netcompany

Version
1.0

Status
Final

Approver
Jesper Dall-Hansen

Author
Jesper Lund Andersen

KOMBIT

AULA

D0100 - User-Interface Guidelines - Widgets - Stage 2.1

References

Reference	Title	Author	Version
[Aula design guide]	D0100 – User Interface Guidelines	Jesper Lund Andersen	Latest
[Widget Guide]	T0150 – Widget Guide	Lasse Poulsen	Latest

Table of contents

1 Background

To make Aula successful as the communication platform for all parties involved in schools and day care institutions, the system will be able to display user interfaces from other systems alongside the relevant communication tools within Aula.

Such user interfaces from other systems are referred to as Widgets, and this guide provides guidelines for the user-interface of a Widget to be used in Aula.

Examples of possible Aula Widgets:

- Widgets from the individual Municipalitys Learning Platform (e.g. Homework or links to digital learning tools)
- Widget from absence registration solution to register student absence
- Widgets for file sharing solutions (Google and Microsoft)

Aula has a Widget Governance Board, responsible for approving both which Widget Suppliers are allowed to access the Widget test environment and for approving the developed Widgets before they are put into the production environment.

This guide is supplemented by the [Widget Guide] that explains technical and governance aspects of widgets. Furthermore, the document is heavily influenced by [Aula design guide] which dictates the look and feel of Aula.

The principles and guidelines in this document should be considered as recommendations to the Widget Supplier to provide the user with the best possible user experience.

Please note that this document is intended to be transformed into of a public-facing website, e.g. designmanual.aula.dk, where design guidelines is maintained and is publicly available. This document is updated continuously throughout design and details are added and refined.

1.1 Target group

This guide is primarily intended for developers responsible for developing Widgets for Aula. Readers are assumed to have knowledge of web development, in particular HTML, JavaScript and CSS.

2 General principles for the user interface

This section describes the overall principles that holds for the user interface. The separate elements will be described in details in the following sections.

While some of the more concrete descriptions in this document is clearly written with a specific platform in mind – e.g. web, but the principles and most other descriptions will be applicable for both app and web.

The main principles for widgets naturally have to support the general principles for the user interface; **Provide an overview, Simplify, Recognizable and User-centric**. These principles are elaborated as:

- **Provide an overview**: The user interface should provide the user with the relevant information. Information overflow is one the major pain points of the current systems, and Aula will remedy that.
- **Simplify**: Aula caters to a vast and divers user group which is cause for a great deal of business- and technical complexity. That is why Aula seek to keep all functionality in the user interface as simple as possible, which yields more reusability and fewer functions and concepts for the user to learn and relate to.
- **Recognizable**: Aula's user interface draws inspiration from a lot of established communication platforms and products, e.g. Facebook for messaging and Outlook for advanced calendar. This is because a great deal of what makes a user interface experience that feels "intuitive", is interacting in a way that is consistent to how the user is used to interact with the system.
- **User-centric**: Aula is designed by putting the user first. This means saving clicks/touches and knowing which tasks are done frequently and which ones done rarely and leveraging this in the design.

Widgets are an integrated part of Aula and can be used in the following ways across web and app:

- As an independent menu in the main navigation
- As a part of the context of a module
- As a part of a group dashboard showed as an overlay

Even though there system wise is a distinction between a module and a widget, there is no need for the users having to make this distinction. This would only disturb the users overview and deviate from the general principles for the user interface.

Widgets being an integrated part of Aula supports one of the main principles for the user interface of providing the users with an overview with the relevant information. Because widgets can be integrated in Aula, the users can click through the specific widgets without ever leaving Aula which gives the users an overall and coherent user experience and saves clicks/touches.

Furthermore, the possibility of showing a module alongside with a widget gives the user a better overview because the different types of information in Aula is better understood in a wider context. Section 3 elaborates how Widgets are intended to behave in the above context.

See [Widget Guide] chapter 2 for an overview on dashboards, modules and widgets and how they cooperate in Aula.

2.1 Standard functions

The user interface apply standard functions as listed below

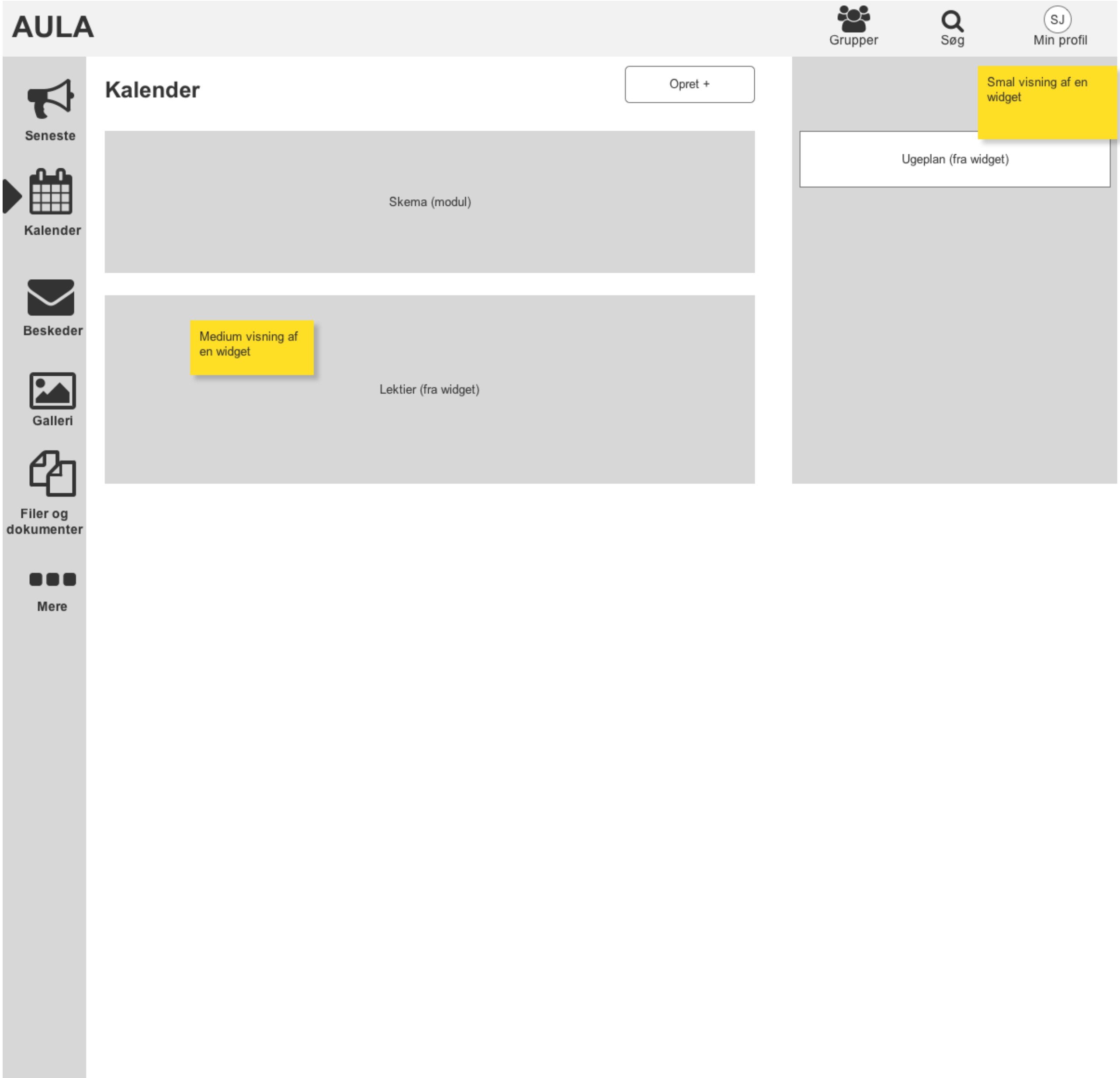
- Modal boxes: should be avoided in Widgets.
- Overlays: Are used to present a group dashboard.
- Enter: Activates the button in focus
- Tab: Moves the curser between elements of the page
- Arrows: Moves the cursor within the field
- Delete: Always ask the user to confirm. Only when confirmed, things will be deleted.
- Information and error messages are always close to where errors are detected
- Help: Using an icon nearby the functionality

