Introducing Bootstrap 4

Jörg Krause

Apress®

Introducing Bootstrap 4

Jörg Krause

Apress®

Introducing Bootstrap 4

Jörg Krause
Berlin, Germany

ISBN-13 (pbk): 978-1-4842-2381-9 ISBN-13 (electronic): 978-1-4842-2382-6
DOI 10.1007/978-1-4842-2382-6

Library of Congress Control Number: 2016961809

Managing Director: Welmoed Spahr
Acquisitions Editor: Louise Corrigan
Development Editor: James Markham
Technical Reviewer: Massimo Nardone
Editorial Board: Steve Anglin, Pramila Balen, Laura Berendson, Aaron Black, Louise Corrigan, Jonathan Gennick, Todd Green, Celestin Suresh John, Nikhil Karkal, Robert Hutchinson, James Markham, Matthew Moodie, Natalie Pao, Gwenan Spearing
Coordinating Editor: Nancy Chen
Copy Editor: Judy Ann Levine
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global, image courtesy of Freepik.

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

This book is for everyone making their first steps into the world of software.
Or for those wish to bring their knowledge up to date.

The future of software development may be the web, the cloud, or where ever.
In any case, it is not on an isolated local system.
This book is part of a project, a set of textbooks, to help you
fight the challenges of web development.

The subjects are not necessarily new, but do form a unique subject.

Contents at a Glance

Contents

About the Author

Jörg Krause works as a trainer, consultant and software developer for major companies worldwide. Build on the experience of 25 years of work with web and many, many large and small projects. Jörg believes it is especially important to have solid foundations. Instead of always running to create the latest framework, many developers would be better advised to create and provide a robust foundation.

Jörg has written over 50 titles in the renowned and prestigious specialist publishers in German and English, including some bestsellers. Anyone who wants to learn this subject compact and fast, its right here. On his website `www.joergkrause.de` much more information can be found.

About the Technical Reviewer

Massimo Nardone has more than twenty-two years experience in security, web/mobile development, cloud, and IT architecture. His true IT passions are in the security and android field.

He has been programming and teaching how to program with android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than twenty years. He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a project manager, software engineer, research engineer, chief security architect, information security manager, PCI/SCADA auditor, and senior lead IT security/cloud/SCADA architect for many years.

His technical skills include: security, android, cloud, Java, MySQL, Drupal, Cobol, Perl, web, and mobile development; MongoDB, d3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, django CMS, Jekyll, Scratch, and many others.

He is currently working as the chief information security officer (CISO) for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Massimo has reviewed more than forty IT books for different publishing company, and he is the coauthor of *Pro Android Games* (Apress, 2015).

Introduction

Bootstrap 4 – the new CSS Framework

This book describes in a compact and clear form the CSS Framework Bootstrap 4. Bootstrap was originally designed, around 2010, on Twitter and was from the outset developed with the idea of "Mobile First" – first for mobile devices . It has since grown to become one of the best and most popular frameworks for the design of websites. Accompanied by a large number to bootstrap engaging design templates – so-called themes – it offers secure and robust style definitions for everyday tasks.

Target Audience

This book is aimed at beginners and web developers who are new to the web world. Bootstrap serves mainly the front-end developer. It is a series of CSS instructions and these additional JavaScript (ECMAScript) libraries, apply where CSS alone is not enough.

Maybe you are also a web designer, who discovered Bootstrap as an excellent way to upgrade your web pages with dynamic elements. Here, you are dealing with texts, with forms, with the presentation of database content, i.e. everything that constitutes a dynamic web site. Then you will need to know one of this band, namely for the creation of a professional user interface, in particular a clear form.

In any case, I tried not to ask an prerequisites or conditions to the reader. You do not need to be a computer scientist, not in perfect command of language, don't need to know rocket science. No matter in what context they have encountered on Bootstrap, you will be able to read this text.

 **Bootstrap** For all examples to understand, you need a working environment for creating web pages. This can be Ruby on Rails, PHP, ASP.NET or Node.js. I recommend node.js for an easy start. It's a nice way to get Bootstrap up and running.

If you have accidentally found this text and cannot do anything with the term "Bootstrap", read it anyway. You will be learning one of the most modern techniques of web development and the future belongs to the circle of excellent developers who can be build good- looking and device-independent sites.

What You Should Know Readers of this series have hardly any requirements. Some HTML cannot harm and who already have seen a static HTML page (the source code, of course) is certainly good and pure. I assume that you have at least a current operating system, on which you will find an editor with which you can create web pages.

Examples

You can find the sample project to this book on Github:

- *https://github.com/joergkrause/Bootstrap4-Book*

The project uses sample files contain the pure HTML. It works best with Visual Studio, but can largely be used independently.

As You Can Read This Text

I will not dictate how you should read this text. In the first draft of the structure, I have tried several variations and found that there exists no ideal form. If I were to orientate myself to the various types of applications, the text would divided into several chapters that are not interrelated. The one or the other readers would then be annoyed that he put a lot of money for a book, where only a fifth is usable. This band solves the problem by reducing it to a very small issue is focused and no "blah-blah" for the inflation of the volume is.

Beginners should read the text as a narrative from the first to the last page. Those who are already somewhat familiar, the less interesting sections for him, he can safely skip sections. If references are required, I have included the corresponding appropriate cross- references.

Realizations

The theme is set not technically easy to master, because scripts are often too extensive and it would be nice if you could support the best optical reading form. I have therefore included extra line breaks used to aid readability, but have no place in the editor of their development environment.

In general, each program code is set to a non-proportional font. In addition, scripts have line numbers: :

```
1   body {
2     color: black;
3   }
```

If you find you need to enter something in the prompt or in a dialog box, this part of the statement is in bold:

```
$ bower install bootstrap
```

The first character is the prompt and is not entered. I use in the book, the Linux prompt, the bash shell. The commands will work, without any exception, unchanged even on Windows, only then STOP **C:>** or something similar at the beginning of the line.

Expressions and command lines are sometimes peppered with all types of characters, and in almost all cases, it depends on each character. Often, I'll discuss the use of certain characters in precisely such an expression. Then the "important" characters with line breaks alone and also in this case, line numbers line numbers are used to reference the affected symbol in the text exactly (note the : character in line 2):

```
1   a.test {
2     :hover {
3       color: red
4     }
5   }
```

The font is non-proportional, so that the characters are countable and opening and closing parentheses are always among themselves.

Symbols

To facilitate the orientation in the search for a solution, there is a whole range of symbols that are used in the text.

Tip This is a tip

Information This is an information

Warning This is a warning

CHAPTER 1

It Begins with CSS

Before you begin to deal with bootstrap, you should understand CSS. As an introduction to this book, there is a short refresher to CSS.

CSS (cascading style definitions)—is a layout and formatting language for creating and formatting markup languages such as HTML. Ideally, the HTML document contains only semantic information, and with CSS this is then formatted in design and typography.

HTML already brings some basic formatting, such as a larger font for headlines, which can be customized with CSS as well as the unformatted elements. Format tags in HTML formatting and their formatting attributes generally should not be used. They are obsolete in HTML 5. It is replaced by CSS.

With CSS, it is also possible to define output types separately for different media such as a monitor (screen), projection (projection), and printing (print).

Basics

An HTML document consists of semantically meaningful awards for headings, paragraphs, lists, and so forth. The CSS statements must be placed in such a way that the browser can assign these elements.

Basically, there are three ways to store these messages and they are as follows:

- The `style`-attribute that any HTML elements knows.
- The element `<style></style>`, which summarizes several style rules.
- The `<link />`-element that references a file that contains multiple style definitions

The preferred way is to use a CSS file. This can be done in the browser cache and the content can be displayed with the appropriate tools. Then you can reduce the bandwidth savings (not because of the bandwidth, but because of the associated performance gain).

Local `<style>`-elements should be used only in exceptional cases, for example, to temporarily change any possible complex files. Such local style instructions have a higher priority. The `style`-attribute expands or modifies the styles once per single element. It has the highest priority over any conflicting rules from the local or imported styles.

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_1

The link to the CSS file takes place in the head section of the HTML document:

```
1   <link rel="stylesheet"
2         type="text/css"
3         href="styles/style.css">
```

Note that the path to the file must be specified relative to the HTML document.

Syntax

The syntax of CSS is relatively simple. The basic structure consists of two building blocks:

1. Selector
2. Ruleset

The selector determines which element or elements the rules apply to.

```
1   Selector {
2     Ruleset
3   }
```

When styles are in `style`-attributes, they are only valid for that item, therefore the selector is eliminated.

The standard rate turn (ruleset) consists of rules. These are to be written in the following form:

```
Style: Parameter;
```

The semicolon at the end is required.

Selector

The selector is the instrument that allows the elements on the page can be achieved selectively. The whole scheme of selectors is fairly comprehensive. At this point, first, the most important building blocks are shown.

Elements (Tags)

With CSS, you can address individual items. The syntax for this is as follows:

```
TagName { Ruleset }
```

If you have all of the elements <p> want to achieve, it is enough to write the following:

```
p { }
```

IDs

Often, a single tag is formatted. The tag must be identifiable, that is, it must have an ID that is accessible. The same HTML attribute contains a string that is achieved in the CSS properties. An ID, however, may only be applied to one element within a document and must therefore occur only once. The syntax for this as follows:

```
#id { }
```

An example of a button looks like this:

```
1   <style>
2   #send {
3     color: red;
4   }
5   </style>
6   <button id="send">Send</button>
```

Classes

Frequently several elements are addressed. These are classes that are written in HTML in the class attribute. You can attribute this to multiple space-separated classes, by combining rulesets.

This saves extensive definitions. Bootstrap utilizes this possibility comprehensively and achieves this with a few rulesets and a large number of modification options. Unlike IDs, they can define many classes of elements, and therefore, they may occur more than once in a document.

The syntax can look like this:

```
.class { }
```

Another example of a button:

```
1   <style>
2   .btn {
3     color: red;
4   }
5   </style>
6   <button class="btn">Send</button>
```

Attributes

Attributes can be referenced using the following syntax of HTML elements:

```
[name] { }
[name="value"] { }
```

An example of another button:

```
1   <style>
2   [data-item] {
3     color: blue;
4   }
5   </style>
6   <a href="link.html" data-item="22">Next</button>
```

If the value of the attribute (to the right of the = sign) is not specified, then only the existence of the attribute is considered sufficient to apply to the rules.

Logical Selection

It often happens that rulesets are to be applied to multiple selectors. For this purpose, an OR logic is required, which is noted as a comma in CSS:

```
a, b { }
```

Between `a` and `b` there is no link, the rule is applied independently to both. The placeholders `a` and `b` in the example may be more complex selectors.

More Selectors

In practice, these selectors are not enough. Table 1-1 provides a compact overview of all the other forms.

Table 1-1. *Simple CSS Selectors*

Symbol	Description
*	Universal / all
`Tag`	Element's name
`.class`	Class (Attribute class)
`#id`	ID (Attribute id)
`[a]`	Attribute presence
`[a=v]`	Attribute value
`[a~=v]`	Attribute contains a value as a standalone word
`[a|=v]`	Attribute contains no value
`[a^=v]`	Attribute starts with value
`[a$=v]`	Attribute value ends with
`[a*=v]`	Attribute contains a value

Dealing with hierarchies is essential because HTML documents are hierarchies, which are often called trees. Figure 1-1 shows the relationship between the elements in the document tree.

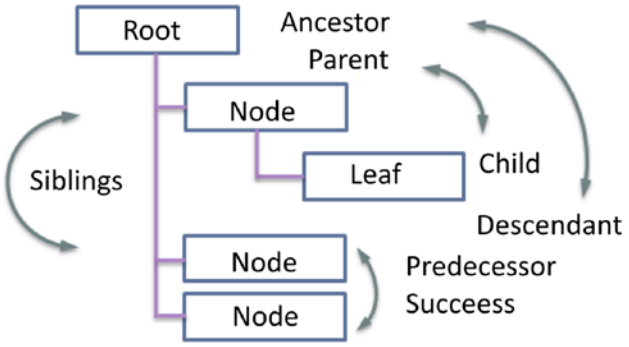


Figure 1-1. *Elements of the hierarchy of a HTML page*

Table 1-2 shows the syntax for CSS.

Table 1-2. *CSS Selectors for Hierarchies*

Symbol	Description
`e > f`	Selection if f is a child element of e
`e f`	Selection, if f is a descendant of e
`e + f`	Selection, if f is a successor of e
`e ~ f`	Selection when e is sibling of f

In contrast to the possibilities of relationships missing selectors for ancestors, predecessors, and parents; you can do this by swapping the elements.

Pseudoselectors are those that have no comparable representation in HTML, but result from the position of elements or use. There are three types of pseudoselectors:

- Static positions (see Table 1-3 for a list of static selectors)
- Selection of areas (see Table 1-4 for a list of CSS selectors for areas)
- Dynamic behavior (see Table 1-5 for a list of dynamic CSS selectors)

Table 1-3. *Static CSS selectors*

Symbol	Description
`::first-line`	First line
`::first-letter`	First letter
`::before`	Before the element
`::after`	After the element
`::selection`	The highlighted (selected) area

Table 1-4. *CSS Selectors for Areas*

Symbol	Description
`:root`	Basic element
`:empty`	Applies only if the item is empty
`:first-child`	The first child element of a list
`:last-child`	The last child element of a list
`:nth-child()`	A particular child element of a list
`:nth-last-child()`	A particular child element by the end of a list
`:only-child`	Valid only when there is only one child element
`:first-of-type`	First child element of a type
`:last-of-type`	Last child element of a type
`:nth-of-type()`	Child element of a type in a list
`:nth-last-of-type()`	Child element of a type by the end of a list
`:only-of-type`	Only this type from a list

Table 1-5. *Dynamic CSS Selectors*

Symbol	Description
`:link`	A hyperlink
`:visited`	A hyperlink that has already been visited
`:hover`	A hyperlink to the hovering the mouse
`:active`	A hyperlink that is active (clicked)
`:focus`	An item that has the focus (blinking cursor)
`:target`	An item that has a target attribute

(*continued*)

Table 1-5. (*continued*)

Symbol	Description
`:disabled`	An item that is disabled (disabled attribute)
`:enabled`	An item that is enabled (not disabled attribute)
`:checked`	An item that is checked (only checkbox)
`:valid`	An element that is valid
`:invalid`	An element that is invalid
`:lang()`	An item that the appropriate `lang=""` attribute is
`:not()`	Negates the following selections (this is an operator)

The examination of the validity of form elements presupposes that they were defined in HTML 5 using attributes such as `maxlength`, `required`, `date`, `email, and so forth`.

Unlike the `lang` attribute the function`lang()` in CSS can determine a fallback to a default culture, that is, responding to the indication of “en-US” on “en”.

Browser Support

No browser currently sees each of the pseudoclasses shown. The online documentation provides up-to-date information about support that is available at that moment.

The Box Model

HTML has two types of forms of representation for elements: flow and block. Flow elements embed themselves in the running text. These elements have dimensions such as width and height, for they depend on the surrounded content. On the other hand, block elements’ dimensions cause adjacent elements to be displaced from their occupied space, and they can define the width and height they occupy by themselves. Moreover, the displacement behavior is highly customizable, up to the desired superposition. With special rules, elements that are actually floating elements are misidentified as block elements. This also can be reversed.

Figure 1-2 shows an example of a box model with the defined properties of the block elements for almost all rectangular areas.

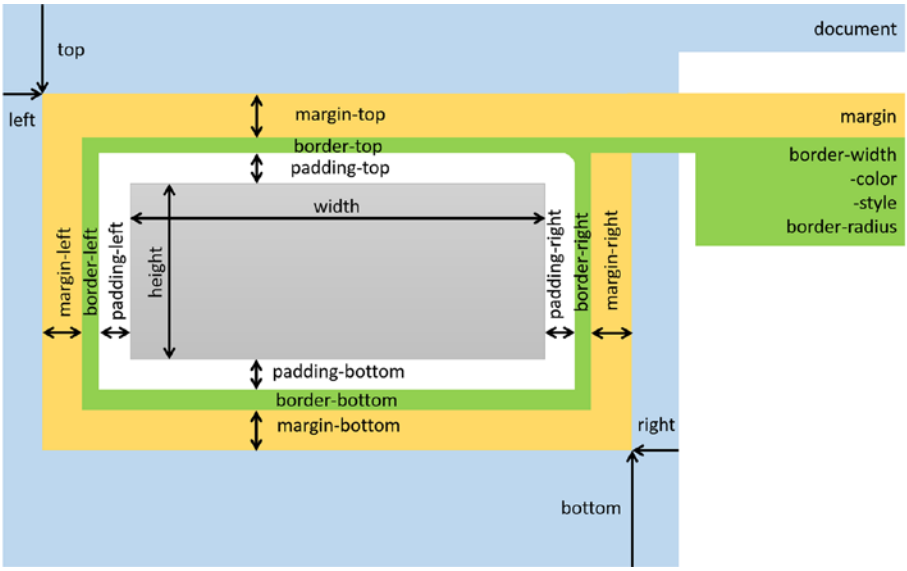


Figure 1-2. *Blocks of a box model*

It is important to realize that the indication of the width and height are not the final dimensions, but the dimensions of the content area mapped. When a frame is enclosing the box, the frames width needs to be charged twice for the calculation of the final width, which makes the box' dimensions on all sides equal to:

```
Width = Margin * 2 + Frame width * 2 + Distance * 2
```

```
Height = Edge * 2 + Frame width * 2 + Distance * 2
```

As there are differences between the borders, frames widths, and spacings, the calculation is accordingly complex.

Blocks of the Box

The building blocks of the box serve individually to prove each part with values. The inner portion refers to the content. The building blocks are:

- `Padding`: The inner distance
- `Border`: The frame
- `Margin`: The outer edge—distance from other elements

Because it is a rectangle within the box, four values are indicated:

- `Top`
- `Right`
- `Bottom`
- `Left`

Counting

The coordinate origin begins at either your screen and/or printed page on the top left. Some rules can be equal to specify multiple values. In such cases, the four values are interpreted in the order shown—starting with the `top` (upper left) and then continuing clockwise.

The `margin` that formed the horizontal distances is absolutely valid. Vertical contrast, the distances under certain circumstances may coincide (collapse). This occurs when no frame (`border`) or spacing (`padding`) is used and no exemption (`clear`) takes place. Here then the lower edge of the upper box is overlaid with the upper edge of the lower box. If there are edges of different size, the wider edge is taken as the whole edge.

Exceptions

From the reunification rules of the edges there are a number of exceptions. For more information, especially for more complex pages, the official documentation on CSS.

The Box Model in CSS3

In an extension of the CSS3, a box model was introduced that allows more flexibility in allocation. Using the `box-sizing` property lets you specify to which extent details are available even from `width` or `height`. Permits are one of the following:

- `content-box`: Default, data is only valid for the content.
- `padding-box`: Information applies to the content and padding.
- `border-box`: Information applies to content, padding, and frame.
- `inherit`: `box-sizing` of the parent element is copied (inheritance).

Compliance with Media

With CSS you may set the presentation of a document for different output media. Media queries are used to assign a stylesheet to a medium.

A media query list for criteria must meet an output medium so that a stylesheet is included for processing. Media queries consist of a media type (e.g., screen or printer), a media characteristic (such as color capability), or a combination of both. By using the possibility of a combination you can tailor a variety of output media stylesheets.

Media queries cannot be quoted in `style`-attributes.

Syntax

Specifying a data type is a simple keyword, for example, `screen`.

If no media query specified or the specified query consists only of only spaces, the default is `all`. Listing 1-1 shows a media query in a HTML document.

Listing 1-1. Media query in the HTML document

```
1   <link rel="stylesheet" href="monitor.css" media="screen">
2   <link rel="stylesheet" href="printer.css" media="print">
```

The `print` media type ensures that the stylesheet *printer.css* is used only when printing. On the other hand, on a screen (`screen`) *monitor.css* is activated.

This procedure has the disadvantage that both stylesheets often contain the same CSS rules. In addition, at least two files are needed.

You also can omit the attribute `media`; then the stylesheet in question applies to all media. Then the only changes need to be stated in the alternative file (see Listing 1-2).

Listing 1-2. Media query with standard document

```
1   <link rel="stylesheet" href="monitor.css">
2   <link rel="stylesheet" href="printer.css" media="print">
```

The rules alternatively can be accommodated directly in the CSS file (see Listing 1-3):

Listing 1-3. Rules for printing

```
1   @media print {
2     /* Rules for printing */
3   }
```

Media have certain characteristics that modify the selection rule. With a screen this can be the number of pixels, for example. Prefixes, such as `min-` and `max-`, allow you to specify areas.

```
1   <link rel="stylesheet" href="pt.css"
2         media="(orientation: portrait)">
```

The stylesheet *pt.css* is included if the contents of the web pages in portrait orientation (portrait) are issued.

```
<style type="text/css" media="(color)">
  /* color. */
</style>
```

The color specified in the `style`-element are processed when the output medium is set to reproduce colors. A black and white printer would then benefit from not having any nonreadable colors on output (e.g., yellow on white).

```
@import 'layout.css' (min-width: 150mm);
```

The stylesheet *layout.css* will apply when the display area of the output medium is at least 150 mm.

Media queries can be grouped with a logical OR. The comma is used the same as the CSS selectors. Grouped queries are completely independent. Apply one of the queries at least once when the declarations are applied.

```
@media print, embossed {
  /* Formats for print media. */
}
```

In this example, a stylesheet is specified, which can be used both for `print` media type and the `embossed` media type.

Several media features may be associated with the `and`. A stylesheet will only be processed if all of the membership criteria are met.

```
@media (min-width: 130mm) and (max-width: 160mm) {
  /* compact Layout */
}
@media print and (color), screen and (color) {
  /* color */
}
```

If the media type listed at the beginning of a media query (as in this note), the keyword `only` or `not only` hides the media query before browsers that do not support this and related combinations. Otherwise, the query is processed as if it was not there. If a media query operator is `not` prefixed, the query is denied.

```
@media only all and (min-width: 150mm) {
  /* Layout */
}
@media not all and (monochrome) {
  /* color */
}
```

This example shows how screens are assigned to a minimum 150 mm wide display area rule. A browser that understands media queries, ignores the *`only`* keyword. By denying the query *monochrome*, the media that can deal with color can be utilized.

Dealing with Units

A special concern is one relative to the lengths *em* or *ex*. In processing these values the default value of the browser is assumed, which has been defined by the user. Normally *em* refers to the current font, which has not yet been defined at the level of the media query. More explanations about units can be found at the end of this chapter.

Each feature also can be used without a value. In this case, it is determined whether the feature is present on the medium used.

```
@media (width) {
  /* The output medium has the characteristic "width" */
}
@media (color) {
  /* The output medium has the characteristic "color capability" */
}
```

Parameter

The feature `width is` described in continuous media as the width of the display area (viewport) and in paged media as the width of a page. The prefixes *min-* and *max-* are allowed to specify limits.

```
@media (width: 60em) {
  /* width corresponds exactly 60em */
}
@media (min-width: 50em) {
  /* width is at least 50em */
}
@media (max-width: 70em) {
  /* width shall not exceed 70em */
}
```

When users cannot be predicted, it is almost always a good idea to use one of the possible prefixes as the actual display width with features that relate to the display area,

The feature `height is` described in continuous media as the height of the display area (viewport) and in paged media as the height of a page. The prefixes *min-* und *max-* are allowed to specify limits.

The features `device-width` (device's width) and `device-height` (device's height) describe the width or height of the output device (e.g., the width of the screen in pixels). The value is a positive length specification. The prefixes *min-* and *max-* are used to specify limits.

```
@media (device-width: 800px) {
  /* width corresponds exactly to 800 Pixel */
}
@media (min-device-width: 800px) {
  /* width is at least 800px */
}
@media (max-device-width: 1024px) {
  /* width is not more than 1024px */
}
```

Even if an output device has certain dimensions, this does not guarantee that the available range is also used. Also, the reported number of the pixels differ from the physical pixels—that is, for example, maybe better the case with retina displays. Likewise, for all devices that do not report their orientation the values may not change. For Apple, the width stays the same in portrait mode—even if the user rotates the tablet and use in landscape format. Thus, you must be aware of orientation.

The feature `orientation` (orientation) describes the page format of an output medium. The orientation corresponds to the value `landscape` (horizontal), when the value of the feature `width` is greater than the value of the feature `height`. Otherwise the orientation corresponds to the value `portrait` (vertical). The value is one of the either `portrait` or `landscape`.

```
@media (orientation: portrait) {
  /* formats for portrait format output media */
}
```

The characteristic `aspect-ratio` (aspect ratio) is the ratio of the feature `width` to feature `height`. The value is a ratio value. The prefixes *min-* and *max-* are used here.

```
@media (aspect-ratio: 4/3) { /* Case 1 */ }
@media (min-aspect-ratio: 4/3) { /* Case 2 */ }
@media (max-aspect-ratio: 4/3) { /* Case 3 */ }
```

In this example, the ratio value is assigned to the 4/3 variants of `aspect-ratio`-feature. The stylesheet is processed if the aspect ratio of the display area (viewport) is 4 to 3 (Case 1).

That is the case for the example with a display area of 492 to 369 pixels.

The stylesheet in Case 2 is applied if the aspect ratio is 4/3 or greater (e.g., 5/3 or 6/3). In Case 3, the stylesheet is processed only when the aspect ratio is 4/3 or less (e.g., 2/3 or 1/3).

The feature `device-aspect-ratio` (device's aspect ratio) describes the ratio of the characteristic `device-width` to the feature `device-height`. The application is similar to the feature `aspect-ratio`.

The feature of `color` describes the number of bits that a device for each color component (i.e., red, green, or blue) is used. If the output device can display their colors, then the value 0 (zero) is true. If different color components use different number of bits, then count the lowest bit number the device used. The value is a non-negative number.

```
@media (color: 2) { /* Simple color layout */ }
@media (min-color: 3) { /* complex color layout */ }
@media (max-color: 2) { /* Simple color layout */ }
```

The characteristic `color-index` describes the number of color definitions in the color table of the output medium. If the media does not have a color table, the value 0 (zero) is true. Usually, the media only has a color chart where the color capability is limited.

```
@media (color-index: 16) {
  /* exactly 16 colors available */
}
@media (min-color-index: 20) {
  /* At least 20 colors are available */
}
@media (max-color-index: 256) {
  /* more than 256 colors are available */
}
```


Support for `color-index` is currently deficient in all browsers.

The feature `monochrome` (black and white) describes the number of bits that are used to describe a black-white hue. If it is not a device that can only display grayscale (also colors), then the value 0 (zero) is true.

```
@media (monochrome: 1) {
  /* only black and white are available */
}
@media (min-monochrome: 4) {
  /* At least 16 gray levels are available */
}
@media (max-monochrome: 8) {
  /* more than 256 gray levels are available */
}
```

The characteristic `light-level` (brightness) describes the ambient lighting conditions, which are detected by the brightness sensor of a camera. The following settings are possible:

- `dim`: Dim lights
- `normal`
- `washed`: Bathed very bright, in light

No firm Lux values are made intentionally as many devices are too different and have their own contrast adjustment technologies (e-ink remains readable in bright light, while liquid crystal displays do not provide more contrast). Because brightness sensors are not often calibrated, the result is difficult to predict.

```
@media (light-level: normal) {
  p {
    background: url("texture.jpg");
    color: #333 }
}
@media (light-level: dim) {
  p {
    background: #222;
    color: #ccc }
  }
@media (light-level: washed) {
  p {
    background: white;
    color: black;
    font-size: 2em; }
}
```

The feature `pointer` (arrow) changes the accuracy of the indication.

Otherwise it is often difficult to distinguish between touch devices on the one hand (smartphones and tablets, as well as consoles like the Wii) and devices with a mouse, touchpad, or digital pen are taken. The following settings are possible:

- `fine`: For devices with mouse, touchpad, or stylus
- `coarse`: For devices with touch or gesture control
- `none`: Only keyboard input possible

```
@media (pointer: coarse) {
  input {
    padding 1em;
    font-size: 2em;
  }
}
```

With touch devices, the font size and the padding of the input field need to be increased accordingly.

Because most devices have multiple input options, their use cannot be predicted. It is possible to use `any-pointer` to find out whether if there is any input device at all.

You can use `pointer` and `any-pointer` if you are unsure about certain devices. However, you can ascertain this in combination with the display width, whether someone is on the go via smartphone, tablet, or desktop computer. This allows you to optimize web pages for specific devices accurately.

The feature `resolution` describes the resolution, which is the density of the pixels on an output medium. If the output medium does not use rectangular pixels (e.g., printers), then the characteristic may be used only in conjunction with a prefix.

In this case, consider queries with `min-resolution` the lowest possible and queries with `max-resolution` the highest possible density of the pixels of the output medium.

```
@media (resolution: 96dpi) {
  /* The resolution is 96 pixels per inch */
}
@media (min-resolution: 200dpcm) {
  /* The resolution is at least 200 points per cm */
}
@media (max-resolution: 300dpi) {
  /* The resolution is a maximum of 300 dots per inch */
}
```

The feature `scan` describes the screen layout of output devices of the type *tv*. This can be done progressively, which is approximately the layout on a computer screen. Then the value applies to progressive, or means of banding (i.e., individual image lines are gradually displayed), and then it applies the value `interlace` to it. Allowed values are `progressive` or `interlace`.

```
@media (scan: progressive) {
  /* Screen Layout */
}
```

The feature `grid` (height) describes the scanning feature of output media. If output devices that represent the contents in a grid, then apply to it a value of one (1), otherwise the value zero (0).

```
@media (grid: 0) {
  /* Numerous font formatting */
}
```

The Viewport

Among the foundations of development for mobile devices, the meta-element is `viewport`. This one line of HTML code ensures correct scaling of the site on the first call on the mobile device.

The browser of the mobile device go first even assuming that sites are not designed for them and the site-wide, the display width exceeds considerably. The browser viewport (the display area) is therefore set, for example, in Mobile Safari to a width of 980 pixels, so most sites are fully visible. This has the disadvantage in that the content is very small, and fonts are illegible. The user must zoom in to see anything.

Set Viewport

The adjustment of the viewport can be very easy to customize via an HTML element. You can adjust the viewport of any website that does not have a standard width.

This way you can ensure that content and display area match—for narrower layouts so no unnecessary space is wasted. The site will be shown in the maximum possible size.

As seen in Figure 1-3, start with the view from *apple.com* on mobile devices (left) and then see the zoomed section with legible words (right).

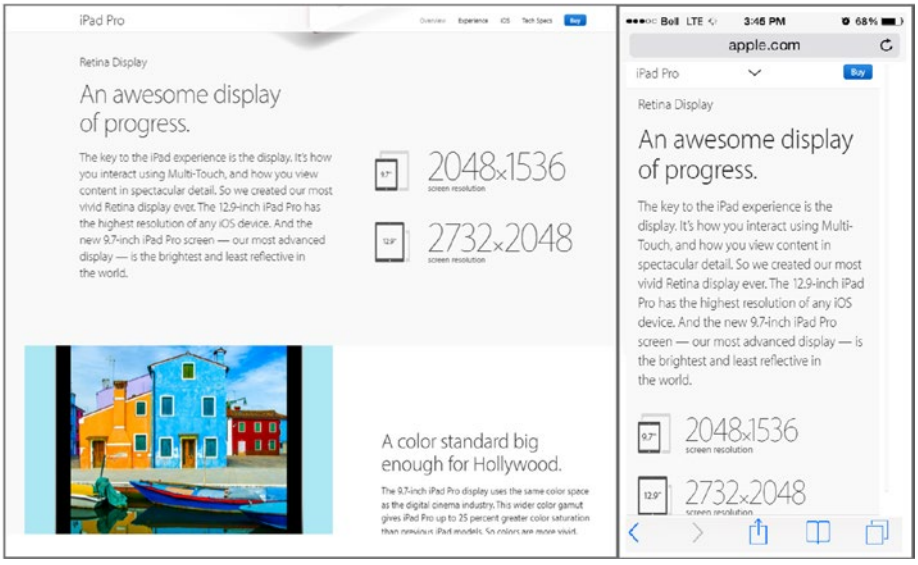


Figure 1-3. *Website with and without zoom*

For the amendment to add the following line in the header of the page, this is evaluated by mobile devices.

```
1   <!DOCTYPE html>
2   <head>
3       <meta name="viewport" content="width=1024" />
4   </head>
5   <body>
6   </body>
```

If the site is specifically created or optimized for mobile devices, you do not-have to specify a fixed width for the viewport. A smartphone has, for example, in portrait format a logical width of 320 px and 480 px in landscape mode (physically, the value will be higher). This would mean that the same content would be shown only in a different zoom level in portrait and landscape modes.

Instead, a formula is used to convert the modes:

```
Width of the Viewport = Width of Device
```

The smartphone now has a width of 320 px in portrait orientation, so precisely 320 px on the site are shown (1:1). Likewise 480 px will then be shown in landscape orientation. This flexible approach is both device-independent and allows the landscape mode to have all the extra space that makes sense in width.

Figure 1-4 demonstrates the effect. Figure 1-4 shows the view of a mobile device with a meta-element with a source code *width=device-width* (left) and without (right image).

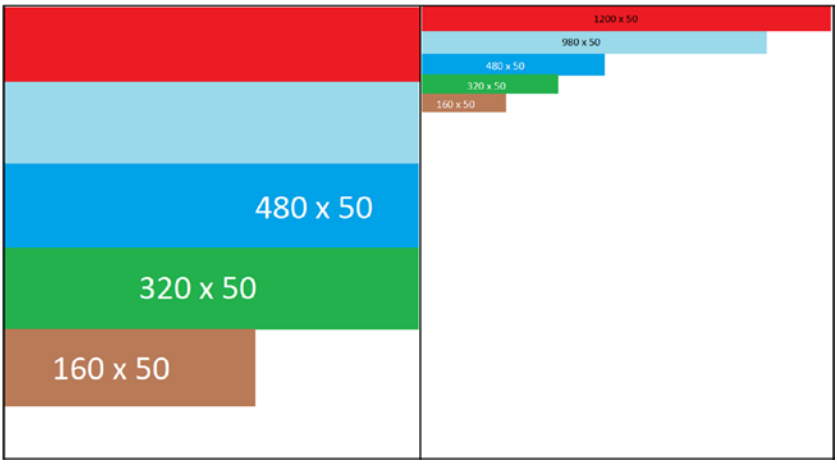


Figure 1-4. *View with and without meta-element*

Viewport Settings

The meta-element for the viewport has in addition to the width more properties that are listed comma-separated.

```
1   <meta name="viewport" content="width=device-width,
2                                  initial-scale=1.0,
3                                  user-scalable=no" />
```

Initial-Scale

The value specifies the initial zoom level. A 1.0 means that the content of 1:1 will be shown. For example, on a screen with 320 px, a 320 px-wide graphic will fill the entire width (see also the screenshot in Figure 1-3). Therefore, a 2.0 results in a 2x magnification.

User-Scalable

With this attribute, you can define whether the user can zoom in on the page (yes) or not (no).

Minimum-Scale and Maximum-Scale

These two properties make it possible to limit the zoom level. Set the maximum scale to 2.0, the content can be increased maximum 2x times.

Units

CSS units express a length specification. This is needed for widths, heights, spacing, margins, and so forth. Unit information consists syntactically from a number and a unit. For the number 0 the unit can be omitted. There are two types of units: absolute and relative.

Absolute Units

Absolute units are as follows:

- `cm`: Centimeters
- `mm`: Millimeters
- `in`: Inches
- `px`: Pixels
- `pt`: points
- `pc`: Picas

Typographic Details

Typographic units such as points and picas have been taken over from the era of paper printing. With paper printing you could define the exact font height when printing: 1 pica is 12 points, 1 point is 1/72 of an inch. In today's world of screens with many sizes, widths and resolutions such information is largely pointless.

