

# Web Design Primer



# CONTENTS

---

**1. What is a URL?**

**2. Naming Conventions**

**3. Project Folder Structure**

**4. Code Guide**

**5. Images Guide**

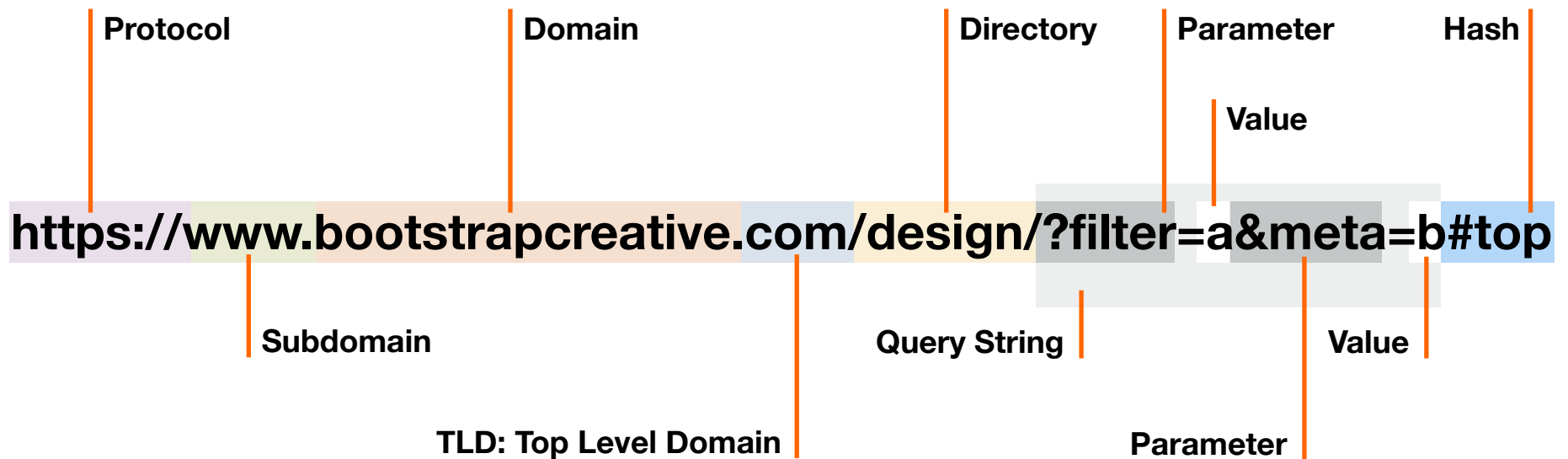
**6. Things I Wish I Knew Earlier**

© 2017 Jacob Lett. All Rights Reserved.

Please do not distribute or share without permission. You have permission to print pages but please do not try and sell it.  
If you have questions email me at [jacoblett@bootstrapcreative.com](mailto:jacoblett@bootstrapcreative.com).

# WHAT IS A URL?

---



# The Parts of a URL

The URL, or uniform resource locator, above corresponds to the folder structure shown to the right. You have the option to not use the `www` subdomain by redirecting any traffic you receive to just your domain. The query string starts with a `?` and all additional parameters start with a `&`.

## Remote Server Files



# NAMING CONVENTIONS

---

# HTML, CSS, JavaScript Naming

A general rule for all naming is to write all characters lowercase and use hyphens instead of spaces between words. If you use spaces your URL will have %20 in it which is hard to read in print.

## HTML

- Lowercase
- Avoid inline styles
- For quick scanning, write classes first
- Always include closing HTML tags
- Only use an ID if absolutely necessary

## CSS

- Lowercase with hyphens between words
- Short as possible without obscuring readability. Avoid names like .s etc.
- Prefix classes based on the closest parent or base class.

## JavaScript

- camelCase
- Short as possible without obscuring readability. Avoid variables like `var a` etc.
- jQuery objects should start with \$ as prefix. This will help you remember what variables are objects.