3 Form factors

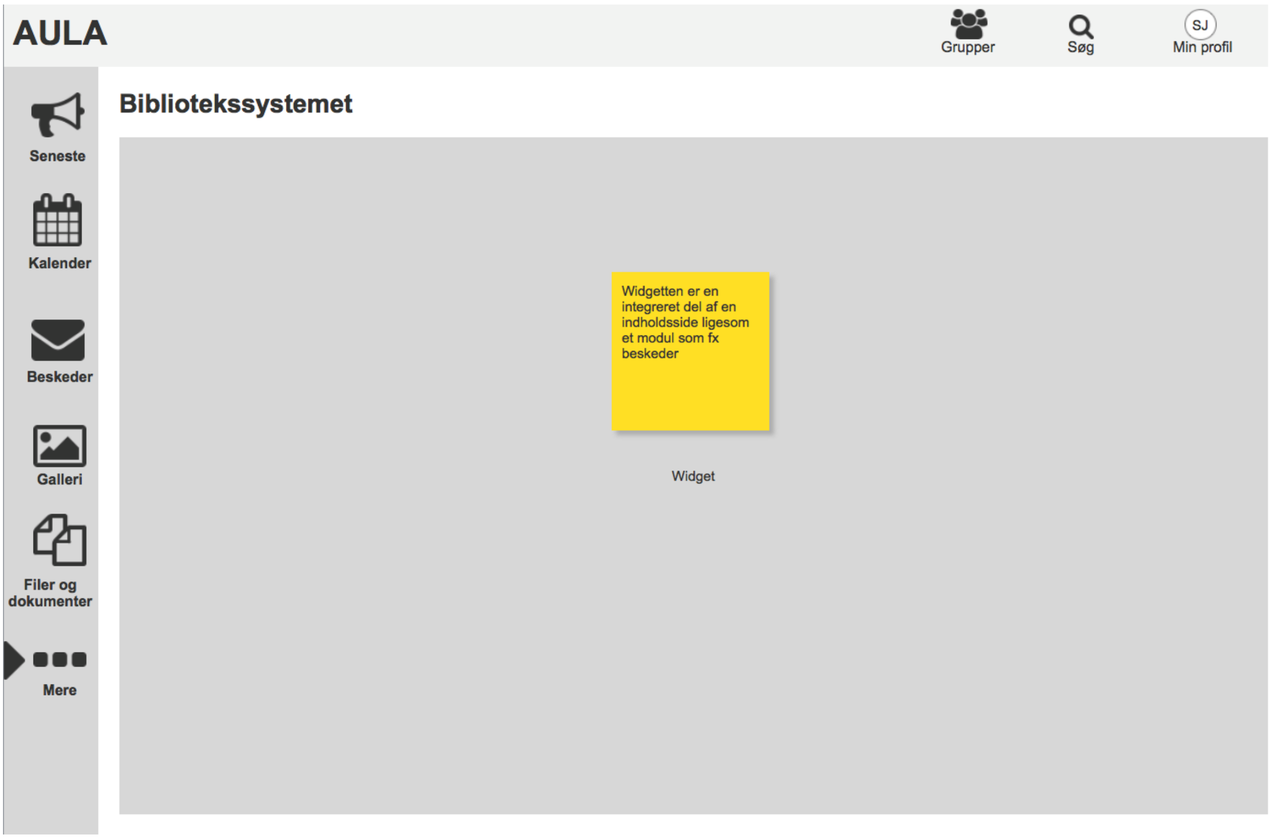
Widgets are expected to be responsive which allowing to scale up the elements as the viewpoints changes. There are defined 3 different breakpoints which are specified in the following:

- A full page version, which can be used only on web
- A medium version, which can be used only on web
- A narrow/small version, which can be used on both web and app.

In the following the different types will be elaborated with examples. See [Widget Guide] chapter 2.3 for a technical perspective on supporting form factors in Aula.



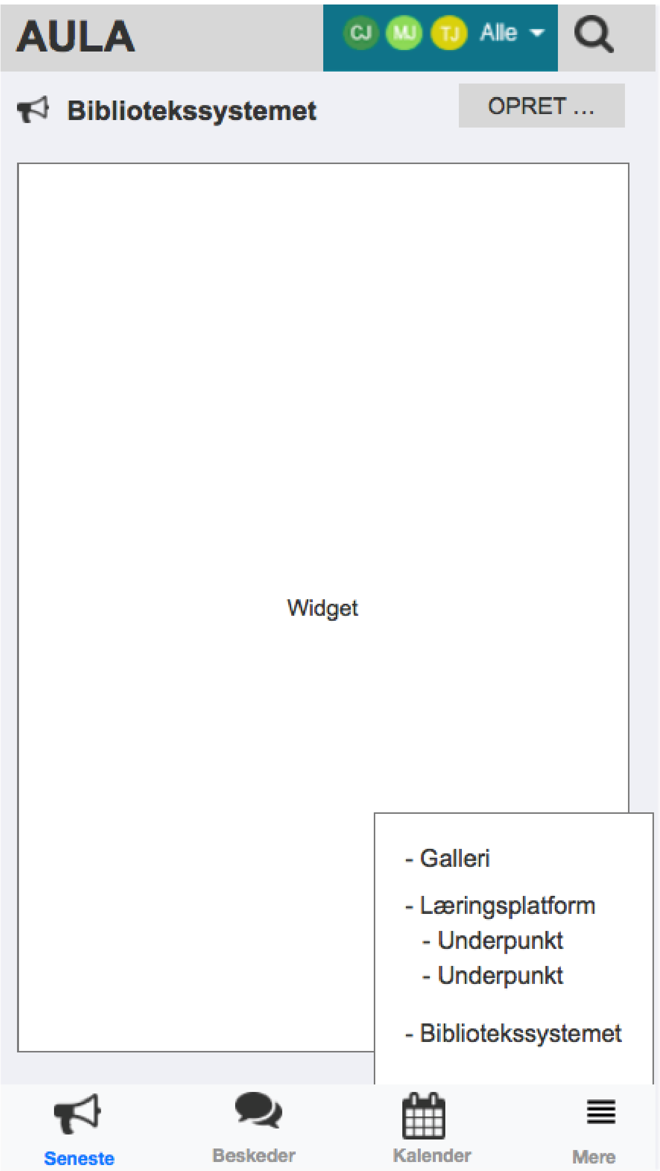
Example 1 demonstrates that widgets can be showed in context with a module in two different sizes. Eg. the module ‘Kalender’ is showed alongside with the widget ‘Lektier in a medium version in the main content field and the widget ‘ugeplaner’ is showed in a small version as a part of the the right column..



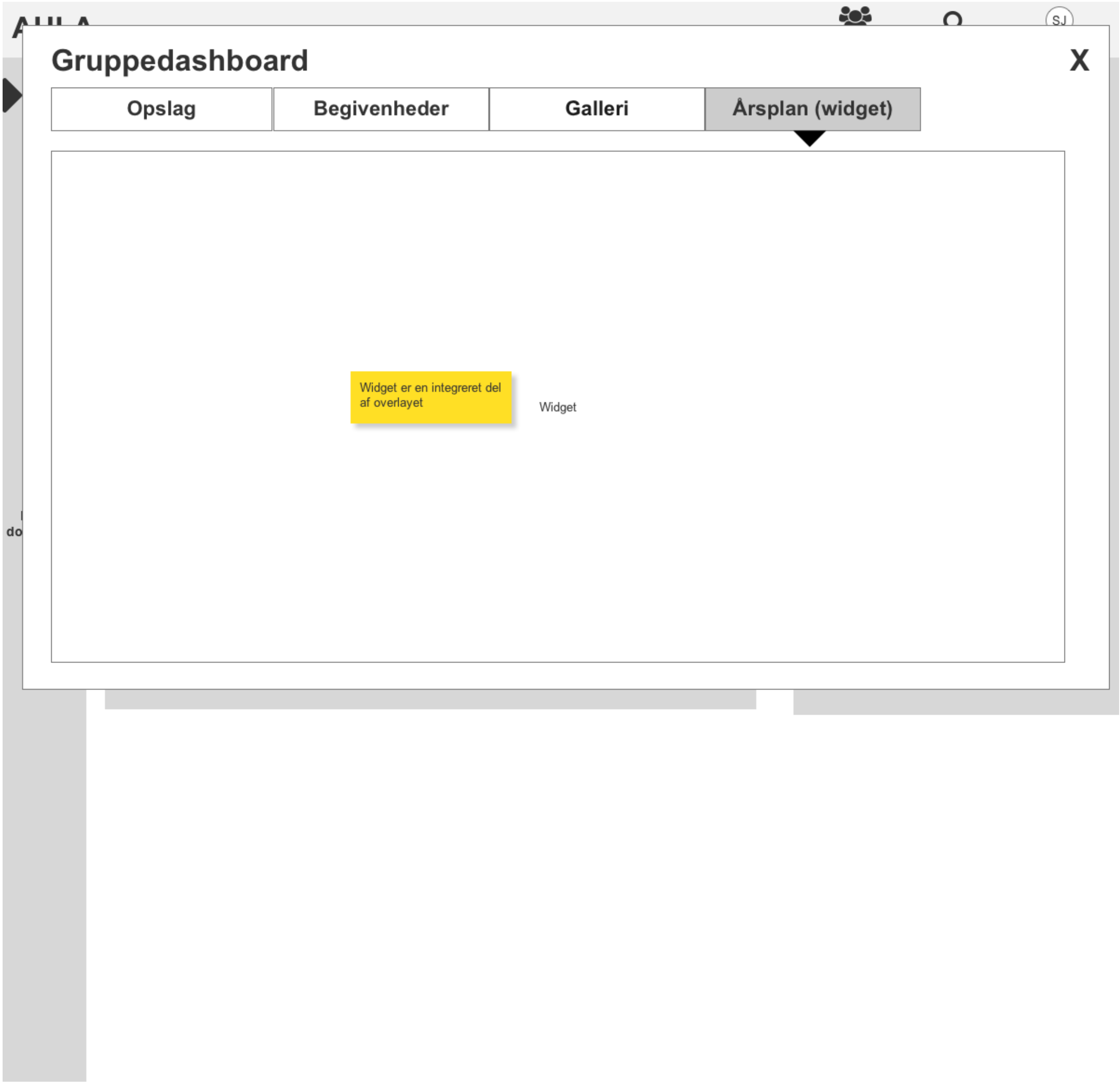
Bibliotekssystemet

Widgetten er en integreret del af en indholdsside ligesom et modul som fx beskeder

Example 2 demonstrates a widget eg. ‘Bibliotekssystemet’ being an independent part of the main navigation. It is an integrated part of a content page like a module eg. ‘Beskeder and is showed in a full page version.



Example 3 demonstrates how the widget ‘Bibliotekssystemet from example 2 is showed in the app in the small version.



