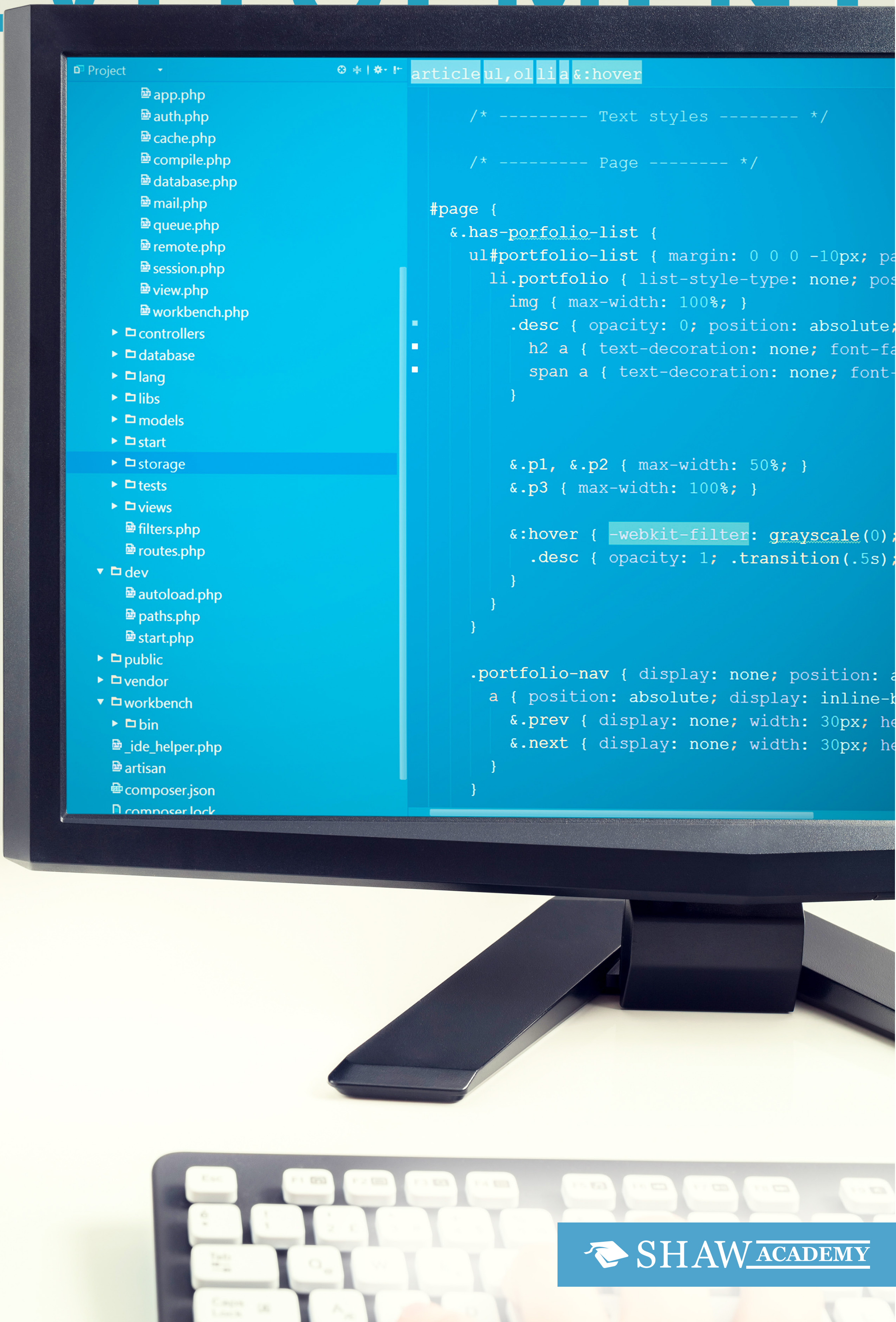
WEB DEVELOPMENT



```
Project
        app.php
        auth.php
        cache.php
        compile.php
        database.php
        mail.php
        queue.php
        remote.php
        session.php
        view.php
        workbench.php
    ▸ controllers
    ▸ database
    ▸ lang
    ▸ libs
    ▸ models
    ▸ start
    ▸ storage
    ▸ tests
    ▸ views
      filters.php
      routes.php
  ▾ dev
      autoload.php
      paths.php
      start.php
  ▸ public
  ▸ vendor
  ▾ workbench
    ▸ bin
    _ide_helper.php
    artisan
    composer.json
    composer.lock
```

```
article ul, ol li a &:hover

    /* --------- Text styles -------- */

    /* --------- Page -------- */

#page {
  &.has-porfolio-list {
    ul#portfolio-list { margin: 0 0 0 -10px; pa
      li.portfolio { list-style-type: none; pos
        img { max-width: 100%; }
        .desc { opacity: 0; position: absolute;
          h2 a { text-decoration: none; font-fa
          span a { text-decoration: none; font-
        }


        &.p1, &.p2 { max-width: 50%; }
        &.p3 { max-width: 100%; }

        &:hover { -webkit-filter: grayscale(0);
          .desc { opacity: 1; .transition(.5s);
        }
      }
    }

    .portfolio-nav { display: none; position: a
      a { position: absolute; display: inline-b
        &.prev { display: none; width: 30px; he
        &.next { display: none; width: 30px; he
      }
    }
```

