Items for Conversation Bot APP

Guide to explain how to create a set of questions/answers (items) for the Conversation Bot APP. The set allow to create from a simple linear list of items to a complex tree with logic.

An item set (`items`) is a `JSON Array Object` nested in a Vue.JS mixing.

```
export default {
  data () {
    return {
      items: [
        {
          id: 'your-firstname',
          question: 'What is your first name?',
          type: 'text',
          required: true,
          minlength: 1,
          maxlength: 50,
          preAnswer: 'My first name is ',
          placeholder: '',
          value: '',
          next: 'your-telephone',
          complete: false,
          mode: 'hidden'
        },
        {
          id: 'your-telephone',
          question: 'What's your best contact number? 📞☎',
          type: 'tel',
          required: true,
          preAnswer: 'You can call me at ',
          placeholder: '07#########',
          value: '',
          next: 'your-email',
          complete: false,
          mode: 'hidden'
        },
        {
          id: 'your-email',
          question: 'How about an email?',
          type: 'email',
          required: true,
          preAnswer: 'You can send me an email to ',
          placeholder: 'name@email.com',
          value: '',
          next: null,
          complete: false,
          mode: 'hidden'
        }
      ]
```

```
    }
  }
}
```

Item types

- **Normal**: Typical item that asks a question and is aspecting for an answer.
- **Message**: Item that is just a single message that does not require any answer skipping automatically to the next item.
- **Fork**: Hidden item that will not display anything but contains a logic to fork the flow of the items.

Item elements in depth

```
{
  id: 'must-be-unique-one-word',
  question: 'This is the question text',
  type: 'text',
  options: { yes: 'I sure do! 👍', maybe: 'I'm unsure.', no: 'I definitely
don't.' },
  required: true,
  min: 10,
  max: 1000,
  step: 10,
  minlength: 1,
  maxlength: 50,
  pattern: ''
  preAnswer: 'Displayed before the input tag ',
  postAnswer: ' displayed after the input tag.',
  placeholder: '',
  value: '',
  next: 'next-item-ID',
  complete: false,
  mode: 'hidden'
}
```

Element	Description	Type	Required	Default Value
`id`	Unique ID	string	yes	
`question`	Question	string / function	yes (no: FORK)	
`type`	Item type	string	yes	
`options`	Answer list	JSON	no	
`required`	`input` validation	boolean	no	`false`
`min`	`input` validation	number	no	
`max`	`input` validation	number	no	

Element	Description	Type	Required	Default Value
`step`	`input` validation	number	no	
`minlength`	`input` validation	number	no	
`maxlength`	`input` validation	number	no	
`pattern`	`input` validation	string	no	
`preAnswer`	Pre answer text	string	no	
`postAnswer`	Post answer text	string	no	
`placeholder`	`input` placeholder	string	no	
`value`	Form value	string	yes (normal)	`''` (empty)
`next`	Next item ID	string / function / null	yes	
`complete`	For item logic	boolean	yes	`false`
`mode`	For item logic	string	yes	`'hidden'`

Element `id`

Must be a unique name no spaces. The IDs will be passed to the API integration on form submission. Lunar will required a set of predefined ID as manadtory.

```
{
  id: 'your-firstname',
  ...
}
```

Element `question`

It is the text that will displayed as question. Normally is a predefined single line of text. A question can by dinamically build base on previous answers or other logic.

```
{
  id: 'your-firstname',
  question: 'What is your first name?',
  ...
}
```

or

```
{
  id: 'your-lastname',
  question: () => {
```

```
      return 'Thank you ' + this.getItem('your-firstname').value + '! 👍
What is your last name?'
  },
  ...
}
```

Element `type`

The item type is the one that influence the behaviour of a particular item.

```
{
  ...
  type: 'item-type-value',
  ...
}
```

A **normal item** (question / answer) will have one the following values:

- `text`: generic open text value
- `number`: generic number value
- `date`: generic date value
- `dob`: DOB value
- `tel`: generic telephone value
- `email`: generic email address
- `options`: set of predefined answers (just one answer can be selected)

A **message item** can have just the value `message`.

A **fork item** must be set as `hidden`.

Element `options`

Used only when `type: 'options'`. Here you can set the possible answers and realted values.

```
{
  ...
  options: {
    value1: 'Answer content 1',
    value2: 'Answer content 2'
  },
  ...
}
```

Base on the selection `value1` or `value2` will be stored inside `value` element.

```
{
  ...
  options: {
    yes: 'I sure do! 👍 ',
    maybe: 'I'm unsure.',
    no: 'I definitely don't.'
  },
  ...
}
```

`<input>` validations

The following parameters are just for validation and reflect the `HTML5` equivalent.

- `required`: if a field is required or not (`true` or `false`)
- `min`: is for number or date input types and rappresent the min value allowed (number or date)
- `max`: is for number or date input types and rappresent the max value allowed (number or date)
- `step`: is for number input type only and allows just multiple values of what has been set to
- `minlength`: is for text input type only and rappresent the min number of charachters typed
- `maxlength`: is for text input type only and rappresent the max number of charachters typed
- `pattern`: is a regular expression that will be used to validate the input value

All of those are not mandatory for any item.