The relationship between pixel (screen dot) and physical space is set by Windows as 1 inch = 2.54 cm = 96 pixels. Standard devices with standard definition deliver a ratio of 1 device pixel = 1 pixel. High resolution devices, such as printers or retina displays provide *n* = 1 device pixel pixels. Table 1-6 gives some examples.

Table 1-6. *Media Query in the HTML Document*

System	Resolution	Device pixels per pixel
Mac	72	1
Windows	96	1
Mobile low	120	1
Mobile medium	160	2
Mobile high	240	2
Retina	300	3

If wish to find out the exact resolution, you can do this by specifying the screen width and height and the size of the screen. At 4.65 inches (smartphone) and 1,280 × 720 pixels it is given by the Pythagorean Theorem:

$$\text{sqrt}(1{,}280^2 \times 720^2) \,/\, 4.65 = 315.8$$

This is rounded up to 316 and generally marketed as 320 dpi. Divided by 96 gives a ratio of 1:3.33, which rounded 3 device pixels per pixel results.

Generally, absolute data should be used only if the output can be determined with certainty. This is possible basically only with printers.

If you need an absolute unity, you should resort to *px* when you use a screen. On a printer you use most *mm* or *pt*.

Relative Units

Relative units use a specific starting point of because it calculates only ratios. Available are the following units:

- `em`: Unit of font size, based on the height in pixels (1 em is the size in pixels of the base fonts letter "M")
- `ex`: Unit of font size, based on the height of the small letter "x"
- `ch`: Unit of font size, based on the width of the digit "0"
- `rem`: Unit of font size, based on the width of the small letter "m" of the root element of the page (body)
- `vw`: Relative to 1% of the width of the viewport (46 cm screen width is 1 VW = 0.46 cm)
- `vh`: Relative to 1% of the height of the viewport
- `vmin`: Relative to 1% of the width of the narrow side of the viewport
- `vmax`: Relative to 1% of the width of the wide side of the viewport
- `%`: Percentage to the original value

The unit used in typography *em* defines the size of the letter "M" about as a measure of unity. This is not the case with CSS, the value is comparable with the browser default font Times New Roman. Which concrete pixel value the browser uses, however, is not clearly defined, there are not 12 pixels as is often claimed, and the more frequently encountered 16 pixels are not a guarantee.

Figure 1-5 shows the default font (here Firefox) actually used 16 pixels for the letter "M" and this is equivalent to specifying 16 px. The red line is 16 pixels wide. As font size was adjusted to 1 em or 16 px at this screenshot,the achieved result is the same.

Figure 1-5. *Measurement of the pixels of the em unit*

If you need a relative unit, you access *em* or *rem*. The unit *rem* has the advantage that it stays constant over the entire page, in contrast *em* applies to the most current font.

CHAPTER 2

Introduction to Bootstrap

Generally, Bootstrap 4 is distributed using the repositories Bower (via Github) and NPM (node package manager). Moreover, you also can create your own distribution and use to the source code that connects/links directly to the website.[1] Bootstrap also utilizes the raw files of the cascading stylesheets language SASS—this is a precompiler that translates into CSS (unlike its predecessor, Bootstrap 3, where the primary language was LESS).

What's New in Bootstrap 4?

This section gives an overview of the changes for those already familiar with Bootstrap 3.

Global Changes

The unit system was changed from pixels (px) in ***rem*** (CSS) or ***em*** (media queries). The global font, which serves as the starting point was increased from 14 pixels to 16 pixels. This is mainly a reference to the higher resolution screens of mobile devices nowadays.

Grid System

So far there have been four raster layers: ***xs***, ***sm***, ***md***, and ***lg***. In the future, there will be another level: ***xl***. This is designed to support extremely large displays (3,000 × 2,000 pixels and more).

Tables

Tables come with new options:

- ***.table-inverse*** inverts the table
- ***.thead-default*** and ***.thead-inverse*** format the header area
- ***.table-sm*** creates smaller tables

[1] http://getbootstrap.com

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_2

- All context classes now have the prefix ***.table-***
- ***.responsive-table*** can now be directly assigned to the table element, the container is no longer needed
- ***.table-reflow*** supports reflow tables

Forms

Instead of the special class ***.row***, ***.form-horizontal*** can now be used. All ***.control***-- and ***.input***--classes were unified to ***.form-control***-. This means that the special classes ***.has-feedback*** and ***.help-block*** are obsolete.

Buttons

The classes ***.btn-default*** and ***.btn-info*** **were** omitted in favor of ***.btn- secondary***. Another new feature is ***.btn-xx-outline*** to create a bezel button.

The class ***.btn-xs*** was removed.

Pulldown Menus

Instead of `<ul><li>`- structures menus can now be formatted as `<div><a>`- blocks directly. These are the classes ***.dropdown-item*** and ***.drop-down-menu*** available for the same. The class ***.dropdown-header*** can now be applied directly to `<h1>` and so forth. The intervals with `</div class="dropdown-divider"> are` created instead of the previous method with `<li>`-tags.

Panels

The display forms Wells, Panels, and Thumbnails were deleted and replaced by a new component with the name Card.

Others

A Progress bar can now be created with the HTML element `<progress>`. Quotations need the class ***.blockquote***. Due to lack of browser support this currently has some limitations.

Installation

You can load bootstrap from a CDN (content delivery network) or locally. The local version can be pulled from Bower, npm, Github, or the Bootstrap website respectively.

CDN

Bootstrap is available via MaxCDN. A CDN enables a website to frequently retrieve used public files from globally distributed servers. When a user from the United States invokes your host in Germany, then the CDN will ensure that the bootstrap files are retrieved from a server in the United States. This relieves your server, the lines of the provider, the Internet in general and the user experiences a faster download. Basically it is a win–win. If you program on the intranet, a CDN is in contrast not to your advantage is not to your advantage. If you expect only local users in Germany, there is no advantage to a CDN.

The bootstrap files are included when using the CDN as follows (the backslash at the end of line conotates the line break and is not part of the URL):

```
<!-- The latest compiled and minimized CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstr\
ap/4.0.0./css/bootstrap.min.css">

<!-- Optional Theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstr\
ap/4.0.0/css/bootstrap-theme.min.css">

<!-- The latest compiled and minimized JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/boot\
strap.min.js"></script>
```

Bootstrap 4 Alpha

This book was written when the alpha was the latest version. The version was ***4.0.0.-alpha.5***. If you use a newer version, you replace all the version numbers of the type ***4.0.0*** accordingly. If you are using a later version, Beta or Release Candidate, then adjust the version string completely. The electronic edition of this book will be continually updated.

Repository for Local Installation

To obtain a local copy, you can use either Bower[2] or npm[3]. Bower is for general front-end development, while NPM is the repository (node package manager) for ***Node.js***. If you develop in Ruby on Rails, you should look for Gem "Bootstrap for SASS" (bootstrap).[4]

[2] http://bower.io
[3] https://www.npmjs.com/
[4] https://github.com/twbs/bootstrap-rubygem.

Install with Bower

You can get Bootstrap's SASS, CSS, JavaScript, and other files using Bower to install and manage.

```
$bower install bootstrap#4.0
```

Bower retrieves data from a separate repository, usually directly from GitHub. Bower itself is a node package so it requires Node. If you have not worked with Bower, please observe the following conditions:

- First, install a Git client suitable for your operating system
- Install Node.js—which automatically will bring the Node Packet Manager **npm**
- Install Bower by NPM

  ```
  $npm install bower -g
  ```

Install using NPM

By using Bootstrap, **npm** is installed as follows:

```
$npm install bootstrap@next
```

If you are using **npm**, you will probably use ***Node.js***. Bind bootstrap as follows in your node application files:

```
require('bootstrap')
```

This command loads bootstraps jQuery plug-ins in the jQuery object. The module ***bootstrap*** itself has exported nothing. You can download the jQuery plug-ins individually by loading the ****/js/***.js*- files in the root directory of the package.

Bootstraps ***package.json*** contains some additional metadata under the following keys:

- sass: Path to Bootstrap's SASS source files
- style: Path to Bootstrap's nonminimized CSS that has been precompiled with the default settings (without adjustment option)

Bootstrap was developed in version 4 with SASS.[5] Find downloading on the current information, to make sure you are using the correct precompiler. SASS is available on all platforms. Bootstrap ***Autoprefixer***, used to work with the vendor prefixes in CSS. If you want to compile Bootstrap by using the SASS source and not use the supplied Gruntfile, you must ensure that the Autoprefixer is integrated into the build process. If precompiled boot-strap files or Gruntfile is used the ***Autoprefixer*** is already integrated. The procedure applies to Gulp or other TaskRunner equivalents if they are to be used.

---

[5] http://sass-lang.com/

Bootstrap can be downloaded in two types of packages—the compiled version and its additional minimized variants. Bootstrap requires jQuery as the basis of the components.

Structure of the CSS *Files*

The complete structure of an environment that uses Bootstrap is as follows:

```
bootstrap/
  ├──────── css/
  │    ├──────── bootstrap.css
  │    ├──────── bootstrap.css.map
  │    ├──────── bootstrap.min.css
  |    ├──────── bootstrap.min.css.map
  │    ├──────── bootstrap-flex.css
  │    ├──────── bootstrap-flex.css.map
  │    ├──────── bootstrap-flex.min.css
  │    ├──────── bootstrap-flex.min.css.map
  │    ├──────── bootstrap-grid.css
  │    ├──────── bootstrap-grid.css.map
  │    ├──────── bootstrap-grid.min.css
  │    ├──────── bootstrap-grid.min.css.map
  │    ├──────── bootstrap-reboot.css
  │    ├──────── bootstrap-reboot.css.map
  │    ├──────── bootstrap-rebooot.min.css
  │    └──────── bootstrap-rebooot.min.css.map
  ├──────── js/
  │    ├──────── bootstrap.js
  │    ├──────── bootstrap.min.js
Font-Awesome/
  ├──────── css/
  │    ├──────── fontawesome.css
  │    ├──────── fontawesome.css.map
  │    ├──────── fontawesome.min.css
  ├──────── fonts/
  │    ├──────── fontawesome-webfont.eot
  │    ├──────── fontawesome-webfont.svg
  │    ├──────── fontawesome-webfont.ttf
  │    ├──────── fontawesome-webfont.woff
  │    ├──────── fontawesome-webfont.woff2
Tether/
  ├──────── js/
  │    └──────── tether.js
  └──────── css/
  |    ├──────── tether.css
       └──────── tether.min.css
```

Precompiled files are the easiest methods to use Bootstrap. The npm versions are minimized (compressed). Tether is an add-on for popover and tooltips. Additionally one needs jQuery.

 If you have a minimizer (sometimes called minifier or uglifier) in your project, you pass the nonminimized files. Some minimizers destroy the code if it has already been compressed by another minimizer.

In addition to CSS and JavaScript, distribution also contains fonts that provide the icons. Font symbols are a particularly compact and simple method to incorporate monochrome icons in websites.

Page Building

Once everything is ready, you can create the first page. This page should provide the basic layout of the entire application. A first version might look like this (backslashes remark line breaks):

```
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="utf-8">
5       <meta name="viewport" content="width=device-width, initial-scale\
6   =1, shrink-to-fit=no">
7       <meta http-equiv="x-ua-compatible" content="ie=edge">
8
9       <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/boo\
10  tstrap/4.0.0/css/bootstrap.min.css" crossorigin="anonymous">
11    </head>
12    <body>
13      <h1>Hello Bootstrap 4</h1>
14
15      <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/\
16  jquery.min.js"></script>
17      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/\
18  bootstrap.min.js" crossorigin="anonymous"></script>
19    </body>
20  </html>
```

The first three metatags from line 4 must be at the beginning of the <head> block.

 If you load this page locally via ***file:///*** consider that some parts not work as expected. Always try to carry out all stages of development with a local web server (IIS Express with Visual Studio, IIS, Node.js, or a local Apache or nginx are perfect for this).

Note that bootstrap itself (line 17) must be loaded after jQuery (line 15).

 Download only from a CDN when the website will be public. For an intranet a local copy is more suitable.

Browser Support

Even with Bootstrap 4 the browser support is still an issue. It has taken a lot of effort to reach as many browsers and platforms here. Table 2-1 shows how it currently looks.

Table 2-1. *Browser Support*

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	OK	OK	N/A	No	N/A
iOS	OK	N/A	No	possibly Not	OK
MacOS X	OK	OK	possibly Not	possibly OK	OK
Windows	OK	OK	possibly OK	OK	Not possible

Chromium and Chrome for Linux, Firefox for Linux, and Internet Explorer 8 should work but are not officially supported. The broadest coverage with browsers you currently have are on MacOS and Windows.

ARIA

Support for barrier-free applications (aaccessible rich Internet applications suite = ARIA[6]) is actually a HTML topic. Some examples in the text are already geared and carry these attributes. Here is an overview of how this works. Bootstrap supports ARIA implicitly and comprehensively, so the theme fits in well here.

HTML5—*the* Role Attribute

The `role`-attribute is set in the relevant HTML tags. It improves semantic markup and thus helps screen readers and other users access to devices in the correct output:

- `banner`: The element is a banner.
- `complementary`: The element adds a section, usually instead of an `<aside>`.
- `content`: Regular content
- `info`: Additional information

[6] http://www.w3.org/TR/wai-aria/

- `form`: Form
- `main`: Main area
- `navigation`: Navigation area
- `search`: Search

The following `role` attribute values describe the structure of the page:

- `article`: Article (text)
- `columnheader`: Column header
- `definition`: Definition
- `directory`: Directory
- `document`: Document
- `group`: Group
- `heading`: Head area
- `img`: Images
- `list`: List
- `listitem`: List item
- `math`: Mathematical formula
- `note`: Note or supplement
- `presentation`: Presentation, display support
- `region`: Area
- `row`: Row
- `rowheader`: Head of a row (left turn)
- `separator`: Separation line in menus or lists
- `toolbar`: Toolbar

An example of an application looks like this:

```
<hr role="separator" />
```

In tags such as `<nav>` or `<aside>`, the `role` attribute is redundant. Therefore, the following would not be necessary:

```
1   <nav role="navigation">
2   <aside role="complementary">
```

ARIA Not an Issue?

In addition to the barrier-free access, ARIA facilitates semantic attributes that take care and help the maintenance of your pages source code. It is always easier to work with `role="banner"` compared to work interspersed with dozens of `<div>` elements.

Optimization

Correctly created bootstrap pages can be considerably larger than classic HTML pages. The stability of the sides has its price. You should therefore carefully consider how the elements are created. A typical example are long lists with many options. This is where the additions come in to the list of elements, the many buttons or menus you want to use. Especially with server side generated code this seems critical and causes little effort. Here is an example:

```
<ul class="list-group">
  <li class="list-group-item">First Element
  <div class="btn-group">
    <button type="button"
            class="btn btn-default dropdown-toggle"
            data-toggle="dropdown" aria-haspopup="true"
            aria-expanded="false">
      Aktion <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">Delete</a></li>
      <li><a href="#">Move</a></li>
      <li><a href="#">Rename</a></li>
      <li role="separator" class="divider"></li>
      <li><a href="#">Download</a></li>
    </ul>
  </div>
  </li>
  <li>... Other elements</li>
  <li>... Other elements</li>
</ul>
```

This list requires about 530 characters, and in UTF-16 this is more than 1 KB (in UTF-8 only the characters outside of ASCII are made of several characters). When you view 40 items on the page, which optically is not a problem because of the drop-downs, it is about 40 KB HTML and a payload of roughly 2 KB (40 times the text for each entry with 50 bytes). In this case it would be worthwhile to use JavaScript.

Listing 2-1 defines a template with the code of one element and the JavaScript then adds it to the running time of each list element. For control, HTML5 attributes are used.

Listing 2-1

```
<ul class="list-group" data-list-target>
  <li class="list-group-item">First Element</li>
  <li>...other elements</li>
  <li>...other elements</li>
</ul>
<div class="btn-group" data-list-template>
  <button type="button"
```

```
 8            class="btn btn-default dropdown-toggle"
 9            data-toggle="dropdown" aria-haspopup="true"
10            aria-expanded="false">
11       Action <span class="caret"></span>
12     </button>
13       <ul class="dropdown-menu">
14         <li><a href="#">Delete</a></li>
15         <li><a href="#">Move</a></li>
16         <li><a href="#">Rename</a></li>
17         <li role="separator" class="divider"></li>
18         <li><a href="#">Download</a></li>
19       </ul>
20   </div>
```

Now this is read by JavaScript using `data-list-template` attributes to address this portion in the code (line 6) and then cloned at the achievable place through `data-list-target` attributes. The code block uses jQuery here:

```
1   // Execution if the document was loaded..
2   $(function(){
3     // Load Template, clone and hide
4     var template = $('[data-list-template]').hide().clone();
5     // Search list items, Attach copied template, Show
6     $('[data-list-target] li').append($(template).show());
7   });
```

The script will only cost about 250 bytes (130 characters without the comments). Instead of a maximum of 40 KB this solution requires less than 2.5 KB—it is 6% of the original size or a decrease of 94%. In addition, the JavaScript code can be outsourced and cached in the browser.

Interactive Surfaces

The script can be further refined so that the items will only appear when the mouse pointer hovers over the entry. The `append` is only performed when the `mouseenter` emerges and `mouseleave` all buttons are removed. Ensure that the events are always in some `$(document).on('click')` pattern, so it works well with the dynamically appended elements. Tip: RTFM jQuery!

CHAPTER 3

Structure of the Page

Bootstrap constructs a horizontal grid on the page, then the elements are being placed within.

Introduction

The basis for the grid is a fixed allocation of the page. Second, to be certain attitudes of the viewing area, the so-called viewport is achieved.

The HTML5 Doctype

The first step is to adjust the website to HTML5. This is accomplished with the correct doctype:

```
1   <!DOCTYPE html>
2   <html lang="en">
3     ...
4   </html>
```

Settings of the Viewing Area

For mobile devices that were supported from the outset, the viewport is now set. The first element at the `<head>`-section is therefore the following metatag:

```
1   <meta name="viewport"
2         content="width=device-width, initial-scale=1" />
```

Definitely a requirement here is the zoom behavior and the scale used at the beginning. The zoom behavior can be turned off with no `user-scalable=no`. The application then feels a little like a native app on a mobile device.

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_3

Zoom Switch Off

It is risky to switch off the zoom function. Users with limited eyesight or users with especially small screens may be reliant on it. When a mobile website should behave like a native app, it must also be built from the ground as an app.

```
<meta name="viewport"
      content="width=device-width,
               initial-scale=1,
               maximum-scale=1,
               user-scalable=no">
```

The Grid System

Already in the early years of the web, there has been an attempt of a kind of virtual grid (grid) to give structure. At the beginning tables were used most often. However, tables are rigid in their horizontal extension—the content determines the width. A scaling to a smaller screen is not possible.

The idea of the table shows the procedure, rather than simply having the cells take -tags and then define the grid-like using CSS. Who has tried it once may have quickly realized that the approach is anything but trivial. CSS frameworks therefore provide a solid foundation. In fact, the screen is more of an array of strips, in which the screen is divided and the width can be adapted.

Container

Bootstrap utilizes a container element to initiate the page and find a basis for the grid. Containers can be used multiple times on the page, but should not be nested.

```
<div class="container">
  ...
</div>
```

Containers of this type provide a solid, centered, and responsive grid. That is, the width adapts jumps in the device's width and then remains stable within a range. The jump areas are referred to as "breakpoints." If you already know the concept of software development—the breakpoints in Bootstrap has nothing in common with this.

Alternatively, the class *.container-fluid* can be used, which always uses the full width of the device:

```
<div class="container-fluid">
  ...
</div>
```

The Grid in Detail

The grid is formed by twelve (12) equal width columns. Predefined classes can be used to place elements on a given column and span a number of columns at a time.

Inside the columns are rows that can be defined to allow an alternate use of the column widths. The procedure should be based on a few rules:

- Series (*.row*) must be in a container (*.container* (fixed width) or *.container-fluid* (full width))
- Series may be used horizontally to place multiple elements side-by-side.
- Columns (*.col-xx-n*) may be placed in rows. Only column elements are immediate child elements of rows.
- Column spacing (gap between columns) is defined using *padding*-rules. These distances are to the left of the first column and the right of the last column with negative intervals (*margin*) balanced. This content is aligned outside of the grid's left-aligned equal.
- If the elements need columns that can be placed within a row and exceed the elements limit of twelve columns, then the whole group will be wrapped.

The structure of the column classes is simple:

- The introduction begins with *col-*
- The middle part determines the jurisdiction for screen widths (*xs*, *sm*, *md*, *lg*)—the breakpoints.
- The number at the end determines the number of columns (1 to 12).

If you use the class *.col-xs-4*, you can have 3 juxtaposing decorated elements (3 × 4 = 12). The higher (larger device) definition applies until it is overwritten. You can use *.col-md-2*, *.col-lg-2,* than "MD" is not valid for very large devices

Device Specific Definitions

Small units with less than 768 px are not defined separately, because this class of device is already the default. The information contained in the variables are defined as follows:

- "xs" <= 544 px
- "sm" <= 768 px
- "md" <= 992 px
- "lg" <= 1200 px
- "xl" > 1200 px

```
@media (min-width: @screen-sm-min) { ... }

@media (min-width: @screen-md-min) { ... }

@media (min-width: @screen-lg-min) { ... }
```

Table 3-1 shows the widths of the dimensions of the columns.

Table 3-1. *Position of Breakpoints*

Symbol	Device Width	Container Width	Column Width
xs	< 544 px	Automatic	Automatic
sm	≥ 544 px	750 px	62 px
md	≥ 768 px	970 px	81 px
lg	≥ 992 px	1,170 px	97 px
xl	≥ 1,200px	1,570 px	112 px

Changes to Bootstrap 3

Note that the breakpoints have shifted from Bootstrap 3. The highest value (1,200 px) is still the highest in the list, but now bears the name *xl*. Another new feature is the smallest value 544 pixels to date already used name *xs*.

The width of the spacing between columns is 30 pixels (15 on each side).

In Listing 3-1 only "md" classes are used. These are only valid with a width of 920 px and more. The DIV elements are therefore arranged among each other on small devices—only on desktops with sufficient width will the items appear next to each other.

Listing 3-1. Pattern grid (Grid.html)

```
<div class="container">
  <div class="row">
    <div class="col-md-1">1</div>
    <div class="col-md-1">2</div>
```

```
      <div class="col-md-1">3</div>
      <div class="col-md-1">4</div>
      <div class="col-md-1">5</div>
      <div class="col-md-1">6</div>
      <div class="col-md-1">7</div>
      <div class="col-md-1">8</div>
      <div class="col-md-1">9</div>
      <div class="col-md-1">10</div>
      <div class="col-md-1">11</div>
      <div class="col-md-1">12</div>
    </div>
    <div class="row">
      <div class="col-md-8">1-8</div>
      <div class="col-md-4">9-12</div>
    </div>
      <div class="row">
      <div class="col-md-4">1-4</div>
      <div class="col-md-4">5-8</div>
      <div class="col-md-4">9-12</div>
    </div>
      <div class="row">
      <div class="col-md-6">1-6</div>
      <div class="col-md-6">7-12</div>
    </div>
  </div>
```

With a few styles to illustrate the areas it looks like Figure 3-1.

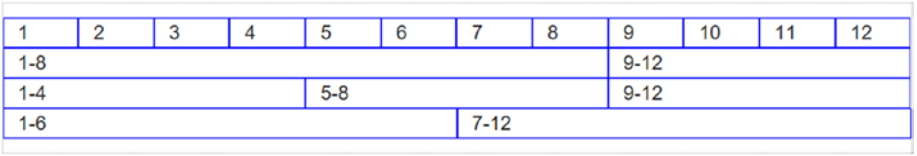


Figure 3-1. *The grid system (desktop)*

On a mobile device the same code produces the screen in Figure 3-2.

Figure 3-2. *The grid system (mobile)*

It may be that mobile devices provide adequate space, at least represented by some columns side-by-side. These are some further definitions of *.col-xs-* and *.col-md-* classes required.

In Listing 3-2, the first row shows that on small devices every 12 or the first 6 columns are used. If the screen is larger, only 8 or 4 columns are used (which are then wider). The second set uses 50% of the screen on small devices and 33% too large (see Figure 3-3). The third row always uses 50% width, no matter what device ("xs" scales high when no further definition follows).

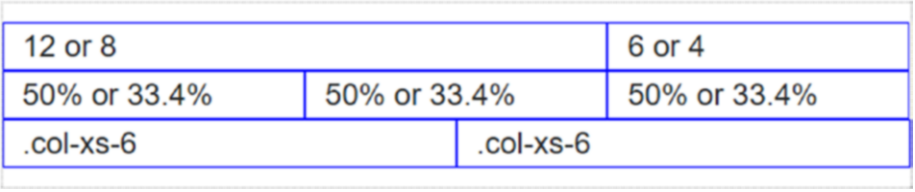


Figure 3-3. *Variable grid system (Desktop)*

Listing 3-2. Pattern grid (Grid_xs.html)

```
<div class="row">
  <div class="col-xs-12 col-md-8">12 or 8</div>
  <div class="col-xs-6 col-md-4">6 or 4</div>
</div>

<div class="row">
  <div class="col-xs-6 col-md-4">50% or 33.4%</div>
  <div class="col-xs-6 col-md-4">50% or 33.4%</div>
  <div class="col-xs-6 col-md-4">50% or 33.4%</div>
</div>

<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

On a mobile device the same code produces the screen as shown in Figure 3-4.

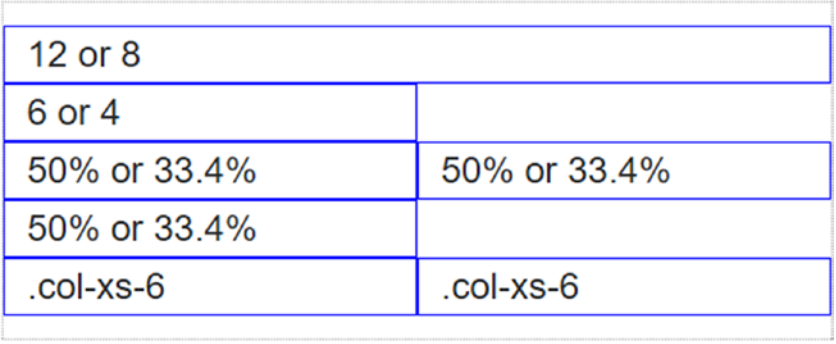


Figure 3-4. *Variable grid system (mobile)*

If more classes are used than other gradations are possible, for example, some tablets with the variant “sm” (see Figure 3-5).

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">
    Small: 12 Tablet: 6 Large: 8
  </div>
  <div class="col-xs-6 col-md-4">
    Small: 6 Large: 4
  </div>
</div>
<div class="row">
  <div class="col-xs-6 col-sm-4">A Small: 6 Tablet: 4</div>
```

```
11      <div class="col-xs-6 col-sm-4">B Small: 6 Tablet: 4</div>
12    <!-- Optional: Wrap at different height -->
13      <div class="clearfix visible-xs-block"></div>
14      <div class="col-xs-6 col-sm-4">C Small: 6 Tablet: 4</div>
15    </div>
```

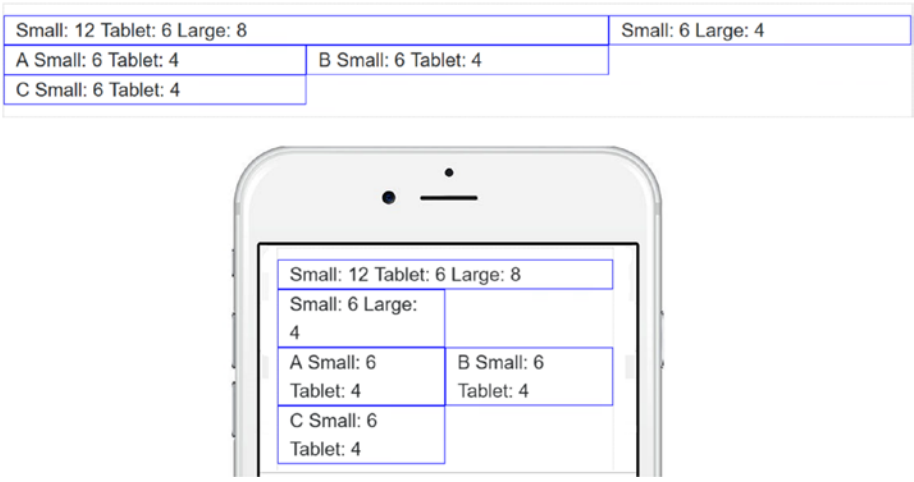


Figure 3-5. *Tablet (top) versus phone*

The limit of 12 columns is not absolute and contents are never clipped. What does not fit into the grid is simply moved to the next virtual line (as shown in Listing 3-3). Items that are in a column definition will be moved as a whole.

Listing 3-3. Pattern grid (Grid_Break.html)

```
1   <div class="row">
2     <div class="col-xs-9">9</div>
3     <div class="col-xs-4">4<br>And more content...</div>
4     <div class="col-xs-6">6<br>More columns hereafter.</div>
5   </div>
```

In this series, 9 and 4 columns are defined. This is 13 and does not fit in into the 12 width. Therefore, the second element (line 3) that the 4 columns spanned, moved to the next line. The other columns simply rearranged themselves as shown in Figure 3-6.

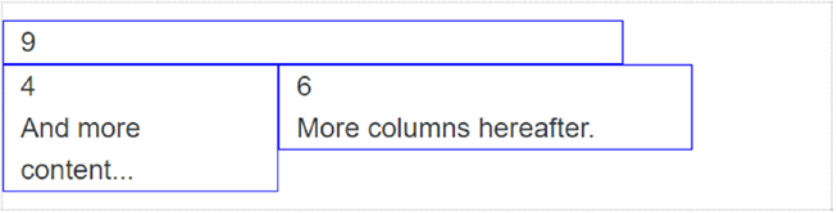


Figure 3-6. *Break off the grid*

When wrapping the elements, it sometimes happens that the content leads to different heights. The DIV elements that usually come to use, extend far into the height of until everything fits. Although you can theoretically use a clipping—cutting the contents—this is probably not required in most cases. A combination of *.clearfix* and other auxiliary classes is more likely to succeed.

```
 1   <div class="row">
 2     <div class="col-xs-6 col-sm-3">Small: 6 Tablet: 3<br />
 3       More<br />
 4       Even more
 5     </div>
 6     <div class="col-xs-6 col-sm-3">Small: 6 Tablet: 3</div>
 7
 8     <div class="clearfix visible-xs-block"></div>
 9
10     <div class="col-xs-6 col-sm-3">Small: 6 Tablet: 3</div>
11     <div class="col-xs-6 col-sm-3">Small: 6 Tablet: 3</div>
12   </div>
```

Figure 3-7 shows the effect: the top half with *.clearfix* (line 8), the bottom without.

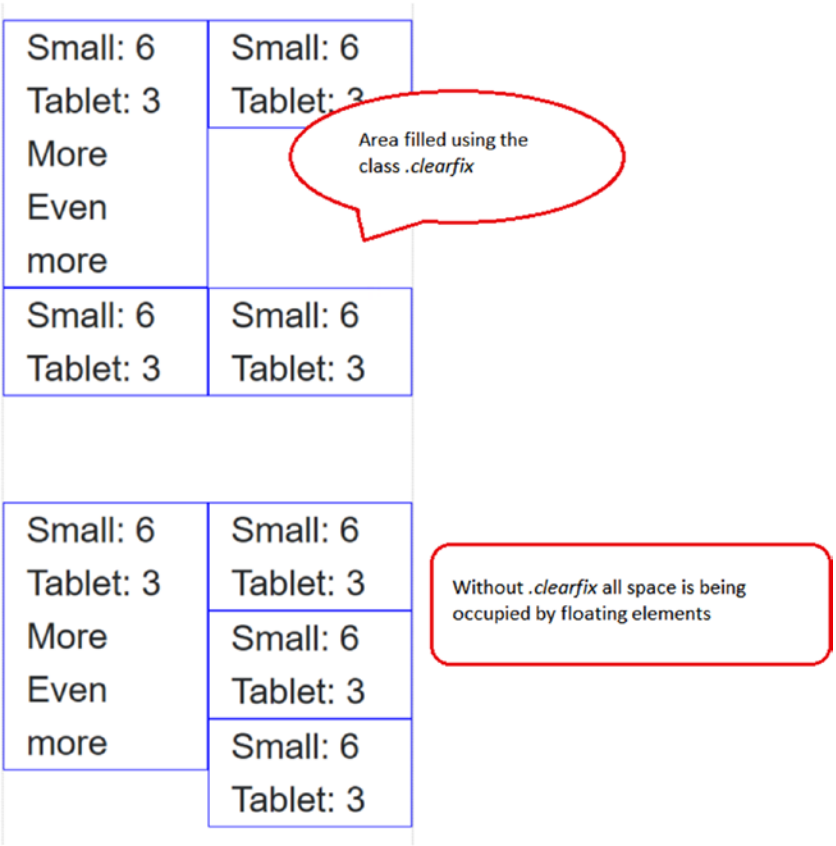


Figure 3-7. *Consideration of different heights*

Utilizing the Width

To take advantage of the full width, *.container-fluid* is used:

```
1   <div class="container-fluid">
2     <div class="row">
3       ...
4     </div>
5   </div>
```

The left and right edges of each 15 px is placed so that the item in the container uses the space to the best advantage as seen in Figure 3-8.

Figure 3-8. *Normal container (above) and with -fluid (below)*

In the following examples, I have dispensed with the container element to keep the code compact. But you must use a container element in practice.

Offsets—Move the Start Column

Offsets move the starting point of a column to the right (see Listing 3-4 for offset use). This is based on the valid point. Already unoccupied columns move the starting point already so the offset just adds on to this (see Figure 3-9). Used classes *.col-xx- offset-*. so hot ("xx" stands again for the base widths of the media queries).

Listing 3-4. Offset use (Grid_Offset.html)

```
<div class="row">
  <div class="col-md-4">4</div>
  <div class="col-md-4 offset-md-4">4 + 4</div>
</div>
<div class="row">
  <div class="col-md-3 offset-md-3">3 + 3</div>
  <div class="col-md-3 offset-md-3">3 + 3</div>
</div>
<div class="row">
  <div class="col-md-6 offset-md-3">6 + 3</div>
</div>
```

Figure 3-9. *Elements in moving columns*

Interlace Columns

Containers must not be nested (see Listing 3-5 and Figure 3-10). However, this is allowed in rows. Therefore, subordinate areas can be created. It is not mandatory that such areas fill out their parent's area and it must not extend until all 12 columns are used.

Listing 3-5. Offset use (Grid_Nested.html)

```
<div class="row">
  <div class="col-sm-9">
```

```
3       Parents Area: 9 Column
4       <div class="row">
5         <div class="col-xs-8 col-sm-6">
6           Child 1: 8 or 6
7         </div>
8         <div class="col-xs-4 col-sm-6">
9           Child 2: 4 or 6
10        </div>
11      </div>
12    </div>
13  </div>
```

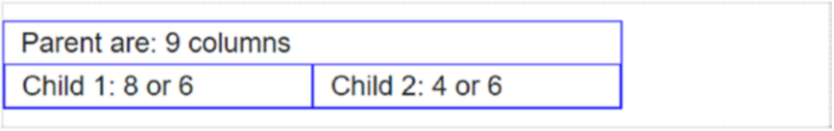


Figure 3-10. *Behavior of nested columns*

Order Column

If individual columns have class styles such as *.push-md-* or *pull-md-*, they can be used, even if different from their natural order. "Push" means pushing columns to the right, while they are pulled to the left with "pull". Figures 3-11 and 3-12 show this for both desktop and mobile, respectively.

```
1   <div class="row">
2     <div class="col-md-9 push-md-3">9 + 3 right</div>
3     <div class="col-md-3 pull-md-9">3 + 9 left</div>
4   </div>
```

Figure 3-11. *Order on a desktop*

Because the sort was defined only for the size "md", the mobile device falls back to the arrangement that results from the pure HTML.