## Examples

---

```
<div class="btn btn-  
default" id="bt-  
action"></div>
```

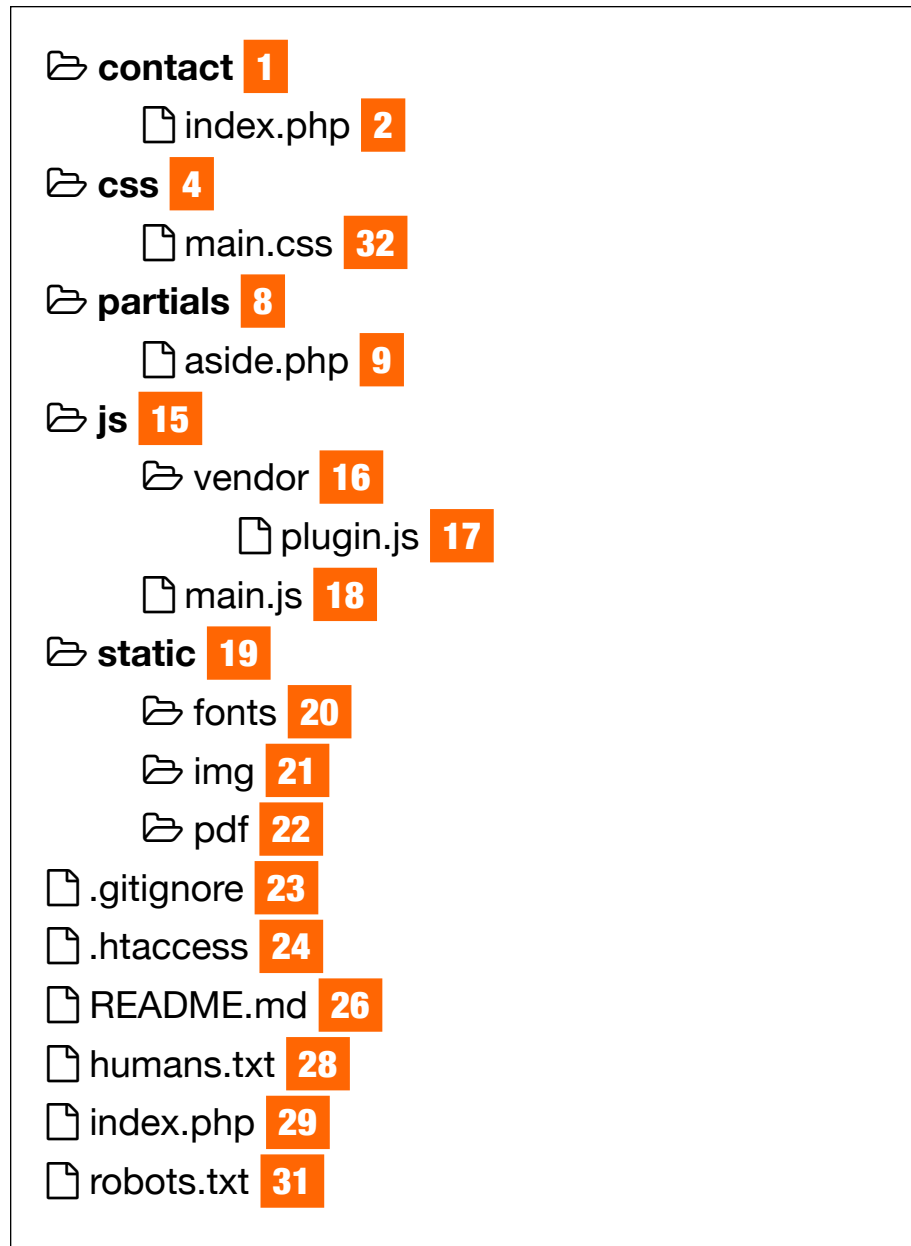
```
.btn {}  
.btn-primary {}  
#btn-action {}
```

```
var $ctaBtn = $("#btn-  
action");  
$ctaBtn.fadeIn("slow");
```

