**Basic Questions**

1. **SASS**
2. **HTML5**
3. **CSS3**
4. **Difference between Bootstrap 3 and 4**
5. **GIT**

**SASS**

1. Explain what is SASS? How it can be used?
   1. SASS stands for syntactically awesome StyleSheets and was created by Hampton Catlin. It is an extension of CSS3, adding nested rules, mixins, variables, selector inheritance, etc.
   2. SASS can be used in three ways
      1. As a command line tool
      2. As a standalone ruby module
      3. As a plugin for any rack-enabled framework
2. List out the key features for SASS?
   1. Key features for SASS included
      1. Full CSS3-compatible
      2. Language extensions such as nesting, variables, and mixins
      3. Many useful functions for manipulating colors and other values
      4. Advanced features like control directives for libraries
      5. Well-formatted, customizable output
3. List out the data types that SASSScript supports?
   1. SASSScript supports seven main data types
      1. Numbers
      2. Strings of texts
      3. Colors
      4. Booleans
      5. Nulls
      6. List of values, separated by space or commas
      7. Maps from one value to another
4. Explain how to define a variable in SASS?
   1. Variables in SASS begin with a “$” sign
   2. Variable assignment is done with “:” colon.
5. Explain what is the difference between SASS and SCSS?
   1. The difference between SASS and SCSS is that,
      1. SASS is a CSS pre-processor with syntax advancements and an extension of CSS3
      2. SASS has two syntax
      3. The first syntax is “SCSS” and it uses the .SCSS extension
      4. While SASS has loose syntax with white space and no semicolons, the SCSS resembles more to CSS
      5. Any CSS valid document can be converted to SASS by simply changing the extension from .CSS to .SCSS.
6. What selector nesting in SASS is used for?
   1. In SASS, selector nesting offers a way for StyleSheet authors to compute long selectors by nesting shorter selectors within each other.
7. Explain what is a @extend function used for in SASS?
   1. In SASS, the $extend directive provides a simple way to allow a selector to inherit the styles of another one. It aims at providing a way for a selector A to extend the styles from a selector B. When doing so, the selector A will be added to selector B so they both share the same declarations. @extend prevents code bloat by grouping selectors that share the same style into one rule.
8. Explain what is the use of the @import function in SASS?
   1. Extends the CSS import rule by enabling import of SCSS and SASS files.
   2. All imported files are merged into a single outputted CSS file.
   3. Can virtually mix and match any file and be certain of all your styles
   4. @import takes a filename to import
9. Why SASS is considered better than less?
   1. SASS allows you to write reusable methods and logic statements, e., loops, and conditionals
   2. SASS user can access compass library and use some awesome features like dynamic sprite map generation, legacy browser hacks and cross-browser support for CSS3 features
   3. Compass also allows you to add an external framework like blueprint, foundation or Bootstrap on top
   4. In less, you can write basic logic statement using “guarded mixin”, which is equivalent to SASS if statements
   5. In less, you can loop through numeric values using recursive functions while SASS allows you to iterate any kind of data.
   6. In SASS you can write your own handy functions
10. Explain what is the use of mixin function in SASS? What is the meaning of DRY-ing out a mixin?
    1. Mixin allows you to define styles that can be re-used throughout the StyleSheet without to resort to non-semantic classes like .float-left.
    2. DRY-ing out of a mixing means splitting it into dynamic and static parts. The dynamic mixin is the one that the user actually going to call, and the static mixin is the pieces of information that would otherwise get duplicated.
11. Explain what SASS maps is and what is the use of SASS maps?
    1. Sass map is a structured data in a hierarchical way and not just a bunch of variables. It can help in organizing the code. Some great use of Sass are
       1. It is very useful when dealing with layers of elements in project.
       2. It can be helpful in color management when there is long list of different color and shade
       3. Use icon map for various social media icons for example: Facebook: ‘e607’ or Twitter: ‘e602’
       4. Unlike other programming libraries, SASS map will consist only of code that is going to be used
12. Explain how SASS comments are different from regular CSS?
    1. Syntax for comments in regular CSS starts wit /\* comments…. \*/, while in SASS there are two type of comment, the single line comments “//” and the multiline CSS comments with “/\*\*/”.
13. Does SASS supports inline comments?
    1. Single line comments “//” will be removed by the .SCSS pre-processor and won’t appear in your .CSS file.
    2. While the comment “\*/” are valid CSS, and will be preserved between the translation from .SCSS to your .CSS file.
14. How interpolation is used in SASS?
    1. In SASS, you can define an element in a variable and interpolate it inside the SASS code. It is useful when you keep your modules in separate files.
15. Explain when can you use the %placeholders in SASS?
    1. %placeholders in SASS is useful when you want to write styles that were meant to be extended, but you don’t want the base styles to be seen in output CSS styles.
16. Is it possible to nest variables within variables in SASS?
    1. Interpolation of variables names is not possible currently in SASS. However, you may use interpolation of placeholders.
17. What are SASS cons and pros?
    1. Pros:
       1. SASS is easy to learn especially for them who has a background of python, ruby or Coffescript and place using functions, writing mixins
       2. CSS can be easily converted to SASS
       3. Throughout the project, you don’t have to repeat similar CSS statements using @extend attribute
       4. It allows to define variables that are usable throughout the entire project
       5. It keeps your responsive project more organized
    2. Cons:
       1. White space sensitive
       2. No inline rules

References

<http://career.guru99.com/top-25-bootstrap-interview-questions/>

<http://career.guru99.com/top-17-sass-interview-questions/>

<http://career.guru99.com/>

**HTML5 Basic Questions**

1. **What is HTML5?**

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

1. **Name some of the new features of HTML5.**

HTML5 introduces a number of new elements and attributes that helps in building a modern websites. Following are great features introduced in HTML5

* 1. **New semantic elements :** <header>, <footer> and <section>
  2. **Forms 2.0:** improvements to HTML web forms where new attributes have been introduced for <input> tag.
  3. **Persistent local storage :** to achieve without resorting to third party plugins.
  4. **Web socket :** a next generation bidirectional communication technology for web applications.
  5. **Server-sent events :** HTML5 introduces events which flow from web server to the web browsers and they are called server-sent events (SSE).
  6. **Canvas :** this supports a two-dimensional drawing surface that you can program with JavaScript.
  7. **Audio & Video :** you can embed audio or video on your web pages without resorting to third-party plugins.
  8. **Geolocation :** now visitors can choose to share their physical location with your web applications.
  9. **Micro data :** this lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
  10. **Drag and Drop :** drag and drop the items from one location to another location on a the same webpage.