Figure 3-12. *Sorting on a mobile device*

CHAPTER 4

Typography

Bootstrap defines styles for some elements directly in HTML, the basic type of ad, typography, and color. In contrast to the classic concept of typography, font, writing, design, and printing material included, Bootstrap is quite modest in its ease of use. The following are specifically regulated:

- The background color of the page is white, `background-color: #fff`
- In the SASS files the following variables are responsible for the basic design:
  - `$font-family-base`
  - `$font-size-base`
  - `$line-height-base`
- Hyperlinks as outstanding elements of web pages are determined by the following variables:
  - `$link-color`
  - Underline links should only be used with floating cursor (`:hover`)

The variables are in the file *_scaffolding.sass* and are easy to find.

Headings

All headings from `<h1>` to `<h6>` are directly supported. You must not use any other classes. But if you wish to use the same font size as a heading, the text should using the classes *.h1* to *.h6*. See Figure 4-1 for an example of headings.

- h1: Bold 2.5 rem / 40 px (Basis × 2.6)
- h2: Bold 2 rem / 32 px (Basis × 2.15)
- h3: Bold 1.75 rem / 28 px (Basis × 1.7)
- h4: Bold 1.5 rem / 24 px (Basis × 1.25)

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_4

- h5: Bold 1.25 rem / 20 px (standard size, this is the basis for calculation)
- h6: Bold 1 rem / 16 px (Basis × 0.85)

```
<h1>h1. Bootstrap Heading</h1>
<h2>h2. Bootstrap Heading</h2>
<h3>h3. Bootstrap Heading</h3>
<h4>h4. Bootstrap Heading</h4>
<h5>h5. Bootstrap Heading</h5>
<h6>h6. Bootstrap Heading</h6>
```

To obtain a lighter (smaller, lighter) appearance, `<small>` oder *.small* benutzt.

```
<h1>h1. Bootstrap Heading <small>Additional text</small></h1>
<h2>h2. Bootstrap Heading <small>Additional text</small></h2>
<h3>h3. Bootstrap Heading <small>Additional text</small></h3>
<h4>h4. Bootstrap Heading <small>Additional text</small></h4>
<h5>h5. Bootstrap Heading <small>Additional text</small></h5>
<h6>h6. Bootstrap Heading <small>Additional text</small></h6>
```

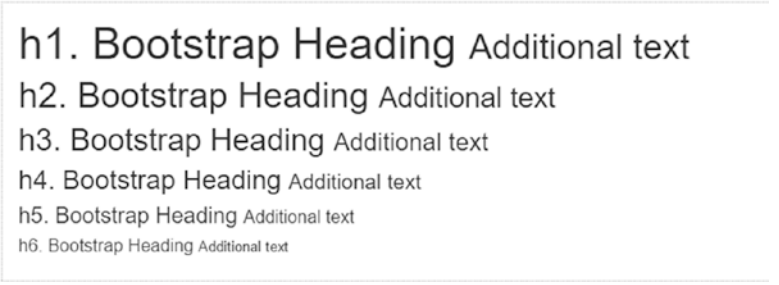


Figure 4-1. *Standard view of headings*

Text and Text Elements

For text and text elements, HTML has many references and has class supplements for special effects.

Basic Font

The default font for the text is 16 px. The line height used is 1.15. For sections created with `<p>` they will receive half the row height, about 10 px.

To single out one section, the class *.lead* is used as shown in Figure 4-2.

```
<p class="lead">Important paragraph...</p>
<p>Standard paragraph </p>
```

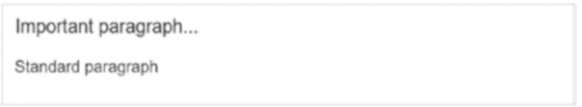


Figure 4-2. *Paragraph style*

Text Elements

Text elements are used for parts of the body text. This is done mainly through HTML, no more classes. The elements also serve to strengthen the semantic content. Figure 4-3 shows various semantic elements in the text. The following text elements are explicitly supported.

`<mark></mark>`

This text is highlighted (as with a highlighter) emphasizing relevance.

`<del></del>`

This indicates that a part of the text has been deleted, which is indicated by a strikethrough.

`<s></s>`

This indicates that a part of the text is no longer relevant.

`<ins></ins>`

This tag indicates that the body part has been added allowing a user to signal a last text change.

`<u></u>`

This tag indicates by underlining that the text provides additional information.

`<em></em>`

Highlights part of the text with emphasis thereby showing its importance.

Another tag fits in this category, `<small></small>`, but it has a less important semantic reference. To achieve a "lighter" display you can reduce the font size to 85%. To do the opposite of this, you should use `<strong></strong>`.

Bootstrap supports the continued nonsemantic elements `<b></b>` and `<i></i>`. They react in the text similar to `<strong></strong>` and `<em></em>`. However `<i></i>` sometimes is used for icons (icon), which do not contain text and therefore ignore the behavior displayed by italic font and enter an element with a certain semantic meaning. However, the behavior and the interpretation is not standard compliant, but more of a stylistic measure.

Some elements are marked more semantically.

```
<abbr title="Abbreviation">Abbr.</abbr>
```

This element identifies abbreviations, but an attribute title is required. The element is marked with a dotted underline and displays the title when the mouse pointer hovers over it. In addition, the font can be reduced with the class *.initialism* to have a less disruptive effect in the text.

```
<abbr title="HyperText Markup Language"
      class="initialism">HTML</abbr>
```

<kbd></kbd>

This element is used to refer to keys of a computer keyboard (<kbd>F12</kbd>).

This is the text highlighted (such as with a highlighter) to emphasize the relevance.

This indicates that a part of the text ~~has been deleted~~. This is indicated by a strikethrough.

This indicates that a part of the text is ~~no longer relevant~~.

This tag indicates that the body part has been added. This allows a user to the last change to be signaled on a text.

This tag indicates that the text part provides additional information to the text. This is indicated by underlining.

Emphasizes text part to highlight its importance.

Figure 4-3. *Semantic elements in the text*

Elements for Text Blocks

The element <address></address> can make it easy to detect addresses. This element is not formatted. If formatting is to be retained additional line breaks with
 elements are required.

```
<address>
  <strong>Twitter, Inc.</strong><br>
  795 Folsom Ave, Suite 600<br>
  San Francisco, CA 94107<br>
  <abbr title="Phone">P:</abbr> (123) 456-7890
</address>
```

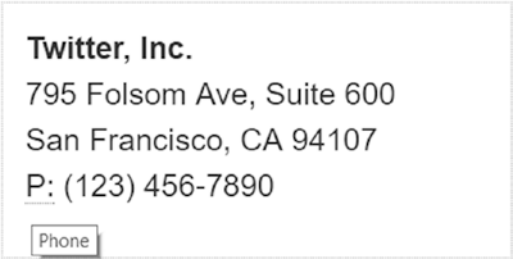


Figure 4-4. *Properly formatted address*

```
<address>
  <strong>Full Name</strong><br>
  <a href="mailto:#">first.last@example.com</a>
</address>
```

Quotes (`<blockquote>`) are also semantic and highlight text that refers to something else, such as a quote.

```
<blockquote class="blockquote">
  <p>Node.js is a modern programming environment.</p>
</blockquote>
```

For the referenced text add this element `<cite></cite>`, the text is then highlighted. The element `<footer></footer>` indicates the source.

```
<blockquote class="blockquote">
  <p>Node.js is a modern programming environment.</p>
  <footer class="blockquote-footer">Found in <cite title="Introduction
to Node.js">Node.js</ci\
te></footer>
</blockquote>
```

On some screens quotes are not immediately recognizable. They are aligned to the right, so structuring them to the side may help (see Figure 4-5 for a right-aligned quote).

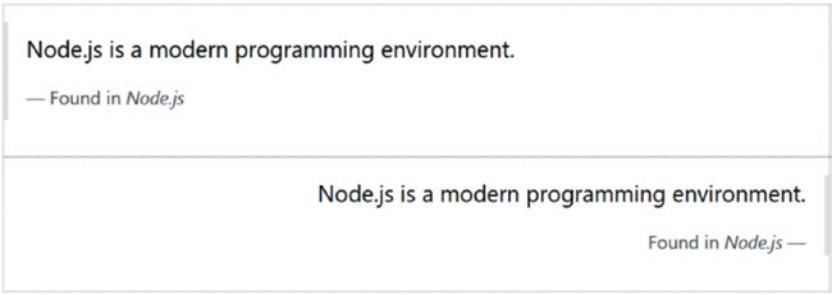


Figure 4-5. *Left and right aligned quote*

This is the purpose of the class *.blockquote-reverse*.

```
1   <blockquote class="blockquote-reverse">
2   ...
3   </blockquote>
```

Often code is explained on web pages. This will fit best in `<code></code>` and when it is in a row and in `<pre></pre>`, the formatting is to be maintained over several lines. If you need to refer to variables in the text use the element `<var></var>`. Use this element in mathematical formulas as well as in the following example and in Figure 4-6.

```
1   <var>f(x)</var> = <var>m</var><var>x</var> + <var>b</var>
```

$$f(x) = mx + b$$

Figure 4-6. *Display of variables or formulas*

To explain expenditures you should use the element `<samp></samp>`.

Orientation

The alignment of text is up to your creative nature.

- *text-left*: Left aligned
- *text-center*: Text is centered
- *text-right*: Right aligned
- *text-justify*: Justified
- *text-nowrap*: Obstruction of upheaval

In addition, there are alignment variants that are only valid for certain viewports.

- *text-xs-left*: Left aligned when the viewport "xs" is used.
- *text-xs-center*: Text is centered when the viewport "xs" is used.
- *text-xs-right*: Right aligned when the viewport "xs" is used.

This also applies for "sm," "md," "lg," and "xl." Listing 4-1 and Figure 4-7 demonstrate some text alignments.

Listing 4-1. Alignment of text (Typo_TextAlign.html)

```
1   <p class="text-left">Left aligned.</p>
2   <p class="text-center">Text is centered. </p>
3   <p class="text-right">Right aligned.</p>
4   <p class="text-justify">Justified ...</p>
5   <p class="text-nowrap">Obstruction of Upheaval, ...</p>
```

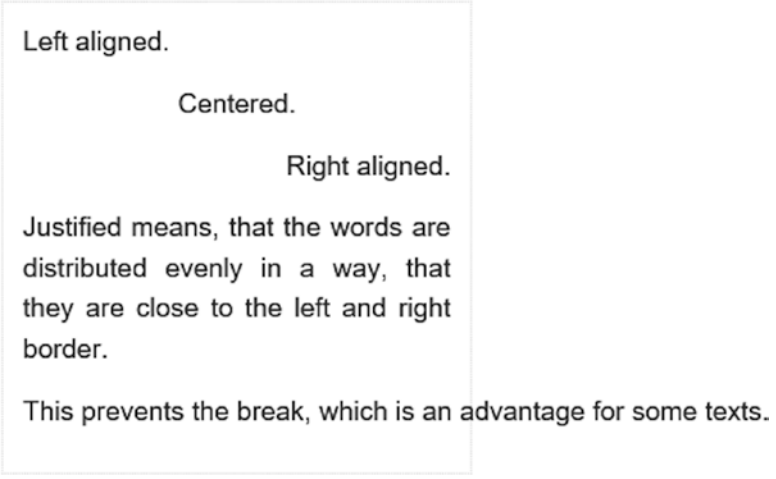


Figure 4-7. *Aligning text*

Justified

Justified (justify) text on screens is generally not a good idea. If you feel the need for justification, remember the width of the column can hold at least 10 words, otherwise the spaces between the words become too large. Although justification does not work well the can remain manageable by using artificial separations with the entity `&shy` (soft hyphen). The delimiters appear only if a break must be made.

Transformations

Transformations convert text from lowercase to uppercase and vice versa. Listing 4-2 and Figure 4-8 provide information on transformations.

- *text-lowercase*: Converts to lowercase
- *text-uppercase*: Converts to uppercase
- *text-capitalize*: First letter of each word is capitalized.
- *text-weight-bold*: Bold without an explicit tag.
- *text-weight-normal*: Normal weight without an explicit tag
- *text-italics*: Italics without an explicit tag

Listing 4-2. Transformations (Typo_TextTransform.html)

```
1   <p class="text-lowercase">lowercase</p>
2   <p class="text-uppercase">uppercase</p>
3   <p class="text-capitalize">title case</p>
```

Figure 4-8. *Text transformation*

Lists

There are two types of lists: with order and in no particular order. Without order they are represented in HTML by bullet points. In Bootstrap you also can waive these symbols and create a simple "element stack."

```
1   <ul>
2     <li>...</li>
3   </ul>
```

Lists with order decorate the elements with consecutive numbers, letters, or roman numerals.

```
<ol>
  <li>...</li>
</ol>
```

If the default style used and there are icons not needed, they can be removed by the class *.list-unstyled*. This will affect only the immediate members not the deeper nesting levels. Note here that the removal of the symbols means that the items are no longer engaged. This must be offset by a distance where appropriate.

```
<ul class="list-unstyled">
  <li>...</li>
</ul>
```

These are often defined as lists, and then lose that nature by CSS. The advantage over tables or definition lists is their particularly compact notation—usually two letters per element. The class *.list-inline* is then necessary to place the elements in horizontal direction (see Figure 4-9).

```
<ul class="list-inline">
  <li>...</li>
</ul>
```

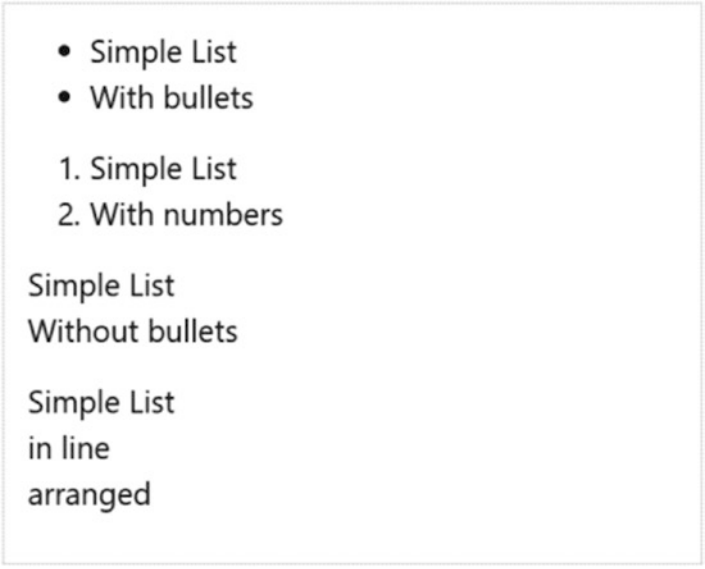


Figure 4-9. *Some styled lists*

If lists are complex a suitable replacement are so called definitions. Here two blocks per entry are actually used—the term (*definition term*, `<dt></dt>`) and the description (*definition description*, `<dd></dd>`). Listing 4-3 shows a list definition.

Listing 4-3. List definition (Typo_Definition.html)

```
<dl>
  <dt>...</dt>
  <dd>...</dd>
</dl>
```

As with simple lists, elements also can be arranged horizontally, in each case the `<dt>`-Element is aligned to the right and the right `<dd>`-element then left-justified. To align definitions just put the definition in a *.row* class and define the size with regular column classes.

```
<dl class="row">
  <dt class="col-md-3">Name</dt>
  <dd class="col-md-4">Anton Muller</dd>
  <dt class="col-md-3">Address</dt>
  <dd class="col-md-4">NYC</dd>
</dl>
```

The `<dl>`-sections are always set among themselves as shown in Figure 4-10. Terms that do not fit in the display area are truncated.

Figure 4-10. *Assembly with two entries (vertically above, horizontally below)*

Tables

It has been noted that tables are not suited to build responsive sites. Nevertheless, tables are suited for sorting and listing data. The width of the table must comply with the criteria of the grid. A placement outside of the grid is possible. In this case Bootstrap offers several styles to improve the presentation. For responsive environments tables are offered with horizontal scrolling. Of course this is only a temporary solution—really responsive tables should not need this.

Styles for Tables

Tables themselves have a native element in HTML. Nevertheless bootstrap relies on the class *.table*. This is because tables are used by many other libraries and plug-ins, such as Calendar. To avoid changes to these tables accidentially by Bootstrap this additional class is required.

```
1   <table class="table">
2     ...
3   </table>
```

Tables consist of three parts: header (`<thead></thead>`), content (`<tbody></tbody>`), and footer (`<tfoot></tfoot>`). The content can occasionally be very long. The readability can be significantly increased by an alternating background color—called the crosswalk effect. You can reach it with the class *.table-striped*. Figure 4-11 gives an example of such a table.

```
1   <table class="table table-striped">
2     ...
3   </table>
```

Another style for the table is *.table-bordered*, a frame is applied to all the cells.

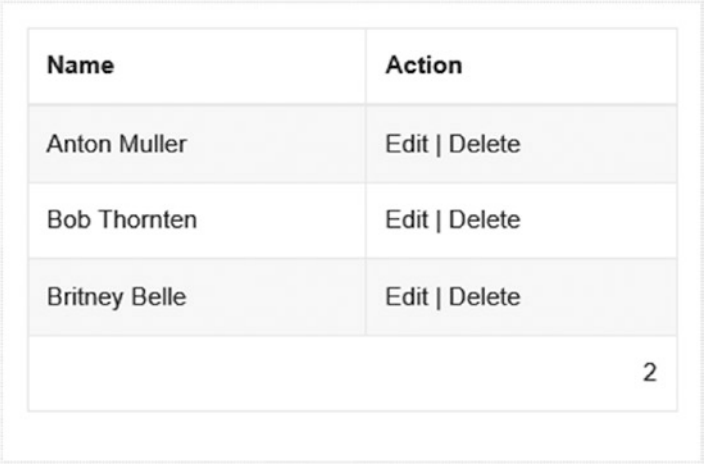
```
1   <table class="table table-bordered">
2     ...
3   </table>
```

More interactive is *.table-hover*, bringing a mouse effect when hovering over rows of a table.

```
1   <table class="table table-hover">
2     ...
3   </table>
```

If space is in short supply, the distances used by the default can be reduced with *.table-condensed*. The table can be reduced by 50%.

```
<table class="table table-condensed">
  ...
</table>
```



Name	Action
Anton Muller	Edit | Delete
Bob Thornten	Edit | Delete
Britney Belle	Edit | Delete
	2

Figure 4-11. *A table that is partly condensed with header and footer*

Columns

Tables also can have column definitions with `<colgroup><col><col></colgroup>`-elements. However, those are not explicitly supported by Bootstrap.

To achieve similar effects, the following CSS code could serve (the colors are examples).

```
colgroup col.success {
  background-color: #dff0d8;
}
colgroup col.error {
  background-color: #f2dede;
}
colgroup col.warning {
  background-color: #fcf8e3;
}
colgroup col.info {
  background-color: #d9edf7;
}
```

The example referres to the context classes that act as follows.

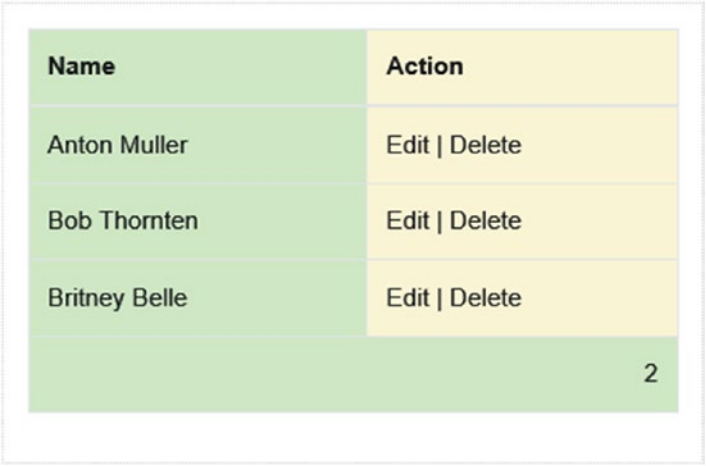
- *.active*: Mouse effect for rows or cells
- *.success*: Positive or success (green)
- *.info*: Information or action (blue)
- *.warning*: Warning or caution signal (orange)
- *.danger*: Negative or danger (red)

```
<!-- Rows -->
<tr class="active">...</tr>
<tr class="success">...</tr>
<tr class="warning">...</tr>
<tr class="danger">...</tr>
<tr class="info">...</tr>

<!-- Cells (td or th) -->
<tr>
  <td class="active">...</td>
  <td class="success">...</td>
  <td class="warning">...</td>
  <td class="danger">...</td>
  <td class="info">...</td>
</tr>
```

The headings of the table are then referenced on the column elements. Figure 4-12 shows a table with column definitions.

```
<table class="table table-bordered table-condensed">
  <colgroup>
    <col class="success" />
    <col class="warning" />
  </colgroup>
  ...
```



Name	Action
Anton Muller	Edit | Delete
Bob Thornten	Edit | Delete
Britney Belle	Edit | Delete
	2

Figure 4-12. *Table with column definitions*

Note that even the column styles collide with *.table-striped*. Avoid *.table-striped,* if your columns make color variations.

Legibility for Screen Readers

If a page is built accessible the context classes have no meaning. The potentially important information that arises from the choice of color is lost. Therefore, when you add additional items, you can mark these with *.sr-only* to make them. Screen readers will still read these items.

New in Bootstrap 4

There are new classes in Bootstrap 4: *.table-inverse* or *.thead-inverse* to invert the font color. Figure 4-13 shows an inverted table.

```
<table class="table table-inverse" >
        <thead>
          <tr>
            <th>Code</th>
            <th>Company</th>
            <th>Price</th>
            <th>Change</th>
            <th>In %</th>
            <th>Opening</th>
            <th>Highest</th>
```

```
            <th>Lowest</th>
            <th>Volume</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>MSC</td>
            <td>Microsoft</td>
        ...
```

Code	Company	Price	Change	In %	Opening	Max	Min	Volume
MSC	Microsoft	$1.38	-0.01	-0.36%	$1.39	$1.39	$1.38	9,395
APP	Apple	$1.99	-0.03	-0.45%	$1.88	$2.15	$1.38	7,741

Figure 4-13. *Inverted table*

Particularly interesting is *.table-reflow*, using this the block of headlines on the first column is transformed as a pivot table.

```
<table class="table table-reflow" >
        <thead>
          <tr>
            <th>Code</th>
            <th>Company</th>
            <th>Price</th>
            <th>Change</th>
            <th>In %</th>
            <th>Opening</th>
            <th>Highest</th>
            <th>Lowest</th>
            <th>Volume</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>MSC</td>
            <td>Microsoft</td>
        ...
```

For this effect, no modification of the table structure is required (see this effect in Figure 4-14).

Code	MSC	APP
Company	Microsoft	Apple
Price	$1.38	$1.99
Change	-0.01	-0.03
In %	-0.36%	-0.45%
Opening	$1.39	$1.88
Max	$1.39	$2.15
Min	$1.38	$1.38
Volume	9,395	7,741

Figure 4-14. *Turned table*

Responsive Tables

In the strictest sense, rresponsive tables do not exist. If *.table-responsive* used, especially in widths below 786 px, a horizontal scrollbar under the table is created. Figure 4-15 shows a table with a horizontal scrollbar. This will prevent the need that the entire page has to scroll horizontally. In addition, *overflow-y: hidden* suppresses all elements that will protrude upward or downward from the table. **Caution**: This can break or shorten menus or lists and disturb the grid.

```
1   <table class="table table-responsive">
2     ...
3   </table>
```

Figure 4-15. *Table with scrollbar*

Note that you no longer need to be apply *.table-responsive* as you did with Bootstrap 3 on a surrounding `<div>`- element.

True Responsive Tables

Bootstrap's responsive answer to tables with .table-reflow is not always satisfactory. In particular, the behavior is not customizable to fit your need. Ultimately, the HTML element `<table>` just is not always what you need. Complex tables with many columns, as those in Figure 4-16, cannot condense as necessary.

Code	Company	Price	Change	In %	Opening	Max	Min	Volume
MSC	Microsoft	$1.38	-0.01	-0.36%	$1.39	$1.39	$1.38	9,395
APP	Apple	$1.99	-0.03	-0.45%	$1.88	$2.15	$1.38	7,741

Figure 4-16. *Full-width table*

But the same table could be changed into a list that could be fully responsive as shown in Figure 4-17.

Code	MSC
Company	Microsoft
Price	$1.38
Change	-0.01
In %	-0.36%
Opening	$1.39
Max	$1.39
Min	$1.38
Volume	9,395
Code	APP
Company	Apple
Price	$1.99
Change	-0.03
In %	-0.45%
Opening	$1.88
Max	$2.15
Min	$1.38
Volume	7,741

Figure 4-17. *Table at narrow width*

By now many commercial and free grid components produce their presentation with tables. It is often not possible and involves considerable effort to adapt Bootstrap to these constructs. Fortunately, with CSS3 the tables can be adapted. When that does not work, the next step is HTML 5. JavaScript has addition tweaks for even smoother tables. Listing 4-4 takes the code of the table from Figures 4-14 and 4-15.

Listing 4-4. Very reponsive table with CSS3 (html)

```
<table class="table" id="notable">
  <thead>
    <tr>
      <th>Code</th>
      <th>Company</th>
      <th>Price</th>
      <th>Change</th>
      <th>In %</th>
      <th>Opening</th>
      <th>Highest</th>
      <th>Lowest</th>
      <th>Volume</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td data-title="Code">MSF</td>
      <td data-title="Company">Microsoft</td>
      <td data-title="Price">$51.38</td>
      <td data-title="Change">-0.01</td>
      <td data-title="In %">-0.36%</td>
      <td data-title="Opening">$51.39</td>
      <td data-title="Highest">$51.39</td>
      <td data-title="Lowest">$51.38</td>
      <td data-title="Volumen">9,395</td>
    </tr>
    <tr>
      <td data-title="Code">APC</td>
      <td data-title="Company">Apple</td>
      <td data-title="Price">$95.46</td>
      <td data-title="Change">-0.03</td>
      <td data-title="In %">-0.45%</td>
      <td data-title="Opening">$91.88</td>
      <td data-title="Highest">$99.15</td>
      <td data-title="Lowest">$91.38</td>
      <td data-title="Volume">7,741</td>
    </tr>
  </tbody>
</table>
```

The matching CSS looks like Listing 4-5. It is effective only through the `@media`-element, when the screen width falls below 768 px.

Listing 4-5. Very reponsive table with CSS3 (css)

```
@media only screen and (max-width: 768px) {

  /* Prevents the default behavior of a table */
  #notable table,
  #notable thead,
  #notable tbody,
  #notable th,
  #notable td,
  #notable tr {
    display: block;
  }

    /* Hide header */
    #notable thead tr {
      position: absolute;
      top: -9999px;
      left: -9999px;
    }

    #notable tr {
      border: 1px solid #ccc;
    }

    #notable td {
      /* Behavior of a number e */
      border: none;
      border-bottom: 1px solid #eee;
      position: relative;
      padding-left: 50%;
      white-space: normal;
      text-align: left;
    }

    #notable td:before {
      /* New header */
      position: absolute;
      /* Simulation of the distances */
      top: 6px;
      left: 6px;
      width: 145%;
      padding-right: 10px;
      white-space: nowrap;
      text-align: left;
```

```
44          font-weight: bold;
45          /* Get headlines from data-title="" */
46          content: attr(data-title);
47        }
48
49    }
```

The only additional expense is the repetitive usage of attributes to determine the lables with `data-title=""`. Alternatively you can (with jQuery) use JavaScript so the header fields get copied automatically.

```
1   $(function () {
2     var t = [];
3     $('thead th').each(function () {
4       t.push($(this).text());
5     });
6     $('tbody tr').each(function() {
7       $(this).find('td').each(function(i, e) {
8         $(e).attr('data-title', t[i]);
9       });
10    });
11  });
```

The script checks for the header fields and creates an array (line 4). Then, it searches for the rows (line 6) and in each row the `data-` attribute of each element of the array is filled in (line 8).

The repeated declaration of `data-title="Code"` is no longer necessary and can be removed completely.

No Bootstrap

The path shown in this section supplements the procedure in Bootstrap, but it is largely independent. The basic design and the distances were, however, taken from Bootstrap.

Auxiliary Classes

Bootstrap includes a number of helper classes for general purposes.

Check Distances

With the containers and the default behavior of the elements, the distances are usually set correctly. However, the result may not correspond to either your design or your aesthetic sensibilities. Even with the use of external components—Bootstrap covers all kinds of elements—there are occasionally unfortunate distances. For this case, there are correction classes. If it is unavoidable a static rule applies a margin by using `margin-top: 25px` emerges.

The peculiarity of the distance classes consists in the relation to the base font size *1rem*. The data are therefore relative.

The structure of the classes is very systematic.

```
{Property}-{Page}-{Size}
```

The property accepts the following values.

- *m*: Margin, the distance to the next element
- *p*: Padding, the inner distance of the contents of the element frame

The page is shown as follows:

- *t*: Set the property `margin-top` or `padding-top`
- *b*: Set the property `margin-bottom` or `padding-bottom`
- *l*: Set the property `margin-left` or `padding-left`
- *r*: Set the property `margin-right` or `padding-right`
- *x*: Set the property for *-left (left side) and *-right (right side)
- *y*: Set the property for *-top (above) and *-bottom (below)
- *a*: Sets the property of all margin or padding rules for all four sides

The size will be determined as follows:

- *0*: Sets all values to 0—no spaces
- *1*: Sets all values to the default distance
- *2*: Sets all values to 1.5 times the standard distance
- *3*: Resets all values to 3.0 times the standard distance; here are some of the definitions used for internal (in SASS).

```
.m-t-0 {
  margin-top: 0 !important;
}

.m-l-1 {
  margin-left: $spacer-x !important;
}

.p-x-2 {
  padding-left: ($spacer-x * 1.5) !important;
  padding-right: ($spacer-x * 1.5) !important;
}

.p-a-3 {
  padding: ($spacer-y * 3) ($spacer-x * 3) !important;
}
```

The variables *$spacer-x* and *$spacer-y* are standard distance.
A special feature is the class *.m-x-auto,* which sets the horizontal distance on auto.

CHAPTER 5

Forms

Forms are fully supported in Bootstrap 4. Many of the components are mainly used to make the forms responsive and can be used with any screen width.

Structure of a Form

Form elements automatically receive the correct formatting. The main class for controls is *.form-control*. Elements that have controllable horizontal extensions such as `<input>`, `<textarea>`, and `<select>` are set to a width of 100% of the parent container. Using *.form-group* the labels and inputs are grouped. They arrange themselves depending on the available width either side-by-side or above one another.

Simple Form Elements

Listing 5-1 gives an example of a typical form.

Listing 5-1. Standard structure of a form (Form_Base.html)

```
<form>
  <div class="form-group">
    <label for="txtMail">eMail</label>
    <input type="email" class="form-control"
           id="txtMail" placeholder="eMail">
  </div>
  <div class="form-group">
    <label for="txtPassword">password</label>
    <input type="password" class="form-control"
           id="txtPassword" placeholder="Password">
  </div>
  <div class="form-group">
    <label for="txtFile">File Selection</label>
    <input type="file" id="txtFile">
    <p class="form-text small">
      Here is the help for uploading.
    </p>
```


J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_5

```
18      </div>
19      <div class="checkbox">
20        <label>
21          <input type="checkbox"> Save
22        </label>
23      </div>
24      <button type="submit" class="btn btn-secondary">
25        Send
26      </button>
27    </form>
```

The outer section is always the element group *.form-group*. The element is again given the *.form-control* class. All the other parts do not require explicit classes. Figure 5-1 shows an example of a simple form.

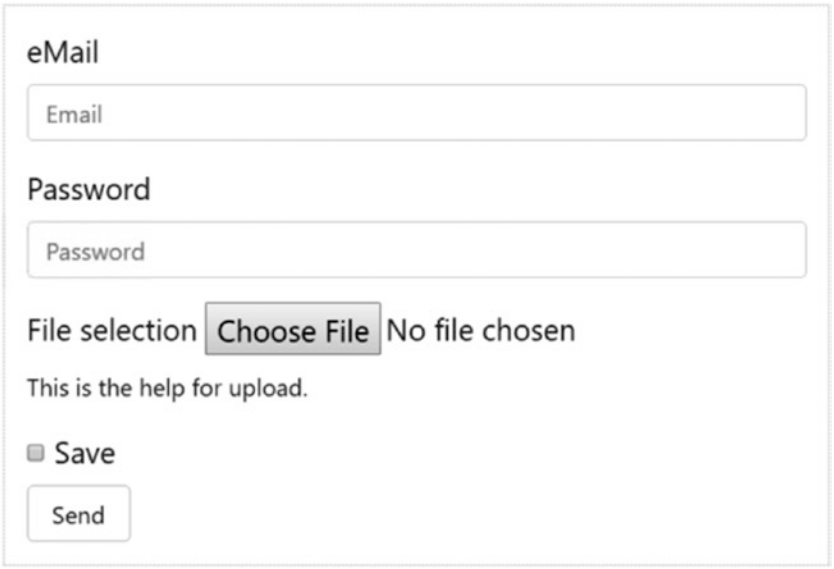


Figure 5-1. *A simple form*


In addition to simple elements, more complex elements can be built out of simpler ones and then put together. This is then referred to as an input group. Element groups and group entries may not be used in mixed input groups, but instead should be nested as a child element.

Single-Line Forms

Single-line forms are available from a width of 768 px and up. "One-line" means that the field name (label), box, and other elements are able to stand side-by-side, as long as the horizontal space is sufficient. They are preceded by *.form-inline*. Listing 5-2 is a compact form. The enclosing `<form>` tag is optional. It may be required by the logic of the page or the browser behavior—Bootstrap does not care. The width of the elements with variable expansion is "auto." Therefore, the width is optimized within the enclosing container. Sometimes, it may be necessary to control the width individually. Figure 5-2 shows a form with horizontal orientation.

Listing 5-2. Compact form (Form_Inline.html)

```
<form class="form-inline">
  <div class="form-group">
    <label for="exampleInputName2">Name</label>
    <input type="text"
           class="form-control"
           id="exampleInputName2"
           placeholder="The Name">
  </div>
  <div class="form-group">
    <label for="exampleInputEmail2">Email</label>
    <input type="email"
           class="form-control"
           id="exampleInputEmail2"
           placeholder="name@email.com">
  </div>
  <div class="form-group">
    <button type="submit" class="btn bnt-secondary float-md-right">
      Submit
    </button>
  </div>
</form>
```

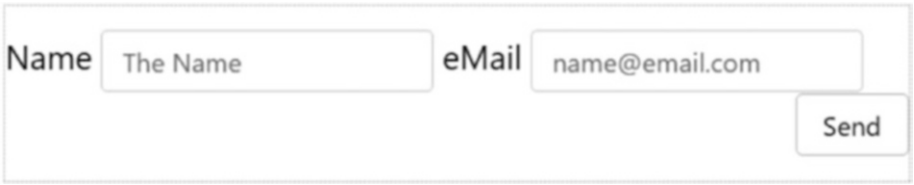


Figure 5-2. *A form with a horizontal orientation (wide)*

Figure 5-3 shows the same form at low screen width.