SHAW ACADEMY

GETTING STARTED

In Lesson 2 you will be shown your text editor for the course

```
This is my first web page
```

Now create a folder called "html" wherever you like to save files on your computer and save the file as "myfirstpage.html".

Be careful. It is important that the extension ".html" is specified

To look at HTML files, they don't even need to be on the web. Open a web browser such as Chrome, Firefox, Safari or Internet Explorer and in the address bar, where you usually type web addresses, type in the location of the file you just saved (for example, "c:\html\myfirstpage.html") and hit return. Alternatively, go to the File menu of the browser, select Open, and browse for the file.

Pow. There it is. Your first web page. How exciting. And all it took was a few typed words.

TAGS, ATTRIBUTES & ELEMENTS

Tags

The basic structure of an HTML document includes tags, which surround content and apply meaning to it.

Change your document so that it looks like this:

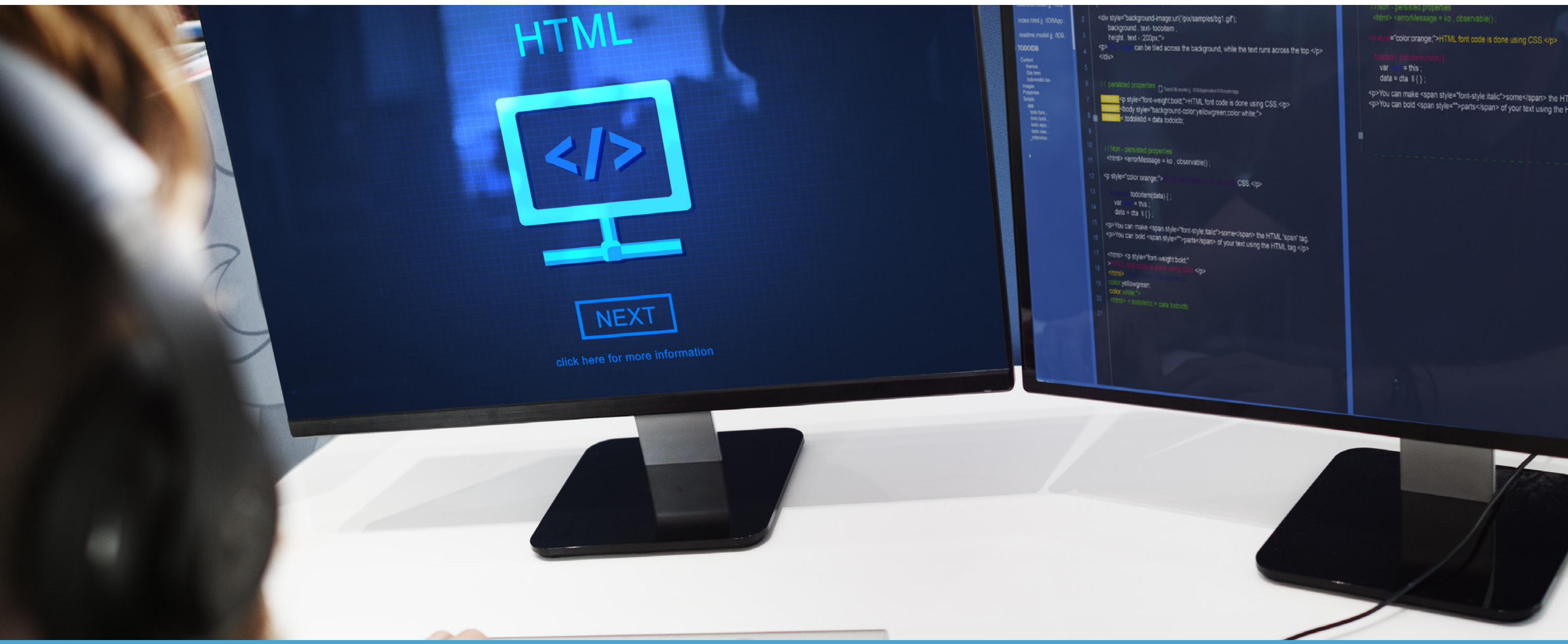
```
<!DOCTYPE html>
<html>
<body>
    This is my first web page </body>
</html>
```

