

# Webudvikling Frontend

# Webudvikling

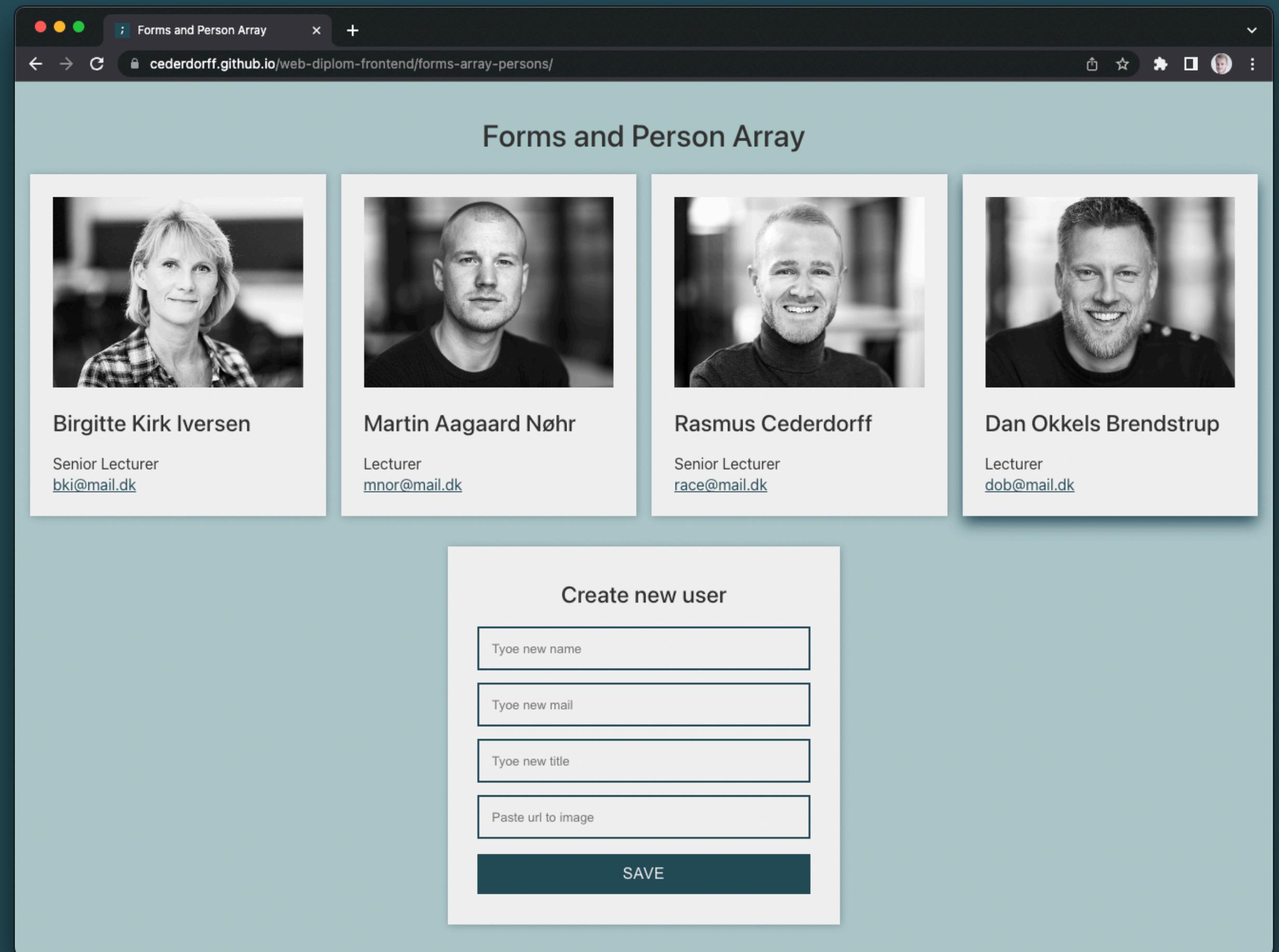


**kea**  
KØBENHAVNS ERHVERVSAKADEMI

# Agenda

- Person Array & JS-koncepter
- Forms, Events & Submit
- Functions & Create New Person
- Evt Intro to Bootstrap

