Objects type for theme setting

- Defining an objects type theme setting
  - Required properties
  - Custom Validations
  - Validations for string types
  - Validations for integer and float types
  - Validations for tags, groups and categories types
  - Resolving group membership
  - Nested objects structure
- Setting description and localization

We are introducing a new `type: objects` to the supported types for theme settings which can be used to replace the existing `json_schema` type which we intend to deprecate soon.

Defining an objects type theme setting

To create an objects type theme setting, first define a top level key just like any theme setting which will be used as the setting's name.

```
links: ...
```

Next add the `type`, `default` and `schema` keywords to the setting.

```
links:
  type: objects
  default: []
  schema: ...
```

`type: objects` indicates that this will be an objects type setting while the `default: []` annotation sets the default value of the setting to an empty array. Note that the default value can also be set to an array of objects which we will demonstrate once the `schema` has been defined.

To define the schema, first define the `name` of the schema like so:

```
links:
  type: objects
  default: []
  schema:
    name: link
```

Next, we will add the `properties` keyword to the schema which will allow us to define and validate how each object should look like.

```
links:
  type: objects
  default: []
  schema:
    name: link
    properties:
      name: ...
```

In the example above, we are stating that the `link` object has a `name` property. To define the type of data that is expected, each property needs to define the `type` keyword.

```
links:
  type: objects
  default: []
  schema:
    name: link
    properties:
      name:
        type: string
```

The above schema definition states that the `link` object has a `name` property of type `string` which means that only string values will be accepted for the property. Currently the following types are supported:

- `string`: Value of property is stored as a string.
- `integer`: Value of property is stored as an integer.
- `float`: Value of property is stored as a float.
- `boolean`: Value of property is `true` or `false`.

- `upload`: Value of property is the attachment URL
- `enum`: Value of property must be one of the values defined in the `choices` keyword.

```
links:
  type: objects
  default: []
  schema:
    name: link
    properties:
      name:
        type: enum
        choices:
          - name 1
          - name 2
          - name 3
```

- `categories`: Value of property is an array of valid category ids.
- `groups`: Value of property is an array of valid group ids.
- `tags`: Value of property is an array of valid tag names.
- `icon`: Value of property is the name of a single icon from the Discourse icon set. Selected icons are automatically added to the sprite sheet, so they can be rendered without being registered separately.

With the schema defined, the default value of the setting can now be set by defining a array in yaml like so:

```
links:
  type: objects
  default:
    - name: link 1
      title: link 1 title
    - name: link 2
      title: link 2 title
  schema:
    name: link
    properties:
      name:
        type: string
      title:
        type: string
```

Required properties

All properties defined are optional by default. To mark a property as required, simply annotate the property with `required: true`. A property can also be marked as optional by annotating the property with `required: false`.

```
links:
  type: objects
  default: []
  schema:
    name: link
    properties:
      name:
        type: string
        required: true
      title:
        type: string
        required: false
```

Blank `string`, `datetime` and `icon` values and empty `categories`, `groups` and `tags` lists count as missing: required properties reject them and optional properties skip their validations. `false` counts as set.

Custom Validations

For certain property types, there are built in support for custom validations which can be declared by annotating the property with the `validations` keyword.

```
links:
  type: objects
  default: []
  schema:
    name: link
    properties:
      name:
        type: string
        required: true
        validations:
          min_length: 1
          max_length: 2048
          url: true
```

Validations for `string` types

- `min_length`: Minimum length of the property. Value of the keyword has to be an integer.
- `max_length`: Maximum length of the property Value of the keyword has to be an integer.
- `url`: Validates that the property is a valid URL. Value of the keyword can be `true/false`.

Validations for `integer` and `float` types

- `min`: Minimum value of the property. Value of the keyword has to be an integer.
- `max`: Maximum value of the property. Value of the keyword has to be an integer.

Validations for `tags`, `groups` and `categories` types

- `min`: Minimum number of records for the property. Value of the keyword has to be an integer.
- `max`: Maximum number of records for the property. Value of the keyword has to be an integer.

Resolving group membership

Object settings can resolve `type: groups` properties to a boolean for the current user. This is useful when theme code only needs to know whether the current user is in one of the configured groups, because `currentUser.groups` only includes groups that are visible to the user.

Add `resolve_group_membership: true` to the `groups` property:

```
menu_sections:
  type: objects
  default:
    - name: section 1
      groups:
        - 1
        - 3
  schema:
    name: menu section
    properties:
      name:
        type: string
      groups:
        type: groups
        resolve_group_membership: true
```

The admin UI and stored setting value still use the original `groups` array. In the frontend runtime `settings` object, Discourse removes the group IDs from each object and adds a boolean with the same property name prefixed by `user_in_`:

```
for (const section of settings.menu_sections) {
  if (section.user_in_groups) {
    // User is in at least one selected group for this section.
  }
}
```

This option is only valid on object schema properties with `type: groups`. It also works on nested object schemas and with automatic groups such as `logged_in_users` and `anonymous_users`.

Nested objects structure

An object can also have a property which contains an array of objects. In order to create a nested objects structure, a property can also be annotated with `type: objects` and the associated `schema` definition.

```
sections:
  type: objects
  default:
    - name: section 1
      links:
        - name: link 1
          url: /some/url
        - name: link 2
          url: /some/other/url
  schema:
    name: section
    properties:
      name:
        type: string
        required: true
      links:
        type: objects
        schema:
          name: link
          properties:
            name:
              type: string
            url:
              type: string
```

Setting description and localization

To add a description for the setting in the `en` locale, create a file `locales/en.yml` with the following format given the following objects type theme setting.

```
sections:
  type: objects
  default:
    - name: section 1
      links:
        - name: link 1
          url: /some/url
        - name: link 2
          url: /some/other/url
  schema:
    name: section
    properties:
      name:
        type: string
        required: true
      links:
        type: objects
        schema:
          name: link
          properties:
            name:
              type: string
            url:
              type: string
```

```
en:
  theme_metadata:
    settings:
      sections:
        description: This is a description for the sections theme setting
        schema:
          properties:
            name:
              label: Name
              description: The description for the property
```

```
            links:
              name:
                label: Name
                description: The description for the property
              url:
                label: URL
                description: The description for the property
```

□□□ #2
□ □□□ □□□ 2026-10-31 13:30:44 UTC
□ □□□ □□□ 2026-10-31 13:30:45 UTC