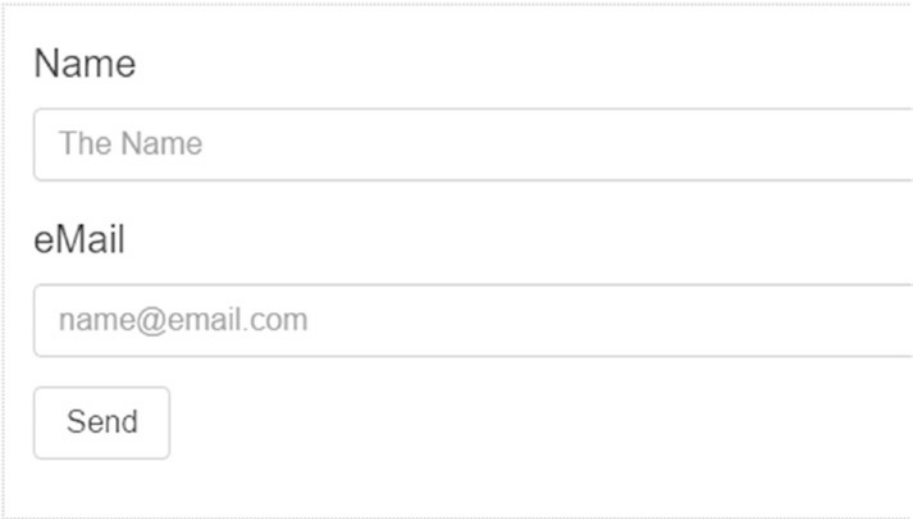


Figure 5-3. *A form with a horizontal orientation (narrow)*

Using Labels

You should always use labels even with one-line forms. Otherwise screen readers cannot produce meaningful speech. Use *.sr-only* so the label will not appear on regular output devices. If an accessible output is required, the standard attributes `aria-label` or `aria-labelledby` should be used. While `aria-label` contains the text directly; it is referenced by `aria-labelledby="id"` to the ID of another element on the page, which provides the label text.

Listing 5-3 shows the use of invisible text labels. For ordinary users, the watermark (`placeholder`) is used; however, on other screen readers, they can see the label. Figure 5-4 shows the form without the label.

Listing 5-3. Form with watermarks and screen readers

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputEmail">
      E-Mail
    </label>
    <input type="email" class="form-control"
           id="exampleInputEmail" placeholder="Email">
  </div>
  <div class="form-group">
    <label class="sr-only" for="exampleInputPassword">
      Password
    </label>
    <input type="password" class="form-control"
           id="exampleInputPassword" placeholder="Password">
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Remember
    </label>
  </div>
  <button type="submit" class="btn btn-secondary">
    Logon
  </button>
</form>
```

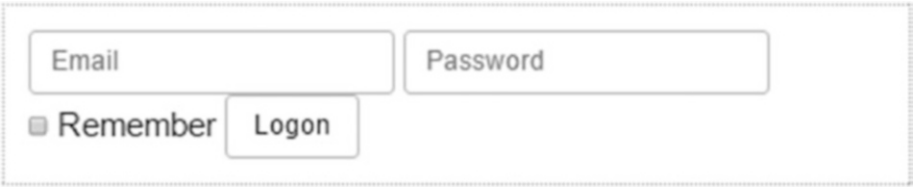


Figure 5-4. *A form without label*

Form Elements with Blocks

Input fields, especially those for text entry, can be supplemented for left or right text, icons, or buttons. This is particularly interesting when the values have units. Use the class *.input-group* with *.input-group-addon* to display elements before or after using *.form-control* to create. This works very well with all the `<input>`-elements, with limited success with `<select>`, but not with `<textarea>`.

Tooltips and overlapping effects also require additional effort. At least, using the optional *container: "body"* in JavaScript is required to avoid side effects. This parameter determines where the dynamic element in the DOM (document object model) of the page will be inserted.

```
$("#toolbarBtn1").dropdown({
    container: 'body'
});
```

The columns of the grid classes cannot mix with the input groups. Instead, the entire input group should be placed in a container if they are required, which in turn is provided with pitches.

Labels are always useful. Although there is no need for a specific layout, screen readers are supported. In Listing 5-4 and Figure 5-5 the use of *.sr-only* is reattached.

Listing 5-4. Fields (Toolbar_Inputgroups.html)

```
<div class="input-group">
  <span class="input-group-addon" id="basic-addon1">@</span>
  <input type="text" class="form-control"
         placeholder="User"
         aria-describedby="basic-addon1">
</div>

<div class="input-group">
  <input type="text" class="form-control"
         placeholder="Email"
         aria-describedby="basic-addon2">
  <span class="input-group-addon" id="basic-addon2">
    @example.com
  </span>
</div>

<div class="input-group">
  <span class="input-group-addon">&euro;</span>
  <input type="text" class="form-control"
         aria-label="Amount (in EUR)">
  <span class="input-group-addon">.00</span>
</div>
```

Figure 5-5. *Fields*

Sizes

Variable sizes can be set relative to the usual four stages. This happened to the whole group, so individual fields do not always have to be equipped with the class. Listing 5-5 and Figure 5-6 shows the sizes of the input fields.

Listing 5-5. Sizes of the input fields (Toolbar_InputSize.html)

```
<div class="input-group input-group-lg">
  <span class="input-group-addon" id="sizing-addon1">@</span>
  <input type="text" class="form-control"
         placeholder="Username"
         aria-describedby="sizing-addon1">
</div>

<div class="input-group">
  <span class="input-group-addon" id="sizing-addon2">@</span>
  <input type="text" class="form-control"
         placeholder="Username"
         aria-describedby="sizing-addon2">
</div>

<div class="input-group input-group-sm">
  <span class="input-group-addon" id="sizing-addon3">@</span>
  <input type="text" class="form-control"
         placeholder="Username"
         aria-describedby="sizing-addon3">
</div>
```

Figure 5-6. *Sizes of the input fields*

Dealing with Checkboxes and Radio Buttons

Checkboxes and radio buttons must be placed right next to input fields. Again, groups are used to force the mapping as shown in Listing 5-6.

Listing 5-6. Fields with options (Toolbar_Inputradio.html)

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="checkbox" aria-label="">
      </span>
    <input type="text" class="form-control" aria-label="">
  </div>
</div>
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="radio" aria-label="">
      </span>
      <input type="text" class="form-control" aria-label="">
    </div>
  </div>
</div>
```

To have a useful function, JavaScript is absolutely necessary here, for example, to activate the input box as seen in Figure 5-7.

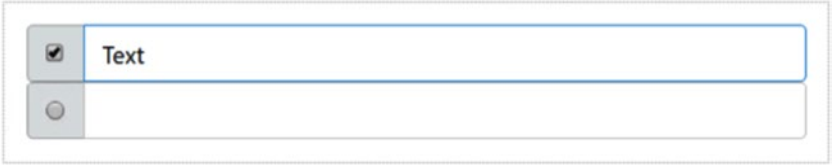


Figure 5-7. *Input fields with options*

Additional Buttons

Buttons in box groups are treated a little differently. They require another nesting of HTML elements based on the class *.input-group-btn*, which is shown in Listing 5-7 and Figure 5-8.

Listing 5-7. Fields with buttons (Toolbar_InputBtn.html)

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-btn">
        <button class="btn btn-default" type="button">
          Go!
        </button>
      </span>
      <input type="text" class="form-control"
             placeholder="Search for ...">
    </div>
</div>
  <div class="col-lg-6">
    <div class="input-group">
      <input type="text" class="form-control"
             placeholder="Search for...">
      <span class="input-group-btn">
        <button class="btn btn-default" type="button">
          Go!
        </button>
      </span>
    </div>
  </div>
</div>
```

Figure 5-8. *Input fields with buttons*

The buttons in turn can be provided with pop-up menus as illustrated in Listing 5-8 and Figure 5-9.

Listing 5-8. Fields with menus (Toolbar_DropDownGroup.html)

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <div class="input-group-btn">
        <button type="button"
                class="btn btn-default dropdown-toggle"
                data-toggle="dropdown" aria-haspopup="true"
                aria-expanded="false">Selection
          <span class="caret"></span>
        </button>
        <ul class="dropdown-menu">
          <li><a href="#">Details</a></li>
          <li><a href="#">Copy</a></li>
          <li><a href="#">Move</a></li>
          <li role="separator" class="divider"></li>
          <li><a href="#">Delete</a></li>
        </ul>
      </div>
      <input type="text" class="form-control" aria-label="">
    </div>
</div>
<div class="col-lg-6">
  <div class="input-group">
    <input type="text" class="form-control" aria-label="">
    <div class="input-group-btn">
      <button type="button"
              class="btn btn-default dropdown-toggle"
              data-toggle="dropdown"
              aria-haspopup="true"
              aria-expanded="false">Selection
        <span class="caret"></span>
      </button>
      <ul class="dropdown-menu dropdown-menu-right">
        <li><a href="#">Details</a></li>
        <li><a href="#">Copy</a></li>
        <li><a href="#">Move</a></li>
        <li role="separator" class="divider"></li>
        <li><a href="#">Delete</a></li>
      </ul>
    </div>
  </div>
</div>
</div>
```

Figure 5-9. *Input fields with menus*

Likewise, dividing buttons into segments is possible as shown in Listing 5-9 and Figure 5-10.

Listing 5-9. Fields with segment buttons (Toolbar_InputBtnForm.html)

```
<div class="input-group">
  <div class="input-group-btn">
    <button class="btn btn-success">Enable</button>
  </div>
  <input type="text" class="form-control" aria-label="...">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="...">
  <div class="input-group-btn">
    <button class="btn btn-warning">Disable</button>
  </div>
</div>
```

Figure 5-10. *Input fields with segment buttons*

Listing 5-10 gives a form element that consists of three components, a Euro symbol, the box, and the characters ".00" as a predefined fractional part.

Listing 5-10. Form with functional blocks *(Form_MultiInput.html)*

```
1   <form class="form-inline">
2     <div class="form-group">
3       <label class="sr-only" for="exampleInputAmount">
4         Amount (in USD)
5       </label>
6       <div class="input-group">
7         <div class="input-group-addon">$</div>
8         <input type="text" class="form-control"
9                id="exampleInputAmount" placeholder="Amount">
10        <div class="input-group-addon">.00</div>
11      </div>
12    </div>
13    <button type="submit" class="btn btn-primary">
14      Wire
15    </button>
16  </form>
```

Here the class *.input-group* is used. The label (lines 3 to 5) appears only on screen readers, but otherwise remains invisible. Figure 5-11 shows the form with no label. The "extras" before and after the element are to be decorated with *.input-group-addon*.

Figure 5-11. *A form with no label*

Horizontal Forms

To place labels and fields, a grid is used. Thus, breaks are suppressed. The complete sections of label from one group are normally separated from another using *.form-horizontal* (see Listing 5-11). The particular class is carried in a container element, it is either the enclosing `<form>` element or an equivalent `<div>` is employed. The class *.row* within this container is not necessary. The required grouping *.form-groups* means that the group behaves like one element. You can use *.row* but some editors complain about it as the visible effects are not achieved.

Listing 5-11. Horizontal form (Form_Horizontal.html)

```
1   <form class="form-horizontal">
2     <div class="form-group">
3       <label for="inputEmail3" class="col-sm-2
4              form-form-control-label">E-Mail</label>
```

```
5       <div class="col-sm-10">
6       <input type="email" class="form-control"
7              id="inputEmail3" placeholder="eMail">
8       </div>
9     </div>
10    <div class="form-group">
11      <label for="inputPassword3" class="col-sm-2
12             form-form-control-label">Password</label>
13      <div class="col-sm-10">
14        <input type="password" class="form-control"
15               id="inputPassword3" placeholder="Password">
16      </div>
17    </div>
18    <div class="form-group">
19      <div class="offset-sm-2 col-sm-10">
20        <div class="checkbox">
21          <label>
22            <input type="checkbox"> Remember access
23          </label>
24        </div>
25      </div>
26    </div>
27    <div class="form-group">
28      <div class="offset-sm-2 col-sm-10">
29        <button type="submit" class="btn btn-secondary">
30          Log In
31        </button>
32      </div>
33    </div>
34  </form>
```

In Figure 5-12 the labels are to the left of the elements. When you are working with layouts with narrow widths (in the figure the width is > 768 px), you must go back to the previously shown scheme about placed labels.

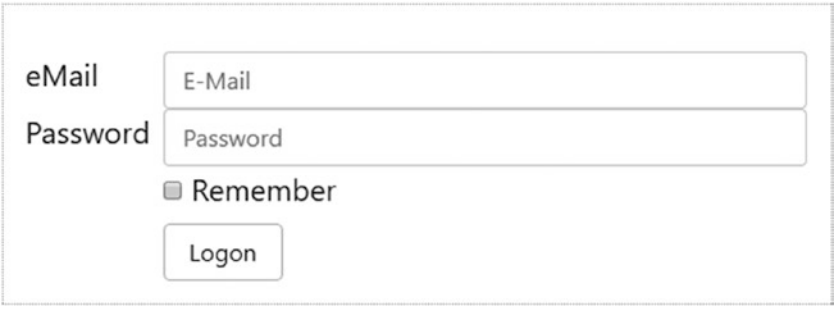


Figure 5-12. *A form with a label next to the field*

Input Elements

Form elements of the group `<input>` require the `type`-Attribute to appear properly. They support virtually all the following features of HTML5 types:

- text
- password
- datetime
- datetime-local
- date
- month
- time
- week
- number
- email
- url
- search
- tel
- color

Input Elements

```
<input type="text"
       class="form-control"
       placeholder="Text input">
```

Text fields—that is, multiline edit boxes, drop out of this scheme:

```
<textarea class="form-control" rows="3" cols="55"></textarea>
```

Checkboxes and radio buttons are function as in the standard HTML. Also locking of an element is supported with the attribute `disabled`. Lock itself works not only with the control, but also relates to the associated label using the class *.disabled*. This class is also associated with *.radio*, *.radio-inline*, *.checkbox*, *.checkbox-inline*, or `<fieldset>`. They can be used as in Listing 5-12 to format the enclosing container fits.

Listing 5-12. Checkboxes and radio buttons (Form_Elements.html)

```
<div class="checkbox">
  <label>
    <input type="checkbox" value="">
```

```
      Option One
    </label>
  </div>
  <div class="checkbox disabled">
    <label>
      <input type="checkbox" value="" disabled>
      Option Two (disabled)
    </label>
  </div>

  <div class="radio">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios1" value="option1" checked>
      Option One
    </label>
  </div>
  <div class="radio">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios2" value="option2">
      Option Two
    </label>
  </div>
  <div class="radio disabled">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios3" value="option3" disabled>
      Option Three
    </label>
  </div>
```

The <label> element enclosing the radio button or checkbox shown here in Figure 5-13 . In this way, a single click triggers the label text from the action, resulting in a more pleasant user experience results.

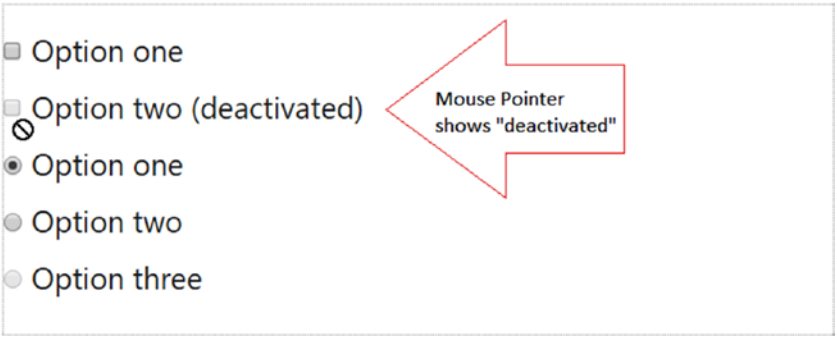


Figure 5-13. *Checkboxes and radio buttons*

The classes *.checkbox-inline* or *.radio-inline* allow a string of elements next to each other (see Listing 5-13 and Figure 5-14).

Listing 5-13. Checkboxes and radio buttons next to each other (Form_ElementsHor.html)

```
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox1" value="1"> 1
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox2" value="2"> 2
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox3" value="3"> 3
</label>

<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions"
         id="inlineRadio1" value="1"> 1
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions"
         id="inlineRadio2" value="2"> 2
  </label>
  <label class="radio-inline">
    <input type="radio" name="inlineRadioOptions"
           id="inlineRadio3" value="3"> 3
</label>
```

Figure 5-14. *Checkboxes and radio buttons next to each other*

If default behavior is required, but no label text is needed, delete the text and the label surrounding the box.

```
<div class="checkbox">
  <label>
    <input type="checkbox" id="blankCheckbox" value="1"
           ariOO61-label="...">
  </label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="blankRadio" id="blankRadio1"
```

```
10              value="1" aria-label="...">
11     </label>
12   </div>
```

Select Elements

Lists and drop-downs (drop-down menus) can be used as usual. However, some browsers turn to internal styles that cannot be influenced with CSS. It could be that menus are a better choice to get full control over the design.

For simple folding menus select one of the following options.

```
1   <select class="form-control">
2     <option>1</option>
3     <option>2</option>
4     <option>3</option>
5     <option>4</option>
6     <option>5</option>
7   </select>
```

With the attribute `multiple` it is possible to select multiple options.

```
1   <select multiple class="form-control">
2     <option>1</option>
3     <option>2</option>
4     <option>3</option>
5     <option>4</option>
6     <option>5</option>
7   </select>
```

Static Texts in Forms

Help texts and static blocks are provided with *.form-control-static* by a defined section <p>. Listing 5-14 shows static elements in the layout and Figure 5-15 shows the static elements in a horizontal layout. Listing 5-15 shows the display fields (help texts) and Figure 5-16 shows.

Listing 5-14. Static elements in the layout (Form_Help.html)

```
1   <form class="form-horizontal">
2     <div class="form-group">
3       <label class="col-sm-2 form-form-control-label">eMail</label>
4       <div class="col-sm-10">
5         <p class="form-control-static">email@example.com</p>
6       </div>
7     </div>
8     <div class="form-group">
9       <label for="inputPassword" class="col-sm-2 form-control-lab\
```

```
el">
      Password
    </label>
    <div class="col-sm-10">
      <input type="password" class="form-control"
             id="inputPassword" placeholder="Password">
    </div>
  </div>
</form>
```

Figure 5-15. *Static elements in horizontal layout*

Listing 5-15. Display fields (Form_HelpInline.html)

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only">eMail</label>
    <p class="form-control-static">email@example.com</p>
  </div>
  <div class="form-group">
    <label for="inputPassword2" class="sr-only">
      Password
    </label>
    <input type="password" class="form-control"
           id="inputPassword2" placeholder="Password">
    </div>
    <button type="submit" class="btn btn-secondary">
      Confirm
    </button>
</form>
```

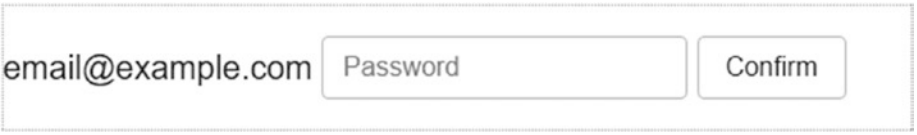


Figure 5-16. *Static elements in horizontal layout*

Behavior of the Form Elements

Form elements respond dynamically to the focus. The frame is removed and when obtaining the focus a soft shadow appears. Trigger is the pseudoclass *:focus*.

Disabled controls use `disabled` (line 5) and are presented somewhat brighter.

```
1   <input class="form-control"
2          id="disabledInput"
3          type="text"
4          placeholder="Locked content..."
5          disabled>
```

If you use `<fieldset>` and `disabled` you can block several elements together. However, the blocking of the user actions relates only to regular form elements (see Listing 5-16 and Figure 5-17). Hyperlinks are not blocked. Although if hyperlinks are formatted as buttons (`<a ... class="btn btn-*">`), they act more like links. The desired effect can be done with JavaScript. However, Bootstrap works only with a "gray" appearance.

Listing 5-16. Locked fields (Form_Disabled.html)

```
 1  <form>
 2    <fieldset disabled>
 3      <div class="form-group">
 4        <label for="disabledTextInput">Disabled Field</label>
 5        <input type="text"
 6               id="disabledTextInput"
 7               class="form-control"
 8               placeholder="Disabled field">
 9      </div>
10      <div class="form-group">
11        <label for="disabledSelect">Disabled Menu </label>
12        <select id="disabledSelect" class="form-control">
13          <option>Option A Disabled</option>
14          <option>Option B Disabled</option>
15        </select>
16      </div>
17      <div class="checkbox">
18        <label>
19          <input type="checkbox"> Checkbox
20        </label>
21      </div>
22      <button type="submit" class="btn btn-primary">
23        Send
24      </button>
25    </fieldset>
26  </form>
```

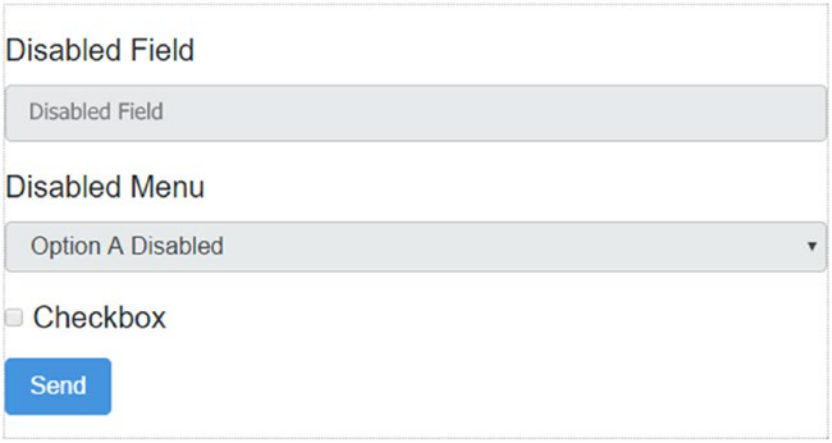


Figure 5-17. *Locked elements (with mouse)*

In addition to blocking or lock elements, you can use a similar condition for "read only." This is based on the HTML attribute `readonly`. The element is brighter and therefore first looks like `disabled`, however, the pointer remains unchanged and does not show the lock icon (see Listing 5-17).

Listing 5-17. Read-only mode (excerpt from Form_ReadOnly.html)

```
1   <input class="form-control"
2          type="text"
3          placeholder="Read Only..."
4          readonly>
```

In particular, the behavior of drop-down lists is different. Unlike `disabled` the user can select from the elements and view them. A selection of data when sending the read-only form still does not succeed. Figure 5-18 shows a read-only form.

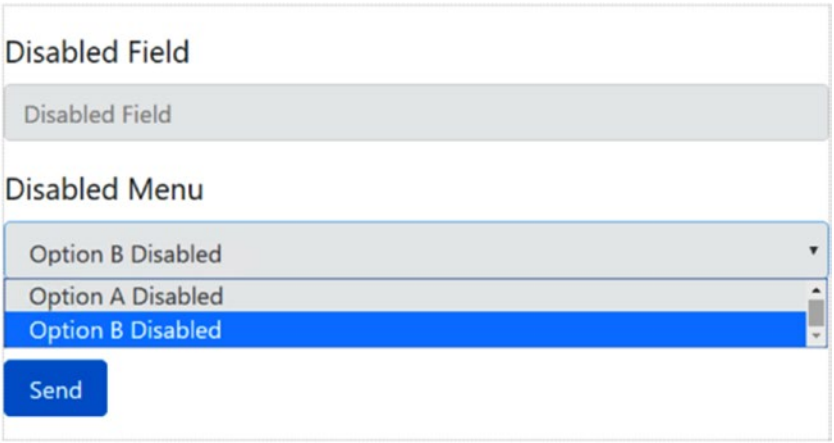


Figure 5-18. *Elements in "read-only" mode*

Validation Information in Forms

Bootstrap basically has three states: error, warning, and success.

These are represented by the following semantic classes.

- *.has-warning*: Warning
- *.has-has-danger*: Error
- *.has-success*: Success

Elements that have either of the classes *.form-controllabel*, *.form-control*, or *.help-block* can be modified to match their status (see Listing 5-18 and Figure 5-19).

Accessibility

The status shows with color. Colorblind users may not benefit from this information. Therefore, the semantic meaning should always accompany the color information. This can be achieved by help texts, symbols, and *.sr-only* regions. Elements that cannot be adapted should carry an `aria-invalid="true"` attribute.

Listing 5-18. Semantic information in the input fields (Form_Semantic.html)

```
<div class="form-group has-success">
  <label class="form-form-control-label" for="inputSuccess1">
    Success message
  </label>
  <input type="text" class="form-control" id="inputSuccess1">
</div>
<div class="form-group has-warning">
  <label class="form-form-control-label" for="inputWarning1">
    A warning
  </label>
  <input type="text" class="form-control" id="inputWarning1">
</div>
<div class="form-group has-danger">
  <label class="form-form-control-label" for="inputError1">
    Fault condition
  </label>
  <input type="text" class="form-control" id="inputError1">
</div>
<div class="has-success">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxSuccess" value="1">
      Successful
    </label>
```

```
    </div>
  </div>
  <div class="has-warning">
    <div class="checkbox">
      <label class="form-control-label">
        <input type="checkbox" id="checkboxWarning" value="2">
        Warning
      </label>
    </div>
  </div>
  <div class="has-danger">
    <div class="checkbox">
      <label class="form-control-label">
        <input type="checkbox" id="checkboxError" value="3">
        Fault condition
      </label>
    </div>
  </div>
```

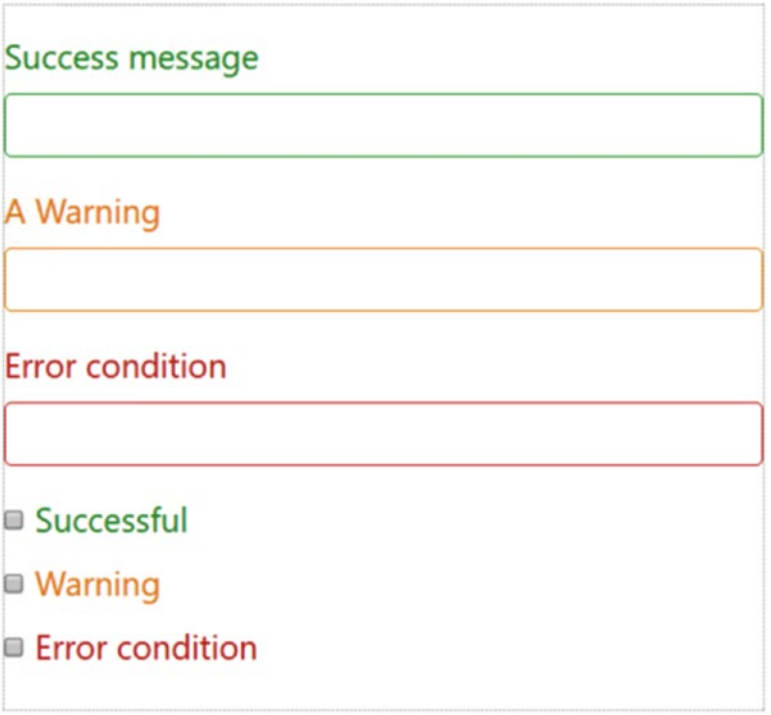


Figure 5-19. *Semantic form elements*

As an option, use states with upgraded symbols. This can be accomplished with *.hasfeedback* and a symbol on the right end of the message areas. These icons will only work if `<input class="form-control">` is used.

Listing 5-19 shows what a completely accessible form can look like.

Listing 5-19. Semantic and disability aware form (Form_AriaSemantic1.html)

```
<form>
  <div class="form-group has-success has-feedback">
    <label class="form-control-label text-success" for="inputSuccess\2">
Success</label>
    <input type="text" class="form-control form-control-success"
    id="inputSuccess2"
          aria-describedby="inputSuccess2Status">
    <span id="inputSuccess2Status" class="sr-only">(Success)</span>
  </div>
  <div class="form-group has-warning has-feedback">
    <label class="form-control-label text-warning" for="inputWarning\2">
With Warning</label>
    <input type="text" class="form-control form-control-warning"
    id="inputWarning2"
          aria-describedby="inputWarning2Status">
    <span id="inputWarning2Status" class="sr-only">(Warning)</span>
  </div>
  <div class="form-group has-danger has-feedback">
    <label class="form-control-label text-danger" for="inputError2">\
    With Error</label>
    <input type="text" class="form-control form-control-danger"
    id="inputError2"
          aria-describedby="inputError2Status">
    <span id="inputError2Status" class="sr-only">(Error)</span>
  </div>
  <div class="form-group has-success has-feedback">
    <label class="form-control-label"
    for="inputGroupSuccess1">Succe\
    ssful Group </label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input type="text" class="form-control form-control-success"
id="inputGroupSuccess1"
            aria-describedby="inputGroupSuccess1Status">
    </div>
    <span id="inputGroupSuccess1Status" class="sr-only">(Success)</s\
pan>
  </div>
</form>
```

Unlike Bootstrap 3, there are some changes. First, the class names were unified. The class *.control-label* is now *.form-controllabel*. It still governs distance, not the color of the label. If you want the label to be highlighted as well, then you have to use the classes *.text-success* or *.text-danger*, respectively.

The Glyphicons (font-based icons) are completely omitted and only the semantic classes *.form-control-success*, *.form-control-error*, and so forth produce output (see Figure 5-20).

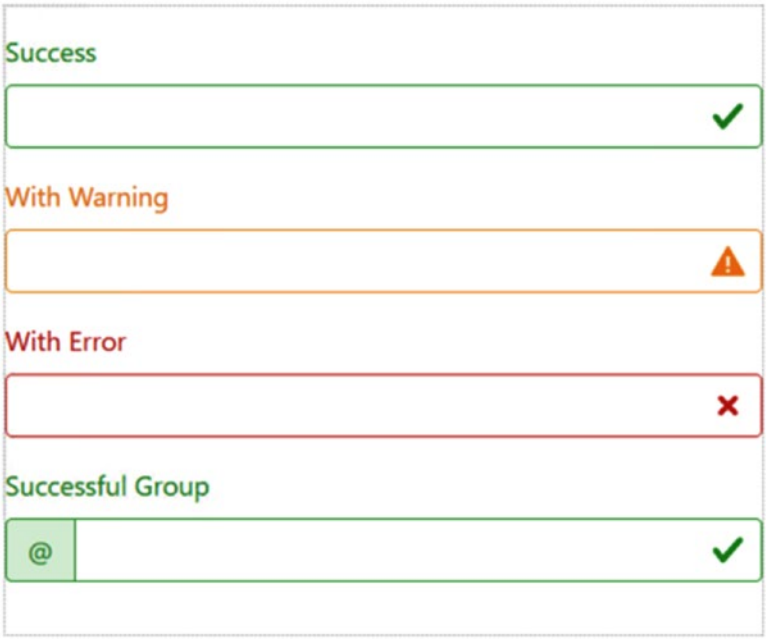


Figure 5-20. *Semantic elements with ARIA support*

Symbols designed in one line or horizontally are defined in Listing 5-20 and Figure 5-21.

Listing 5-20. Semantic and accessibility form (Form_AriaSemantic.html)

```
<form>
  <div class="form-group has-success has-feedback">
    <label class="form-control-label col-sm-3"
           for="inputSuccess3">
      Success
    </label>
    <input type="text" class="form-control form-control-success"
           id="inputSuccess3"
           aria-describedby="inputSuccess3Status">
    <span id="inputSuccess3Status" class="sr-only">(Success)</span>
  </div>
```

```
    <div class="form-group has-error has-feedback">
      <label class="form-control-label col-sm-3"
             for="inputGroupError2">Error</label>
      <div class="input-group">
        <span class="input-group-addon">@</span>
        <input type="text" class="form-control form-control-error"
               id="inputGroupError2"
               aria-describedby="inputGroupSuccess2Status">
        </div>
        <span id="inputGroupSuccess2Status" class="sr-only">(Error)
  </spa\n>
    </div>
  </form>
```

Figure 5-21. *Semantic elements with ARIA support*

Listing 5-21 gives a form with a success message (see Figure 5-22).

Listing 5-21. Semantic messages (Form_Success.html)

```
<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="form-form-control-label" for="inputSuccess4">
      Success
    </label>
    <input type="text" class="form-control"
           id="inputSuccess4"
           aria-describedby="inputSuccess4Status">
    <span class="glyphicon glyphicon-ok form-control-feedback"
           aria-hidden="true"></span>
    <span id="inputSuccess4Status" class="sr-only">
      (Success)
    </span>
  </div>
</form>
```

```
<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="form-form-control-label" for="inputGroupSuccess3">
      Successful group
    </label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input type="text" class="form-control"
             id="inputGroupSuccess3"
             aria-describedby="inputGroupSuccess3Status">
    </div>
    <span class="glyphicon glyphicon-ok form-control-feedback"
          aria-hidden="true"></span>
    <span id="inputGroupSuccess3Status" class="sr-only">
      (Success)
    </span>
  </div>
</form>
```

Figure 5-22. *Semantic messages*

Optional symbols with hidden `.sr-only` sections support screen readers. The icon is automatically properly aligned as shown in Listing 5-22.

Listing 5-22. Messages for screen readers (Form_SuccessSR.html)

```
<div class="form-group has-success has-feedback">
  <label class="form-form-control-label sr-only" for="inputSuccess5">
    Hidden label
  </label>
  <input type="text" class="form-control"
         id="inputSuccess5"
         aria-describedby="inputSuccess5Status">
  <span class="fa fa-check form-control-feedback"
        aria-hidden="true"></span>
  <span id="inputSuccess5Status" class="sr-only">
    (Success)
  </span>
</div>
<div class="form-group has-success has-feedback">
```

```
    <label class="form-form-control-label sr-only"
           for="inputGroupSuccess4">
      Successful Group
    </label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input type="text" class="form-control"
             id="inputGroupSuccess4"
             aria-describedby="inputGroupSuccess4Status">
    </div>
    <span class="fa fa-check form-control-feedback"
          aria-hidden="true"></span>
    <span id="inputGroupSuccess4Status" class="sr-only">
      (Success)
    </span>
  </div>
```

Form Elements in the Grid

Use the usual techniques when formatting width and height. For the width make use of the grid, with classes such as *.col-lg-3* or *.col-md-5* (see Listing 5-23 and Figure 5-23).

Listing 5-23. Arrangement in the grid (Form_Grid.html)

```
<div class="row">
  <div class="col-xs-2">
    <input type="text"
           class="form-control"
           placeholder="2 columns">
  </div>
  <div class="col-xs-3">
    <input type="text"
           class="form-control"
           placeholder="3 columns">
  </div>
  <div class="col-xs-4">
    <input type="text"
           class="form-control"
           placeholder="4 columns">
  </div>
</div>
```

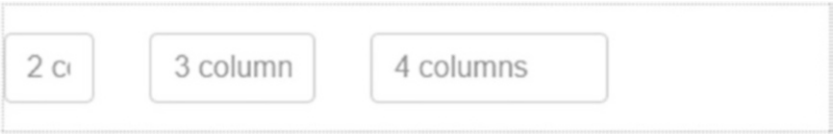


Figure 5-23. *Shrinked elements*

Adaptation of the Field Height

There are two classes for controlling height: *.input-sm* and *.input-lg* (see Listing 5-24 and Figure 5-24).

Listing 5-24. Size of the elements (Form_ElementsSize.html)

```
<input class="form-control input-lg"
       type="text" placeholder="Big">
<input class="form-control"
       type="text" placeholder="Normal">
<input class="form-control input-sm"
       type="text" placeholder="Small">

<select class="form-control input-lg">...</select>
<select class="form-control">...</select>
<select class="form-control input-sm">...</select>
```

Figure 5-24. *Elements in various sizes*

The size of horizontal running forms can be controlled globally. For this purpose, the class *.form-horizontal* can use either class *.form-group-lg* or *.form-group-sm* (see Listing 5-25 and Figure 5-25).

Listing 5-25. Size of the elements (Form_ElementsSizeHor.html)

```
<form>
  <div class="form-group row">
  <label class="col-sm-2 form-control-label"
         for="formGroupInputLarge">Big Label</label>
  <div class="col-sm-10">
      <input class="form-control form-control-lg" type="text"
             id="formGroupInputLarge" placeholder="Big input">
  </div>
  </div>
  <div class="form-group row">
  <label class="col-sm-2 form-control-label small"
         for="formGroupInputSmall">Small Label</label>
  <div class="col-sm-10">
      <input class="form-control form-control-sm" type="text"
             id="formGroupInputSmall" placeholder="Small input">
  </div>
  </div>
</form>
```

Figure 5-25. *Elements aligned*

Help Text in Forms

Help texts for forms are often needed (see Listing 5-26 and Figure 5-26). You have one for display and one for screen readers to support accessible pages. For these to work, the matching *aria* attributes should always be used with the Bootstrap classes together.

Listing 5-26. Help text in forms (Form_ElementsHelp.html)

```
<label class="sr-only" for="inputHelpBlock">
  Your entry
</label>
<input type="text" id="inputHelpBlock" class="form-control"
       aria-describedby="helpBlock">
...
```

```
7   <span id="helpBlock" class="text-gray-dark small">
8     You can enter what here.
9   </span>
```

Figure 5-26. *Help texts*

Buttons

Buttons can be found in one form or another in each web application. Bootstrap buttons are seen as a design element that is separate from the technical framework. Technically, a "submit" appears only once in a form, which almost always triggers a POST request. If, however, you need access to the server via a GET request, a hyperlink is used. In design terms Bootstrap now provides both on the same level. Buttons can be created with `<a>`, `<button>`, or `<input>` (see Listing 5-27 and see Figure 5-27).

Listing 5-27. Buttons (Form_Buttons.html)

```
1   <a class="btn btn-secondary" href="#" role="button">
2     Link
3   </a>
4   <button class="btn btn-secondary" type="Submit">
5     Button
6   </button>
7   <input class="btn btn-secondary"
8          type="button" value="Input">
9   <input class="btn btn-secondary"
10         type="submit" value="Submit">
```

Figure 5-27. *Buttons—several variants of the same effect*

However there is one restriction: in navigation bars built with *.nav* or *.navbar* only the common element `<button>` is allowed.

If `<a>` tags are used as a button, they should only serve the navigation and not trigger as a hyperlink because this is not the expected behavior. Moreover, it should be emphasized by the attribute `role="button"`. Even then it is questionable whether this is a good idea as not all browsers can ensure that the function will behave as expected. So take advantage of the distinct buttons and whenever possible it is always preferable to use `<button>`.

Semantic Buttons

Buttons are available in seven variants with the following semantic expressions (see Listing 5-28 and Figure 5-28).

- *Primary*: Raises the main function, visual effect by blue color amplified
- *Secondary*: Standard, gray, general function
- *Success*: Success, green, positive, or affirmative action
- *Info*: Information, azure, clarified highlighting critical or special actions
- *Warning*: Warning, orange, critical or complex action, should be taken with caution
- *Danger*: Danger, red, dangerous or irreversible action, negative
- *Link*: Formats a button as a link, often referred to as reduced action, additional behavior

Listing 5-28. Semantic buttons

```
<button type="button" class="btn btn-secondary">Std.</button>

<button type="button" class="btn btn-primary">Primary</button>

<button type="button" class="btn btn-success">Succes</button>

<button type="button" class="btn btn-info">Info</button>

<button type="button" class="btn btn-warning">Warning</button>

<button type="button" class="btn btn-danger">Danger</button>
```

Figure 5-28. *Semantic buttons*

Assigning semantic meanings on the color labels may be restricted by size in some environments. Additional labels, which are equipped with *.sr-only* should emphasize the intended effect.

Size and Appearance

For the larger or smaller button classes *.btn-lg*, *.btn-sm*, or *.btn-xs* should be used (see Figure 5-29). They complement the base class *btn* and are independent of color. The standard size is ready for touch operation. Mouse controlled sites will look better with the smaller buttons.