Example 4 demonstrates how a widget eg. ‘Årsplan’ is an integrated part of an overlay. And is showed in the full page version.

4 Accessibility considerations

Aula follows the principles and guidelines supplied by W3C in order to provide a truly accessible solution. This is described by the Web Content Accessibility Guidelines (WCAG 2.0).

Any Widget supplier is expected to adhere to the same guidelines and principles described by WCAG 2.0 and this document specifically.

Section 4.1.1 describes the overall principles supplied by W3C, section 4.1.2 describes the guidelines for fulfilling those principles, as well as how these will be fulfilled in Aula. The remaining sections will cover more in depth detailed fulfillment of selected principles.

4.1 WCAG AA

WCAG AA covers a wide range of recommendations for making web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more usable to users in general. WCAG 2.0 success criteria are written as testable statements that are not technology-specific.

To provide accessibility to the necessary extend, this document will shortly wrap up the most important points from WCAG, such that developers will have the best possible prerequisites to make qualified decisions to ensure accessibility.

4.1.1 Principles of WCAG

At the top are four principles that provide the foundation for Web accessibility: *perceivable*, *operable*, *understandable*, and *robust*. The principles are describes in more details below:

1. Perceivable - Information and user interface components must be presentable to users in ways they can perceive.
    - This means that users must be able to perceive the information being presented (it can't be invisible to all of their senses)
2. Operable - User interface components and navigation must be operable.
    - This means that users must be able to operate the interface (the interface cannot require interaction that a user cannot perform)
3. Understandable - Information and the operation of user interface must be understandable.
    - This means that users must be able to understand the information as well as the operation of the user interface (the content or operation cannot be beyond their understanding)
4. Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.
    - This means that users must be able to access the content as technologies advance (as technologies and user agents evolve, the content should remain accessible)

If any of these are not true, users with disabilities will not be able to use the Web. It is necessary for the developers that they relate to these principles when user interface components are developed in the solution.

4.1.2 Guidelines

Under each principle there is a list of guidelines that address the principle. There are a total of 12 guidelines. One of the key objectives of the guidelines is to ensure that content is directly accessible to as many people as possible, and capable of being re-presented in different forms to match different peoples' sensory, physical and cognitive abilities.

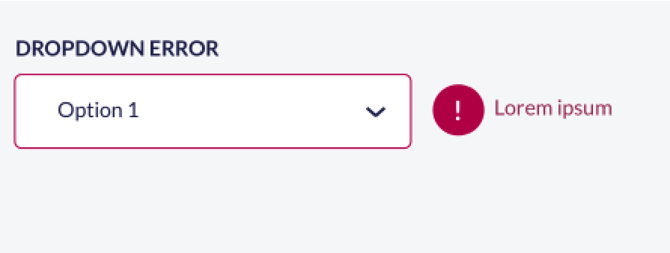
The guidelines are not testable, but represent the frame of overall objectives, that can help the developers to understand the success criteria and implement the techniques better.

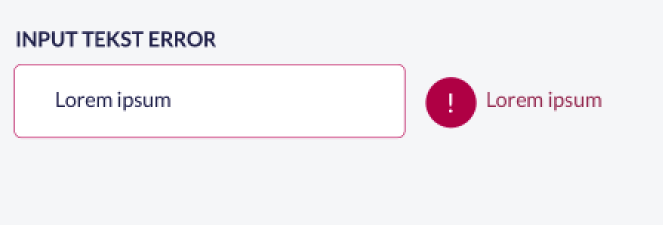
The guidelines are each related to Aula with examples and considerations. Any Widget is expected to support WCAG AA on the same level as Aula.

A wide range of the guidelines that WCAG has proposed, is considered in the markup presents in the html guide. By following the html guide many of the guidelines will implicitly be followed. The guidelines grouped by the overall principle are listed below:

- Perceivable
    - Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.
        - In Aula all images and non-text content should have an alternative text title/description, e.g. via an ALT-parameter in <IMG>-tags, and its equivalent on mobile app clients.
    - Provide alternatives for time-based media.

        - All non-user-generated video and audio media should be supplied with alternative .
    - Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
        - In Aula structure and navigation should be determined either programmatically or via text.
    - Make it easier for users to see and hear content including separating foreground from background.
        - In addition to color contrast as elaborated in section 5.1, text should be be resized up to 200% without the loss of functionality – partially assisted by a responsive design pattern.
- Operable
    - Make all functionality available from a keyboard.
        - Aula can be operated through a keyboard interface without requiring specific timings for individual keystrokes.
    - Provide users enough time to read and use content.
        - For accessibility purposes widgets should avoid having time-limited displays.
    - Do not design content in a way that is known to cause seizures.
        - In Aula graphical elements should use calm and serene colors, and flashing colors should be avoided. This means no more than three flashes in a second.
    - Provide ways to help users navigate, find content, and determine where they are.
        - In Aula content can be navigated through several ways. If the content is referenced, it is linked, and relevant content can be found through search.
- Understandable
    - Make text content readable and understandable.
        - The (human) language of the page should be part of the <HEAD>-section of the page on web, and part of the locale on the App. Default is Danish.
    - Make Web pages appear and operate in predictable ways.
        - Navigational- and functional elements are consistent throughout Aula-design, and part of the IA Concept. E.g. the main navigational menu on the left, and consistent use file pickers.
    - Help users avoid and correct mistakes.
        - Error messages will be presented in context in a easily understandable format. See section 6.5 for an elaboration of this.





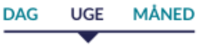
- Robust
    - Maximize compatibility with current and future user agents, including assistive technologies.
        - Aula will generate valid HTML5

Use visual means to indicate action/response

(Level A)

Colour is not the only visual tool, which is used to communicate information, indicate action, ask for response or separate a visual element.

Example

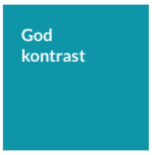


Color contrast ratio min. 4,5:1

(Level AA)

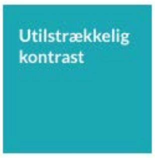
The visual presentation of text and photos of text should have a contrast ratio of at least 4,5:1.

Example



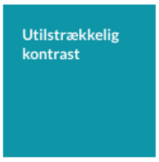
Good contrast

14pt, bold



Insufficient contrast

14pt, bold – Too bright



Insufficient contrast

12pt, bold – small text

Except in following situations:

Large font: Text and photos of text in large font has a contrast ratio of at least 3:1; Large font: 18pt or 14pt bold.

Non-important: Text or photos of text, which takes part in an inactive user interface component, only appears as accessory, is not visible for some, or which is a part of an image, containing other important visual content, is not a part of the contrast requirement.

Logotypes: Text, which is a part of a logo or brand name is not subject to the contrast requirement.

4.1.3 Graphics – and other resources of non-textual character

All graphics should have a description of the function of the graphics in the alt attribute such that a screen reader can read these.

- The alt attribute is used to include a short description of the graphics or the picture
- Graphs and diagrams should also be accompanied by a descriptive text in the alt attribute like above described

4.1.4 Form-elements

4.1.4.1 Labels on form elements

Labels or instructions are provided when content requires user input such as checkboxes or radio buttons.

Instructions for form-elements should be clickable by using the tag <LABEL>, such that the clickable area is bigger and thereby more accessible.

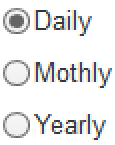
4.1.4.2 Radio buttons

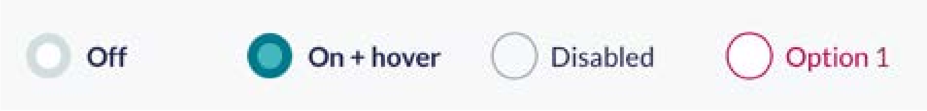
Radio buttons are used when there are two or more choices that are mutually excluding. This ensures that the user can only choose one option.

Note that on Apps, the OS may have a different representation of the radio buttons, and in that case that will be used.

A click in a radio button cannot lead to a reload of the page or any other action without a accompanied click on a button. Radio buttons can therefore not be used to enhance visibility on certain parts of the page.

Multiple radio button should be displayed vertically underneath each other as shown below:



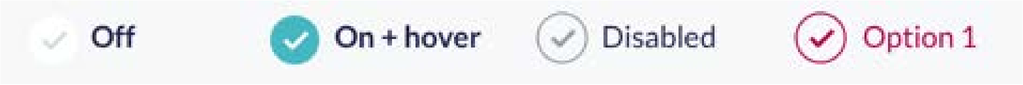


4.1.4.3 Check boxes

Check boxes are used when there is a list of available options where the user can choose 0 or more of the options. Check boxes are therefore functioning independent of each other.

A click in a check box cannot cause a reload of the page or any other action without being accompanied by a click on a button. Check boxes can therefore not be used to enhance certain parts of the page.

A multiple number of check boxes are placed vertically underneath each other like the example from the radio buttons.



4.1.4.4 Keyboard-accessibility

The site is made keyboard accessible by having a logical tab sequence for desktop. The expected detailed tab sequence should follow the listed priority

1. top menu

2. left menu

3. Content area

4. right menu

Insite each area the sequence should be from top to bottom - left to right.

The tab sequence starts in the field that has focus. Unless otherwise stated, this will be the search bar.

4.1.5 Support for assistive tools

Aula specifically supports the following assistive technology on the platforms supported by Aula and the specific technology:

- AppWriter from WizKids
- CD-Ord from MV Nordics
- IntoWords from MV Nordics

Any widget is expected to have the same level of support for the above tools as Aula.

5 Design elements

This chapter provides the guidelines for design elements in Widget user interface – such as colors and typography.

