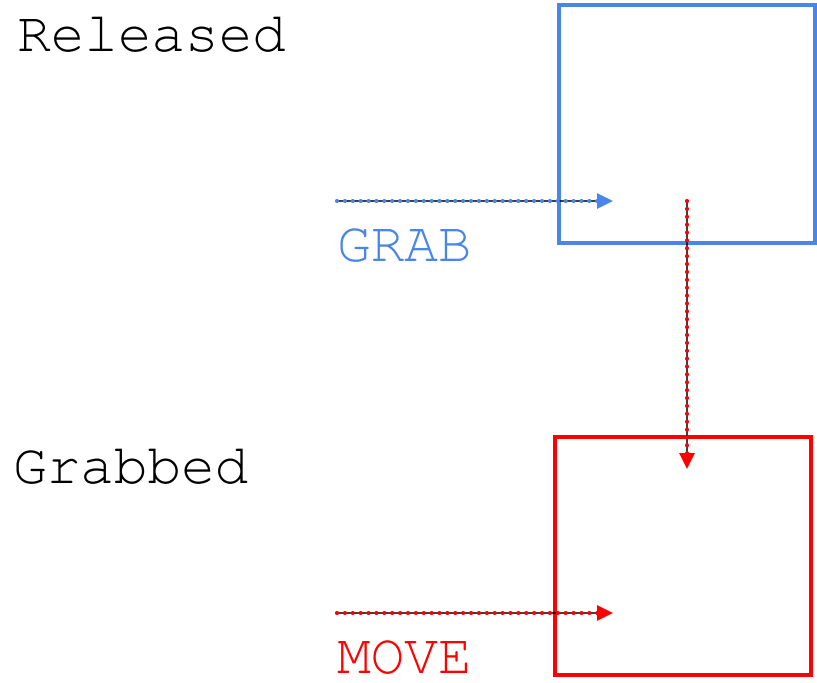
INTERACTIONS WITH STATECHARTS



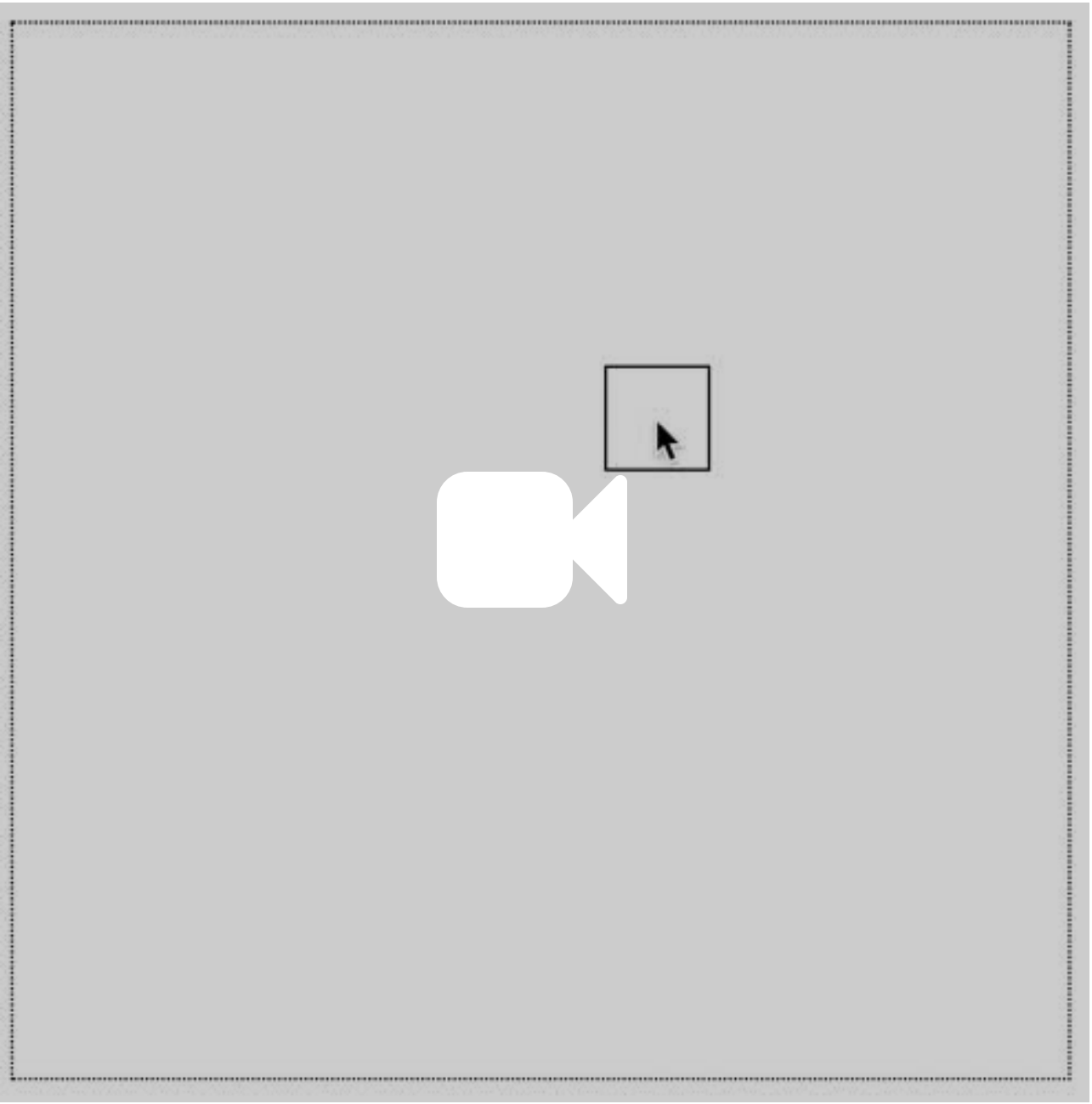


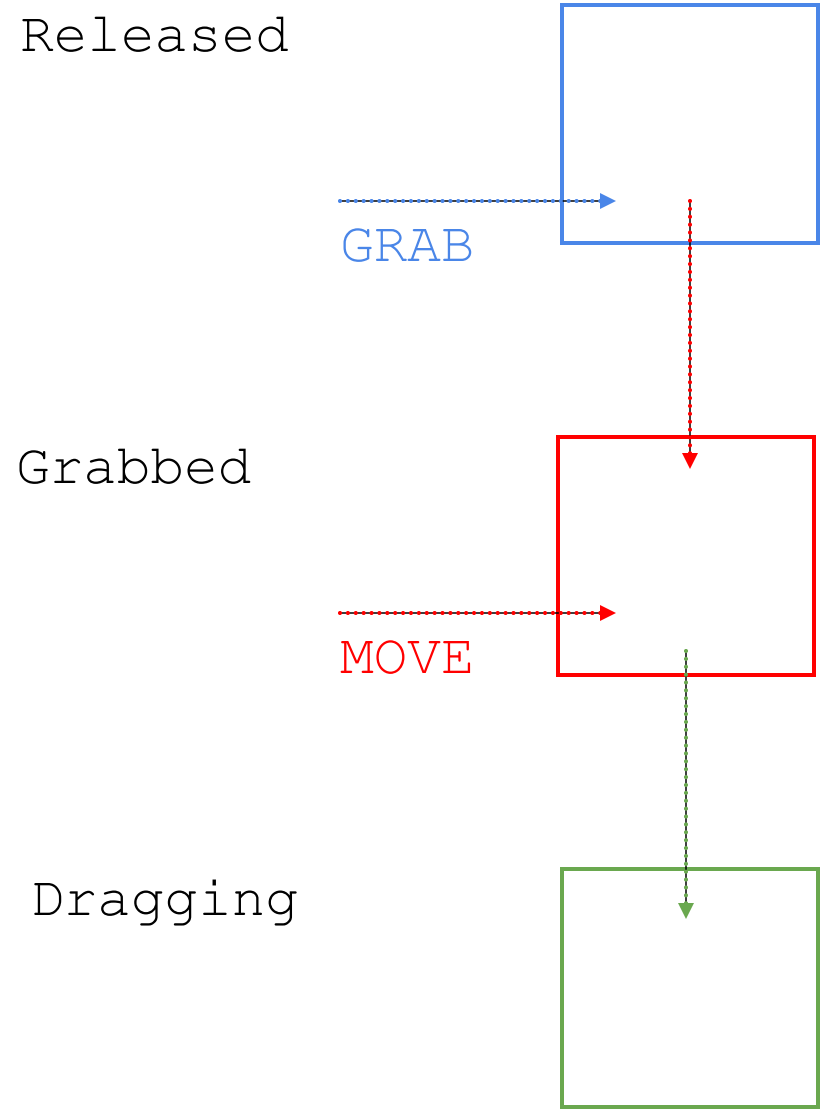
INTERACTIONS WITH STATECHARTS



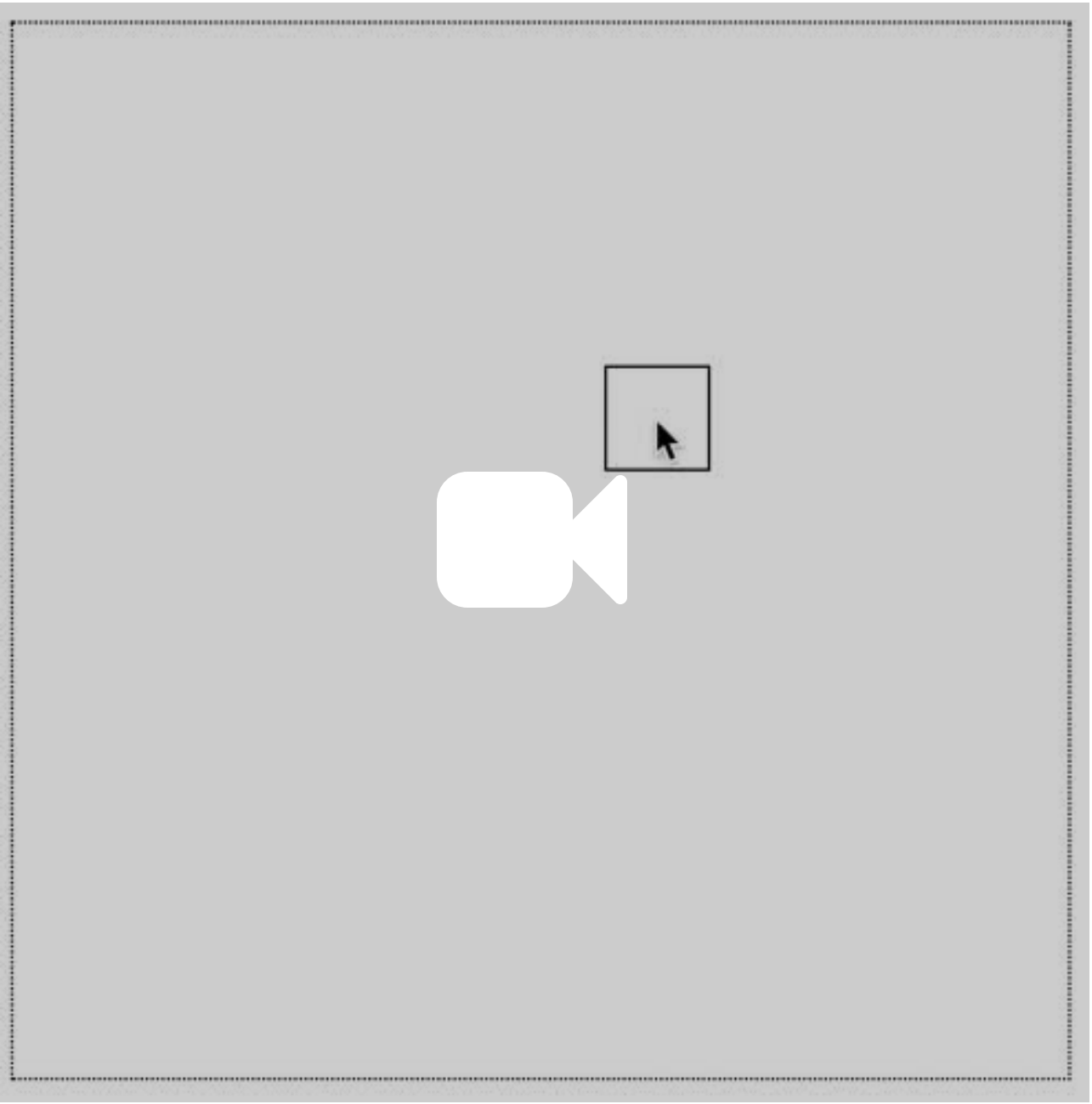


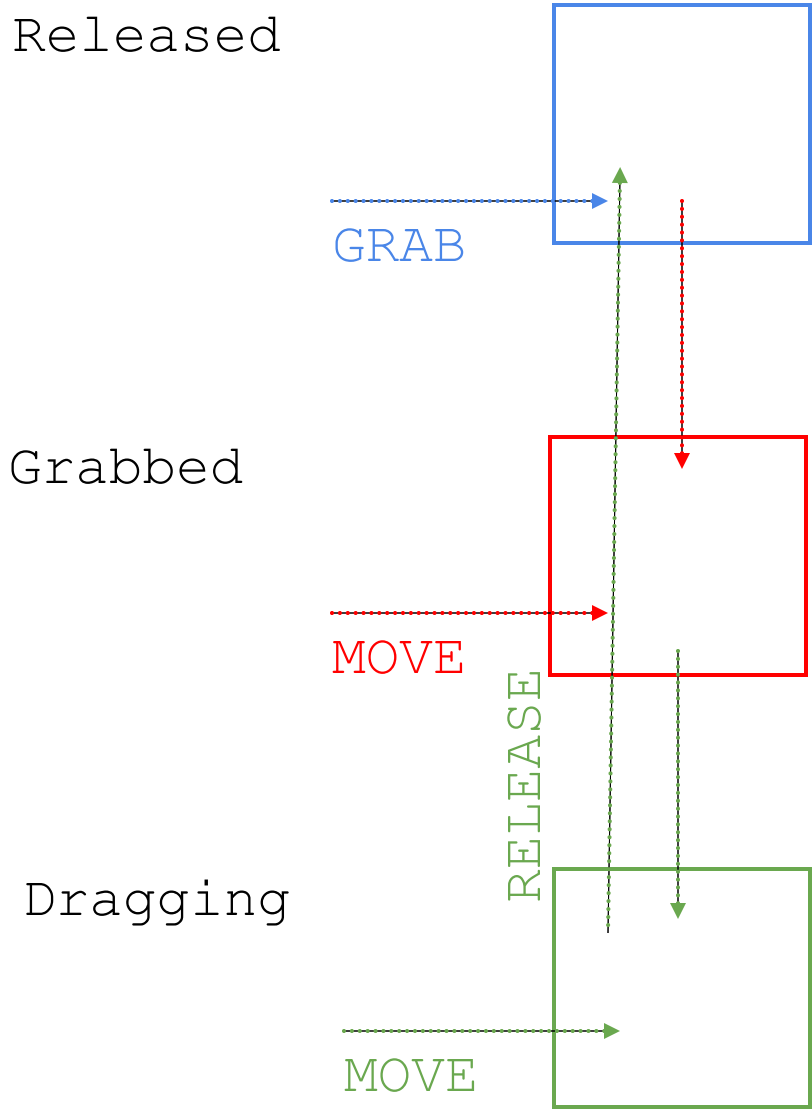
INTERACTIONS WITH STATECHARTS



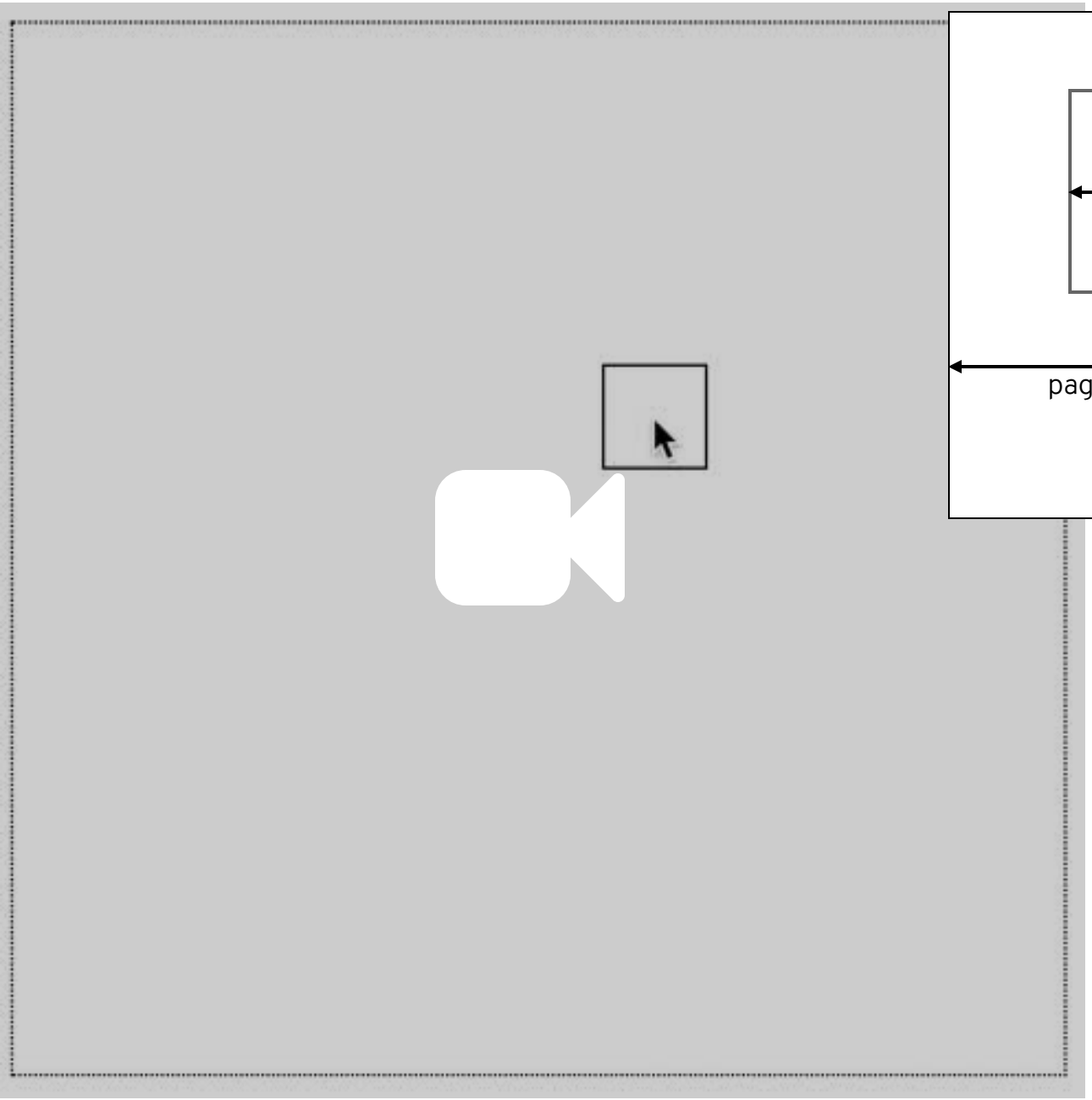


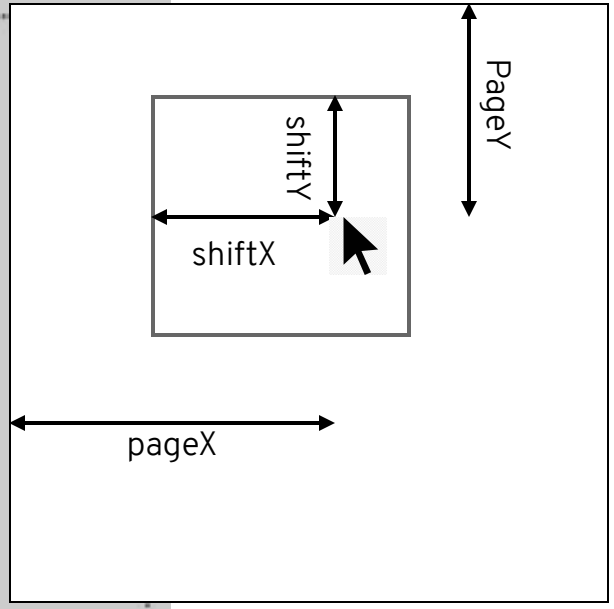
INTERACTIONS WITH STATECHARTS





INTERACTIONS WITH STATECHARTS

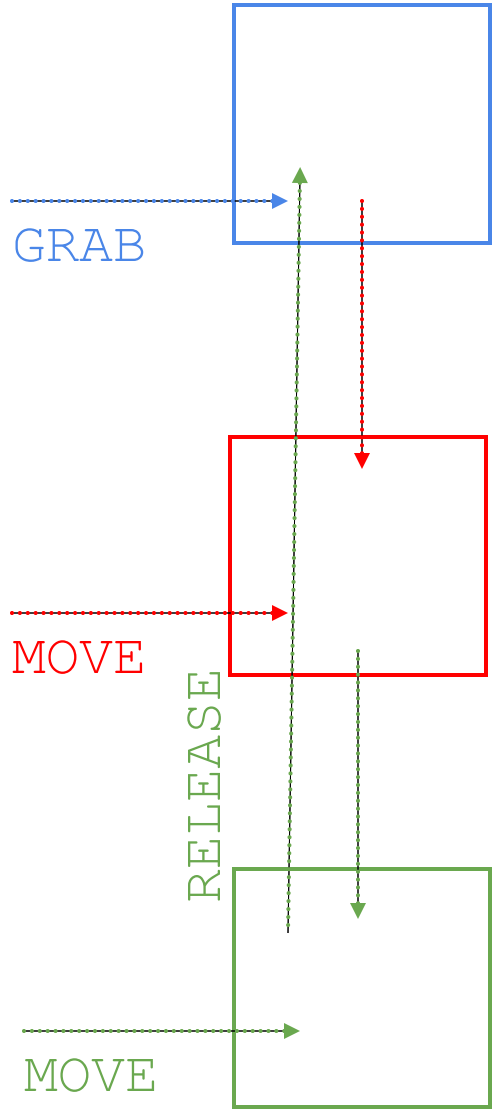




Released

Grabbed

Dragging



EVENT LISTENERS AND STATECHARTS

```
onMouseDown = () => {
  sendEvent({ type: "GRAB", data: { shiftX, shiftY
} });
}

onMouseMove = () => {
  sendEvent({
    type: "MOVE",
    data: { x: event.pageX, y: event.pageY }
  });
}

onMouseUp = () => {
  sendEvent("RELEASE");
};
```

BEFORE

```
