



# HTML: Introduction

Juan “Harek” Urrios  
@xharekx33

# What is HTML?

---

HTML stands for **HyperText Markup Language**. Is the language that describes the semantic structure of a website so web browsers can render the pages and users can view or hear them.

It is the brainchild of Sir Tim Berners-Lee who in 1991 came up with a document titled "HTML tags" describing 18 elements that could be used to describe web pages.



# How to create a web page

---

To create a simple web page all you need is a text editor and a browser.

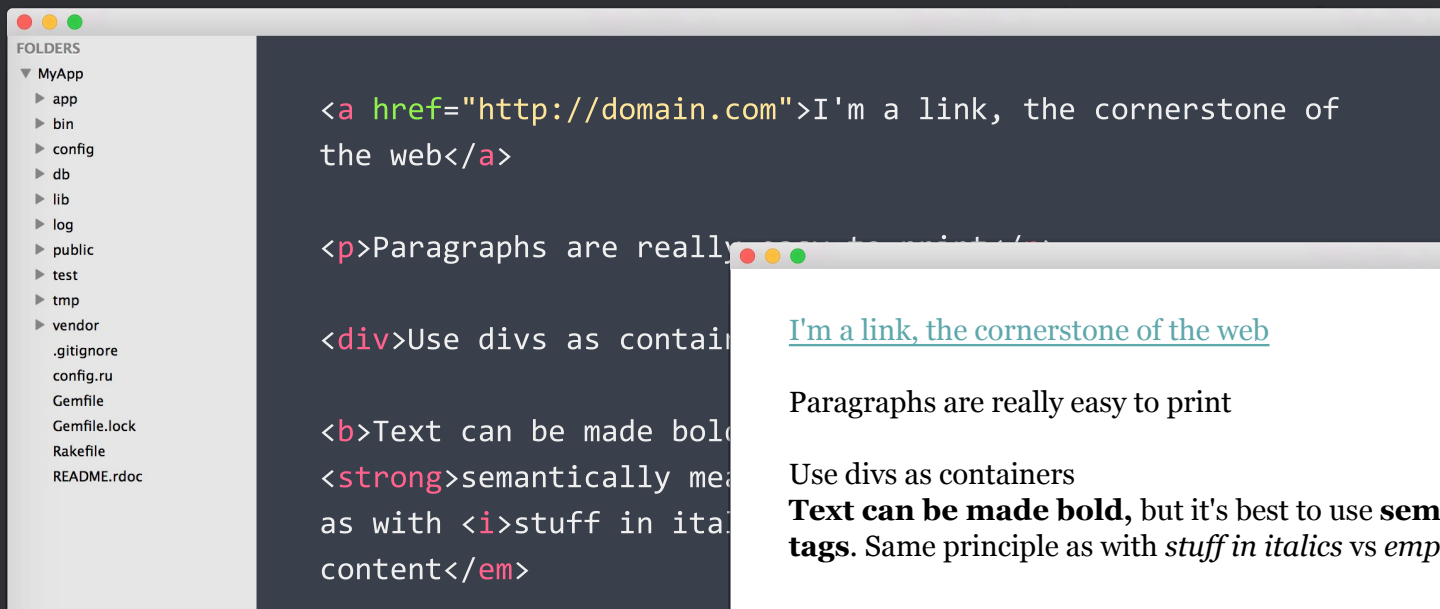
Create a new file, foo.html, add some HTML code, save it and open the file in your browser. Whenever you change something, just reload the page in the browser to see them.

For a more convenient and quick option for running your tests, you can also use tools such as [codepen.io](https://codepen.io) or [jsfiddle.net](https://jsfiddle.net)



# Tags and attributes

HTML consists of **tags**, that describe the elements in the page and its structure and **attributes** that will provide values to enable or modify their functionality.



```
<a href="http://domain.com">I'm a link, the cornerstone of  
the web</a>
```

```
<p>Paragraphs are really easy to print</p>
```

```
<div>Use divs as containers</div>
```

```
<b>Text can be made bold</b>
```

```
<strong>semantically meaningful</strong>  
as with <i>stuff in italics</i>  
content</em>
```

I'm a link, the cornerstone of the web

Paragraphs are really easy to print

Use divs as containers

**Text can be made bold**, but it's best to use **semantically meaningful tags**. Same principle as with *stuff in italics* vs *emphasized content*

# What are tags and attributes good for?

---

Using tags and attributes you can:

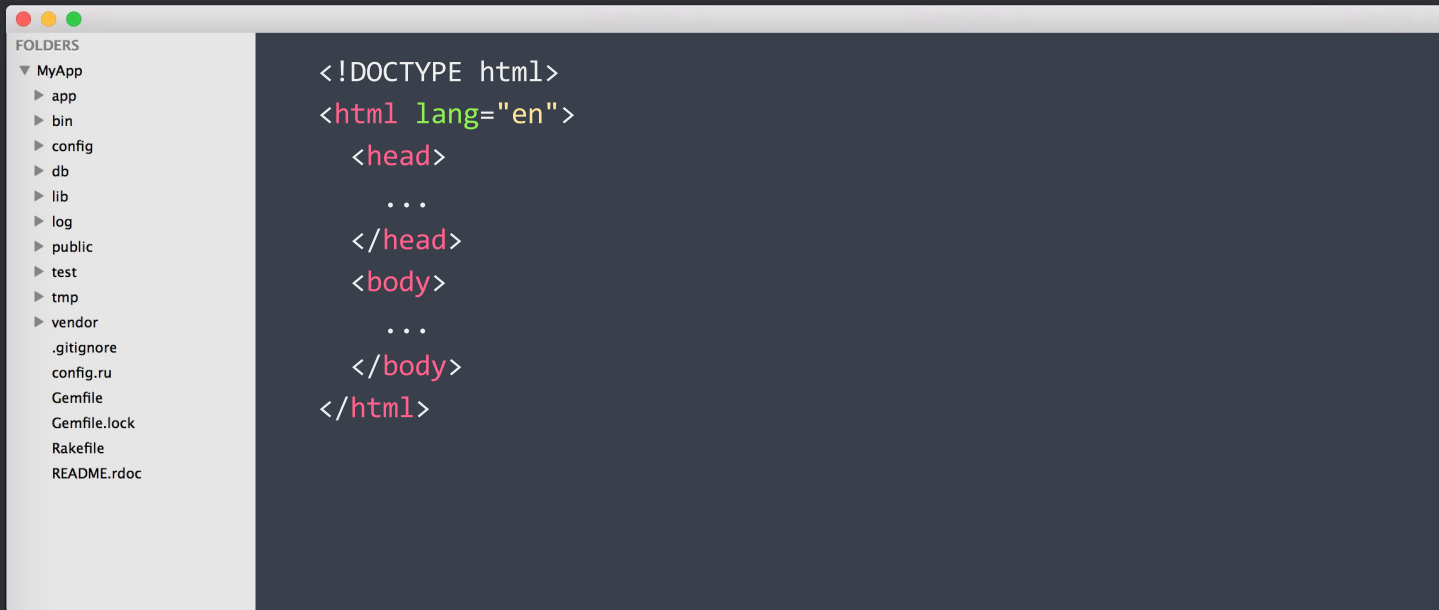
- Add content to the page.
- Wrap it and add structure and meaning to it.
- Style it.
- Load assets (images, video, stylesheets, javascript...).
- Make it interactive.
- Tell the browser how your page should be rendered and how it should behave.

Browsers will not display the actual tags, but will use them to render the page.



# A basic HTML page

---



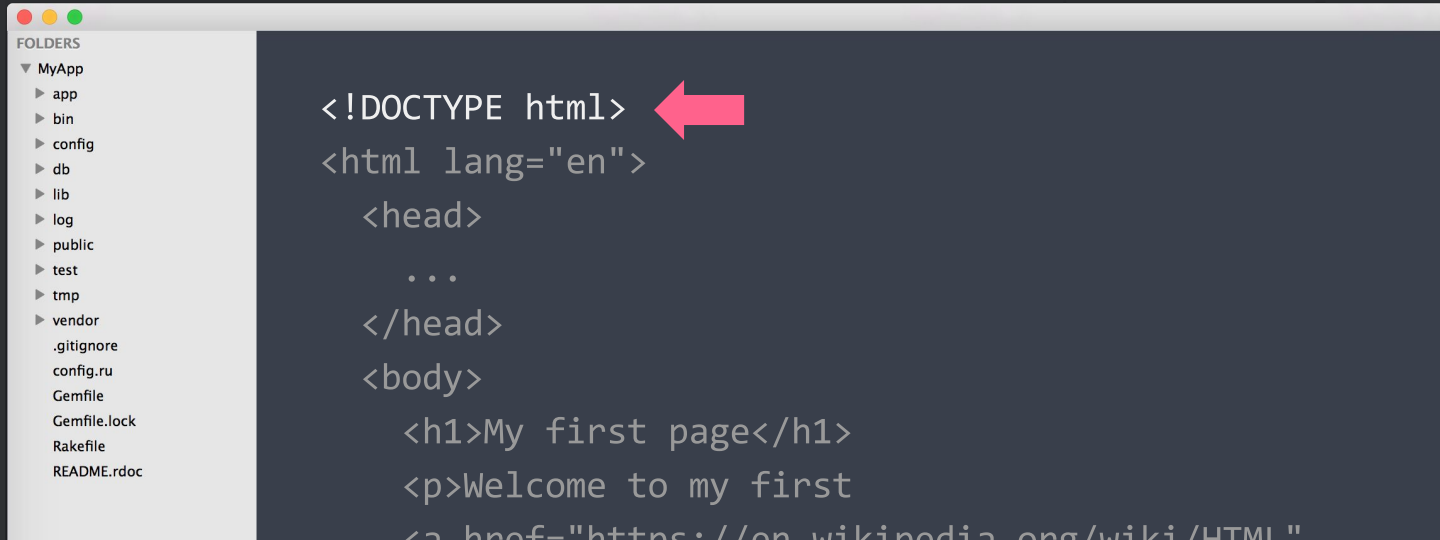
The image shows a code editor window with a sidebar on the left and a main editing area on the right. The sidebar, titled 'FOLDERS', lists a project structure under 'MyApp' including folders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area contains the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```



# The doctype

The doctype (Document Type Declaration) specifies the type (and version) of markup a document is written in, so the browser know what to expect and renders it properly. Always include it at the beginning of your pages.



The screenshot shows a code editor window with a sidebar on the left labeled 'FOLDERS' containing a tree view of a project named 'MyApp'. The main editor area displays HTML code. A pink arrow points to the first line of the code, which is the doctype declaration.

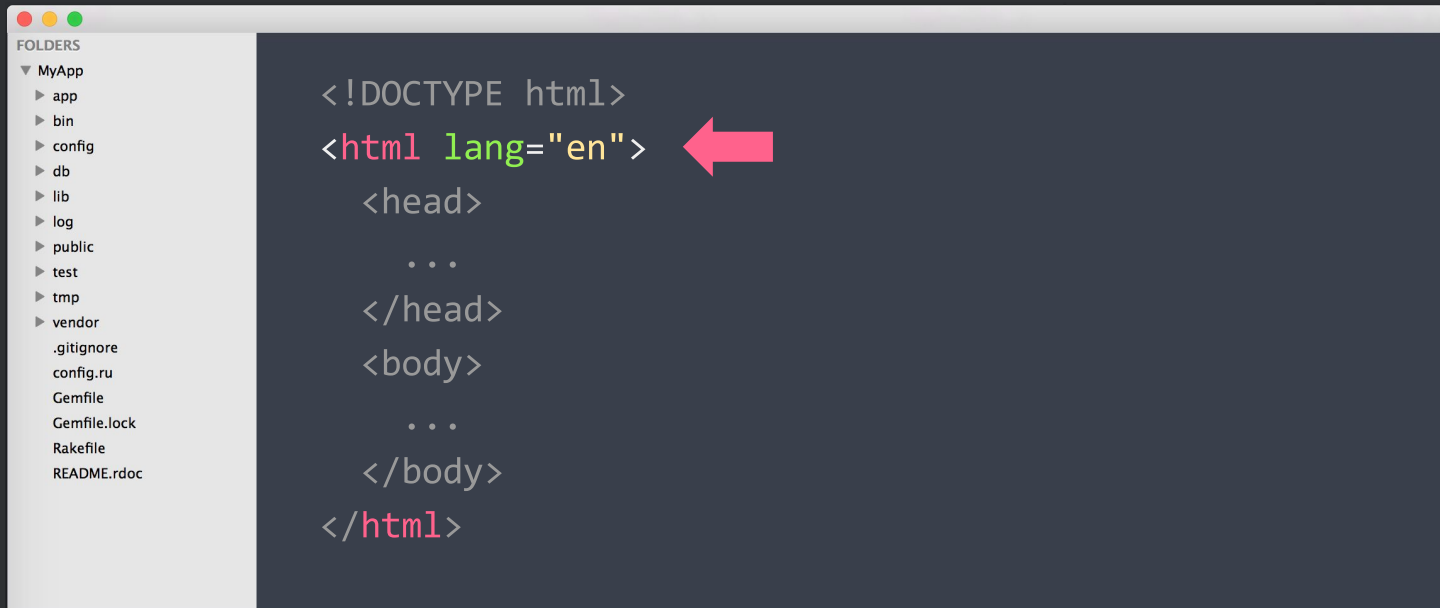
```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    <h1>My first page</h1>
    <p>Welcome to my first
    <a href="https://en.wikipedia.org/wiki/HTML"
```



@xharekx33

# The `<html>` tag

Tells the browser that this is a HTML document: it represents the start of the document and serves as a container for all the rest of the elements in the page.