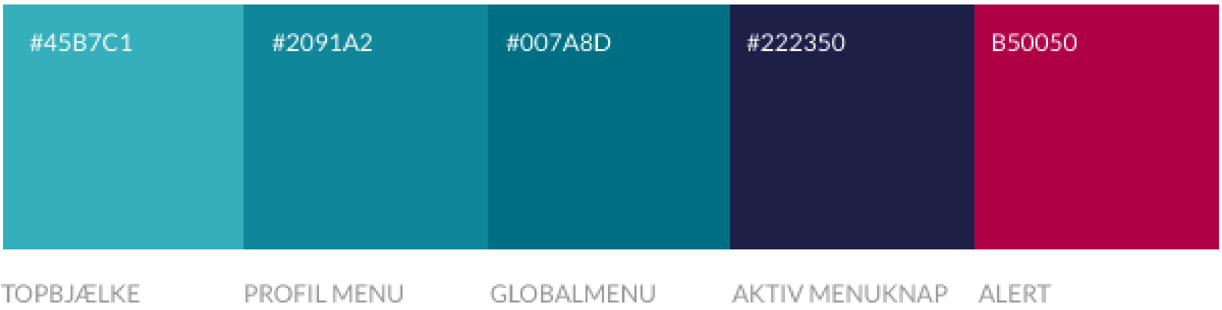
5.1 Colors

5.1.1 Standard color palette for the target groups

Contrast level WCAG AA

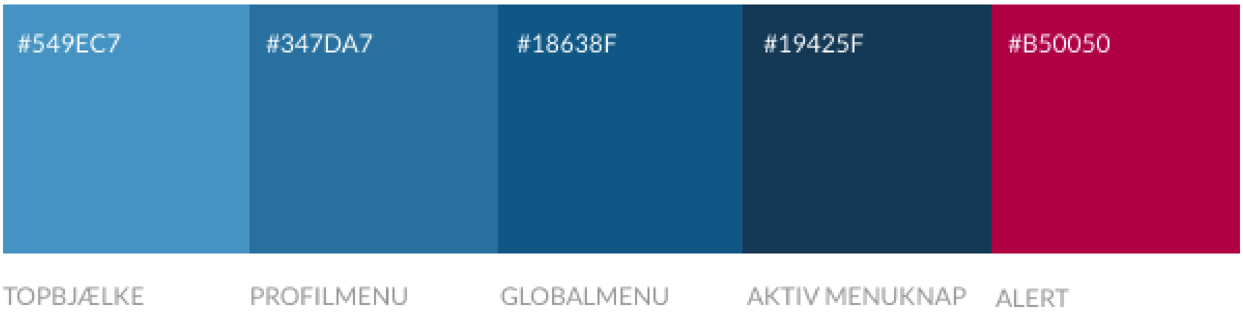
Staff

Green: Calmed, balanced, relaxed, natural, growth

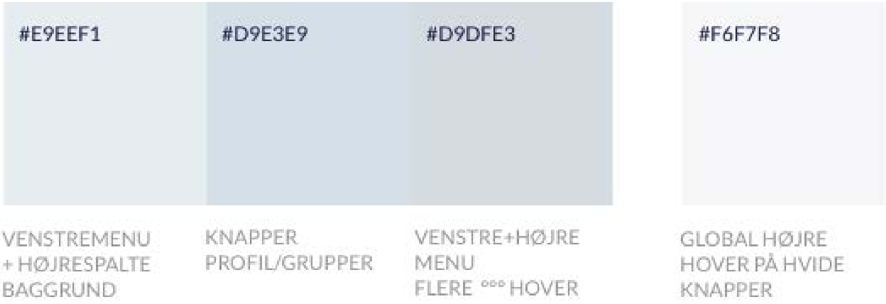


Parents/Children

Blue: Intelligence, trust, cooperation, sincerity, order, harmony, wisdom, reliability, stability

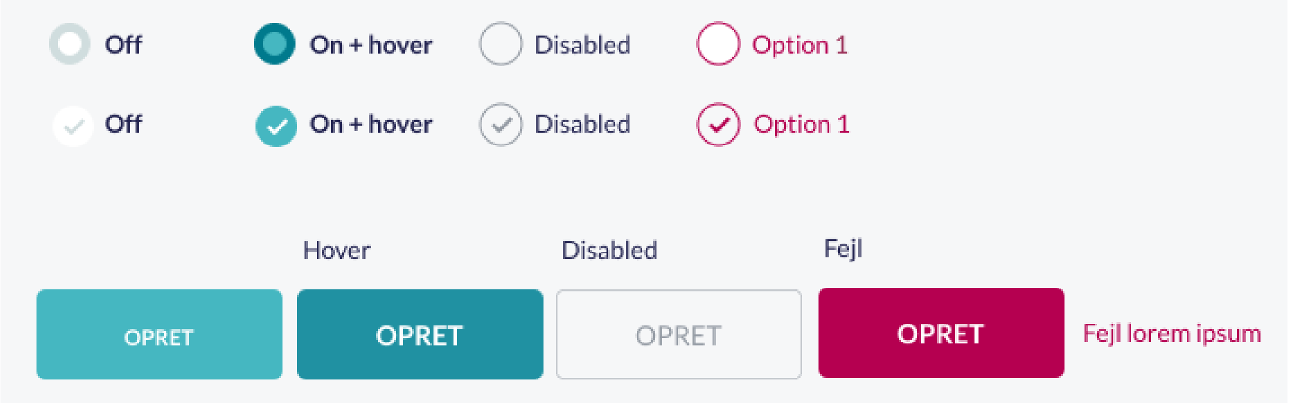


Background color across palettes

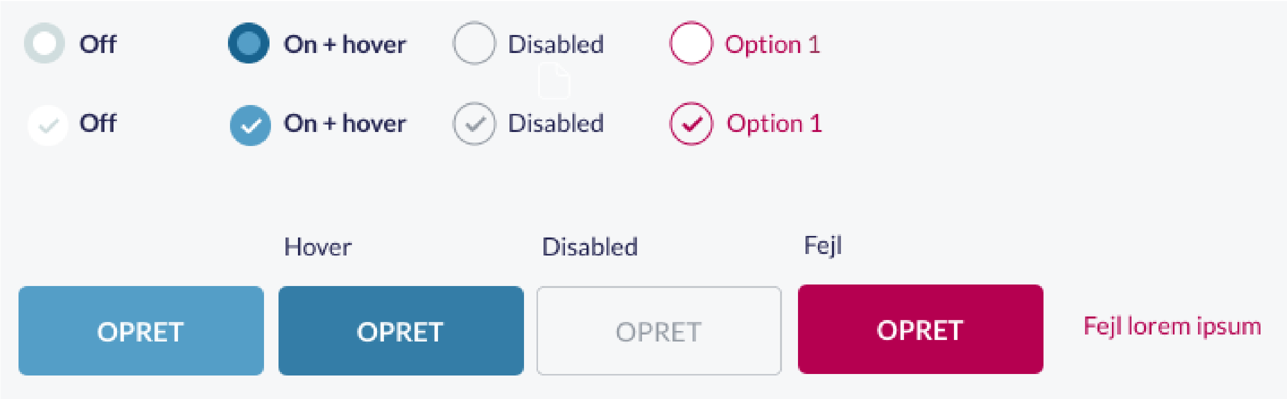

#E9EEF1
#D9E3E9
#D9DFE3
#F6F7F8
VENSTREMENU + HØJRESPALTE BAGGRUND
KNAPPER PROFIL/GRUPPER
VENSTRE+HØJRE MENU FLERE °°° HOVER
GLOBAL HØJRE HOVER PÅ HVIDE KNAPPER

Example of usage

Green palette



Blue palette

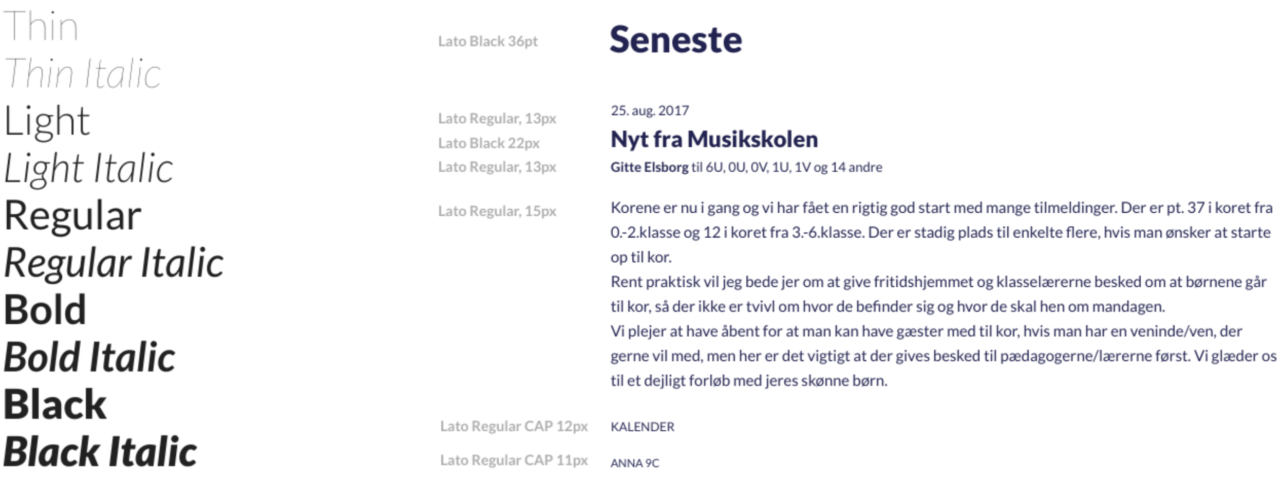


5.2 Typography

5.2.1 Fonts

The font Lato is designed especially for web. The semi-round forms give Lato a warm expression, and the strong structure gives stability and reliability.