1. **Which browsers support HTML5?**

The latest versions of apple safari, google chrome, Mozilla Firefox, and opera all support many HTML5 features and internet explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and android phones all have excellent support for HTML5.

1. **Is HTML5 backward compatible with old browsers?**

Yes! HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. New features build on existing features and allow you to provide fallback content for older browsers.

It is suggested to detect support for individual HTML5 features using a few lines of JavaScript.

1. **Are HTML tags case sensitive?**

No!

1. **What is the purpose of 'section' tag in HTML5?**

This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.

1. **What is the purpose of ‘article’ tag in HTML5?**

This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.

1. **What is the purpose of ‘aside’ tag in HTML5?**

This tag represents a piece of content that is only slightly related to the rest of the page.

1. **What is the purpose of 'header' tag in HTML5?**

This tag represents the header of a section.

1. **What is the purpose of 'footer' tag in HTML5?**

This tag represents a footer for a section and can contain information about the author, copyright information, et cetera.

1. **What is the purpose of 'nav' tag in HTML5?**

This tag represents a section of the document intended for navigation.

1. **What is the purpose of 'dialog' tag in HTML5?**

This tag can be used to mark up a conversation.

1. **What is the purpose of 'figure' tag in HTML5?**

This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

1. **What are custom attributes in HTML5?**

A custom data attribute starts with data- and would be named based on your requirement. Following is the simple example –

<div class="example" data-subject="physics" data-level="complex">

...  
</div>

The above will be perfectly valid HTML5 with two custom attributes called data-subject and data-level. You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

1. **What is Web Forms 2.0?**

Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and remove a great deal of the need for tedious scripting and styling that was required in HTML4.

1. **What is the purpose of datetime input control in Web form 2.0?**

It represents a date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601 with the time zone set to UTC.

1. **What is the purpose of datetime-local input control in Web form 2.0?**

It represents a date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601 with no time zone information.

1. **What is the purpose of date input control in Web form 2.0?**

It represents a date (year, month, day) encoded according to ISO 8601.

1. **What is the purpose of month input control in Web form 2.0?**

It represents a date consisting of a year and a month encoded according to ISO 8601.

1. **What is the purpose of week input control in Web form 2.0?**

It represents a date consisting of a year and a week number encoded according to ISO 8601.

1. **What is the purpose of time input control in Web form 2.0?**

It represents a time (hour, minute, seconds, fractional seconds) encoded according to ISO 8601.

1. **What is the purpose of number input control in Web form 2.0?**

This control accepts only numerical value. The step attribute specifies the precision, defaulting to 1.

1. **What is the purpose of range input control in Web form 2.0?**

The range type is used for input fields that should contain a value from a range of numbers.

1. **What is the purpose of email input control in Web form 2.0?**

This accepts only email value. This type is used for input fields that should contain an e-mail address. If you try to submit a simple text, it forces to enter only email address in email@example.com format.

1. **What is the purpose of url input control in Web form 2.0?**

This accepts only URL value. This type is used for input fields that should contain a URL address. If you try to submit a simple text, it forces to enter only URL address either in http://www.example.com format or in http://example.com format.

1. **What is the purpose of 'output' tag in HTML5?**

HTML5 introduced a new element &lt;output&gt; which is used to represent the result of different types of output, such as output written by a script.

1. **What is the purpose of 'placeholder' attribute in HTML5?**

HTML5 introduced a new attribute called placeholder. This attribute on &lt;input&gt; and &lt;textarea&gt; elements provides a hint to the user of what can be entered in the field. The placeholder text must not contain carriage returns or line-feeds.

1. **What is the purpose of 'autofocus' attribute in HTML5?**

This is a simple one-step pattern, easily programmed in JavaScript at the time of document load, automatically focus one particular form field.

1. **What is the purpose of 'required' attribute in HTML5?**

HTML5 introduced a new attribute called required which would insist to have a value in an input control.

1. Can you use SVG tags directly in HTML5 without any plugin?

Yes! HTML5 allows embeding SVG directly using &lt;svg&gt;...&lt;/svg&gt; tag.

1. **Can you use MathML tags directly in HTML5 without any plugin?**

Yes! The HTML syntax of HTML5 allows for MathML elements to be used inside a document using &lt;math&gt;...&lt;/math&gt; tags.

1. **What are the drawbacks of cookies?**

*Cookies have following drawbacks-*

* 1. Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
  2. Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
  3. Cookies are limited to about 4 KB of data . Not enough to store required data.

1. **What do you mean by session storage in HTML5?**

HTML5 introduces the sessionStorage attribute which would be used by the sites to add data to the session storage, and it will be accessible to any page from the same site opened in that window i.e. session and as soon as you close the window, session would be lost.

1. **What do you mean by local storage in HTML5?**

HTML5 introduces the localStorage attribute which would be used to access a page's local storage area without no time limit and this local storage will be available whenever you would use that page.

1. **When a session storage data gets deleted?**

The Session Storage Data would be deleted by the browsers immediately after the session gets terminated.

1. **When a local storage data gets deleted?**

local storage data has no time limit. To clear a local storage setting you would need to call localStorage.remove('key'); where 'key' is the key of the value you want to remove. If you want to clear all settings, you need to call localStorage.clear() method.

1. **What is Server Side Events in HTML5?**

Along with HTML5, WHATWG Web Applications 1.0 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE). Using SSE you can push DOM events continously from your web server to the visitor's browser.

The event streaming approach opens a persistent connection to the server, sending data to the client when new information is available, eliminating the need for continuous polling.