```
<p>
  <button type="button" class="btn btn-primary btn-lg">
    Big
  </button>
  <button type="button" class="btn btn-secondary btn-lg">
    Big
  </button>
</p>
<p>
  <button type="button" class="btn btn-primary">
    Standard
  </button>
  <button type="button" class="btn btn-secondary">
    Standard
  </button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">
    Small
  </button>
  <button type="button" class="btn btn-secondary btn-sm">
    Small
  </button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs">
    Tiny
  </button>
  <button type="button" class="btn btn-secondary btn-xs">
    Tiny
  </button>
</p>
```

Figure 5-29. *Size of the buttons*

Sometimes buttons will completely fill their containers. This is the purpose of the class *.btn-block* (see Listing 5-29).

Listing 5-29. Buttons horizontally extended (Form_ButtonsContainer.html)

```
<button type="button"
        class="btn btn-primary btn-lg btn-block">
  Block
</button>
<button type="button"
        class="btn btn-secondary btn-lg btn-block">
  Block
</button>
```

Note here that this only affects the width. Appropriate height must be reached separately with *.btn-lg* or other size as necessary (see Figure 5-30).

Figure 5-30. *Buttons horizontally expanded*

In addition to a base color, buttons may have states. The "pressed" state is reached or when formatted hyperlinks *.active*. The following was inserted for a barrier-free expression `aria-pressed="true"`.

```
<button type="button" class="btn btn-primary btn-lg active">
  Primary
</button>
<button type="button" class="btn btn-secondary btn-lg active">
  Standard
</button>
```

New in Bootstrap 4 is displaying framed buttons.

The visual impact of previous buttons is avoided. So far the elements benefit from a weak gradient, which produced a slight 3D effect. They are often used in newer sites with modern flat, reduced designs that seem all but overloaded. Listing 5-30 shows you how to grab the "outline" buttons.

Listing 5-30. Semantic buttons with frame

```
<button type="button"
        class="btn btn-outline-secondary">Std.</button>

<button type="button"
        class="btn btn-outline-primary">Primary</button>

<button type="button"
        class="btn btn-outline-success">Succes</button>

<button type="button"
        class="btn btn-outline-info">Info</button>

<button type="button"
        class="btn btn-outline-warning">Warning</button>

<button type="button"
        class="btn btn-outline-danger">Danger</button>
```

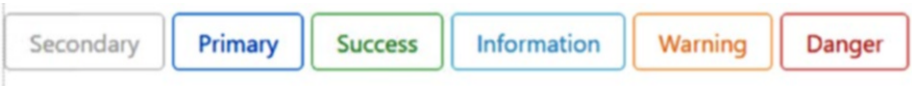


Figure 5-31. *Semantic buttons with frame*

States

The disabled state can be achieved on hyperlinks and on buttons. First an example for hyperlinks:

```
<a href="#"
   class="btn btn-primary btn-lg active"
   role="button">
  Primary
</a>
<a href="#"
   class="btn btn-secondary btn-lg active"
   role="button">
  Link
</a>
```

The “disabled” state indicates that the button cannot be pressed. For this, the HTML attribute `disabled` is used. Figure 5-32 shows various states of a button.

```
<button type="button"
        class="btn btn-lg btn-primary"
        disabled="disabled">Primary</button>
<button type="button"
        class="btn btn-secondary btn-lg"
        disabled="disabled">Standard</button>
```

If a hyperlink is formatted as a button the class *.disabled* is being used.

```
<a href="#"
  class="btn btn-primary btn-lg disabled"
  role="button">Primary</a>
<a href="#"
  class="btn btn-secondary btn-lg disabled"
  role="button">Link</a>
```

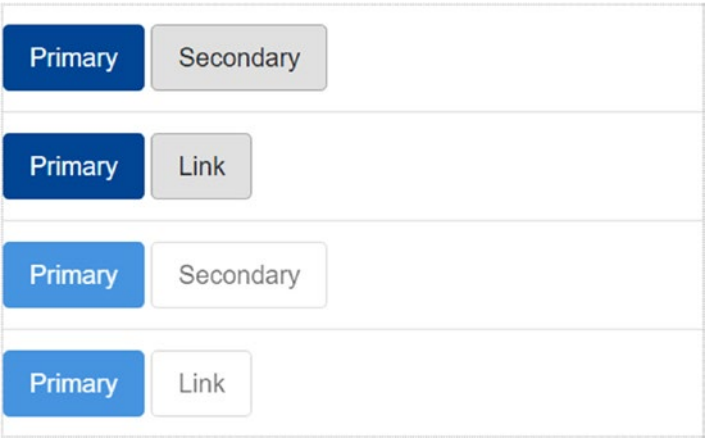


Figure 5-32. *Various states of a button*

Note that the cut-off of elements with CSS classes is merely cosmetic. Also the use of *pointer-events: none* internally does not prevent those elements from being selected with the keyboard, and they can still trigger actions. You need to check with JavaScript that the function will coincide with the presentation.

CHAPTER 6

Additional Modules

This chapter is about images, effects, and symbols. The only images supported are some effects that are available here.

Symbols

The symbols—font based icons—are the basis of the symbol functions. The glyphs used in Bootstrap 3 are called Halflings (a reference to *Lord of the Rings*) and are usually not free. Bootstrap 4 does not contain the Glyphicons anymore, and you have to use one of the many font libraries.

Why Symbol Fonts?

Font-based icons avoid the problem that individual symbols lead to a flood of additional requests on the server. Instead, all the symbols are loaded as a font—that is, in one file. However, the symbol is then like a letter. It is changeable in size and extent, but it may take only one color. Fonts are also mainly flat, 3D effects are eliminated. For fast, modern websites so-called glyph fonts are an established technology.

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_6

Alternatives

The following alternatives are currently available (see Figure 6-1 for a sample of Octicons):

- Font Awesome[1]—675 symbols
- Octicons, the Github Icons[2]—160 symbols
- Elegant Icon Font[3]—360 symbols
- Typicons[4]—336 symbols
- Meteocons[5]—40+ weather symbols
- Open Iconic[6]—223 symbols, which can be up to 8 px down

This is only a small selection and provided here to encourage you to seek the right support before your first design experiments.

[1] http://fortawesome.github.io/Font-Awesome/
[2] https://octicons.github.com/
[3] http://www.elegantthemes.com/blog/resources/elegant-icon-font
[4] http://typicons.com
[5] http://www.alessioatzeni.com/meteocons
[6] https://useiconic.com/open

Figure 6-1. *The free symbol font Octicons*

Use of Symbols

The following example illustrates the procedure using the library *Font Awesome*. Get the files with Bower, for example:

```
$ bower install Font-Awesome
```

The configuration file *bower.json* looks like this (excerpt):

```
{
  "name": "Demo",
  "private": true,
  "dependencies": {
    "bootstrap": "4.0",
    "Font-Awesome": "4.7.0"
  }
}
```

In the HTML file another link comes to the CSS definitions:

```
<link href="../bower_components/Font-Awesome/css/font-awesome.css"
      rel="stylesheet" />
```

To keep the definition compact, all symbols are based on a base class and a class for the icon, which achieves the effect. Note that the symbol is technically a character. If an icon appears on a button in the text or as an addition of a text, it should be separated by a space.

Symbol classes should always be an exclusive item, and you should not divide the element with other classes. When in doubt, add an extra `<span>`-Element. Avoid child elements as well. If symbols have no semantic meaning, but merely serve decorative purposes, avoid problems with screen readers with `aria-hidden="true"`. Conversely, you should explicitly support screen readers when the symbol has a meaning, and add text hidden with `.sr-only`.

```
<span class="fa fa-search"
      aria-hidden="true"></span>
```

Use on Buttons

Listing 6-1 and Figure 6-2 give some examples to show how symbols can be used on buttons.

Listing 6-1. Buttons with icons (Icons_Btn.html)

```
<button type="button"
        class="btn btn-secondary"
        aria-label="Left Align">
```

```
    <span class="fa fa-align-left"
          aria-hidden="true">
    </span>
  </button>

  <button type="button"
          class="btn btn-secondary btn-lg">
    <span class="fa fa-star"
          aria-hidden="true">
    </span> Star
  </button>
```

Figure 6-2. *Buttons with icons*

Messages

In a message symbols are suitable for highlighting or for an opportunity to dismiss the message (see Listing 6-2 and Figure 6-3).

Listing 6-2. Messages with symbols (Icons_Messages.html)

```
  <div class="alert alert-danger" role="alert">
    <span class="fa fa-exclamation"
          aria-hidden="true"></span>
    <span class="sr-only">Error</span>
    Please enter a valid address
  </div>
```

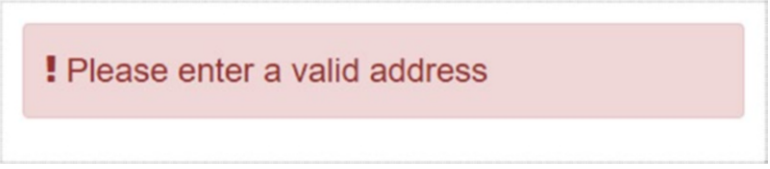


Figure 6-3. *Messages with symbols*

Common Symbols

You can define an icon that closes dialogs or messages as follows (see Figure 6-4).

```
1   <button type="button" class="close" aria-label="Close">
2   Message
3   </button>
4   <span aria-hidden="true">&times;</span></button>
```

Figure 6-4. *Effect of a "close" button (nonfunctional)*

To indicate a context menu use a *.caret* as shown in Figure 6-5.

```
1   <span class="caret"></span>
```

Figure 6-5. *Effect of context menus (non functional)*

Responsive Images

Images typically have natural expansion. On large screens this is not critical, however shown on small devices they may leave the viewing area. With *.img-fluid* you can present better images.

This class enforces three actions:

- *max-width: 100%*
- *height: Auto*
- *display: Block*

Thus, the image is scaled to the device width and remains stable. Should it be extremely reduced compared to the natural size, you may want to consider scaling with a server-side solution. By not doing so you waste a lot of bandwidth and have little effect on the user.

If *.img-fluid* is used, there are other options available. The class *.center-block* centers the image in the container.

SVG and IE 8 to 10

In Internet Explorer 8 to 10 SVG (scalable vector graphics) images are scaled unfavorably with *.img-fluid*. To solve this, you should add this style rule *width: 100%/9* to each image. Bootstrap does not do this automatically because it has disadvantages for other formats.

```
1   <img src="..." class="img-fluid" alt="Responsive image">
```

The class *.img-responsive* from Bootstrap 3 is no longer available. It was replaced by *.img-fluid*.

It is easy to put images in special forms when using following class (see Listing 6-3 Figure 6-6):

- *.img-thumbnail*: Thumbnails

Listing 6-3. Effects on pictures (Image_Effects.html)

```
1   <img src="..." alt="..." class="img-thumbnail">
```

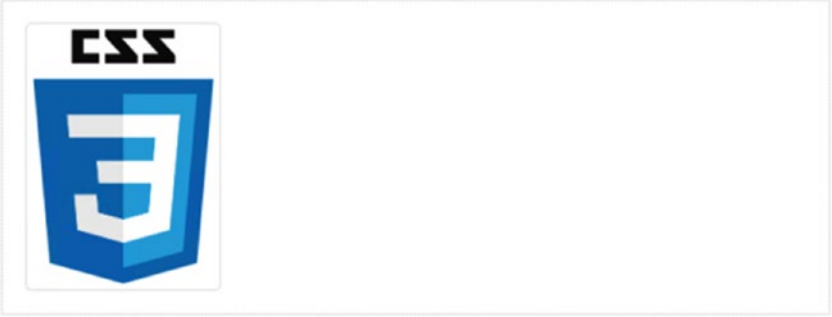


Figure 6-6. *Effects on pictures*

Thumbnails

Thumbnails are usually displayed in their own grid to accommodate as many of them on the page as you wish.

Thumbnails in Bootstrap 4

Cards replace the previous panels, wells, and thumbnail components.

For more information about the display of images, see section on Cards.

Embedded Sources

Embedded sources for video, audio, or external HTML pages are based on `<iframe>-`, `<embed>-`, `<video>-`, and `<audio>` elements.

The commonly used attributions, such as `frameborder="0"` for frames, are supplied automatically. The class *embed-responsive* is used with an additional class that specifies the correct display and format:

- *.embed-responsive-16by9*: Format 16:9
- *.embed-responsive-4by3*: Format 4:3

```
<!-- 16:9 aspect ratio -->
<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>

<!-- 4:3 aspect ratio -->
<div class="embed-responsive embed-responsive-4by3">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>
```

Colors and Backgrounds

Colors and backgrounds are global, so they provide variations of elements that have been provided.

Text Color

Text colors are again some classes that have a slightly semantic meaning (see Listing 6-4 and Figure 6-7). The primary classes are as follows:

- *text-muted*: Suppressed, bright, and gray
- *text-primary*: Primary, main action, or primary statement, blue, important
- *text-success*: Success, green, positive, or success
- *text-info*: Information, azure, highlight, attention needed
- *text-warning*: Warning, orange, action is consequential or message is critical
- *text-danger*: Danger, red, errors or irreversible, serious warning

Listing 6-4. Colors for text (Text_Colors.html)

```
<p class="text-muted">...</p>
<p class="text-primary">...</p>
<p class="text-success">...</p>
<p class="text-info">...</p>
<p class="text-warning">...</p>
<p class="text-danger">...</p>
```

Figure 6-7. *Text colors*

When allocated to an item, <p> is here just another example that does not have the desired effect, the use of another element may be helpful. Again it is true that the weak semantic meaning is not accessible and screen readers do not reflect this information. Additional details that are hidden with *.sr-only* again are the right solution.

Background Color

Available background colors are similar classes that have a slightly semantic meaning (see Figure 6-8). The primary criteria are as follows:

- *Primary*: Primary, main action or primary statement, blue
- *Success*: Success, green, positive or success
- *Info*: Information, violet, highlight, attention needed
- *Warning*: Warning, orange, action is consequential or message is critical
- *Danger*: Danger, red, errors or irreversible, serious warning

```
<p class="bg-faded">...</p>
<p class="bg-primary">...</p>
<p class="bg-success">...</p>
<p class="bg-info">...</p>
<p class="bg-warning">...</p>
<p class="bg-danger">...</p>
```

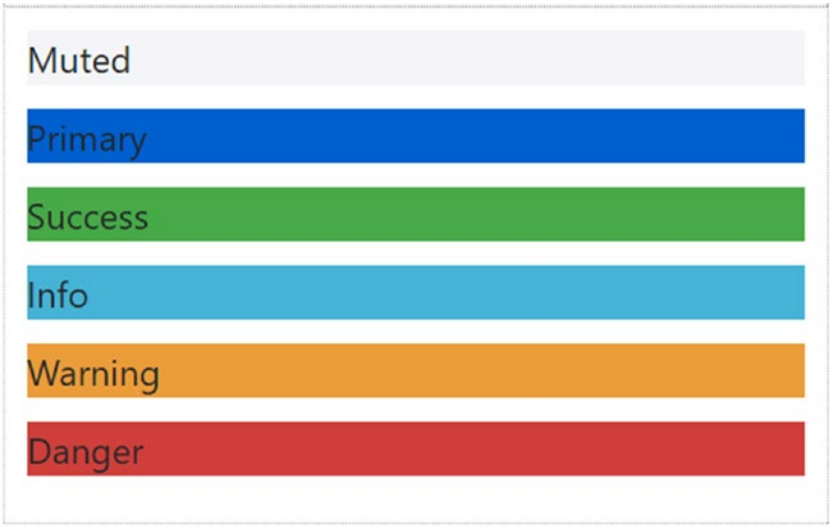


Figure 6-8. *Background colors*

When allocating to an item, `<p>` here is again an example that is does not have the desired effect, so the use of another `<div>` element can be helpful. The weak semantic meaning is not appropriate for accessibility and screen readers do not reflect this information. Again, additional details that are hidden with *.sr-only*, are the right solution.

Alignment of Elements in Flow

The natural flow (sequence of elements) on a page can be changed by the classes *.float-xx-left* and *.float-xx-right*. The element is always pulled to the right or left edge, even if this means the natural order is changed. Internally it is based on the style rule `float`. The placeholder "xx" must be replaced with a breakpoint hint, such as "sm", "md", or "lg". The element will be arranged if the breakpoint applies.

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
```

This should not be done in navigation bars, where there are the specific classes *.navbar-left* or *.navbar-right*.

Centering

With *text-xx-center* an element in the container is centered:

```
<div class="text-md-center">...</div>
```

 Never use the outdated HTML element `<center>`!

Break: Clearfix

To interrupt the flow use *.clearfix* on the parent element.

```
<div class="clearfix">...</div>
```

To break the flow of the HTML elements it is pretty easy with *display:block*. In fact this belongs to the eternal clearfix history and is one of the most complex hacks in the CSS world. So what is behind it? First the definition in Bootstrap's SASS-source code:

```
@mixin clearfix() {
  &::after {
    content: "";
    display: table;
    clear: both;
  }
}
```

This code produces pseudo-elements and sets the display mode to *table*. This creates an anonymous table cell in block format. The `:before` rule prevents the upper edge from falling together with the previous element (collapsed). This prevents a "stranger" distance that is disturbing. The `:after` rule generates the actual movement of the item onto the next line.

Show and Hide Content

With the class *.text-hide* contents are explicitly hidden. The application succeeds only for block elements, such as `<div>`.

```
1   <div class="text-hide">...</div>
```

Showing or hiding also can be made dependent on the device width. For this purpose, the following classes are used ("<>" are placeholders):

- *.hidden-xs*: Hide only with "xs"
- *.hidden-sm*: Hide only with "sm"
- *.hidden-md*: Hide only with "md"
- *.hidden-lg*: Hide only with "lg"
- *.hidden-xl*: Hide only with "xl"

The classes are then available in three variants ("<>" are placeholders):

- *.visible-<>-block: display: block;*
- *.visible-<>-inline: display: inline;*
- *.visible-<>-inline-block: display: inline-block;*

Comparable to the code for screen content can be shown or hidden when printing, too:

- *.visible-print-block*
- *.visible-print-inline*
- *.visible-print-inline-block*
- *.hidden-print*

The final document class looks, for example, like this: *.visible-xs-block*. The area that is designated will be displayed only if the screen is less than 768 px wide.

CHAPTER 7

Components

Components are the building blocks taken from the Bootstrap library and provide certain functionalities, which go beyond the mere use of CSS. Components are made from CSS with a combination of HTML, JavaScript, and font libraries.

Drop-Down Menus

Drop downs—context or pop-up menus—are an integral part of many forms. The interaction is complex and partly based on JavaScript. It must not be activated separately, the `data-`attributes attach here, namely `data-toggle="dropdown"`. Even menus benefit from symbols.

In Bootstrap 4 menus are not created with `<ul><li></li></ul>` any longer, but with `<div>` and regular buttons or anchor tags that match *.dropdown-menu* or *.dropdown-item* and information can be added to them (see Listing 7-1 and Figure 7-1).

Listing 7-1. Menu folds downward (Icons_DropDown.html)

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle"
            type="button"
            id="dropdownMenu1"
            data-toggle="dropdown"
            aria-haspopup="true" aria-expanded="true">
    Fold down
    <span class="caret"></span>
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu1">
    <a class="dropdown-item" href="#">Edit</a>
    <a class="dropdown-item" href="#">Delete</a>
    <a class="dropdown-item" href="#">Details</a>
    <a class="dropdown-item" href="#">Lock</a>
  </div>
</div>
```


J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_7

Figure 7-1. *Menu folds downward*

With the class *.dropup* the menu can folds to the top instead (see Listing 7-2 and Figure 7-2).

Listing 7-2. Menu tilted upward (Icons_DropUp.html)

```
<div class="dropup">
  <button class="btn btn-default dropdown-toggle"
            type="button"
            id="dropdownMenu2" data-toggle="dropdown"
            aria-haspopup="true" aria-expanded="false">
     Fold Up
     <span class="caret"></span>
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
     <a class="dropdown-item" href="#">Edit</a>
     <a class="dropdown-item" href="#">Delete</a>
     <a class="dropdown-item" href="#">Details</a>
     <a class="dropdown-item" href="#">Lock</a>
  </div>
</div>
```

Figure 7-2. *Menu tilted upward*

Alignment of the Menu

A normal menu is left-aligned. With *.dropdown-menu-right* it can be changed to be aligned right in the container (see Listing 7-3 and Figure 7-3). Otherwise, the menus will be in the normal flow of the page. It is safer to use this than to use *.pull-right*, which is not officially supported. If an alignment is completed in a parent element that needs to be changed, use *.dropdown-menu-left*.

Listing 7-3. Alignment in the container (Icons_DropDown_Right.html)

```
<div class="dropdown-menu dropdown-menu-right"
    aria-labelledby="dLabel">
     ...
</div>
```

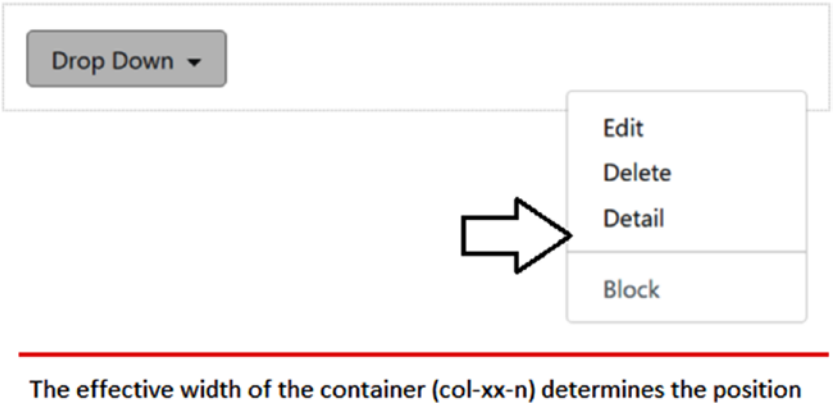


Figure 7-3. *Alignment in the container*

Decorative Elements

Decorative elements complement the menus. They should be used primarily to improve the readability of long menus and not just as decoration.

Subheads in Menus

Inactive subheadings can be added for decoration. A normal heading element of suitable size can be used; `<h6>` usually works very well (see Listing 7-4 and Figure 7-4).

Listing 7-4. Subtitle (Icons_Dropdown_Header.html)

```
<div class="dropdown-menu"
    aria-labelledby="dropdownMenu3">
  ...
```

```
4     <h6 class="dropdown-header">Important</h6>
5     ...
6   </div>
```

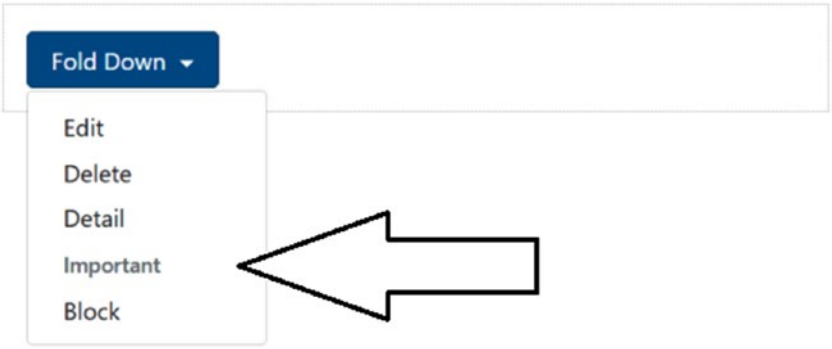


Figure 7-4. *Subtitle*

Divider Line

Separator lines can be used to optically separate a series of links (see Listing 7-5 and Figure 7-5). This makes long menus easier to read. You can achieve this effect with the class *.dropdown-divider*. As an alternative, you can also use `<hr />`.

Listing 7-5. Divider line (Icons_DropDown_Sep.html)

```
1   <div class="dropdown-menu"
2       aria-labelledby="dropdownMenuDivider">
3     ...
4     <div role="separator" class="dropdown-divider" />
5     ...
6   </div>
```

Figure 7-5. *Divider line*

Deactivated Links

You can disable `<li>` a menu item with *.disabled*. Listing 7-6 and Figure 7-6 shows the method and the results.

Listing 7-6. Deactivated entry (Icons_DropDown_Dis.html, partially)

```
<div class="dropdown-menu" aria-labelledby="dropdownMenu1">
  <a class="dropdown-item" href="#">Edit</a>
  <a class="dropdown-item" href="#">Delete</a>
  <a class="dropdown-item" href="#">Details</a>
  <div role="separator" class="dropdown-divider"></div>
  <a class="dropdown-item disabled" href="#">Block</a>
</div>
```

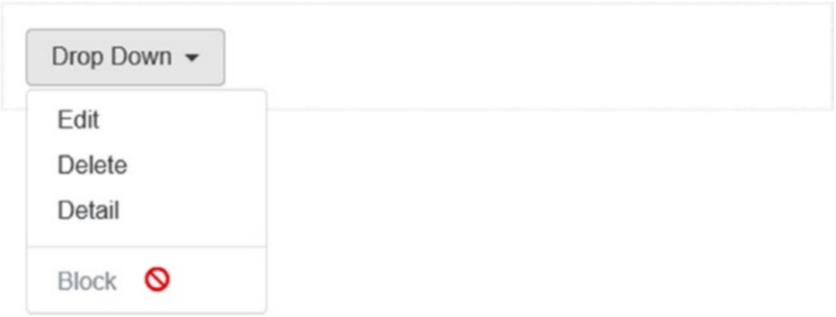


Figure 7-6. *Deactivated entry*

Toolbars

Toolbars are technically groups of buttons. The buttons are horizontally next to each other. Embedded checkboxes and radio buttons require JavaScript to work correctly. This does not have to be activated manually (see Listing 7-7 and Figure 7-7).

Tooltips and superimposed dialogs (pop-overs) need special settings—`container: 'body'` in the options section of JavaScript—to specify where the item should dock at the DOM to trigger the rendering process. *body* is most the suitable. However, each container element can be addressed.

```
$("#testsettracesBtn1").popover({
    container: 'body'
});
```

The semantic meaning is supported with the `role`-attribute, either `role="group"` or `role="toolbar"`. As discussed in previous chapters, the support screen readers is useful and is achieved with `aria-label` or `aria-labelledby`.

```
<div class="btn-group" role="group" aria-label="...">
  <button type="button" class="btn btn-default">Left</button>
  <button type="button" class="btn btn-default">Mid</button>
  <button type="button" class="btn btn-default">Right</button>
</div>
```

Complex toolbars be built with a combination of `<div class="btn-group">` within `<div class="btn-toolbar">`.

```
<div class="btn-toolbar" role="toolbar" aria-label="">
  <div class="btn-group" role="group" aria-label="">...</div>
  <div class="btn-group" role="group" aria-label="">...</div>
  <div class="btn-group" role="group" aria-label="">...</div>
</div>
```

With the *.btn-group*-group-classes a complete set of buttons can be made with resize. The usual suffixes "xs," "sm," and "lg" are available—the default value is "md."

```
 <div class="btn-group btn-group-lg"
      role="group"
      aria-label="...">...</div>
 <div class="btn-group"
      role="group"
      aria-label="...">
  ...
 </div>
 <div class="btn-group btn-group-sm"
      role="group"
      aria-label="...">
 ...
 </div>
 <div class="btn-group btn-group-xs"
      role="group"
      aria-label="...">
 ...
 </div>
```

To create buttons with combined pop-up menus, you can nest the *.btn-group*-classes.

Listing 7-7. Toolbar with buttons (Toolbar.html)

```
 <div class="btn-toolbar" role="toolbar" aria-label="Toolbar">
   <div class="btn-group" role="group">
     <button type="button"
             class="btn btn-secondary dropdown-toggle"
             data-toggle="dropdown"
             aria-haspopup="true"
```

```
            aria-expanded="false">
        Sort
        <span class="caret"></span>
      </button>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Descending</a>
        <a class="dropdown-item" href="#">Ascending</a>
      </div>
    </div>
    <div class="btn-group" role="group">
      <button class="btn btn-danger">Delete</button>
    </div>
    <div class="btn-group" role="group">
      <button class="btn btn-primary">Detail</button>
    </div>
  </div>
```

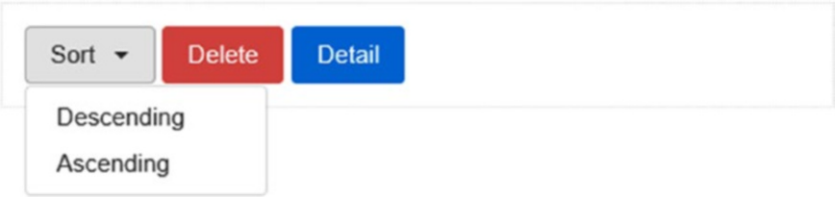


Figure 7-7. *Toolbar with buttons*

Vertical Alignment

Until now the assumption was that toolbars always ran horizontally. If you place them on the edge, it is better to place them vertically. Listing 7-8 shows how to do it. The library *Font Awesome* is used for the symbols here. The corresponding CSS also must be involved.

Listing 7-8. Vertical toolbar with buttons (Toolbar_Vertical.html)

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar">
  <div class="btn-group-vertical" role="group">
    <button class="btn btn-info">
      <span class="fa fa-plus"></span>
    </button>
    <button class="btn btn-info">
      <span class="fa fa-minus"></span>
    </button>
    <button class="btn btn-danger">Delete</button>
    <button class="btn btn-primary">Detail</button>
  </div>
</div>
```

Shared buttons (SplitButton) or pop-up menus cannot be used—the menus are misplaced. Figure 7-8 shows a vertical toolbar with buttons.

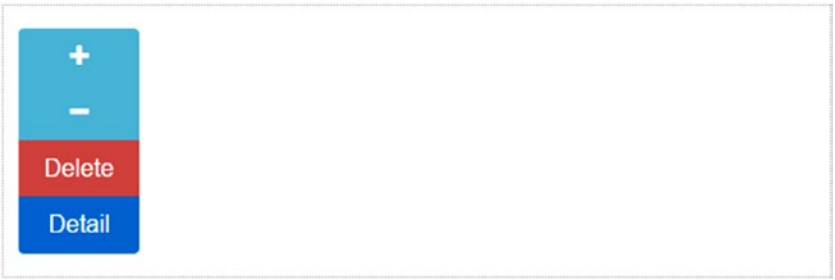


Figure 7-8. *Vertical toolbar with buttons*

General Options

You can apply some effects overall to the toolbars. The previously possible extension that applied to the entire width (*.justified**-Options) no longer exists.

To navigate within a page, the `<a>`-Tag is used but it is formatted as a button. It is important here to maintain a semantic reference and use the attribute `role="button"`.

```
<div class="btn-group"
     role="group" aria-label="...">
  <a class="btn btn-primary" role="button" href="#">Return</a>
  <a class="btn btn-primary" role="button" href="#">2</a>
  <a class="btn btn-primary" role="button" href="#">End</a>
</div>
```

Menu Button

Each button can trigger pop-up menus. The button can appear in one or two parts.

Simple Button with Menu

The arrangement must be in a *.btn-group*-class. The function requires JavaScript. If you do not use with the entire library, the *DropDown-Plugin* is required (see Listing 7-9 and Figure 7-9).