The screenshot shows a code editor window with a sidebar on the left listing folders and files. The main editor area displays HTML code. A red arrow points to the `<html lang="en">` tag. The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

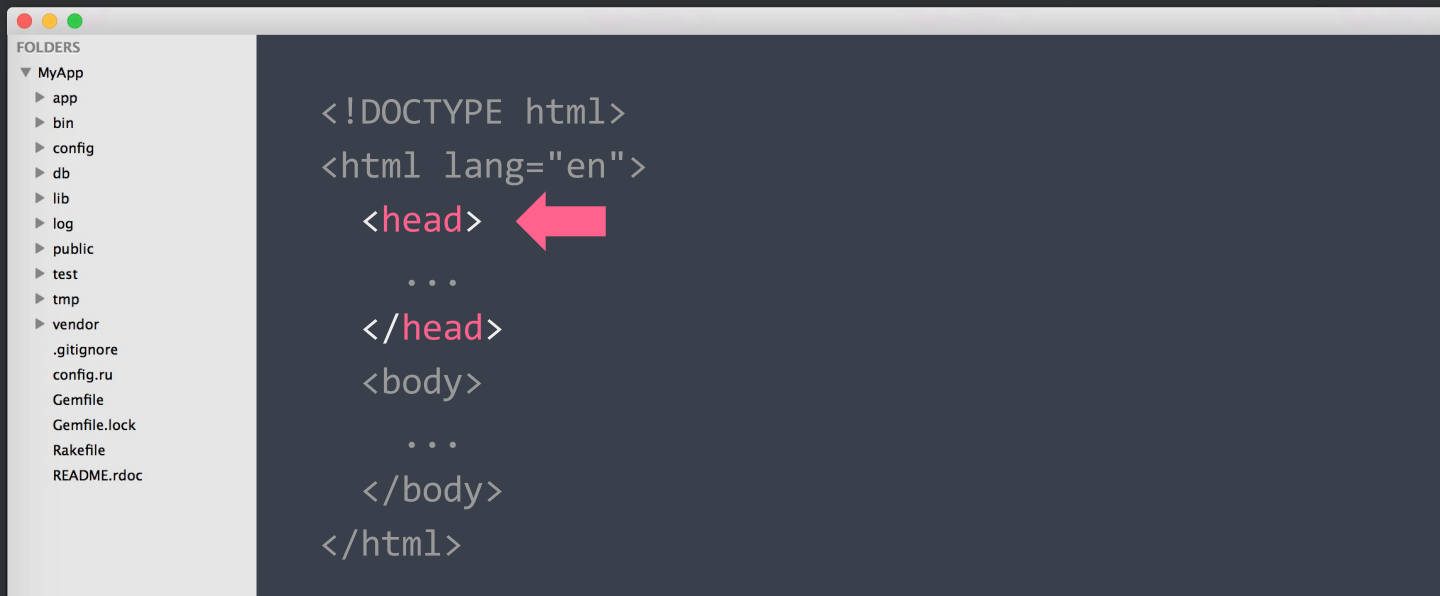


@xharekx33



# The `<head>` tag

Contains all the properties and metadata of the document including its title, base url and links and or definitions for stylesheets.



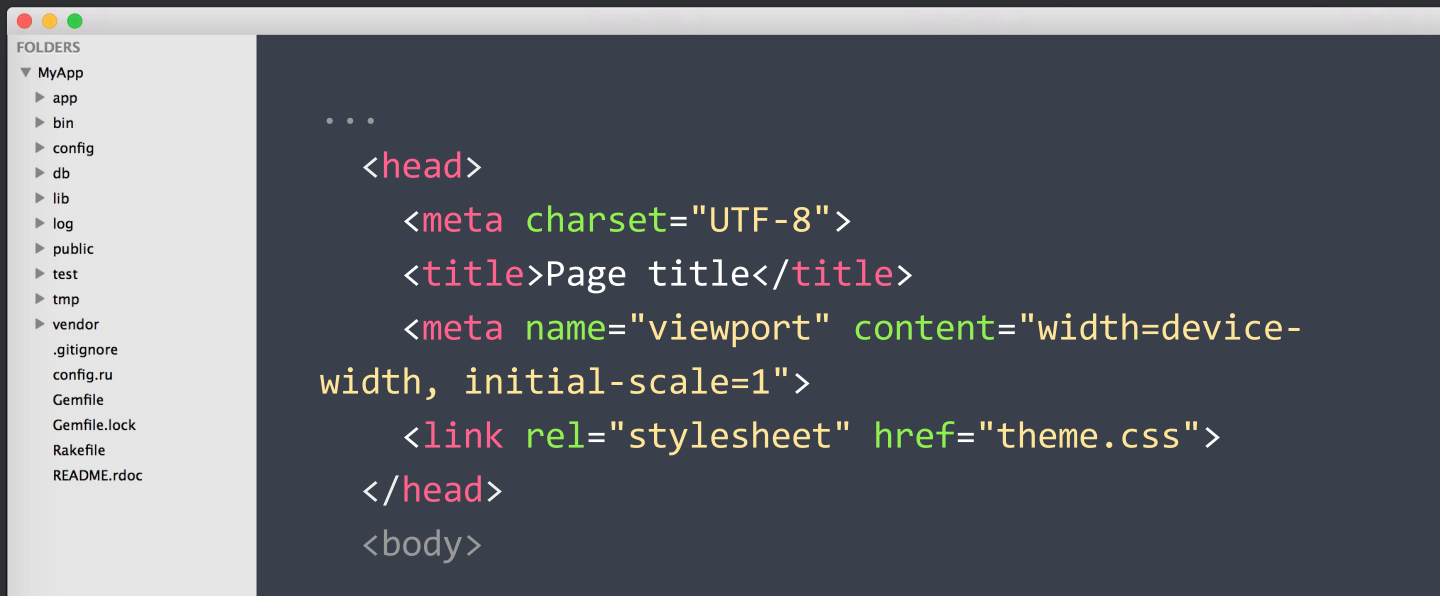
```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```



@xharekx33

# <head>: What goes inside?

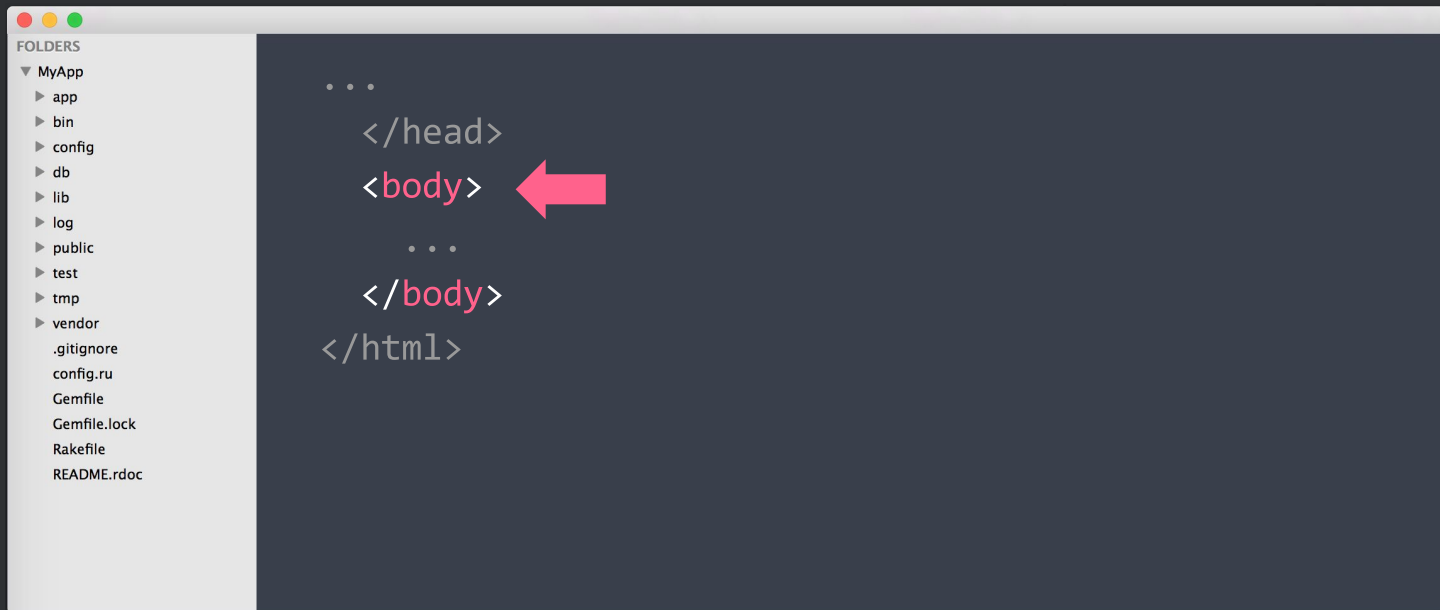
Important stuff to include in your <head>: title, charset, viewport, styles...



@xharekx33

# The `<body>` tag

Defines the section where the main content of the page will be placed. It contains everything that will be displayed on the viewport.

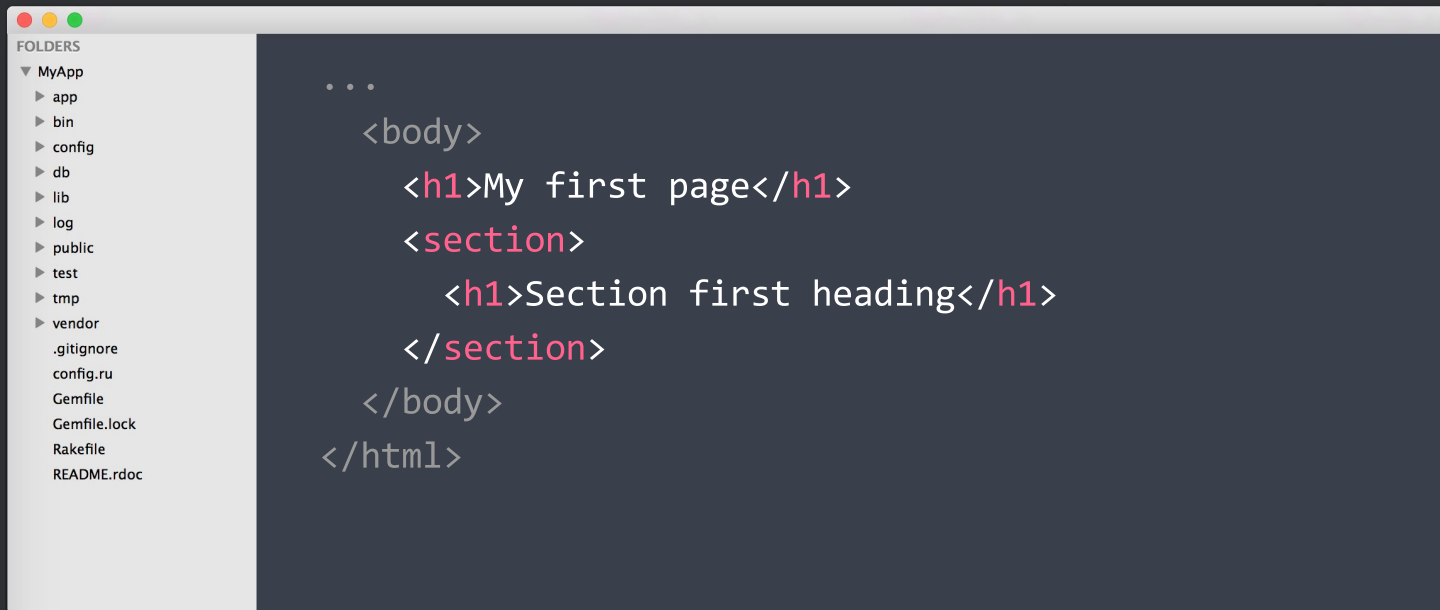


@xharekx33

# Basic tags: Headings - `<h1>` to `<h6>`

---

The `<h1>` to `<h6>` tags are used to define text-only labels for sections of content on the page, defined implicitly or explicitly.



The screenshot shows a code editor window with a sidebar on the left displaying a file tree under the name 'MyApp'. The main editor area contains the following HTML code:

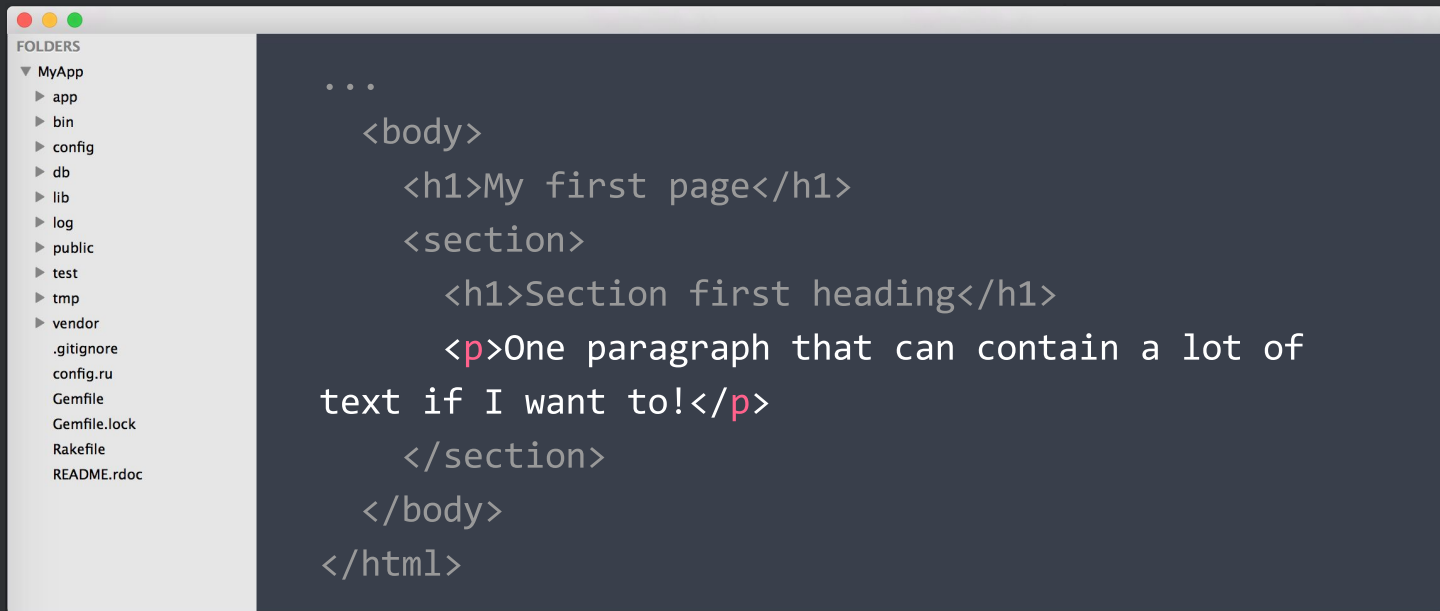
```
...  
<body>  
  <h1>My first page</h1>  
  <section>  
    <h1>Section first heading</h1>  
  </section>  
</body>  
</html>
```



@xharekx33

# Basic tags: Paragraphs - `<p>`

Used to define paragraphs. Browsers will automatically add space before and after to separate them from the rest of content.



The image shows a code editor window with a sidebar on the left and a main editing area. The sidebar, titled 'FOLDERS', shows a tree structure with 'MyApp' expanded, containing subfolders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', along with files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area contains the following HTML code:

```
...  
  <body>  
    <h1>My first page</h1>  
    <section>  
      <h1>Section first heading</h1>  
      <p>One paragraph that can contain a lot of  
text if I want to!</p>  
    </section>  
  </body>  
</html>
```

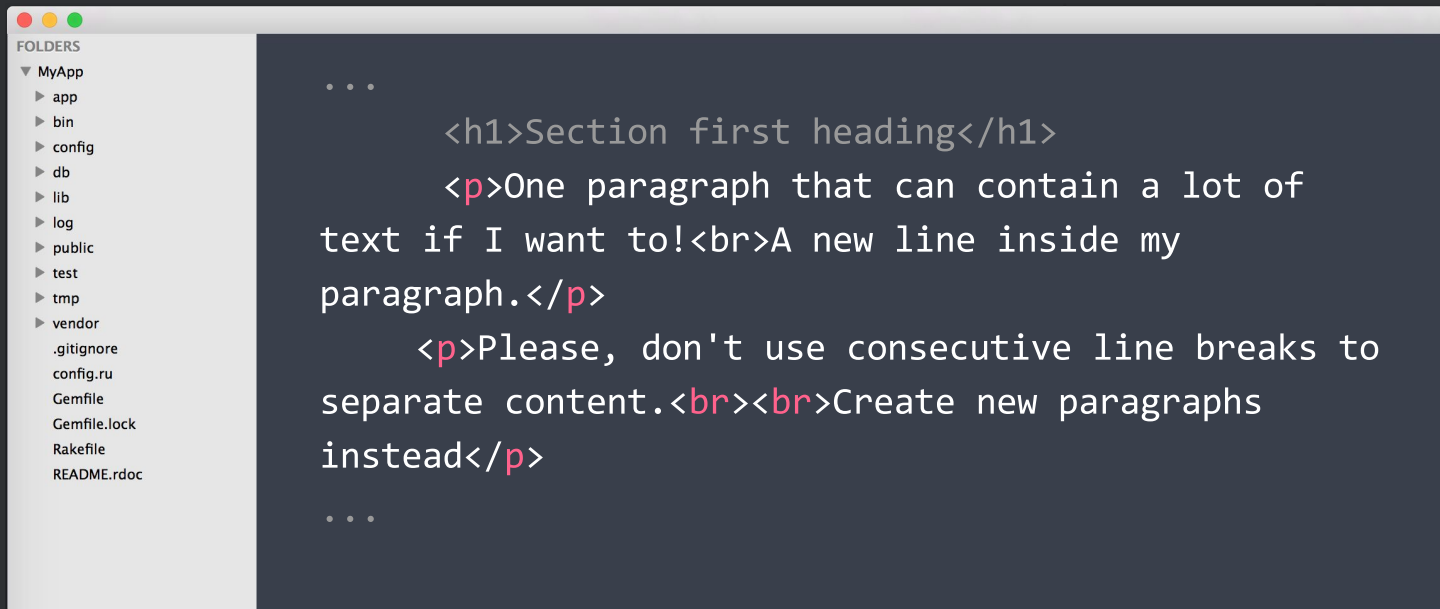


@xharekx33

# Basic tags: Line breaks - `<br>`

---

Line breaks are used to make text jump to the next line, usually for formatting purposes.



The screenshot shows a code editor window with a sidebar on the left listing folders and files. The main editor area contains HTML code demonstrating the use of the `<br>` tag. The code is as follows:

```
...  
<h1>Section first heading</h1>  
<p>One paragraph that can contain a lot of  
text if I want to!<br>A new line inside my  
paragraph.</p>  
<p>Please, don't use consecutive line breaks to  
separate content.<br><br>Create new paragraphs  
instead</p>  
...
```

The sidebar on the left shows a folder named 'MyApp' with subfolders 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor'. Below these are files: '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'.

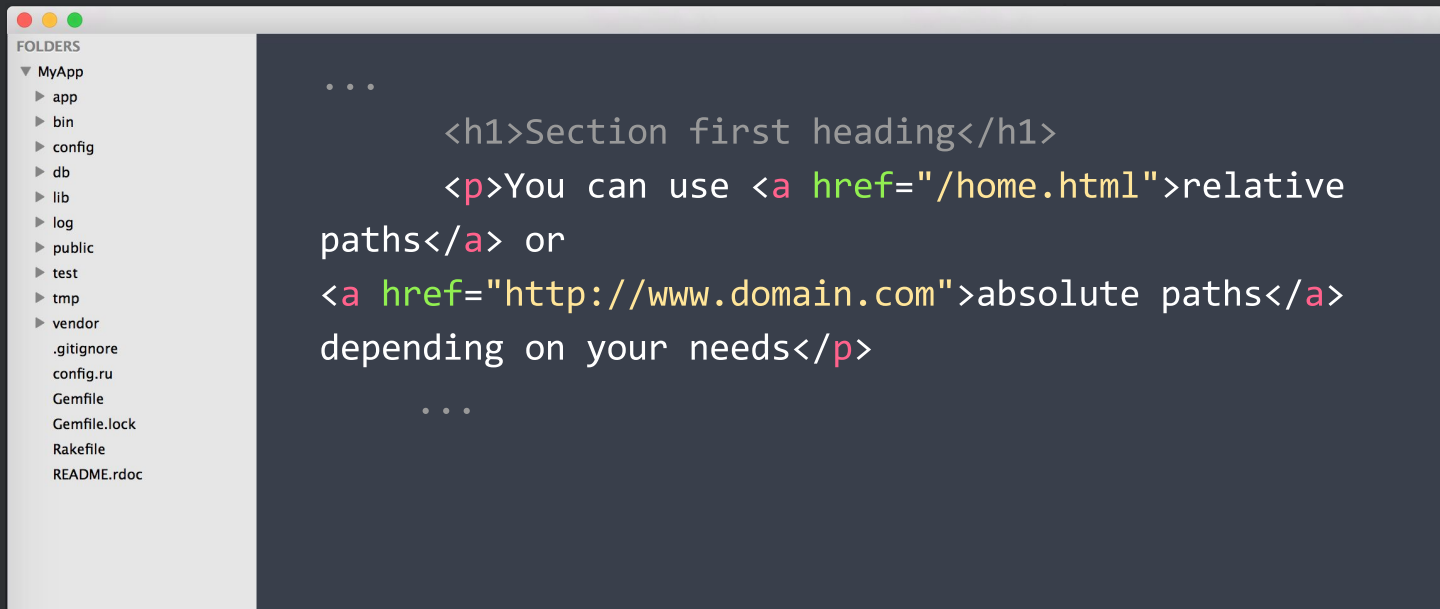


@xharekx33

# Basic tags: Links - `<a>`

---

The cornerstone of the web. They create hypertext links.

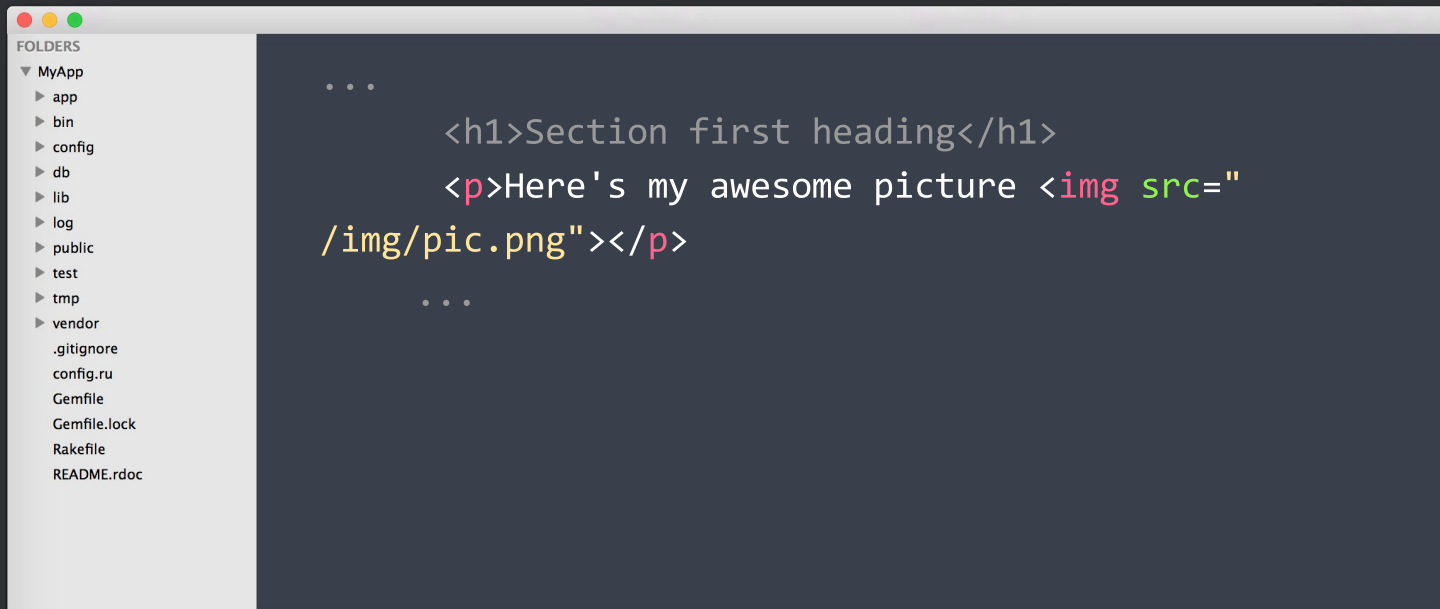


@xharekx33

# Basic tags: Images - `<img>`

---

Defines an image in the page.



The screenshot shows a code editor window with a sidebar on the left and a main editing area. The sidebar, titled 'FOLDERS', shows a project structure with a 'MyApp' folder containing subfolders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area displays HTML code with syntax highlighting. The code consists of three lines: an opening ellipsis '...', a heading tag '

# ``` ... <h1>Section first heading</h1> <p>Here's my awesome picture </p> ... ```

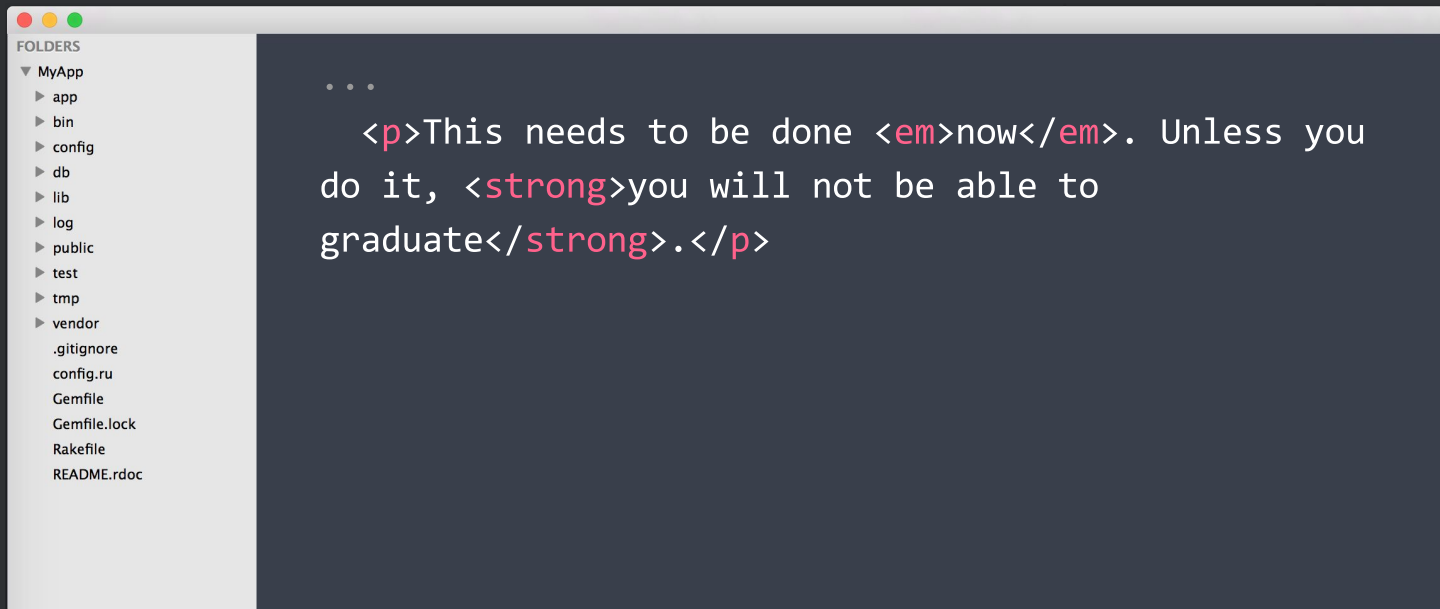


@xharekx33



# Basic tags: Emphasis - `<em>` and `<strong>`

`<em>` is used to stress emphasis of its contents and `<strong>` to stress importance.



The screenshot shows a code editor with a sidebar on the left displaying a file tree under 'FOLDERS'. The main editor area contains a code block with the following HTML structure:

```
...  
<p>This needs to be done <em>now</em>. Unless you  
do it, <strong>you will not be able to  
graduate</strong>.</p>
```

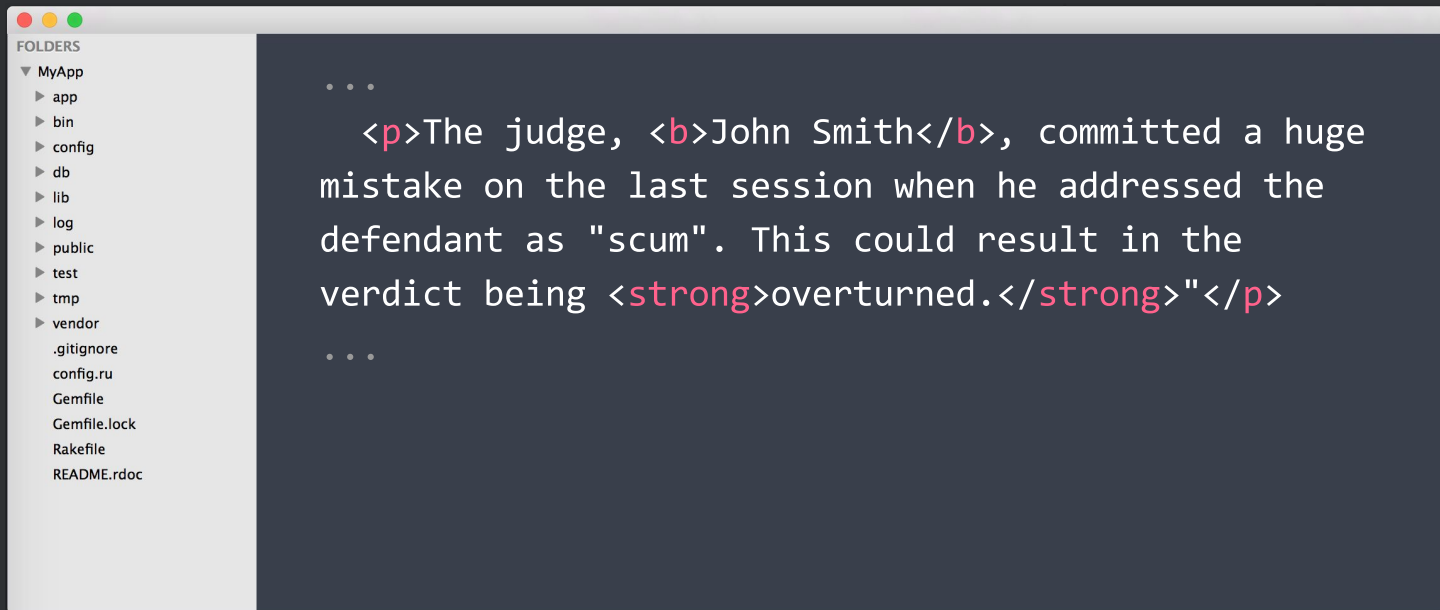


@xharekx33

# <em> and <strong> VS <i> and <b>

---

Although by default they may look the same stylistically, <i> and <b> are strictly presentational, while <em> and <strong> carry semantic meaning



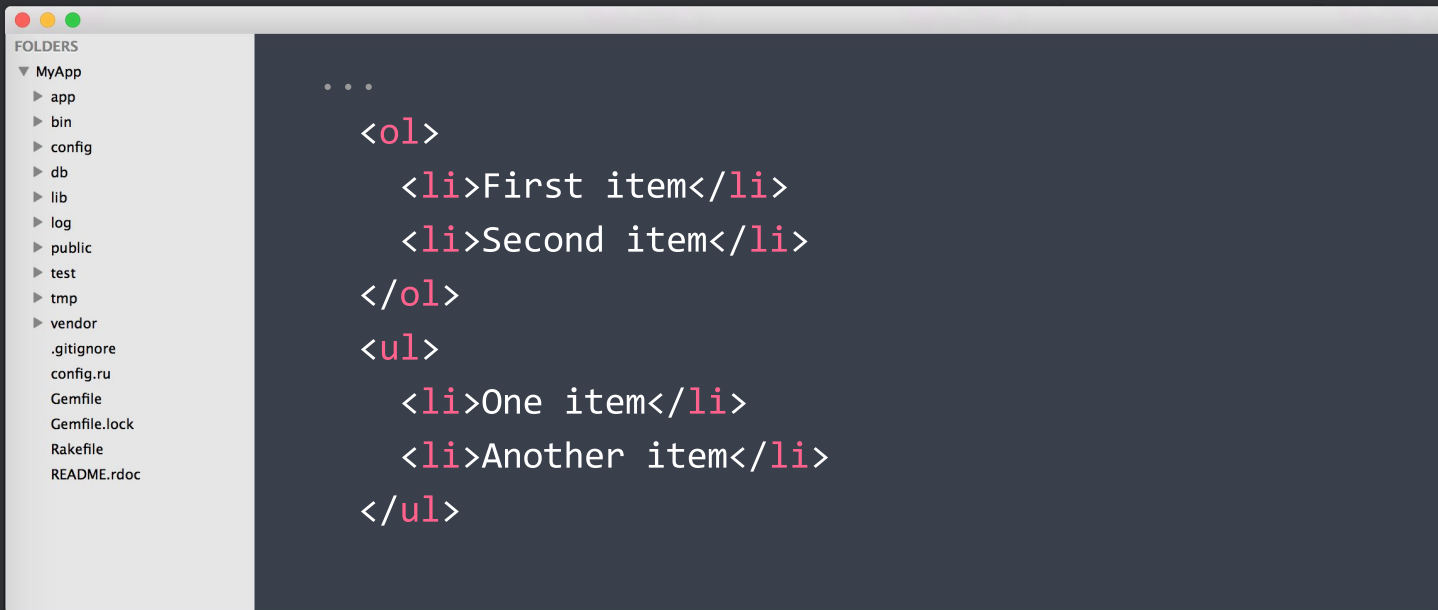
The image shows a code editor window with a sidebar on the left and a main editing area on the right. The sidebar, titled 'FOLDERS', shows a project structure with a 'MyApp' folder containing subfolders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area displays a code snippet with HTML tags. It starts with three dots, followed by a paragraph tag <p>. Inside the paragraph, the text 'The judge, 'John Smith', committed a huge mistake on the last session when he addressed the defendant as "scum". This could result in the verdict being 'overturned.' is shown. The tags <b> and </b> are used for the name 'John Smith', and <strong> and </strong> are used for the word 'overturned'. The paragraph tag is closed with </p>. The snippet ends with three dots.

```
...  
  <p>The judge, <b>John Smith</b>, committed a huge  
mistake on the last session when he addressed the  
defendant as "scum". This could result in the  
verdict being <strong>overturned.</strong>"</p>  
...
```



# Basic tags: Lists - `<ol>` and `<ul>`

Define ordered and unordered lists. Use this instead of adding numbers or bullet points by hand.



The image shows a code editor window with a sidebar on the left and a main editing area on the right. The sidebar, titled 'FOLDERS', shows a project structure for 'MyApp' with subfolders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor'. It also lists files: '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area contains the following HTML code:

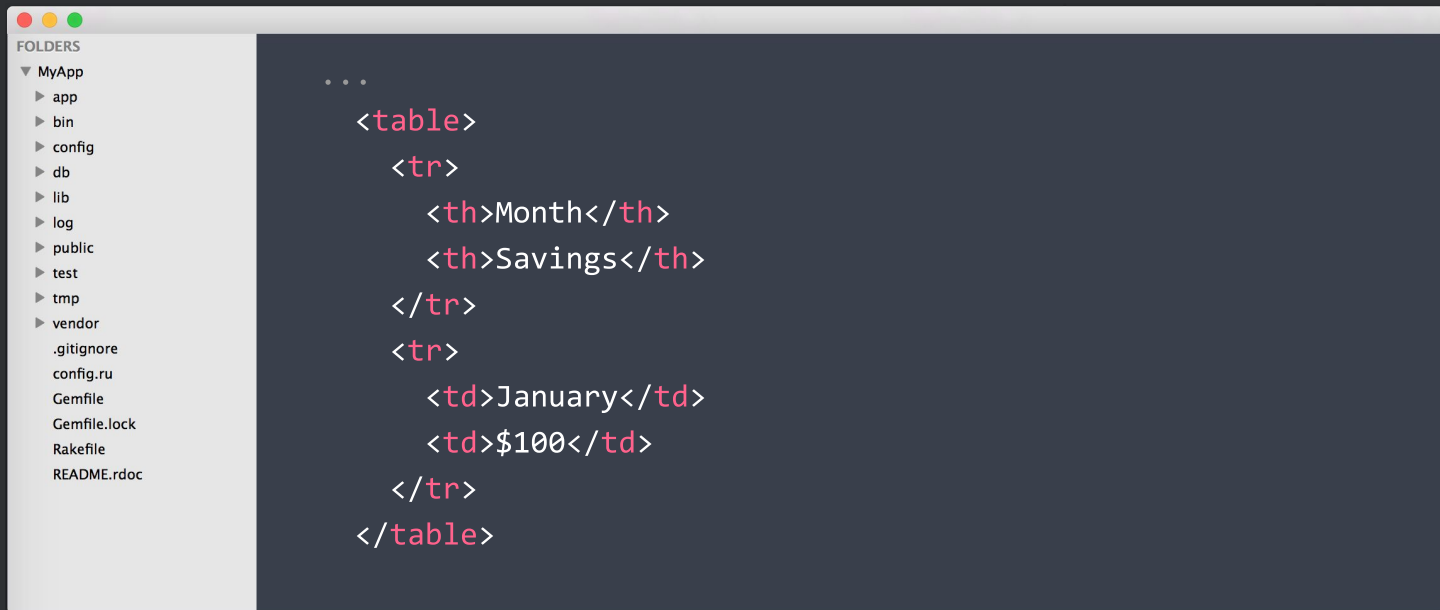
```
...  
<ol>  
  <li>First item</li>  
  <li>Second item</li>  
</ol>  
<ul>  
  <li>One item</li>  
  <li>Another item</li>  
</ul>
```



@xharekx33

# Basic tags: Tables - `<table>`

Defines a table to represent tabular data. DON'T use them to create layouts! This is not the 1990s!

A screenshot of a code editor window. On the left is a sidebar with a file explorer titled 'FOLDERS'. It shows a project named 'MyApp' with subfolders 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor'. Below these are files: '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area shows a code snippet for an HTML table. The code is: 