It is sharp and easy to ready in small sizes, while it gives a distinctive and soft expression in large font size (Lato Black).



Thin
Thin Italic
Light
Light Italic
Regular
Regular Italic
Bold
Bold Italic
Black
Black Italic

Lato Black 36pt — **Seneste**

Lato Regular, 13px — 25. aug. 2017
Lato Black 22px — **Nyt fra Musikskolen**
Lato Regular, 13px — **Gitte Elsborg** til 6U, 0U, 0V, 1U, 1V og 14 andre

Lato Regular, 15px — Korene er nu i gang og vi har fået en rigtig god start med mange tilmeldinger. Der er pt. 37 i koret fra 0.-2.klasse og 12 i koret fra 3.-6.klasse. Der er stadig plads til enkelte flere, hvis man ønsker at starte op til kor.
Rent praktisk vil jeg bede jer om at give fritidshjemmet og klasselærerne besked om at børnene går til kor, så der ikke er tvivl om hvor de befinder sig og hvor de skal hen om mandagen.
Vi plejer at have åbent for at man kan have gæster med til kor, hvis man har en veninde/ven, der gerne vil med, men her er det vigtigt at der gives besked til pædagogerne/lærerne først. Vi glæder os til et dejligt forløb med jeres skønne børn.

Lato Regular CAP 12px — KALENDER

Lato Regular CAP 11px — ANNA 9C

6 Page elements

This chapter provides the guidelines for page elements in Widget user interface – in general, in forms, icons, tables and error and information messages.

6.1 UI elements

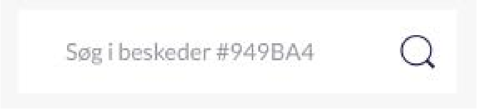
Create new	⊕	Sign up	↗	Filtering	DAG UGE MÅNED
See more	•••	Chosen profile	✓		

Scroll	◀ ▶	Text link	🔗
		Atach file	📎
Link bottom arrow	›	Download doc	🗋
Dropdown arrow	⌄	Notification	2
See more recipients	⌄	Group tag	👥
close	✕		

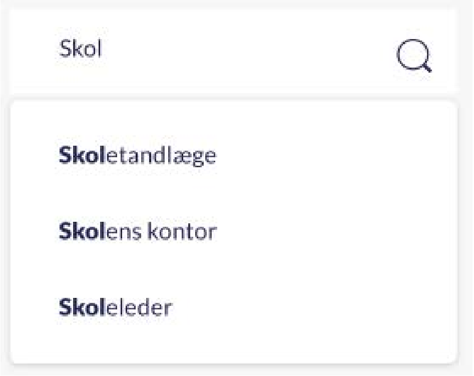
6.2 Form elements

6.2.1 Search field

The search field is shown in three different states depending on their level of functionality they need to support – from the basic search field to an multi-level type-ahead that includes selection.



Søg i beskeder #949BA4

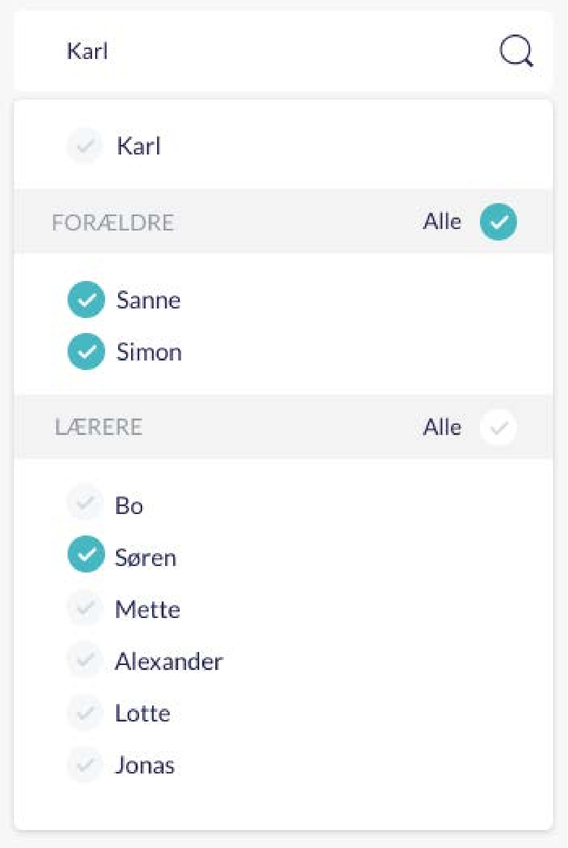


Skol

Skoletandlæge

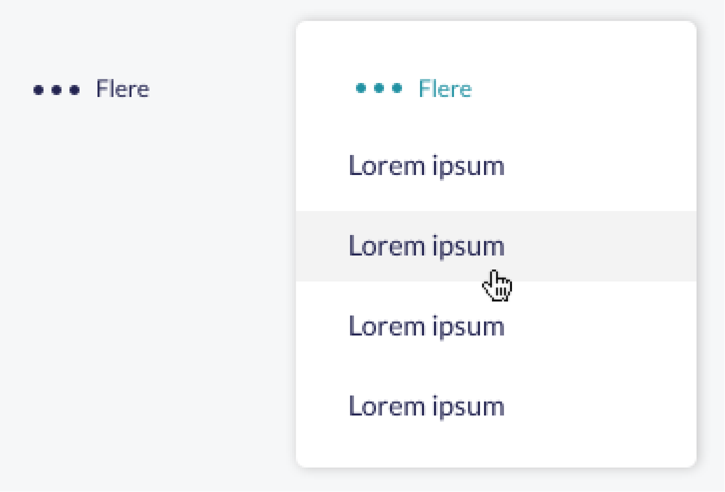
Skolens kontor

Skoleleder



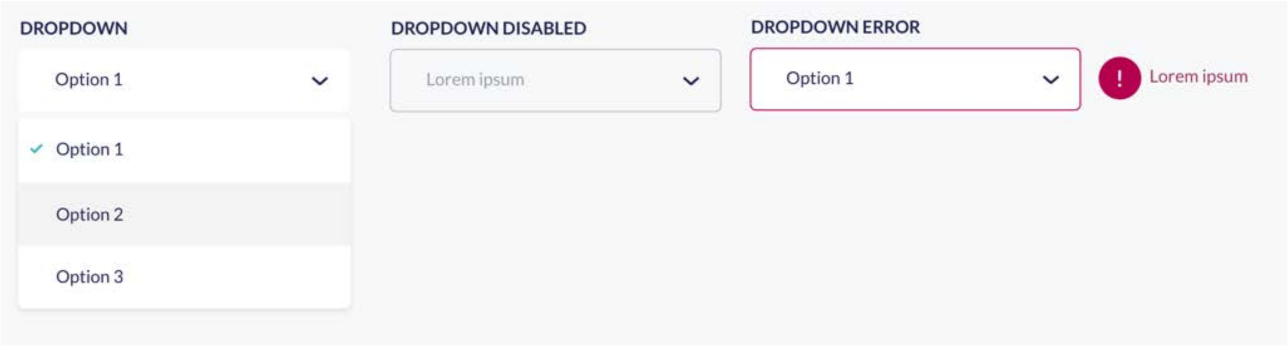
6.2.2 Additional options

The additional options can be used to embed additional (typically secondary) options to an element. The options is displayed in list format as an overlay, as displayed below.