# PROJECT FOLDER STRUCTURE

---

**Figure 1:** Project Folder - PHP



## Common Project Folder Structures

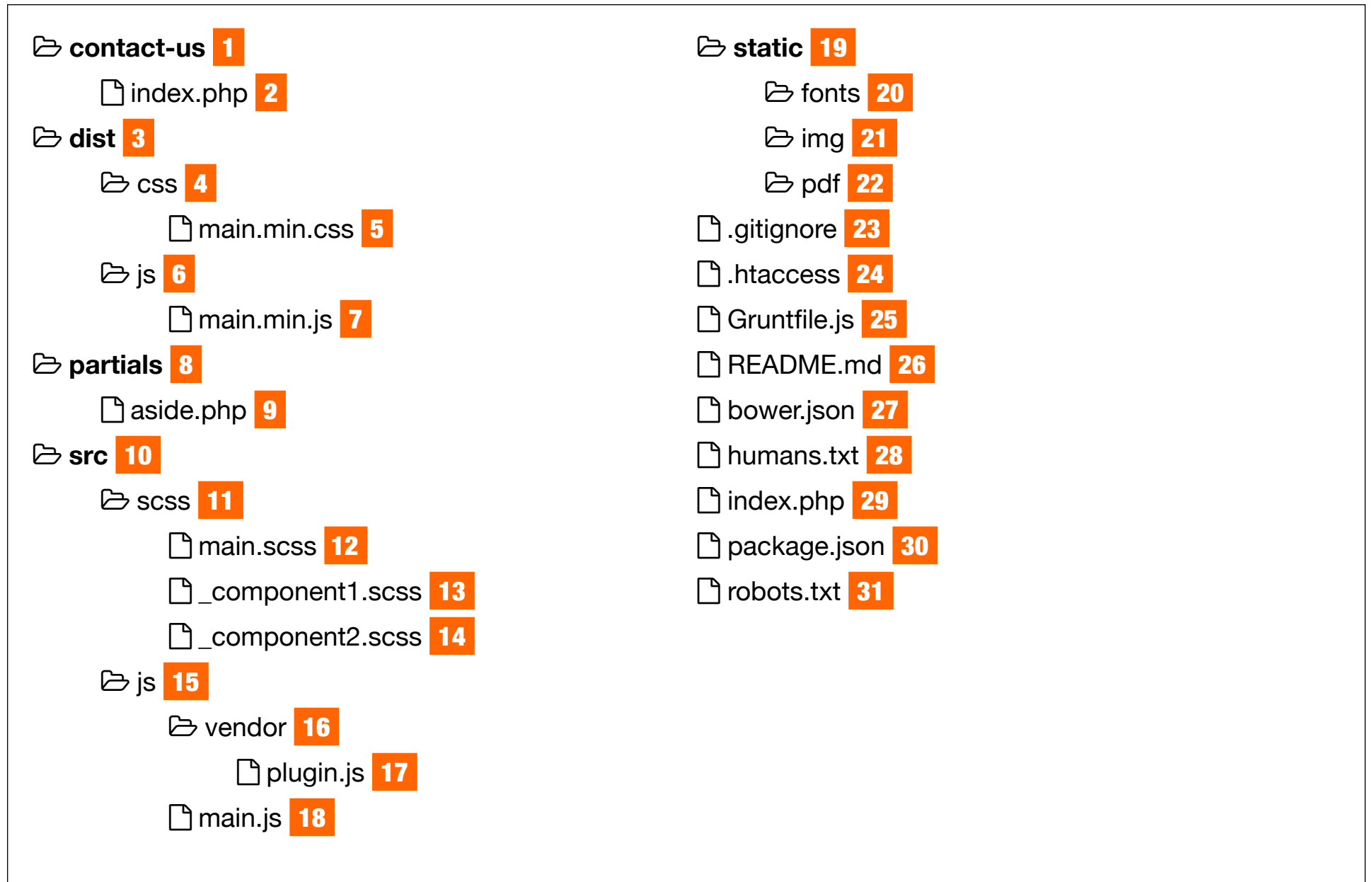
You have a lot of freedom when it comes to naming files and folders for your project. However there are some common naming conventions that can help keep your projects organized and minimize any confusion when working in a team.

**Figure 1** shows a static HTML site.

**Figure 2** shows a site using PHP and node.js to compile Sass. The project is hosted on GitHub pages.



**Figure 2:** Project Folder - PHP, Node.js, Grunt, Bower, Sass



**Below are descriptions for each file and folder in these different types of projects.**

1. A subdirectory of your main site. The URL for this would be `www.yourdomain.contact/`
2. By default Apache, the http server, looks for an index file inside of folders
3. **Dist** stands for distribution. The files inside this folder are created after compiling Sass or JavaScript through a pre-processor. In this example I used Grunt to run the Sass code through lib-sass which converts it to regular CSS.
4. This folder holds your final stylesheets.
5. Your minified stylesheet that was compiled from your Sass files
6. This folder holds your final JavaScript files
7. Your minified script file that is a merged and minified collection of all of your scripts. This helps to minimize server requests and speed up page load.
8. The partials folder contains any code snippets you include into your page templates. This way you can update one file and every instance gets auto updated because the server includes this file when it loads a page.
9. A common partial is a sidebar used on a site blog
10. **src** is short for source folder. This contains code that needs to be pre-processed, combined, and minified. So our Sass and JavaScript files go in here.
11. **scss** is short for Sassy CSS. This folder contains all of your Sass files which use the `.scss` file extension.
12. This is the main stylesheet that will be combined into the `dist/css` folder
13. A Sass file containing styles for a given component. It is imported into the `main.scss` file. Imported sass files have an underscore prefix like `_component1.scss`