Now save the document again, go back to the web browser and reload the page.

The appearance of the page will not have changed at all, but the purpose of HTML is to apply meaning, not presentation, and this example has now defined some fundamental elements of a web page.

The first line on the top, `<!DOCTYPE html>`, is a **document type declaration** and it lets the browser know which flavor of HTML you're using (HTML5, in this case). It's very important to stick this in - If you don't, browsers will assume you don't really know what you're doing and act in a very peculiar way.

To get back to the point, `<html>` is the **opening tag** that kicks things off and tells the browser that everything between that and the `</html>` **closing tag** is an HTML document. The stuff between `<body>` and `</body>` is the main content of the document that will appear in the browser window.



Closing tags

The `</body>` and `</html>` put a close to their respective elements (more on elements in a moment).

Not all tags have closing tags like this (`<html></html>`) some tags, which do not wrap around content will close themselves. The line-break tag for example, looks like this : `<br>` - a line break doesn't hold any content so the tag merrily sits by its lonely self. We will come across these examples later. All you need to remember is that all tags with content between them should be closed, in the format of opening tag → content → closing tag. It isn't, strictly speaking, always a requirement, but it's a convention we're using in these tutorials because it's good practice that results in cleaner, easier to understand code. You might come across "self-closing" tags, whereby a `br` tag, for example, will look like "`<br />`" instead of simply "`<br>`". This is a remnant of XHTML, a form of HTML based on another markup language called XML. HTML5 is much more chilled out than XHTML and will be happy with either format. Some developers prefer one way, some prefer the other. We tossed a coin and decided to stick with the simpler version.

Attributes

Tags can also have **attributes**, which are extra bits of information. Attributes appear inside the opening tag and their values sit inside quotation marks. They look something like `<tag attribute="value">Margarine</tag>`. We will come across tags with attributes later.

Once again, quotation marks aren't always essential but it is a good-practice convention and uses for consistency and clarity. We suggest you do the same.

Elements

Tags tend not to do much more than mark the beginning and end of an element. Elements are the bits that make up web pages. You would say, for example, that everything that is in between (and includes) the `<body>` and `</body>` tags is the body element. As another example, whereas "`<title>`" and "`</title>`" are **tags**, "`<title>Rumple Stiltskin</title>`" is a title **element**.

Page Titles

```
<!DOCTYPE html>
<html>
<head>
    <title>My first web page</title>
</head>
<body>
    This is my first web page
</body>
</html>
```

We have added two new elements here, that start with the **head** tag and the **title** tag (and see how both of these close).

The head element (that which starts with the `<head>` opening tag and ends with the `</head>` closing tag) appears before the body element (starting with `<body>` and ending with `</body>`) and contains information **about** the page. The information in the head element does not appear in the browser window.

We will see later on that other elements can appear inside the head element, but the most important of them is the **title** element.

If you look at this document in the browser (save and reload as before), you will see that "My first web page" will appear on a tab or the title bar of the window (not the actual canvas area). The text that you put in between the title tags has become the title of the document (surprise!). If you were to add this page to your "favourites" (or "bookmarks", depending on your browser), you would see that the title is also used there.

PARAGRAPHS

Go back to your text editor and add another line to your page:

```
<!DOCTYPE html>
<html>
<head>
    <title>My first web page</title>
</head>
<body>
    This is my first web page
    How exciting
</body>
</html>
```

Look at the document in your browser.