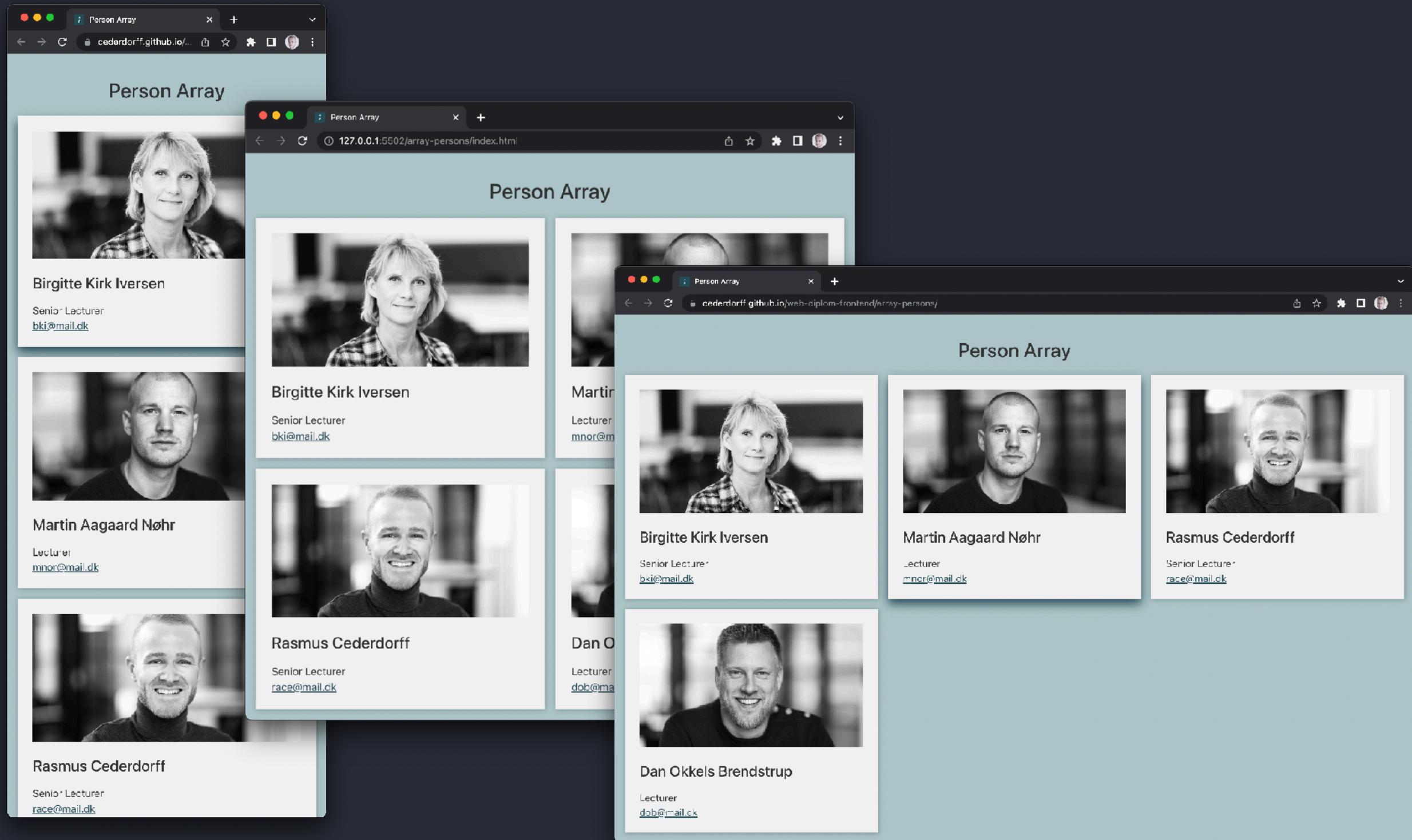
# Todays End Goal



<https://cederdorff.github.io/web-diplom-frontend/forms-array-persons/>

# Person Array with CSS Grid

<https://race.notion.site/Person-Array-with-CSS-Grid-667a504e43d14c17a8a6f2d1c4a4975c>