14. **js** is short for JavaScript. This folder holds all of the JavaScript files you will want to merge and minify into one file. This will help your page load faster.
15. The **vendor** folder contains any third-party libraries or scripts you did not write and used in your project. Another name for these files are dependencies, because you depend on them for your project to work properly.
16. Depending on how many plugins your project has, you may want to have each as separate files. In this example, I created one **plugins.js** file and copied and pasted the plugin code into this file with comment headings above each plugin.
17. **main.js** is your core JavaScript that is ultimately referenced and loaded into your HTML page. Other common names you might use are `app.js` or the brand name.
18. The static folder contains any files not being compiled or modified by a task runner like Grunt/Gulp.
19. Folder that holds any web fonts referenced in your CSS
20. **img** is short for image and contains your .jpg, .gif, .png, and .svg files. For guidance on when and where to use each format use this reference.
21. **pdf** is short for postscript data file. All of your PDF files go in this folder
22. When hosting your project on GitHub you will see a .gitignore file which lets you add file and folder names to be excluded from your repository
23. The **.htaccess** file is read by Apache web server, server software that receives your request to view webpages over HTTP. You can use this file to configure server settings, add page redirects, and create URL rewrite urls.

24. Grunt is used in node.js projects to perform a series of tasks you define through packages. With Bootstrap projects like this, you can use Grunt to compile Sass into CSS, merge multiple files into one file, then minify the code.
25. **Readme.md** files are automatically loaded to the homepage of repositories on GitHub. They are used to provide basic documentation of the project. The .md file extension stands for markdown, which is a simplified version of HTML that needs to be converted to HTML with a preprocessor.
26. Bower is a node.js program that downloads the latest versions of libraries and frameworks into your projects. This file tells Bower exactly what version of each package you need. Bower is run through the command line.

---

*Don't undervalue your work.  
Seek criticism, not praise.  
Always keep learning & don't  
be a static learner: do this by  
reading books, magazines,  
blogs and by practicing.  
Collect & share things.  
Teach others. Never give up.  
Keep practicing. Again,  
keep practicing.  
- Jacob Cass, web designer*

---

---

*Design is a plan for arranging elements in such a way as best to accomplish a particular purpose.*