You might have expected your document to appear as you typed it, on two lines, but instead you should see something like this:

This is my first web page how exciting.

This is because web browsers don't usually take any notice of what line your code is on. It also doesn't take any notice of spaces (you would get the same result if you typed "This is my first web page How exciting").

If you want text to appear on different lines or, rather, if you intend there to be two distinct blocks of text (because, remember, HTML is about meaning, not presentation), you need to explicitly state that.

The line-break tag can also be used to separate lines like this, but is not recommended.

```
<p>This is my first web page</p>
<p>How exciting</p>
```

The p tag is used for **paragraphs**.

Look at the results of this. The two lines will now appear on two lines because the browser recognizes them as separate paragraphs.

Think of the HTML content as if it were a book - with paragraphs where appropriate.

Emphasis

You can emphasize text in a paragraph using `em` (emphasis) and `strong` (strong importance).

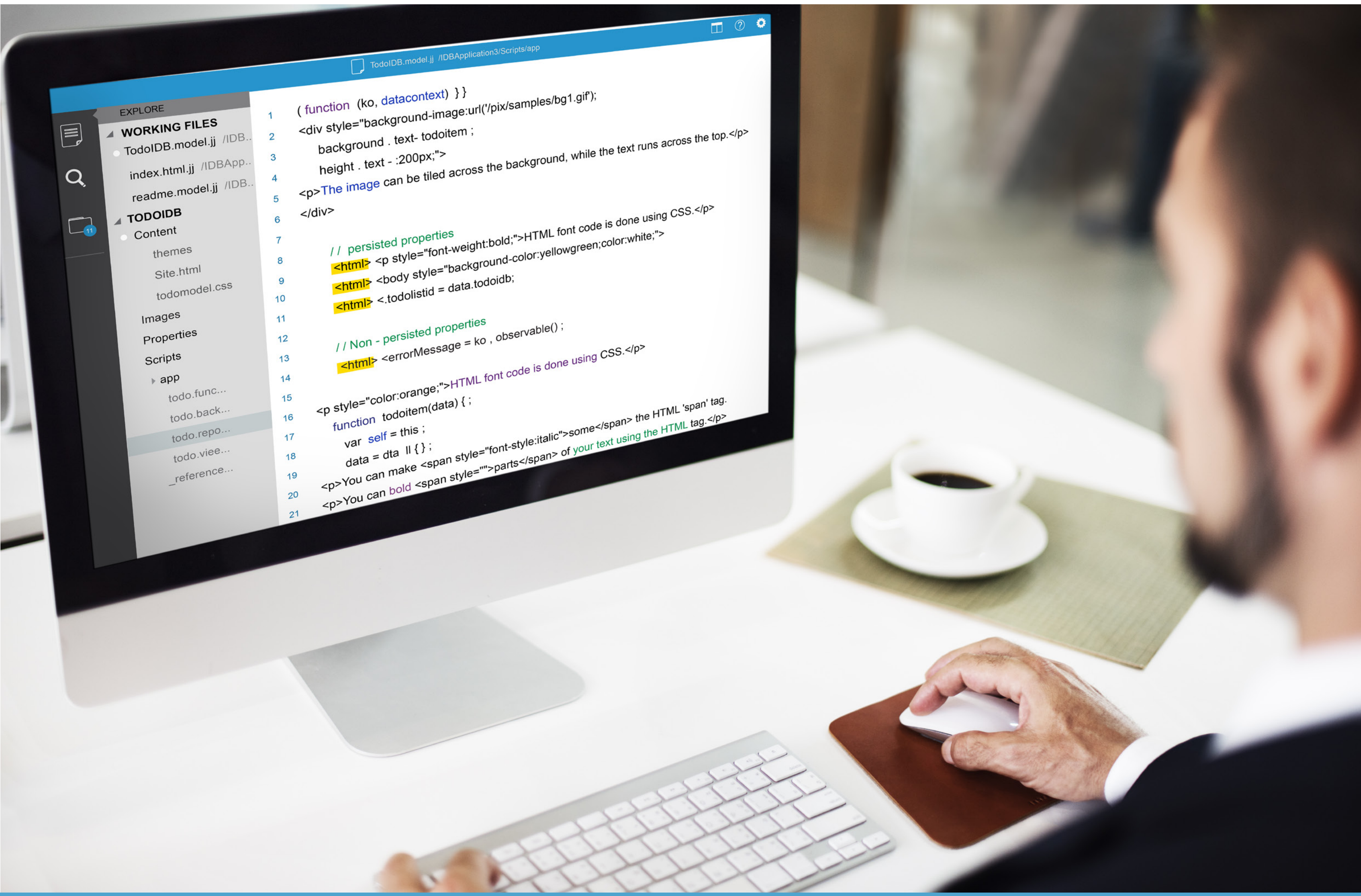
```
<p>Yes, that really <em>is</em> exciting. <strong>Warning:</strong>
level of excitment may cause you to explode.</p>
```