box.onmousedown = function(event) {
  // (1) prepare to moving: make absolute and on top by z-index
  box.style.position = 'absolute';
  box.style.zIndex = 1000;
  // move it out of any current parents directly into body
  // to make it positioned relative to the body
  document.body.append(box);
  // ...and put that absolutely positioned ball under the pointer

  moveAt(event.pageX, event.pageY);

  // centers the ball at (pageX, pageY) coordinates
  function moveAt(pageX, pageY) {
    box.style.left = pageX - box.shiftX + 'px';
    box.style.top = pageY - box.shiftY + 'px';
  }

  function onMouseMove(event) {
    moveAt(event.pageX, event.pageY);
  }

  // (2) move the ball on mousemove
  document.addEventListener('mousemove', onMouseMove);

  // (3) drop the ball, remove unneeded handlers
  box.onmouseup = function() {
    document.removeEventListener('mousemove', onMouseMove);
    box.onmouseup = null;
  };
};
```

AFTER

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

source: https://javascript.info/mouse-drag-and-drop

Scene #5

ADDITIONAL BENEFITS

STATECHARTS READ LIKE ENGLISH

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

When it's grabbed, it can move

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

When it's grabbed, it can move

As soon as it's moving, we update its current position and move it.

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

When it's grabbed, it can move

As soon as it's moving, we update its current position and move it.

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

When it's grabbed, it can move

As soon as it's moving, we update its current position and move it.