*- Charles Eames*

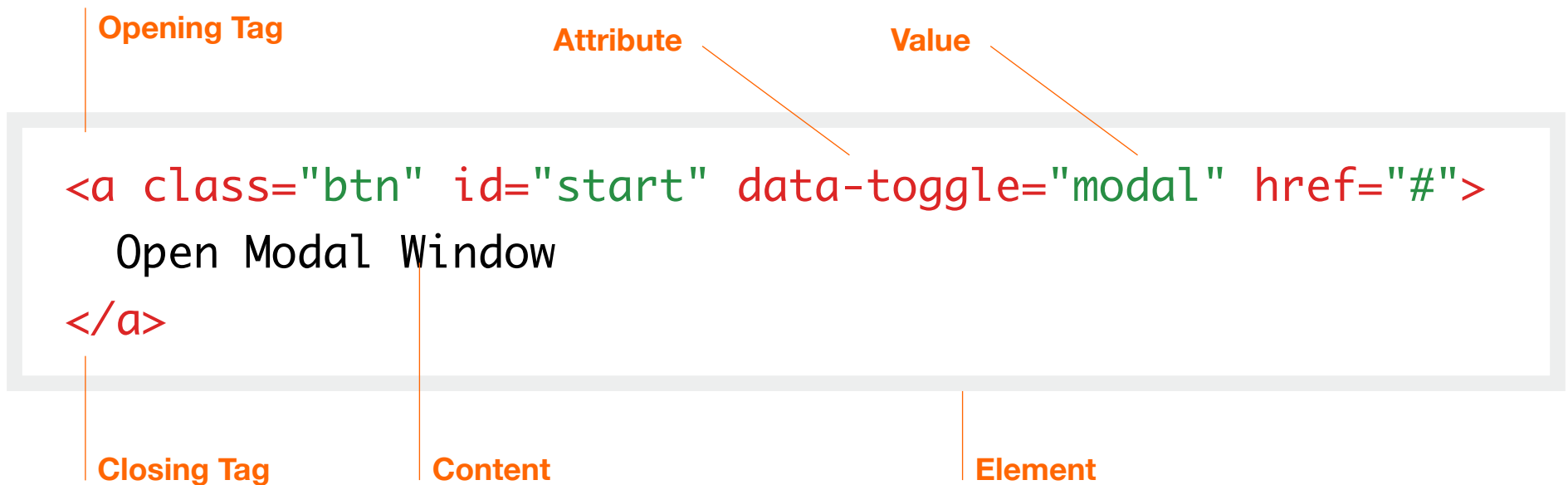
---

- 27.** `humans.txt` is optional but helps other developers who may pick up the project in the future. This file contains contact information for the creators in case there are questions.
- 28.** When a page is requested from an Apache web server it looks for a index file. Since we are running PHP our project will load `index.php` and be our homepage.

- 29.** All `node.js` projects require a file `package.json`. This file contains the project settings and commands when starting a project. The nice thing about this file is if someone downloaded your project from GitHub they could run the command `npm install` and have an exact replica of your environment. Which is what makes `node.js` projects easy to collaborate with other developers.
- 30.** The `robots.txt` file is read by Google and other search index crawlers. This file is used to tell the bots what you allow them to index and what you want to keep private.
- 31.** For static websites you will often see an unminified `css` file. If you want a minified version you will need to create this manually using <https://cssminifier.com/>

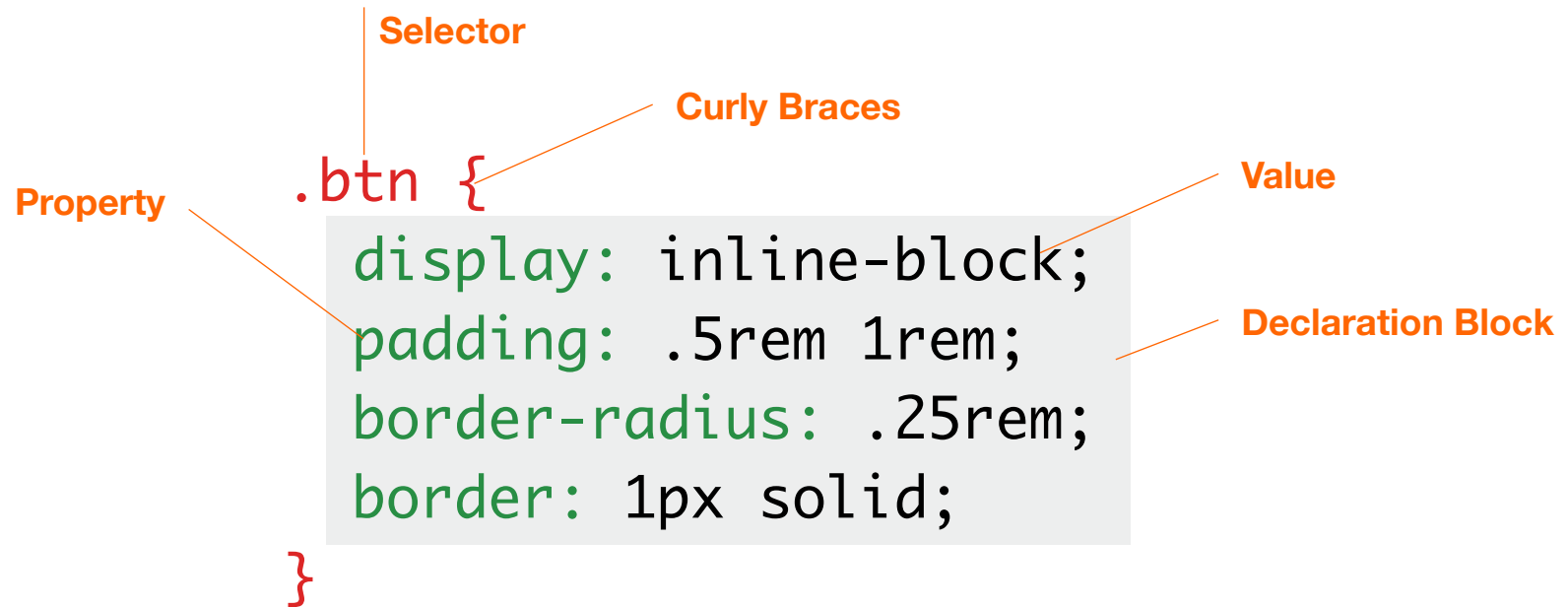
# CODE GUIDE

---



## HTML Element Parts

Short for HyperText Markup Language, the authoring language used to create documents on the World Wide Web. HTML defines the structure and layout of a web document by using a variety of tags and attributes.



