

# Open-Source Software Practice

## 9. Web App

Instructor: Jaemin Jo (조재민, [jmjo@skku.edu](mailto:jmjo@skku.edu))  
Interactive Data Computing Lab (*IDCLab*),  
College of Computing and Informatics,  
Sungkyunkwan University

# Review: HTML & CSS


---

- **HTML** for the structure of a web page
- **CSS** for the style
- **JavaScript** for the interaction
- An HTML document consists of tags (hierarchy!).
  - `<html>`, `<head>`, `<body>`, `<p>`, `<ul>`, `<li>`, `<img>`, `<table>`, ...
- CSS rule = selector + (property: value)
  - Id selector: `#id`, class selector: `.class`, tag name selector: `tag_name`
  - Properties: color, margin, padding, background, font-size, ...
  - Selector specificity

- I am not good at design.. Can I reuse the CSS files that designers made?
- **Front-end toolkits:** reusable CSS and JS code for Web development
- **Bootstrap:** one of the most popular front-end toolkits (open source)
  - <https://getbootstrap.com/>
  - v5.2.2
  - Originally developed by Twitter
- CSS components: layout, form, button, list, navigation, ...
- JS components: dialog, toast message, accordion, ...

# Bootstrap

**New in v5.2** CSS variables, responsive offcanvas, new utilities, and more!

The Bootstrap logo, a purple shield-like shape with a white letter 'B' inside.

## Build fast, responsive sites with Bootstrap

Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.

```
$ npm i bootstrap@5.2.2
```

[Read the docs](#)

Currently **v5.2.2** · [Download](#) · [v4.6.x docs](#) · [All releases](#)

# Bootstrap

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min."
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bund
  </body>
</html>
```

## CDN links

As reference, here are our primary CDN links.

Description	URL
CSS	<a href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css">https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css</a>
JS	<a href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js">https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js</a>

# New Starter Template

- <https://getbootstrap.com/docs/5.2/examples/starter-template/>

```
<!DOCTYPE html>
<html>
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" crossorigin="anonymous"
">

  <title>Page Title</title>
  <style>
    /* CSS Code */
  </style>
</head>
<body>
  <!-- HTML Code -->