Listing 7-9. Toolbar with drop-down menu (Toolbar_DropDown.html)

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar">
  <div class="btn-group" role="group">
    <button type="button" class="btn btn-secondary">File</button>
  </div>
```

```
5     <div class="btn-group" role="group">
6       <button type="button" class="btn btn-secondary dropdown-toggle"
7               data-toggle="dropdown"
8               aria-haspopup="true" aria-expanded="false">
9         Action <span class="caret"></span>
10      </button>
11      <div class="dropdown-menu">
12        <a class="dropdown-item" href="#">Show</a>
13        <a class="dropdown-item" href="#">Moving</a>
14        <a class="dropdown-item" href="#">Copy</a>
15        <div role="separator" class="dropdown-divider"></div>
16        <a class="dropdown-item" href="#">Delete</a>
17      </div>
18    </div>
19  </div>
```

Figure 7-9. *Toolbar with drop-down menu*

The trigger for JavaScript is the attribute `data-toggle="dropdown"` (line 4). The black triangle image on the menu option is produced with the class *.caret* (line 6).

Split-Button Menu

A split button (SplitButton) is created simply with a further `<button>`- element. Usually, such a button triggers a default option from the list of options offered from a menu (see Listing 7-10 and Figure 7-10).

Listing 7-10. Toolbar with split button (Toolbar_Split.html)

```
1   <div class="btn-toolbar" role="toolbar" aria-label="Toolbar">
2     <div class="btn-group">
3       <button type="button" class="btn btn-danger">Standard</button>
4       <button type="button" class="btn btn-danger dropdown-toggle"
5               data-toggle="dropdown"
6               aria-haspopup="true" aria-expanded="false">
```

```
        <span class="caret"></span>
        <span class="sr-only">Open Menu</span>
      </button>
      <div class="dropdown-menu">
         <a class="dropdown-item" href="#">Standard</a>
         <a class="dropdown-item" href="#">More</a>
         <a class="dropdown-item" href="#">Even more</a>
         <div role="separator" class="dropdown-divider"></div>
         <a class="dropdown-item" href="#">Others</a>
      </div>
    </div>
  </div>
```

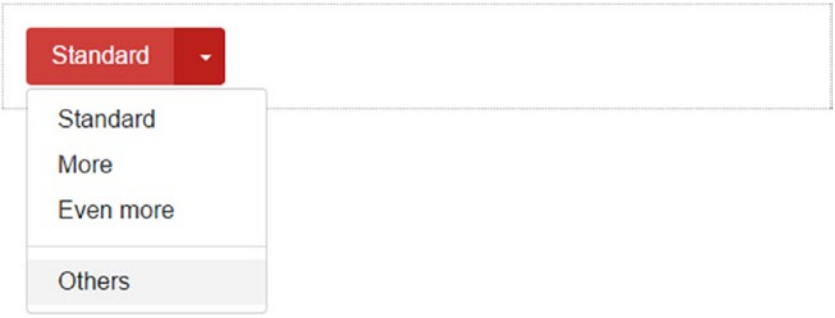


Figure 7-10. *Toolbar with split button*

Sizes of Menu Buttons

Basically, you can use any size of menu buttons (see Listing 7-11 and Figure 7-11). The menu adapts to the styles *btn-lg* and *btn-sm*. However, *btn-md* is the default and does not need to be specified.

Listing 7-11. Menu sizes (Toolbar_Size.html)

```
<div class="btn-group">
  <button class="btn btn-secondary btn-lg dropdown-toggle"
          type="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
    Huge <span class="caret"></span>
  </button>
  <div class="dropdown-menu">
     ...
   </div>
</div>

```

```
<div class="btn-group">
  <button class="btn btn-secondary dropdown-toggle"
          type="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
    Normal <span class="caret"></span>
  </button>
  <div class="dropdown-menu">
    ...
  </div>
</div>

<div class="btn-group">
  <button class="btn btn-secondary btn-sm dropdown-toggle"
          type="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
    Small <span class="caret"></span>
  </button>
  <div class="dropdown-menu">
     ...
  </div>
</div>
```

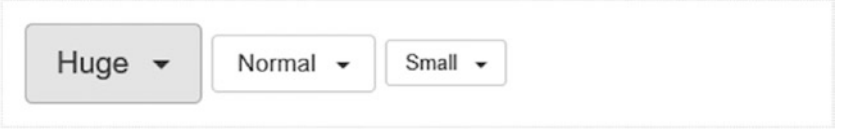


Figure 7-11. *Menu sizes*

Note that the details have been changed as compared to those of Bootstrap 3. The style *.btn-xs* is no longer supported. The smallest form is "sm."

Special Menu Options

With the help of the class *.dropup* the menu can be made visible.

```
<div class="btn-group dropup">
 <button type="button" class="btn btn-secondary">
   Folding
 </button>
 <button type="button"
         class="btn btn-default dropdown-toggle"
         data-toggle="dropdown"
         aria-haspopup="true" aria-expanded="false">
   <span class="caret"></span>
```

```
      <span class="sr-only">open menu</span>
    </button>
    <div class="dropdown-menu">
      <!-- Here are menu items -->
    </div>
  </div>
```

Navigation

The navigation elements are all initiated by a common base class *.nav*. When navigating by tabulators, JavaScript is required. To support the accessibility attribute `role="navigation"` should be used in the logic overlying containers. This gives the navigation the necessary semantic meaning. The important thing is not to do so on the `<ul>`- element, but on the surrounding `<nav>` or `<div>` element.

Basically distinction should be made in navigation between content navigation and navigation action. Content can be accessed via links, buttons, or tabs. However navigation action takes place over buttons, menus, toolbars, menu strips, and so forth. Forms are among the content. Only the submit button, for example, is an action.

Tabs

Tabs are an ideal way to logically separate multiple forms or compartmentalize a large form into manageable areas (see Listing 7-12 and Figure 7-12).

Listing 7-12. Tabs (Nav_tab.html)

```
<ul class="nav nav-tabs">
  <li class="nav-item" role="presentation">
    <a data-toggle="tab" class="nav-link active" href="#">
      Start
    </a>
  </li>
  <li class="nav-item" role="presentation">
    <a data-toggle="tab" class="nav-link" href="#">
      Profile
    </a>
  </li>
  <li class="nav-item" role="presentation">
    <a data-toggle="tab" class="nav-link" href="#">
      News
    </a>
  </li>
</ul>
```

Note here the positioning of classes *.nav-item* and *.nav-link*.

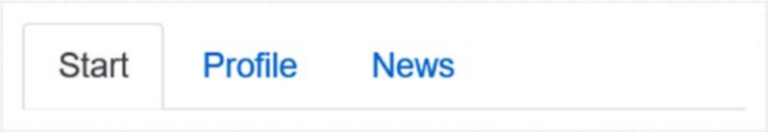


Figure 7-12. *Tabs*

For a complete functioning, JavaScript is required. Bootstrap brings the required code. It is activated via the attribute `data-toggle="tab"`. Tabs that are to be activated with their own code do not carry this attribute.

Navigation Buttons (pills)

For *.nav-pills* and other tabulators with identical structure, to reach a more button-like design use Listing 7-13.

Listing 7-13. Navigation buttons (Nav_Pills.html)

```
<ul class="nav nav-pills">
  <li role="presentation" class="nav-item">
    <a href="#" class="nav-link active" data-toggle="pill">
      Start
    </a>
  </li>
  <li role="presentation" class="nav-item">
    <a class="nav-link" data-toggle="pill" href="#">
      Profile
    </a>
  </li>
  <li role="presentation" class="nav-item">
    <a class="nav-link" data-toggle="pill" href="#">
      News
    </a>
  </li>
</ul>
```

Note in Figure 7-13 the positioning of classes *.nav-item* and *.nav-link*.

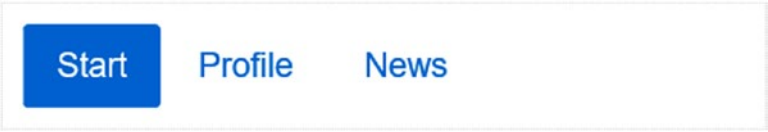


Figure 7-13. *Navigation buttons (pills)*

The “pills” elements mentioned also can be arranged vertically. For this purpose, *.nav-stacked* is used in addition.

```
<ul class="nav nav-pills nav-stacked">
  ...
</ul>
```

JavaScript is required for full functionality. Bootstrap brings the code. It is activated via the attribute `data-toggle="pill"`. Tabs that are to be activated with their own code do not carry this attribute.

Universal Settings

Universal settings affect all navigation elements. This concerns following:

- Orientation
- Deactivation
- Folding menus

Orientation

All items that can be placed horizontally, suffer when viewed if the content differs drastically. It feels more natural when the elements use the space to the utmost. To facilitate the decision, a broad representation engages until 768 px screen width. The elements are distributed uniformly over the entire screen, similar to justification. The effect can be achieved with *.nav-justified*. On wider screens, the elements are placed left-aligned again.

```
<ul class="nav nav-tabs nav-justified">
  ...
</ul>
<ul class="nav nav-pills nav-justified">
  ...
</ul>
```

Deactivation

It may happen that the content in question or part of the form does not need to be reached. In this case the item is disabled (see Listing 7-14 and Figure 7-14). It is then grayed out and does not respond to the mouse or touch actions.

Listing 7-14. Disabled navigation buttons (Nav_Pills_Deactivated.html)

```
<ul class="nav nav-pills">
  ...
  <li role="presentation" class="nav-item">
    <a class="nav-link disabled" href="#">Deactivated Link</a>
  </li>
  ...
 </ul>
```

Figure 7-14. *Disabled navigation buttons*

Expansion through Pop-Up Menus

Tabs can be expanded by folding menus (see Listing 7-15 and Figure 7-15). However, you should certainly think twice about this because it makes using it complicated, and it is not a typical menu position. It may be that the user does not realize this.

Listing 7-15. Navigation buttons with (Nav_Tab_Menu.html)

```
<ul class="nav nav-tabs">
  <li role="presentation" class="nav-item">
    <a class="nav-link active" href="#"
      data-toggle="tab">Files
    </a>
  </li>
  <li role="presentation" class="nav-item dropdown">
    <a class="nav-link dropdown-toggle"
       data-toggle="dropdown"
       href="#" role="button"
       aria-haspopup="true"
       aria-expanded="false">
     Quick Menu <span class="caret"></span>
   </a>
   <div class="dropdown-menu">
     <a class="dropdown-item" href="#">Loading</a>
     <a class="dropdown-item" href="#">Unloading</a>
   </div>
  </li>
  <li role="presentation" class="nav-item">
    <a class="nav-link" href="#" data-toggle="tab">Backup</a>
  </li>
</ul>
```

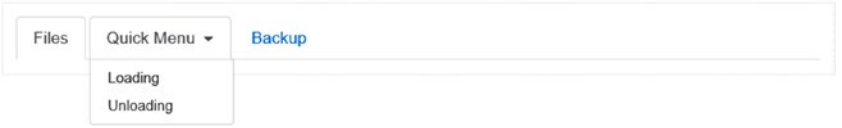


Figure 7-15. *Navigation buttons with menu*

Moreover, the drop-down menu can be used with either `data-toggle="dropdown"` foldable or `data-toggle="tab"`. So it has either the behavior of a tab or a menu but not both. A pull-down menu without a menu function is meaningless. So the only remaining option is to use a part of the menu tabs without the tab function—a rare combination need.

Pills can do this ofcourse as well and behave in the identical way:

```
<ul class="nav nav-pills">
  ...
  <li role="presentation" class="nav-item dropdown">
    <a class="nav-linkdropdown-toggle"
       data-toggle="dropdown" href="#"
       role="button"
       aria-haspopup="true" aria-expanded="false">
      Dropdown <span class="caret"></span>
    </a>
    <div class="dropdown-menu">
      ...
    </div>
  </li>
  ...
</ul>
```

Navigation Bar

The navigation bar (navbar) is a highly responsive element. You usually begin with a page that has complex content, with a menu bar at the top (see Listing 7-16). If the width is too small, the items run out at the sight. This is where Bootstrap completely rebuilds the navigation bar. For this purpose, the elements are now arranged vertically and the user also can decrease the view in the width, so the menu does not cover the entire page.

What cannot be immediately solved is the width of the content. Without intervention the headings on the navigation elements do not fit and will wrap around another line. This enlarges the menu below and will eventually run out of the view. Because of the complexity and the strong engagement in the design of the page, this is a manual rework only and Bootstrap only provides the framework. Possible additional measures are the following.

- Reduce the number of elements—must all elements really be accessible from the first page out?
- Depending on the width of the screen elements, they are hidden dynamically—maybe some functions are not useful for a small screen?
- Adjust the menus targeted to all screens, possibly through their own media spaces.

The collapse of the broad menu to narrow (collapse) requires JavaScript—if not all components are loaded—the collapse plug-in. The switch between narrow and wide defaults to 768 px.

The matching attributes for barrier-free access should be used again, in particular `role="navigation"`.

Listing 7-16. Navigation buttons with menu (Nav_Navbar.html))

```
<nav class="navbar navbar-dark bg-primary">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggler hidden-sm-up"
              data-toggle="collapse"
              data-target="#n1"
              aria-expanded="false">
        <span class="sr-only">Logo</span>

      </button>
      <a class="navbar-brand" href="#">Logo</a>
    </div>
    <div class="collapse navbar-toggleable-xs" id="n1">
      <ul class="nav navbar-nav">
        <li class="nav-item active">
          <a class="nav-link" href="#">
            File
            <span class="sr-only">File</span>
          </a>
        </li>
        <li class="nav-item dropdown">
          <a href="#" class="nav-link dropdown-toggle"
             data-toggle="dropdown"
             role="button"
             aria-haspopup="true" aria-expanded="false">
           Functions <span class="caret"></span>
         </a>
         <div class="dropdown-menu">
           <a class="dropdown-item" href="#">Copy</a>
           <a class="dropdown-item" href="#">Move</a>
           <a class="dropdown-item" href="#">Show</a>
```

```
32              <div role="separator" class="dropdown-divider"></div>
33              <a class="dropdown-item" href="#">Delete</a>
34              <div role="separator" class="dropdown-divider"></div>
35              <a class="dropdown-item" href="#">Rename</a>
36            </div>
37          </li>
38        </ul>
39
40        <form class="form-inline navbar-form navbar-left"
41              role="search">
42          <div class="form-group">
43            <input type="text" class="form-control"
44                   placeholder="Keyword">
45          </div>
46          <button type="submit" class="btn btn-secondary">
47            Search
48          </button>
49        </form>
50      </div>
51    </div>)
52  </nav>
```

The navigation bar) is introduced with a logo or symbol. At this point, text or images may appear (see Figures 7-16 and 7-17).

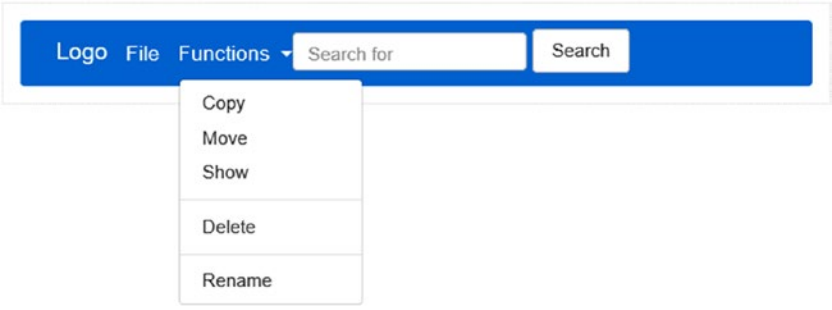


Figure 7-16. *Navigation bar (full width)*

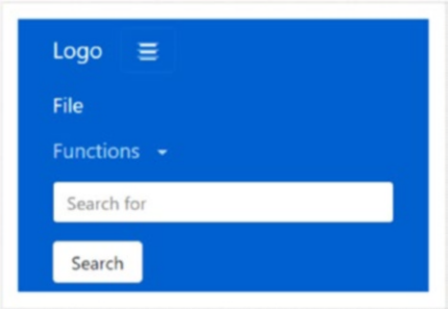


Figure 7-17. *Navigation bar with a small screen*

New for Bootstrap 4 are new color options for the navigation bar.

- *.navbar-dark*
- *.navbar-light*

These classes define the foreground color. For this to work, a matching background color, for example *.bg-primary* must also be selected.

Form Elements

Navigation bars sometimes offer actions that enable them not have their own form. For example, search functions as well checkboxes are commonly found on navigation bars. With *.navbar-form* the matching distances are configured. The form element is aligned within the navigation element.

```
<form class="navbar-form navbar-left" role="search">
  <div class="form-group">
    <input type="text"
           class="form-control"
           placeholder="Keyword" />
  </div>
  <button type="submit" class="btn btn-secondary">
    Search
  </button>
</form>
```

On a small screen form elements rarely can be optimally placed. Take here separate forms in consideration, and try to simplify the arrangement further.

Continue to work for barrier-free access, even if there is no space in the navigation bar. In such cases using *.sr-only* in combination with `aria-label` and `aria-labelledby` or the `title`- attribute is best.

Buttons, Hyperlinks, and Text

Regular buttons are useful to trigger exclusive actions because of their diverse configurability. For this, the class is *.navbar-btn* used in `<button>`-elements where `<a>` and `<input>` are equally usable.

```
<button type="button" class="btn btn-default navbar-btn">
  Log In
</button>
```

With *.navbar-link* hyperlinks can be used that fulfill special tasks, such as responding to registration and available page functions as shown in Listing 7-17 and Figure 7-18.

Listing 7-17. Application link (Nav_NavbarLink.html)

```
<p class="navbar-text navbar-right">
  Logged on as <a href="#" class="navbar-link">
    Joerg Krause</a>
</p>
```

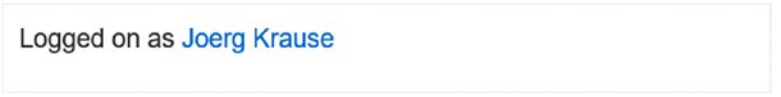


Figure 7-18. *Application link*

The item appears primarily as text. Content without action, plain text, uses the class *.navbar-text*. The element `<p>` supports the takeover of the right colors.

```
<p class="navbar-text">Registered as Joerg Krause</p>
```

The alignment of the elements is accomplished with *.navbar-left* or *.navbar-right*, respectively. Because the elements usually consist of `<li>` the uniform alignment is done on the surrounding `<ul>`.

Position of the Bar

An almost esoteric discussion occasionally arises over the question of whether the bar constantly remains visible. On the one hand, valuable space is permanently occupied, on the other hand, the user should not be forced to scroll a long way. Meanwhile some designers solve this dilemma by two navigation bars. First, the visitor sees a large and elaborately designed one, standing at the top of the page. He then scrolls down and this bar is replaced by a very narrow, fine, simple bar. Thus, the space requirement is minimal and the navigation is still guaranteed. In each case the bar must be fixed at the top:

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    ...
  </div>
</nav>
```

The bar takes up space on the top of the page, though the definition is usually far deeper. This content is not superimposed intentionally, you may need a total of page height to get the upper distance:

```
body { padding-top: 70px; }
```

Standard Navigation Bar

The standard navigation bar (if no theme is used) is 50 px high.

```
<nav class="navbar navbar-default navbar-fixed-bottom">
  <div class="container">
    ...
  </div>
</nav>
```

A static bar is also at the top of this page, but scrolls upward away. With *.navbar-static-top*, this effect is achieved. A correction of the spacing of the content is not necessary.

```
<nav class="navbar navbar-default navbar-static-top">
  <div class="container">
    ...
  </div>
</nav>
```

A reversal of the standard colors is achieved with *.navbar-inverse*. You can combine this effect with all other options.

```
<nav class="navbar navbar-inverse">
  ...
</nav>
```

Breadcrumb Navigation

Complex navigations are often confusing for visitors to the site. The question is always: "Where am I?" The purpose of the navigation path is to enable a path through the navigation hierarchy. This component is called in English "breadcrumb" (bread crumbs,

the meaning of a crumbs trail). The word is also in the English language not a household name (there should be people who have nothing to do with computers here, too). The word "path navigation" is clearer and more appropriate (see Listing 7-18 and Figure 7-19).

Listing 7-18. Breadcrumb navigation (Nav_Breadcrumb.html)

```
<ol class="breadcrumb">
  <li class="breadcrumb-item"><a href="#">Home</a></li>
  <li class="breadcrumb-item"><a href="#">Library</a></li>
  <li class="breadcrumb-item active">Data</li>
</ol>
```

Figure 7-19. *Breadcrumb navigation*

Ideally, the path shows not only where the user actually is, but also allows navigation. Therefore preferable hyperlinks are used. The active element (class *.active*) is deactivated because the user is already at this location.

Page Scrolling

Even page-by-page scrolling belongs to the navigation elements. Usually it occurs in the context of data tables. But even multipage content pages can be browsed.

A typical setup for a few pages or a reasonably constant number of pages looks like this:

```
« 1 2 3 4 5 »
```

Listing 7-19 produces the page scrolling shown in Figure 7-20.

Listing 7-19. Page scrolling (Nav_Pagination.html)

```
<nav>
  <ul class="pagination">
    <li class="page-item">
      <a href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <li class="page-item"><a href="#">1</a></li>
    <li class="page-item"><a href="#">2</a></li>
    <li class="page-item"><a href="#">3</a></li>
    <li class="page-item"><a href="#">4</a></li>
    <li class="page-item"><a href="#">5</a></li>
```

```
      <li class="page-item">
        <a href="#" aria-label="Next">
          <span aria-hidden="true">&raquo;</span>
        </a>
      </li>
    </ul>
  </nav>
```

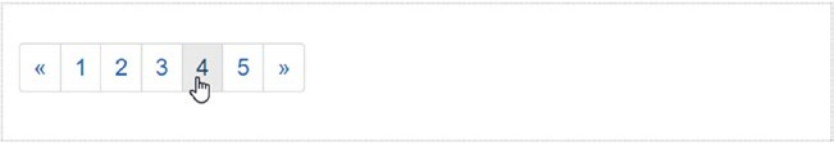


Figure 7-20. *Page scrolling*

If the number of pages is indefinite, sometimes a simple back and forth pattern is recommended.

Back Next

```
<nav>
  <ul class="pager">
    <li><a href="#">Back</a></li>
    <li><a href="#">Next</a></li>
  </ul>
</nav>
```

This is even more striking when the solution uses symbols or is complemented with entities, such as in the following.

<- Older New ->

```
 <nav>
   <ul class="pagination">
   <li class="page-item prev">
      <a href="#">
      <span aria-hidden="true">&larr;</span> Older
      </a>
   </li>
   <li class="page-item next">
       <a href="#">Newer <span aria-hidden="true">&rarr;</span></a>
   </li>
   </ul>
 </nav>
```

If individual pages are inactive, for example, to display the currently selected, either *.disabled* or *.active* is being used.

The following code shows how accomplish this (the output "current" is only for screen readers).

```
<nav>
  <ul class="pagination">
    <li class="page-item disabled">
      <a href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <li class="active">
      <a href="#">1
        <span class="sr-only">(Current)</span>
      </a>
    </li>
    ...
  </ul>
</nav>
```

A nonselectable option can also be achieved for the arrows as in Listing 7-20 and shown in Figure 7-21.

Listing 7-20. Simple leaves (Nav_PaginationON.html)

```
<nav>
  <ul class="pagination">
    <li class="page-item disabled">
      <span>
        <span aria-hidden="true">&laquo;</span>
      </span>
    </li>
    <li class="active">
      <span>1
        <span class="sr-only">(Current)</span>
      </span>
    </li>
    ...
  </ul>
</nav>
```

```
<nav>
  <ul class="pagination">
    <li class="prev disabled"><a href="#">
      <span aria-hidden="true">&larr;</span> Older</a>
    </li>
```

```
      <li class="next">
        <a href="#">Newer
          <span aria-hidden="true">&rarr;</span>
        </a>
      </li>
    </ul>
  </nav>
```

Figure 7-21. *Navigation arrow buttons with partial deactivation*

Size

The default size can be enlarged (*.pagination-lg*) or shrinked (*.pagination-sm*) in one step.

```
<nav><ul class="pagination pagination-lg">...</ul></nav>
<nav><ul class="pagination">...</ul></nav>
<nav><ul class="pagination pagination-sm">...</ul></nav>
```

Tags

Tags are markings that highlight the text.

```
<h3>Example <span class="tag tag-default">New</span></h3>
```

The appearance can be adjusted in six stages as shown in Listing 7-21 and Figure 7-22 (with the usual semantic alignment—the colors are not a design element).

- *tag-default*: Standard, gray
- *tag-primary*: Primary statement, blue
- *tag-success*: Success message, green
- *tag-info*: Information, azure
- *tag-warning*: Warning, orange
- *tag-danger*: Danger or fault, red

Listing 7-21. Labels (Label.html)

```
<span class="tag tag-default">Standard</span>
<span class="tag tag-primary">Primary</span>
<span class="tag tag-success">Success</span>
<span class="tag tag-info">Info</span>
<span class="tag tag-warning">Warning</span>
<span class="tag tag-danger">Danger</span>
```

Figure 7-22. *Labels*

Identification Labels

Identification labels (tag-pills) are less inflationary when used than regular labels, hence they are more exclusive and striking. Their corners are more rounded and complement the overall picture. When placed on buttons the backgrounds are transparent, so the rounded corners are not visible (see Listing 7-22 and Figure 7-23).

New in Bootstrap 4

This presentation form is new to Bootstrap 4. It eliminates the class *badge*. Badges have a similar graphic concept as labels and appear very similar. The amendment simplifies the handling of markings.

Listing 7-22. Tag-Pills (LabelPillsAnchor.html)

```
<a href="#">Inbox
  <span class="label label-pill bg-danger">42</span>\
</a>

<button class="btn btn-primary" type="button">
  <span class="tag tag-pill">4</span> Messages
</button>
```

Figure 7-23. *Tag-Pills in action*

The application is found mostly in the context of frequently changing data. The user's view is selectively captured. Because possible actions are often hidden behind the presentation of data, it makes sense to embed the badges in the navigation elements. Listing 7-23 shows the number of new messages and the user can then access the new page by clicking on the item (see Figure 7-24).

Listing 7-23. Bagdes in navigation (LabelPills.html)

```
<ul class="nav nav-pills" role="tablist">
  <li role="presentation" class="active">
    <a href="#" data-toggle="pill"
                class="btn btn-sm btn-primary">
      Start
      <span class="tag tag-pill">42</span>
    </a>
  </li>
  <li role="presentation">
    <a href="#" data-toggle="pill"
                class="btn btn-sm btn-danger">Profile</a>
  </li>
  <li role="presentation">
    <a href="#" data-toggle="pill"
                class="btn btn-sm btn-info">
      <span class="tag tag-pill">3</span> Messages
    </a>
  </li>
</ul>
```

Figure 7-24. *Labels in the navigation*

 Labels, as the content is variable, are sometimes empty. In such cases, they become invisible.

Big Screen (jumbotron)

The jumbotron is a big, bold surface with rounded corners, usually only on the home page. It contains a slogan, sayings, or product that is presented exclusively as demonstrated in Listing 7-24 and Figure 7-25.

Listing 7-24. Introduction of a page (Jumbotron.html)

```
<div class="jumbotron">
  <h1>Hello Bootstrap!</h1>
  <p>...</p>
  <p>
    <a class="btn btn-primary btn-lg"
       href="#" role="button">
      Read more...
    </a>
  </p>
</div>
```

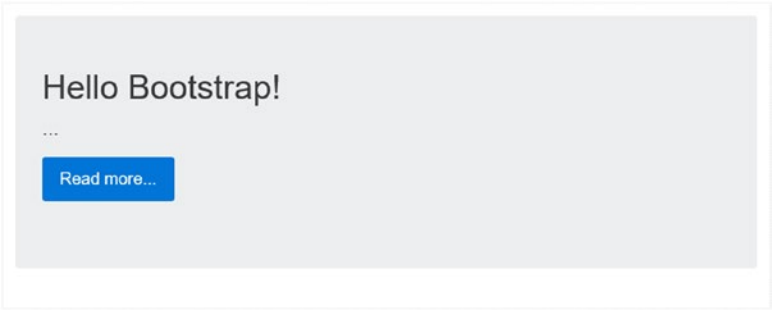


Figure 7-25. *Introduction on a page*

Normally, the item is in the container, and therefore it has a lateral distance. If that is not the case, the container also can be placed inside.

The rounded corners will also disappear as shown in Figure 7-26:

```
<div class="jumbotron">
  <div class="container">
    ...
  </div>
</div>
```

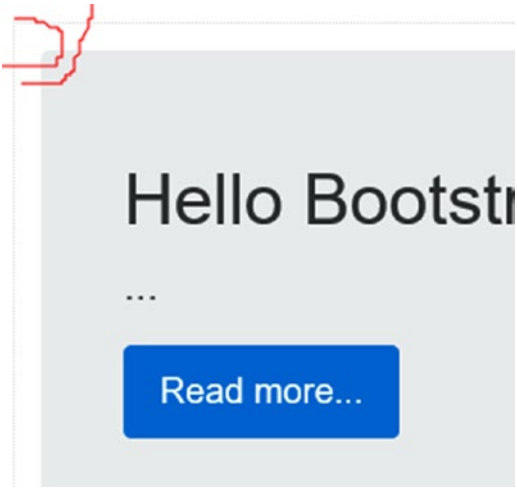


Figure 7-26. *Variant of jumbotron*

Page Headers

Page headers separate sections and give a more comprehensive view than typographic headings alone (see Listing 7-25 and Figure 7-27).

Listing 7-25. Headings with a dividing line (PageHeader.html)

```
<div class="page-header">
  <h1>Example page header
      <small>Subtext for header</small>
  </h1>
</div>
```

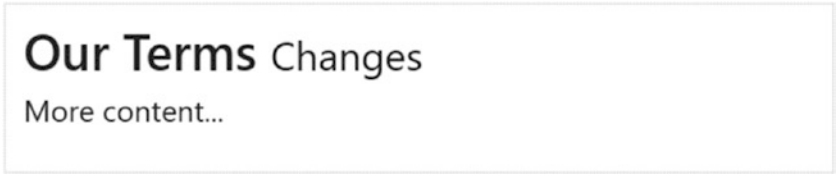


Figure 7-27. *Headings with a dividing line*

Messages

Messages provide information in certain contexts. There are four variants.

- *alert-success*: Success message, green
- *alert-info*: Information, azure
- *alert-warning*: Warning, orange
- *alert-danger*: Error, red

Note that the primary and standard variants are missing. These are messages without meaning because messages should explicitly emphasize a particular state as shown in Listing 7-26 and Figure 7-28.

Listing 7-26. Output messages (Alerts.html)

```
1   <div class="alert alert-success" role="alert">...</div>
2   <div class="alert alert-info" role="alert">...</div>
3   <div class="alert alert-warning" role="alert">...</div>
4   <div class="alert alert-danger" role="alert">...</div>
```

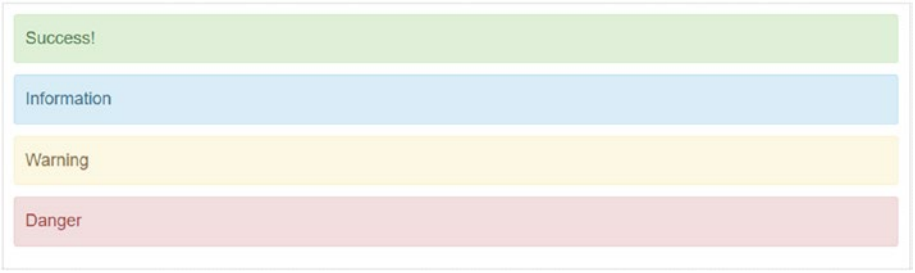


Figure 7-28. *Output messages*

Because messages often show up without being asked, they should be able to be removed. That is the purpose of the class *.alert-dismissible* along with a button or an icon for closing. JavaScript is required to do this. The script responds to the attribute `data-dismiss="alert"` (see Listing 7-27 and Figure 7-29).

Listing 7-27. Messages that are closable (Alerts_Dismiss.html)

```
<div class="alert alert-warning alert-dismissible"
     role="alert">
  <button type="button" class="close"
          data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
  <strong>Attention!</strong> Please check the data.
</div>
```

Figure 7-29. *Messages that are closable*

Messages that direct the user to further information can be very large. If you use hyperlinks for this purpose, this is accomplished with the class *.alert-link*, which is shown in Listing 7-28 and Figure 7-30.

Listing 7-28. Messages with further link (Alerts_Link.html)

```
<div class="alert alert-success" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-info" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-warning" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-danger" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
```

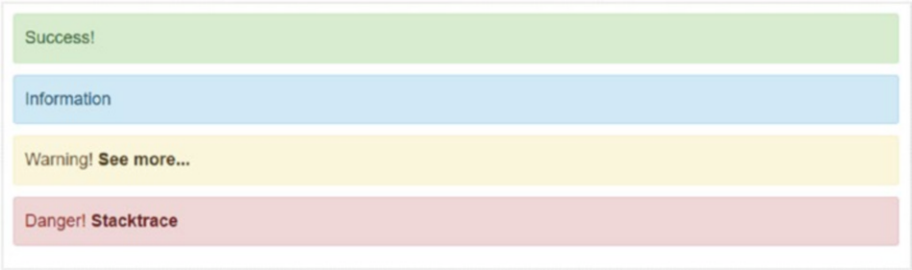


Figure 7-30. *Messages with further link*

Progress Bar

Longer-term actions benefit from the progress bar. The implementation used by Bootstrap is purely a design element that needs to be supplemented for a meaningful indicator with JavaScript. In addition, currently no browsers fully support the required CSS3. Bootstrap 4 uses the native HTML element `<progress>` as shown in Listing 7-29.

Listing 7-29. Passive progress bar (Progress.html)

```
1   <progress class="progress"
2             aria-valuenow="60"
3             aria-valuemin="0"
4             aria-valuemax="100"
5             value="60%"
6             max="100">
7   </progress>
```

Note that the actual width of the active zone is generated with a `width` style (line 6). This value must be manipulated to move the position of the progress bars (see Figure 7-31).

Figure 7-31. *Passive progress bar*

In Bootstrap 3 the label was placed in a `<div>`- element, while the `<progress>`-element was not supported. This possibility no longer exists. In this respect, the use is true standards compliant, but also less flexible.

A progress bar can show not only a position, but also a context. The following (as shown in Listing 7-30 and Figure 7-32) are the semantically classes available:

- *progress-bar-success*: Success, green
- *progress-bar-info*: Information, azure
- *progress-bar-warning*: Warning, orange
- *progress-bar-danger*: Danger or critical, red

Listing 7-30. Semantic progress bar (Progress_Semantic.html)

```
<progress class="progress progress-success"
          role="progressbar"
          aria-valuenow="40"
          aria-valuemin="0"
          aria-valuemax="100"
          value="40" max="100">
  <span class="sr-only">40% Complete (success)</span>
</progress>
<progress class="progress progress-info"
          role="progressbar"
          aria-valuenow="20"
          aria-valuemin="0"
          aria-valuemax="100"
          value="20" max="100">
  <span class="sr-only">20% Complete</span>
</progress>
<progress class="progress progress-warning"
          role="progressbar"
          aria-valuenow="60"
          aria-valuemin="0"
          aria-valuemax="100"
          value="60" max="100">
  <span class="sr-only">60% Complete (warning)</span>
</progress>
<progress class="progress progress-danger"
          role="progressbar"
          aria-valuenow="80"
          aria-valuemin="0"
          aria-valuemax="100"
          value="80" max="100">
  <span class="sr-only">80% Complete (danger)</span>
</progress>
```

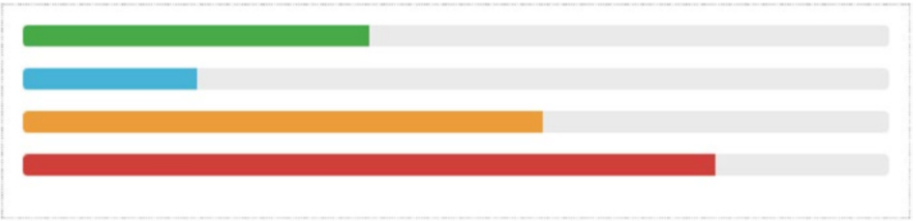


Figure 7-32. *Semantic progress bar*

An obliquely running strip is a nice effect, which emphasizes the animation effect (see Listing 7-31 and Figure 7-33). Its worth in very slow running progress bars and unspecific progress bars as well.