Server-sent events standardizes how we stream data from the server to the client.

1. **How to utilize a server-sent event in HTML5?**

To use Server-Sent Events in a web application, you would need to add an &lt;eventsource&gt; element to the document.

The src attribute of &lt;eventsource&gt; element should point to an URL which should provide a persistent HTTP connection that sends a data stream containing the events.

The URL would point to a PHP, PERL or any Python script which would take care of sending event data consistently.

1. **What are the steps of server side scripts for SSE?**

Server side script should send Content-type header specifying the type text/event-stream as follows-

print "Content-Type: text/event-stream\n\n";

After setting Content-Type, server side script would send an Event - tag followed by event name. Following example would send Server-Time as event name terminated by a new line character.

print "Event: server-time\n";

Final step is to send event data using Data - tag which would be followed by integer of string value terminated by a new line character as follows-

$time = localtime();

print "Data: $time\n";

1. **What are web sockets?**

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage event handler.

Following is the API which creates a new WebSocket object.

var Socket = new WebSocket(url, [protocal] );

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

1. **What is the purpose of Socket.readyState atribute of WebSocket?**

The readonly attribute readyState represents the state of the connection. It can have the following values:

* 1. A value of 0 indicates that the connection has not yet been established.
  2. A value of 1 indicates that the connection is established and communication is possible.
  3. A value of 2 indicates that the connection is going through the closing handshake.
  4. A value of 3 indicates that the connection has been closed or could not be opened.

1. **What is the purpose of Socket.bufferedAmount atribute of WebSocket?** **What is the purpose of Socket.bufferedAmount atribute of WebSocket?**

The readonly attribute bufferedAmount represents the number of bytes of UTF-8 text that have been queued using send() method.

1. **What is the purpose of 'canvas' tag in HTML5?**

HTML5 element &lt;canvas&gt; gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

1. **What is the purpose of 'audio' tag in HTML5?**

HTML5 supports &lt;audio&gt; tag which is used to embed sound content in an HTML or XHTML document. The current HTML5 draft specification does not specify which audio formats browsers should support in the audio tag. But most commonly used audio formats are ogg, mp3 and wav.

You can use &lt;source&gt; tag to specify media along with media type and many other attributes. An audio element allows multiple source elements and browser will use the first recognized format.

1. **What is the purpose of 'video' tag in HTML5?**

HTML5 supports &lt;video&gt; tag which is used to embed a video file in an HTML or XHTML document.The current HTML5 draft specification does not specify which video formats browsers should support in the video tag. But most commonly used video formats are-

* 1. **ogg-** Ogg files with Thedora video codec and Vorbis audio codec.
  2. **Mpeg4 -** MPEG4 files with H.264 video codec and AAC audio codec.

You can use &lt;source&gt; tag to specify media along with media type and many other attributes. An audio element allows multiple source elements and browser will use the first recognized format.

1. **What is Geolocation API in HTML?**

HTML5 Geolocation API lets you share your location with your favorite web sites. A Javascript can capture your latitude and longitude and can be sent to backend web server and do fancy location-aware things like finding local businesses or showing your location on a map.

Today most of the browsers and mobile devices support Geolocation API. The geolocation APIs work with a new property of the global navigator object ie. Geolocation object which can be created as follows:

var geolocation = navigator.geolocation;

The geolocation object is a service object that allows widgets to retrieve information about the geographic location of the device.

1. **What is purpose of getCurrentPosition() method of geolocation object of HTML5?**

This method retrieves the current geographic location of the user.

1. **What is purpose of watchPosition() method of geolocation object of HTML5?**

This method retrieves periodic updates about the current geographic location of the device.

1. **What is purpose of clearPosition() method of geolocation object of HTML5?**

This method cancels an ongoing watchPosition call.

1. **What are Web Workers?**

Web Workers do all the computationally expensive tasks without interrupting the user interface and typically run on separate threads.

Web Workers allow for long-running scripts that are not interrupted by scripts that respond to clicks or other user interactions, and allows long tasks to be executed without yielding to keep the page responsive.

<https://www.toptal.com/html5/interview-questions>

<https://www.tutorialspoint.com/html5/html5_interview_questions.htm>

<http://www.skilledup.com/articles/html-html5-interview-questions-answers>

<https://www.codeproject.com/Articles/702051/important-HTML-Interview-questions-with-answe>

**CSS3**

The full form of CSS is cascading style sheet. It is a styling language which is simple enough for HTML elements. It is popular in web designing, and its application is common in XHTML also.

1. **Explain what a class selector is and how it’s used**
   1. A class can be thought of as a grouped collection of CSS attributes applied to HTML elements. This allows you to apply the same styling to multiple HTML elements.
   2. Class methods can be called by inserting a ‘class’ property and name within an HTML element, then calling the class name with ‘.’ In the CSS document.
2. **What are pseudo classes and what are they used for?**
   1. Pseudo classes are similar to classes, but are not explicitly defined in the markup, and are used to add additional effects to selected HTML elements such as link colors, hover actions, etc.
   2. Pseudo classes are defined by first listing the selector, followed by a colon and then pseudo-class element. E.g. a:link {color: red}
3. **Explain the ways to apply CSS style to a web page:**
   1. Inline: though this method often goes against best practices, it’s easily done by inserting a ‘style’ attribute inside an HTML element.
   2. Embedded/internal: done by defining the head of an HTML document by wrapping characteristics in a <style> tag.
   3. Linked/External: CSS is placed in an external .CSS file, and linked to the HTML document with a <link> tag. This can also be accomplished using the ‘@import’, however, this can slow page load time and is generally not advised.
4. **What is grouping and what is it used for?**
   1. Grouping allows you to apply the same style to multiple elements with a single declaration. This is done by grouping the selectors into a list, separated by comas.
   2. Grouping helps memory usage and enhances readability.
5. **What is an id selector and how is it used?**
   1. IDs are used to identify and apply styling to a single specific HTML element. IDs are defined within the HTML page by inserting and ID selector in the HTML element.
   2. ID selectors are defined within the CSS page by calling a “#” followed by the name of the ID:
   3. ID selector are unique and can only be applied a single element.
6. **What is a class selector and how does it differ from an ID selector?**
   1. Class selectors are used to apply style to multiple HTML identified with the same class.
   2. Class selectors are called within the CSS document by a “.” Followed by the class name:
   3. The main difference is that the same class selector can be applied to multiple HTML elements, whereas ID selectors are unique.
7. **What are child selectors?**
   1. Child selectors are another way to group and style a set of elements that descend from a parent element.
   2. A child selector is matched by calling two or more elements, separated by a “>” sign to indicate inheritance.
8. **What are the different CSS properties used to change dimensions and what values can they accept?**
   1. height and width: sets a specific height or width
      1. auto, length, %, inherit
   2. max-height and max-width: sets a maximum height
      1. auto, length, %, inherit
   3. min-height and min-width: sets a minimum height
      1. auto, length, %, inherit