- Variables, Objects & Arrays
- Loops & For-of-loop
- Template String
- DOM-Manipulation
- If/else
- Improvements, naming & best practice

# CSS Grid

The screenshot shows a web page titled "Person Array". It features a 2x2 grid of portrait photos and names. The top-left cell contains a photo of Birgitte Kirk Iversen, a Senior Lecturer, with the email bki@mail.dk. The top-right cell contains a photo of Martin Aagaard Nøhr, a Lecturer, with the email mnor@mail.dk. The bottom-left cell contains a photo of Rasmus Cederdorff, a Senior Lecturer, with the email race@mail.dk. The bottom-right cell contains a photo of Dan Okkels Brendstrup, a Lecturer, with the email dob@mail.dk. The browser's developer tools are open, showing the DOM structure and CSS styles for the grid container.

```
<section id="content" class="grid-container"></section>
```

```
/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
}
```

```
@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}
```

```
@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

# Forms

Collect user inputs

In HTML we use the `<form>` element  
with `<input>` fields to create forms

The screenshot shows a web browser window with the title "HTML Forms" from w3schools.com. The page content includes a brief description of what an HTML form is used for, followed by an "Example" section. The example shows a simple form with two input fields: "First name:" containing "John" and "Last name:" containing "Doe". Below the fields is a "Submit" button. A green "Try it Yourself »" button is also present. Further down, there is a section titled "The <form> Element" with a code snippet illustrating its structure.

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

**Example**

First name:  
John

Last name:  
Doe

Submit

Try it Yourself »

**The `<form>` Element**

The HTML `<form>` element is used to create an HTML form for user input:

```
<form>
  .
  form elements
  .
</form>
```

# Inputs

Mostly used inside the  
`<form>` element

`<input>` fields can be displayed in  
different ways with different types.

See all types: [HTML Input Types](#)

The `<input>` Element

The HTML `<input>` element is the most used form element.

An `<input>` element can be displayed in many ways, depending on the `type` attribute.

Here are some examples:

Type	Description
<code>&lt;input type="text"&gt;</code>	Displays a single-line text input field
<code>&lt;input type="radio"&gt;</code>	Displays a radio button (for selecting one of many choices)
<code>&lt;input type="checkbox"&gt;</code>	Displays a checkbox (for selecting zero or more of many choices)
<code>&lt;input type="submit"&gt;</code>	Displays a submit button (for submitting the form)
<code>&lt;input type="button"&gt;</code>	Displays a clickable button

All the different input types are covered in this chapter: [HTML Input Types](#).

```
<form>
  <input type="text" name="name" placeholder="Type new name">
  <input type="email" name="mail" placeholder="Type new mail">
  <input type="text" name="title" placeholder="Type new title">
  <input type="url" name="url" placeholder="Paste url to image">
  <button>Save</button>
</form>
```

when you ask Rasmus  
for help and he says  
"Read documentation"



# Forms & Inputs

```
<form onsubmit="saveUser(event)">
  <h2>Create new user</h2>
  <input type="text" name="name" placeholder="Type new name">
  <input type="email" name="mail" placeholder="Type new mail">
  <input type="text" name="title" placeholder="Type new title">
  <input type="url" name="url" placeholder="Paste url to image">
  <button>Save</button>
</form>
```

```
function saveUser(event) {
  event.preventDefault(); // prevent form refreshing page
  const form = event.target; // save reference to form in variable

  // create new person object based on input values
  const newPerson = {
    name: form.name.value,
    mail: form.mail.value,
    title: form.title.value,
    img: form.url.value
  };

  persons.push(newPerson); // add new peron object to array (persons)
  displayPersons(); // make sure to reload all persons
  form.reset(); // reset (clear) input fields
}
```

HTML

JavaScript

# Forms & Inputs

```
<form onsubmit="saveUser(event)">  
  <h2>Create new user</h2>  
  <input type="text" name="name" placeholder="Type new name">  
  <input type="email" name="mail" placeholder="Type new mail">  
  <input type="text" name="title" placeholder="Type new title">  
  <input type="url" name="url" placeholder="Paste url to image">  
  <button>Save</button>  
</form>
```

```
function saveUser(event) {  
  event.preventDefault(); // prevent form refreshing page  
  const form = event.target; // save reference to form in variable  
  