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"></script>

  <script>
    /* JS Code */
  </script>
</body>
</html>
```

# Exploring Bootstrap

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

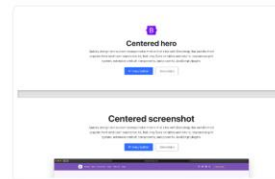
## Snippets

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



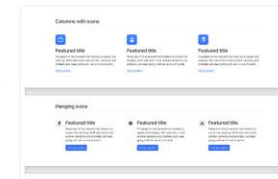
### Headers

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



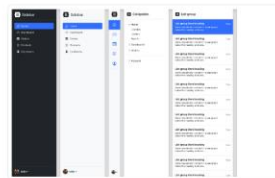
### Heroes

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



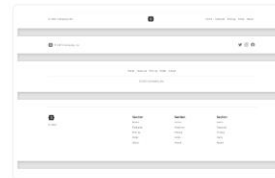
### Features

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



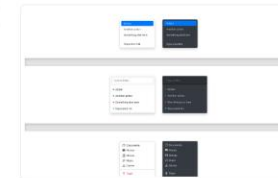
### Sidebars

Common navigation patterns ideal for offcanvas or multi-column layouts.



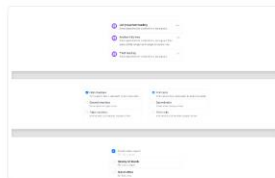
### Footers

Finish every page strong with an awesome footer, big or small.



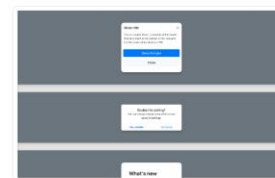
### Dropdowns

Enhance your dropdowns with filters, icons, custom styles, and more.



### List groups

Extend list groups with utilities and custom styles for any content.

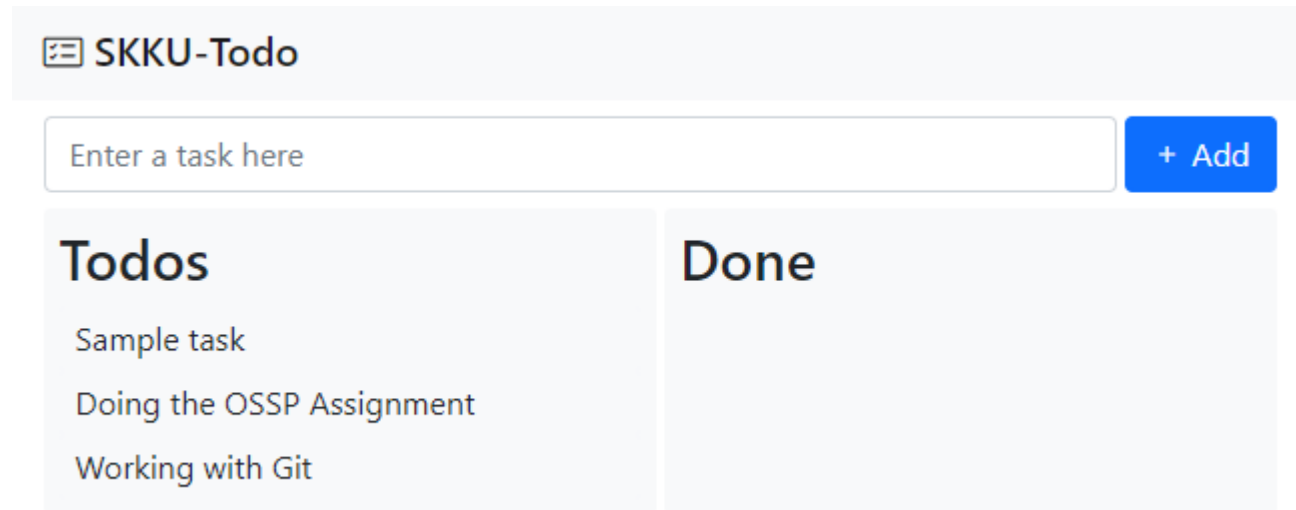


### Modals

Transform modals to serve any purpose, from feature tours to dialogs.

# SKKU-Todo

- Let's create a Web application SKKU-Todo.
- SKKU-Todo is a simple task management app.
- Features:
  - Add a task
  - Remove a task
  - Mark as done
  - Save and restore



The screenshot shows the SKKU-Todo web application interface. At the top, there is a header with a checklist icon and the text "SKKU-Todo". Below the header is a form with a text input field containing the placeholder "Enter a task here" and a blue button with a plus sign and the text "+ Add". Below the form are two columns. The left column is titled "Todos" and contains three task items: "Sample task", "Doing the OSSP Assignment", and "Working with Git". The right column is titled "Done" and is currently empty.



# Adding a Container

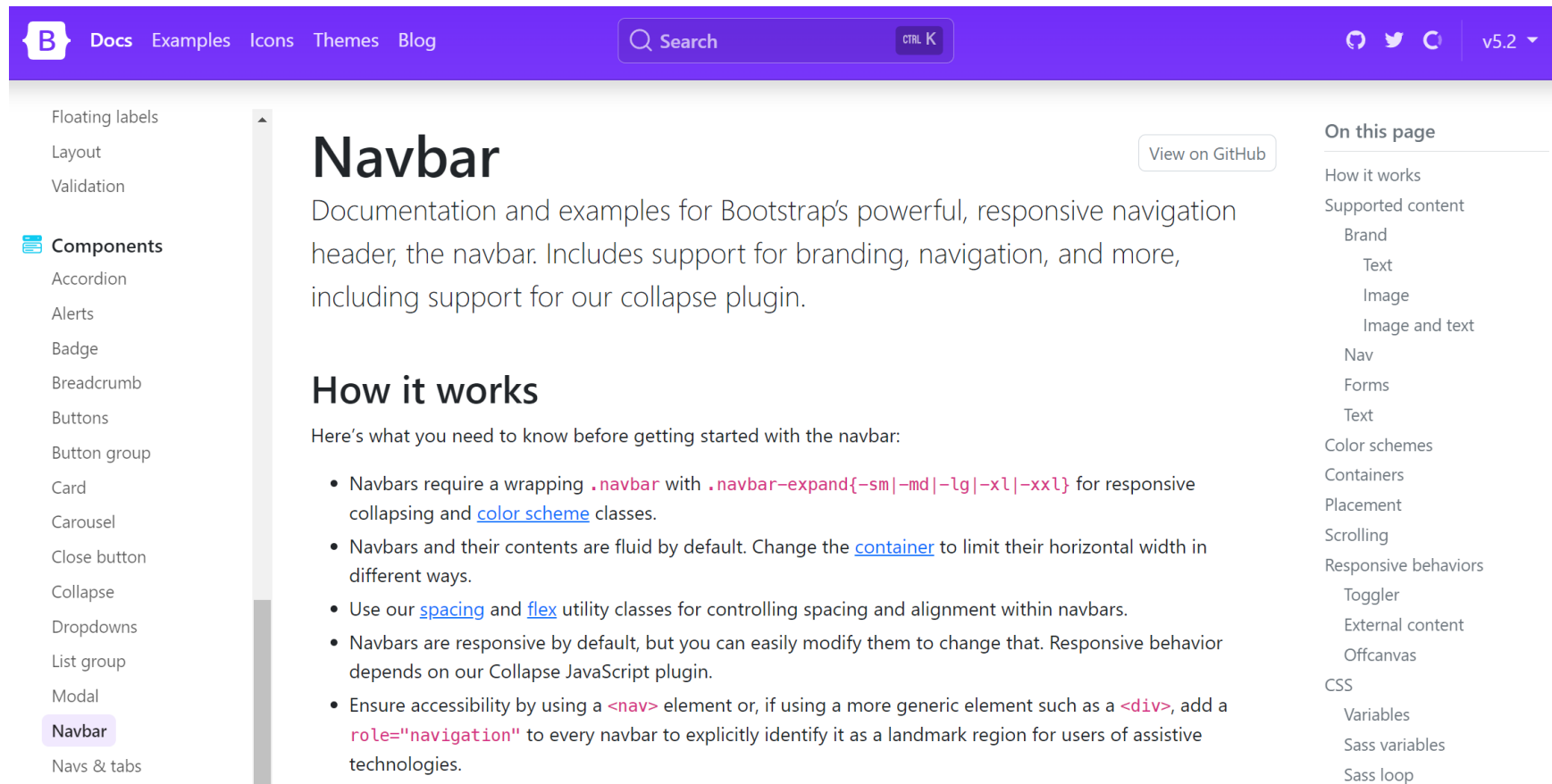
- Let's define a global container first.
  - All components will reside in this container.
- Create `<div>` and set a class name "container".
- The default width of the container is 1,320px.
- Who defined the style? Bootstrap did!

```
<div class="container">  
  This is the container!  
</div>
```

This is the container!

# Adding a Navbar

- All apps start with an awesome logo. Let's create a navbar.



The screenshot shows the Bootstrap 5.2 documentation page for the Navbar component. The page has a purple header with navigation links (Docs, Examples, Icons, Themes, Blog), a search bar, and social media icons. The left sidebar lists various components, with 'Navbar' highlighted. The main content area features the title 'Navbar', a brief description, and a 'How it works' section with a list of key points. The right sidebar contains a table of contents for the page.

## Navbar

Documentation and examples for Bootstrap's powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more, including support for our collapse plugin.

### How it works

Here's what you need to know before getting started with the navbar:

- Navbars require a wrapping `.navbar` with `.navbar-expand{<sm>|<md>|<lg>|<xl>|<xxl>}` for responsive collapsing and [color scheme](#) classes.
- Navbars and their contents are fluid by default. Change the [container](#) to limit their horizontal width in different ways.
- Use our [spacing](#) and [flex](#) utility classes for controlling spacing and alignment within navbars.
- Navbars are responsive by default, but you can easily modify them to change that. Responsive behavior depends on our Collapse JavaScript plugin.
- Ensure accessibility by using a `<nav>` element or, if using a more generic element such as a `<div>`, add a `role="navigation"` to every navbar to explicitly identify it as a landmark region for users of assistive technologies.

#### On this page

- How it works
- Supported content
  - Brand
  - Text
  - Image
  - Image and text
- Nav
- Forms
- Text
- Color schemes
- Containers
- Placement
- Scrolling
- Responsive behaviors
  - Toggler
  - External content
  - Offcanvas
- CSS
  - Variables
  - Sass variables
  - Sass loop

# Adding a Navbar

Navbar Home Link Dropdown ▾ Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" d
            Dropdown
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Too complex for us!

Navbar

Navbar