<http://www.skilledup.com/articles/25-css-interview-questions-answers>

<https://www.toptal.com/css/interview-questions>

<http://career.guru99.com/top-50-csscascading-style-sheet-interview-questions/>

**Difference between Bootstrap 3 and 4**

Bootstrap 44 is a major rewrite of almost the entire project. The most notable changes are summarized immediately below, followed by more specific class and behavioral changes to relevant components.

1. Browser support
   1. Dropped ie8, ie9 and iOS 6 support. V4 is now only IE10+ and iOS7+. For sites needing either of those, use v3.
2. Global changes
   1. Flexbox is enabled by default. In general this means a move away from floats and more across our components.
   2. Switched from LESS to SASS for our source CSS file.
   3. Switched from “px” to “rem” as our primary CSS unit, though pixels are still used for media queries and grid behavior as viewports are not affected by type size.
   4. Global font-size increased from “14px” to “16px”.
   5. Added new grid tier for smaller device at “576px” and below (our new “xs” tier).
   6. Replaced the separate optional theme with configurable options via SCSS variables (e.g. “$enable-gradients: true”)
3. Grid system
   1. Added support for flexbox in the grid mixins and predefined classes.
   2. As part of flexbox, included support for vertical and horizontal alignment classes.
   3. Overhauled grid mixins to merge “make-col-span” into “make-col” for a singular mixin.
   4. Added a new “sm” grid tier below “768px” for mew granular control. We now have **xs, sm, md, lg** and **XL.** This also means every tier has been bumped up one level (so .col-md-6 in v3 is now .col-lg-6 in v4).
   5. Changed grid system media query breakpoints and container widths to account for new grid tier and ensure columns are evenly divisible by 12 at their max width.
   6. Grid breakpoints and container widths are now handled via SASS maps ($grid-breakpoints and $container-max-widths) instead of a handful of separate variables. These replace the “#screen-\*” variables entirely and allow you to fully customize the grid tiers.
   7. Media queries have also changed. Instead of repeating our media query declarations with the same value each time, we now have “@include media-breakpoint-up/down/only”. Now, instead of writing @media (min-width: @screen-sm-min) {…}, you can write @include media-breakpoint-up(sm) {…}.
4. Components
   1. Dropped panels, thumbnails, and wells for a new all-encompassing component, cards.
   2. Dropped the Glyphicons icon font. If you need icons, some options are:
      1. The upstream version of **Glyphicons**
      2. **Octicons**
      3. **Font awesome**
   3. Dropped the affix jQuery plugin. We recommend using a position: sticky polyfill instead. [See the HTML5 Please entry](http://html5please.com/#sticky) for details and specific polyfill recommendations.
      1. If you were using affix to apply additional, non-position styles, the polyfill might not support your use case. One option for such uses is the third-party scrollpos-styler library.
   4. Dropped the pager component as it was essentially slightly customized buttons.
   5. Refactored nearly all components to use more un-nested classes instead of children selectors.
5. MISC
   1. Non-responsive usage of Bootstrap is no longer supported.
   2. Dropped the online customizer in favor of more extensive setup documentation and customized builds.
6. By component
   1. This list highlights key changes by component between v3.x.x and v4.0.0.
7. Reboot
   1. New to Bootstrap 4 is the reboot, a new StyleSheet that builds on normalize with our own somewhat opinionated reset styles? Selectors appearing in this file only use elements-there are no classes here. This isolates our reset styles from our component styles for a more modular approach. Some of the most important resets this includes are the box-sizing: border-box change, moving from em to rem units on many elements, link styles, and many form element resets.
8. Typography
   1. Moved all .text- utilities to the \_utilities.scss file.
   2. Dropped .page-header as, aside from the border, all its styles can be applied via utilities.
   3. .dl-horizontal has been dropped. Instead, use .row on <dl> and use grid column classes (or mixins) on its <DT> and <dd> children.
   4. Custom <blockquote> styling has moved to classes -- .blockquote and the .blockquote-reserve modifier.
   5. .list-inline now requires that its children list items have the new .list-inline-item class applied to them.
9. Images
   1. Renamed .img-responsive to .img-fluid.
   2. Renamed .img-rounded to .rounded.
   3. Renamed .img-circle to .rounded-circle.
10. Tables
    1. Nearly all instances of the > selector have been removed, meaning nested tables will now automatically inherit styles from their parents. This greatly simplifies our selectors and potential customizations.
    2. Responsive tables no longer require a wrapping element. Instead, just put the .table-responsive right on the <table>
    3. Renamed .table-condensed to .table-sm for consistency.
    4. Added a new .table-inverse option.
    5. Added table header modifiers: .thead-default and .thead-inverse.
    6. Renamed contextual classes to have a .table- -prefix. Hence .active, .success, .warning, .danger and .table-info to .table-active, .table-success, .table-warning, .table-danger and .table-info.
11. Forms
    1. Moved element resets to the \_reboot.scss file.
    2. Renamed .control-label to .form-control-label.
    3. Renamed .input-lg and .input-sm to .form-control-lg and .form-control-sm, respectively.
    4. Dropped .form-group-\* classes for simplicity’s sake. Use .form-control-\* classes instead now.
    5. Dropped .help-block and replaced it with .form-text for block-level help text. For inline help text and other flexible options, use utility classes like .text-muted.
    6. Horizontal forms overhauled:
       1. Dropped the .form-horizontal class requirement.
       2. .form-group no longer applies styles from the .row via mixin, so .row is now required for horizontal grid layouts (e.g. <div class=”form-group row”>).