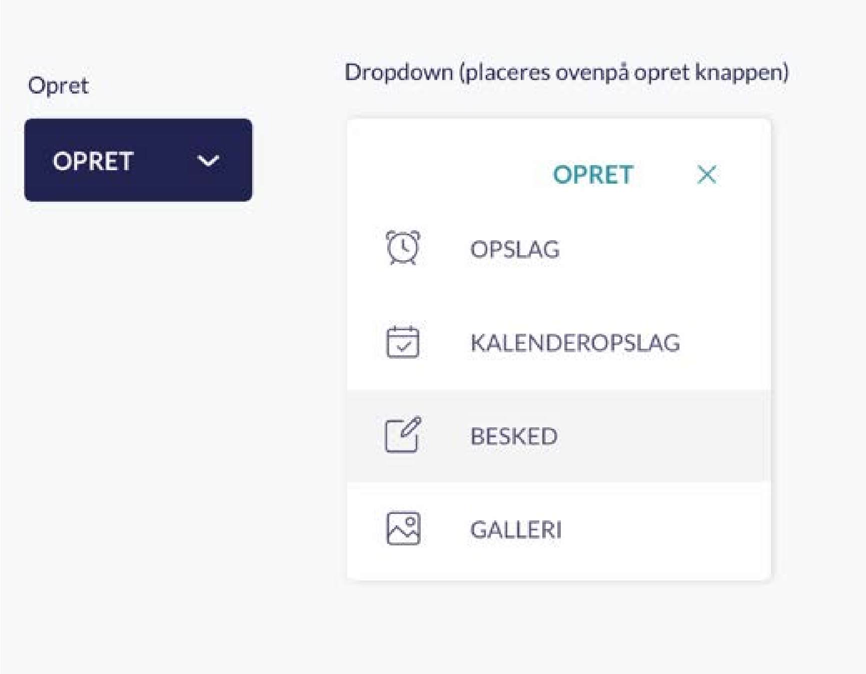


6.2.3 Dropdown

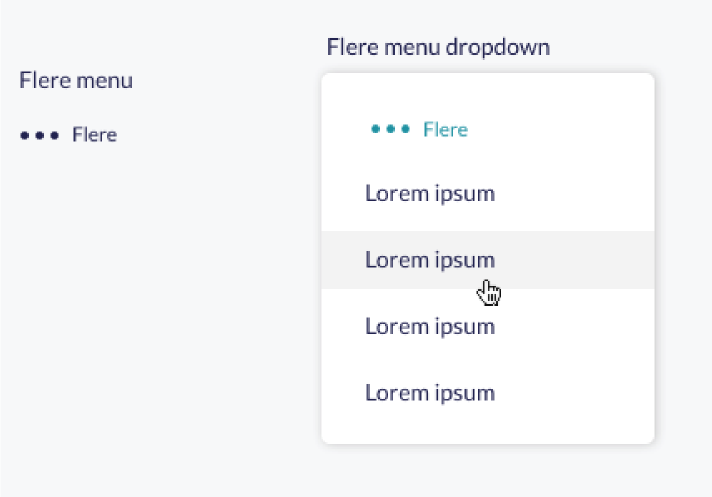
Dropdowns are used to allow the user to choose a value from a list. At an inactive state the dropdown displays a single value and when activated it displays a list. When the user selects a value the dropdown closes and become inactive.



A dropdown can also be placed on top of a button if the button has more values to choose from. The title of the button is still visible for the user so the context of the action is clear.

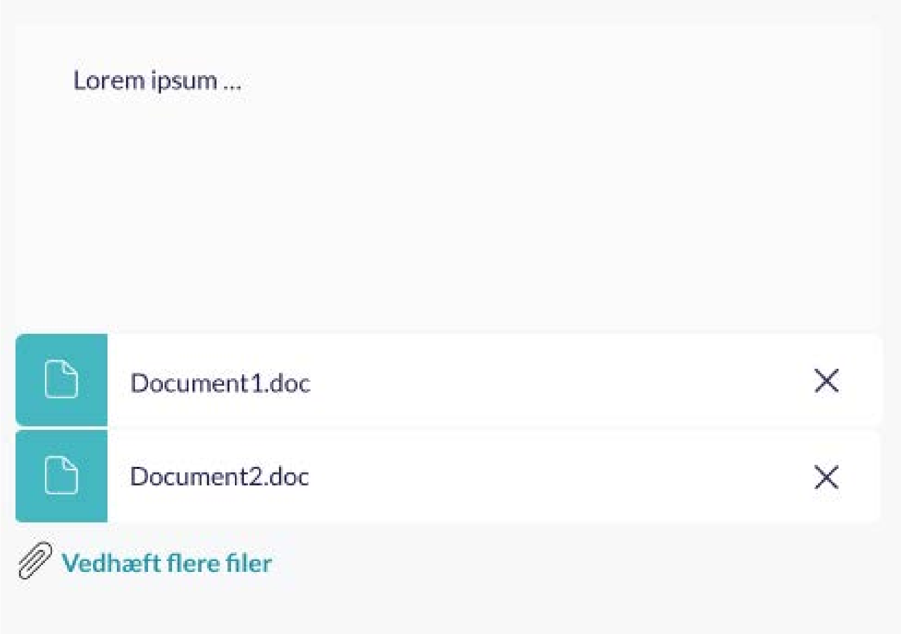


That also applies when there is a menu with more values.



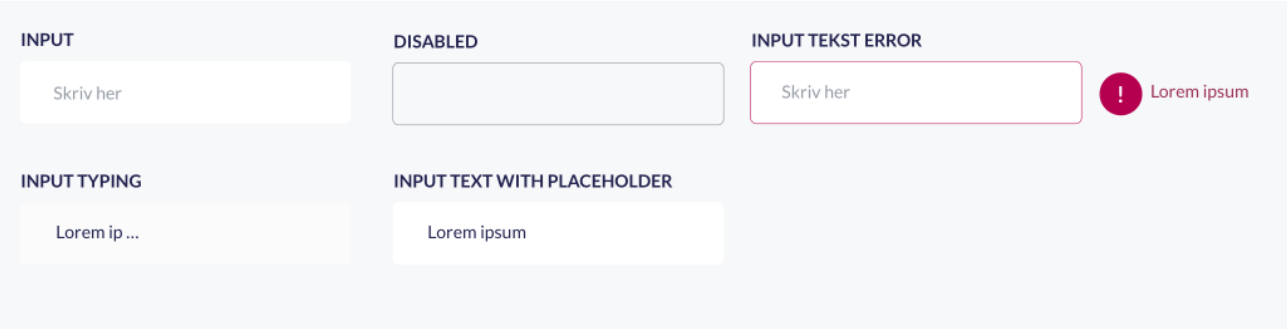
6.2.4 Attachments

When the user attaches a file it will be showed as a button below the input text field. It is possible for the user to remove the file by clicking on th ‘x’.



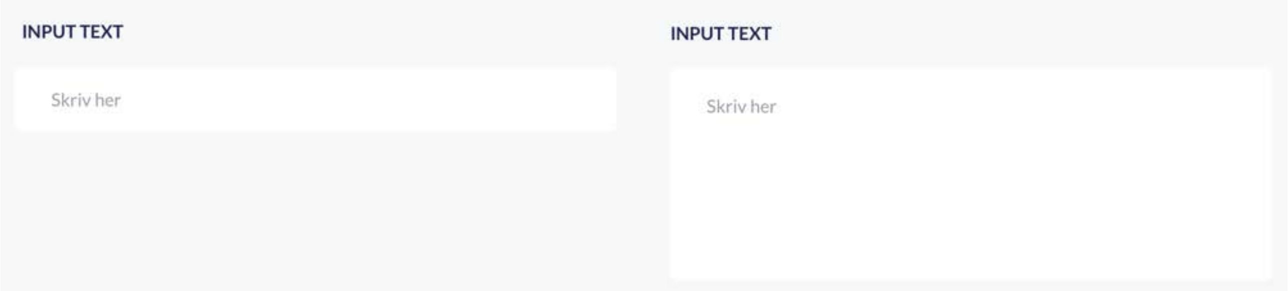
6.2.5 Input fields

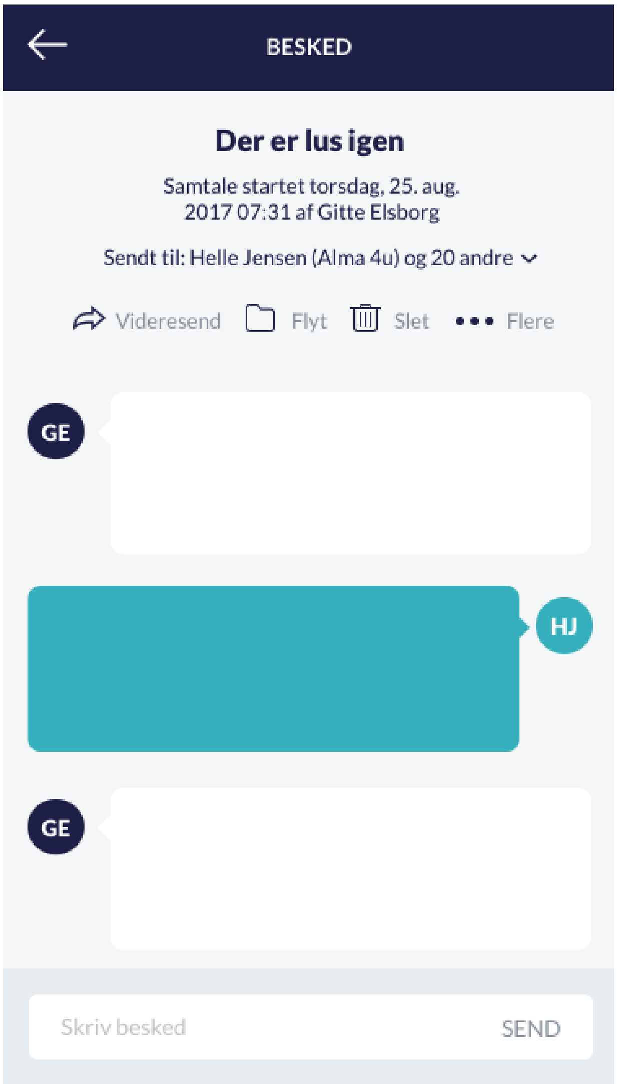
The input fields are used to enable the user to write text information. The input fields can have different states, see the examples below.



6.2.6 Text boxes

Textboxes will be narrow until it is activated in order to make the page more manageable for the user. There is no reason to show functionality to the user before it is needed in order to keep the user interface simple and manageable. For now, this principle is used for messages and creation of calendar events.