```
<!-- As a link -->
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>

<!-- As a heading -->
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <span class="navbar-brand mb-0 h1">Navbar</span>
  </div>
</nav>
```

Let's use this one.

# Adding a Navbar

- Copy and paste one example above the global container.
- Change the class name “container-fluid” to “container”.

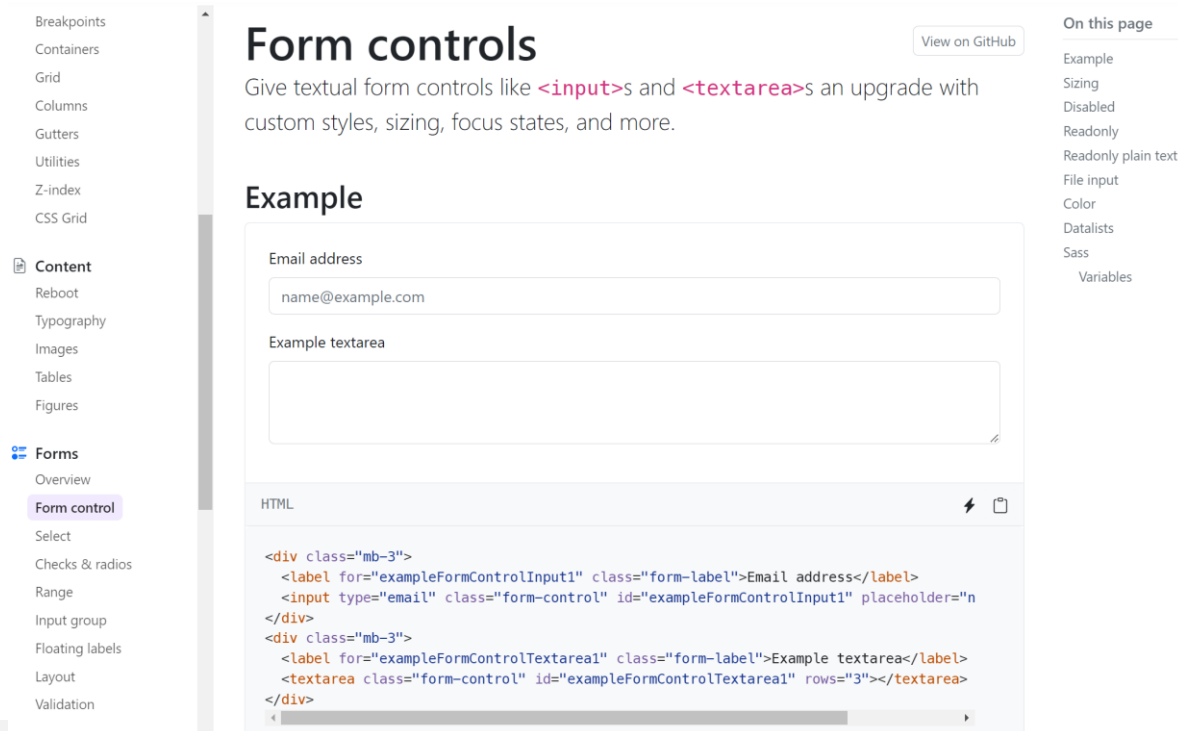
```
<nav class="navbar navbar-light bg-light">
  <div class="container">
    <span class="navbar-brand mb-0 h1">SKKU-Todo</span>
  </div>
</nav>
<div class="container">
  This is the container!
</div>
```

SKKU-Todo

This is the container!

# Adding a Form

- We need a form where the user can enter a task name.
  - A text box + a button
- Let's find an example in the documentation.



The screenshot shows the Material Design documentation page for "Form controls". The left sidebar contains a navigation menu with categories like Breakpoints, Containers, Grid, Columns, Gutters, Utilities, Z-index, CSS Grid, Content, Reboot, Typography, Images, Tables, Figures, Forms, Overview, Form control (highlighted), Select, Checks & radios, Range, Input group, Floating labels, Layout, and Validation. The main content area has the title "Form controls" and a subtitle "Give textual form controls like `<input>`s and `<textarea>`s an upgrade with custom styles, sizing, focus states, and more." Below this is an "Example" section showing a form with an "Email address" input field and an "Example textarea". The bottom of the page shows the HTML code for these elements. On the right side, there is a "On this page" section with links to Example, Sizing, Disabled, Readonly, Readonly plain text, File input, Color, Datalists, Sass, and Variables.

## Form controls

Give textual form controls like `<input>`s and `<textarea>`s an upgrade with custom styles, sizing, focus states, and more.

### Example

Email address

Example textarea

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="n
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">Example textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
```

# Adding a Form

- Seems that the first example is suitable for us.
- Change the text inside `<label>` to “Task”.
- Change the input type from “email” to “text” to receive any text.
- Change the placeholder to “Enter a task”.

## Form controls

[View on GitHub](#)

Give textual form controls like `<input>`s and `<textarea>`s an upgrade with custom styles, sizing, focus states, and more.