       3. Added new .form-control-label class to vertically center labels with .form-controls.
       4. Added custom forms support (form checkboxes, radios, selects and file inputs).
12. Buttons
    1. Renamed .btn-default to .btn-secondary
    2. Dropped the .btn-xs class entirely as .btn-sm is proportionally much smaller than v3’s
    3. The stateful button feature of the button.js jQuery plugin has been dropped. This includes the $().button (string) and $().button (‘reset’) methods. We advise using a tiny bit of custom JavaScript instead, which will have the benefit of behaving exactly the way you want it to.
       1. Note that the other features of the plugin (button checkboxes, button radios, and single-toggle buttons) have been retained in v4.
13. Button group
    1. Rewrote component with flexbox.
    2. Dropped the .btn-group-xs class entirely given removal of .btn-xs.
    3. Removed explicit spacing between button groups in button toolbars; use margin utilities now.
    4. Improved documentation for use with other components.
14. Dropdowns
    1. Switched from parent selectors to singular classes for all components, modifiers, etc.
    2. Simplified dropdown styles to no longer ship with upward or downward facing arrows attached to the dropdown menu.
    3. Dropdowns can be built with <div> or <ul> now.
    4. Rebuilt dropdown styles and markup to provide easy, built-in support for <a> and <button> based dropdown items.
    5. Renamed .divider to .dropdown-divider.
    6. Dropdown items now require .dropdown-item.
    7. Dropdown toggles no longer require an explicit <span class=”caret”></span>; this is now provided automatically via CSS’s: after on .dropdown-toggle.
15. Grid system
    1. Added a new 576px grid breakpoint as sm. meaning there are now five total tiers (xs, sm, md, lg and XL).
    2. Renamed the responsive grid modifier classes from .col-{breakpoint}-{modifier}-{size} to. {modifier}- {Breakpoint}-{size} for simpler grid classes. For example, instead of .col-md-3.col-md-push-9 it’s .col-md-3.push-md-9
    3. Overhauled the grid mixins to merge make-col and make-col-span into a single make-col mixin, thereby ensuring mixins and predefined classes utilize the same float/flex behaviors.
    4. Added flexbox utility classes for grid system components.
16. List groups
    1. Rewrote component with flexbox.
    2. Replaced a.list-group-item with an explicit class, .list-group-item-action, for styling link and button versions of list group items.
    3. Added .list-group-flush class for use with cards.
17. Modal
    1. Rewrote component with flexbox.
    2. Given move to flexbox, alignment of dismiss icons in the header is likely broken as we’re no longer using floats. Floated contents comes first, but with flexbox that’s no longer the case. Update your dismiss icons to come after modal titles to fix.
    3. The remote option (which could be used to automatically load and inject external content into a modal) and the corresponding loaded.bs.modal event were removed. We recommend instead using client-side tinplating or a data binding framework, or calling jQuery.load yourself.
18. Navs
    1. Rewrote component with flexbox.
    2. Dropped nearly all > selectors for simpler styling via un-nested classes.
    3. Instead of HTML-specific selectors like .nav>li>A we used separate classes for .nav, .nav-item, and .nav-link. This makes your HTML more flexible while bringing along increased extensibility.
19. Navbar
    1. The navbar has been entirely rewritten in flexbox with improved support for alignment, responsiveness and customization.
    2. Responsive navbar behaviors are now applied to the .navbar class via the required .navbar-toggleable-{breakpoint} where you choose where to collapse the navbar. Previously this was a LESS variable modification and required recompiling.
    3. .navbar-default is now .navbar-light, though .navbar-inverts remains the same. One of these is required on each navbar. However, these classes no longer set background-color; instead they essentially only affect color.
    4. Navbars now require background declaration of some kind. Choose from our background utilities (.bg-\*) or set your own with the light/inverse classes above for mad customization.
    5. Given flexbox styles navbars can now use flexbox utilities for easy alignment options.
    6. .navbar-toggle is now .navbar-toggler and has different styles and inner markup (no more three <span>).
    7. Dropped the .navbar-form class entirely. It’s no longer necessary; instead, just use .form-inine and apply margin utilities as necessary.
    8. Navbars no longer include margin-bottom or border-radius by default. Use utilities as necessary.
    9. All examples featuring navbars have been updated to include new markup.
20. Pagination
    1. Rewrote component with flexbox.
    2. Explicit classes (.page-item, .page-link) are now required on the descendants of .pagination
    3. Dropped the .pager component entirely as it was little more than customized outline buttons.
21. Breadcrumbs
    1. An explicit class, .breadcrumb-item, is now required on the descendants of .breadcrumb
22. Labels, badges, and tags
    1. Renamed .label to .tag to disambiguate from the <label> element.
    2. Dropped the .badge component as it was nearly identical to labels/tags. Use the .tab-pill modifier together with the label component instead for that rounded look.
    3. Tags are no longer floated automatically in list groups and other components. Utility classes are now required for that.
23. Panels, thumbnails, and wells
    1. Dropped entirely for the new card component.
24. Panels
    1. .panel to .card, now built with flexbox.
    2. .panel-default removed and no replacement.
    3. .panel-group removed and no replacement. .card-group is not a replacement, it is different.
    4. .panel-heading to .card-header
    5. .panel-title to .card-title. Depending on the desired look, you may also want to use heading elements or classes (e.g. <h3>, .h3) or bold elements or classes (e.g. <strong>, <b>, .font-weight-bold). Note that .card-title, while similarly named, produces a different look than .panel-title.
