# HTML
# Best Practices Guide

Version: 2013.11.18

This section includes best practices regarding development of HTML source code. Details on each checklist item are discussed later in the document.

## Checklist

- ❑ Always use the HTML5 declaration (`<!DOCTYPE html5>`) at the top of HTML documents
- ❑ Always use the UTF-8 character set declaration at the beginning of the <head>
- ❑ Always use the latest IE rendering engine when applicable
- ❑ Always set the viewport equal to the device's width and height
- ❑ Do not include the "type" attribute on <script> and <style> tags
- ❑ Use <link> tag to load an external stylesheet
- ❑ Never use invalid null <script> or  tags
- ❑ Do not use <script> tags to load AMD based modules
- ❑ Never have duplicate "id" attributes
- ❑ Always be consistent with element and attribute case and quoting
- ❑ Do not use a value for boolean attributes
- ❑ Use semantic tag names where applicable
- ❑ Use HTML5 features instead of JavaScript toolkit where possible
- ❑ Avoid deprecated elements and attributes

## Accessibility Checklist

- ❑ Always validate your page to ensure its well formed
- ❑ Use A11Y tools to ensure compliance
- ❑ Use input tags for nodes with event handlers
- ❑ Use linked or enclosing `<label>` tags for all input elements
- ❑ Use CSS for layout, not tables

# Discussions

## HTML5 Basic Template

Shown below is a basic HTML5 document template.

| HTML 5 Document | Description |
|---|---|
| `<!DOCTYPE html5>` | Defines this as an HTML5 type document |
| `<html>` | Beginning of the document structure |
| `  <head>` | General conventions for children tags<br>● <meta> tags first<br>● Other preamble tags (ie <title>)<br>● <link> tags next<br>● <script> tags last |
| `    <meta charset="utf-8">` | UTF8 Character set declaration |
| `    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">` | Use latest IE rendering engine |
| `    <meta name="viewport" content="width=device-width, height=device-height">` | Set device's viewport equal to the device's actual width and height.<br>Note: device-height works properly in iOS7 but may have problems on iOS5 and iOS6.<br>In iOS7, when the soft keyboard is displayed, the height of the viewport now changes to be the remaining visible space, which is different behavior from previous versions of iOS. |
| `    <style …>`<br>`    <script …>` | Next link to your CSS stylesheets.<br>Load script tags AFTER style tags. |
| `  </head>` | |
| `  <body>`<br>`  </body>` | |
| `</html>` | Close out the document |

Note: When dealing with html templates or other document fragments, you should not include any of the following tags; doctype, html, head, body. You should just use a master enclosing  tag to represent to top level element of your fragment.

### Always use the UTF-8 character set declaration at the beginning of the <head>

The sooner the browser knows for sure what character set to use, the faster it can begin requesting resources, and rendering the page. Always place the following as the very first element within the <head> of the document:

```
<meta charset="utf-8">
```

References:

- Performance implications of Charset

## HTML5 Enhancements

### Semantic Elements

Semantic element names provide a meaning to a tag. HTML5 has improved semantic comprehension by adding many new element names.

- Good: `<header>`, `<nav>`, etc.
- Bad: `<div class="header">`, or `<span id="mainNav">`

### Added Features

HTML has included many enhancements that were historically provided by 3rd party JavaScript libraries.

- Example attributes include: autofocus, placeHolders, hidden, draggable, required
- Example input types: email, url, search
- Fall back to shims if necessary for older browsers.

## Elements / Tags

Elements (or tags) make up the structure of an HTML document. All elements should be lowercase. While the browser doesn't care, you should be consistent to improve readability and maintainability of your HTMl files.

Many HTML4 elements are deprecated and should be replaced by CSS in order to separate content from presentation.

- e.g. <font>, , etc.

- See HTML5 Obsolete tags for a full list of deprecated tags

###  tag

 tags are required to have a closing . You may not use the null body syntax on  tags.