```
...  
<table>  
  <tr>  
    <th>Month</th>  
    <th>Savings</th>  
  </tr>  
  <tr>  
    <td>January</td>  
    <td>$100</td>  
  </tr>  
</table>
```

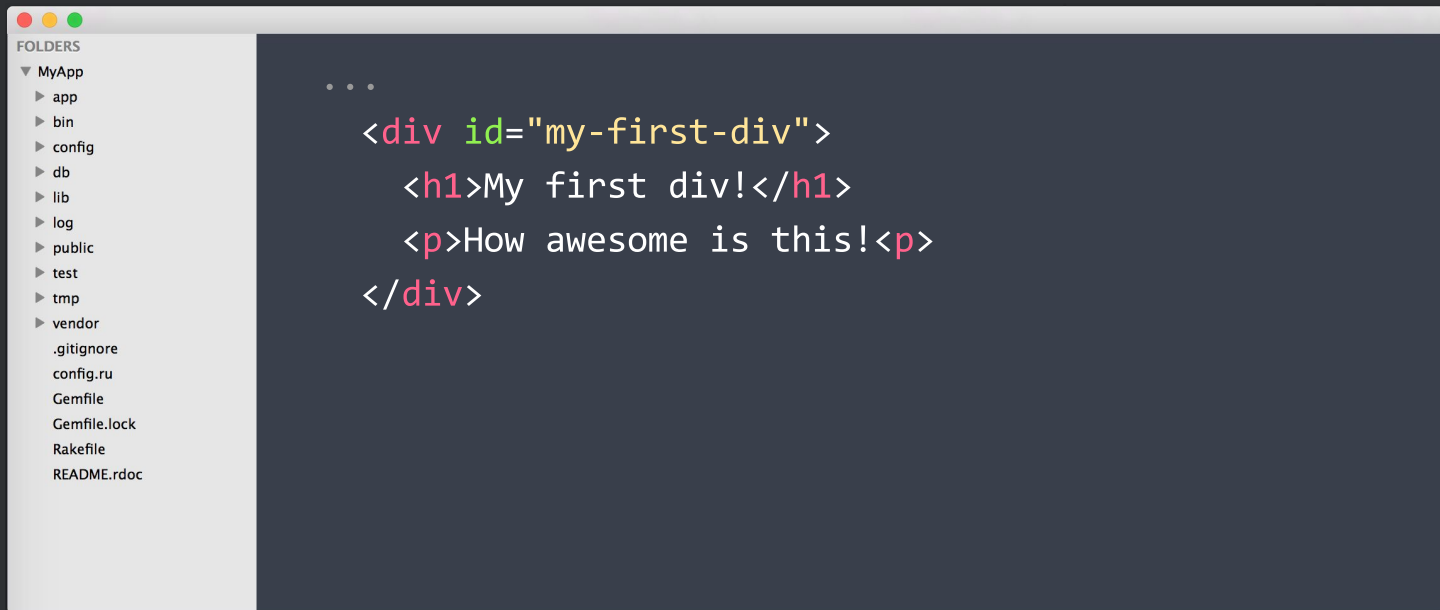


@xharekx33

# Basic tags: Divs - `<div>`

---

Defines a division or a section in an HTML document. It's a meaningless container used to group block elements.



The screenshot shows a code editor window with a sidebar on the left and a main editing area on the right. The sidebar, titled 'FOLDERS', shows a project structure with a 'MyApp' folder containing subfolders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editing area displays the following HTML code:

```
...  
<div id="my-first-div">  
  <h1>My first div!</h1>  
  <p>How awesome is this!<p>  
</div>
```

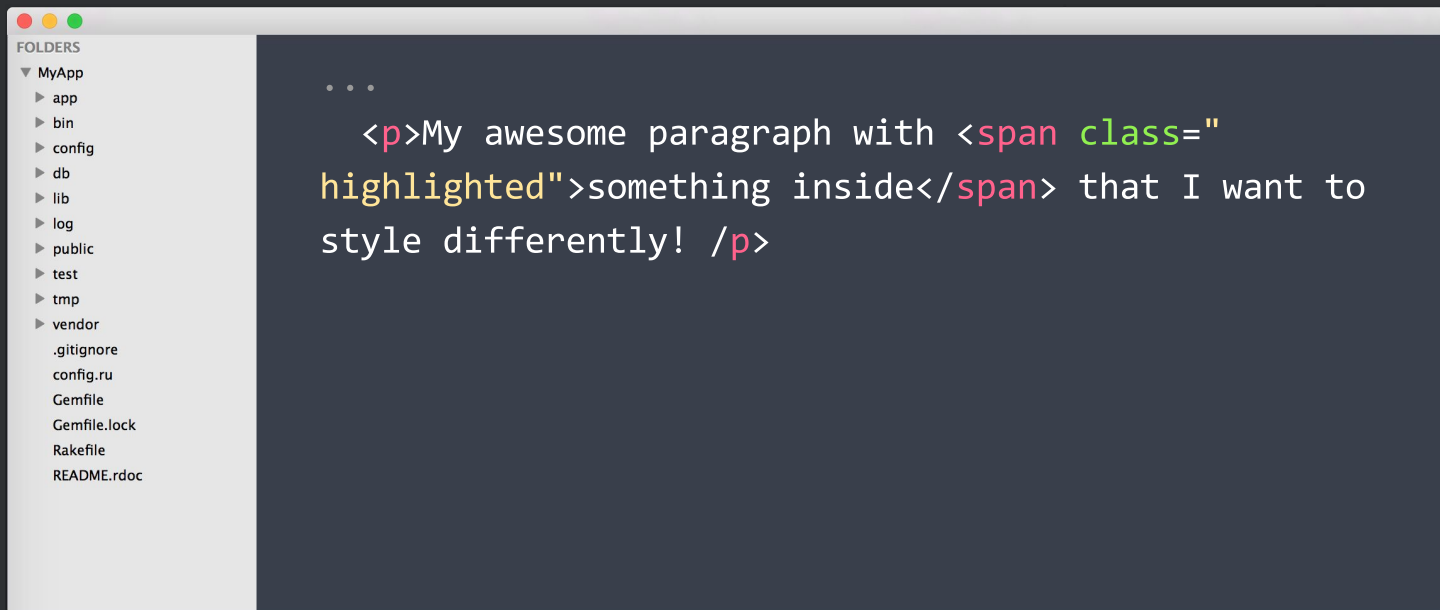


@xharekx33

# Basic tags: Spans - `<span>`

---

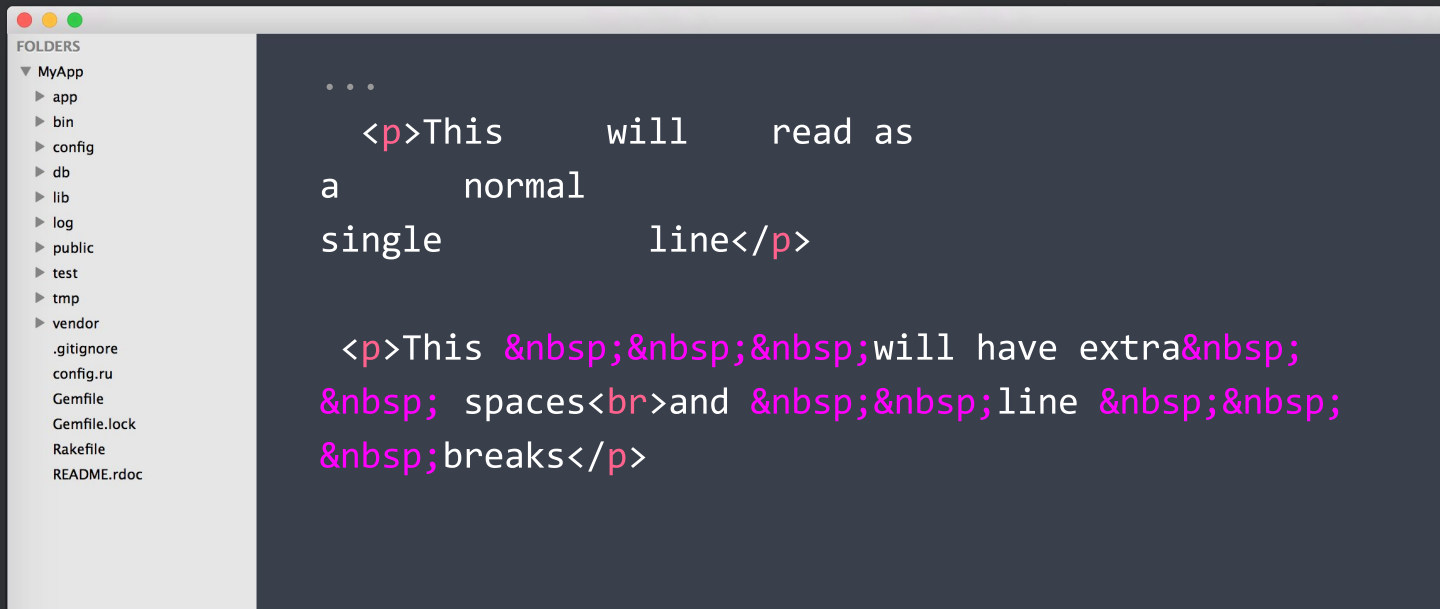
It is used to group inline-elements in a document for styling purposes or because they share common attributes.



@xharekx33

# HTML & whitespace: `&nbsp;`;

HTML collapses consecutive spaces, tabs, and new lines into a single space. If you need to add whitespace use `&nbsp;` (non-breaking space).

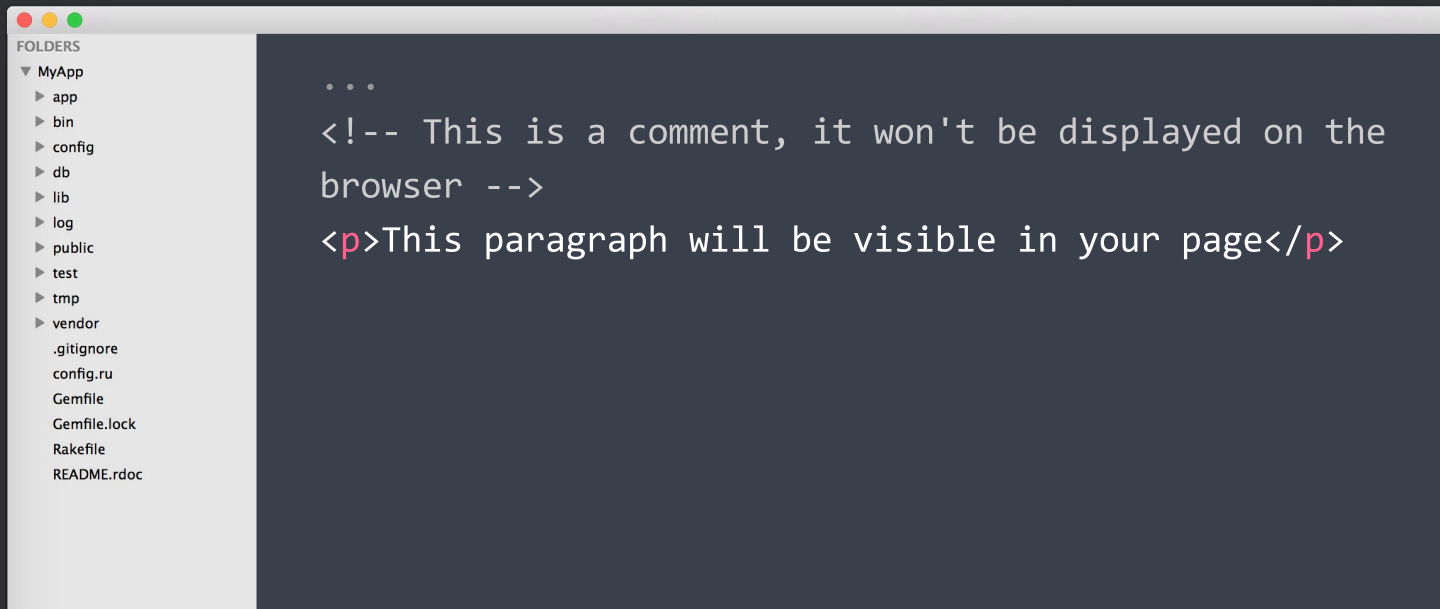


@xharekx33

# HTML comments

---

You can add comments to your code, to take notes, clarify stuff, but they will not be displayed on the page.



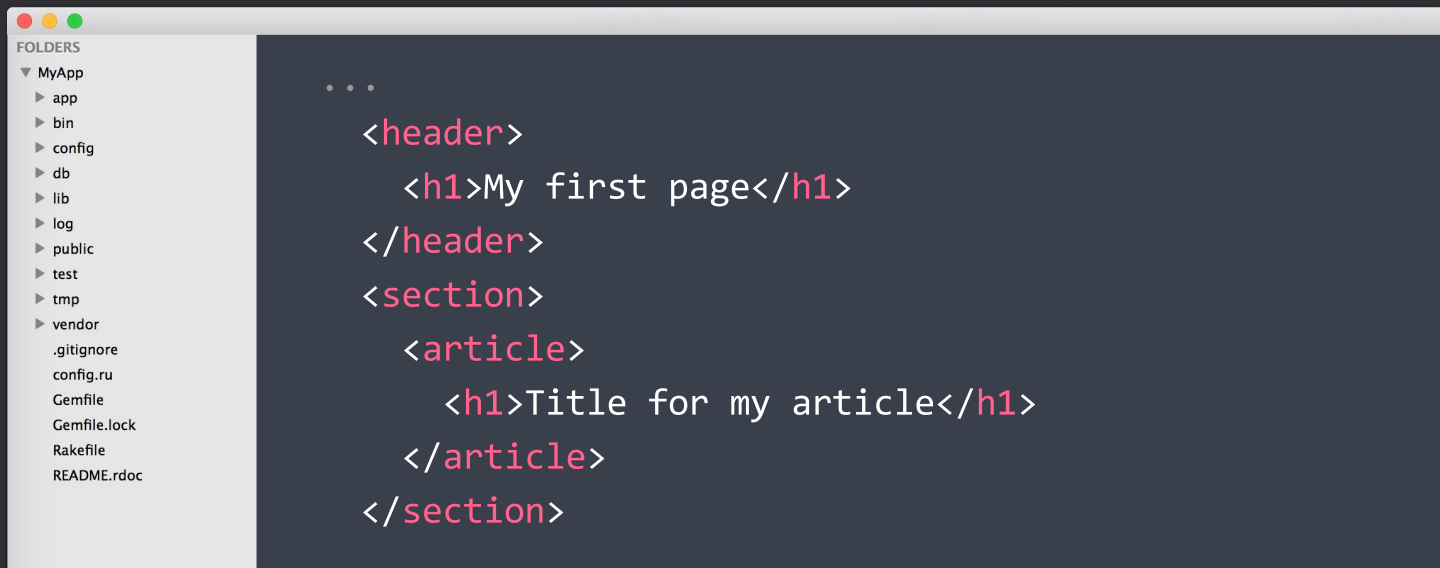
@xharekx33



# HTML5 document structure

---

HTML5 provides a lot of tags to indicate structure and provide semantic meaning to your page: `<section>`, `<header>`, `<footer>`, `<aside>`, `<nav>` and `<article>`. Better than simple divs.

A screenshot of a code editor interface. On the left, a sidebar titled 'FOLDERS' shows a tree structure for 'MyApp' with subfolders: app, bin, config, db, lib, log, public, test, tmp, and vendor. Below these are files: .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc. The main editor area displays HTML code with syntax highlighting. The code starts with three dots, followed by a header section containing an h1 tag with the text 'My first page'. This is followed by a section containing an article with an h1 tag with the text 'Title for my article'.