    6. .panel-body to .card-block
    7. .panel-footer to .card-footer
    8. .panel-primary to .card-primary and .card-inverse (or use .bg-primary on .card-header).
    9. .panel-success to .card-success and .card-inverse (or use .bg-success on .card-header).
    10. .panel-info to .card-info and .card-inverse (or use .bg-info on .card-header).
    11. .panel-warning to .card-warning and .card-inverse (or use .bg-warning on .card-header).
    12. .panel-danger to .card-danger and .card-inverse (or use .bg-danger on .card-header).
25. Tooltips
    1. Removed support for “auto” placement options.
26. Popovers
    1. Removed support for auto placement options.
    2. Carousel
27. Carousel
    1. Overhauled the entire component to simplify design and styling. We have fewer styles for you to override, new indicators, and new icons.
    2. All CSS has been un-nested and renamed, ensuring each class is prefixed with .carousel-.
       1. For carousel items, .next, .prev, .left and .right are now .carousel-item-next, .carousel-item-prev, .carousel-item-left and .carousel-item-right.
       2. .item is now .carousel-item.
       3. For prev/next controls, .carousel-control.right and .left are now .carousel-control-right and –left, meaning they no longer require a specific base class.
       4. Removed all responsive styling, deferring to utilities (e.g. showing captions on certain viewports) and custom styles as needed.
       5. Removed image overrides for images in carousel items, deferring to utilities.
       6. Tweaked the carousel example to include the new markup and styles.
28. Utilities
    1. Made display utilities responsive (e.g. .d-none and d-{sm, md, lg, xl}-none).
    2. Added .float-{sm, md, lg, XL}-{left, right, none} classes for responsive floats and removed .pull-left .pull-right since they’re redundant to .float-left and .float-right.
    3. Added responsive variations to our text alignment classes .text-{sm, md, lg, XL}-{left, center, right}.
    4. Added new margin auto utilities for all sides, plus vertical and horizontal shorthand’s.
    5. Added boatload of flexbox utilities.
    6. Dropped .center-block for the new .mx-auto class.
29. Vendor prefix mixins
    1. Bootstrap 3’s vendor prefix mixins, which were deprecated in v3.2.0, have been removed in Bootstrap 4. Since we use Autoprefixer, they’re no longer necessary.
    2. Removed the following mixins:
       1. Animation, animation-delay, animation-direction, animation-duration, animation-fill-mode, animation-iteration-count, animation-name, animation-timing-function, backface-visibility, box-sizing, content-columns, hyphens, opacity, perspective, perspective-origin, rotate, rotateX, rotate, scale, scaleX, scaleY, skew, transform-origin, transition-delay, transition-duration, transition-property, transition-timing-function, transition-transform, translate, translate3D, user-select.
30. Documentation
    1. Our documentation received an upgrade across the board as well. Here’s the low down:
    2. We’re still using Jekyll, but we have custom plugins in the mix:
       1. Example.rb is a fork of the default highlight.rb plugin, allowing for easier example-code handling.
       2. Callout.rb is a similar fork of that, but designed for our special docs callouts.
    3. All docs’ content has been rewritten in markdown (instead of HTML) for easier editing.
    4. Pages have been reorganized for simpler content and a more approachable hierarchy.
    5. We moved from regular CSS to SCSS to take full advantage of Bootstrap’s variables, mixins and more.
31. Responsive utilities
    1. All @screen- variables have been removed in v4.0.0 use the media-breakpoint-up (), media-breakpoint-down (), or media-breakpoint-only () SASS mixins or the $grid-breakpoints SASS map instead.
    2. The responsive utility classes have also been overhauled.
    3. The .hidden and .show classes have been removed because they conflicted with jQuery’s $(…).hide () and $(…).show () methods. Instead, try toggling the [hidden] attribute, use inline styles like style=”display: none” and style=”display: block” or toggle the .invisible class.
    4. The old classes (.hidden-xs .hidden-sm .hidden-md .hidden-lg .visible-xs-block .visible-xs-inline .visible-xs-inline-block .visible-sm-block .visible-sm-inline .visible-sm-inline-block.visible-md-block .visible-md-inline .visible-md-inline-block .visible-lg-block .visible-lg-inline .visible-lg-inline-block) are gone.
    5. They have been replaced by .hidden-xs-up .hidden-xs-down .hidden-sm-up .hidden-sm-down.hidden-md-up .hidden-md-down .hidden-lg-up .hidden-lg-down.
    6. The .hidden-\*-up classes hide the element when the viewport is at the given breakpoint or larger (e.g. .hidden-md-up hides an element on medium, large, and extra-large devices).
    7. The .hidden-\*-down classes hide the element when the viewport is at the given breakpoint or smaller (e.g. .hidden-md-down hides an element on extra-small, small, and medium devices).
    8. Rather than using explicit .visible-\* classes, you make an element visible by simply not hiding it at that screen size. You can combine one .hidden-\*-up class with one .hidden-\*-down class to show an element only on a given interval of screen sizes (e.g. .hidden-sm-down.hidden-xl-up show the element only on medium and large devices).
    9. Note that the changes to the grid breakpoints in v4 means that you’ll need to go one breakpoint larger to achieve the same results (e.g. .hidden-md is more similar to .hidden-lg-down than to .hidden-md-down). The new responsive utility classes don’t attempt to accommodate LESS common cases where an element’s visibility can’t be expressed as a single contiguous range of view sizes; you will instead need to use custom CSS in such cases.
32. Misc notes to prioritize
    1. Removed the min—moz-device-pixel-ratio typo hack for retina media queries
    2. Changes buttons’ [disabled] to: disabled as IE9+ Supports: disabled. However fieldset [disabled] is still necessary because native disabled fieldsets are still buggy in IE11.
33. Additional notes
    1. Removed support for styled nested tables (for now)