- Good: `<div id="myTarget"></div>`
- Bad: `<div id="myTarget" />`

### <link> tag

Used to load external resources into a document. Typically used for loading external stylesheets.

### <script> tag

Script tags are used to either include a JavaScript source file into a document (using the src="uri" attribute), or to define inline JavaScript.

Under most circumstances, you should keep all JavaScript in external ".js" files. This improves overall app maintainability and separation of concerns. The exception to this rule is if you need to preset any configuration values prior to loading a script (i.e. setting dojoConfig), or to otherwise bootstrap your app once scripts are loaded.

Care should be used as to where you place <script> tags. If they are placed in the <head> section, then the scripts are loaded and processed prior to the rendering of the

Do not include the "type" attribute on script tags as they are no longer used in HTML5.

- Good: `<script src="libs/myJsLib.js"></script>`
- Avoid: `<script type="text/javascript" src="libs/myJsLib.js"></script>`

Script tags are required to have a closing </script> tag

- Good: `<script src="./js/hello.js"></script>`
- Bad: `<script src="./js/hello.js" />`

Use AMD require("module") to load built layer files and scripts, rather than <script> tags. The exceptions include the main "loader script" (ie r.js or dojo.js), and 3rd party libraries that are not AMD compatible.

### <style> tag

Do not include the "type" attribute on style tags.

- Good: `<script src="libs/myJsLib.js"></script>`

- Avoid: `<script type="text/javascript" src="libs/myJsLib.js"></script>`

## Attributes

Attributes are used to alter the definition of an element. Attributes consist of name (key), and typically a string value. All attribute keys should be lowercase. While the browser doesn't care, you should be consistent to improve readability and maintainability of your HTML files.

Attributes with a value (non-boolean) should always be quoted. It is recommended that double quotes be used to avoid having to escape apostrophe's within the string.

- Good: `<div label="It's my life"></div>`
- Poor: `<DIV Label='It\'s my life'></Div>`

Boolean attributes (e.g. autofocus, autocomplete, required) do not have a value. Their existence implies true. In HTML5 this means that `required="false"` is equivalent to `required="true"`!

- Good: `<input type="email" required />`
- Bad, as its still required: `<input type="email" required="false"/>`

### "id" attribute

The "id" attribute is used to uniquely identify a single element within the DOM tree. When used, it must be a unique string. When dealing with dynamic content and automatically loaded content, duplicate Id's can occur which can result in difficult to resolve bugs. The is particularly common in portal and mashup environment.

Using "id" should be avoided when possible. Exceptions to this rule are when using the <label> tag, as it requires a [for='id'] attribute that points to an external input field. In widget templates, you can use the widget's instance ID to create a "local" ID attribute. E.g.

```
<label for="${id}_email">Email:</label>
<input id="${id}_email" .../>
```

An alternative is to use CSS classname addressing and queries. eg.

```
<div class="myFirstName"></div>

var div = ${".myFirstName"}[0];
```

For Dojo, use "attach-points" in dijit templates, which are local to that dijit's instance. eg.

```
<div data-dojo-attach-point="dapFirstName"></div>
```

## Accessibility

For a great introduction to proper accessibility in Web Apps, take Google's Introduction to Web Accessibility course.

Always validate your page to ensure its well formed. This is the number one cause of accessibility enabled devices failing to process a page.

Use A11Y tools to ensure compliance:

- Screen readers: ChromeVox, and others
- Rational tools

Use HTML5 semantic nodes like <nav> and <header>

Use proper input tags (eg <button> and <a>) for nodes with event handlers like `onClick`.

Use linked or enclosing <label> tags for all input elements.

### Use CSS for layout, not tables

It is a hard habit to break, but many developers still fall back to using <table>'s for layout. This causes a lot of issues for A11Y and screen readers. Tables are for tabular content, not layout. Either use CSS positioning, or a layout based toolkit (like backbone), or the new FlexBox CSS rules.

## Reference Links

- Accessibility Programming Guide for iOS
- Android Designing for Accessibility
- Blackberry Accessibility
- Microsoft UI Automation