Traditionally, browsers will display `em` in italics and `strong` in bold by default but they are not the same as `i` and `b` tags which (aelmthough they hasvte rboenegn tenuously redefined in HTML5) have their origins in italic and bold - remember - HTML isn't for presentation. If you want to emphasize something visually (making something italic, for example), you almost certainly want to give it general emphasis. You can't speak in italics.

Emphasis

The line-break tag can also be used to separate lines like this:

```
This is my first web page<br> How exciting
```



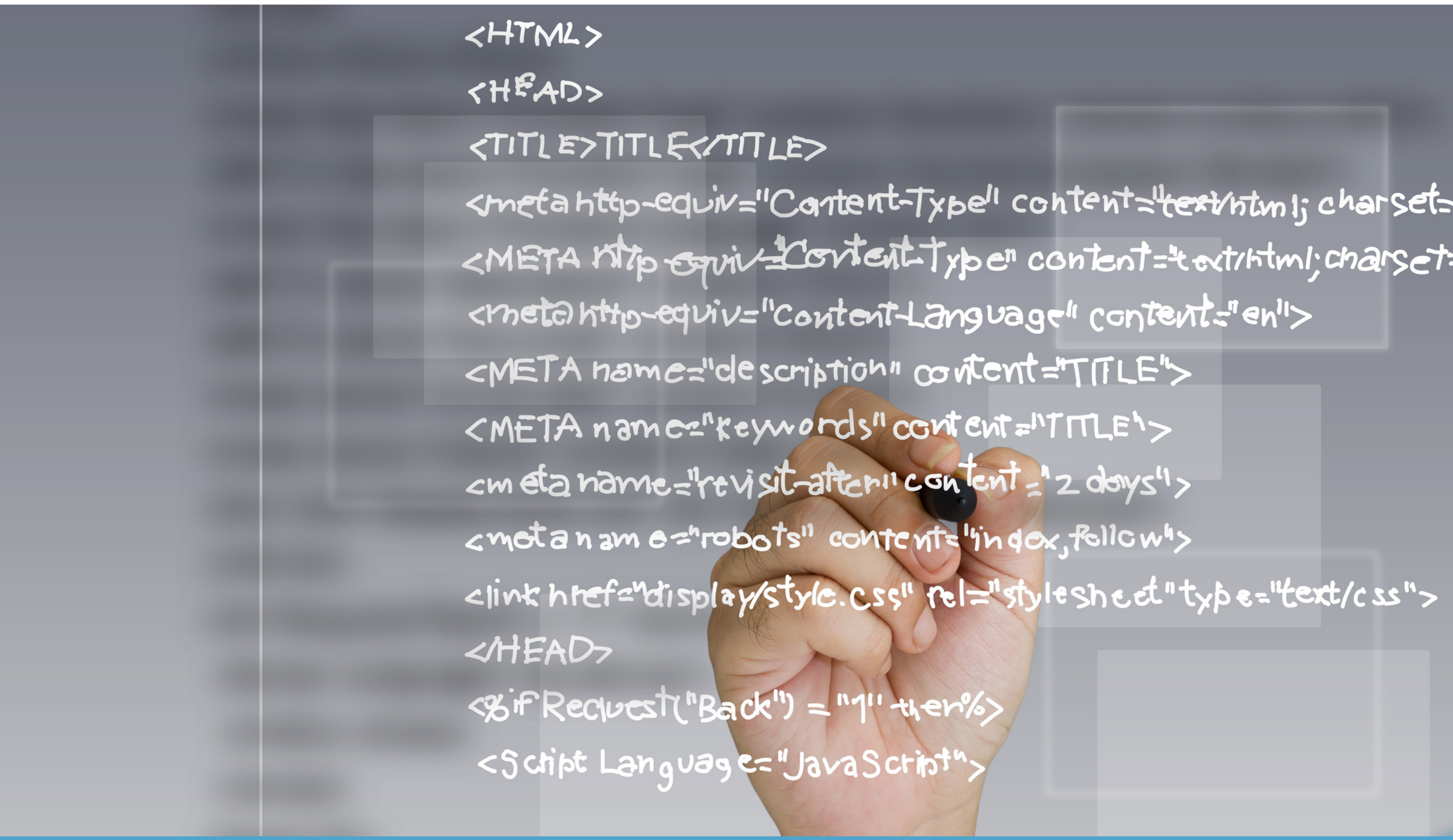
```
( function  (ko, datacontext) } }
<div style="background-image:url('/pix/samples/bg1.gif');
    background . text- todoitem ;
    height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>

    / /  persisted properties
    <html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
    <html> <body style="background-color:yellowgreen;color:white;">
    <html> <.todolistid = data.todoidb;

    / / Non - persisted properties
    <html> <errorMessage = ko , observable() ;

<p style="color:orange;">HTML font code is done using CSS.</p>
  function  todoitem(data) { ;
    var  self = this ;
    data = dta  II { } ;
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag.
<p>You can bold <span style="">parts</span> of your text using the HTML tag.</p>
```