### Example

Email address

name@example.com

Example textarea

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name"
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">Example textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
```

Copy

# Adding a Form

```
<nav class="navbar navbar-light bg-light">
  <div class="container">
    <span class="navbar-brand mb-0 h1">SKKU-Todo</span>
  </div>
</nav>
<div class="container">
  <div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">Task</label>
    <input type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter a task here">
  </div>
</div>
```

SKKU-Todo

Task

# Adjusting the Width

- Don't you think the container is too wide?
- Let's adjust its width by adding a CSS rule.

SKKU-Todo

Task

Enter a task here

```
<style>
  .container {
    width: 640px;
  }
</style>
```

SKKU-Todo

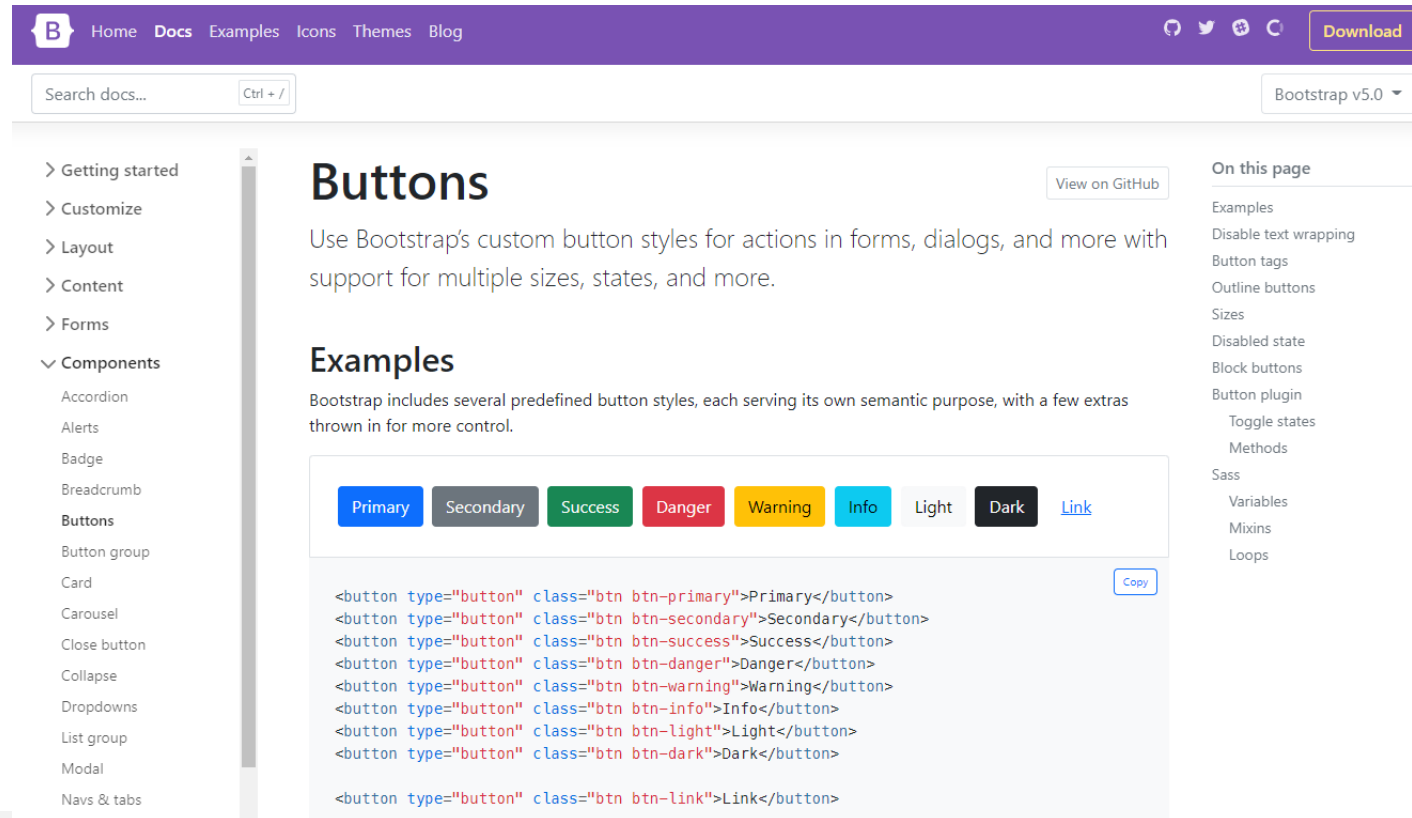
Task

Enter a task here



# Adding a Button

- Let's add a button below the form!
- Again, there are some examples.



The screenshot shows the Bootstrap 5.0 documentation page for Buttons. The page has a purple header with navigation links: Home, Docs, Examples, Icons, Themes, Blog, and a Download button. A search bar and a version selector (Bootstrap v5.0) are also present. The left sidebar shows a tree of components, with 'Buttons' selected under 'Components'. The main content area is titled 'Buttons' and includes a description: 'Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.' Below this is an 'Examples' section with a visual row of buttons: Primary (blue), Secondary (gray), Success (green), Danger (red), Warning (yellow), Info (cyan), Light (light gray), Dark (dark gray), and a Link (blue text). A 'Copy' button is next to the code block. The code block contains the following HTML snippets:

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```

On the right side, there is a 'On this page' section with links to various examples and components.

# Adding a Button

- Let's use a button with the class "btn-primary".
- Put the button in <div> just below <input>.

```
<div class="container">
  <div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">Task</label>
    <input type="text" class="form-
control" id="exampleFormControlInput1" placeholder="Enter a task here">
    <button type="button" class="btn btn-primary">Add</button>
  </div>
</div>
```

SKKU-Todo

Task

# Adding Lists

- We need two lists, one for todos and the other for completed tasks.
- We will use the following markup.

```
<div class="container">
  <div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">Task</label>
    <input type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter a task here">
    <button type="button" class="btn btn-primary">Add</button>
  </div>

  <div>
    <h3>Todos</h3>
    <div id="todo-list">
      <div class="task">Sample task</div>
    </div>
  </div>
  <div>
    <h3>Done</h3>
    <div id="done-list">