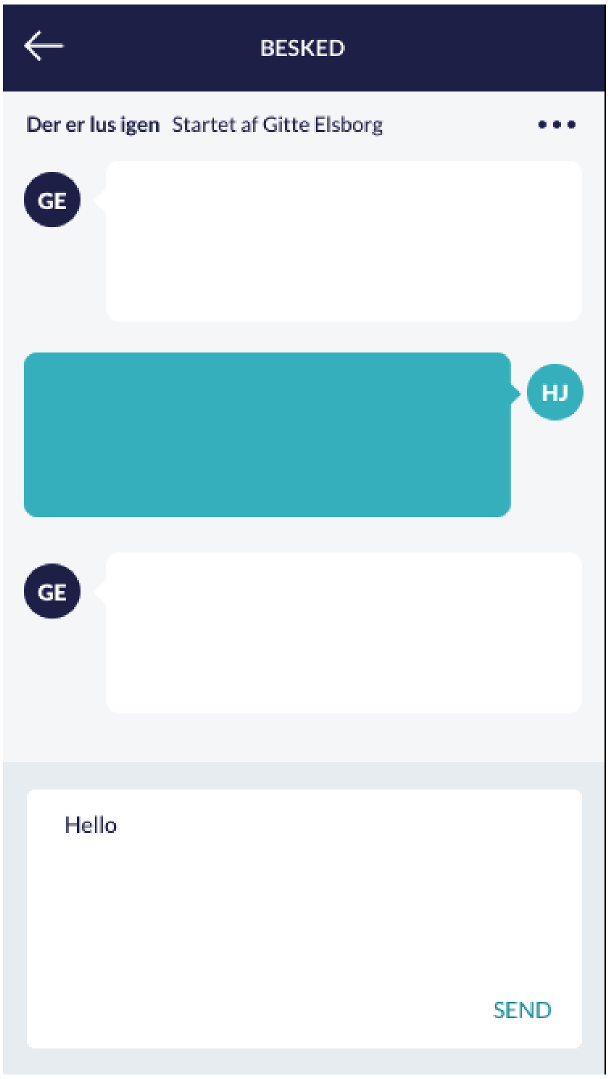


Der er lus igen

Samtale startet torsdag, 25. aug. 2017 07:31 af Gitte Elsborg

Sendt til: Helle Jensen (Alma 4u) og 20 andre

Skriv besked



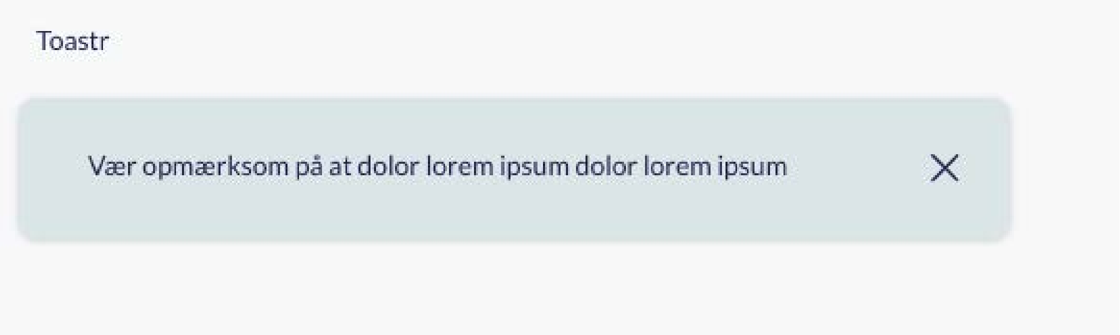
Der er lus igen Startet af Gitte Elsborg

Hello

6.2.7 Information boxes

There are two different types of information boxes that can be represented for the users.

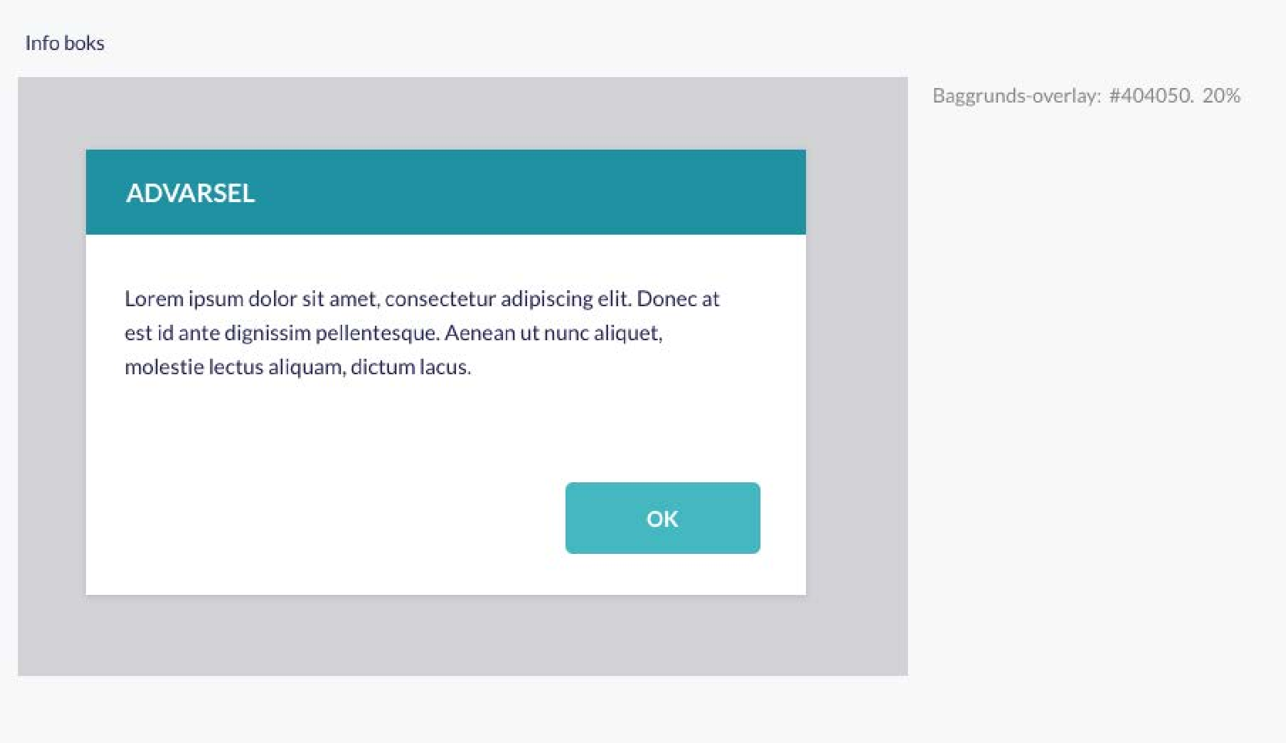
One of them is the ‘toastr’ which are alerts used to give the user non-intrusive-feedback after specific action s. This type of alerts disappears by itself or can be clicked away by the user.



Toastr

Vær opmærksom på at dolor lorem ipsum dolor lorem ipsum

The other type is the dialogbox which can be used to make the user aware of an action made by the user. The user is forced to make an action eg. by accepting the consequences of the specific action.



Info boks

Baggrunds-overlay: #404050. 20%

ADVARSEL

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec at est id ante dignissim pellentesque. Aenean ut nunc aliquet, molestie lectus aliquam, dictum lacus.

6.3 Icons

Simple icons with thin strokes and round forms. It works well in small sizes. Especially icons are subject to extensive user testing, and may change during design. The list below represents the current icons and they use in Aula

Beskeder (Messages)

Galleri (Gallery)

Kalender (Calendar)

Recent

Files & Documents

Search

6.3.1 Icons for smaller kids

A specific set of icons to the children is going to be developed.

6.4 Tables

The solution will ensure that tables are used consistently in behavior and appearance across screens in the Solution.

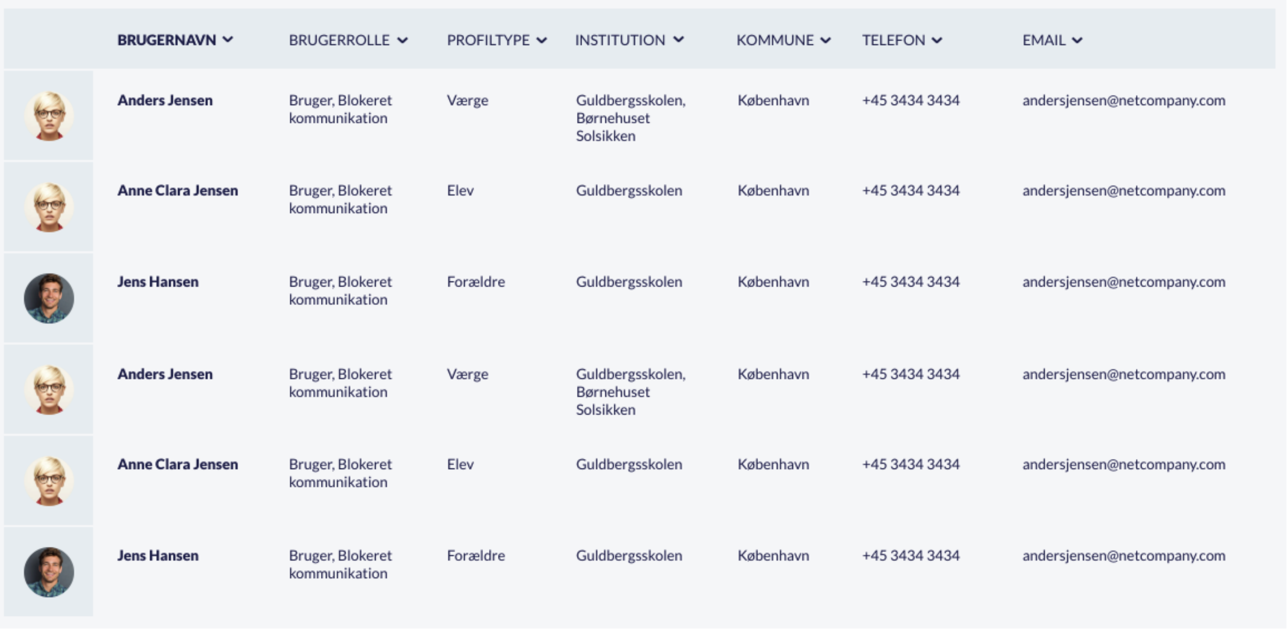
a) Each table has a heading, which briefly describes the content of the table.

b) The first row of the table contains a heading for each column.

c) If a heading is too long to be displayed in its entirety, the number of characters in the header will be displayed as possible and the entire heading will appear in the tool tip.

d) Data in the table can be sorted by column by pressing column heading.

e) All data elements in a column must be of the same type. Amounts and numbers must be right-handed in a column. All other data types must be left-handed. Each heading must be adjusted in the same way as the data elements in the column.