Headings

They are h1, h2, h3, h4, h5 and h6, h1 being the almighty emperor of headings and h6 being the lowest pleb
.
Change your code to the following:

```
<!DOCTYPE html>
<html>
<head>
    <title>My first web page</title>
</head>
<body>
    <h1>My first web page</h1>
    <h2>What this is</h2>
    <p>A simple page put together using HTML</p>
    <h2>Why this is</h2>
    <p>To learn HTML</p>
</body>
</html>
```

Note that the h1 tag is only used once, as the main heading of the page. h2 to h6, however, can be used as often as desired, but they should always be used in order, as they were intended. For example, an h4 should be a sub-heading of an h3, which should be a sub-heading of an h2.



```
<HTML>
<HEAD>
<TITLE>TITLE</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=
<META http-equiv="Content-Type" content="text/html;charset=
<meta http-equiv="Content-Language" content="en">
<META name="description" content="TITLE">
<META name="keywords" content="TITLE">
<meta name="revisit-after" content="2 days">
<meta name="robots" content="index,follow">
<link href="display/style.css" rel="stylesheet" type="text/css">
</HEAD>
<%if Request("Back") = "1" then%>
<Script Language="JavaScript">
```

Lists

There are three types of list; **unordered lists**, **ordered lists** and definition lists. We will look at the first two here.

Unordered lists and ordered lists work the same way, except that the former is used for non-sequential lists with list items usually preceded by bullets and the latter is for sequential lists, which are normally represented by incremental numbers.

The `ul` tag is used to define unordered lists and the `ol` tag is used to define ordered lists. Inside the lists, the `li` tag is used to define each list item.

Change your code to the following:

```
<!DOCTYPE html>
<html>
<head>
    <title>My first web page</title>
</head>
<body>
    <h1>My first web page</h1>
    <h2>What this is</h2>
    <p>A simple page put together using HTML</p>
    <h2>Why this is</h2>
    <ul>
        </li> Because I've fallen in love with my computer
and want to give it some HTML loving. </li>
    <ul>
<body>
<html>
```