Listing 7-31. Semantic with stripes (Progress_Semantic_Striped.html)

```
<progress class="progress progress-success progress-striped"
          role="progressbar"
          aria-valuenow="40"
          aria-valuemin="0"
          aria-valuemax="100"
          value="40" max="100">
  <span class="sr-only">40% Complete (success)</span>
</progress>
<progress class="progress progress-info progress-striped"
          role="progressbar"
          aria-valuenow="20"
          aria-valuemin="0"
          aria-valuemax="100"
          value="20" max="100">
  <span class="sr-only">20% Complete</span>
</progress>
<progress class="progress progress-warning progress-striped"
          role="progressbar"
          aria-valuenow="60"
          aria-valuemin="0"
          aria-valuemax="100"
          value="60" max="100">
  <span class="sr-only">60% Complete (warning)</span>
</progress>
<progress class="progress progress-danger progress-striped"
          role="progressbar"
          aria-valuenow="80"
          aria-valuemin="0"
          aria-valuemax="100"
          value="80" max="100">
  <span class="sr-only">80% Complete (danger)</span>
</progress>
```

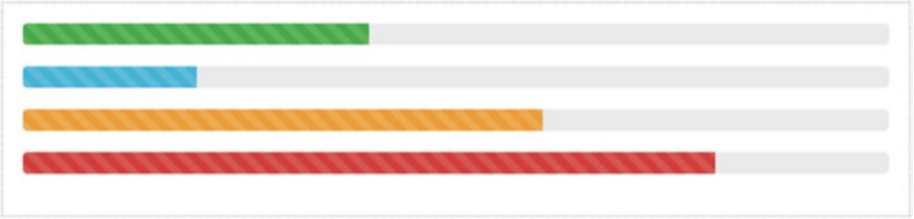


Figure 7-33. *Semantic progress bar with stripes*

More animation can be reached with *.active*, as shown in Figure 7-34.

```
<progress class="progress progress-striped active"
          role="progressbar"
          aria-valuenow="80"
          aria-valuemin="0"
          aria-valuemax="100"
          value="80" max="100">
  <span class="sr-only">80% Complete (danger)</span>
</progress>
```

Figure 7-34. *Interactive progress bar*

The behavior is generated by this small script using jQuery:

```
 $(function() {
   var options = $('[name="inlineRadioOptions"]').length;
   $(':checkbox').on('click', function () {
   var active = $('[name="inlineRadioOptions"]:checked').length;
   var value = Math.round(active / options * 100);
     $('#progress-bar')
     .attr('value', value)
     .attr('aria-valuenow', value)
   ;
   });
 });
```

The script determines the number of options (line 2). Then each time you click the number of active members is determined by a checkbox (line 4). The display and the width is controlled by the then calculated percentage. The test in line 9 hides the number 0, so that the display without bars is empty.

 Multiple parts in a bar feature complex sets of values. This somewhat exotic way was possible with Bootstrap 3. However, it is no longer offered with the `<progress>`-Element in Bootstrap 4. Either continue the procedure of Bootstrap 3 by manipulating the width of `<div>` elements and using the class *.progress-bar* or better, dispense with on nonstandard designs.

Media

Media are videos, audio files, or similar sensitive information.

```
<div class="media">
  <div class="media-left">
    <a href="#">
      <img class="media-object" src="..." alt="...">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading">Media heading</h4>
    ...
  </div>
</div>
```

The classes *.media-left* and *.media-right* align the element on the page. They must always be placed in a container with *.media-body*.

Orientation

Both images and media elements line up generally.

```
<div class="media">
  <div class="media-left media-middle">
    <a href="#">
      <img class="media-object" src="..." alt="...">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading">Middle aligned media</h4>
    ...
  </div>
</div>
```

Media Lists

To organize media in lists, some additional HTML is required.

```
<ul class="media-list">
  <li class="media">
    <div class="media-left">
      <a href="#">
        <img class="media-object" src="..." alt="...">
      </a>
    </div>
    <div class="media-body">
      <h4 class="media-heading">Media heading</h4>
      ...
    </div>
  </li>
</ul>
```

Common Lists

Lists are grouped collections of elements (see Listing 7-32 and Figure 7-35). The elements are disorganized, without a guide mark and are visually compacted.

Listing 7-32. Lists for grouping (ListGroups.html)

```
<ul class="list-group">
  <li class="list-group-item">These options</li>
  <li class="list-group-item">here</li>
  <li class="list-group-item">for all users</li>
  <li class="list-group-item">this application</li>
  <li class="list-group-item">generally available</li>
</ul>
```

Figure 7-35. *Lists for grouping*

Tags in Lists

Tags can be placed in the text of the list items in lists at any point (see Listing 7-33 and Figure 7-36).

Listing 7-33. Lists with tags (ListTags.html)

```
<ul class="list-group">
  <li class="list-group-item">
    <span class="tag tag-danger">14</span>
    Options
  </li>
  <li class="list-group-item">
    <span class="tag tag-info">7</span>
    Application
  </li>
</ul>
```

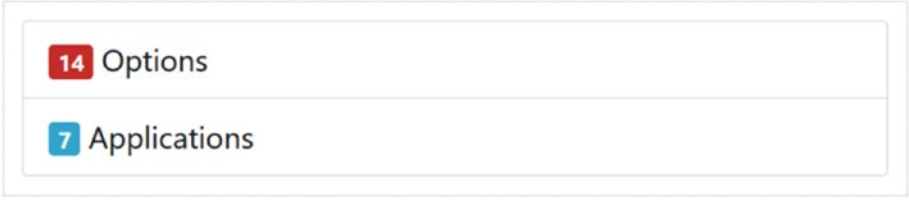


Figure 7-36. *Lists with tags*

Links in Lists

You also can use hyperlinks in lists (see Listing 7-34). This practical approach makes more individualized menus that look similar but behave different.

Listing 7-34. Lists of links (ListLinks.html)

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    These options
  </a>
  <a href="#" class="list-group-item">are available</a>
  <a href="#" class="list-group-item">to all users</a>
  <a href="#" class="list-group-item">of this app</a>
</div>
```

This presentation looks very similar to buttons—only the function corresponds to a hyperlink (see Figure 7-37). You should be wary of these representations because the behavior is not immediately apparent to the user.

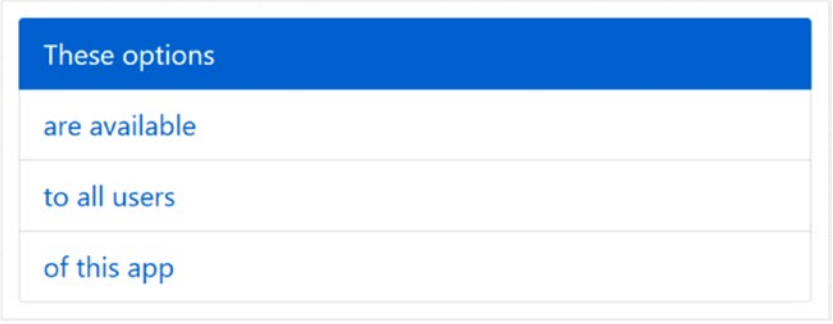


Figure 7-37. *Lists of links*

Buttons in Lists

You can use buttons in lists (see Listing 7-35 and Figure 7-38). Unlike normal buttons, these are *not* based on the class *.btn*. They provide a frame, but are not colored on the entire surface as with normal buttons.

Listing 7-35. Lists with buttons (ListBtns.html)

```
<div class="list-group">
  <button type="button" class="list-group-item list-group-item-success">
    Option A
  </button>
```

```
    <button type="button" class="list-group-item list-group-item-danger">
      Option B
    </button>
    <button type="button" class="list-group-item list-group-item-info">
      Option C
    </button>
  </div>
```

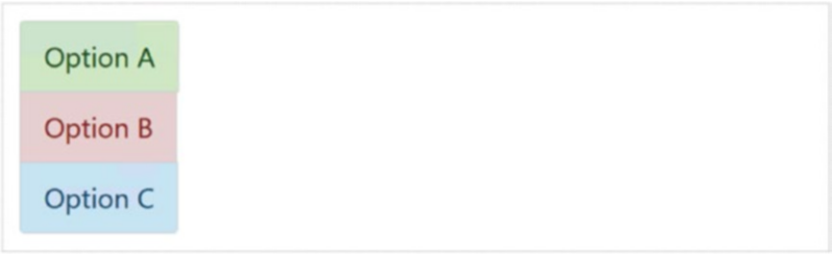


Figure 7-38. *Lists with buttons*

With the class *.disabled* within *.list-group-item* the contents will appear disabled.

```
<div class="list-group">
<a href="#" class="list-group-item disabled">
  These options
</a>
<a href="#" class="list-group-item">are here</a>
<a href="#" class="list-group-item">for all users</a>
<a href="#" class="list-group-item">this application</a>
<a href="#" class="list-group-item">generally available</a>
</div>
```

List elements can have a semantic meaning (see Listing 7-36 and Figure 7-39).

- *list-group-item-success*: Success, green
- *list-group-item-info*: Information, azure
- *list-group-item-warning*: Warning, orange
- *list-group-item-danger*: Danger, red

Active elements can also be highlighted with the *.active* class.

Listing 7-36. Semantic lists with and without links (ListSemantics.html)

```
<ul class="list-group">
  <li class="list-group-item list-group-item-success">
    Success
  </li>
```

```
    <li class="list-group-item list-group-item-info">
      Info
    </li>
    <li class="list-group-item list-group-item-warning">
      Warning
    </li>
    <li class="list-group-item list-group-item-danger">
      Danger
    </li>
  </ul>
  <div class="list-group">
    <a href="#"
       class="list-group-item list-group-item-success">
      Success
    </a>
    <a href="#"
       class="list-group-item list-group-item-info">
      Info
    </a>
    <a href="#"
       class="list-group-item list-group-item-warning">
      Warning
    </a>
    <a href="#"
       class="list-group-item list-group-item-danger">
      Danger
    </a>
  </div>
```

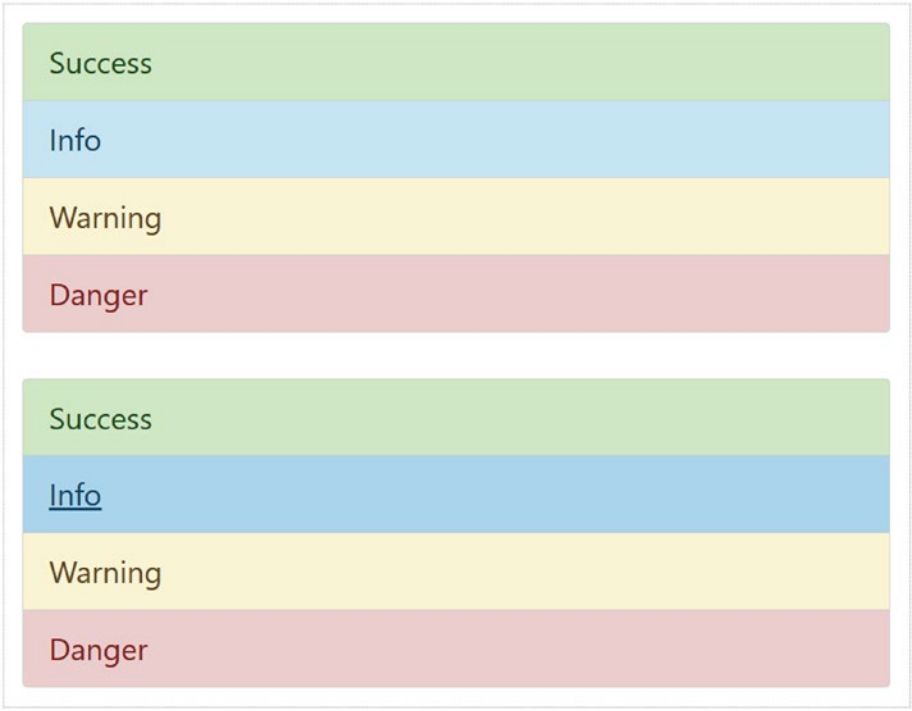


Figure 7-39. *Semantic lists with and without links*

The list elements themselves are very flexible and can include almost all of the other components as shown in Listing 7-37 and Figure 7-40.

Listing 7-37. Lists with headings (ListHeader.html)

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    <h4 class="list-group-item-heading">Titel</h4>
    <p class="list-group-item-text">...</p>
  </a>
</div>
```

Figure 7-40. *Lists with headings*

Cards

Cards are highlighted areas that serve to structure the page. These areas may have a semantic meaning. They are very diverse and serve a wide range of tasks.

Panels in Bootstrap 4

Cards replace the previous components panels, wells, and thumbnails.

Semantic meaning is achieved with the following additional classes.

- *card-default*: Standard, gray, no meaning
- *card-primary*: Primary statement, blue, data or earnings
- *card-success*: Success, green, status message
- *card-info*: Information, light blue, other information, supplements
- *card-warning*: Warning, orange, status message
- *card-danger*: Error, red, status message

The default panel looks as in Listing 7-38 and as shown in Figure 7-41.

Listing 7-38. Card (Cards_Danger.html)

```
<div class="card card-danger card-inverse">
  <div class="card-text">
    Example text
  </div>
</div>
```

Figure 7-41. *Card with class .card-danger*

Semantic variants have the same shape and change the color of the background like buttons.

```
<div class="card card-primary">...</div>
<div class="card card-success">...</div>
<div class="card card-info">...</div>
<div class="card card-warning">...</div>
<div class="card card-danger">...</div>
```

If the font color has an insufficient contrast, you can get a better representation using the class *.card-inverse*. This makes sense, at least in *danger* class where the font color changes for the dark red background from black to white.

The standard menu is preceded by *.card*. A general internal spacing (padding) is achieved with *.card-block*. The text is created in *.card-text*.

Headings

Cards can create complex reports and can include a header area. This part is preceded by *.card-header*. In the header, the size can be varied with `<h1>-<h6>`.

First, Listing 7-39 provides the default formatting of the heading and Figure 7-42 shows a heading.

Listing 7-39. Card with heading (Karten_Header.html)

```
<div class="card card-block">
  <div class="card-header">Message</div>
  <p class="card-body">
    Message text
  </p>
</div>
```

Figure 7-42. *Card with heading*

To style headings use a code that looks like the one in Listing 7-40 and shown in Figure 7-43.

Listing 7-40. Card with title (Cards_Header2.html)

```
<div class="card card-block">
  <h2 class="card-header">Message</h2>
  <p class="card-body">
    Message Text
  </p>
</div>
```

Figure 7-43. *Card with title and heading element*

It is further possible to place sub-headings on *.card-subtitle*.

Footers

The footer areas of panels appear in *.panel-footer*. The footer does not inherit semantic information and it is up to you to add information here (see Listing 7-41 and Figure 7-44). Footers are for the placement of buttons, and these can bring their own semantic context, if necessary. Resulting actions are then programmed separately as the footer provides no interactive elements.

Listing 7-41. Semantic card with footer (Cards_Footer.html)

```
<div class="card card-success">
  <p class="card-text">
    This is the content...
  </p>
  <div class="card-footer">Footer</div>
</div>
```

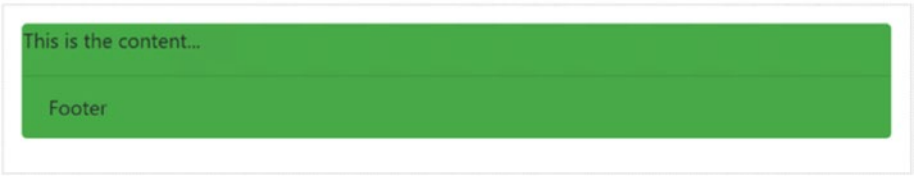


Figure 7-44. *Semantic card with footer*

Cards with Tables

The card presentation of data provides tables for output, too. Note the limited possibilities for responsive layouts that tables provide though. Because cards already have a frame, tables should be drawn without a frame.

The placement may be made in *.card-text* or placed outside. If it is made within, the table should get additional distance—if space permits (see Listing 7-42 and Figure 7-45).

Listing 7-42. Card with table (Cards_Table.html)

```
<div class="card">
  <div class="card-header">Titel</div>
  <div class="card-block">
    <p>...</p>
  </div>

  <!-- Table -->
  <table class="table table-sm">
    ...
  </table>
</div>
```

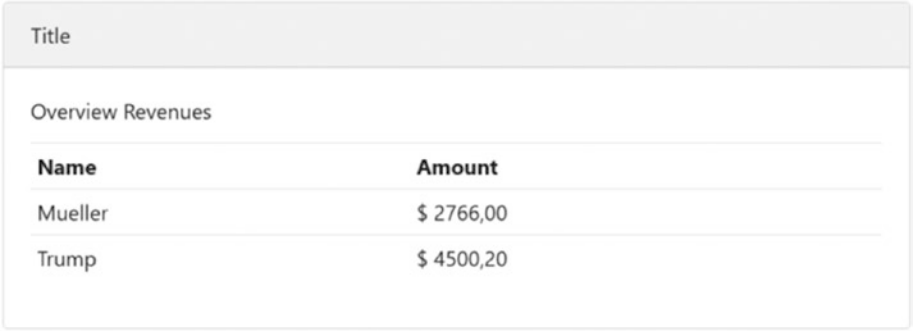


Figure 7-45. *Card with table*

Within *.card-text* there is a seamless transition between content and table.

Cards with Lists

Lists can be simply placed in cards as shown in Listing 7-43 and Figure 7-46.

Listing 7-43. Card with list (Cards_List.html)

```
<div class="card">
  <div class="card-header">Knowledge</div>
    <div class="card-block">
      <ul class="list-group list-group-flush">
        <li class="list-group-item">Bootstrap</li>
        <li class="list-group-item">jQuery</li>
        <li class="list-group-item">AngularJS</li>
        <li class="list-group-item">ReactJS</li>
      </ul>
    </div>
  </div>
```

Figure 7-46. *Card with list*

Card with Pictures

Images can be as simple as a map in the background (overlay). The superimposition of text is achieved by *.card-img-overlay* (see Listing 7-44 and Figure 7-47).

Listing 7-44. Card with background image (Cards_Image.html)

```
1   <div class="card text-center"
2        style="width: 300px">
3     <img class="card-img-top"
4          src="../Res/Background.jpg" alt="Lamp">
5     <div class="card-img-overlay card-inverse text-right">
6       <h4 class="card-title">Recieved?</h4>
7       <p class="card-text">This is Bootstrap 4</p>
8     </div>
9     <div class="card-block">
10      It continues with new styles...
11    </div>
12  </div>
```

The image should be tailored to the width of the card, as shown in Listing 7-44 on line 2 to 300 px. Please note that such fixed values may not be responsive.

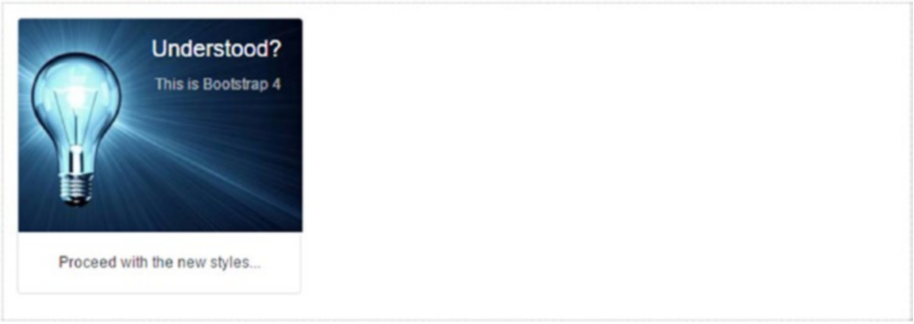


Figure 7-47. *Card with background image*

There is the option to combine the width and the distance to the column classes for placement in the grid.

Additional functions for placing images are as follows:

- *.card-img-top*: Aligning the image upward
- *.card-img-bottom*: Aligning the image downward

These classes are directly applied to the `<img />` tag.

Cards in the Grid

The last example demonstrated how the width can be adjusted directly by a style. Bootstrap has no explicit classes for this purpose. It is better to adjust the placement of the grid, as shown in Listing 7-45 and Figure 7-48.

Listing 7-45. Cards in the grid (Cards_Grid.html)

```
<div class="container-fluid sample">
  <div class="row">
    <div class="col-xs-6">
      <div class="card card-success">
        <div class="card-block card-inverse">
          <h4 class="card-title">Message</h4>
          <p class="card-text">
           This action was successful.
          </p>
          <a href="#" class="card-link">
             More...
          </a>
```

```
            <a href="#" class="card-link">
               Back...
            </a>
          </div>
        </div>
      </div>
      <div class="col-xs-6">
        <div class="card card-warning">
          <div class="card-block">
            <h4 class="card-title">Message</h4>
            <p class="card-text">
              Something went wrong.
            </p>
            <a href="#" class="btn btn-danger">
              Exception...
            </a>
          </div>
        </div>
      </div>
    </div>
  </div>
```

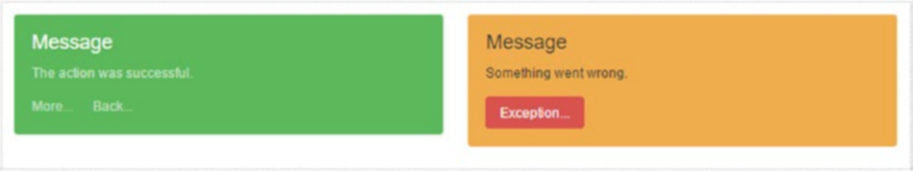


Figure 7-48. *Cards in the grid*

Deck of Cards

Cards can be arranged horizontally so that all the elements have the same height (see Listing 7-46 and Figure 7-49).

Listing 7-46. Align cards (Cards_Deck.html)

```
<div class="container-fluid sample">
<div class="card-deck-wrapper">
   <div class="card-deck">
   <!-- Card 1 -->
   <div class="card">
       <div class="card-header">Help 1</div>
       <div class="card-block">
       <p class="card-text">Help text.</p>
```

```
9           </div>
10       </div>
11       <!-- Card 2 -->
12          <div class="card">
13           <div class="card-header">Help 2</div>
14           <div class="card-block">
15           <p class="card-text">Help text and
16               <a href="#" class="card-link">more help</a>.
17            </p>
18            <p class="card-text">This is more text than in the first box</p>
19            </div>
20        </div>
21        </div>
22    </div>
23    </div>
```

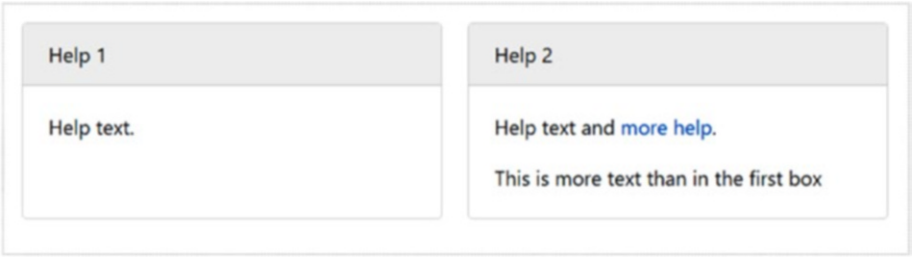


Figure 7-49. *Align cards*


The horizontal alignment may be connected to the column layout, so that you also can control the width.

Card Groups

Cards can be horizontally arranged so that all the elements have the same height. The groups are—as opposed to the stack—directly connected to each other. For this purpose, the card group is packaged in a class *.card-group* (see Listing 7-47 and Figure 7-50).

Listing 7-47. Card group (Cards_Group.html)

```
<div class="card-group">
  <!-- Card 1 -->
  <div class="card card-success">
    <div class="card-header">Info</div>
    <div class="card-block">
      <p class="card-text">Info text here</p>
    </div>
  </div>
  <!-- Card 2 -->
  <div class="card card-info">
    <div class="card-header">Message</div>
    <div class="card-block">
      <p class="card-text">Here comes more...</p>
    </div>
  </div>
</div>
```

Figure 7-50. *Card group*


The horizontal alignment may be connected to the column layout, so that the width can be controlled.

Cards in Columns

Cards can be distributed horizontally in columns. So complex layouts with dedicated designs of blocks are more possible. For this purpose, the card group is packaged in a class *.card-columns* (see Listing 7-48 and Figure 7-51).

Listing 7-48. Card columns (Cards_Columns.html)

```
<div class="container sample">
  <div class="row">
    <div class="card-columns">
      <!-- Card 1 -->
      <div class="card">
        <div class="card-header">Card 1</div>
        <div class="card-block">
          <p class="card-text">This is the content.</p>
        </div>
      </div>
      <!-- Card 2 -->
      <div class="card">
        <div class="card-block">
          <h4 class="card-title">Card 2</h4>
          <p class="card-text">This is the content.</p>
        </div>
      </div>
      <!-- Card 3 -->
      <div class="card card-outline-info">
        <div class="card-header">Card 3</div>
        <div class="card-block card-info">
          <p class="card-text">
            Here is the content.
            A <a href="#" class="card-link">Link</a>.
          </p>
        </div>
        <div class="card-footer">Footer</div>
      </div>
      <!-- Card 4 -->
      <div class="card card-outline-warning">
        <div class="card-header">Card 4</div>
        <div class="card-block card-warning">
          <p class="card-text">This is the content.</p>
        </div>
        <div class="card-footer">Footer</div>
      </div>
      <!-- Card 5 -->
      <div class="card">
        <div class="card-block">
          <h4 class="card-title">Card 5</h4>
          <p class="card-text">This is the content.</p>
        </div>
      </div>
```

```
        <!-- Card 6 -->
        <div class="card card-outline-success">
          <div class="card-header">Card 6</div>
          <div class="card-block card-success">
            <p class="card-text">
              Here is the content.
              A <a href="#" class="card-link">Link</a>.
            </p>
          </div>
        </div>
        <!-- Card 7 -->
        <div class="card">
          <div class="card-header">Card 7</div>
          <div class="card-block">
            <p class="card-text">
              Here is the content
              A <a href="#" class="card-link">Link</a>.
            </p>
          </div>
          <div class="card-footer">End</div>
        </div>
        <!-- Cars 8 -->
        <div class="card">
          <div class="card-block">
            <h4 class="card-title">Card 8</h4>
            <p class="card-text">Here is the content.</p>
          </div>
        </div>
      </div>
    </div>
  </div>
```

In this example the semantic classes *.card-outline-<>* is used to assign the color to the border only. This replaces the regular *.card-<>* classes that give cards a semantic meaning.

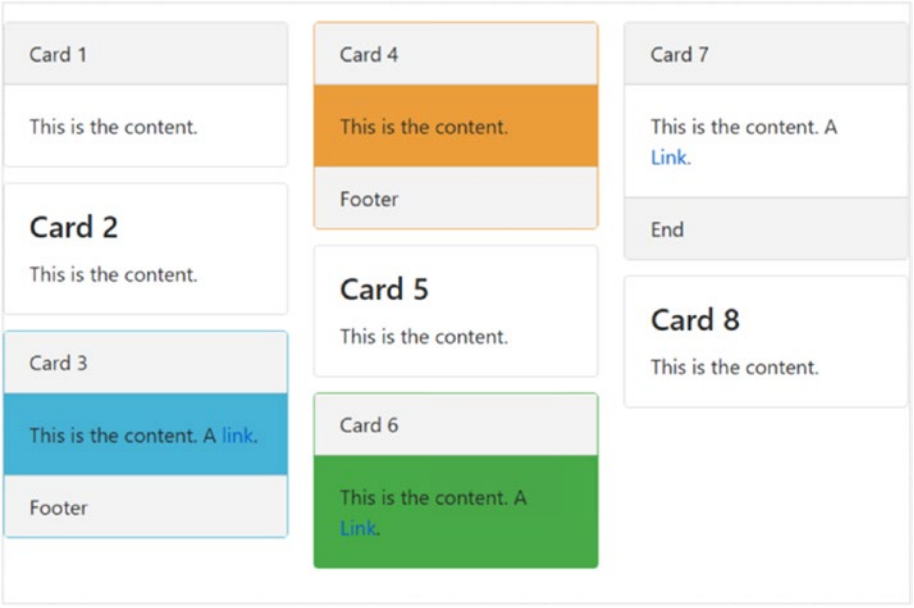


Figure 7-51. *Card columns*

CHAPTER 8

Active Components

Active components in bootstrap not only have support in JavaScript, but that support can be programmed directly, too. Technically there are jQuery plug-ins. If the entire bootstrap library is not used, it is possible that there some dependencies to parts of the library must be respected.

Setup and Activation

The files *bootstrap.js* and *bootstrap.min.js* (minimized) already contain all the components. If these are already integrated, no additional measures are necessary. To use them, there are two options: Either you use the HTML5 attributes `data-` or the JavaScript API. For the purposes of modern HTML5 programming using attributes is the preferred way. The use of HTML5 attributes is preferred because it needs less script code, and thus there is less risk of making mistakes when programming.

The use of API for HTML5 attributes may be in rare cases inappropriate or disturbing. If so, it is possible to disable them globally:

```
1   $(document).off('.data-api')
```

If only a single plug-in is affected, you also can disable just that plug-in:

```
1   $(document).off('.alert.data-api')
```

In more complex scenarios, there is often the temptation to use multiple plug-ins with a component. That does not work. Imagine a button, which activates a modal dialog and simultaneously displays a tooltip. Although this may be useful in design, it does not work. This can be achieved simply by another enclosing member. The outer member—without frames and distances—controls the tooltip and the inner one the modal dialog.

jQuery UI?

Although jQuery is the basis of Bootstrap components, jQuery UI competes with Bootstrap and should not be used simultaneously under any circumstances.

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6_8

The Programming Interface

With websites created entirely in JavaScript, which is quite common, there is no possibility using the attributes as an interface. This can be accessed directly on the API of the components:

```
1   $('.btn.danger').button('toggle').addClass('fat');
```

All components process an *Options* object that needs to be provided as JSON. The actual values are dependent on the component.

```
1   $('#myModal').modal()                    // Only Standard
2   $('#myModal').modal({ keyboard: false })  // With Option
3   $('#myModal').modal('show')              // Trigger action
```

Each component has a constructor: `$.fn.popover.Constructor`.
The constructor returns a property `DEFAULTS` that allows access to the default values.

```
1   $.fn.modal.Constructor.DEFAULTS.keyboard = false;
```

 `$.fn` is the access to registered plug-ins in jQuery. This part is not specific to Bootstrap.

A specific instance can be obtained as follows:

```
$('[rel="popover"]').data('popover'');
```

This is an example of a "popover."

Conflict Prevention

If jQuery is still used in addition to other libraries or jQuery plug-ins, it may lead to name conflicts. As JavaScript does not have namespaces, you may need to manually help here. This is the purpose of the method `noConflict`.

```
1   var bootstrapButton = $.fn.button.noConflict()
2   $.fn.bootstrapBtn = bootstrapButton
```

In line 1, an instance of the plug-in is retrieved. This is then assigned a new name in line 2, which does not interfere with other elements.

In this context, versions are important. These can be retrieved with `VERSION`:

```
1   $.fn.tooltip.Constructor.VERSION // => "4.0.0"
```

Events

Bootstrap components generate some private events. Most of these are available in two versions—occurrence of the event and at the end of processing. Accordingly, the verbs are available: basic and past tense ("show" and "shown"). All events are in their own namespaces ".bs.".

As some events are handled internally, you can suppress them by *preventDefault*. This is the typical way for jQuery.

```
1   $('#myModal').on('show.bs.modal', function (e) {
2     if (!data) return e.preventDefault()
3   })
```

Here the code in line 2 prevents the standard behavior. The modal dialog, which actually appears in "show" is suppressed.

Transitions

Transitions are largely covered by CSS3. To support older browsers, there are covered by the auxiliary library *transitions.js*, mapping the effects in JavaScript. This library is part of *bootstrap.js* and does not need to be installed separately. This is only necessary when parts of the library are used.

Transition effects can be bothersome. Therefore, they can be switched off globally:

```
1   $.support.transition = false
```

Animations

Animations—whether as direct effect or transition—are only funny the first time. Omit them and the users benefit.

Applications of the module include things such as:

- Dialogues are gently faded in and out.
- Switching between tabs is changed gently.
- Messages can be smoothly displayed.
- The Carousel has replaced its content with animation.

The effect is triggered by the class *.fade*.

Modal Dialogs

Modal dialogs are from *modals.js*. This library is part of *bootstrap.js* and does not need to be installed separately.

Features

Modal dialogs seize exclusive focus and demand an unconditional response from the user. You should never use modal dialogs without possibility of interaction. A button or an icon for closing is always required.

A modal dialog is exclusive, so only one dialog at a time can be open.

Normally you should place modal dialogs on top of the page. This prevents conflicts with other settings. When and where they are called, however, is completely independent.

On mobile devices, it could happen that the first element of the dialogue is not in focus, even if the attribute `autofocus` is used. If necessary, you can use a little JavaScript to post help:

```
1   $('#myModal').on('shown.bs.modal', function () {
2     $('#myInput').focus()
3   })
```

A standard dialog consists of three parts:

- Header area
- Content
- Footer area

The title comes in the header area, the contents then fills the dialogue, and action buttons are placed in the footer as shown in Listing 8-1:

Listing 8-1. Modal dialog (Modal.html)

```
1   <div class="modal fade">
2     <div class="modal-dialog">
3       <div class="modal-content">
4         <div class="modal-header">
5           <button type="button" class="close"
6                   data-dismiss="modal"
7                   aria-label="Close">
8             <span aria-hidden="true">&times;</span>
9           </button>
10          <h4 class="modal-title">Titel</h4>
11        </div>
12        <div class="modal-body">
13          <p>The content &hellip;</p>
14        </div>
```

```
15          <div class="modal-footer">
16            <button type="button" class="btn btn-default"
17                    data-dismiss="modal">Close</button>
18            <button type="button" class="btn btn-primary">
19              Save
20            </button>
21          </div>
22        </div>
23      </div>
24    </div>
```

To display the dialog, an action on the page is required. This may look like Listing 8-2 and Figure 8-1:

Listing 8-2. Trigger for dialogue (Modal.html)

```
1   <button type="button" class="btn btn-primary btn-lg"
2           data-toggle="modal"
3           data-target="#myModal">
4     Show Dialog
5   </button>
```

Figure 8-1. *Modal dialog*

As with previous examples, a barrier-free access is useful. First take the attributes `role="dialog"` and `aria-labelledby="..."`, the latter should point to an element that contains the title. As the dialogue itself, the contents are decorated with `role="document"`. If a description is required, which may not appear useful in normal operation because the dialogue is in the context or contains images and symbols, use beyond `aria-describedby` on the element with the class *.modal*.

Sizes

Modal dialogs have three sizes, which are set by two classes:

- *.bs-example-modal-lg*: Greater than normal
- *.bs-example-modal-sm*: Smaller than normal

```
<button type="button" class="btn btn-primary"
        data-toggle="modal"
        data-target=".bs-example-modal-lg">
  Big dialogue
</button>

<div class="modal fade bs-example-modal-lg"
     tabindex="-1"
     role="dialog"
    aria-labelledby="myLargeModalLabel">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

```
<button type="button" class="btn btn-primary"
        data-toggle="modal"
        data-target=".bs-example-modal-sm">Small</button>

<div class="modal fade bs-example-modal-sm"
     tabindex="-1"
     role="dialog"
     aria-labelledby="mySmallModalLabel">
  <div class="modal-dialog modal-sm">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

Animations are created with *.fade*. This can be annoying and should not be used on mobile devices.

 The effect of fading requires significant processing power and reduces the battery life in mobile devices.

Dialog with Grid

Dialogs can be extensive. Therefore, they also can benefit from grids. To achieve this the body (*.modal-body*) of the dialog gets a new container *.container-fluid* within. In this container, the normal grid classes are applicable. The outer width depends on the width of the dialog, the values are in percentage terms (see Listing 8-3 and Figure 8-2).