    </div>
  </div>
</div>
```

## SKKU-Todo

Task

Add

## Todos

Sample task

Done

# Styling the Interface

- The current interface is ugly! Let's improve it.
  1. Side-by-side layout
  2. Adjusting the space around components
  3. Using icons

SKKU-Todo

Task

Add

Todos

Sample task

Done

# Styling the Interface

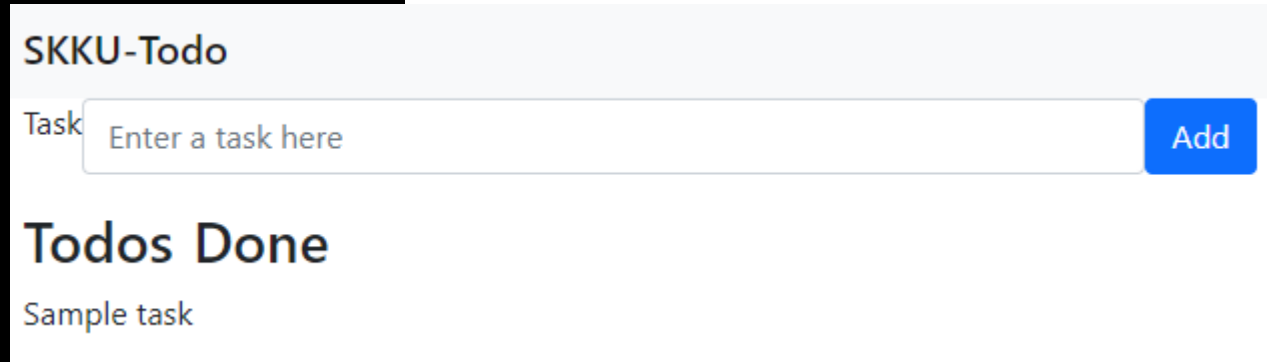
- By default, HTML block elements are stacked vertically.
- If you want to place them side-by-side, put them in a flexbox.
  - Fortunately, Bootstrap provides a lot of examples. See <https://getbootstrap.com/docs/5.2/utilities/flex/>
  - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- If you give a class name “d-flex” to <div>, children under <div> are placed side-by-side by default.
  - “d” means “display”.

# Styling the Interface

- The components in the first flexbox should be vertically aligned to center.
- The two lists (Todos and Done) should occupy the same width.

```
<div class="d-flex mb-3">
  <label for="exampleFormControlInput1" class="form-label">Task</label>
  <input type="text" class="form-
control" id="exampleFormControlInput1" placeholder="Enter a task here">
  <button type="button" class="btn btn-primary">Add</button>
</div>
```

```
<div class="d-flex">
  <div>
    <h3>Todos</h3>
    <div id="todo-list">
      <div class="task">Sample task</div>
    </div>
  </div>
  <div>
    <h3>Done</h3>
    <div id="done-list">
      <div class="task">Sample task</div>
    </div>
  </div>
</div>
```



The screenshot shows a web application titled "SKKU-Todo". It features a form with a label "Task" and a text input field with the placeholder "Enter a task here". To the right of the input field is a blue button labeled "Add". Below the form, there are two sections: "Todos" and "Done". The "Todos" section contains a single task item labeled "Sample task". The "Done" section is currently empty.

# Styling the Interface

- Add “align-items-center” class to the first flexbox.
- Add “flex-grow-1” class to the wrappers of the two lists.

```
<div class="container">
  <div class="d-flex align-items-center mb-3">
    <label for="exampleFormControlInput1" class="form-label">Task</label>
    <input type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter a task here">
    <button type="button" class="btn btn-primary">Add</button>
  </div>

  <div class="d-flex">
    <div class="flex-grow-1">
      <h3>Todos</h3>
      <div id="todo-list">
        <div class="task">Sample task</div>
      </div>
    </div>
    <div class="flex-grow-1">
      <h3>Done</h3>
      <div id="done-list">
      </div>
    </div>
  </div>
</div>
```

## SKKU-Todo

Task

Add

## Todos

Sample task

## Done

# Styling the Interface

- Still, the interface looks very unorganized.
- This is because the space between components is inconsistent.
- Let's adjust padding and margin of components.

SKKU-Todo

Task

Enter a task here

Add

Todos

Sample task

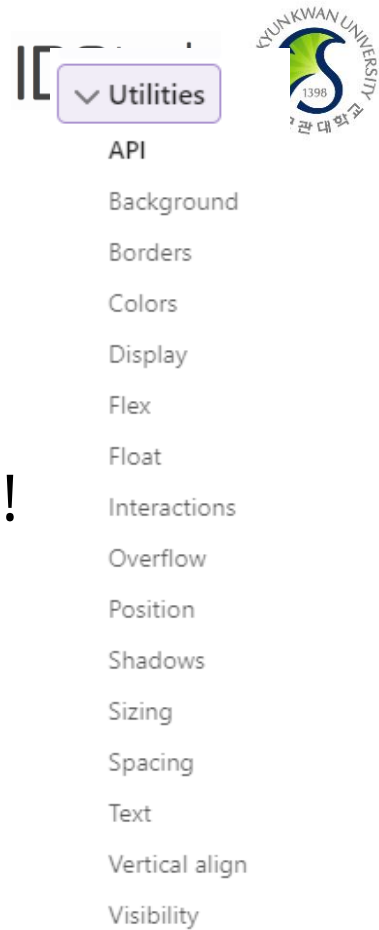
Done



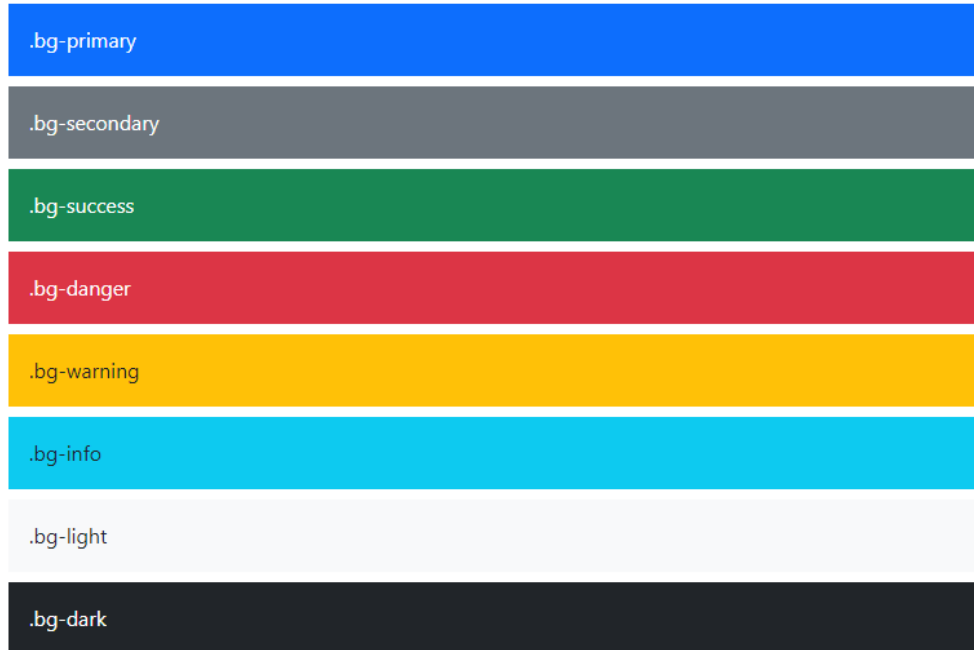


# Styling the Interface

- Bootstrap provides **utility classes** that are very useful to control the appearance of tags without adding CSS rules.
  - If you use these classes well, you don't have to use CSS rules at all!
- 
- Background colors
  - Borders
  - Colors
  - Padding and margin
  - Shadow
  - Text



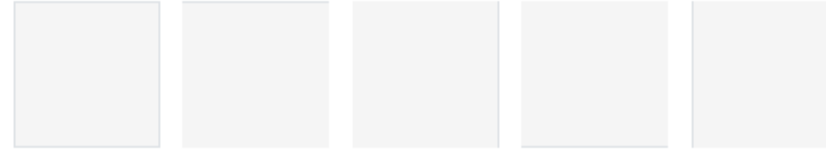
# Styling the Interface



.bg-body

.bg-white

.bg-transparent