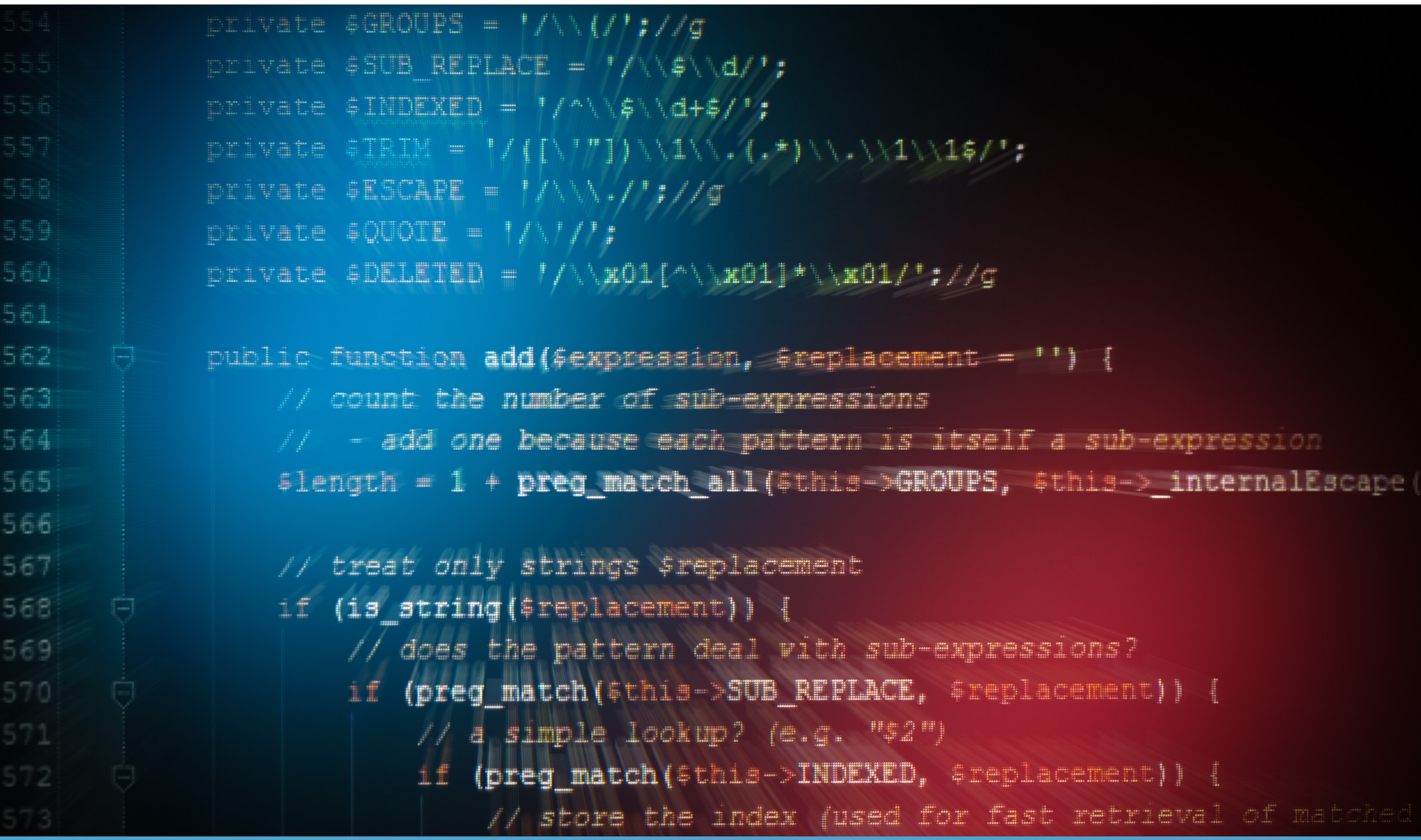
If you look at this in your browser, you will see a bulleted list. Simply change the **ul** tags to **ol** and you will see that the list will become numbered.

Lists can also be included in lists to form a structured hierarchy of items.

Replace the above list code with the following:

```
</ul> Because I’ve fallen in love with my computer and want
    <h1>My first web page</h1>
    <h1>
        To show off
        <ol>
            <li>To my boss</li>
            <li>To my friends</li>
            <li>To my cat</li>
            <li>To the little talking duck in my brain</li>
        <ol>
    <hi>
        </li> Because I’ve fallen in love with my computer
and want to give it some HTML loving. </li>
    <ul>
```