Listing 8-3. Complex dialogue with grid (Modal_Complex.html)

```
<div class="modal fade" role="dialog" id="myModal"
     aria-labelledby="gridSystemModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close"
                data-dismiss="modal"
                aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title" id="gridSystemModalLabel"> Title</h4>
      </div>
      <div class="modal-body">
        <div class="container-fluid">
          <div class="row">
            <div class="col-md-4">4</div>
            <div class="col-md-4 offset-md-4">4 4</div>
          </div>
          <div class="row">
            <div class="col-md-3 offset-md-3">3 3</div>
            <div class="col-md-2 offset-md-4">2 4</div>
          </div>
          <div class="row">
            <div class="col-md-6 offset-md-3">6 3</div>
          </div>
          <div class="row">
            <div class="col-sm-9">
              Level 1: .col-sm-9
              <div class="row">
                <div class="col-xs-8 col-sm-6">
                  Level 2: .col-xs-8 .col-sm-6
                </div>
                <div class="col-xs-4 col-sm-6">
                  Level 2: .col-xs-4 .col-sm-6
                </div>
              </div>
            </div>
          </div>
        </div>
```

```
40          </div>
41          <div class="modal-footer">
42                 <button type="button"
43                      class="btn btn-secondary"
44                      data-dismiss="modal">
45                  Close
46                 </button>
47                 <button type="button"
48                      class="btn btn-primary">
49                  Save
50                 </button>
51              </div>
52           </div>
53       </div>
54    </div>
55    <div class="container sample">
56       <button type="button"
57            class="btn btn-primary btn-lg"
58            data-toggle="modal"
59            data-target="#myModal">
60         Show dialog
61       </button>
62    </div>
```

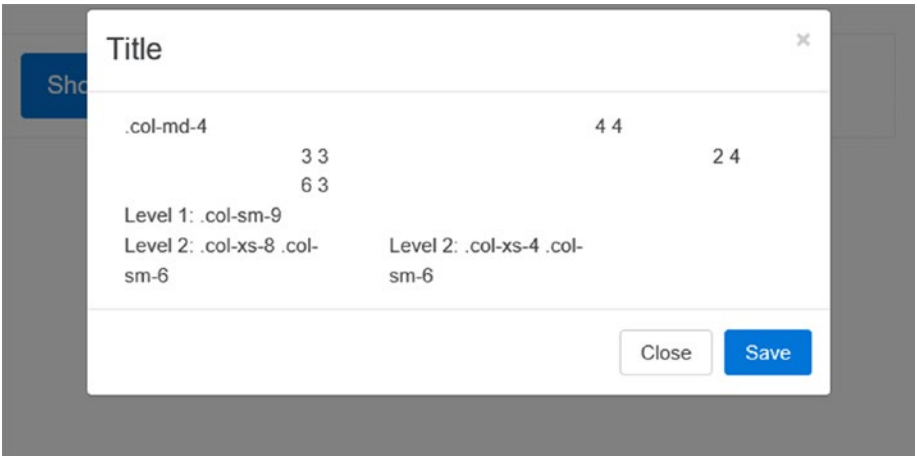


Figure 8-2. *Grid dialog*

The button or action that triggers the dialogue is initiated can transmit more values to modify the dialog. Listing 8-4 shows how this is accomplished by `data-` attributes.

Listing 8-4. Controlled dialog (Modal_Data.html)

```
<button type="button" class="btn btn-primary"
        data-toggle="modal"
        data-target="#exampleModal"
        data-whatever="anna@muster.de">
   Open Anna
</button>
<button type="button" class="btn btn-primary"
        data-toggle="modal"
        data-target="#exampleModal"
        data-whatever="berta@muster.de">
  Open Berta
</button>
<button type="button" class="btn btn-primary"
        data-toggle="modal"
        data-target="#exampleModal"
        data-whatever="chri@muster.de">
  Open Chris
</button>


<div class="modal fade" id="exampleModal" tabindex="-1"
     role="dialog" aria-labelledby="exampleModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close"
                data-dismiss="modal"
                aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title" id="exampleModalLabel">
          New message
        </h4>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group">
            <label for="recipient" class="control-label">
              Receiver:
            </label>
            <input type="text" class="form-control"
                   id="recipient">
          </div>
          <div class="form-group">
            <label for="message-text" class="control-label">
              Message:
```

```
47              </label>
48              <textarea class="form-control"
49                        id="message-text">
50              </textarea>
51            </div>
52          </form>
53        </div>
54        <div class="modal-footer">
55          <button type="button" class="btn btn-default"
56                  data-dismiss="modal">
57            Conclude
58          </button>
59          <button type="button" class="btn btn-primary">
60            Send
61          </button>
62        </div>
63      </div>
64    </div>
65  </div>
```

The evaluation then takes some JavaScript as follows:

```
1   $(function() {
2      $('#exampleModal').on('show.bs.modal', function(event) {
3        var button = $(event.relatedTarget);
4        var recipient = button.data('whatever');
5        var modal = $(this);
6        modal.find('.modal-title')
7             .text('Message' + recipient);
8        modal.find('#Recipient').val(recipient);
9      });
10  });
```

In line 2, the button is determined. In line 3 the private attribute `data-whatever is` accessed to copy the data in the dialog. This is the easiest way via the DOM of dialogue (See Figure 8-3).

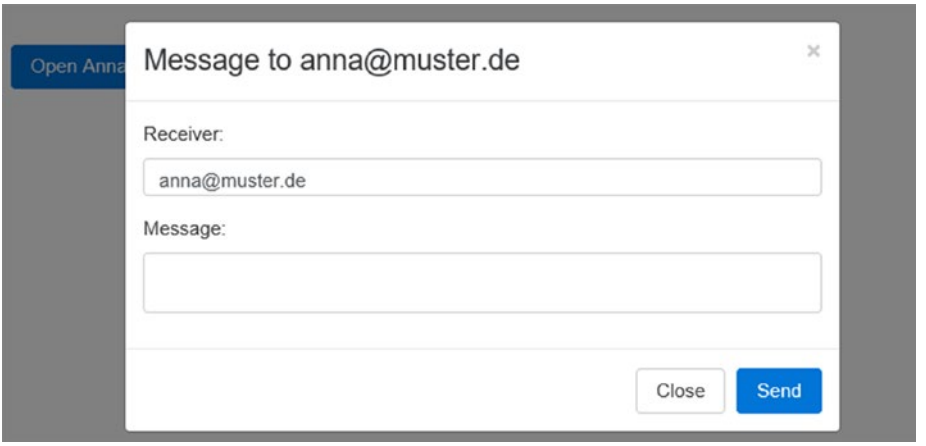


Figure 8-3. *Controlled dialog*

General Information about Behavior

Modal dialogs will give the page a slightly opaque look to catch the attention of the user. That is done internally by adding the class *.modal- open* to the body element. You can give to this class more styles to customize the effect. Agent *.modal-backdrop* used another class that ensures that the dialog will be closed when the user clicks outside (despair click).

The trigger behavior is based on attributes:

- `data-toggle="modal"`: Trigger, for example, on a button
- `data-target="#foo"`: Target for dialogue *#foo*
- `href="#foo"`: Alternative specifying the destination for dialog *#foo*
- `id="foo"`: Decoration of the dialogue itself (destination ID)

```
<button type="button"
        data-toggle="modal"
        data-target="#myModal">
  Show Modal
</button>
```

In JavaScript, the selectors of jQuery are used:

```
$('#myModal').modal(options)
```

Options

Options may be set as a `data--`Attribute in HTML or JSON code. The suffix of the `data--` Attribute corresponds to the name of the JSON property. Table 8-1 lists options, Table 8-2 lists actions, and Table 8-3 lists events for modal dialogs.

Table 8-1. *Options for Modal Dialogs*

Name	Type	Description
Backdrop	Boolean or "static"	`True` will close when clicking outside; "static" suppressed closing.
Keyboard	Boolean	Closes the dialog when `ESC` is pressed.
Show	Boolean	Instant display during initialization

Table 8-2. *Actions for Modal Dialogs*

Name	Description
Show	Dialog appears
Toggle	Toggle display state asynchronously
Hide	Close
HandleUpdate	Repositioned by moving or show the scrollbar

Table 8-3. *Events for Modal Dialogs*

Name	Description
show.bs.modal	Dialog appears
hide.bs.modal	Dialog closing
shown.bs.modal	Appeared (on transitions and animations will be serviced)
hidden.bs.modal	Has been closed (on transitions and animations will be serviced)
loaded.bs.modal	Is loaded

Use

The calls to your code look like this:

```
.modal(options)
```

The complete code based on HTML code with the ID *myModal* might look like this:

```
1   $('#myModal').modal('show')
```

Please read the following Fluent syntax on line 4, which will be continued with the object to trigger further actions.

```
1   $('#myModal').modal({
2     keyboard: false
3   })
4   .modal('toggle')
```

For events to respond add the callback function:

```
1   $('#myModal').on('hidden.bs.modal', function (e) {
2     // Event code...
3   });
```

Pull-Down Menu (Dropdown)

The drop-down menu is available in a simple form without JavaScript support. The possible interactions enhance the function of navigation bar (navbar), tabs, and pills.

Common Information about Behavior

Show invisible elements dynamically and again hide can be carried out by turning on and off the class *.open*. On mobile devices *.dropdown-backdrop* will work for the whole screen and "tap" for the event. That is, it does not automatically close the menu on such devices, but must be closed with extra finger pressure before another menu item can be selected.


The attribute `data-toggle="dropdown"` is always required.

Options

The pull-down menu does not options, but see Table 8-4 for actions and Table 8-5 for events for drop menus.

Table 8-4. *Actions for Drop Menus*

Name	Description
Toggle	Display state asynchronously

Table 8-5. *Events for Drop Menus*

Name	Description
show.bs.dropdown	Show
hide.bs.dropdown	Close
shown.bs.dropdown	Display was (on transitions and animations will be serviced)
hidden.bs.dropdown	Has been closed (on transitions and animations will be serviced)
loaded.bs.dropdown	Is loaded

Usage

Based on the code of the static folding the HTML menu is practically identical:

```
<div class="dropdown">
  <button id="dLabel" type="button"
          data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
    Trigger drop
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

Hyperlinks are used and instead of the `href`-attribute you use `data-target` instead of `href="#"`.

```
<div class="dropdown">
  <a id="dLabel" data-target="#" href="http://example.com"
     data-toggle="dropdown" role="button"
     aria-haspopup="true" aria-expanded="false">
    Trigger drop
    <span class="caret"></span>
  </a>

  <ul class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

The code to use is as follows:

```
$('.dropdown-toggle').dropdown()
```

For events to respond you add the callback function:

```
$('#myDropdown').on('show.bs.dropdown', function () {
  // Do something...
})
```

Scroll Bar Supervisor (ScrollSpy)

This component recognizes the position of the scroll bar. The jump destinations can be dependent on the update position of the scroll bar in a navigation bar. This is especially interesting for very long pages, where the jump destinations in a list of hyperlinks at the side are left or right and above that are always visible. When the user clicks on a link, the page jumps to the target position. This is normal browser behavior. However, if the scroll bar is used, a certain position sometimes appears. With the *Scrollbar-supervisor* the menu is then suitably adjusted to the position so that the current visible section is highlighted. Thus, for the menu to display this effect, it must remain visible while scrolling. The purpose should be tacking navigation. This can be achieved with `position: sticky`.

The affix component from Bootstrap 3 no longer exists.

Use

First, a navigation component is needed. The resolution of the jump destinations must be done on *id*. An object that was created with `<a href="#home">home</a>` must lead to a matching `<div id="home"></div>`. Invisible elements are ignored. The criterion for visibility applies which jQuery when using the pseudoselectors: `:visible` indicators. This monitors the scroll area of an element. It needs `position:relative` to be positioned. This is usually `<body>` and there is no further action required. It is another element that also must `overflow-y: scroll` and a height must also be specified.

The activation is done with `data-spy="scroll"` on the monitored element, for example, `<body>`. Then `data-target` is put on the *id* or class of the parent element of a component *.nav* that was started with.

```
<style>
body {
  position: relative;
}
</style>

<body data-spy="scroll" data-target="#navbar-example">
  ...
  <div id="navbar-example">
    <ul class="nav nav-tabs" role="tablist">
      ...
    </ul>
  </div>
  ...
</body>
```

When using JavaScript it looks like this:

```
$('body').scrollspy({ target: '#navbar-example' })
```

Dynamically added, elements must call *'refresh'*.

```
$('[data-spy="scroll"]').each(function () {
  var $spy = $(this).scrollspy('refresh')
})
```

For only events respond to add the callback function:

```
$('#myScrollspy').on('activate.bs.scrollspy', function () {
  // tu was...
})
```

Options

Options may be set as a `data--`attributes in HTML or JSON code.

The suffix of the `data--`attributes corresponds to the name of the JSON property. Table 8-6 lists the options, Table 8-7 lists the actions, and Table 8-8 lists the events for (*ScrollbarSpy*).

Table 8-6. *Options for (ScrollbarSpy)*

Name	Type	Description
Offset	Number	Distance from the top, is responsive to the appearance from. The default value is 10 pixels.

Table 8-7. *Action for (ScrollbarSpy)*

Name	Description
Refresh	Synchronize dynamic elements

Table 8-8. *Events for (ScrollbarSpy)*

Name	Description
activate.bs.scrollspy	Activated

Pinned Navigation (Affix)

This component is no longer used Bootstrap 4. Use `position: sticky` directly or as a polyfill for older browsers. The polyfill is an extension to CSS3 that extends the function with JavaScript. A good implementation is *StickyFill*:

- *https://github.com/wilddeer/stickyfill*

This is used as follows in CSS:

```
1   .sticky {
2     position: -webkit-sticky;
3     position: sticky;
4     top: 0;
5   }
```

It is activated via JavaScript, as with jQuery:

```
1   $('.sticky').Stickyfill();
```

Install Stickyfill as follows:

```
bower install Stickyfill -save
```

Reversible Tabs (Tab)

Reversible tabulators can be modified by other elements. This is only supported on one level—not for nested elements.

Use

Activation in JavaScript looks like this:

```
1   $('#myTabs a').click(function (e) {
2     e.preventDefault()
3     $(this).tab('show')
4   })
```

The activation can be carried out flexibly by means of selectors:

```
1   $('#myTabs a[href="#profile"]').tab('show')
2   $('#myTabs a:first').tab('show')
3   $('#myTabs a:last').tab('show')
4   $('#myTabs li:eq(2) a').tab('show')
```

The code in line 1 uses a name; line 2 selects the first tab, line 3 the last. Line 4 is the third (2, that is, the value is 0-based) selected.

The attribute `data-toggle="tab"` or `data-toggle="pill"` does this in the markup (see Listing 8-5).

Listing 8-5. Interactive tabs

```
<div>

  <!-- Nav tabs -->
  <ul class="nav nav-tabs" role="tablist">
    <li role="presentation" class="active">
      <a href="#home" aria-controls="home"
         role="tab" data-toggle="tab">Home</a>
    </li>
    <li role="presentation">
      <a href="#profile" aria-controls="profile"
         role="tab" data-toggle="tab">Profile</a>
    </li>
    <li role="presentation">
      <a href="#messages" aria-controls="messages"
         role="tab" data-toggle="tab">Messages</a>
    </li>
    <li role="presentation">
      <a href="#settings" aria-controls="settings"
         role="tab" data-toggle="tab">Settings</a>
    </li>
  </ul>

  <!-- Tab-content -->
  <div class="tab-content">
    <div role="tabpanel" class="tab-pane active" id="Home">
      ...
    </div>
    <div role="tabpanel" class="tab-pane" id="Profile">
      ...
    </div>
    <div role="tabpanel" class="tab-pane" id="Messages">
      ...
    </div>
    <div role="tabpanel" class="tab-pane" id="Settings">
      ...
    </div>
  </div>

</div>
```

For only events respond to add the callback function:

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
  e.target // NeW Tab
  e.relatedTarget // Previous Tab
})
```

Options

Options may be set as a `data-`attributes in HTML or JSON code. The suffix of the `data-` attributes corresponds to the name of the JSON property. Table 8-9 lists the options, Table 8-10 lists the actions, and Table 8-11 lists the events for *Tabs*.

Table 8-9. *Options for Tabs*

Name	Type	Description
Offset	Number	Distance from the top, is responsive to the appearance from. The default value is 10 pixels.

Table 8-10. *Actions for Tabs*

Name	Description
Show	Show the component

Note that there is no "hide" because one tab is always active. If one is enabled, all others are hidden.

Table 8-11. *Events for Tabs*

Name	Description
hide.bs.tab	Tab is hidden
show.bs.tab	Tab is shown
hidden.bs.tab	Tab was hidden (by animation)
shown.bs.tab	Tab has been displayed (after animation)

Tooltip

Tooltips serve to provide useful information for the user (see Listing 8-6). They respond to a floating (hover) mouse. Because showing to many tooltips is critical to the performance of the browser they must be self-activated—the attributes alone are not enough.

The tooltips require an additional library, "tether.js". There are also different styles available. See "tether.css" for details and themes, that are available in the same folder. Tether is a dependency that is automatically installed with Bootstrap 4.

Listing 8-6. Tooltips for buttons

```
<button type="button" class="btn btn-default"
        data-toggle="tooltip" data-placement="left"
        title="Tooltip on left">Tooltip left</button>

```

```
 5   <button type="button" class="btn btn-default"
 6           data-toggle="tooltip" data-placement="top"
 7           title="Tooltip on top">Tooltip top</button>
 8
 9   <button type="button" class="btn btn-default"
10           data-toggle="tooltip" data-placement="bottom"
11           title="Tooltip on bottom">Tooltip bottom</button>
12
13   <button type="button" class="btn btn-default"
14           data-toggle="tooltip" data-placement="right"
15           title="Tooltip on right">Tooltip right</button>
```

The activation always takes place in the script code. The following code activates all tooltips when the page loads.

```
1   $(function () {
2     $('[data-toggle="tooltip"]').tooltip()
3   })
```

For a tooltip that looks like this:

```
1   $('#example').tooltip(options)
```

The right to markup could be written as follows:

```
1   <a href="#" data-toggle="tooltip"
2      title="A tip will appear!"> Levitate mouse</a>
```

The markup generated is as follows:

```
1   <div class="tooltip top" role="tooltip">
2     <div class="tooltip-arrow"></div>
3     <div class="tooltip-inner">
4       A tip will appear!
5     </div>
6   </div>
```

Multiline Links

Sometimes the tips are somewhat more complex and have multiple lines. Normally, the text will automatically wrap and center. With `white-space: nowrap;` this upheaval is prevented. If such a tip in groups of buttons (with *.btn-group* or *.input-group*) is used, further measures are needed. In JavaScript, the container must be specified where the generated code is implanted in the DOM. In most, `container: 'body'` is sufficient.

General Tips

Tooltips for invisible elements are not a good idea. The tip is not correctly placed, because the position of the reference element cannot be determined. View the reference element only and release the tooltip After its parent element becomes visible.

Otherwise tooltips should only be used if the elements are being triggered by keystrokes. This can be achieved if you are limited to input elements. If this is not possible, use the attribute `tabindex` for an item to make it explicitly accessible.

If tooltips are visible on the deactivated elements, a wrapper must be built. It is best to add another `<div>` element and start the tooltip from there.

```
1   $('#myTooltip').on('hidden.bs.tooltip', function () {
2     // Do something...
3   })
```

Options

Options may be set as a `data-`attributes in HTML or JSON code. The suffix of the `data-`attributes corresponds to the name of the JSON property. Table 8-12 lists the options, Table 8-13 lists the actions, and Table 8-14 lists the events for tooltips.

Table 8-12. *Options for Tooltips*

Name	Type	Description
Animation container delay	Boolean string or `false` number, object	Animation on the appearance; positioning in another element, delay in milliseconds (default: 0), can also be an object `{ "show": 10, "hide": 10 }`, then apply for the ads and hide different values
html placement	Boolean string, function	HTML is allowed in the text “top,” “bottom,” “left,” “right,” “auto,” “auto” placed so that the tooltip is always displayed. For a function, they are determined by position itself
Selector	String	To select the destination as a selector
Template	String	Template in HTML
Title	String	Content (text) of tooltips
Trigger	String	Trigger: click, hover, focus, manual. Standard: “hover focus”
Viewport	String, object, function	Standard: `{ selector: 'body', padding: 0 }` or a selector or a selection function

The default template of the tooltip is as follows:

```
<div class="tooltip" role="tooltip">
  <div class="tooltip-arrow"></div>
  <div class="tooltip-inner"></div>
</div>
```

The text appears in *.tooltip-inner*.

Table 8-13. *Action for Tooltips*

Name	Description
Show	Show
Hide	Hide
Destroy	Suppress hiding and events

Table 8-14. *Events for Tooltips*

Name	Description
hide.bs.tab	Tooltip is hidden.
show.bs.tab	Tooltip is displayed.
hidden.bs.tab	Tooltip hidden (by animation)
shown.bs.tab	Tooltip displays (after animation)

Content Overlay (Popover)

The somewhat cumbersome term content overlay (popover) represents a brief appearance, but not a modal dialog, usually without interaction. The most important applications are giving detailed help. This is always appropriate when tooltips are no longer sufficient. Because many overlays are critical to the performance of the browser, they must be explicitly enabled—the attributes alone are not enough.

Content overlays can be static or dynamically activated. Static—constantly visible elements—serve design purposes usually.

Application

The popover feature requires an additional library, “tether.js”. There are also different styles available. See “tether.css” for details and themes, that are available in the same folder. Tether is a dependency that is automatically installed with Bootstrap 4. It's the very same as for tooltips.

It can be activated when the page loads with the following code:

```
$(function () {
  $('[data-toggle="popover"]').popover();
})
```

For a single item, this looks like this (the selector leads to the button):

```
$('#buttonid').popover({});
```

Depending on the content of a button, the application could be critical (see Listing 8-7 and Figure 8-4). Will the effect being bound to button groups (*.btn-group* or *.input-group*) the container element should appear in the `<body>`. It can otherwise result in side-effects (too broad, loss of rounded corners). It also is not recommended to enable content overlays on hidden elements. Bootstrap uses the coordinates of the triggering element for positioning and without the viewing area this may be inaccurate or invalid. In the deactivated elements you can use content overlays, however, than an enclosing container is required (usually a `<div>`).

If the triggering element is a hyperlink, then it can occur on narrow screens. Only then the content of the hyperlink is multiline and the content superimposition is both horizontal and vertical. This can be inconvenient because it interferes with the text of the triggering element. With the style `white-space: nowrap;` on the hyperlink that behavior can be avoided.

The general use as a help text for buttons is as follows:

```
<button type="button" class="btn btn-lg btn-danger"
        data-toggle="popover" title="The Title"
        data-content="Here is, for example, help">Help?</button>
```

Listing 8-7. Popover place(popover.html)

```
<button type="button" class="btn btn-default"
        data-container="body"
        data-toggle="popover"
        data-placement="left"
        data-content="This is a useful help text.">
  Placement left
</button>

<button type="button" class="btn btn-default"
        data-container="body"
        data-toggle="popover"
        data-placement="top"
        data-content="This is a useful help text.">
  Placement above
</button>

```

```
<button type="button" class="btn btn-default"
        data-container="body"
        data-toggle="popover"
        data-placement="bottom"
        data-content="This is a useful help text.">
  Placement below
</button>

<button type="button" class="btn btn-default"
        data-container="body"
        data-toggle="popover"
        data-placement="right"
        data-content="This is a useful help text.">
  Placement right
</button>
```

Figure 8-4. *Popover placement*

Small disturbing elements such as content overlays can be easily removed again. It is easiest for the user that the next action removes the element. The event "Focus" can accomplish this because it responds to every other element that gets focus. In addition the attributes `role="button"` and `tabindex` available.

```
<a tabindex="0" class="btn btn-lg btn-danger"
  role="button"
  data-toggle="popover"
  data-trigger="focus"
  title="Help is at hand"
  data-content="This is the help text!">
  Can be removed
</a>
```

 For this to always work, the trigger should be a hyperlink, not a button.

Options

Options may be set as `data-`attributes in HTML or JSON code. The suffix of the `data--` attributes corresponds to the name of the JSON property. Table 8-15 lists the options, Table 8-16 lists the actions, and Table 8-17 lists the events for popovers.

Table 8-15. *Options for Popovers*

Name	Type	Description
Animation	Boolean	Animation on appearance
Container	String or `false`	Positioning at another element
Delay	Number, object	Delay in milliseconds (Default: 0) can also be an object: `{ "show": 10, "hide": 10 }` then apply for ads and hides different values
HTML	Boolean	HTML is allowed in the text.
Content	String	Default content if the element delivers nothing
Placement	String, function	"Top," "bottom," "left," "right," "auto," "auto" placed so that the tooltip always displays. When can a function be the position itself determines.
Selector	String	To select the destination as a selector
Template	String	Template in HTML
Title	String	Content (text) of tooltips
Trigger	String	Trigger: click, hover, focus, manual. Standard: "hover focus"
Viewport	String, object, function	Standard: `{ selector: 'body', padding: 0 }` or a selector or a selection function

The default template of popover looks like this:

```
1  <div class="popover" role="tooltip">
2    <div class="arrow"></div>
3    <h3 class="popover-title"></h3>
4    <div class="popover-content"></div>
5  </div>
```

The text appears in *.popover-content*.

Table 8-16. *Actions for Popovers*

Name	Description
Show	Show
Hide	Hide
Toggle	Switch view state
Destroy	Suppress hiding and events

Table 8-17. *Events for Popovers*

Name	Description
hide.bs.popover	Popover is hiding
show.bs.popover	Popover appears
hidden.bs.popover	Popover was hidden (by animation)
shown.bs.popover	Popover was shown (by animation)
inserted.bs.popover	Return to "show," if the item is placed in the DOM

Message (Alert)

Appearance produces the same messages to *.alert* with drawn shapes. With the additional interaction you get the ability to hide the element (to close). The message itself also may include other actions. The disable function is created automatically when `data-dismiss="alert"` is added to the element that triggers the closing process.

A "close" button may look like this:

```
<button type="button"
        id="myAlert"
        class="close"
        data-dismiss="alert"
        aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
```

Activation is carried out using the method `alert()`:

```
$('#myAlert').alert();
```

It may react to closing:

```
$('#myAlert').on('closed.bs.alert', function () {
  // tu was...
})
```

Options

Options do not exist. The design is accomplished with HTML. Table 8-18 lists the actions and Table 8-19 lists the events for alerts.

Table 8-18. *Action for Alerts*

Name	Description
Close	Hide message

Table 8-19. *Events for Alerts*

Name	Description
close.bs.popover	Message is closed
closed.bs.popover	Message was closed (after animation)

Action Buttons (Button)

Action buttons respond to states and show them.

Toggle Button

With `data-toggle="button"` you get a button for the user to switch back and forth. It looks like a button, but behaves logically as a checkbox. If this switch should start the page active, it should have the class *.active* and carry the attribute `aria-pressed="true"`.

```
<button type="button" class="btn btn-primary"
        data-toggle="button"
        aria-pressed="false" autocomplete="off">
  Simple Switcher
</button>
```

By the same token, this also can be used for groups. Here `data-toggle="buttons"` is used on an element with the class *.btn-group*. The elements should then be of type `checkbox` or `radio`. The design can still be carried out as a button. This behaves as a part of many components and is less technical than using a number of checkboxes or radio buttons.

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="checkbox" autocomplete="off" checked> 1
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> 2
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> 3
  </label>
</div>
```

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="radio" name="options"
           id="option1" autocomplete="off" checked> 1
  </label>
```

```
    <label class="btn btn-primary">
      <input type="radio" name="options"
             id="option2" autocomplete="off"> 2
    </label>
    <label class="btn btn-primary">
      <input type="radio" name="options"
             id="option3" autocomplete="off"> 3
    </label>
  </div>
```

Options

Options do not exist. The design is done with HTML. Table 8-20 lists the action for buttons. The only possibility for change via code is to change the text on the button:

```
$('#myButton').button('New Text');
```

Table 8-20. *Action for Buttons*

Name	Description
Toggle	Toggle
Reset	Reset

Content Insertion (Collapse)

Similar content overlay does the content display (or content suppression, depending on how you look at it). In any case, this element is used to display only temporarily unnecessary content and then releases valuable space again.

Application

What is needed are first-inducing elements, which are either hyperlinks or buttons as shown in Listing 8-8:

Listing 8-8. Content insertion (collapse.html)

```
<a class="btn btn-primary"
    role="button"
    data-toggle="collapse"
    href="#collapseExample"
    aria-expanded="false"
    aria-controls="collapseExample">
  Via Link
</a>
```

```
 9   <button class="btn btn-outline-danger"
10           type="button"
11           data-toggle="collapse"
12           data-target="#collapseExample"
13           aria-expanded="false"
14           aria-controls="collapseExample">
15     Via Button
16   </button>
17   <div class="collapse" id="collapseExample">
18     <div class="card card-block card-text">
19       ...
20     </div>
21   </div>
```

The trigger to switch the group is `data-toggle="collapse"`. This either can be a link (with `href="#targetId"`) or a button (with `data-target="selector"`).

The lower portion of the message text is displayed only when one of the buttons has been clicked as shown in Figure 8-5.

Figure 8-5. *Content display*

The Aria support is valued internally by the script and should be complemented by matching attributes:

- `aria-expanded:` Indicates which group is open
- `aria-controls:` Indicates which group is controlled by link
- `aria-labelledby:` Indicates which head area refers to the group

Content Groups—The Accordion

The accordion is a frequently used element that offers many design frameworks. Technically, this is a group of navigation elements and dynamic panels, which are displayed exclusively in each case.

The individual components have all been introduced. In Bootstrap the accordion is not a stand-alone component, but a combination of basic building blocks.

First, look at the example in Listing 8-9.

Listing 8-9. Content groups (accordion.html)

```
<div id="accordion" role="tablist" aria-multiselectable="true">
  <div class="card card-success"
       role="tab">
    <h4 class="card-header"
        id="headingOne">
      <a href="#collapseOne"
         data-toggle="collapse"
         data-parent="#accordion"
         aria-expanded="true"
         aria-controls="collapseOne">
      Group 1</a>
    </h4>
    <div id="collapseOne"
         class="collapse in"
         role="tabpanel"
         aria-labelledby="headingOne">
      <div class="card-block card-text">
        A lot of text in group 1.
      </div>
    </div>
  </div>
  <div class="card"
       role="tab">
    <h4 class="card-header"
        id="headingTwo">
      <a href="#collapseTwo"
         data-toggle="collapse"
         data-parent="#accordion"
         aria-expanded="false"
         aria-controls="collapseTwo">Group 2</a>
    </h4>
    <div id="collapseTwo"
         class="collapse"
         role="tabpanel"
         aria-labelledby="headingTwo">
      <div class="card-block card-text">
        A lot of text in group 2.
      </div>
    </div>
  </div>
  <div class="card"
       role="tab">
    <h4 class="card-header"
        id="headingThree">
      <a href="#collapseThree"
         data-toggle="collapse"
```

```
47            data-parent="#accordion"
48            aria-expanded="false"
49            aria-controls="collapseThree">
50          Group 3</a>
51        </h4>
52        <div id="collapseThree"
53            class="collapse"
54            role="tabpanel"
55            aria-labelledby="headingThree">
56          <div class="card-block card-text">
57            A lot of text in group 3.
58          </div>
59        </div>
60      </div>
61    </div>
```

The code uses the `data-`attributes and design of the in Bootstrap 4 newly introduced *.card* class. The trigger for the switching of the group is `data-toggle="collapse`. This can either be a link (with `href="#targetId"`) or a button (with `data-target="Selector"`). So when you open a group, the group that is already open closes automatically, which is `data-parent="id"` and uses the ID that refers to the surrounding container element.

The initial state is determined by the class *.in*. Such decorated elements are open on page load as shown in Figure 8-6.

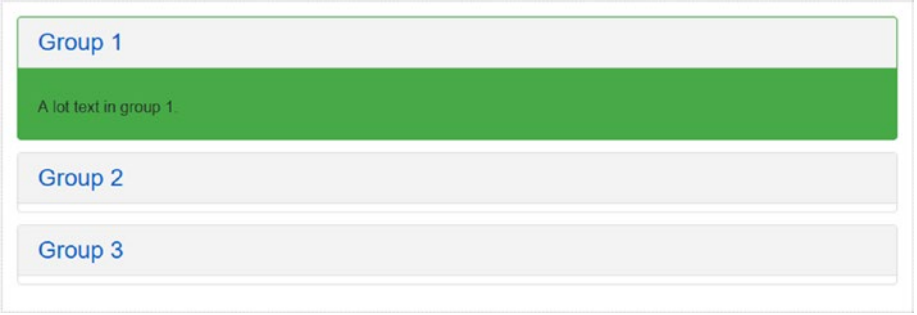


Figure 8-6. *Contents display*

Activating the code—if `data-`attributes are not used—is as follows:

```
1   $('.collapse').collapse();
```


Use either `data--`attributes or JavaScript. If both are used simultaneously, the script gets confused and the accordion behaves illogically.

Options

Options may be set as a `data--`attributes in HTML or JSON code.

The suffix of the `data--`attributes corresponds to the name of the JSON property. Table 8-21 lists the options, Table 8-22 lists the actions, and Table 8-23 lists the events for collapse.

Table 8-21. *Options for Collapse*

Name	Type	Description
parent	String	Selector for the parent element automatic switch
toggle	Boolean	Toggles the collapsable element on invocation

Table 8-22. *Action for Collapse*

Name	Description
Show	Show the element
Hide	Hide the element
Toggle	Switch view state

Table 8-23. *Events for Collapse*

Name	Description
hide.bs.collapse	Content is hidden
show.bs.collapse	Content is displayed
hidden.bs.collapse	Contents were hidden (by animation)
shown.bs.collapse	Content has been displayed (after animation)

Image Roundabout (Carousel)

The gyro images (often called carousel) are used to display a picture or content path, which has always just one element that is currently visible.

Application

Listing 8-10 demonstrates an example (see Figure 8-7 for a visual image):

Listing 8-10. Image roundabout (caroussel.html)

```
<div class="container sample" id="example">
<div id="carousel-example-generic"
        class="carousel slide" data-ride="carousel">
    <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0"
    class="active"></li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></li>
    </ol>

    <div class="carousel-inner" role="listbox">
    <div class="carousel-item active text-md-center">
       <img src="../Res/css3.png">
       <div class="carousel-caption">
       CSS 3
       </div>
    </div>
    <div class="carousel-item text-md-center">
       <img src="../Res/html5.png">
       <div class="carousel-caption">
        HTML 5
       </div>
    </div>
    <div class="carousel-item text-md-center">
       <img src="../Res/es6.png">
       <div class="carousel-caption">
       ES 6
       </div>
    </div>
    </div>
    <a class="left carousel-control"
       href="#carousel-example-generic"
       role="button"
       data-slide="prev">
    <span class="fa fa-chevron-left"
          aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
    </a>
    <a class="right carousel-control"
       href="#carousel-example-generic"
       role="button"
       data-slide="next">
```

```
41      <span class="fa fa-chevron-right"
42              aria-hidden="true"></span>
43      <span class="sr-only">Next</span>
44      </a>
45    </div>
46    </div>
```

Figure 8-7. *An image of the gyro images*

The activation code is as follows:

```
1   $('.carousel').carousel();
```

 This component is not suitable for accessible environments.

Options

Options may be set as a `data-`attributes in HTML or JSON code. The suffix of the `data-`attributes corresponds to the name of the JSON property. Table 8-24 lists the options, Table 8-25 lists the actions, and Table 8-26 lists the events for carousels.

Table 8-24. *Options for Carousel*

Name	Type	Description
Interval	Number	Time until the next element, default is 5,000 ms
Pause	String	Stops the indexing, as long as the mouse is over the element
Wrap	Boolean	Continuous indexing or stepwise
Keyboard	Boolean	Responds to the keyboard

Table 8-25. *Actions for Carousel*

Name	Description
Cycle	Change to the next item
Pause	Stop
Prev	Previous entry
Next	Next entry

Table 8-26. *Events for Carousel*

Name	Description
slide.bs.carousel	Start a movement
slid.bs.carousel	End a movement

Index

J. Krause, *Introducing Bootstrap 4*, DOI 10.1007/978-1-4842-2382-6

D, E

F

G

H

I, J, K

L

M

N, O

P, Q

R

S

T

U

V, W, X, Y, Z