```
<span class="border"></span>
<span class="border-top"></span>
<span class="border-end"></span>
<span class="border-bottom"></span>
<span class="border-start"></span>
```



```




```

# Styling the Interface

`.text-primary`

No shadow

`.text-secondary`

`.text-success`

`.text-danger`

Small shadow

`.text-warning`

`.text-info`

Regular shadow

`.text-light`

`.text-dark`

Larger shadow

`.text-body`

`.text-muted`

`.text-white`

`.text-black-50`

`.text-white-50`

```
<div class="shadow-none p-3 mb-5 bg-light rounded">No shadow</div>
<div class="shadow-sm p-3 mb-5 bg-body rounded">Small shadow</div>
<div class="shadow p-3 mb-5 bg-body rounded">Regular shadow</div>
<div class="shadow-lg p-3 mb-5 bg-body rounded">Larger shadow</div>
```

Copy

# Styling the Interface

---

- There are utility classes for padding and margin.
- {property}{sides}-{size}
- property: m (margin), p (padding)
- sides: t (top), b (bottom), s (start or left), e (end or right), x (left and right), y (top and bottom)
- size: integer from 0 to 5 where 1 = 0.25rem

# Styling the Interface

---

- “ms-1”: set the left margin to .25rem
- “px-2”: set the left and right paddings to .5rem
- “ms-1 ps-1”: set the left padding and margin to .25rem
- “m-1”: set the margin (all sides) to .25rem
- “m-0 p-0”: remove all padding and margin

# Styling the Interface

- “w-50” is equivalent to “width: 50%”.
- <label> was removed for simplicity.
- Note that I did not use any CSS rule at all.

```
<div class="container">
  <div class="d-flex align-items-center mb-2 mt-2">
    <input type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter a task here">
    <button type="button" class="btn btn-primary ms-1">Add</button>
  </div>

  <div class="d-flex">
    <div class="flex-grow-1 bg-light rounded-2 p-2 me-1 w-50">
      <h3>Todos</h3>
      <div id="todo-list">
        <div class="task bg-light p-1 rounded-2 ps-2">Sample task</div>
      </div>
    </div>
    <div class="flex-grow-1 bg-light rounded-2 p-2 w-50">
      <h3>Done</h3>
      <div id="done-list">
      </div>
    </div>
  </div>
</div>
```

SKKU-Todo

Task

Add

Todos

Done

Sample task

SKKU-Todo

Add

Todos

Done

Sample task

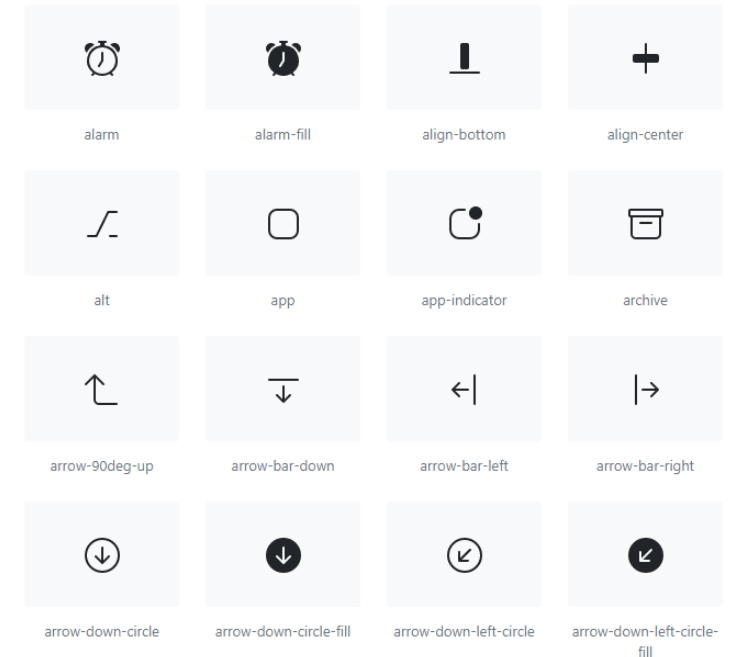
# Styling the Interface

- Let's give the finishing touch by adding icons.
- Bootstrap also provides an icon extension.
  - <https://icons.getbootstrap.com/>



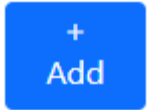
```
<!-- Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap... >
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.4.1/font/bootstrap-icons.css">
```

## Icons




# Styling the Interface

- If your button breaks like the image on the right, add “text-nowrap”.



