The HTML

By default, HTML5 Boilerplate provides two `html` pages:

- `index.html` - a default HTML skeleton that should form the basis of all pages on your website
- `404.html` - a placeholder 404 error page

`index.html`

The `no-js` Class

The `no-js` class is provided in order to allow you to more easily and explicitly add custom styles based on whether JavaScript is disabled (`.no-js`) or enabled (`.js`). Using this technique also helps avoid the FOUC.

Language Attribute

Please consider specifying the language of your content by adding a value to the `lang` attribute in the `<html>` as in this example:

```
<html class="no-js" lang="en">
```

The order of the `<title>` and `<meta>` tags

The charset declaration (`<meta charset="utf-8">`) must be included completely within the first 1024 bytes of the document and should be specified as early as possible (before any content that could be controlled by an attacker, such as a `<title>` element) in order to avoid a potential encoding-related security issue in Internet Explorer.

Meta Description

The `description` meta tag provides a short description of the page. In some situations this description is used as a part of the snippet shown in the search results.

```
<meta name="description" content="This is a description">
```

Google's Create good meta descriptions documentation has useful tips on creating an effective description.

Mobile Viewport

There are a few different options that you can use with the `viewport` meta tag. You can find out more in the MDN Web Docs. HTML5 Boilerplate comes with a simple setup that strikes a good balance for general use cases.

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

If you want to take advantage of edge-to-edge displays of iPhone X/XS/XR you can do so with additional viewport parameters. Check the WebKit blog for details.

Open Graph Metadata

The Open Graph Protocol allows you to define the way your site is presented when referenced on third party sites and applications (Facebook, Twitter, LinkedIn). The protocol provides a series of meta elements that define the details of your site. The required attributes define the title, preview image, URL, and type (e.g., video, music, website, article).

```
<meta property="og:title" content="">
<meta property="og:type" content="">
<meta property="og:url" content="">
<meta property="og:image" content="">
```

In addition to these four attributes there are many more attributes you can use to add more richness to the description of your site. This just represents the most basic implementation.

To see a working example, the following is the open graph metadata for the HTML5 Boilerplate site. In addition to the required fields we add `og:description` to describe the site in more detail.

```
<meta property="og:url" content="https://html5boilerplate.com/">
<meta property="og:title" content="HTML5 ★ BOILERPLATE">
<meta property="og:type" content="website">
<meta property="og:description" content="The web's most popular front-end template
which helps you build fast, robust, and adaptable web apps or sites.">
<meta property="og:image" content="https://html5boilerplate.com/icon.png">
```

Web App Manifest

HTML5 Boilerplate includes a simple web app manifest file.

The web app manifest is a simple JSON file that allows you to control how your app appears on a device's home screen, what it looks like when it launches in that context and what happens when it is launched. This allows for much greater control over the UI of a saved site or web app on a mobile device.

It's linked to from the HTML as follows:

```
<link rel="manifest" href="site.webmanifest">
```

Our site.webmanifest contains a very skeletal "app" definition, just to show the basic usage. You should fill this file out with more information about your site or application

Favicons and Touch Icon

The shortcut icons should be put in the root directory of your site. `favicon.ico` is automatically picked up by browsers if it's placed in the root. HTML5 Boilerplate comes with a default set of icons (include favicon and one Apple Touch Icon) that you can use as a baseline to create your own.

Please refer to the more detailed description in the Extend section of these docs.

The Content Area

The central part of the boilerplate template is pretty much empty. This is intentional, in order to make the boilerplate suitable for both web page and web app development.

Modernizr

HTML5 Boilerplate uses a custom build of Modernizr.

Modernizr is a JavaScript library which adds classes to the `html` element based on the results of feature test and which ensures that all browsers can make use of HTML5 elements (as it includes the HTML5 Shiv). This allows you to target parts of your CSS and JavaScript based on the features supported by a browser.

Starting with version 3 Modernizr can be customized using the modernizr-config.json and the Modernizr command line utility.

What About Polyfills?

If you need to include polyfills in your project, you must make sure those load before any other JavaScript. If you're using a polyfill CDN service, like polyfill.io, just put it before the other scripts in the bottom of the page:

```
  <script src="js/vendor/modernizr-3.11.7.min.js"></script>
  <script src="https://polyfill.io/v3/polyfill.min.js"></script>
  <script src="js/app.js"></script>
</body>
```

When you have a bunch of polyfills to load in, you could also create a `polyfills.js` file in the `js/vendor` directory or include the files individually and combine them using a build tool. Always ensure that the polyfills are all loaded before any other JavaScript.

There are some misconceptions about Modernizr and polyfills. It's important to understand that Modernizr just handles feature checking, not polyfilling itself. The only thing Modernizr does regarding polyfills is that the team maintains a huge list of cross Browser polyfills.

jQuery

As of v8.0.0 we no longer include jQuery by default. Web development has changed a lot since we started this project and while many millions of sites still use jQuery there are many sites and applications that don't. 10 years ago jQuery *was* JavaScript for most developers. That's not the case any more so we've made the decision to remove jQuery from the project.

If you're interested in including it, you can easily install jQuery using the following command:

```
npm install jquery
```

You can then copy the minified file into the `vendor` folder and add jQuery to the `index.html` manually.

To load jQuery from a CDN with a local fallback you can use the following:

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"
integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
<script>window.jQuery || document.write('<script src="js/vendor/jquery-
3.6.0.min.js"><\/script>')</script>
```