When it's moving, it can be released

```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS READ LIKE ENGLISH

Box is released at first

When it's released, it can be grabbed

As soon as it's grabbed, we remember mouse position and box position, prepare its styles and move it.

When it's grabbed, it can move

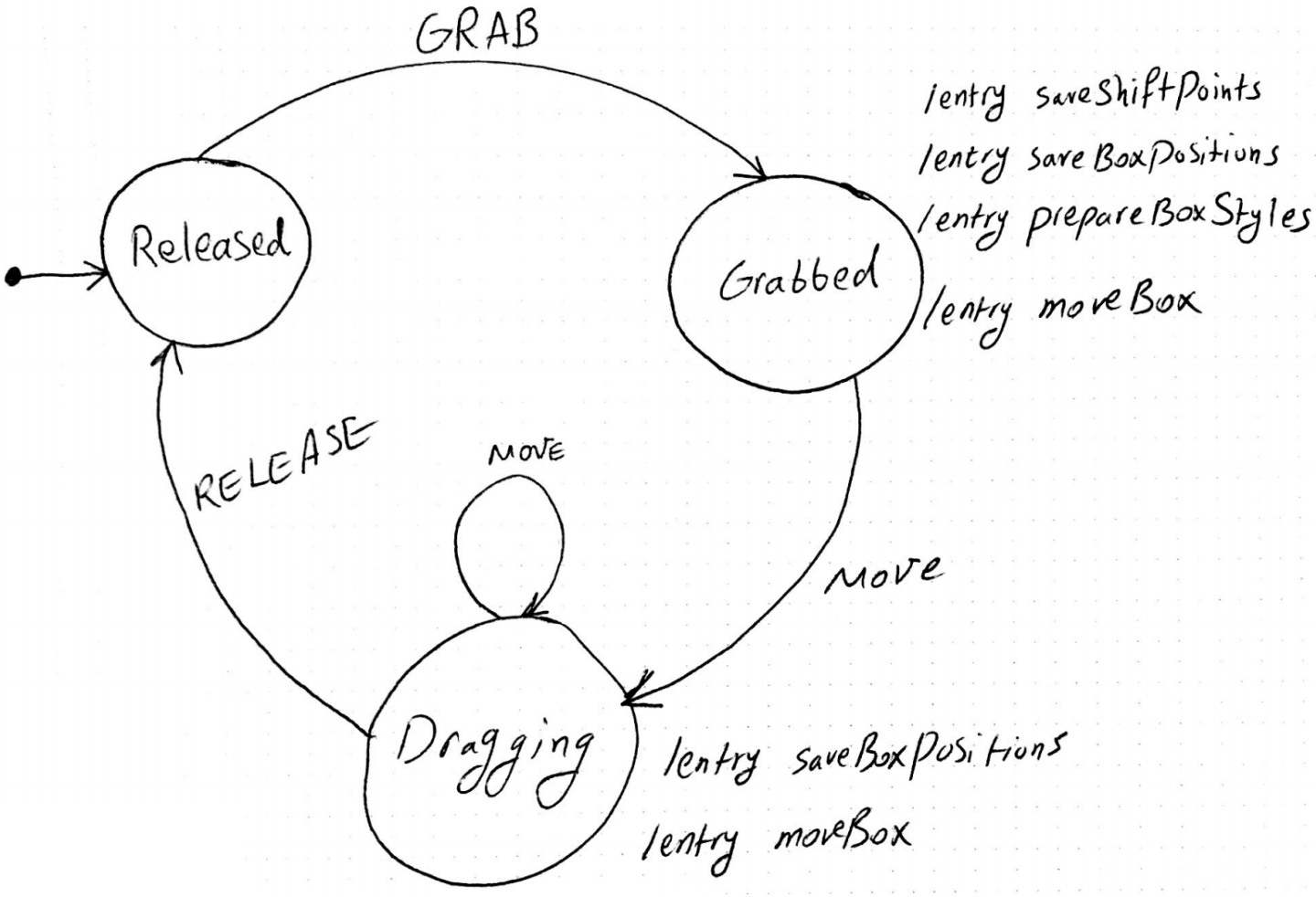
As soon as it's moving, we update its current position and move it.

As long as it's moving, we keep moving it which means continuously updating its position.

When it's moving, it can be released

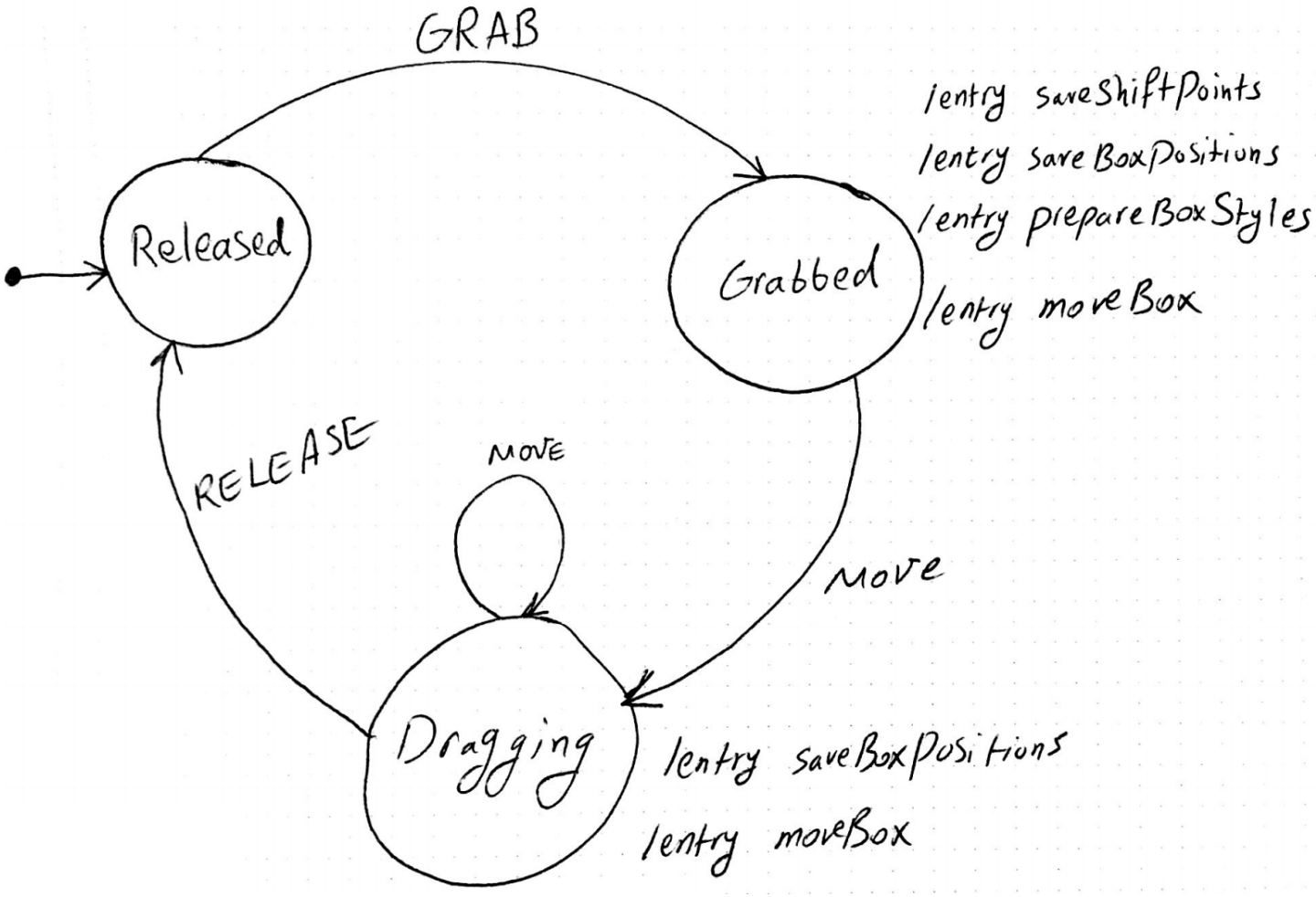
```
{
  initial: "released",
  context: {
    shiftX: 0,
    shiftY: 0,
    pageX: 0,
    pageY: 0
  },
  states: {
    released: {
      on: {
        GRAB: {
          target: "grabbed"
        }
      }
    },
    grabbed: {
      entry: [
        "saveShiftPoints",
        "saveBoxPositions",
        "prepareBoxStyles",
        "moveBox"
      ],
      on: {
        MOVE: "dragging"