	BRUGERNAVN ⌄	BRUGERROLLE ⌄	PROFILTYPE ⌄	INSTITUTION ⌄	KOMMUNE ⌄	TELEFON ⌄	EMAIL ⌄
	Anders Jensen	Bruger, Blokeret kommunikation	Værge	Guldbergsskolen, Børnehuset Solsikken	København	+45 3434 3434	andersjensen@netcompany.com
	Anne Clara Jensen	Bruger, Blokeret kommunikation	Elev	Guldbergsskolen	København	+45 3434 3434	andersjensen@netcompany.com
	Jens Hansen	Bruger, Blokeret kommunikation	Forældre	Guldbergsskolen	København	+45 3434 3434	andersjensen@netcompany.com
	Anders Jensen	Bruger, Blokeret kommunikation	Værge	Guldbergsskolen, Børnehuset Solsikken	København	+45 3434 3434	andersjensen@netcompany.com
	Anne Clara Jensen	Bruger, Blokeret kommunikation	Elev	Guldbergsskolen	København	+45 3434 3434	andersjensen@netcompany.com
	Jens Hansen	Bruger, Blokeret kommunikation	Forældre	Guldbergsskolen	København	+45 3434 3434	andersjensen@netcompany.com

6.5 Error messages

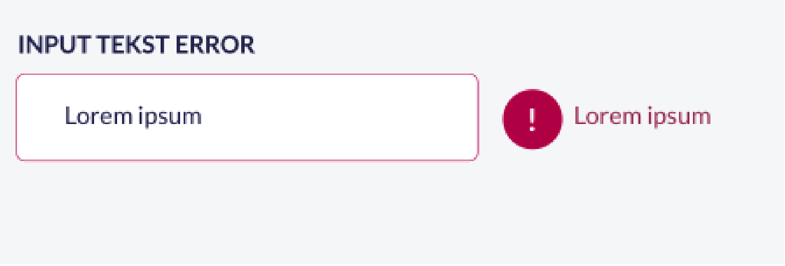
The solution must contain clearly understandable messages that help the user to perform his workflows, correcting errors that do not interfere with the user's workflow more than necessary. Eg. should a message stating that an action has gone well does not require the user to press 'OK button' or the like.

An error message may include due to an error in one or more entry fields cf. Requirement 194, after which the user should be referred to an error message that must be visible without the user moving around the screen.

The texts in the error messages should appear as constructive, visible and accurate as possible.

To ensure that the User sees the error message, buttons must not be inactive for errors or missing entries.

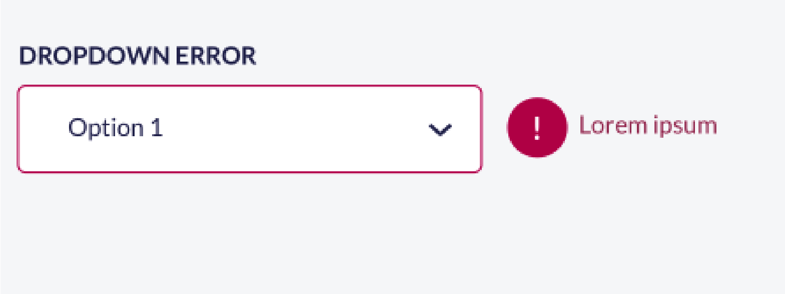
To the extent that the solution in the Solution leads to an error situation, one or more action-oriented error messages are formed that explain in a clear and understandable language the error



INPUT TEKST ERROR

Lorem ipsum

! Lorem ipsum



Furthermore, the solution should display a notification when significant action has been taken, where the user expects a feedback.

Examples of actions that require feedback are the creation of new Begivenheder (events) in the Kalender (Calendar) or importing images to the Galleri (Gallery).

6.6 Information boxes / Feedback

There are two different types of information boxes that can be represented for the users; a Toastr or a Dialogbox.

Toastr

A ‘toastr’ is a non modal alert used to display brief, non-intrusive-feedback to the user after specific actions. It is primarily relevant for full-screen widgets, as Aula does not currently allow access to the its Toastr-functionality.

In a small popup the Toastr provides simple feedback about an operation. It only fills the amount of space required for the message and the current activity remains visible and interactive.

The Toastr doesn't include any interactions. If there is a need for interaction, a Dialogbox is to be used instead.

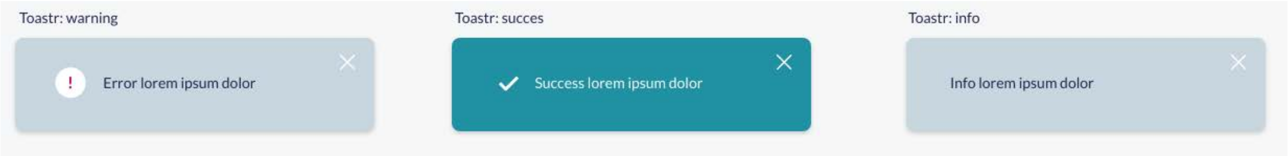
The Toastr automatically disappears after a timeout or can be clicked away by the user.

Placement and settings

The Toastr is placed in the top right. It has the following default settings:

Property	Duration (ms) / Method	Description
Show	300	Duration it takes to execute the show animation
Hide	1000	Duration it takes to execute the hide animation
Time out	5000	Duration before the Hide method is called.
Show method	FadeIn	Animation used to show the element.
Hide method	FadeOut	Animation used to hide the element.

Types



Error lorem ipsum dolor

Success lorem ipsum dolor

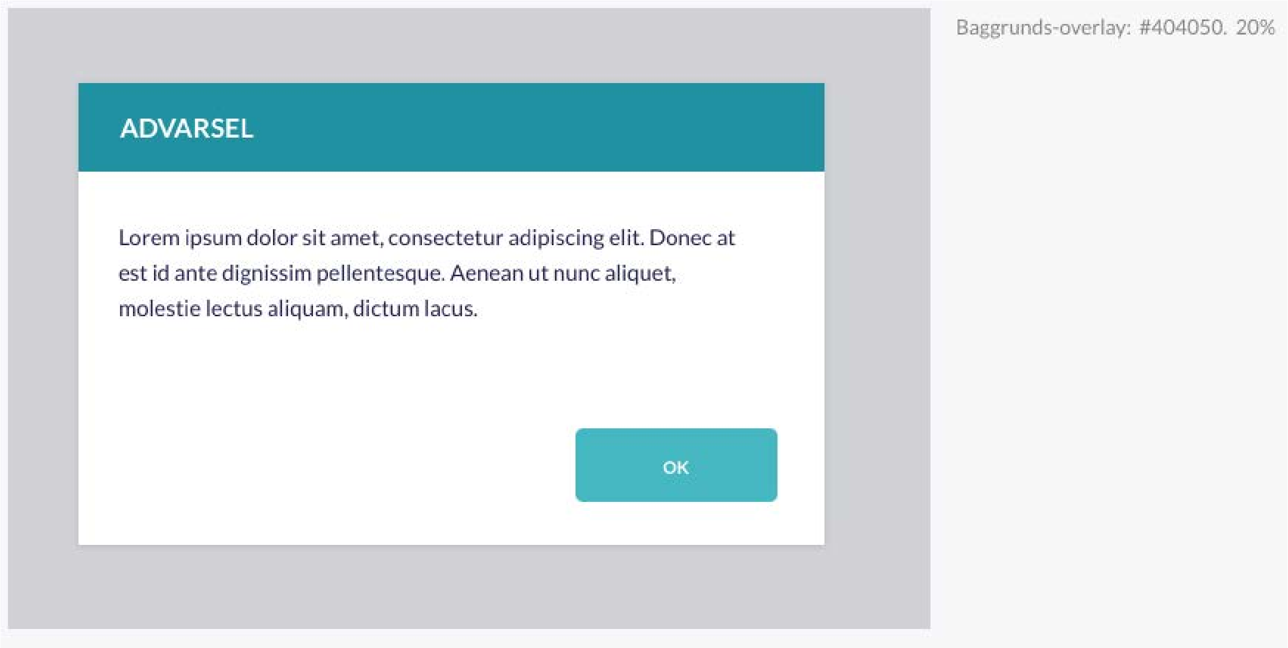
Info lorem ipsum dolor

Currently there are 3 different types of toastr:

- *Succes*: Used to provide information regarding a completed action. The text should inform, which action has been completed.
- *Neutral information*: Used to provide general information.
- *Warning*: Used in situations, where the user should be aware that a certain action has consequences

DialogBox

A dialogbox is used to display information that is crucial, and requires the user’s attention. The dialogbox is by definition intrusive, as it requires the user to click to dismiss it, and its use should be carefully considered. The dialogbox is accompanied by an overlay of the background, such that the current screen is blocked. However, the background overlay is only relevant for full screen widgets.



ADVARSEL

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec at est id ante dignissim pellentesque. Aenean ut nunc aliquet, molestie lectus aliquam, dictum lacus.