```
...  
<header>  
  <h1>My first page</h1>  
</header>  
<section>  
  <article>  
    <h1>Title for my article</h1>  
  </article>  
</section>
```

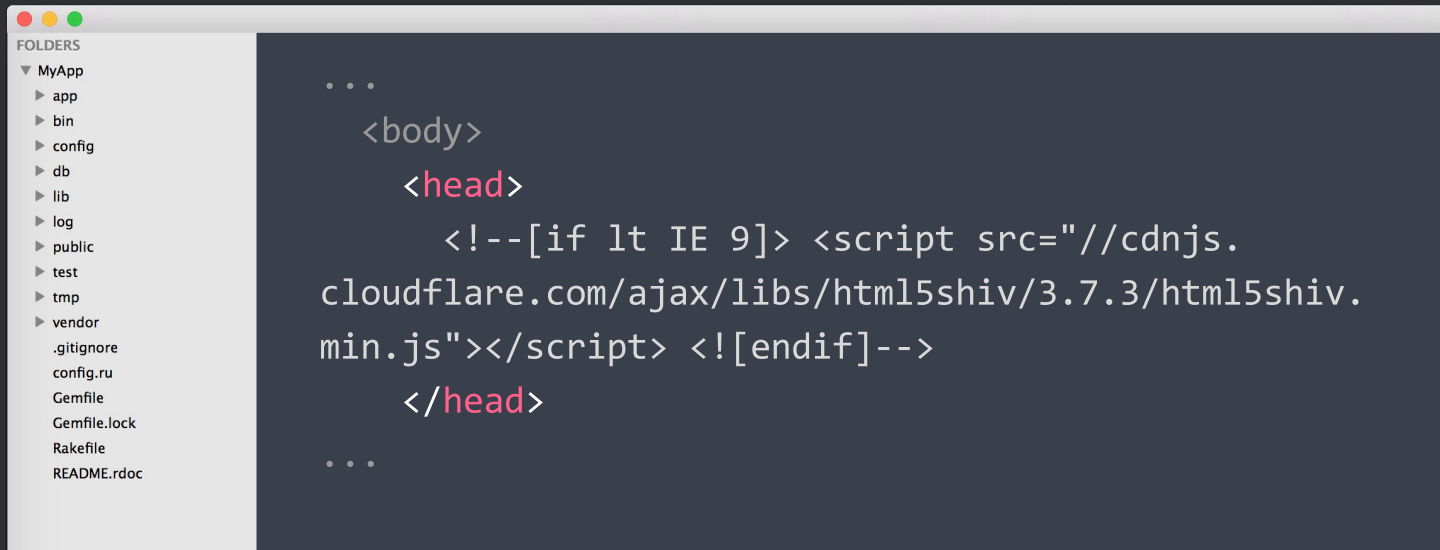


@xharekx33

# Browser support and the HTML5 shiv

---

Unfortunately old browsers do not support the use of these new HTML5 structural elements. In order to use and style them, we can include this code inside our `<head>` tag



@xharekx33

# Forms

Forms allow users to send data to the website. Using forms, the website can take input from the site visitor and send it so it can be processed by a back-end application

A screenshot of a code editor window. On the left is a sidebar with a 'FOLDERS' section containing a tree view of a project named 'MyApp'. The tree includes folders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area shows a snippet of HTML code for a form, with three dots at the top left of the code block.

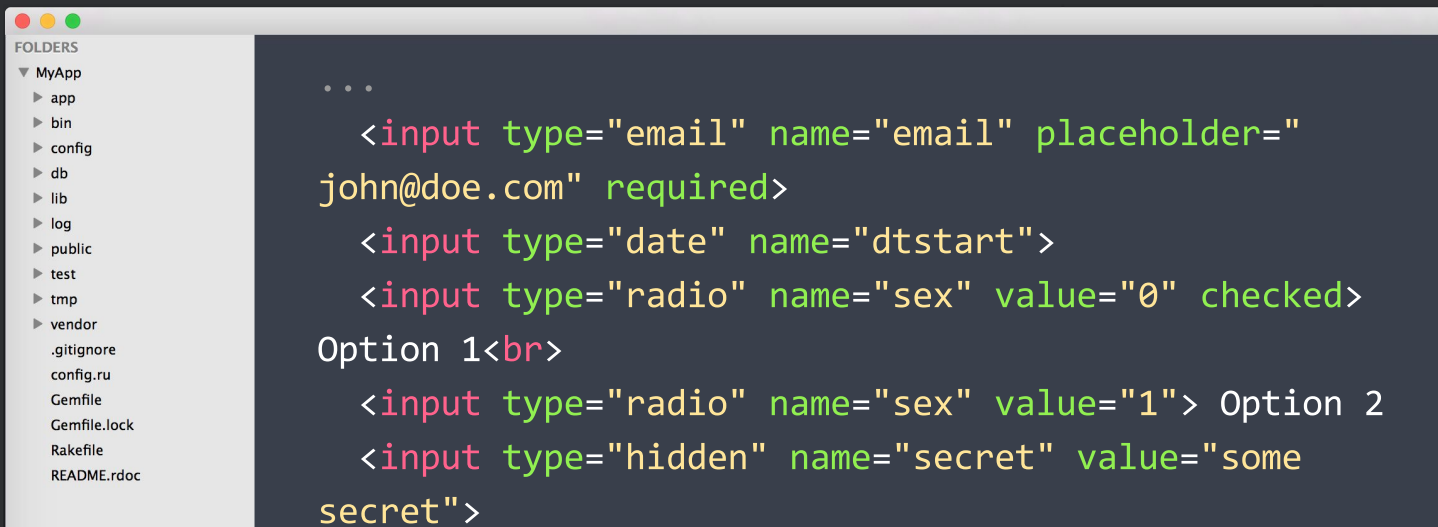
```
...  
<form>  
  <label for="my_field">My field</label>  
  <input type="text" name="my_field">  
  <input type="submit" value="Save">  
</form>
```



@xharekx33

# Forms: field types

Forms support many kinds of different input types: text, button, checkbox, date, email, file, hidden, password, radio, search, submit...



The image shows a code editor window with a sidebar on the left displaying a file tree under 'FOLDERS' with 'MyApp' expanded. The main editor area contains HTML code for various input types. The code is as follows:

```
...  
<input type="email" name="email" placeholder="john@doe.com" required>  
<input type="date" name="dtstart">  
<input type="radio" name="sex" value="0" checked>  
Option 1<br>  
<input type="radio" name="sex" value="1"> Option 2  
<input type="hidden" name="secret" value="some secret">
```



@xharekx33

# HTML and backwards compatibility

---

The good and bad thing about HTML is how flexible and backwards compatible it is. The browser will make its best attempt to render ANYTHING, no matter how mangled or outdated it is.

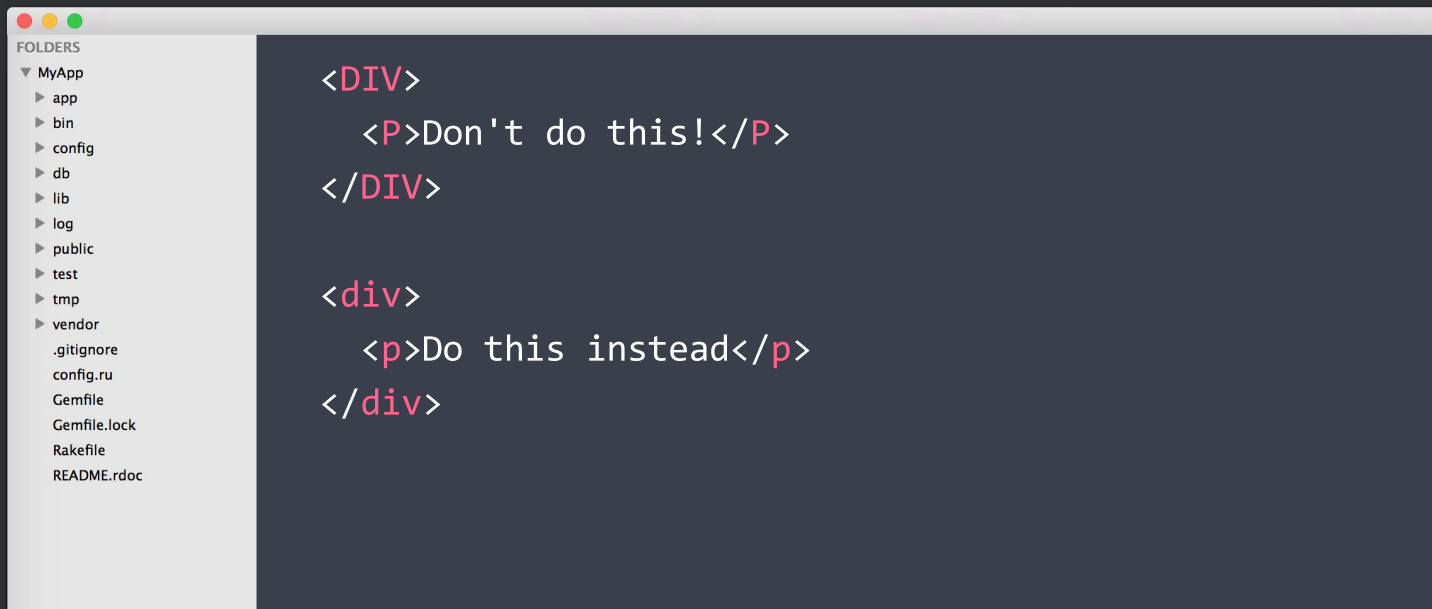
Part of this clusterfuck of spotty implementations, different coding styles and inconsistent rendering between browsers is a product of the history of HTML and its evolution: HTML 2.0 to HTML 4.01, XHTML 1 & 2, HTML5, the IETF, the W3C, the WHATWG...



# Best practices

---

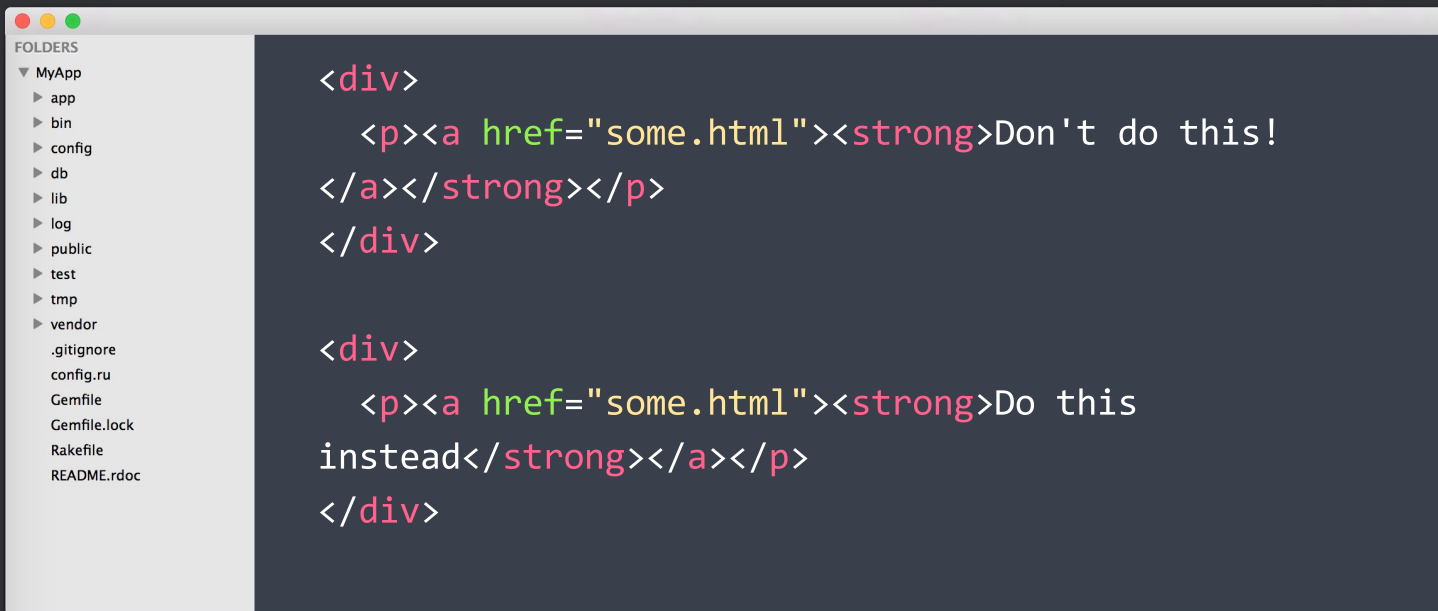
Use lowercase for all your elements



@xharekx33

# Best practices

Use proper nesting!



**FOLDERS**

- ▼ MyApp
  - ▶ app
  - ▶ bin
  - ▶ config
  - ▶ db
  - ▶ lib
  - ▶ log
  - ▶ public
  - ▶ test
  - ▶ tmp
  - ▶ vendor
  - .gitignore
  - config.ru
  - Gemfile
  - Gemfile.lock
  - Rakefile
  - README.rdoc

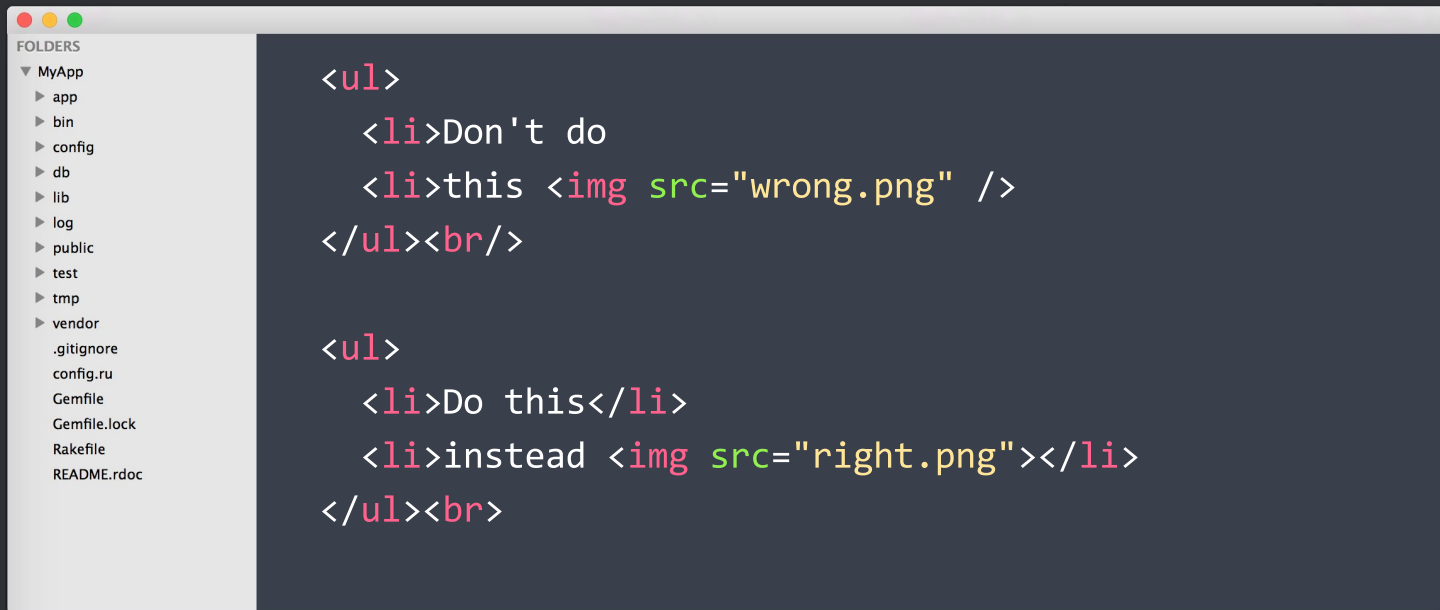
```
<div>  
  <p><a href="some.html"><strong>Don't do this!  
</a></strong></p>  
</div>  
  