      }
    },
    dragging: {
      entry: [
        "saveBoxPositions",
        "moveBox"
      ],
      on: {
        MOVE: "dragging",
        RELEASE: "released"
      }
    }
  }
}
```

STATECHARTS VISUALIZE LOGIC



STATECHARTS VISUALIZE LOGIC

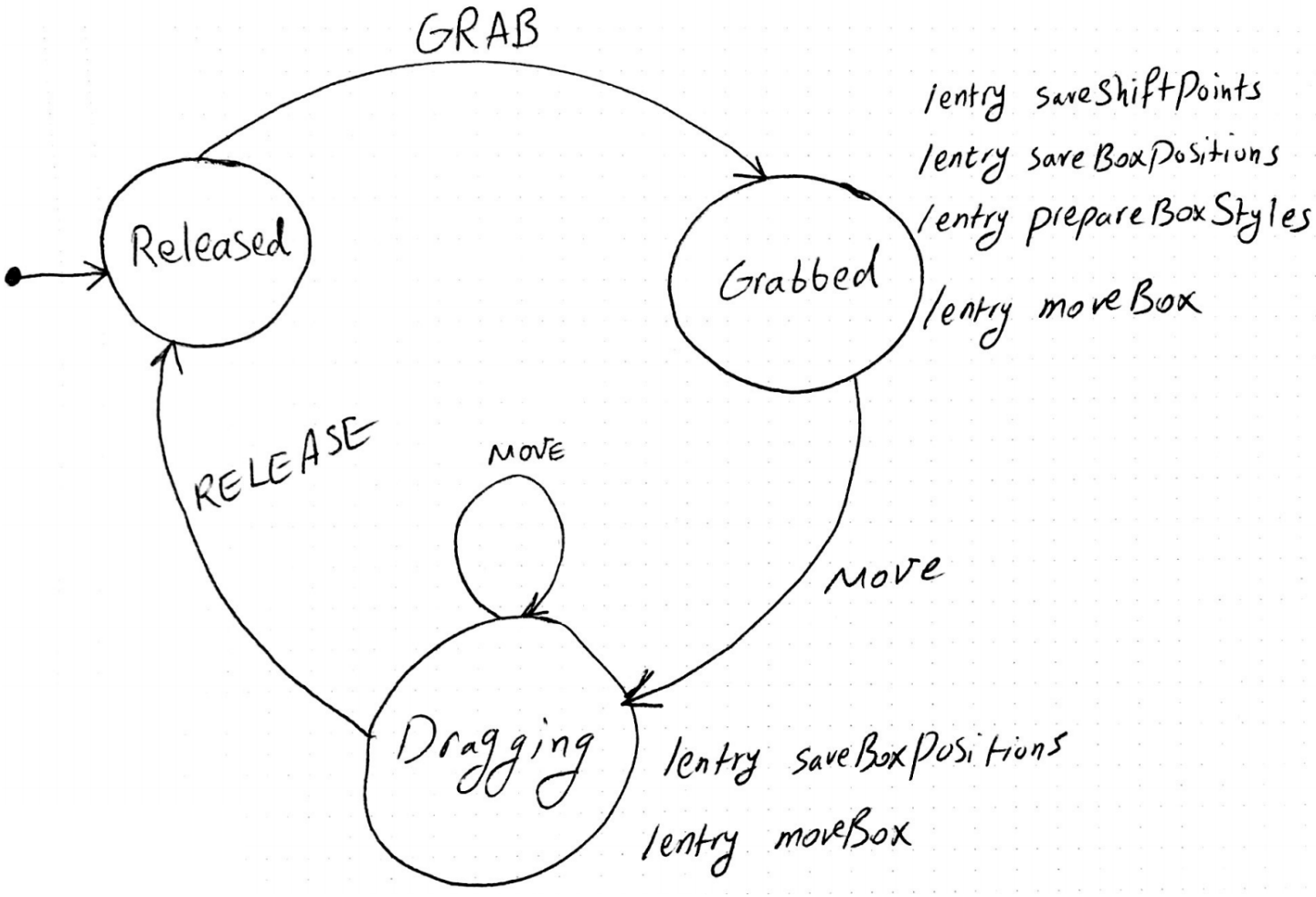
GENERATE DIRECTED GRAPH



STATECHARTS VISUALIZE LOGIC

GENERATE DIRECTED GRAPH

SHOWCASE STATE PATHS

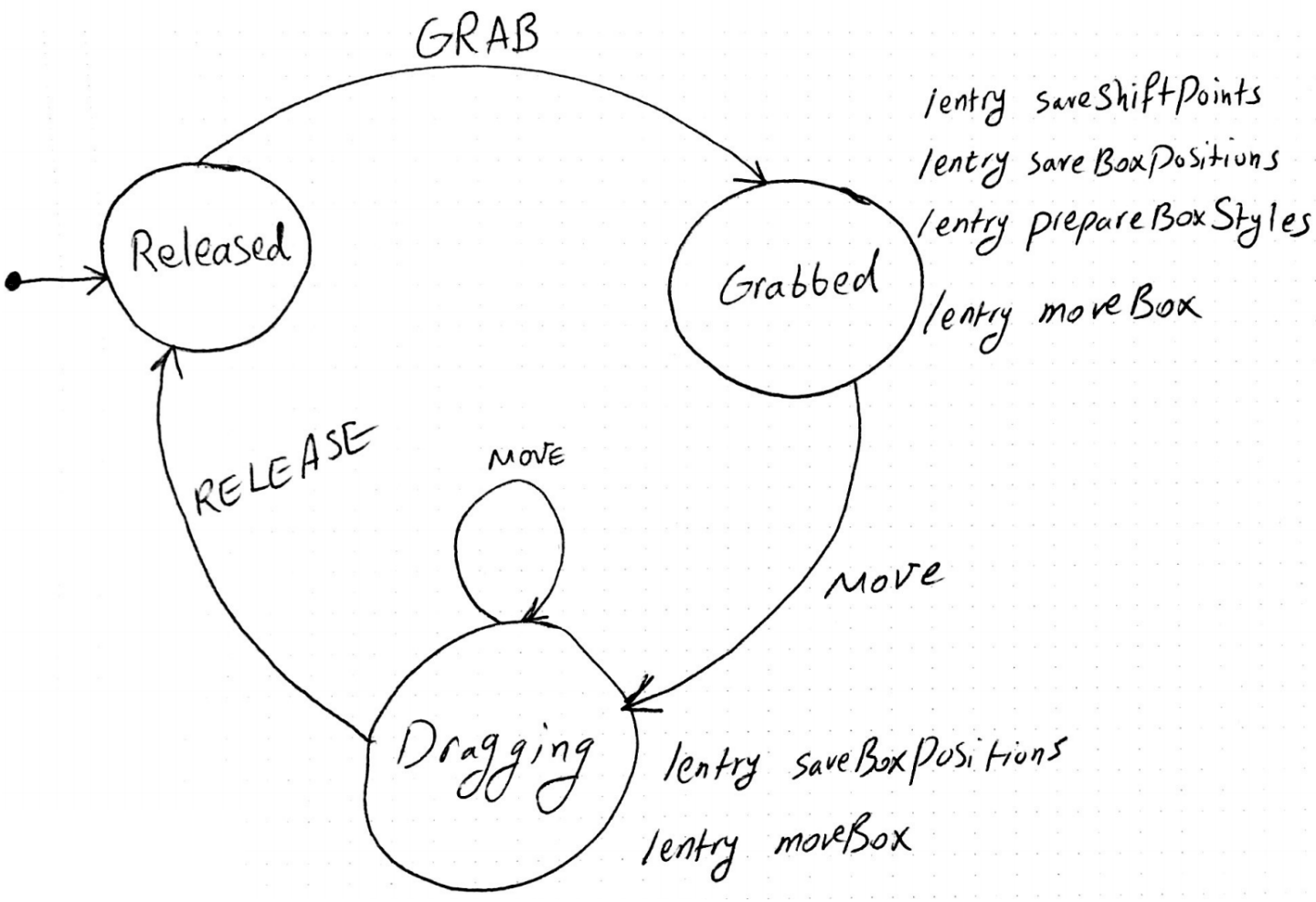


STATECHARTS VISUALIZE LOGIC

GENERATE DIRECTED GRAPH

SHOWCASE STATE PATHS

PATHS CAN BE USED BY QA TEAM TO TEST FOR EDGE CASES



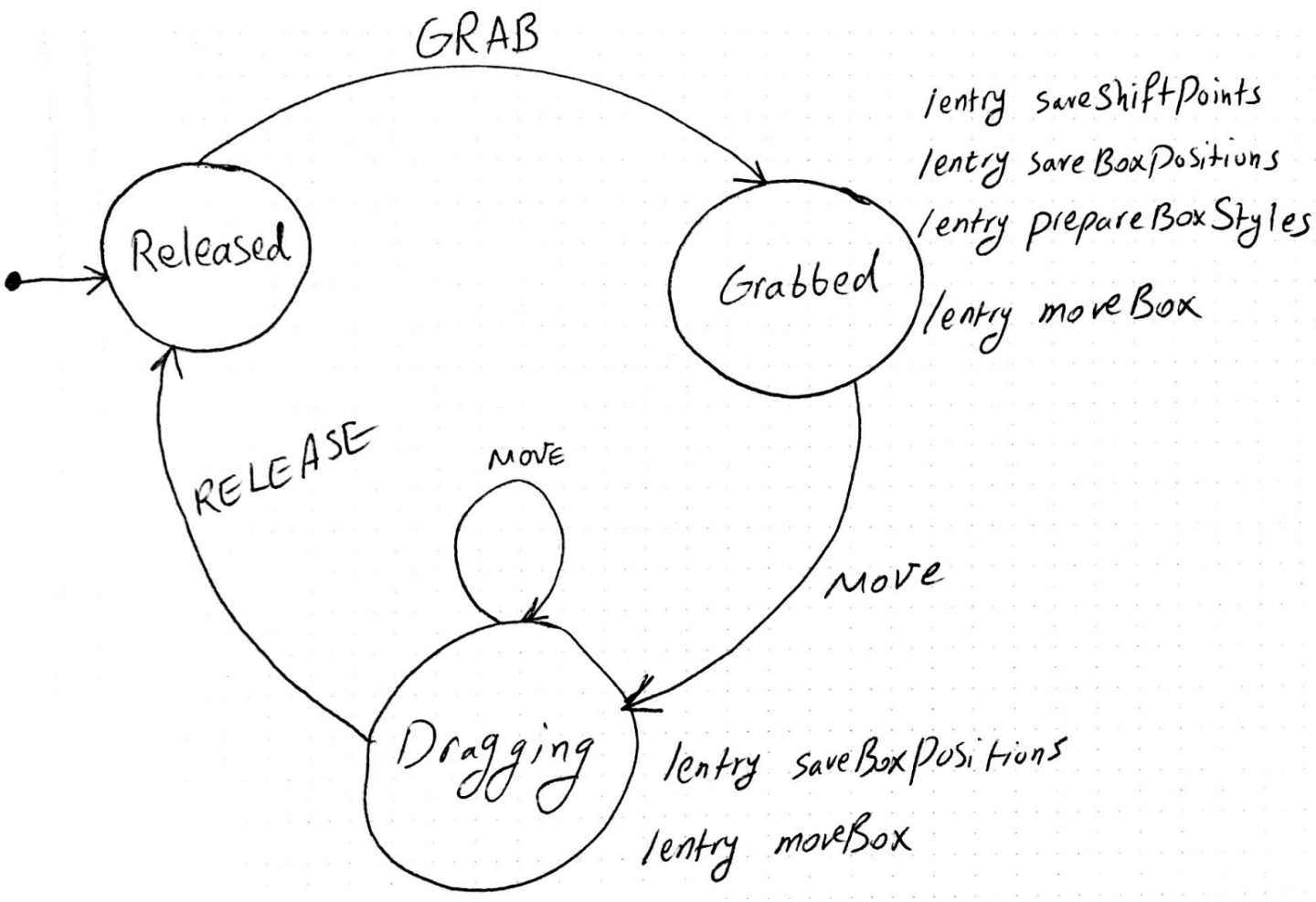
STATECHARTS VISUALIZE LOGIC

GENERATE DIRECTED GRAPH

SHOWCASE STATE PATHS

PATHS CAN BE USED BY QA TEAM TO TEST FOR EDGE CASES

CROSS COMPETENCE TEAMS



STATECHARTS VISUALIZE LOGIC

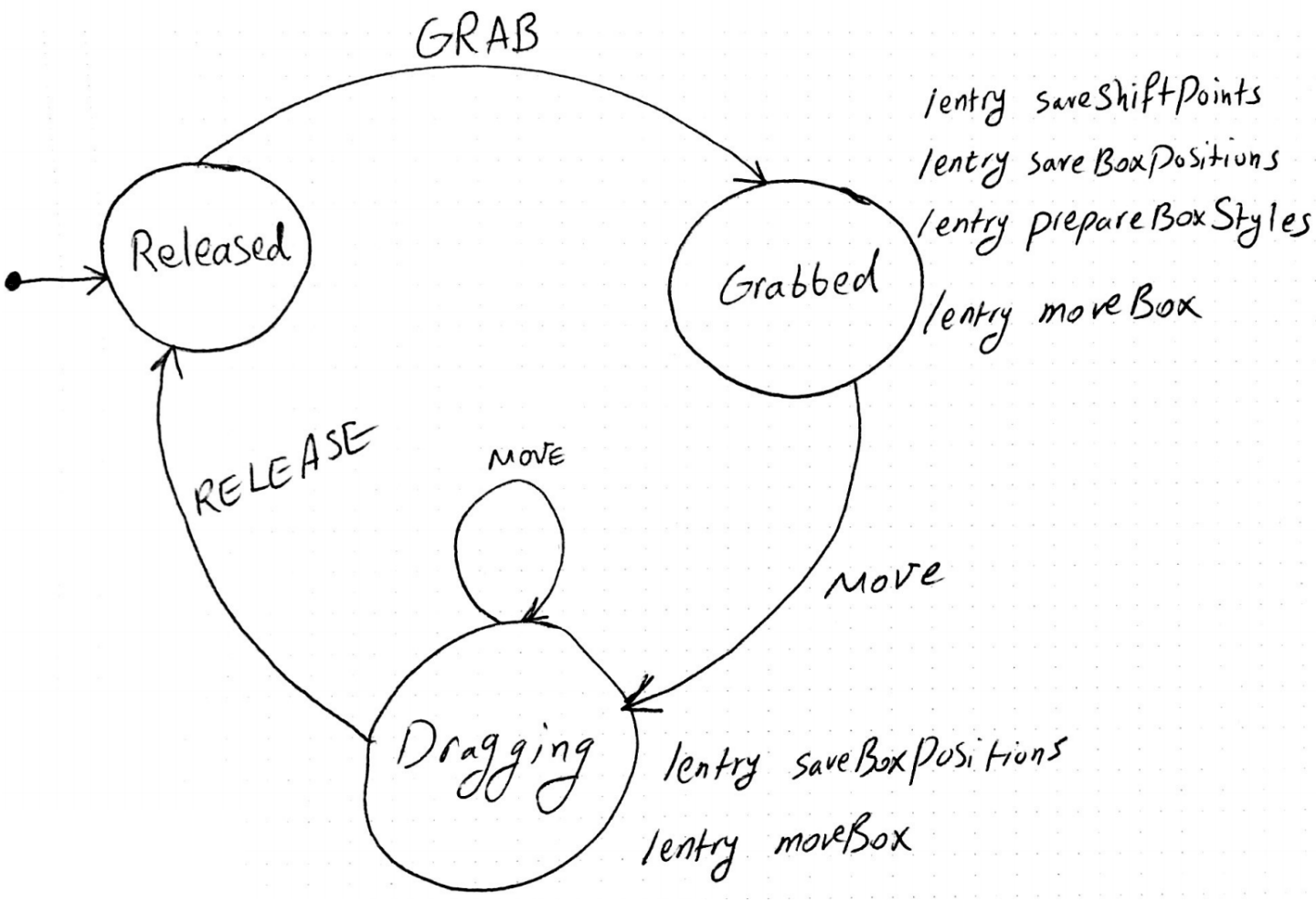
GENERATE DIRECTED GRAPH

SHOWCASE STATE PATHS

PATHS CAN BE USED BY QA TEAM TO TEST FOR EDGE CASES

CROSS COMPETENCE TEAMS