  // create new person object based on input values  
  const newPerson = {  
    name: form.name.value,  
    mail: form.mail.value,  
    title: form.title.value,  
    img: form.url.value  
  };  
  
  persons.push(newPerson); // add new peron object to array (persons)  
  displayPersons(); // make sure to reload all persons  
  form.reset(); // reset (clear) input fields  
}
```

HTML

JavaScript

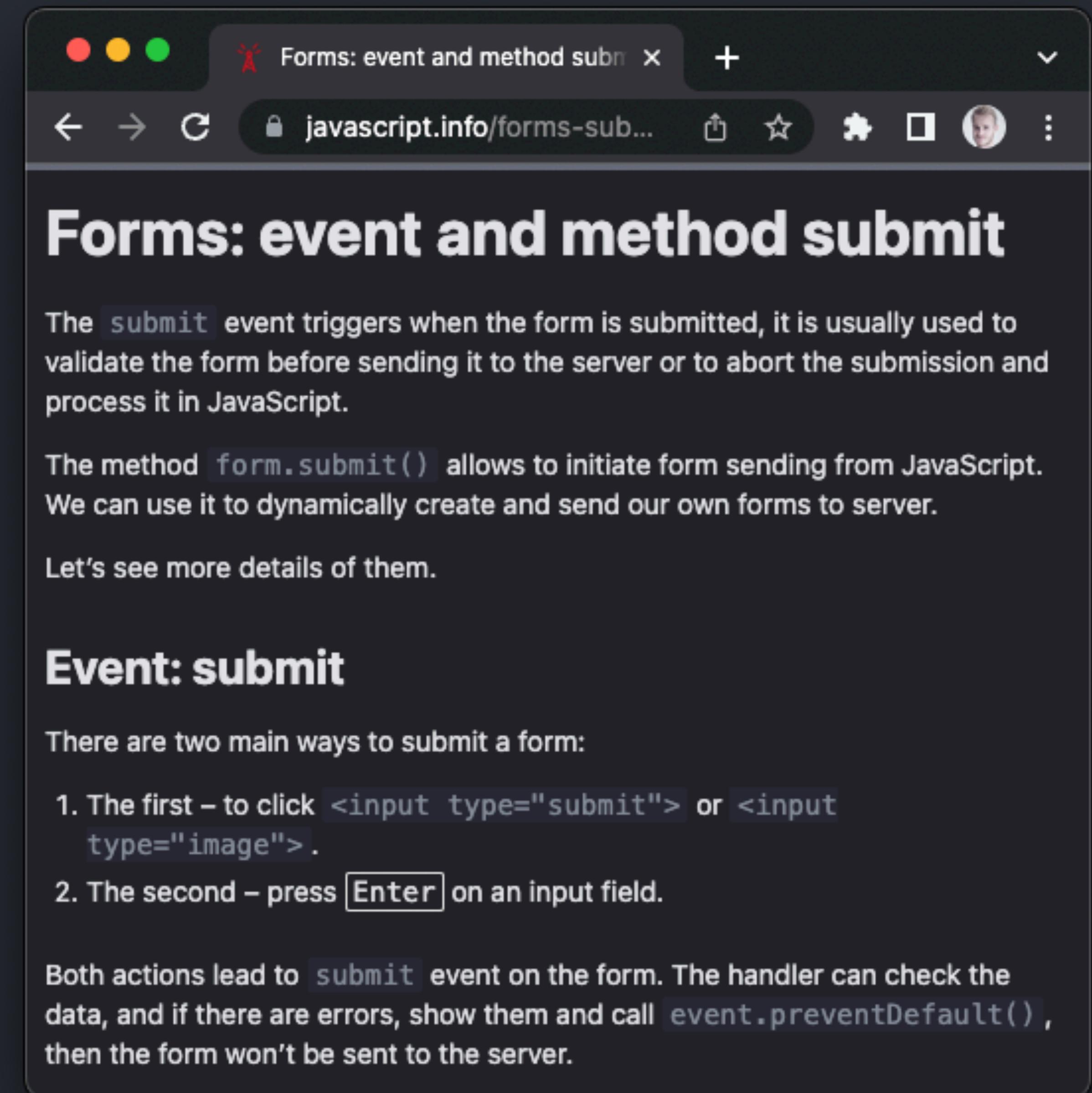
# Submit event