<div>  
  <p><a href="some.html"><strong>Do this  
instead</strong></a></p>  
</div>
```



@xharekx33

# Best practices

Always close your tags, but don't use trailing slashes on self-closing elements

A screenshot of a code editor with a dark theme. On the left is a sidebar with a 'FOLDERS' section containing a tree view of a project named 'MyApp'. The tree includes folders like 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files like '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area displays two code snippets. The first snippet shows an unclosed list tag and a self-closing image tag with a trailing slash. The second snippet shows the same list tag properly closed and the image tag without a trailing slash.

```
<ul>
  <li>Don't do
  <li>this 
</ul><br/>

<ul>
  <li>Do this</li>
  <li>instead </li>
</ul><br>
```



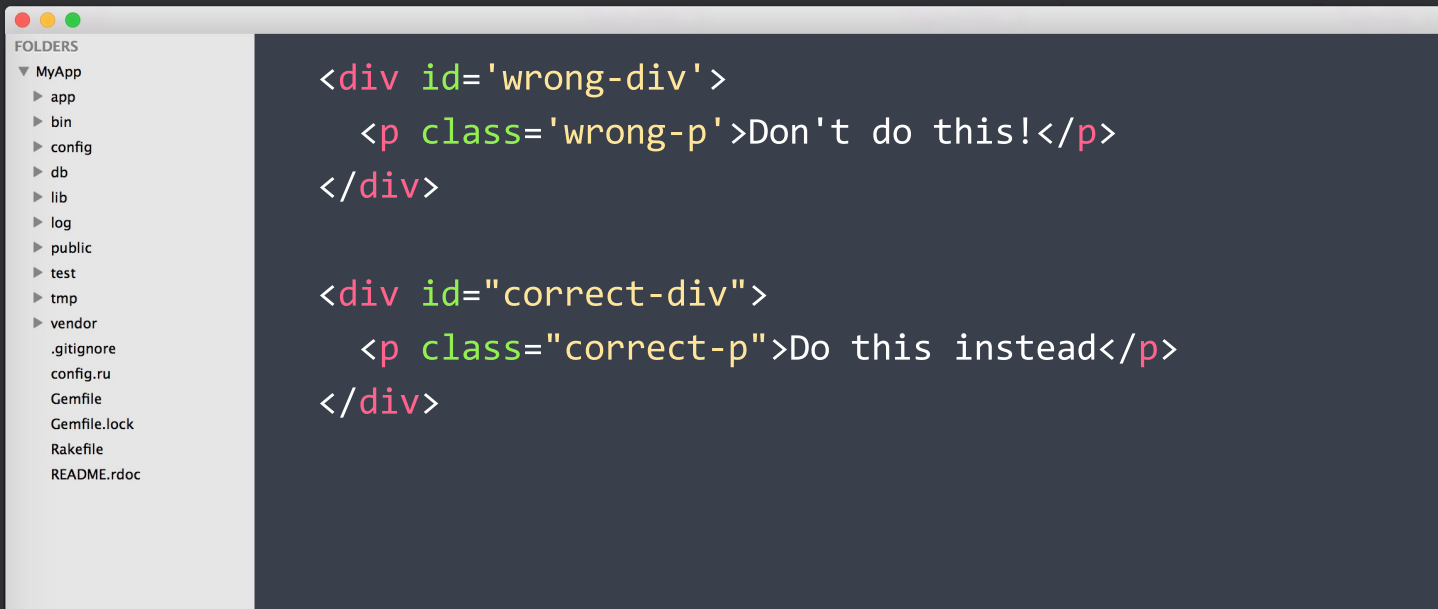
@xharekx33



# Best practices

---

Use double quotes for all your attributes.