ONBOARDING



STATECHARTS VISUALIZATION IN PULL REQUESTS

Open **fix(gatsby-source-filesystem): Fix timing issue - processing nodes while flushing** #17519

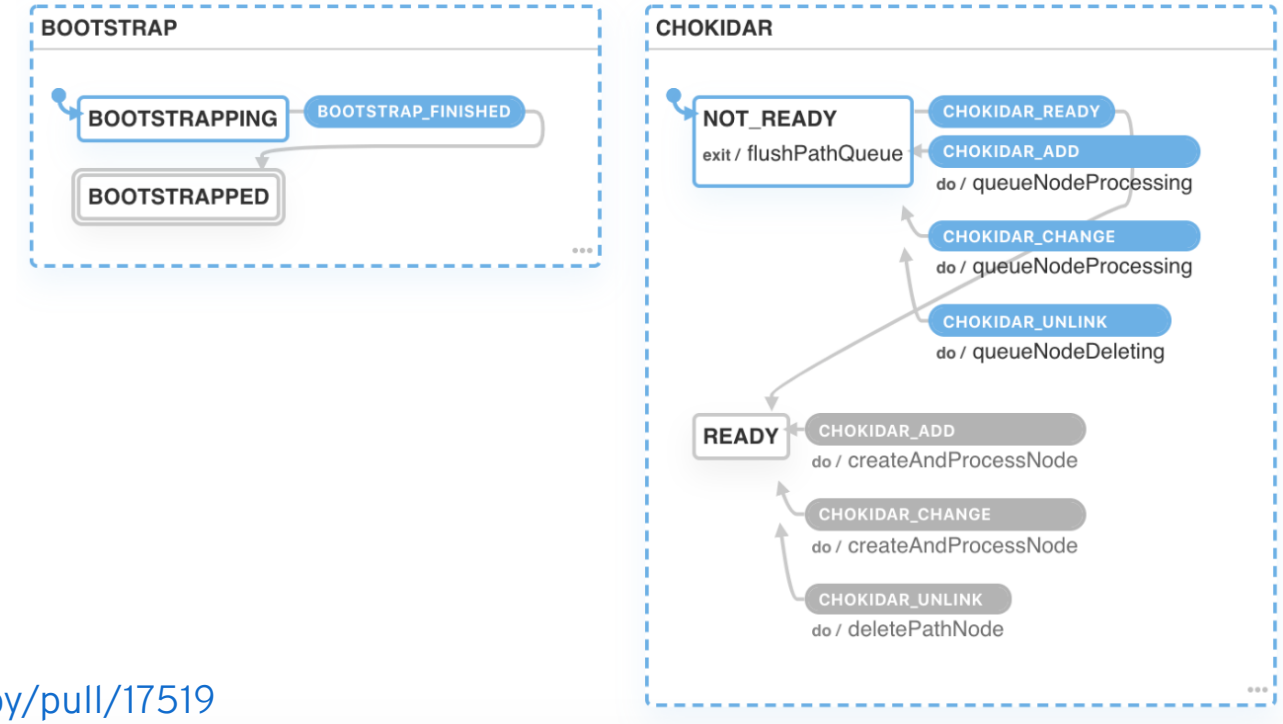
Andarist wants to merge 1 commit into `master` from `fix/gatsby-source-filesyst…`

Followup to #17192 (comment)

Description

This PR aims to solve a minor timing issue. Right now nodes can be created/changed/removed instantly **while** flushing path queue, so even before a returned promise settles. It has been suggested by **@pieh** that this might be a slight problem, although surely an edge case.

Before



muescha

core

pieh

At least 1 approving review is required to merge this pull request.

Assignees

No one assigned

Labels

status: WIP

Projects

None yet

Milestone

No milestone

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

Source: https://github.com/gatsbyjs/gatsby/pull/17519

STATECHARTS DECOUPLE LOGIC FROM IMPLEMENTATION

ABSTRACT DECLARATIVE JSON