## CSS Rule Set

A rule set is a single section of CSS including the selector, the curly braces, and the different lines with properties and values. The code in the example below comprises one rule set.



```
$(function() {  
    // enable toggles everywhere  
    $('[data-toggle="tooltip"]').tooltip();  
  
    var variableName = "global variable";  
    var $bodyID = $("body").attr("id");  
    console.log("Body ID #" + $bodyID);  
    // Body ID #home  
  
}); // document ready - end
```

**Document Ready Function**

**jQuery DOM Selector**

**camelCase Variable Names and Prefix jQuery Variables with \$**

**Javascript comments**

**Calls the Bootstrap 4 Tooltip Function and Runs It**

**Console Log Function Writes a Message in Chrome Dev Tools. Use for Testing.**

## jQuery and the DOM

jQuery is a DOM (Document Object Model) manipulation library. The DOM is a tree-structure representation of all the elements of a Web page. jQuery simplifies the syntax for finding, selecting, and manipulating these DOM elements.

**Figure 3:** Webpage Clean Code Example

```
<!DOCTYPE html>
<html lang="en"> 1
<head>
  <meta charset="utf-8"> 2
  <meta http-equiv="X-UA-Compatible" content="IE=edge"> 3
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no"> 4
  <meta name="description" content="">
  <meta name="author" content="">
  <link rel="icon" href="">
  <title>Bootstrap Tutorial</title>
  <!--
#####
C S S - bootstrap, custom styles
#####
-->
  <!-- Bootstrap core CSS -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css">
  <!-- Custom styles for this template -->
  <link rel="stylesheet" href="css/main.css">
</head>
<body>

5 <div class="container">
  <h1>Page Header</h1>
  <div class="row"> 6
    <div class="col-md-8" id="main-content" data-background="image.jpg"> 7
      <p>Page description paragraph</p>
    </div>
```

```

    <!-- /.col -->
    <div class="col" id="sidebar">
      <p>Sidebar text description</p>
    </div>
  </div>
  <!-- /.row --> 8
</div>
<!-- /.container -->
<!--
#####
J A V A S C R I P T - jQuery, Bootstrap, plugins Placed at the end of the
document so the pages load faster 9
#####
-->
<!-- jQuery - Grab from a CDN and if not available grab a local copy -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.0.0/jquery.min.
js"></script>
<!-- Save a local copy of jquery to your project in case the CDN is down -->
<script>window.jQuery || document.write('<script src="js/vendor/jquery.min.
js"></script>')</script> 10
<!-- Bootstrap dependency script - tether, used for tooltips -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.2.0/js/tether.min.
js"></script>
<!-- Bootstrap JS -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/
bootstrap.min.js"></script>

<!-- Custom Javascript -->
<script src="js/main.js"></script>
</body>
</html>

```

**Below are descriptions for each label on the clean code example, Figure 3.**

1. This tag tells the browser and search engines what language the document is written in.
2. The charset is very important for proper page rendering of special characters. When this is included you should not need to use HTML entities like &copy; but just paste the character in your document. For more information read this [article on how utf-8 works](#).

---

*Remember, code is your house  
and you have to live in it.*

*- Michael Feathers, web developer*

---

3. This tag allows you to choose what version of Internet Explorer the page should be rendered as. This should be as high as possible in the head of the document. Edge Mode like we have in this example, tells users viewing the page in IE to use the highest display mode possible.
4. This is one of the essential elements to have a responsive site. The other being media queries. This meta tags tells the browser how to control the pages dimensions and scaling. The viewport is the visible area of the users browser. So in this example, we are setting the width to match the screen width of the device. The initial scale is set to 1 to prevent a zoomed out version on mobile causing the user to pinch and zoom. The shrink to fit value prevents iPhone Safari from zooming out to fit elements outside the viewport.

5. In your code text editor make sure your tab indents are made with two spaces and not actually tabs. This will make sure code formatting is consistent if you copy and paste between editors.
6. Similar to an excel spreadsheet, a row is a horizontal container of cells or columns.
7. In this two column layout we will have a wide area for the main content and a narrow column for a sidebar. I have also decided to use data attributes to specify a background image I will set in my JavaScript. I listed the attributes in a particular order (class, ID, data) so my document is easy to scan.
8. It is a good practice to add HTML comments to the closing tags so that you can quickly know the page structure of your document. It can be a jumbled mess without them.