```
<nav class="navbar navbar-light bg-light">
  <div class="container">
    <span class="navbar-brand mb-0 h1"><i class="bi bi-card-checklist"></i> SKKU-Todo</span>
  </div>
</nav>
<div class="container">
  <div class="d-flex align-items-center mb-2 mt-2">
    <input type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter a task here">
    <button type="button" class="btn btn-primary ms-1 text-nowrap"><i class="bi bi-plus"></i> Add</button>
  </div>
</div>
```

 SKKU-Todo

+ Add

### Todos

Sample task

### Done



# Adding Interaction

---

- Let's make our application ***interactive***.
- When the user clicks on the “Add” button, the text in the text box will be appended to the “Todos” list.
- To this end, we will use JavaScript.
- If the “Add” button is clicked,
- Read the task name in the textbox.
- Append the name to the list.
- Clear the textbox.

# Adding Interaction

- Before we add interaction, let's set up some ids to relevant elements.

```
<div class="container">
  <div class="d-flex align-items-center mb-2 mt-2">
    <input type="text" class="form-control" id="task-input" placeholder="Enter a task here">
    <button type="button" id="add" class="btn btn-primary ms-1 text-nowrap"><i class="bi bi-plus"></i>
      Add</button>
  </div>

  <div class="d-flex">
    <div class="flex-grow-1 bg-light rounded-2 p-2 me-1 w-50">
      <h3>Todos</h3>
      <div id="todo-list">
        <div class="task bg-light p-1 rounded-2 ps-2">Sample task</div>
      </div>
    </div>
    <div class="flex-grow-1 bg-light rounded-2 p-2 w-50">
      <h3>Done</h3>
      <div id="done-list">
      </div>
    </div>
  </div>
</div>
```

# Adding Interaction

---

- Let's rewrite our algorithm with ids.
  - If the "Add" button is clicked,
  - Read the task name in the textbox.
  - Append the name to the list.
  - Clear the textbox.
- If **#add** is clicked,
- Read the text in **#task-input**.
- Append the text to **#todo-list**.
- Clear **#task-input**.

# Event Handling

---

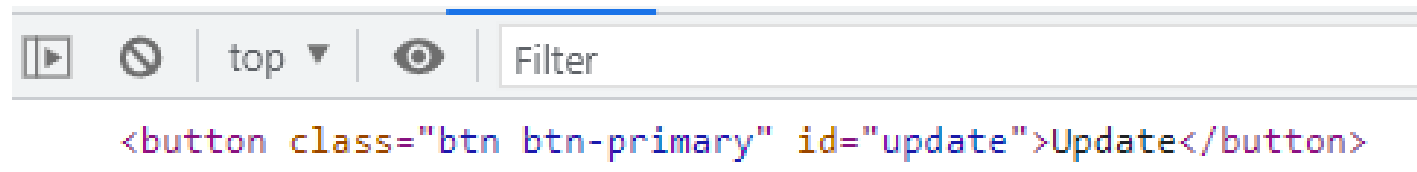
- We want to call a function when a click event happens.
  - A web page has finished loading.
  - An input field was changed.
  - A button was clicked.
  - ...
- We attach an **event handler** to an HTML element (*#add*).
- An event handler is a function that is called whenever the corresponding event occurs.

# Event Handling

- First, select the element to which the event handler is attached.
- *document.querySelector("#add")* returns an HTML element whose id is "add".
  - Same as selectors in CSS

```
<button class="btn btn-primary" id="update">Update</button>

<script>
  let updateBtn = document.querySelector("#update");
  console.log(updateBtn);
</script>
```



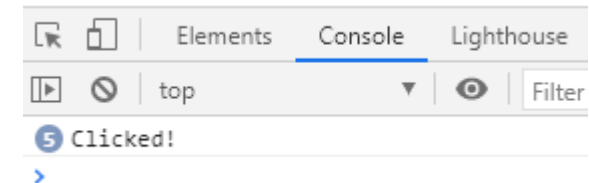
The screenshot shows a web browser's developer console. At the top, there are icons for 'Run', 'Disable', 'Top', and 'Filter'. Below these, the selected HTML element is displayed: `<button class="btn btn-primary" id="update">Update</button>`.

# Event Handling

- `(element).addEventListener(type, listener)` adds a function *listener* that will be called whenever the specified event occurs.
  - *type*: "click", "dblclick", "mouseenter", "mouseleave", "keydown", ...
  - <https://developer.mozilla.org/en-US/docs/Web/Events>

```
let button = document.querySelector("#add");

button.addEventListener("click", () => {
  console.log('Clicked!');
  // 1. Read the text in #task-input.
  // 2. Append the text to #todo-list.
  // 3. Clear #task-input.
})
```



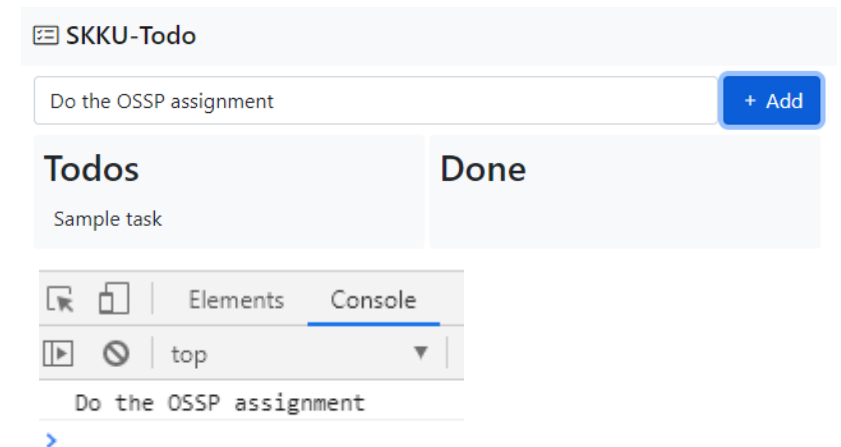
# Getting the Input

- To get the text in an input box, you first select the input box first.
  - `document.querySelector("#task-input")`
- `(element).value` contains the value of `element`.

```
let button = document.querySelector("#add");

button.addEventListener("click", () => {
  // 1. Read the text in #task-input.
  let input = document.querySelector("#task-input");
  let task = input.value;
  console.log(task);

  // 2. Append the text to #todo-list.
  // 3. Clear #task-input.
})
```



# Creating an Element

- *document.createElement(tagName)* creates a new tag in memory.
- You must append the created tag to an element that is on screen to make it visible.
- Don't forget to add classes for styling.