```
{
  initial: "A",
  states: {
    A: {},
    B: {}
  }
}
```

IMPLEMENTATION

```
statechart.withConfig({
  actions: {},
  services: {},
  delay: {}
})
```

PLATFORM API

NEXT TIME SOMEONE ASKED HOW HARD CAN IT BE?

NEXT TIME SOMEONE ASKED HOW HARD CAN IT BE?

ANSWER: LET'S DRAW ITS STATECHARTS AND SEE!

Scene #6

RECAP

RECAP

Fig. 36.

RECAP

A NEW MODELING LAYER

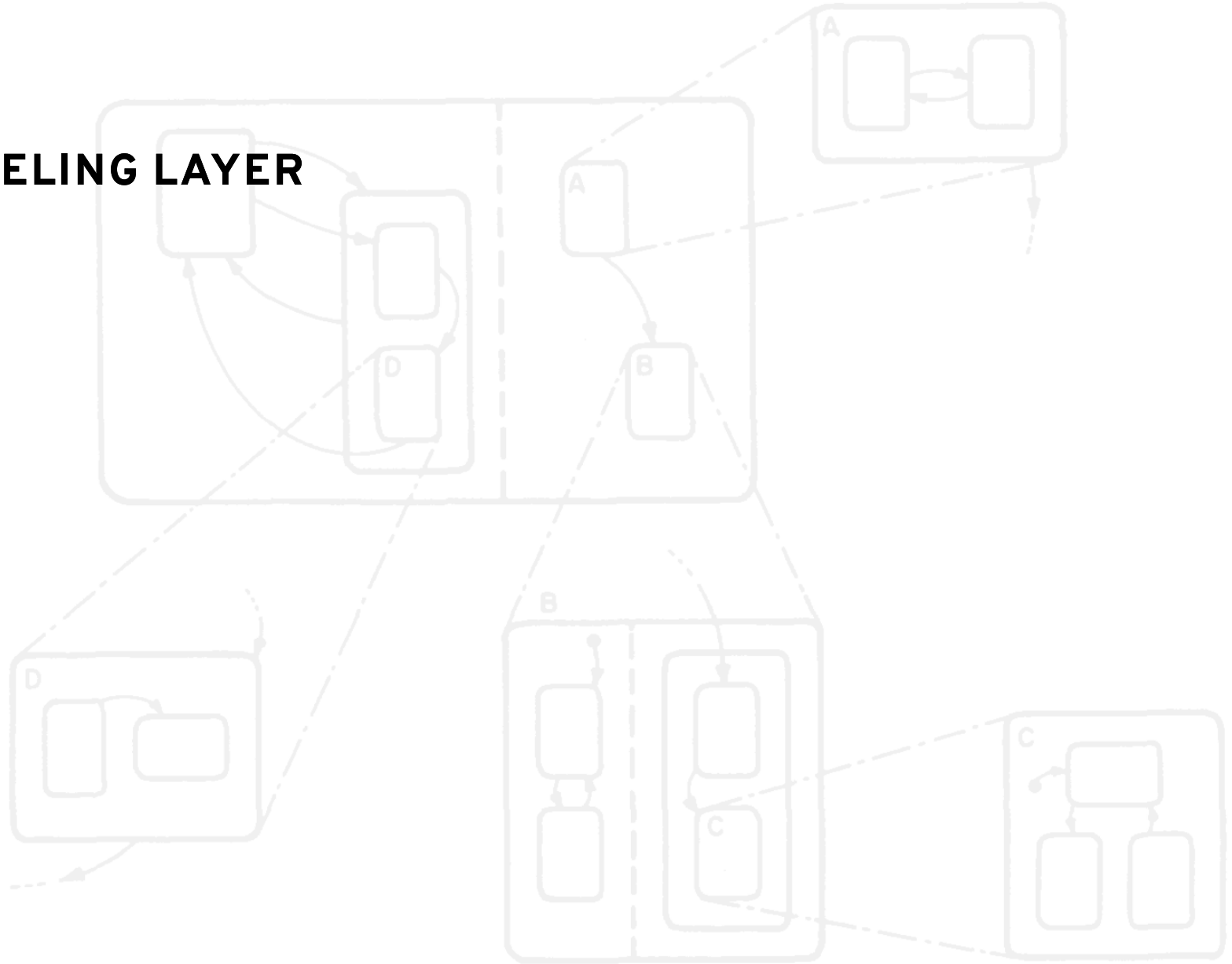


Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

FINITE STATE VS INFINITE STATE

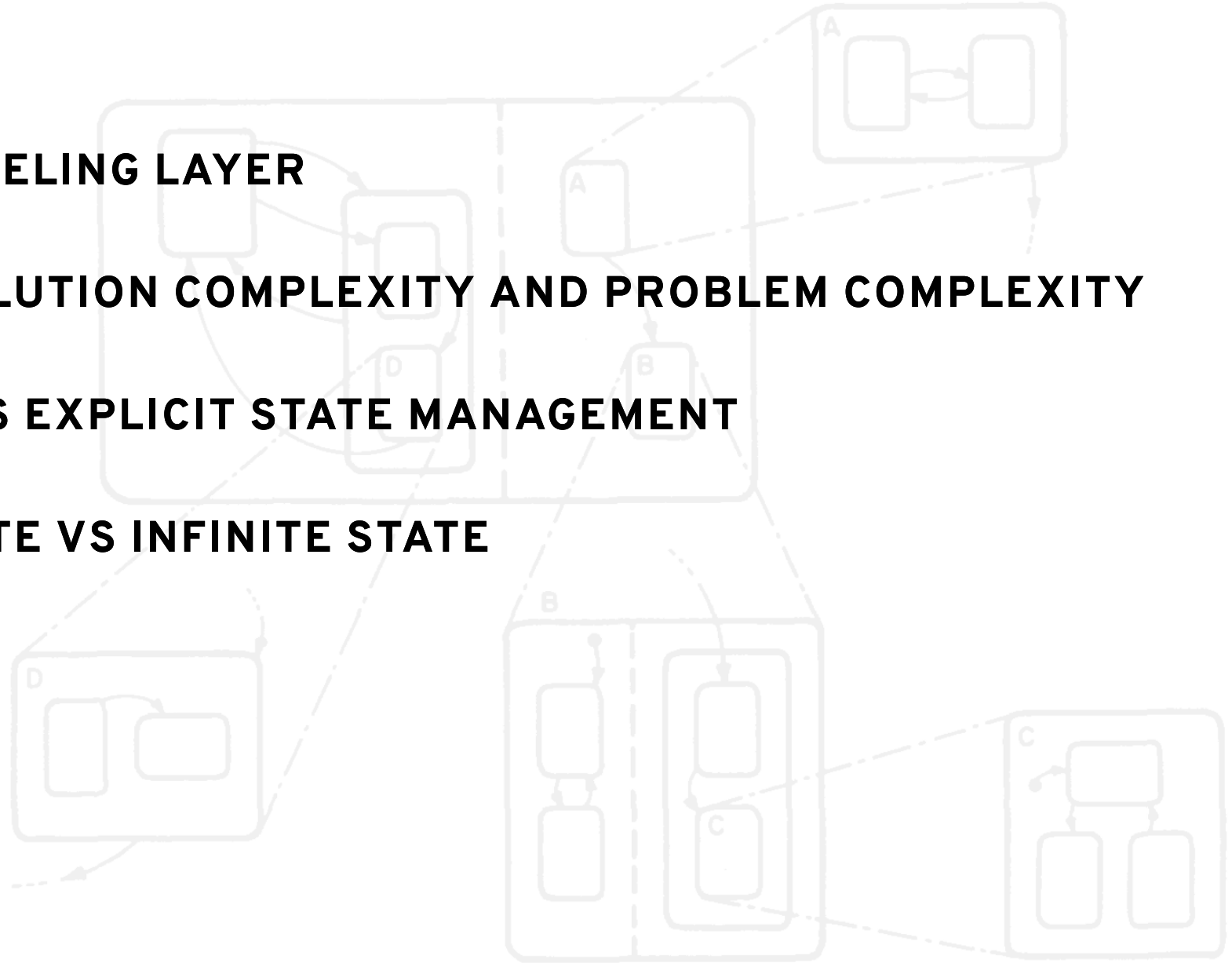


Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

FINITE STATE VS INFINITE STATE

STATECHARTS FOR PRACTICAL MODELING COMPLEX APPLICATIONS

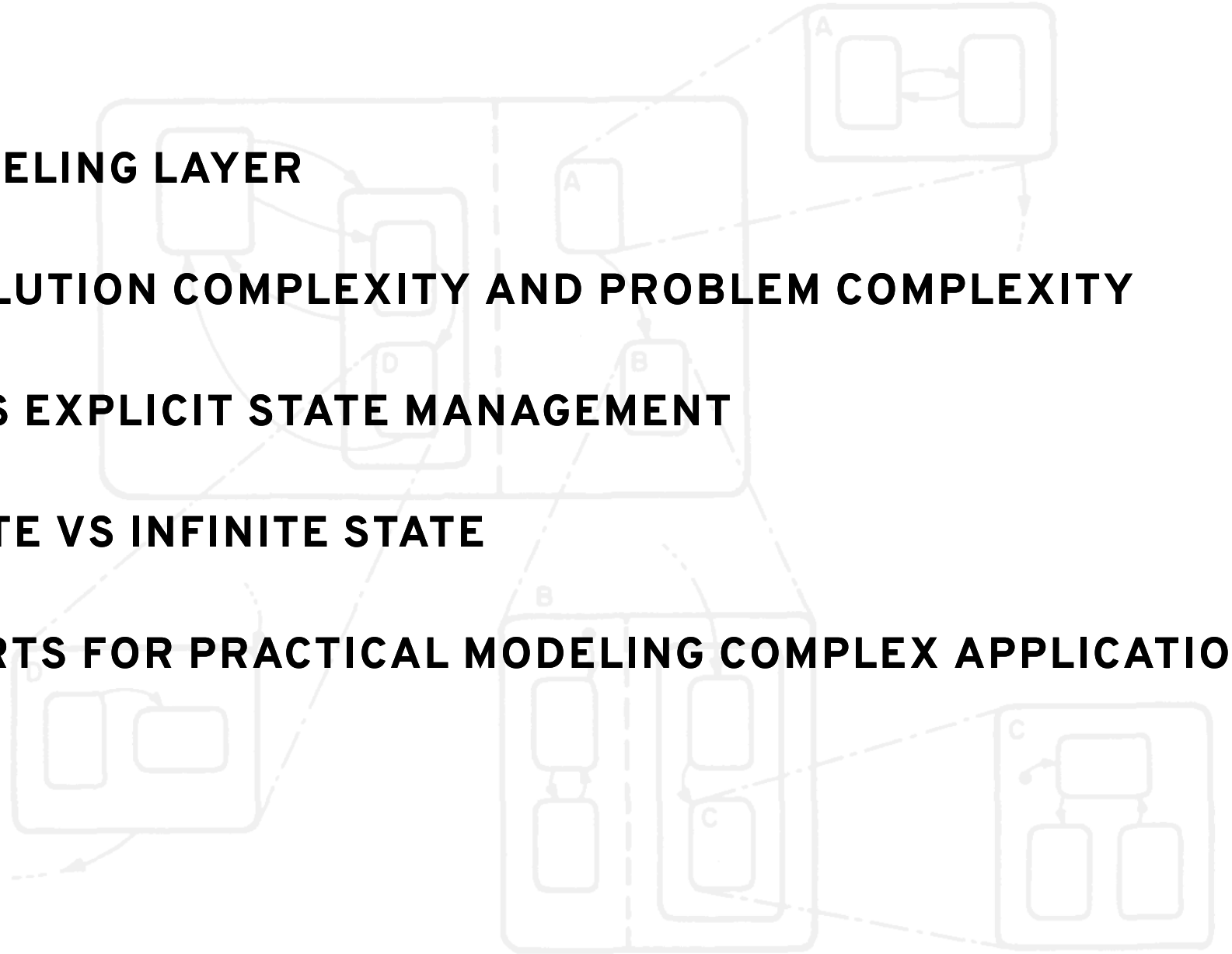


Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

FINITE STATE VS INFINITE STATE

STATECHARTS FOR PRACTICAL MODELING COMPLEX APPLICATIONS

MAKE IMPOSSIBLE STATES, IMPOSSIBLE