Element `preAnswer`

Text to be show before the answer or `<input>`.

Element `postAnswer`

Text to be show after the answer or `<input>`.

Element `placeholder`

Placeholder text for the `<input>`.

Element `value`

It will store the `<input>` value set for a particular item.

Element `next`

After each item has answered the APP move on the new one. Here is where you can say where to go next. `next` can contains the following:

- `next-item-id`: simple item `id` where to go next
- `function`: to apply a logic where to go next based on the value just selected or previous anwer or something else.

- `null`: this item is the last one, so now an API call will be triggered to submit the form.

Move to `your-lastname`:

```
{
  ...
  next: 'your-lastname',
  ...
}
```

Last item:

```
{
  ...
  next: null,
  ...
}
```

Logic based on current answer value:

```
{
  ...
  next: (val) => {
    switch (val) {
      case 'yes':
        return 'your-lastname'
      case 'no':
        return null
    }
  },
  ...
}
```

Logic based on external values (normally used on fork item):

```
{
  ...
  next: () => {
    // Logic: A || B
    if (this.isValue('A', 'yes') || this.isValue('B', 'yes')) {
      return 'ID-C'
    }
    return 'ID-D'
  },
  ...
}
```

Element `complete`

It is for internal use and must always be present and set as below.

```
{
  ...
  complete: false,
  ...
}
```

Element `mode`

It is for internal use and must always be present and set as below.

```
{
  ...
  mode: 'hidden'
}
```

Message item

Item that is just a single message that does not require any answer skipping automatically to the next item.

```
items: [
  {
    id: 'MSG-A',
    question: 'So, let's get you covered!',
    type: 'message',
    next: 'MSG-B',
    complete: false,
    mode: 'hidden'
  },
  {
    id: 'MSG-B',
    question: 'Tell me about yourself...',
    type: 'message',
    next: 'your-firstname',
    complete: false,
    mode: 'hidden'
  },
  ...
]
```

Fork item

Hidden item that will not display anything but contains a logic to fork the flow of the items.

A fork item is used to branch the normal item's flow and potentially create complex item's tree.

Fork items have the `next` element as function with a logic related to previous item's values or other external values.

```
items: [
  ...
  {
    id: 'FORK-PARTNER',
    type: 'hidden',
    next: () => {
      if (this.isValue('partner-included', 'yes')) {
        return 'partner-firstname'
      }
      return 'amount'
    },
    complete: true,
    mode: 'hidden'
  },
  ...
]
```

Normal items

Typical item that asks a question and is aspecting for an answer.

Name

```
items: [
  ...
  {
    id: 'your-firstname',
    question: 'What is your first name?',
    type: 'text',
    required: true,
    minlength: 1,
    maxlength: 50,
    pattern: pattern.name,
    preAnswer: 'My first name is ',
    placeholder: '',
    value: '',
```

```
    next: 'your-lastname',
    complete: false,
    mode: 'hidden'
  },
  ...
}
```

Options

```
items: [
  ...
  {
    id: 'your-smoke-status',
    question: 'Remember, no white lies!',
    type: 'options',
    options: { Y: 'I smoke sometimes 🚬💨', N: 'No, it's not for me.' },
    value: '',
    next: 'your-dob',
    complete: false,
    mode: 'hidden'
  },
  ...
}
```

Date / DOB

```
items: [
  ...
  {
    id: 'your-dob',
    question: () => {
      return 'Now, what is your date of birth, ' + this.getItem('your-
firstname').value + '? 🎂🍾🥂 '
    },
    type: 'date',
    required: true,
    min: this.ageToDbDate(79),
    max: this.ageToDbDate(18),
    preAnswer: 'My birthday is ',
    placeholder: 'dd/mm/yyyy',
    value: '',
    next: 'amount',
    complete: false,
    mode: 'hidden'
  },
  ...
}
```

Number

```
items: [
  ...
  {
    id: 'amount',
    question: 'Brilliant. Just say how much',
    type: 'number',
    min: 10000,
    max: 1000000,
    required: true,
    preAnswer: 'I'd like £ ',
    placeholder: '######',
    postAnswer: ' a month.',
    value: '',
    next: 'your-telephone',
    complete: false,
    mode: 'hidden'
  },
  ...
}
```

Telephone

```
items: [
  ...
  {
    id: 'your-telephone',
    question: 'What's your best contact number? 🇬🇧 📱 📞 ☎',
    type: 'tel',
    required: true,
    pattern: pattern.phonesUK,
    preAnswer: 'My number is ',
    placeholder: '07#########',
    value: '',
    next: 'your-email',
    complete: false,
    mode: 'hidden'
  },
  ...
}
```

Email

```
items: [
  ...
  {
```

```
      id: 'your-email',
      question: 'Do you have an email?',
      type: 'email',
      required: true,
      pattern: pattern.email,
      preAnswer: 'I'm email address is ',
      placeholder: 'name@email.com',
      value: '',
      next: 'Z',
      complete: false,
      mode: 'hidden'
    },
    ...
  }
```