```
<div id="todo-list">  
  <div class="task bg-light p-1 rounded-2 ps-2">Sample task</div>  
</div>
```

```
button.addEventListener("click", () => {  
  // 1. Read the text in #task-input.  
  let input = document.querySelector("#task-input");  
  let task = input.value;  
  
  // 2. Append the text to #todo-list.  
  let newTask = document.createElement("div");  
  newTask.classList.add("task", "bg-light", "p-1", "rounded-2", "ps-2");  
  newTask.textContent = task;  
  
  let todoList = document.querySelector("#todo-list");  
  todoList.appendChild(newTask);  
  
  // 3. Clear #task-input.  
})
```



# Clearing the Input

- Finally, clear the input once a task is added.

 SKKU-Todo

+ Add

### Todos

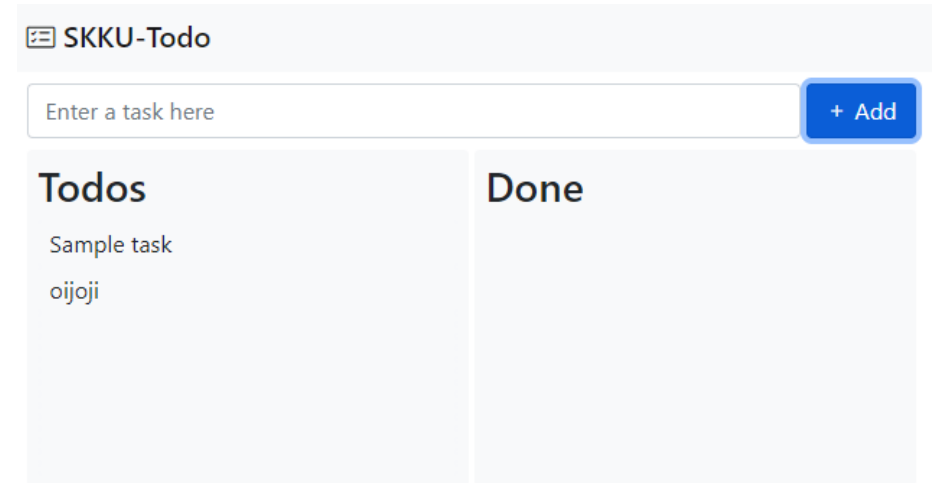
- Sample task
- Doing the OSSP assignment
- Working with Git

### Done

```
button.addEventListener("click", () => {  
  // 1. Read the text in #task-input.  
  let input = document.querySelector("#task-input");  
  let task = input.value;  
  
  // 2. Append the text to #todo-list.  
  let newTask = document.createElement("div");  
  newTask.classList.add("task", "bg-light", "p-1", "rounded-2", "ps-2");  
  newTask.textContent = task;  
  
  let todoList = document.querySelector("#todo-list");  
  todoList.appendChild(newTask);  
  
  // 3. Clear #task-input.  
  input.value = "";  
})
```

# Clearing the Input


- Empty tasks are added even though I didn't enter anything to the input box.
  - The list keeps growing!
- Let's add an if statement to filter out this case.



```
// 1. Read the text in #task-input.  
let input = document.querySelector("#task-input");  
let task = input.value;  
  
if (!task.length) return;
```

# Let's Publish

- Congrats! We just developed a Web app.
- Let's publish it to the Web.
- How? GitHub provides a free Web hosting service!

 SKKU-Todo

+ Add

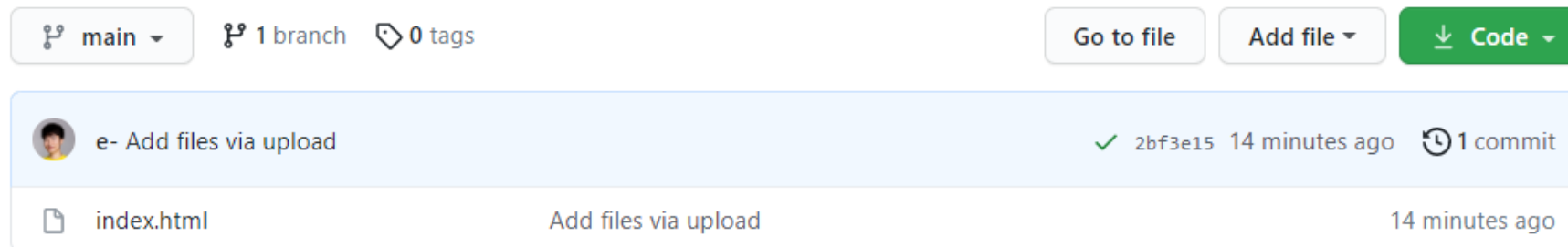
### Todos

- Sample task
- Doing the OSSP Assignment
- Working with Git

### Done

# Let's Publish

- Create a GitHub repository named “skku-todo”.
  - You can use a different name, but the name will be part of url.
- Make sure the name of the HTML file is “index.html”.
  - “index.html” means the landing page.
- Commit the HTML file to the repository.




# Let's Publish

- In your GitHub repository, go to Settings -> Pages.
- Set the source to **“Branch: main”**.
- Click on “Save”.


## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

 Your site is ready to be published at <https://e-.github.io/skku-todo/>

### Source

Your GitHub Pages site is currently being built from the `main` branch. [Learn more.](#)


 Branch: main ▼

 / (root) ▼

Save

# Let's Publish

- Web demo: <https://e-github.io/skku-todo/>
- GitHub repository: <https://github.com/e-/skku-todo>

 SKKU-Todo

+ Add

### Todos

- Sample task
- Doing the OSSP Assignment
- Working with Git

### Done

# Summary: Web App

- In this class, we developed a simple Web app for task management.
  - Next time, we will extend its features.
- **Bootstrap** is a collection of CSS styles (+ JS code) that can accelerate web development.
  - Container, navbar, button, forms, and many utility classes!
- **JavaScript** brings interactivity to your app.
  - There are special functions that bridge HTML and JS, such as *document.querySelector*.
  - *(element).addEventListener(name, handler)* sets up a function that is called each time a specific event happens.
  - Accessing the value of `<input>` tags via the *value* attribute.