---

*Writing clean code is what you must do in order to call yourself a professional. There is no reasonable excuse for doing anything less than your best.*  
- Robert Martin, author of Clean Code

---

9. I like to use large comment blocks to make it easier to find main code blocks when scrolling through a document.
10. Since jQuery is a widely used library, chances are good most people have it already downloaded in their browser cache. But we also want to provide some fallbacks in case they do not. First we will request it from the Google CDN (content delivery network) and if that is down the user can download it from our server.

# CSS Component Structure

One important thing to understand is Bootstrap is focused around components and utility classes. When writing your custom styles it is recommended to keep your styles organized by component instead of pages.

If you are creating a new component that is not part of Bootstrap, you can write it in the format shown in the example to the right. Start with your base styles that all variations have in common so you are not redundant with styles.

Next, write your sub component styles and variation styles. Any media queries styles should be added underneath each component and not in a separate stylesheet. This will greatly improve future maintenance because you will know what styles are impacted when a component changes.

```
/*
 * Component section heading
 * Component description and use
 */

/* base - shared styles */
.component { width: 220px; }

/* Sub-component */
.component-heading {
  display: block;
  width: 100px;
  font-size: 1rem;
}

/* variant - alert color */
.component-alert {
  color: #ff0000;
}

/* variant - success color */
.component-success {
  color: #00ff00;
}

@media (min-width: 480px) {
  .component-heading { width:auto; }
}
```

# CSS Code Best Practices

1. Try and list properties in this order: 1. Positioning, 2. Box model (display, float, width, etc), 3. Typography (font, line-height), 4. Visuals (background, border, opacity), 5. Misc (CSS3 properties)
2. Any rule set with multiple declarations should be split to separate lines because syntax errors on Line numbers would be hard to find.
3. Use soft-tabs set to two spaces, set encoding to UTF-8
4. When using multiple CSS files, break them down by component instead of page.
5. Keep media queries as close to their relevant rule sets whenever possible. Don't bundle them all in a separate stylesheet or at the end of the document.
6. Do not use @import because it slows down page load.
7. Place closing braces of declaration blocks on a new line.
8. End all declarations with a semi-colon to prevent errors.
9. Lowercase all hex values. For example, #fff instead of #FFF.
10. Avoid specifying units for zero values. For example, margin: 0; instead of margin: 0px;.

## Sources



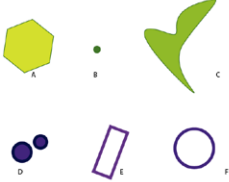
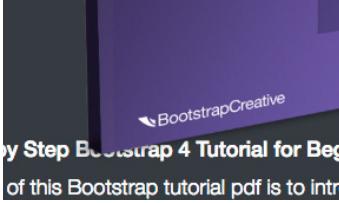
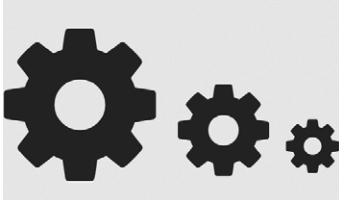
[Code Guide](#) by Mark Otto, creator of Bootstrap  
[Google Style Guide](#)

# IMAGES GUIDE

---



# Image Format Comparison

JPG	GIF	PNG8	PNG32	SVG
				
<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Raster</li> <li>• Lossy <b>1</b></li> </ul> <p><b>When to Use</b></p> <ul style="list-style-type: none"> <li>• Photographs with a lot of colors, shapes, and forms.</li> <li>• Good with gradients to lessen banding</li> </ul>	<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Raster</li> <li>• Some Transparency <b>2</b></li> </ul> <p><b>When to Use</b></p> <ul style="list-style-type: none"> <li>• Animations</li> <li>• Solid colors, symbols, and line artwork</li> <li>• Web banners</li> </ul>	<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Raster</li> <li>• Some Transparency <b>2</b></li> <li>• Lossless</li> </ul> <p><b>When to Use</b></p> <ul style="list-style-type: none"> <li>• Solid colors, symbols, and line artwork</li> </ul>	<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Raster</li> <li>• Full Transparency</li> <li>• Lossless</li> </ul> <p><b>When to Use</b></p> <ul style="list-style-type: none"> <li>• When you want a jpeg with transparency</li> </ul>	<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Vector</li> <li>• Transparency</li> </ul> <p><b>When to Use</b></p> <ul style="list-style-type: none"> <li>• Icons, logos, text that you want to be high quality on retina displays</li> </ul>

1. Lossless and lossy compression are terms that describe whether or not, in the compression of a file, all original data can be recovered when the file is uncompressed.
2. 1-bit transparency. Pixels are either solid or completely transparent, but never partially see-through.