```
<form onsubmit="saveUser(event)">
  <h2>Create new user</h2>
  <input type="text" name="name" placeholder="Type new name">
  <input type="email" name="mail" placeholder="Type new mail">
  <input type="text" name="title" placeholder="Type new title">
  <input type="url" name="url" placeholder="Paste url to image">
  <button>Save</button>
</form>
```

```
function saveUser(event) {
  event.preventDefault(); // prevent form refreshing page
  const form = event.target; // save reference to form in variable

  // create new person object based on input values
  const newPerson = {
    name: form.name.value,
    mail: form.mail.value,
    title: form.title.value,
    img: form.url.value
  };

  persons.push(newPerson); // add new peron object to array (persons)
  displayPersons(); // make sure to reload all persons
  form.reset(); // reset (clear) input fields
}
```



The screenshot shows a web browser window with the title "Forms: event and method submit". The URL in the address bar is "javascript.info/forms-submit". The main content of the page is titled "Forms: event and method submit" and contains the following text:

The `submit` event triggers when the form is submitted, it is usually used to validate the form before sending it to the server or to abort the submission and process it in JavaScript.

The method `form.submit()` allows to initiate form sending from JavaScript. We can use it to dynamically create and send our own forms to server.

Let's see more details of them.

## Event: submit

There are two main ways to submit a form:

1. The first – to click `<input type="submit">` or `<input type="image">`.
2. The second – press `Enter` on an input field.

Both actions lead to `submit` event on the form. The handler can check the data, and if there are errors, show them and call `event.preventDefault()`, then the form won't be sent to the server.

# Prevent Default

```
<form onsubmit="saveUser(event)">
  <h2>Create new user</h2>
  <input type="text" name="name" placeholder="Type new name">
  <input type="email" name="mail" placeholder="Type new mail">
  <input type="text" name="title" placeholder="Type new title">
  <input type="url" name="url" placeholder="Paste url to image">
  <button>Save</button>
</form>

function saveUser(event) {
  event.preventDefault(); // prevent form refreshing page
  const form = event.target; // save reference to form in variable

  // create new person object based on input values
  const newPerson = {
    name: form.name.value,
    mail: form.mail.value,
    title: form.title.value,
    img: form.url.value
  };

  persons.push(newPerson); // add new peron object to array (persons)
  displayPersons(); // make sure to reload all persons
  form.reset(); // reset (clear) input fields
}
```

The screenshot shows a web browser window with the title "preventDefault() Event Method". The URL in the address bar is "w3schools.com/jsref/event\_preventdefault.asp". The browser interface includes standard controls like back, forward, and search, along with navigation links for "HTML" and "CSS".

## Definition and Usage

The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

For example, this can be useful when:

- Clicking on a "Submit" button, prevent it from submitting a form
- Clicking on a link, prevent the link from following the URL

**Note:** Not all events are cancelable. Use the cancelable property to find out if an event is cancelable.

**Note:** The preventDefault() method does not prevent further propagation of an event through the DOM. Use the stopPropagation() method to handle this.

[https://www.w3schools.com/jsref/event\\_preventdefault.asp](https://www.w3schools.com/jsref/event_preventdefault.asp)

# Events & EventListener

Assign an onclick event to a button element:

```
<button onclick="displayDate()">Try it</button>
```

Assign an onclick event to a button element:

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

```
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
```

Add an event listener that fires when a user clicks a button:

```
document.getElementById("myBtn").addEventListener("click", displayDate);
```

The screenshot shows a web browser window with a dark theme. The title bar reads "HTML DOM Event Object" and the address bar shows "w3schools.com/jsref/dom\_obj\_event.asp". The main content area has a header "HTML DOM Events" with navigation buttons "Previous" and "Next". Below the header, a text block states: "HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document. Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button). For a tutorial about Events, read our [JavaScript Events Tutorial](#)." A table lists various HTML events with their descriptions and categories:

Event	Description	Belongs To
<a href="#">abort</a>	The event occurs when the loading of a media is aborted	<a href="#">UiEvent</a> , <a href="#">Event</a>
<a href="#">afterprint</a>	The event occurs when a page has started printing, or if the print dialogue box has been closed	<a href="#">Event</a>
<a href="#">animationend</a>	The event occurs when a CSS animation has completed	<a href="#">AnimationEvent</a>
<a href="#">animationiteration</a>	The event occurs when a CSS animation is repeated	<a href="#">AnimationEvent</a>
<a href="#">animationstart</a>	The event occurs when a CSS animation has started	<a href="#">AnimationEvent</a>
<a href="#">beforeprint</a>	The event occurs when a page is about to be printed	<a href="#">Event</a>
<a href="#">beforeunload</a>	The event occurs before the document is about to be unloaded	<a href="#">UiEvent</a> , <a href="#">Event</a>
<a href="#">blur</a>	The event occurs when an element loses focus	<a href="#">FocusEvent</a>
<a href="#">canplay</a>	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	<a href="#">Event</a>
<a href="#">canplaythrough</a>	The event occurs when the browser can play through the media without stopping for buffering	<a href="#">Event</a>
<a href="#">change</a>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	<a href="#">Event</a>
<a href="#">click</a>	The event occurs when the user clicks on an element	<a href="#">MouseEvent</a>

# UI FRAMEWORK

COLLECTION OF HTML, CSS & JAVASCRIPT

STYLING, UI ELEMENTS & ‘READY TO USE’ COMPONENTS

STANDARDISED UI

TYPOGRAPHY, GRID SYSTEM, RESPONSIVE UTILITIES, NAVIGATION, ICONS,  
FONTS, UI ELEMENTS, ETC.



Foundation  