Et voilà. A list within a list. And you could put another list within that. And another within that. And so on and so forth.



```
    private $GROUPS = '/\\(/';//g
    private $SUB_REPLACE = '/\\$\\d/';
    private $INDEXED = '/^\\$\\d+$/';
    private $TRIM = '/([\'"])\\1\\.(.*)\\.\\1\\1$/';
    private $ESCAPE = '/\\./';//g
    private $QUOTE = '/\'/';
    private $DELETED = '/\\x01[^\\x01]*\\x01/';//g

    public function add($expression, $replacement = '') {
        // count the number of sub-expressions
        //  - add one because each pattern is itself a sub-expression
        $length = 1 + preg_match_all($this->GROUPS, $this->_internalEscape(

        // treat only strings $replacement
        if (is_string($replacement)) {
            // does the pattern deal with sub-expressions?
            if (preg_match($this->SUB_REPLACE, $replacement)) {
                // a simple lookup? (e.g. "$2")
                if (preg_match($this->INDEXED, $replacement)) {
                    // store the index (used for fast retrieval of matched
```

Links

So far you've been making a stand-alone web page, which is all very well and nice, but what makes the Internet so special is that it all **links** together.

The "H" and "T" in "HTML" stand for "hypertext", which basically means a system of linked text.

An **anchor** tag (a) is used to define a link, but you also need to add something to the anchor tag — the **destination** of the link.

Add this to your document:

```
<!DOCTYPE html>
<html>
<head>
    <title>My first web page</title>
</head>
<body>
    <h1>My first web page</h1>
    <h2>What this is</h2>
    <p>A simple page put together using HTML</p>
    <h2>Why this is</h2>
    <p>To learn HTML </P>
    <li> To show off</li>
    <h2>Where to find the tutorial</h2>
    <p><a href=”http://www.shawacademy.com”>Shaw</a></p>
</body>
</html>
```

The destination of the link is defined in the `href` attribute of the tag. The link can be absolute, such as "http://www.shawacademy.com", or it can be relative to the current page. So if, for example, you had another file called "flyingmoss.html" in the same directory then the line of code would simply be `<a href="flyingmoss.html">The miracle of moss in flight</a>` or something like this.

A link does not have to link to another HTML file, it can link to any file anywhere on the web.

A link can also send a user to another part of the same page they are on. You can add an id attribute to just about any tag, for example `<h2 id="moss">Moss</h2>`, and then link to it by using something like this: `<a href="#moss">Go to moss</a>`. Selecting this link will scroll the page straight to the element with that ID.

Images

Things might seem a little bland and boring with all of this text formatting. Of course, the web is not just about text, it is a multi-media extravaganza and the most common form of sparkle is the image. The img tag is used to put an image in an HTML document and it looks like this:

```
<img src="http://www.shawacademy.com/badge1.gif" width="120"
height="90" alt="HTML tag">
```

The attribute tells the browser where to find the image. Like the tag, this can be absolute, as the above example demonstrates, but is usually relative. For example, if you create your own image and save it as "alienpie.jpg" in a directory called "images" then the code would be <img src="images/alienpie.jpg"...

The width and height attributes are necessary because if they are excluded, the browser will tend to calculate the size as the image loads, instead of when the page loads, which means that the layout of the document may jump around while the page is loading.
The alt attribute is the alternative description. This is an accessibility consideration, providing meaningful information for users who are unable to see the image (if they are visually impaired or have disabled images in their browser, for example).

Note that, like the br tag, because the img element does not enclose any content, no closing tag is required.

The construction of images for the web is a little outside of the remit of this website, but it is worth noting a few things...

The most commonly used file formats used for images are JPEGs, GIFs, and PNGs. They are compressed formats, and have very different uses.

A JPEG (pronounced "jay-peg") uses a mathematical algorithm to compress the image and will distort the original slightly. The lower the compression, the higher the file size, but the clearer the image.

JPEGs are typically used for images such as photographs.

A GIF (pronounced "jif") can have no more than 256 colors, but they maintain the colors of the original image. The lower the number of colors you have in the image, the lower the file size will be. GIFs also allow any pixel in the image to be transparent.

GIFs are typically used for images with solid colors, such as icons or logos.

A PNG (pronounced "ping") replicates colors, much like a GIF, but allows 16 million colors as well as alpha transparency (that is, an area could be 50% transparent).

PNGs are typically used for versatile images in more complex designs BUT they are not fully supported by some older browsers.

The web is forever getting faster and faster but you obviously want your web pages to download as quickly as possible. Using super-high resolution images isn't doing your or your user's bandwidth (or patience) any favors. Image compression is a great tool and you need to strike a balance between image quality and image size. Most modern image manipulation programs allow you to compress images and the best way to figure out what is best suited for yourself is trial and error.

SHAW ACADEMY