# Responsive Images

```
<picture>
  <source srcset="https://dummyimage.com/2000x400/000/fff" media="(min-width: 1400px)">
  <source srcset="https://dummyimage.com/1400x400/000/fff" media="(min-width: 768px)">
  <source srcset="https://dummyimage.com/800x400/000/fff" media="(min-width: 576px)">
  <img srcset="https://dummyimage.com/600x400/000/fff" alt="" class="d-block img-fluid">
</picture>
<!-- If a picture looks blurry on a retina device you can add a high res img like this
-->
<source srcset="img/blog-post-1000x600-2.jpg, blog-post-1000x600-2@2x.jpg 2x"
media="(min-width: 768px)">
```

## Picture Element

The [picture element](#) gives you a lot of control on how your image looks on different breakpoints and retina displays. As you resize your window the browser will load the necessary image. It takes more work up-front to build the images but the control is worth it in prominent locations like carousels. [CodePen](#) of various image proportions. If you need to support IE11 and below use this [polyfill](#).

## When to Use

- When you want to change how an image looks on different breakpoints (size, cropping, etc.)
- Carousels and Image cards

## Responsive Images *continued*

```

```

### Image srcset

[Image srcset](#) is an attribute added to an image tag and provides various images for the browser to use depending on the viewport width. It is best used when you need little control on how it is cropped and sized. But you want to speed up page load on mobile and get rid of image pixilation on retina displays. If you need to support IE11 and below use this [polyfill](#).

One challenge with this solution is that the image is loaded on page load and does not change when the browser is resized due to image caching. To work around this, I found using the Chrome extension [Cache Killer](#) helps when testing sites.

### When to Use

- Blog post images
- Any image you want to look the same (same proportions and image) but just want to increase resolution.

# THINGS I WISH I KNEW EARLIER

---



### 1. Know and embrace common proportions and measurements (21:9, 16:9, 4:3, 3:2).

When creating layouts you will often place videos and [images at pre-determined proportions](#). Your design will be more cohesive if you use those proportions for other elements like carousel images and product images.

### 2. A great design doesn't guarantee great results.

Design and test for the majority of users and focus on making it a great experience there first. Use tools like Google Analytics and [Hotjar](#) to monitor how people use your site and look for ways to make things easier to

use. If you have never used these tools, you will be surprised to find your beautiful design doesn't actually achieve the goals your first set out to solve.

Your client or boss could love your design at launch and three months later ask you why they are not getting leads or not ranking #1 for a certain keyword phrase.

### 3. GitHub projects can have multiple versions of files stored inside one folder on your machine called branches.

If you switch branches with GitHub desktop the files are changed automatically to the new branch. This was a mental shift for me because I was so used to having different versions of projects in different folders on my computer. I was also afraid of losing or overriding something.

#### 4. Designing and building responsive websites require 3x the time & effort of a static desktop site.

Make sure you factor time in for quality assurance testing after you complete the initial build. Responsive design requires buy-in from designers, developers, writers, and executives.

#### 5. Feelings of failure are normal.

No matter how much I learn I will always have times when I experience feelings of inferiority. Sometimes I get assigned a new project and I cannot figure it out. Everything I try doesn't work and I have exhausted every Google search I could think of. The longer the problem goes without being solved the worse I feel. Then I start asking myself questions like, "How can you be a developer if you can't solve this?" or "Your boss is going to think you are a fraud and will fire you for not knowing how to solve this."

Sound familiar?

---

*Failure is an event.*

*Not a person.*

*- Zig Ziglar*

---

What has helped me is allowing more time necessary to learn. Often I would force myself to solve something by the end of the day or before lunch. Sometimes it just takes more time to solve the problem we are facing. Taking a break and do something not computer related helps to clear our mind. So many times I have struggled with some code for an entire day and being so stubborn to stop until it was fixed. Then when I finally did take a break, I would come back and find the smallest thing causing the problem. Like a period instead of a comma. : /