Start here, build everywhere.



# WHY USE A UI FRAMEWORK?

PREBUILD LAYOUT, STYLING & FUNCTIONALITY

SIMPLIFIES & SPEEDS UP THE DEV PROCESS

FOCUS ON REQUIREMENTS & USER-NEEDS

BROWSER COMPABILITY

STRUCTURE, SEMANTICS & STANDARDIZATION

DEMO, TEMPLATES & THEMES

RAPID PROTOTYPING

MINIMAL CUSTOM STYLING

# WHY USE A UI FRAMEWORK?

BEST PRACTICE

UI & UX

WELL-TESTED

PROTOTYPING

# UI FRAMEWORK



# PROJECT TEMPLATE



# BOOTSTRAP

THE MOST POPULAR FRAMEWORK FOR BUILDING RESPONSIVE &  
MOBILE-FIRST WEB APPS WITH HTML, CSS & JAVASCRIPT



# BOOTSTRAP

CSS CLASSES, COMPONENTS & FUNCTIONALITY



# PREBUILD CSS & STYLING

## TYPOGRAPHY, COLORS, STYLES, CONTAINERS, ETC



**B Examples · Bootstrap v5.2**

[Docs](#) [Examples](#) [Icons](#) [Themes](#) [Blog](#)

v5.2

# Examples

Quickly get a project started with any of our examples ranging from using parts of the framework to custom components and layouts.

[Download examples](#) [Download source code](#)

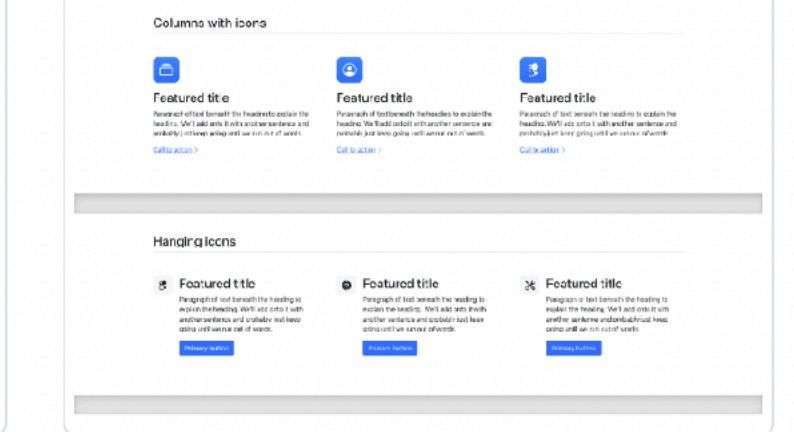
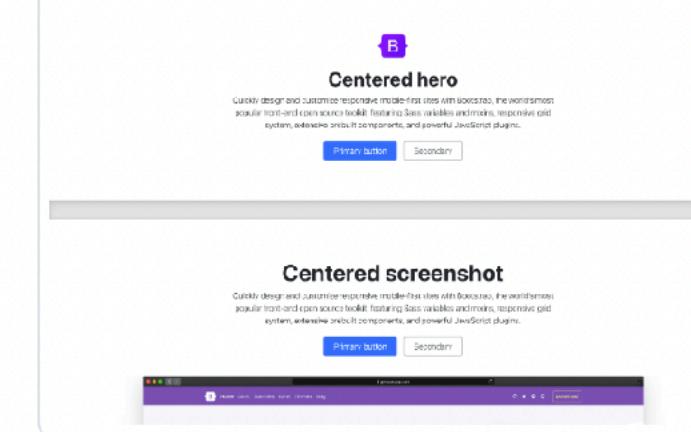
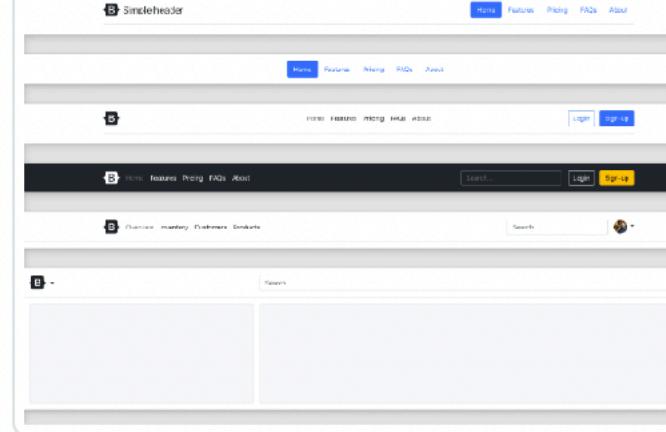


Your new development career awaits. Check out the latest listings.  
ads via Carbon

---

## Snippets

Common patterns for building sites and apps that build on existing components and utilities with custom CSS and more.



### Headers

Display your branding, navigation, search, and more with these header components

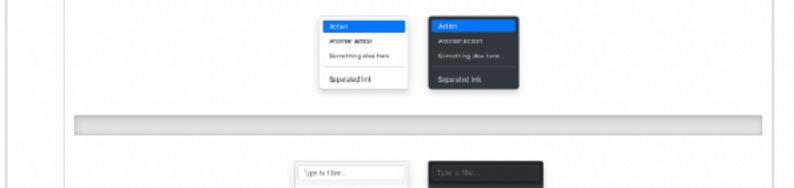
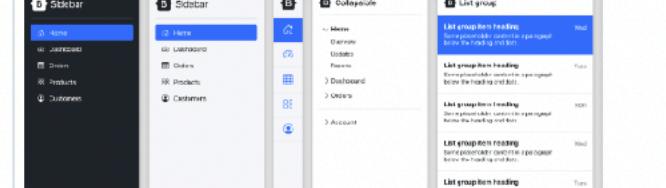
### Heroes

Set the stage on your homepage with heroes that feature clear calls to action.

### Features

Explain the features, benefits, or other details in your marketing content.

<https://getbootstrap.com/docs/5.2/examples/>





# USE THE DOCS

... COPY AND PASTE

<https://getbootstrap.com/docs/5.2/getting-started/introduction/>

Bootstrap template

cederdorff.github.io/web-diplom-frontend/bootstrap-template/

Navbar Home Link Dropdown ▾ Disabled

Search

Search

**Birgitte Kirk Iversen**  
Senior Lecturer  
[bki@mail.dk](mailto:bki@mail.dk)

**Martin Aagaard Nøhr**  
Lecturer  
[mnor@mail.dk](mailto:mnor@mail.dk)

**Rasmus Cederdorff**  
Senior Lecturer  
[race@mail.dk](mailto:race@mail.dk)

**Dan Okkels Brendstrup**  
Lecturer  
[dob@mail.dk](mailto:dob@mail.dk)

<https://cederdorff.github.io/web-diplom-frontend/bootstrap-template/>