**Comparison**

|  |  |  |
| --- | --- | --- |
| Component | Bootstrap 3 | Bootstrap 4 (Alpha 6) |
| Global |  |  |
| Source CSS Files | LESS | SCSS |
| primary CSS unit | px | rem |
| media queries unit | px | px |
| global font size | 14px | 16px |
| default fonts | Helvetica Neue, Helvetica, Arial, sans-serif | uses a "native font stack" (User's system fonts), with a fallback to Helvetica Neue, Arial and sans-serif |
| Grid |  |  |
| grid tiers | 4 tier grid system (xs, sm, md, and lg) | 5 tier grid system (xs, sm, md, lg, and xl) |
| offsetting columns | Uses col-\*-offset-\* classes to offset columns. For example, col-md-offset-4 | Uses offset-\*-\* classes to offset columns. For example, offset-md-4. |
| tables |  |  |
| inverse tables | not supported | Added inverse tables with the .table-inverse class. |
| table head styles | not supported | Added table head styles with the .thead-default and .thead-inverse classes. |
| condensed tables | .table-condensed | .table-sm |
| contextual classes | Bootstrap 3 doesn't use the .table- prefix for its contextual classes. For example, Bootstrap 3 uses .active whereas Bootstrap 4 uses .table-active. Other than that, both versions use the same 5 contextual keywords (active, success, info, warning, and danger). | Added the .table prefix for its contextual classes. |
| responsive tables | The .table-responsive class must be added to a parent <div> element. | can add .table-responsive to the actual <table> element |
| reflow tables | not supported | Reflow tables have now been dropped. |
| forms |  |  |
| horizontal forms | require the .form-horizontal class forms don't require .row when using grids | Dropped .form-horizontal class - it is no longer needed to display horizontal forms. Forms require the .row class when using grids. |
|  | Use .control-label when using grids for form layout. | It uses .col-form-label \* when using grids for form layout. \*note initially used .form-control-label but subsequently changed this to .col-form-label |
| checkboxes and radio buttons | Uses .radio, .radio-inline, .checkbox or .checkbox-inline to display checkboxes and radio buttons. | Uses .form-check, .form-check-label, .form-check-input, and .form-check-inline. |
| form control size | Use .input-lg and -sm to increase or decrease the size of an input control. | Use .form-control-lg and -sm to increase / decrease the size of the control. |
| help text | Use the .help-block class to display help text. | It uses .form-text class to display help text. |
| validation and feedback icons | The .form-control-\* classes are not available in 3. To present icons on the input fields using 3 you need to use Glyphicons. | Dropped the .has-feedback class. It is no longer required with the introduction of the .form-control-\* classes. |
| legends | No classes for styling the form legends. | Provides the option of using .col-form-legend on <legend> elements to style it more like a label. |
| custom forms | not supported | In 4 introduced custom forms completely custom form elements that replace the browser defaults. |
| Buttons |  |  |
| Semantic Style | Includes the .btn-default and -info classes. The .btn-secondary class isn't available in 3. | Introduce the .btn-secondary class. Dropped .btn-default class.  Note: .btn-info class was initially dropped in 4 but it has reappeared again. |
| outline buttons | not supported | Introduce the .btn-outline-\* classes for styling buttons with an outline color. |
| button size | .btn-xs class is available | dropped .btn-xs class (only .btn-sm and .btn-lg are available) |
| Images |  |  |
| responsive Image | Use .img-responsive class. | Use .img-fluid class. |
| image alignment | Use .pull-right, left and .center-block helper classes. | Use .m-x-auto instead of .center-block to align block-level images.  Can also use the various .pull-\*-right and .pull-\*-left responsive helper classes, as well as the .text-\*-left, .text-\*-center, and .text-\*-right helper classes on the image's parent.  Can use the various .pull-\*-none classes to disable floating. |
| Media Objects |  |  |
| classes | Includes many different classes for media objects, including .media, .media-body, .media-object, .media-heading, .media-right, .media-left and .media list and .media-body. | Uses just .media class. Margins can be applied using spacer utilities. Media objects are flexbox enabled in 4, so the various flexbox classes can also be applied  (such as reordering, etc.) |
| Dropdowns |  |  |
| structure | Apply dropdowns to lists (i.e. using <ul> and <li>). | Dropdowns can be built with <ul> or <div>. Apply the .dropdown-item to a <a> or <button> element and wrap them all in a <div> (or <ul>) with a .dropdown-menu class applied. |
| menu headers | Apply .dropdown-header to the <li> tag. | Apply .dropdown-header to <h1> - <h2> tags (as Bootstrap no longer uses <li> tags to build dropdowns). |
| dividers | Apply the .divider class to the <li> element (because it uses lists to build dropdowns). | Apply the .dropdown-divider to the <div> element. |
| disabled menu items | Apply the .disabled class to the <li> element. | Apply the .disabled class to the <a> element. |
| Button Groups |  |  |
| Justified? | Supports justified buttons groups (via the .btn-group-justified class). | Not supported. |
| Extra small? | Supports extra small button groups (via the .btn-group-xs class). | Not supported. |
| navs |  |  |
| inline navs | there is no .nav-inline class | Can use the .nav-inline class to explicitly specify navs to be displayed inline. |
| navbars |  |  |
| colors | Limited (preset) color options. Supported inverse navbars but not the other classes. | Now (preset) color options. Introduced the .navbars-light class, and allows the .bg-\* classes to be used on navbars. |
| navbar alignment | use .navbar-right, .navbar-left to align components within the navbar | Can either use spacing utilities such as .mr-auto, or any of the flexbox alignment utilities? |
| navbar forms | Add the .navbar-form class to forms within navbars. | Dropped .navbar-form class. It's no longer necessary |
| fixed navbars | Uses .navbar-fixed-top and .navbar-fixed-bottom to fix navbars to the top or bottom. | Uses .fixed-top and .fixed-bottom. |
| Pagination |  |  |
| default pagination | Only requires .pagination to be added (to the <ul> element that represents the list of pages). | Must also add .page-item to each <li> element and .page-link to each <a> element. |
| pagers | Use .previous and .next for aligning pagers. | Pagers have been dropped in Bootstrap 4 (alpha 3). |
| labels |  |  |
| pill labels | The .label-pill class is not available. However, Bootstrap 3 does have badges (which achieves a similar visual effect). | Labels have been replaced by badges in 4. Badges can use the .badge-pill for rounded corners. |
| tags |  |  |
| Supported? | no | Tags have now been renamed to "badges". There replace labels from 3. |
| jumbo Tron |  |  |
| full-width | The .jumbotron-fluid class is not required on full-width jumbotrons. | Introduced the .jumbotron-fluid class for full-width jumbotrons. |
| progress bars |  |  |
| Uses <progress>? | Doesn’t use the <progress> for progress bars. Instead, applies progress bar classes to nested <div> elements. | Using the <progress> element was abandoned in Alpha 6. In 4 now uses the <div> element again. |
| Glyphicons |  |  |
| Supported? | supported | Not supported. |
| Typography | Bootstrap styles are applied to the <blockquote> element by default. | Introduced the .blockquote class for styling the <blockquote> element (i.e. styling this element is now opt-in). |
| page Headers | .page-header classes is supported | .page-header classes is not supported |
| description lists | .dl-horizontal class was used to declare a horizontal list. | Horizontal lists are now declared with the .row class on the <dl> tag, then any of the grid system's predefined classes to the <dt> and <dd> tags. |
| Non-Responsive Usage |  |  |
| Supported? | Supported. You can specify a layout to be non-responsive | Not supported. |
| List Groups |  |  |
| Linked List Items / Button List Items | Apply .list-group-item to the <a> element. | Apply .list-group-item-action to the <a> element. |
| Collapse |  |  |
| Show Content | Uses .in to have content expanded upon page load. | Uses .show to have content expand upon page load. |
| Cards |  |  |
| Supported? | not supported | Introduced in 4. Cards replace functionality that was previously provided by panels, wells, and thumbnails. |
| Panels | Supported. | Not Supported. Use Cards Instead. |
| Wells |  |  |
| Supported? | Supported | Not Supported. Use Cards Instead. |
| Thumbnails |  |  |
| Supported? | Supported | Not Supported. Use Cards Instead. |
| Breadcrumbs |  |  |
| Classes | Uses the .breadcrumb class against the <ul> tag. | Also requires .breadcrumb-item to be applied against all <li> elements that make up the breadcrumb.  Breadcrumbs can also be used outside of lists. For example, the .breadcrumb-item could be applied to a bunch of <a> elements that are wrapped in a .breadcrumb element. |
| carousels |  |  |
| carousel Item | Use .item class. | Use .carousel-item class. |
| affix |  |  |
| Supported? | Yes | No |