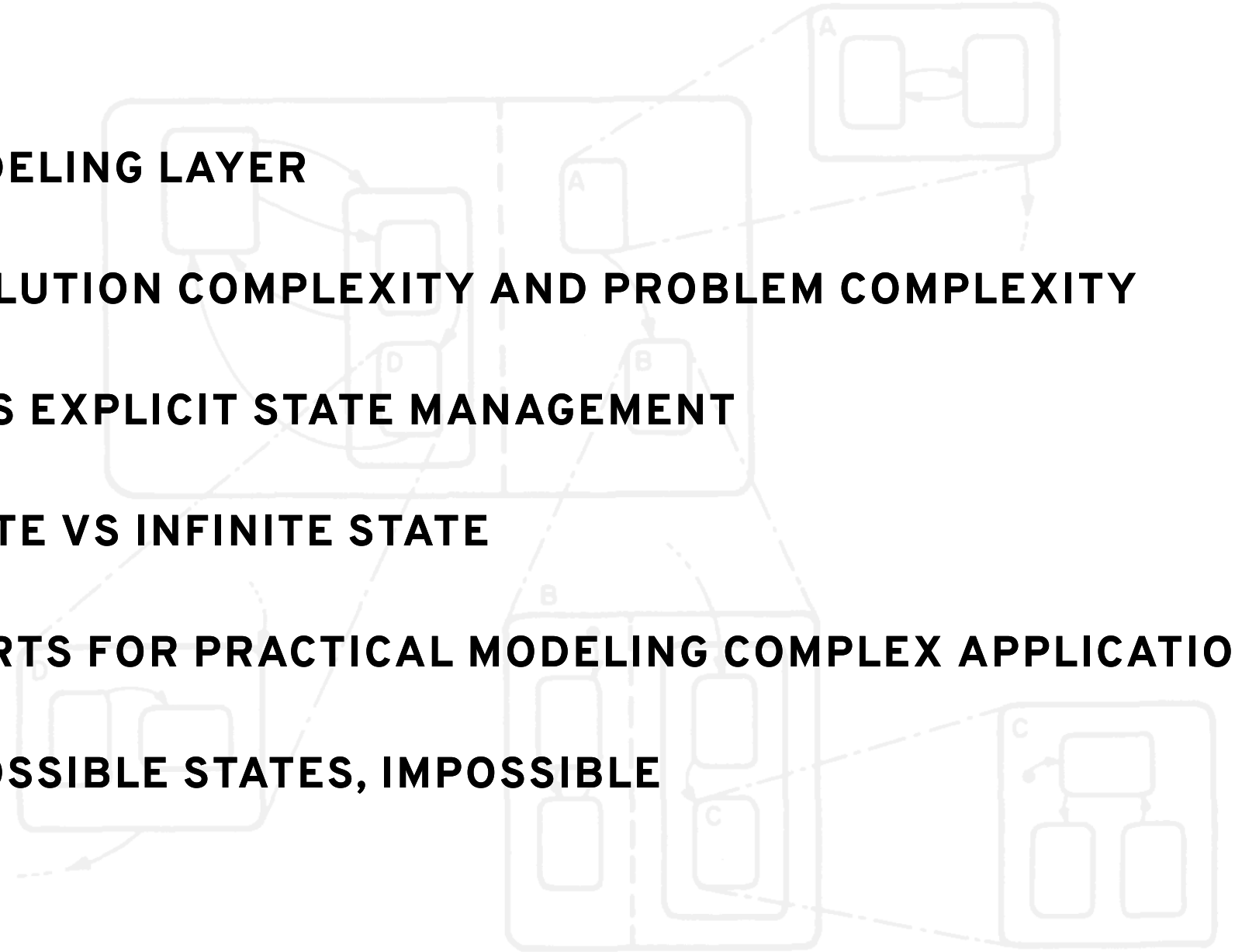


Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

FINITE STATE VS INFINITE STATE

STATECHARTS FOR PRACTICAL MODELING COMPLEX APPLICATIONS

MAKE IMPOSSIBLE STATES, IMPOSSIBLE

AVOID MUTUAL EXCLUSIVITY PROBLEMS

Fig. 36.

RECAP

A NEW MODELING LAYER

LEVELS SOLUTION COMPLEXITY AND PROBLEM COMPLEXITY

IMPLICIT VS EXPLICIT STATE MANAGEMENT

FINITE STATE VS INFINITE STATE

STATECHARTS FOR PRACTICAL MODELING COMPLEX APPLICATIONS

MAKE IMPOSSIBLE STATES, IMPOSSIBLE

AVOID MUTUAL EXCLUSIVITY PROBLEMS

STATECHARTS FOR KNOWLEDGE SHARING AND COMMUNICATION

THINK IN STATES

CHECK THESE OUT

THE WORLD OF STATECHARTS

statecharts.github.io

xstate.js.org

`xstate`

`@xstate/react`

`@xstate/test`

`@xstate/graph`

`@xstate/fsm`

🤖 Robot

thisrobot.life

DON'T LET YOUR SOFTWARE BULLY YOU

THANK YOU!

СПАСИБО!

@farzad_yz

Slides at:

in-pursuit-of-finite-states.netlfiy.com