```
<div id='wrong-div'>
  <p class='wrong-p'>Don't do this!</p>
</div>

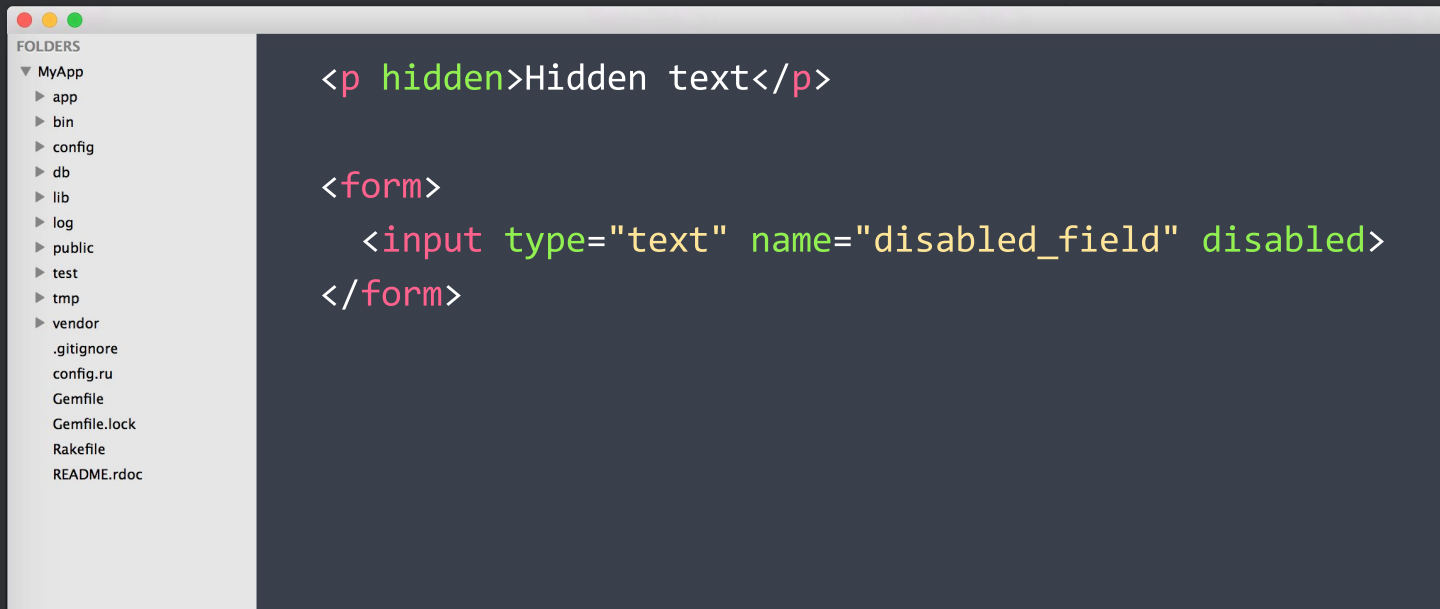
<div id="correct-div">
  <p class="correct-p">Do this instead</p>
</div>
```



@xharekx33

# Best practices

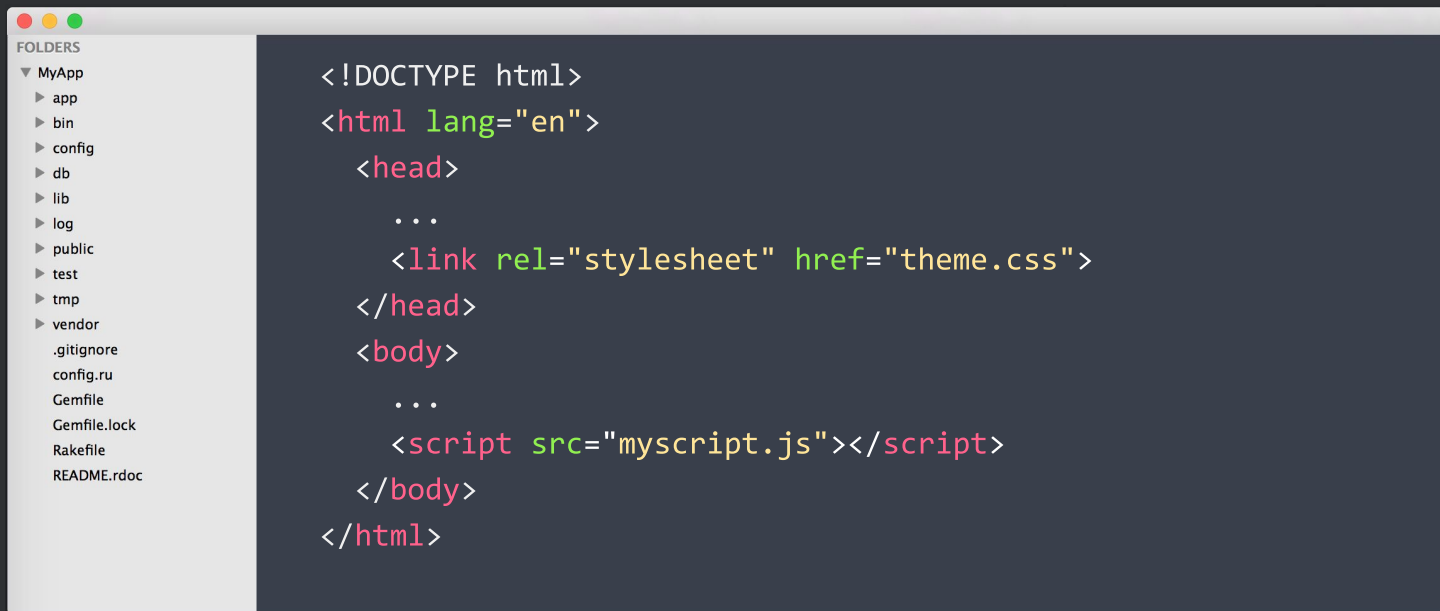
Boolean attributes (hidden, disabled, selected, checked...) don't need a value. Their presence is enough.



@xharekx33

# Best practices

Don't use inline css and load your javascript at the end.

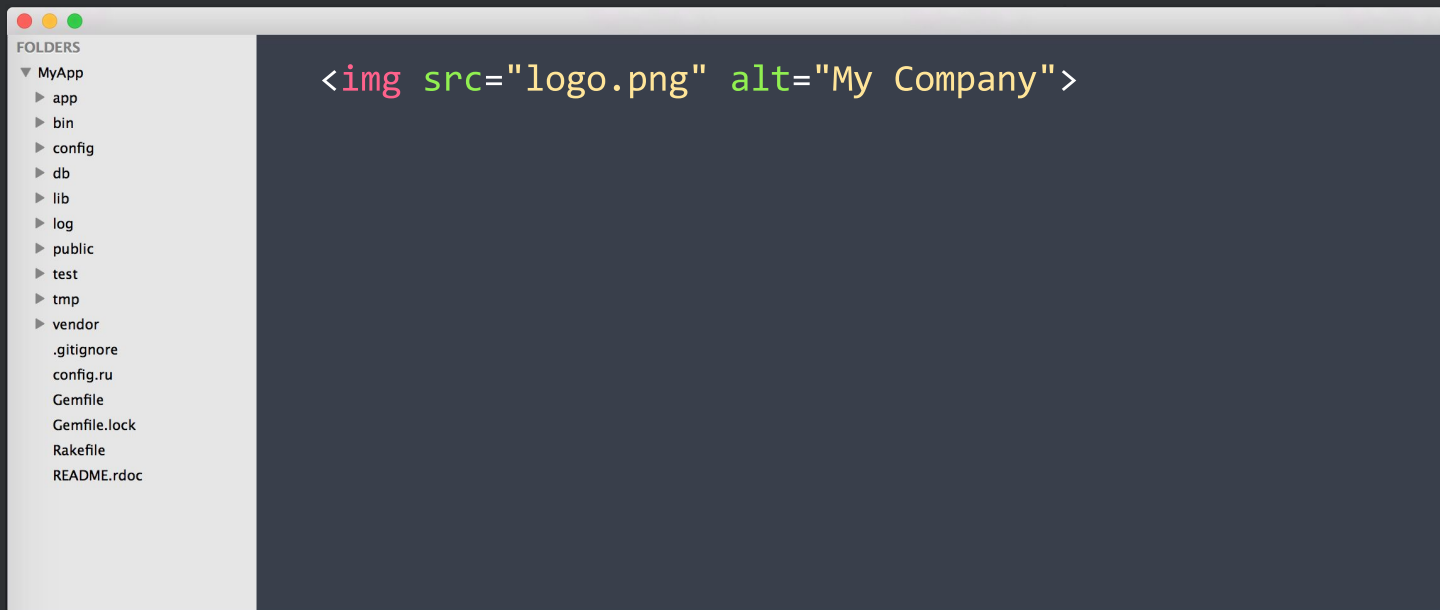


@xharekx33

# Best practices

---

Use alt texts with images for screen readers and proper indexing.

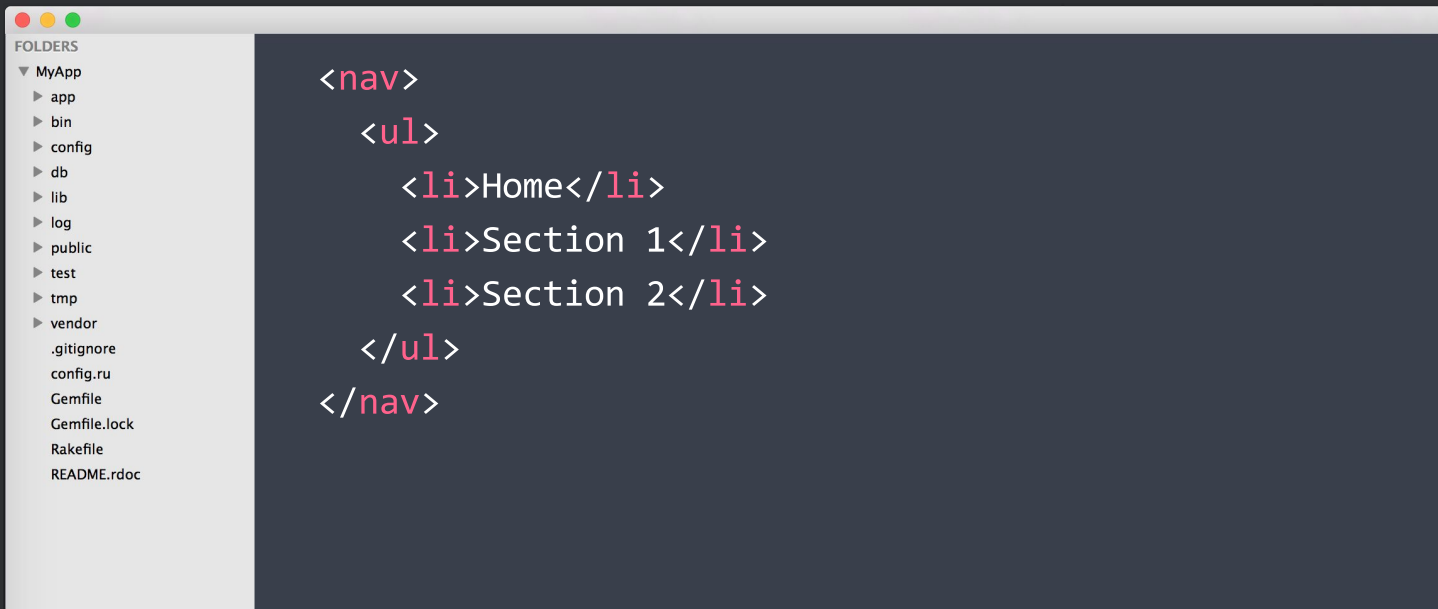


@xharekx33

# Best practices

---

Use ul for navigation.

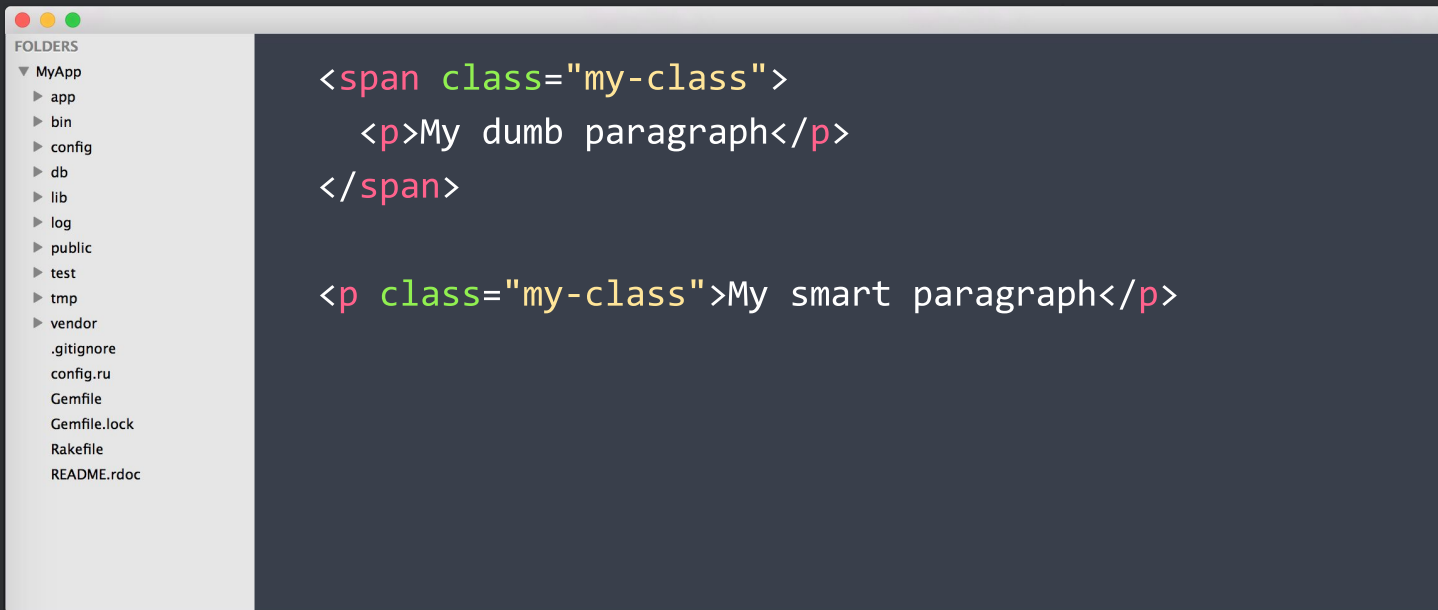


@xharekx33

# Best practices

---

Don't use superfluous parent elements.



The screenshot shows a code editor interface. On the left is a file explorer with the following structure:

- FOLDERS
  - ▼ MyApp
    - ▶ app
    - ▶ bin
    - ▶ config
    - ▶ db
    - ▶ lib
    - ▶ log
    - ▶ public
    - ▶ test
    - ▶ tmp
    - ▶ vendor
  - .gitignore
  - config.ru
  - Gemfile
  - Gemfile.lock
  - Rakefile
  - README.rdoc

The main code area contains two HTML snippets:

```
<span class="my-class">  
  <p>My dumb paragraph</p>  
</span>
```

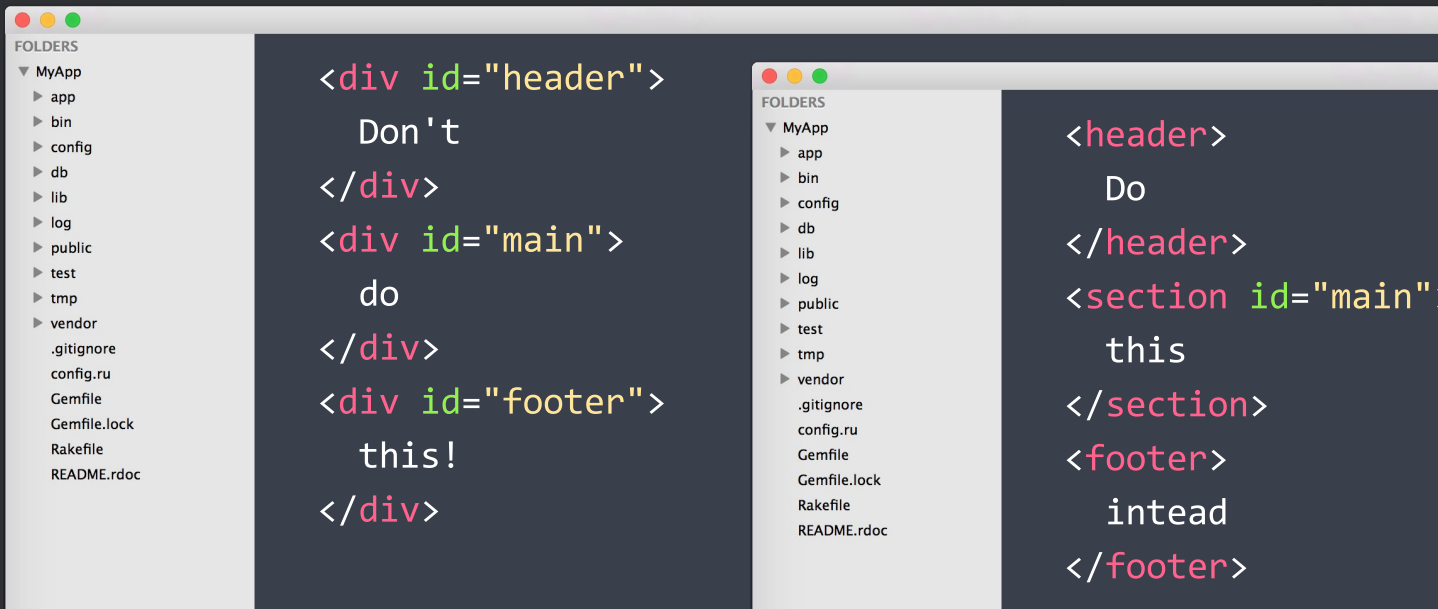
```
<p class="my-class">My smart paragraph</p>
```



@xharekx33

# Best practices

Prevent div-itis! Use the correct structural element for each case.



The image shows two side-by-side code editor windows, each with a 'FOLDERS' sidebar on the left. The sidebar lists a project structure: MyApp, app, bin, config, db, lib, log, public, test, tmp, vendor, .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc.

**Left Editor:** The main code area contains the following HTML structure:

```
<div id="header">
  Don't
</div>
<div id="main">
  do
</div>
<div id="footer">
  this!
</div>
```

**Right Editor:** The main code area contains the following HTML structure:

```
<header>
  Do
</header>
<section id="main">
  this
</section>
<footer>
  intead
</footer>
```

# Exercise

Create markup for a blog post with a comment form and a couple comments:

- Use different and appropriate containers for the blog post, the comment form and the comments.
- The blog post must have: title, subtitle, date, author (with a link to their website), text, and image, a list, and some pull quotes.
- The comment form must have: name field, email field, space to write, a "remember my data" option and a submit button.