References

<https://v4-alpha.getbootstrap.com/migration/>

<http://www.quackit.com/bootstrap/bootstrap_4/differences_between_bootstrap_3_and_bootstrap_4.cfm>

**GIT**

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick references.

**Install GIT**

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

1. **Configure Tooling**

Configure user information for all local repositories.

* 1. Git config –global user.name “[name]”

Sets the name you want attached to your commit transactions

* 1. Git config –global user.email “[email address]”

Sets the email you want attached to your commit transactions

* 1. Git config –global color.ui auto

Enables helpful colorization of command line output

1. **Create Repositories**

Start a new repository or obtain one from an existing URL.

* 1. Git init [project-name]

Creates a new local repository with the specified name

* 1. Git clone [url]

Downloads a project and its entire version history

1. **Make Changes**

Review edits and craft a commit transaction

* 1. Git status

Lists all new or modified files to be committed

* 1. Git diff

Shows file differences not yet staged

* 1. Git add [file]

Snapshots the file in preparation for versioning

* 1. Git diff –staged

Shows file differences between staging and the last file version

* 1. Git reset [file]

Unstages the file, but preserve its contents

* 1. Git commit –m “Message”

Records file snapshots permanently in version history

1. **Group Changes**

Name a series of commits and combine completed efforts

* 1. Git branch

Lists all local branches in the current repository

* 1. Git branch [branch-name]

Creates a new branch

* 1. Git checkout [branch-name]

Switches to the specified branch and updates the working directory

* 1. Git merge [branch]

Combines the specified branch’s history into the current branch

* 1. Git branch –d [branch-name]

Deletes the specified branch

1. **Refactor Filenames**

Relocate and remove versioned files

* 1. Git rm [file]

Deletes the file from the working directory and stages the deletion

* 1. Git rm –cached [file]

Removes the file from version control but preserves the file locally

* 1. Git rm [file-original] [file-renamed]

Changes the file name and prepares it for commit

1. **Suppress Tracking**

Exclude temporary files and paths

* 1. \*. Log build/temp-\*

A text file named .gitignore suppresses accidental versioning of files and paths matching the specified patterns

* 1. Git ls-files –other –ignored –exclude-standard

Lists all ignored files in this project

1. **Save Fragments**

Shelve and restore incomplete changes

* 1. Git stash

Temporarily stores all modified tracked files

* 1. Git stash pop

Restores the most recently stashed files

* 1. Git stash list

Lists all stashed change sets

* 1. Git stash drop

Discards the most recently stashed change set

* 1. Git reset [file]

Unstages the file, but preserve its contents

* 1. Git commit –m “Message”

Records file snapshots permanently in version history

1. **Review History**

Browse and inspect the evolution of project files

* 1. Git log

Lists version history for the current branch

* 1. Git log –follow [file]

Lists version history for a file, including renames

* 1. Git diff [first-branch] … [second-branch]

Shows content differences between two branches

* 1. Git show [commit]

Outputs metadata and content changes of the specified commit

1. **Redo Commits**

Erase mistakes and craft replacement history

* 1. Git reset [commit]

Undoes all commits after [commit], preserving changes locally

* 1. Git reset –hard [commit]

Discards all history and changes back to the specified commit

1. **Synchronize Changes**

Register a repository bookmark and exchange version history

* 1. Git fetch [bookmark]

Downloads all history from the repository bookmark

* 1. Git merge [bookmark]/[branch]

Combines bookmark’s branch into current local branch

* 1. Git push [alias] [branch]

Uploads all local branch commits to GitHub

* 1. Git pull

Downloads bookmark history and incorporates changes